



DEWEsoft®

measurement innovation

DEWESoft7 DCOM Manual

Version: 1.1.0

General information

Printing notice

The information contained in this document is subject to change without notice.

Dewesoft GmbH (Dewesoft) shall not be liable for any errors contained in this document.
DEWESOFT MAKES NO WARRANTIES OF ANY KIND WITH REGARD TO THIS DOCUMENT, WHETHER EXPRESS OR IMPLIED. DEWESOFT SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Dewesoft shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory, in connection with the furnishing of this document or the use of the information in this document.

Copyright notice

© 2015 Dewesoft GmbH

All rights reserved. May not be duplicated or disseminated in any fashion without the express written permission of Dewesoft GmbH.

Use Austrian law for duplication or disclosure.

This document contains information which is protected by copyright. All rights are reserved.

Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Dewesoft GmbH
Grazerstrasse 7
A-8062 Kumberg
Austria / Europe
<http://www.dewesoft.com>

Trademark notice

All trademarks are acknowledged to be the property of their owners, with all rights and privileges thereto. No infringement is intended.

Disclaimer

Dewesoft makes no claim about the efficacy or accuracy of the information contained herein. Use of this manual is entirely at the user's own risk. Under no circumstances will Dewesoft assume any liability caused by the use, proper or improper, of this manual or the information, textual, graphical or otherwise, contained within it.

Table of Contents

DCOM Manual	1
Guide	2
Automation	2
DCOM Server Registration	2
Add-Ons for DEWESoft	5
Plug-Ins	6
Prerequisites for DEWESoft Custom Plug-ins	7
Custom Visual Controls	8
Custom Mathematics	8
Custom Export	9
Custom Export Call Diagram	10
Custom Import	11
General	11
Legend	11
The Buffer Structure	12
Sample Rates	15
Channels	17
Channel Index	18
Channel Index Example	21
Synchronism	23
Numeric Channels	24
Textual Channels	24
Array Channels	25
Control Channels	26
Scaling	27
Data Types	29
DateTime	29
TDateTime	30
HRESULT	30
TColor	30
ColourCodes	30
Mathematics	31
XML Setup	32

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
GUI Navigation													
Measure Mode 34													
Analysis Mode 35													
Terms													
Calculation Delay 36													
GUID 41													
nil (aka. NULL, nothing) 41													
Null 42													
UUID 42													
Variant 42													
Data Files													
Start Events													
What's new in DEWESoft 7													
Plugin Examples													
Finding Plugins													
Delphi													
Delphi: Prepare Project 48													
Delphi: Implementation 57													
Delphi: Prepare for DCOM 65													
Delphi: Register Plugin 68													
Delphi: Test the Plugin 69													
Delphi: Sourcecode 72													
Visual Basic													
Visual Basic: Prepare Project 76													
Visual Basic: Implementation 81													
Visual Basic: Prepare for DCOM 84													
Visual Basic: Register Assembly 87													
Visual Basic: Signing the assembly 89													
Visual Basic: Test the Plugin 91													
Visual Basic: Troubleshooting 93													
Visual Basic: Sourcecode 94													
Visual C++													
Visual C++: Prepare Project 97													
Visual C++: Implementation 109													
Visual C++: Prepare for DCOM 114													
Visual C++: Register Plugin 116													
Visual C++: Test the Plugin 118													
Visual C++: Troubleshooting 120													
Visual C++: Sourcecode 125													

How to	131
How To Mount Dewesoft Channels	131
How To Write Data To Channels	135
How To Find Channels	137
How To Read Data From Channels	138
Reference	140
Interfaces	140
IAISetupScreen	140
SetColumnVisible	141
Show ChannelSetup	142
IAOChannel	143
Ampl	144
FilterFreq1	144
FilterFreq2	145
FilterOrder	145
FilterProtoType	146
FilterType	146
Offset	147
Phase	148
Range	148
WaveForm	149
IAOGroup	149
AOChannels	150
AmplChangeFactor	150
ControlsClock	152
DCChangeFactor	152
DeltaFreq	154
Freq	154
FreqChangeFactor	155
LogSw eep	156
OperationMode	157
PhaseChangeFactor	157
SampleRate	159
Show InfoChannels	159
StartFreq	160
StartTime	161
StopFreq	161
StopTime	162

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
SweepMode													
IAlarmCond													162
Avail													164
CustomName													164
EndAlarm													165
Index													165
Name													165
Status													165
StopOption													166
StopTime													166
StopTrigger													167
Trigger													168
IAlarms													169
ActiveCount													170
ActiveItem													170
Add													170
Count													170
Item													171
Remove													171
IAmplChain													171
Count													171
IOControl													172
Item													172
IAmplChainList													172
Count													172
Item													173
IAmplInterface													173
ChainList													173
IOControl													173
SubInterface													174
IAmplInterfaces													174
MainInterface													176
IAmplifier													177
IOControl													177
IApp													177
AISetupScreen													177
AOGetManualAvail													178
AOGroup													178
AOSetManual													179

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Acquiring													179
ActiveScreen													180
ActualRunMode													180
Alarms													181
AlwaysEnableTrigger													181
AmplInterfaces													182
Analyze													182
AveragedCPB													182
AveragedFFT													182
CAN													183
CalcScopeTrig													183
ChangeComPort													183
ChangeDaqType													184
ConfigMode													185
Daq													185
DaqGroup													185
Data													185
DataLost													185
DisableKeyboardShortcuts													186
DisableStoring													186
Enabled													186
EventList													186
ExecuteModulesFunction													187
ExportData													189
ExportDataEx													189
FileNameSettings													191
FirstScanDonePercent													191
FixedExternalClock													191
GetInterfaceVersion													191
GetDewesoftVersion													192
GetSpecDir													193
GlobalHeader													193
GoToInstruments													194
HardwareSetup													194
HasFRF													195
Height													195
HideCaptionBar													195
IniFileDir													195
Init													196
InitScopeTrig													196

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
IsAcqRunning													196
IsSetupMode													197
LastKey													197
Left													197
LoadDBC													197
LoadDisplaySetup													198
LoadEngine													199
LoadFile													199
LoadModuleSetup													200
LoadProject													200
LoadSequence													200
LoadSetup													201
LoadSetupFromXML													201
MainDataDir													202
MainWindow Handle													202
MainWndMessage													202
ManualStart													202
ManualStop													203
MasterClock													204
Math													205
Measure													205
MeasureSampleRate													205
MeasureSampleRateEx													205
MenuClick													206
Modules													206
NETMode													207
New Setup													208
NotifyTrackingChanged													208
OfflineCalc													208
Parent													208
PauseStoring													209
PowerModules													209
PrintScreen													209
ProjectManager													210
ReducedRate													210
RegType													210
RemoteControlled													211
ResumeStoring													211
SaveSetup													212
SaveSetupToXML													212

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Screens	213											
SendCommand	214											
SendKey	214											
SetFullScreen	214											
SetHeaderData	215											
SetInstrument	215											
SetMainDataDir	216											
SetMainToolBar	216											
SetRemoteMode	217											
SetScopeParams	217											
SetScopeUsed	219											
SetScreenIndex	219											
SetStoreMode	220											
SetupSampleRate	221											
SetupScreen	221											
Show CaptionBar	222											
Show InstrumentsInFullScreen	222											
Show PropertyFrame	223											
Show SROptions	224											
Show SensorEditor	224											
Show StoreOptions	225											
Show Style	226											
Start	228											
StartModuleScan	228											
StartStoring	228											
StayOnTop	229											
Stop	229											
StopModuleScan	229											
StoreEngine	230											
SuppressMessages	230											
TimerInterval	231											
Timing	232											
Top	232											
Trigger	233											
UpdateHardw areSetup	233											
UpdateSetupScreen	233											
UsedDatafile	233											
UsedSetupfile	234											
UserInterface	234											
Version	234											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Video	234											
Visible	235											
Width	235											
WriteErrorLog	235											
WriteErrorMessage	235											
ZeroAllAutoChannels	236											
IArrayInfo	237											
AxisDef	237											
ColorArr	237											
DimCount	237											
DimSizes	237											
Init	238											
ItemChannels	238											
NameArr	238											
SyncSource	238											
IAveragedFFT	238											
AveCount	239											
AverageType	239											
CalculateFromPos	239											
GetCPBData	239											
GetCPBXData	240											
GetChannels	240											
GetData	240											
GetFFTData	241											
Lines	241											
Overlap	241											
Window	242											
IAxisDef	242											
AxisType	243											
CursorChannel	243											
FloatValues	243											
Name	243											
Precision	243											
Size	244											
StartValue	244											
StepValue	244											
StringValues	244											
_Unit	244											
ICAN	245											
Count	245											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Item	246											
SupportsOutput	246											
ICANContext	246											
GetClock	246											
GetClockOffset	246											
PortCount	247											
Ports	247											
ICANMsg	247											
AddData	247											
ICANPort	248											
Capture	248											
EnableOutput	249											
EndRead	249											
GetBaudRate	249											
GetBaudRateList	250											
MessageCount	251											
ReadMessage	251											
SendFrame	252											
SetBaudRate	253											
StartRead	253											
TotalErrMsgCount	253											
TotalMsgCount	253											
ICANPortContext	254											
BaudRate	254											
Captured	254											
GetMsg	254											
ListenOnly	255											
SetErrMsgCount	255											
SetTotalMsgCount	255											
Termination	255											
Used	256											
ICNTGroup	256											
Count	256											
Item	256											
ICamera	257											
FrameBufSize	257											
FrameDataSize	257											
FrameList	258											
FramePos	258											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
FrameSizeInBytes	258											
GetBitmapInfoHeader	258											
Name	259											
Used	259											
IChannel	259											
AbsMax	259											
AbsMin	259											
AddAsyncByteSample	260											
AddAsyncData	260											
AddAsyncDoubleSample	260											
AddAsyncInt64Sample	261											
AddAsyncIntegerSample	261											
AddAsyncShortintSample	261											
AddAsyncSingleSample	262											
AddAsyncSmallintSample	262											
AddAsyncString	262											
AddByteSample	263											
AddData	263											
AddDoubleSample	263											
AddIn64Sample	264											
AddIntegerSample	264											
AddIntegerSampleWithCalc	264											
AddShortintSample	264											
AddShortintSampleWithCalc	265											
AddSingleSample	265											
AddSingleSamples	265											
AddSmallintSample	266											
AddSmallintSampleWithCalc	266											
AddWordSample	266											
ArrayChannel	267											
ArrayInfo	267											
ArraySize	267											
Async	267											
BitCount	268											
Bytes	268											
CalcDelay	268											
CalcSRDiv	269											
ChNo	269											
ChangeThreshold	269											
ControlChannelFlags	270											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
ControlChannelState													270
CreateConnection													271
DBBufSize													271
DBDataSize													271
DBPos													271
DBTimeStamp													272
DBValues													272
DBValuesDouble													272
DStartDataAvail													273
DStopDataAvail													273
DStopDataAvailDir													273
DataType													274
Description													274
DiscreteList													275
ExpectedAsyncRate													276
ExportOrder													276
Exported													278
FastCalc													278
FastCalcInt32													279
FirstIBLevel													279
FirstX													279
GetChannelSetup													279
GetDBAddress													280
GetIBValues													281
GetIndex1													281
GetOfflineStatus													282
GetRBValues													282
GetScaledData													283
GetScaledDataEx													283
GetScaledDataEx1													284
GetTSAAddress													284
GetTSData													285
GetTSDDataEx													285
GetTSDDataEx1													286
GetUnscaledData													286
GetUnscaledDataEx													286
GetUnscaledDataEx1													287
GetValueAtAbsPos													287
GetValueAtAbsPosDouble													288
Group													288

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
IBBufSize	289											
IBDataSize	289											
IBDataSizeEx	289											
IBPos	289											
IBPosEx	290											
IBValues	290											
IBValuesEx	290											
IncDBSamples	291											
Index	291											
IndexEx	291											
IsControlChannel	292											
IsSingleValue	292											
LogicalIndex	292											
LogicalName	292											
MType	292											
MainDisplayColor	293											
Measurement	294											
Name	294											
Offset	295											
RBBufSize	295											
RBDataSize	295											
RBPos	295											
RBValues	296											
SRDiv	297											
SRDivType	298											
Scale	298											
ScaleValue	299											
ScaleValueDouble	299											
Scale_	299											
SecondX	299											
SelectorIndex	300											
SelectorIndexLevel	300											
SelectorIndexStartLevel	300											
SetAsStringChannel	300											
SetAsync	301											
SetChannelSetup	301											
SetDataType	301											
SetFreezeMode	301											
SetIsSingleValue	302											
SetSRDiv	302											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
SetSRDivType													303
Settings													303
Show ChannelSetup													303
Show n													304
SingleValue													304
Stored													304
Tag													305
Text													305
Typical.MaxValue													306
Typical.MinValue													306
Unit_													306
UpdateXML													307
Used													308
UserScaleMax													309
UserScaleMin													310
IChannelConnection													311
AType													311
BlockSize													313
Channel													313
GetDataBlocks													313
GetDataBlocks1													313
GetDataValues													314
GetDataValues1													314
GetTSBlocks													314
GetTSBlocks1													315
GetTSValues													315
GetTSValues1													316
NumBlocks													316
NumValues													316
Overlap													316
Reset													317
Start													317
IChannelGroup													317
ExportRate													317
GetIndexName													317
Name													318
IChannelGroup2													318
GetIndexNameShort													318
IChannelGroups													318
Count													319

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Item	319											
IChannelList	319											
Count	319											
Item	319											
IChannelListEx	320											
AddCh	320											
Clear	320											
SetCh	320											
ICntChannel	320											
AdvCntMode	321											
BaseMode	321											
CanAutoCalculate	321											
CardChannel0	321											
CardChannel1	321											
CntAux	322											
CntAuxInv	322											
CntDoManualReset	322											
CntEncoderMode	322											
CntEncoderZero	323											
CntEventWithZero	323											
CntFilter	323											
CntGate	323											
CntGateInv	323											
CntMode	324											
CntNew ValueUpdateMode	324											
CntPair	324											
CntResetOnStartMeasure	324											
CntSignalZero	325											
CntSource	325											
CntSourceInv	325											
CntUpDow nMode	325											
DIChannels	325											
TrigLevels	326											
TrigLevelsCombined	326											
ICustomDAQ	326											
CheckSampleRate	326											
GetBitResolution	327											
GetBufferSize	327											
GetCNTBitResolution	328											
GetCardName	328											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
GetChannelGain													328
GetCurrentTime													329
GetDWTypeLibVersion													329
GetData													329
GetDeviceCode													330
GetDeviceType													330
GetMinMax													330
GetOptionName													331
GetOptionsCount													331
GetSampleRates													332
Get_CardFound													332
Get_NumCNTChannels													332
Get_NumChannels													333
HideSetupFrame													333
SetApp													333
Show SetupFrame													334
StartAcq													334
StopAcq													334
ICustomDAQ2													335
OnMessage													335
ICustomExport													335
EndChannel													335
EndDataFolder													336
EndExport													336
EndHeader													336
EndInfo													337
EndValue													337
Get_AbsoluteTime													337
Get_DataCount													338
Get_ExportType													338
Get_Extension													338
Get_FileName													339
Get_SupportsAsync													339
Get_SupportsSRDiv													339
Get_TimeIncrease													340
Set_AbsoluteTime													340
Set_DataCount													341
Set_FileName													341
Set_TimeIncrease													341
StartAbsValue													341

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
StartChannel	342											
StartDataField	342											
StartDataFolder	343											
StartExport	343											
StartInfo	343											
StartTimeField	344											
StartValue	344											
WriteAsyncValue	344											
WriteInfoDate	345											
WriteInfoInteger	345											
WriteInfoSingle	345											
WriteInfoString	346											
WriteValue	346											
ICustomExport2	346											
GetDWTypeLibVersion	347											
Get_SupportsDouble	347											
SetAbsMax	347											
SetAbsMin	348											
SetApp	348											
SetChannel	348											
SetChannelColor	349											
SetDoubleFloat	349											
SetRangeMax	349											
SetRangeMin	350											
SetTrigOffset	350											
StartEvents	350											
StopEvents	351											
WriteAsyncDoubleValue	351											
WriteDoubleValue	351											
WriteEvent	352											
ICustomExport3	352											
OnEvent	352											
IDIChannel	353											
DlFilter	354											
DlInvert	354											
TrigLevels	354											
IDIGroup	355											
Count	355											
Item	356											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
IDIPort	356											
ApplyDBBuf	356											
IDAq	356											
CanAutoCalculate	356											
CardCount	356											
DaqType	357											
DataLost	357											
GetCNTTrgLevel	357											
GetDITrgLevel	358											
GetDeviceCode	358											
GetDeviceInfo	359											
IOControl	359											
SetCNTTrgLevel	359											
SetDITrgLevel	360											
SetDeviceCalDate	360											
IDAqChannel	361											
AutoZero	361											
CardBitResolution	361											
CardGain	361											
CardOffset	361											
CustomSensorOffset	362											
CustomSensorScale	362											
GetBoardOpt	362											
GetSensor	362											
GetSensorType	363											
ModuleGain	363											
ModuleOffset	363											
ModuleType	363											
SetBoardOpt	363											
SetCardGain	364											
SetSensor	364											
IDAqData	364											
Address	364											
CopyToString	365											
CopyUnitToString	365											
CurrentSource	365											
DaqNNames	365											
DaqNNamesCount	366											
FREQAInputCoupling	366											
FREQAOutputFilter	366											

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
FREQATriggerLevel	366											
FilterCode	367											
Filters	367											
FiltersCount	367											
HighpassType	367											
ICPInput	368											
ModuleAmpl	368											
ModuleError	368											
ModuleOffset	368											
ModuleType	369											
Name	369											
Overflow	369											
RangeCode	370											
Ranges	370											
RangesCount	370											
Remote	370											
ShortCopyToString	371											
ThermLinearize	371											
VRange	371											
IDaqGroup	371											
Count	372											
Item	372											
IData	372											
ActiveChannels	372											
AllChannels	372											
AnalyseMode	373											
ApplyChannels	373											
BuildChannelList	373											
CurrentPos	374											
CurrentPosD	374											
EndDataSync	375											
EndStamp	375											
EndStampD	376											
ExternalClock	377											
ExternalTrigger	377											
FindChannel	378											
FindChannelByIndex	378											
FindChannelByIndex1	378											
FindChannelByIndexEx	379											
FirstTimeStamp	379											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
FreezeMode	380											
GetIndexName	381											
GetIndexName1	381											
GetIndexNameShort	381											
GetIndexNameShort1	382											
GetSamplesAcquired	382											
Groups	383											
IBAbsMidRate	384											
IBAbsRate	384											
IBLevels	384											
IBRate	384											
InputGroups	384											
MRealTimeStamp	385											
MaxCalcDelay	385											
MeasureMode	385											
SRDivLCM	386											
SampleRate	386											
SampleRateEx	386											
Samples	386											
SetExternalClock	387											
SetStartStoreTimeUTC	387											
Show nChannels	387											
StartDataSync	387											
StartStamp	388											
StartStampD	388											
StartStoreTime	389											
StartStoreTimeUTC	389											
UsedChannels	390											
IDataSection	390											
DataCount	390											
ReadData	391											
ReadData1	391											
Time	391											
TrigPos	391											
IDataSections	392											
Count	392											
Item	392											
IDewePlugin	392											
OnMessage	392											
IDigitalTrigLevel	393											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Coupling	393											
ReTrigLevel	393											
TrigLevel	393											
TrigType	394											
IDiscreteItem	394											
Caption	395											
Color	396											
Value	397											
IDiscreteList	398											
Add	398											
Count	399											
Find	399											
Item	399											
Remove	399											
IDisplayFrameTemplate	400											
AddChannel	400											
AddItemChannel	400											
CreateCustomGroupAndControl	400											
CreateGroupAndControl	401											
GroupName	401											
SetupDOMDoc	402											
TemplateName	402											
IDisplayFrameTemplates	402											
Add	402											
Clear	402											
Count	403											
Item	403											
IDisplayTemplate	403											
DH	403											
DW	403											
DisplayFrameTemplates	404											
IEvent	404											
Data	404											
PosDir	405											
PosMid	405											
TimeStamp	405											
TrigInfo	405											
Type_	405											
IEventList	406											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Count	407											
Item	407											
IExportFrame	408											
Apply	408											
HideFrame	409											
SetExpApp	409											
Show Frame	409											
IFileNameSettings	409											
AutoCreate	410											
AutoFlipAbsTime	410											
AutoFlipFile	411											
AutoFlipSize	411											
AutoFlipUnit	412											
BaseFileName	413											
MultiFileStartIndex	413											
UseDate	415											
UseMultiFile	415											
UseTime	417											
IGHObject	417											
Caption	419											
Data	420											
ObjType	421											
IGlobalHeader	424											
Count	425											
Item	426											
IImportChannel	426											
IImportGroup	427											
MountChannel	427											
IIndexChanger	427											
ChangePluginChIndex	427											
ChangePluginChIndex1	428											
IInputGroup	428											
Guid	429											
Index	429											
Name	429											
Properties	429											
IInputGroups	429											
Count	430											
Item	430											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
ILoadEngine													
CloseFile	430											
DataSections	430											
FileOpened	431											
GetVideoCompressDone	431											
IsVideoCompressDone	431											
NextDBLoad	431											
NumBlocks	432											
ReducedOnly	432											
Reload	432											
ReloadBlock	432											
ReloadEx	433											
ShrinkFile	433											
StartDBLoad	433											
StartVideoCompress	434											
StopVideoCompress	434											
VideoLoadEngines	434											
IMasterClock													
GetCurrentTime	435											
IMath													
AddObj	435											
Count	436											
FindObjByID	436											
MathObject	436											
RemoveObj	436											
IMathChannel													
DefaultMax	437											
DefaultMin	437											
DefaultRes	437											
IMathContext													
InputChannels	438											
MountChannel	438											
MountChannelEx	438											
MountInputGroup	439											
OutputChannels	439											
IMathFrameContext													
Apply	440											
IMathItem													
InputChannels	441											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
OutputChannels													
IMathModule													
Id													
MathObject													
IMathObjContext													
IMathObject													
Count													
FindModuleByID													
Id													
MathGUID													
MathModule													
MathObjContext													
MathType													
Name													
IModule													
ClearModule													
DaqData													
DetectModule													
FREQAFindTriggerLevel													
FillModule													
GetDataPad													
GetSerialNumber													
Index													
ModuleType													
PadData													
SetDaq													
SetDaqAddress													
SetModule													
SetPad													
IModules													
Count													
Item													
INothing													
IOfflineCalc													
Calculate													
StoreCalculatedChannels													
IPadData													
Address													
ConfigCode													

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
CopyToString	450											
CopyUnitToString	450											
Data	451											
ModuleAmpl	451											
ModuleOffset	451											
ModuleType	452											
Name	452											
RangeCode	452											
RangeIndex	453											
Ranges	453											
RangesCount	453											
ShortCopyToString	453											
SpeedCode	454											
IPlugin	454											
Configure	454											
Initiate	454											
OnGetData	455											
OnStartAcq	455											
OnStartStoring	455											
OnStopAcq	455											
OnStopStoring	456											
OnTrigger	456											
SetApp	456											
SetData	456											
IPlugin2	457											
ClearChannelsInstance	457											
Configure	457											
HideFrame	457											
Id	458											
Initiate	458											
LoadSetup	458											
New Setup	459											
OnGetData	459											
OnOleMsg	459											
OnStartAcq	459											
OnStartStoring	460											
OnStopAcq	460											
OnStopStoring	460											
OnTrigger	461											
SaveSetup	461											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Show Frame	461											
UpdateFrame	462											
Used	462											
IPlugin3	462											
GetDWTypeLibVersion	462											
OnAfterCalcMath	463											
OnAfterStartAcq	463											
OnAfterStopAcq	463											
OnAlarm	463											
OnBeforeStartAcq	464											
OnBeforeStopAcq	464											
OnBigListLoad	464											
OnExit	465											
OnGetClock	465											
OnGetSetupData	465											
OnHideHWFrame	465											
OnRepaintFrame	466											
OnResizeFrame	466											
OnShow HWFrame	466											
OnStartSetup	467											
OnStopSetup	467											
OnTriggerStop	467											
ProvidesClock	467											
SetCANPort	468											
IPlugin4	468											
OnEvent	468											
IPluginChannel	474											
AlwaysReserveMemoryInSetup	474											
AsyncBufSize	474											
DefaultMax	474											
DefaultMin	475											
DefaultRes	475											
FreeMemory	475											
LongName	475											
MarkAsOffline	475											
PluginGUID	476											
ReserveMemory	476											
SetChNo	476											
SetIndex	477											
SetSettings	477											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
IPluginChannelXMLHelper 478													
ExtractNextChannel 478													
FindNode 479													
MountAllChannels 479													
StartExtractChannels 479													
IPluginGroup 480													
AddIndexName 480													
AddIndexNameEx 480													
ClearAllChannels 481													
FindChannel 481													
FindInputGroup 481													
MountChannel 482													
MountChannelEx 482													
MountInputGroup 483													
UnmountChannel 483													
IPluginLicense 484													
GetHardw areCode 484													
GetRegTypeWanted 484													
GetTrustedCode 484													
SetLicenseCode 485													
IPluginLicense2 485													
GetLicenseCode 485													
IPowerModule 485													
FFTBlockSize 485													
FFTSampleRate 486													
GetFFT 486													
GetVectorScopeData 486													
LoadFromXML 486													
LoadFromXML1 487													
ModuleIndex 487													
SaveToXML 487													
SaveToXML1 487													
IPowerModules 488													
Add 488													
Count 488													
Item 488													
Remove 489													
IProjectManager 489													
ChangeProject 489													

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
GetCurrentProject													490
GetProjects													490
IProperties													490
Add													490
Count													491
Item													491
IRegistrationHelper													491
CheckRegistration													491
IScreen													491
GetCursor													492
Id													492
IsCurrent													493
Name													493
SetCursor													493
Show													494
ZoomIn													494
ZoomOut													494
IScreens													494
Count													495
Current													495
Item													495
ISetupMessages													496
Add													496
IStoreEngine													497
AddNew Event													497
Allow IBSkipping													498
FileName													498
FileSize													498
IsTriggering													498
Paused													499
StartStoreTimeChanged													499
StoreMode													499
Storing													500
TrackingOffset													500
ISyncSource													501
IsSyncSource													501
SampleRate													501
ITiming													501
Tracking													501

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
ITrig 501													
GetTrigIndexEx 502													
NotOrList 502													
OrList 503													
TrigIndex 505													
ITrigInfo 505													
Channel 505													
DeltaTime 505													
Direction 505													
Direction1 506													
Level1 506													
Level2 506													
Manual 506													
Mode 506													
TrigValue 507													
ITrigger 507													
HoldoffTime 507													
HoldoffTimeUsed 508													
PostTime 508													
PostTimeExtensionUsed 508													
PostTimeUsed 508													
PreTime 509													
PreTimeUsed 509													
StartTrig 509													
StopTrig 509													
ITriggerCondList 510													
Add 510													
Count 510													
Item 510													
Remove 510													
ITriggerCondition 511													
AddChannel 512													
Channels 512													
ClearChannels 512													
DeleteChannel 512													
DeltaTime 513													
Direction 513													
Direction1 514													
Level1 515													

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
Level2													
Mode	516											
TimeCond	517											
TimeFormat	518											
TimeUnit	518											
TimeValue	519											
TrigType	520											
TrigValue	520											
IUserInterface													
ChangeSetupScreen	521											
Show TrigCondSetup	522											
IVCContext													
BroadcastPosChanged	522											
BroadcastScaleChanged	523											
DChannels	523											
DInputGroups	523											
DataRegionChanged	523											
Repaint	524											
IVideo													
CameraCount	524											
Cameras	525											
IVideoFrame													
BufSize	525											
GetData	525											
GetTS	525											
IVideoLoadEngine													
GetFramesInfo	526											
IVideoLoadEngines													
Count	526											
Item	526											
Enumerations													
AOOperationMode	527											
AOSweepMode	527											
AOWaveForm	528											
ConnTypes	528											
ControlChFlags	528											
CustomCANMessages	529											
CustomDAQMessages	530											

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
CustomExpEventIDs	530											
CustomImpMessages	531											
CustomMathFrameMessages	532											
CustomMathMessages	533											
CustomVCMessages	537											
EventIDs	538											
EventReason	538											
EventType	538											
ExportTypes	539											
GHOObjectType	539											
ImportStatus	540											
ImportTypes	540											
MathMultipassType	540											
MenuItems	541											
ModulesFunction	541											
SetupMessageType	541											
SpecDirectory	542											
TAxisType	542											
TSRDivType	543											
XMLType	543											
Types	544											
DaqDeviceInfo	544											
ITestRecord	544											
T_CANFrame	545											
T_ChIndex	545											
T_RecordPosition	546											
T_ReducedRec	546											
Constants	546											
IOCodes	546											
IOCodes for IAmplInterface	547											
IO Codes for Chain properties	547											
IO Codes for Amplifier properties	548											
IO Codes for Amplifier commands	552											
Document History	553											

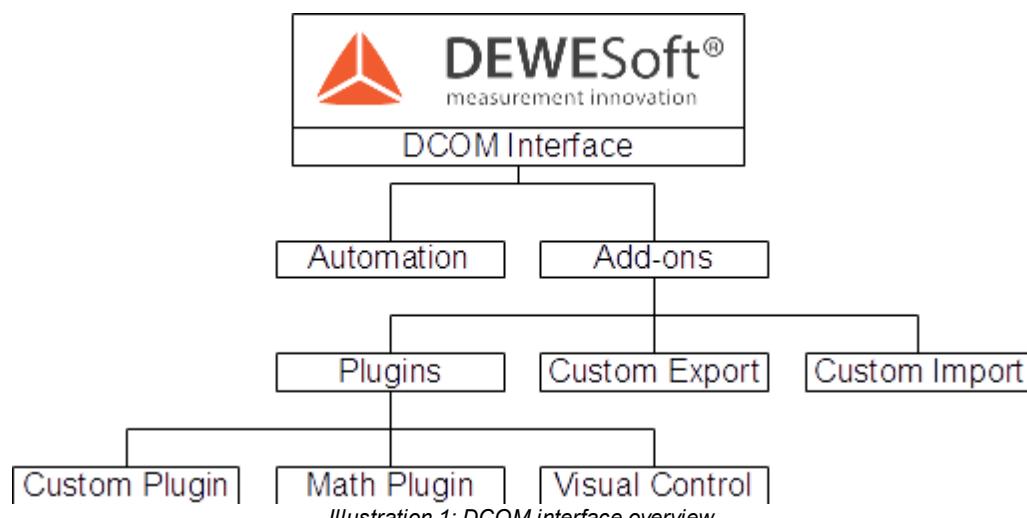
DCOM Manual

This manual should help developers to use DEWESoft's DCOM Interface. The interface as documented in this paper is available in *DEWESoft®* 7 or newer.

The manual consists of 2 major parts:

- [Guide](#): includes comprehensive explanation and examples about DCOM and *DEWESoft®*
- [Reference](#): includes an extensive reference describing all the interfaces, methods, enumerations types and constants of *DEWESoft®*'s DCOM interface

As illustrated in the figure below, the DCOM Interface of DEWESoft 7 and newer offers the possibility to expand the possibilities of *DEWESoft®* with *Add-ons* on the one hand and to implement other applications controlling *DEWESoft®*, which is called *automation*, on the other hand.



[Automation Applications](#) for *DEWESoft®* can be developed with any programming language that supports DCOM: e.g. Borland Delphi, MS Visual Basic, MS Visual C++/C# or NI LabVIEW (automation only).

Add-ons can be developed with any programming language that supports DCOM and can build a DLL file which will then be loaded by *DEWESoft®* at startup.

see also:

- <http://www.microsoft.com/com/default.mspx>
- [http://msdn.microsoft.com/library/ms680573\(VS.85\).aspx](http://msdn.microsoft.com/library/ms680573(VS.85).aspx)

1 Guide

The DCOM guide includes comprehensive explanation and examples about DCOM and *DEWEsoft®*.

The reader should have a basic understanding of DCOM and a sound understanding of his programming language and IDE.

1.1 Automation

The DEWESoft® application can be used as DCOM server: so that you can write a custom control application (in any programming language that supports DCOM) to take control of DEWESoft®.

Once the control application has connected to DEWEsoft®, it can start/stop recording, show *DEWEsoft®* setups, access *DEWEsoft®* data and much more.

Some programming languages that support DCOM: .NET, C#, Visual Basic, Automation: Matlab, Delphi, C++, ...

Other ways to get access to *DEWESoft®* data:

- you may read the *DEWESoft®* data files directly: see [Data Files](#)
 - you may use the DEWE-NET option protocol (it's a *DEWESoft®* specific TCP/IP based communication format)
 - you may use Active-X technology: we provide a *DEWEX.OCX* file

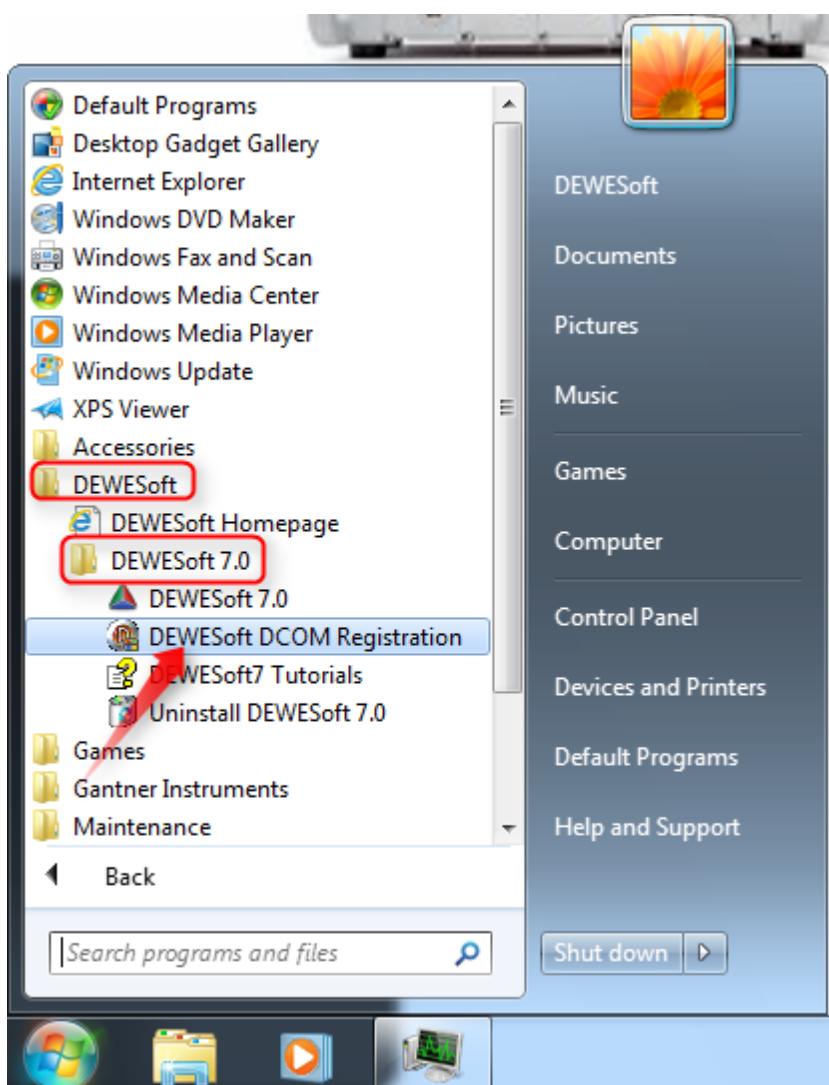
see also: [DCOM Server Registration](#)

1.1.1 DCOM Server Registration

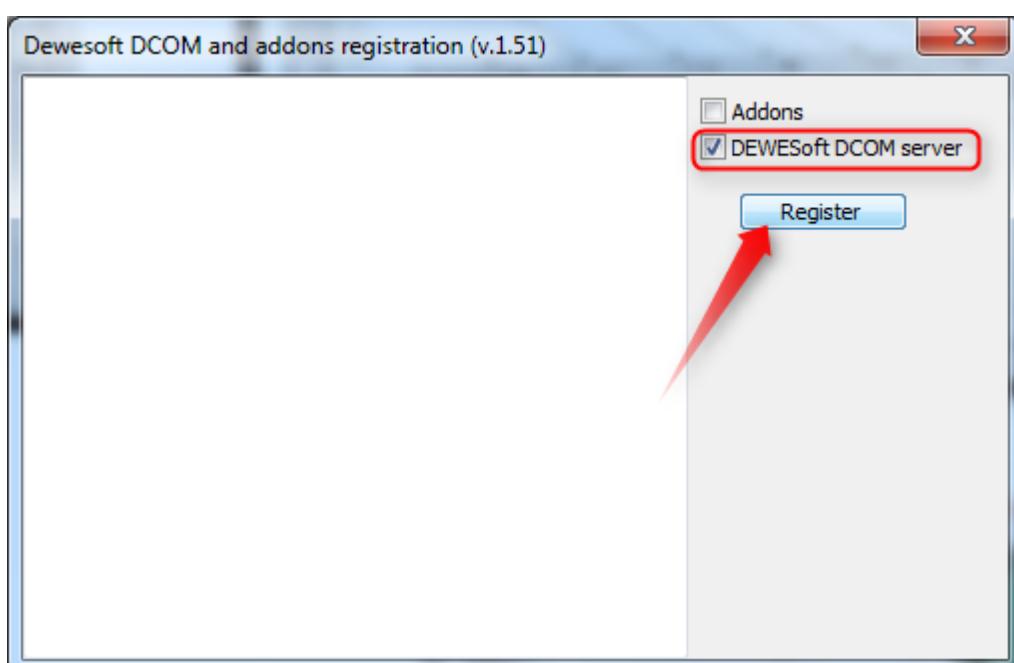
When you start DEWEsoft® it will automatically register its application object (App Object) as DCOM server.

Note: on Windows 7 you may need administrator rights if UAC (User Account Control) is activated.

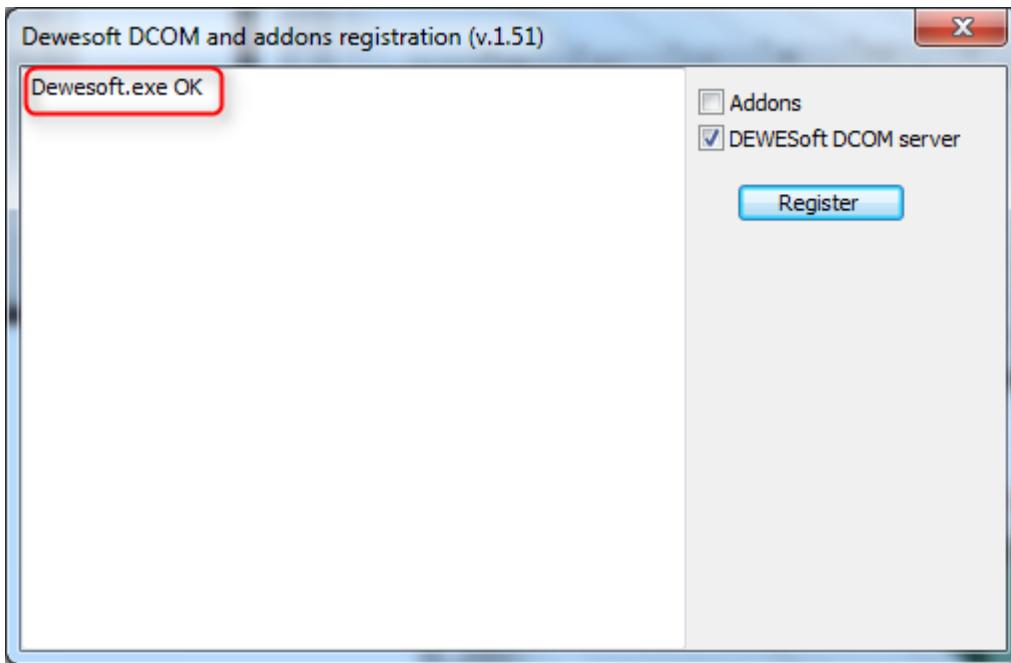
You can also register the DCOM server manually with the tool [DEWESoft DCOM Registration](#) tool;



When the program has started, activate the *DEWESoft DCOM server* checkbox and click the *Register* button.



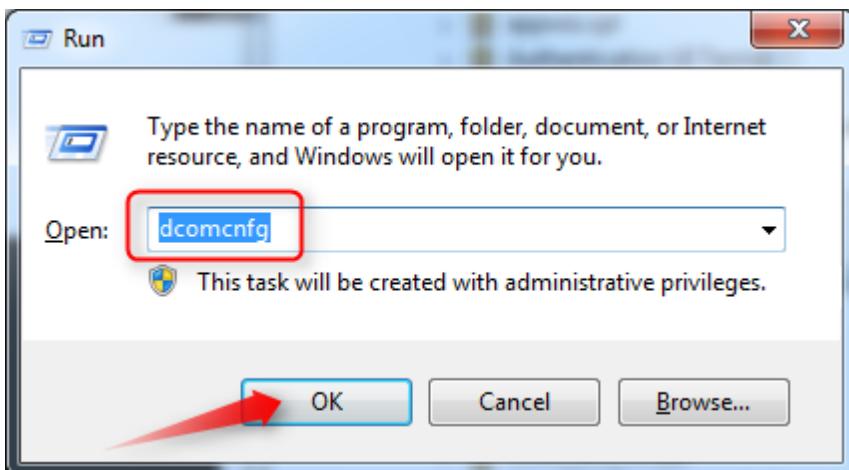
When the registration is finished, you should see the message '*Dewesoft.exe OK*'.



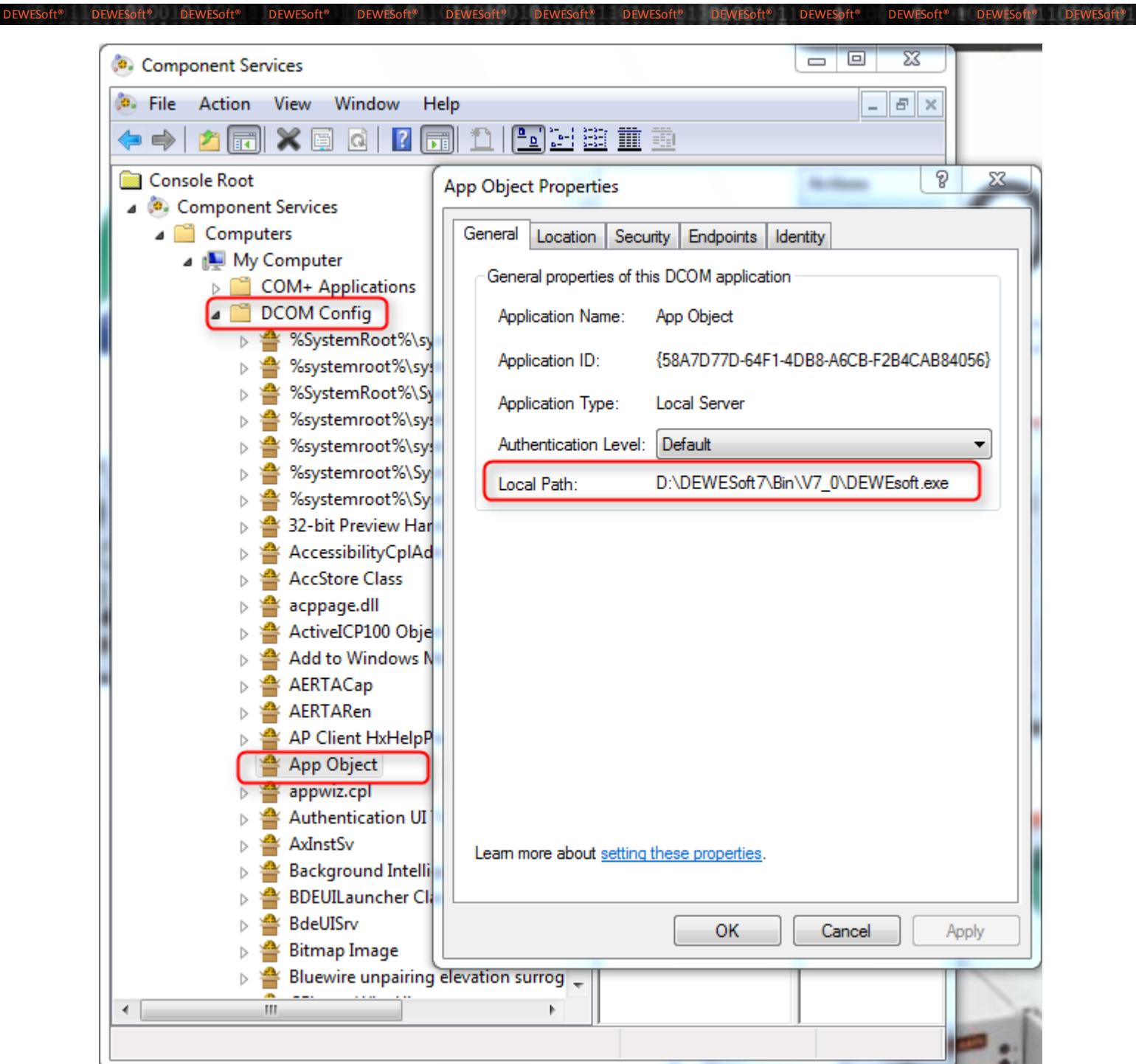
When the registration has succeeded you can see the DEWESoft® application object (*App Object*) in the Windows DCOM configuration tool.

To start the Windows DCOM configuration tool just execute the command `dcomcnfg.exe` (e.g. hold down the Windows key and press `R`, then enter `dcomcnfg` and press `OK`):

NOTE: on 64-bit systems you must start the 32 bit version with this command: `mmc comexp.msc /32`
see <http://goo.gl/sI3h9j> for details



In the Windows DCOM configuration tool you can navigate to *DCOM Config* and there you should find *App Object*:



1.2 Add-Ons for DEWESoft

Gives an overview of the various Add-On types that you can use to expand the power of **DEWESoft®**.

1.2.1 Plug-Ins

On the one hand, the motivation for creating plug-ins for *DEWESoft®* can be to gain access to acquired data during measurement e.g. in order to make some special calculations on it or to output the data via RS-232, etc. On the other hand, the task of a plug-in could be to add further data channels to *DEWESoft®*. This could be data of a special sensor (e.g. connected via RS-232) or data which is derived from special calculations on data currently measured by *DEWESoft®*. In both ways, coming from *DEWESoft®* or being added to *DEWESoft®*, data can be either synchronous or asynchronous.

When a plug-in is registered to the windows registry before or on startup of *DEWEsoft®*, as described later, it will be listed in the “Plugins” tab of the hardware setup as shown in the image below. If a plug-in is checked here, it will appear in the “Plugins” tab of the measurement setup screen too.

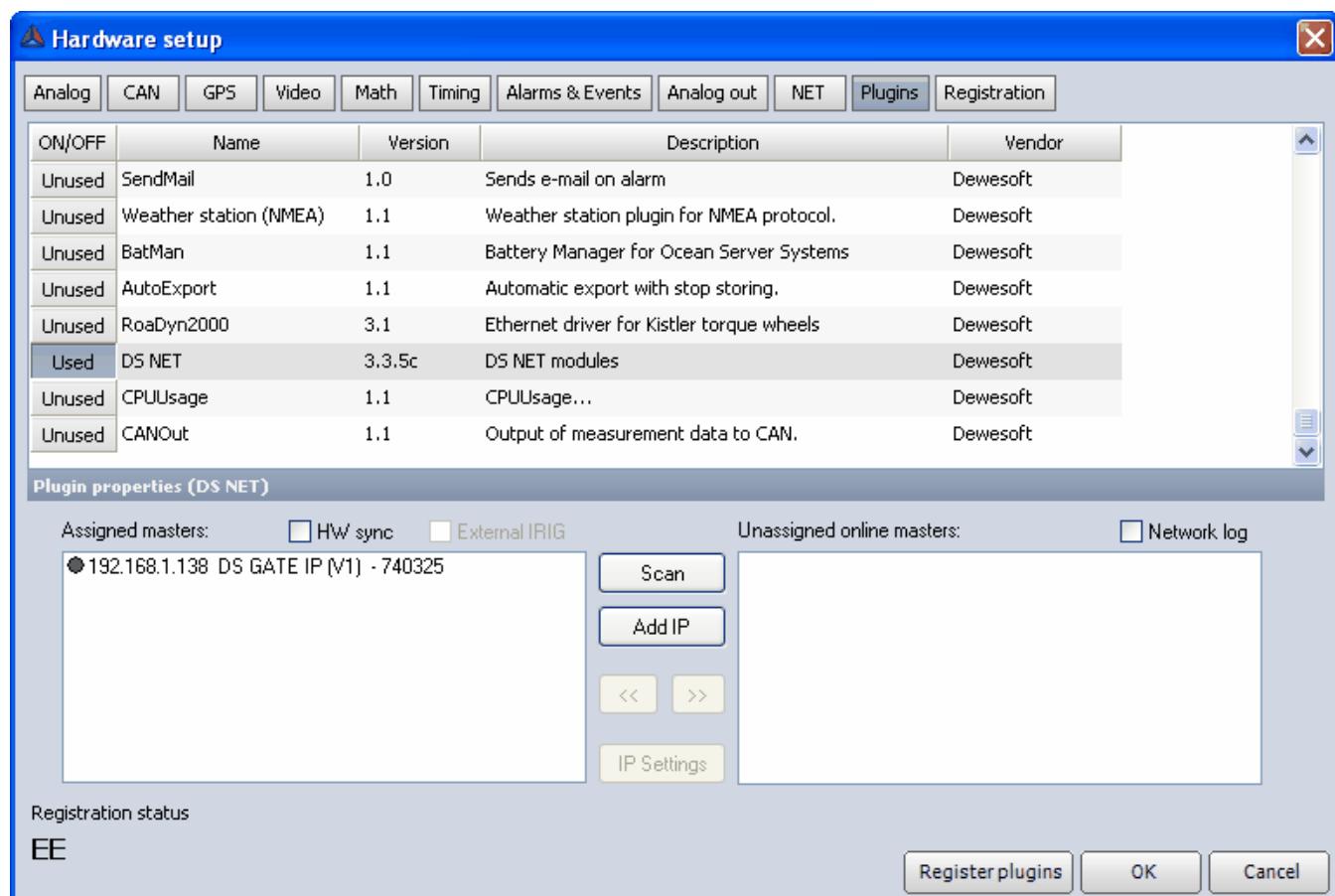


Illustration 2: Hardware Setup: Plugins

A plug-in, which is an ActiveX library (ActiveX DLL), can consist of two main parts. The one part is the plug-in implementation itself, which is necessary in any case. The other part is a frame or a window which task is to allow the user to change some settings of the plug-in. Whether this part is a frame or a separate window depends on the development environment used for creating the plug-in. In case of a plug-in designed in Delphi, using a frame would be a suitable solution. This frame will appear directly embedded into the Plugins tab of the measurement setup-screen of **DEWESoft®**:

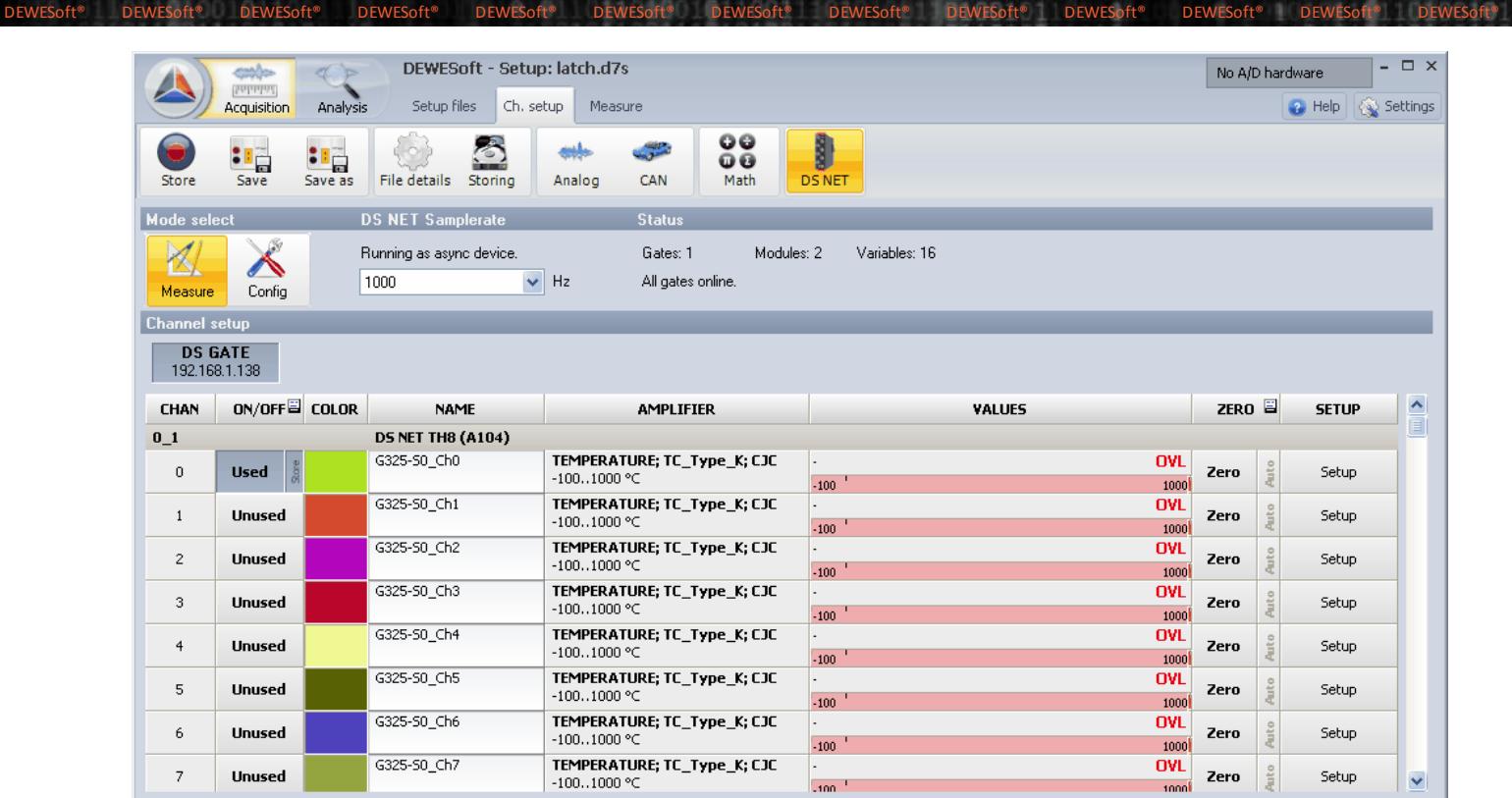


Illustration 3: Embedded Plugin

In case of a plug-in created by means of a programming environment not supporting “frames”, e.g. Visual Basic, it is not possible to create such a frame being embedded into *DEWESoft®*. Therefore settings of a plug-in have to be made in a separate window. This window can be opened by clicking the “Configure”-button which will appear instead of the frame in the Plugins tab.

1.2.1.1 Prerequisites for DEWESoft Custom Plug-ins

The [IPlugin2](#) interface is available in *DEWESoft®* Version 6.2 and newer. With version 6.3 of *DEWESoft®* the interface [IPlugin3](#) is introduced, offering additional functionalities for plug-ins. [IPlugin3](#) is derived from [IPlugin2](#), so it has to be used together with the functions of [IPlugin2](#).

DEWESoft® Version 7 introduced a new interface called [IPlugin4](#) which is a generic one and has only one function:

```
procedure OnEvent(EventID: EventIDs; InParam: OleVariant; var OutParam: OleVariant);
```

Since the parameters are variant types, this function can be used for any events. So even if new events are introduced, you need not update your Plug-Ins any more.

It is recommended to use the interfaces [IPlugin2](#), [IPlugin3](#) and [IPlugin4](#) for the development of any new plug-ins.

The older versions of *IPlugin* will still remain supported in *DEWESoft®* in order to work in conjunction with any plug-ins developed by customers so far.

1.2.1.2 Custom Visual Controls

Custom Visual Controls offers the ability to create user-defined visual controls for *DEWEsoft®*. Those *Custom Visual Control Add-Ons* will show up in the *Design mode* and can be drag&dropped like any native *DEWEsoft®* visual control to the measurement screen.

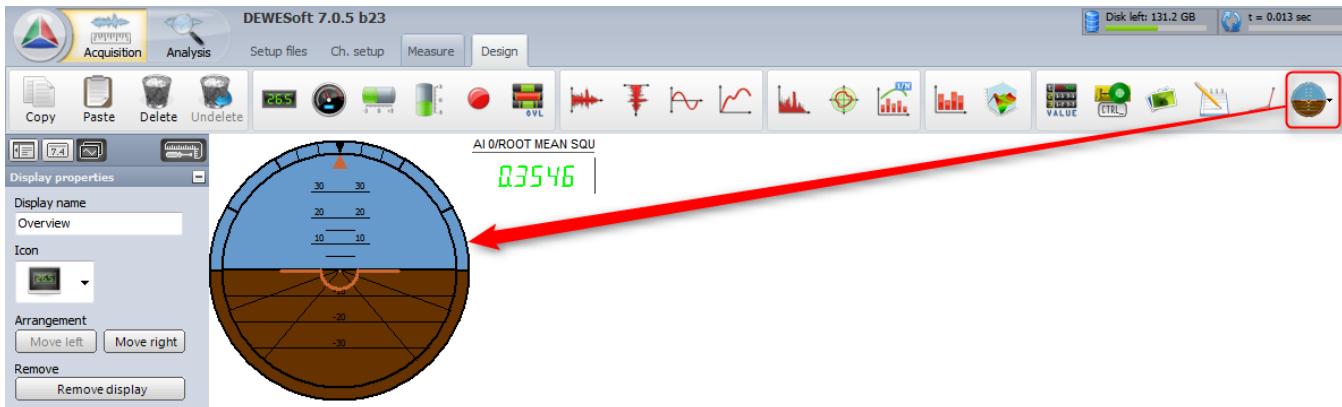


Illustration 4: Custom Visual Control

A *Custom Visual Control* is an ActiveX library having the extension .vc. If a *Custom Visual Control* library resides in the subdirectory *Addons* of the *DEWEsoft®* installation folder, it will be recognized on start-up.

see also: [IDewePlugin](#)

1.2.1.3 Custom Mathematics

Custom Mathematics offers the ability to create user-defined mathematical Add-Ons for *DEWEsoft®*. Those *Custom Mathematics Add-Ons* will show up in the *Math* section of the channel setup.

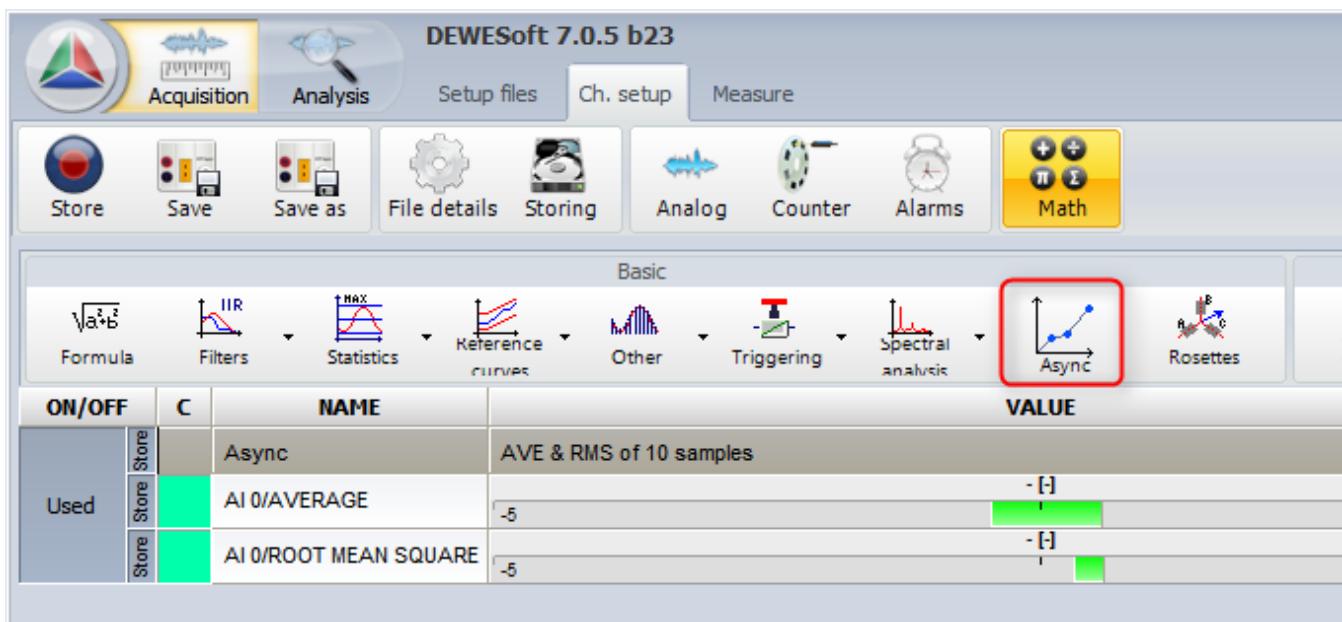


Illustration 5: Custom Math

A *Custom Mathematics Add-On* is an ActiveX library having the extension “.mth”. If a *Custom Mathematics Add-On* library resides in the subdirectory “Addons” of the *DEWESoft®* installation folder, it will be recognized on start-up.

see also: [IDewePlugin](#)

1.2.2 Custom Export

Custom Export offers the ability to create user-defined file exports for *DEWESoft®*. Such a *Custom Export* will be added to the list of exports within *DEWESoft®*.

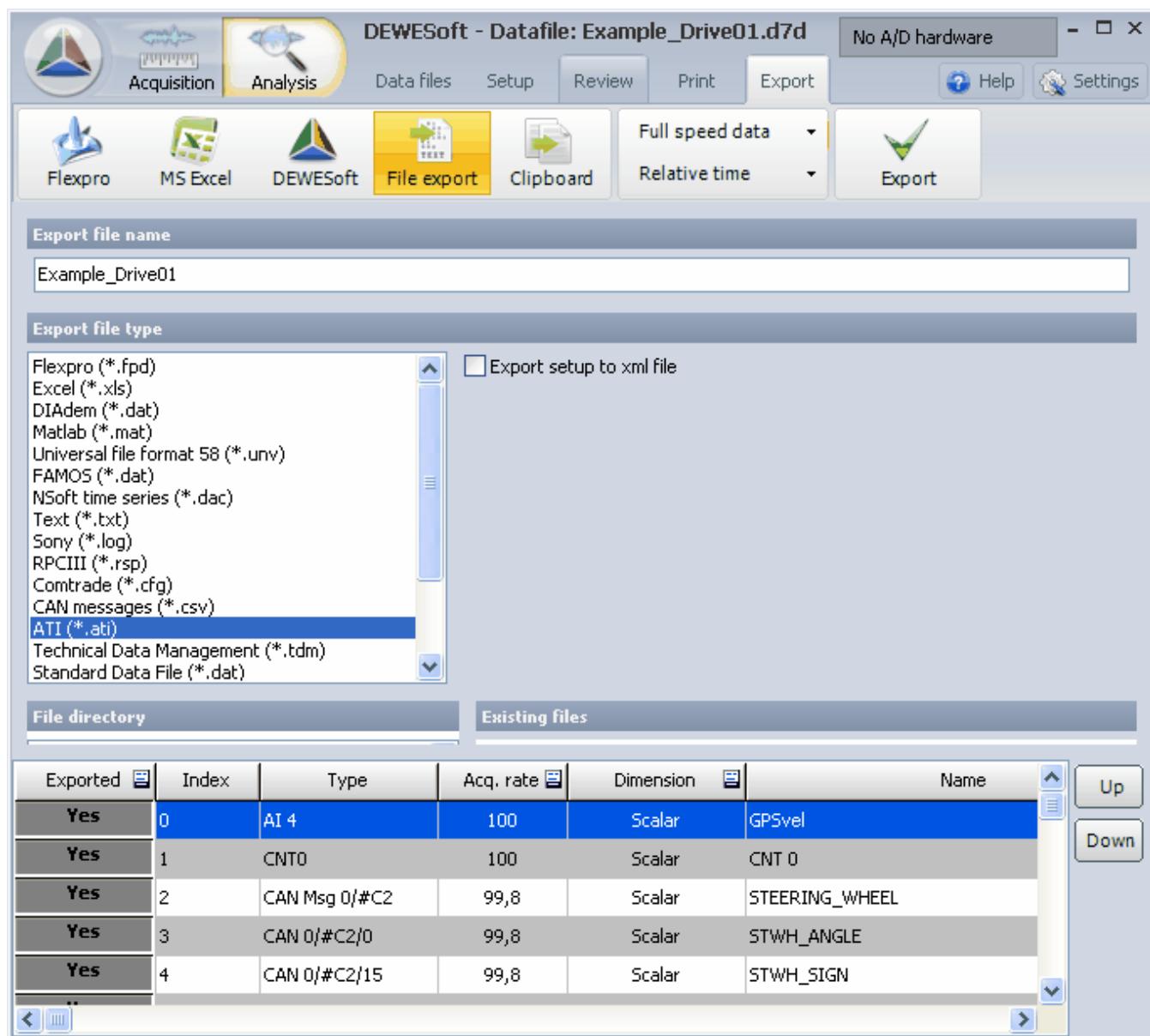


Illustration 6: Custom Export

A *Custom Export* is an ActiveX library having the extension .exp. If a *Custom Export* library resides in the subdirectory *Addons* of the *DEWESoft®* installation folder, it will be recognized on start-up.

see also: [ICustomExport](#), [ICustomExport2](#), [ICustomExport3](#)

1.2.2.1 Custom Export Call Diagram

This is the sequence in which the most important functions of a custom export will be called:

1. [ShowFrame](#)
2. [Apply](#)
3. [GetDWTypeLibVersion](#)
4. [Set_AbsoluteTime](#)
5. [Set_FileName](#)
6. [Get_SupportsAsync](#)
7. [Get_SupportsSRDiv](#)
8. [Get_SupportsDouble](#)
9. [StartExport](#)
10. [SetApp](#)
11. [Set_TimeIncrease](#)
12. [StartInfo](#)
13. [WriteInfoString](#)
14. [WriteInfoDate](#)
15. [WriteInfoInteger](#)
16. [WriteInfoSingle](#)
17. [WriteInfoString](#)
18. [EndInfo](#)
19. [StartEvents](#)
for each event [WriteEvent](#) is called

20. [StopEvents](#)
21. [SetTrigOffset](#)
22. [StartDataFolder](#)
23. [Set_DataCount](#)
24. [Get_ExportType](#)
25. [StartTimeField](#)

for each channel the following sequence will be called:

1. [StartDataField](#)
2. [SetChannel](#)
3. [SetAbsMin](#)
4. [SetAbsMax](#)
5. [SetChannelColor](#)
6. [SetRangeMin](#)
7. [SetRangeMax](#)
8. [SetDoubleFloat](#)

26. [EndHeader](#)

27. Dependant on the [ExportType](#), one of the following sequences will be called:

etValueBased

for each time-stamp to be exported:

1. [StartValue](#)
for each export channel, [WriteValue](#) is called
2. [EndValue](#)

etChannelBased

for each channel:

1. [Set_TimeIncrease](#)
2. [Set_DataCount](#)
3. [SetChannel](#)
4. [SetAbsMin](#)
5. [SetAbsMax](#)
6. [SetChannelColor](#)
7. [SetRangeMin](#)
8. [SetRangeMax](#)
9. [SetDoubleFloat](#)
10. [StartChannel](#)
for each value [WriteValue](#) / [WriteAsyncValue](#)
11. [EndChannel](#)

28. [EndDataFolder](#)

1.2.3 Custom Import

Custom Import offers the ability to create user-defined file imports to *DEWESoft®*.

The *Custom Import Add-Ons* will show up in the file type selection drop-down box in *Analyze mode*:

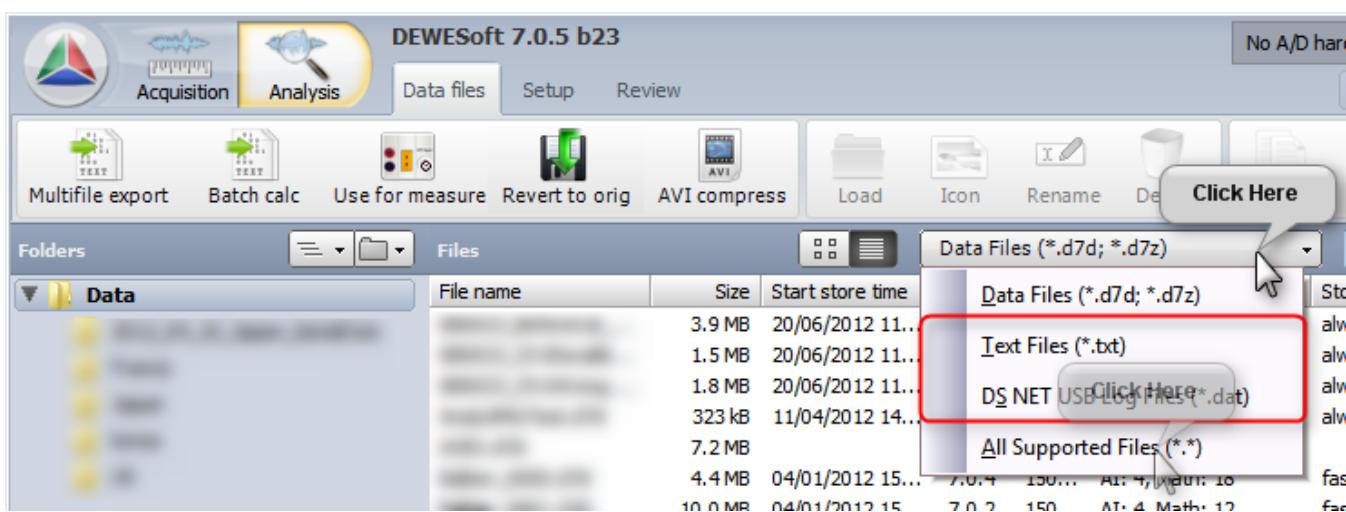


Illustration 7: Custom Import Filters

A *Custom Import* is an ActiveX library having the extension .imp. If a *Custom Import* library resides in the subdirectory Addons of the *DEWESoft®* installation folder, it will be recognized on start-up.

see also: [ImportGroup](#), [ImportChannel](#)

1.3 General

General information about the DCOM interface.

1.3.1 Legend

an overview of the icons we use in the DCOM reference.

Type	Access		Special	
		Read-only access		Automation

Type	Access		Special	
 Event		Read-Write access		Custom Export
 Enumeration				Plug-Ins
 Type				
 Property				
 Indexed Property				
 Constant				

1.3.2 The Buffer Structure

The data acquired by DEWEsoft® is stored according to the buffer structure illustrated in the illustration below.

At first the data is stored to the so-called direct buffer.

After having acquired a certain number of values, the minimum, maximum, average and RMS values are calculated and stored to the first level intermediate buffer.

There may be several levels of intermediate buffers (see [IData.IBLevels](#)).

So when there are again a certain number of values in the first level intermediate buffer, their minimum/maximum/average/RMS values will be stored in the second level intermediate buffer, and so on.

Note: in older versions of DEWEsoft® there was only one intermediate buffer and also one reduced buffer (which is not used any longer).

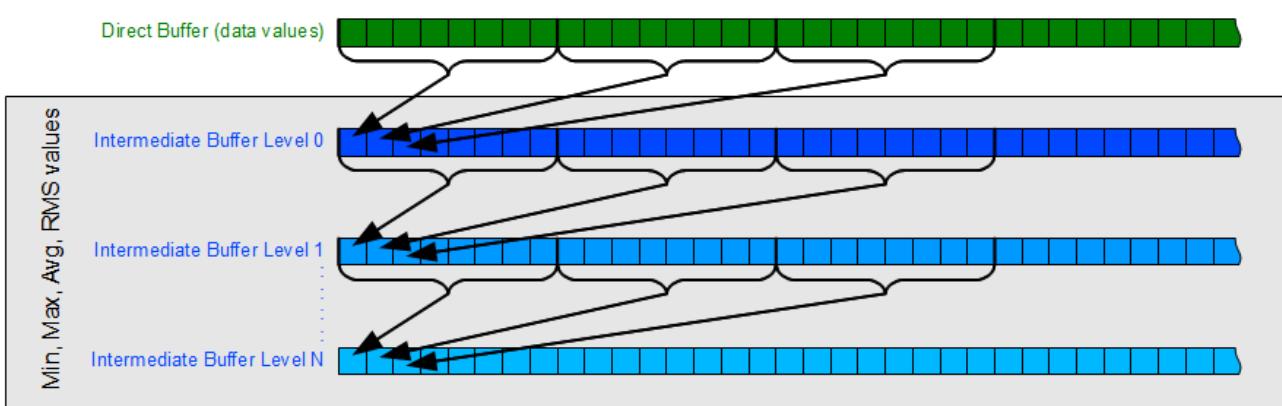


Illustration 8: DEWESoft Buffer Structure

Each of the buffers is a logical ring buffer implemented as an array. There are several important properties and functions that you need to know, to understand the logic in behind.

Example

Name	Description
IChannel . DBBufSize	This is the number of elements that the buffer can hold. It will be calculated by DEWEsoft® when you mount the channel and will be constant during the measurement (i.e. it will not change). But between measurements the size may change dependant on sample rate, etc.
IChannel . DBDataSize	This is the number of already acquired data samples that are stored in the buffer. At the start of the measurement this is 0 and will be increased with every acquired data sample to a maximum of IChannel . DBBufSize , which means that the ring buffer is now completely filled. If more samples are added to the buffer, the oldest samples will be overwritten, but IChannel . DBDataSize will not change (and stay at the value of IChannel . DBBufSize) until the end of the measurement.
IChannel . DBPos	This is the next position within the buffer to be written to. At the start of the measurement it will be 0, then it will increase up to IChannel . DBBufSize -1. and will finally wrap around to zero (and start all over again).

The only difference between the ring-buffers (direct and intermediate) is that the variables names (on [IChannel](#)) are slightly different:

for the first level of intermediate buffers: [IBBufSize](#), [IBDataSize](#), [IBPos](#)

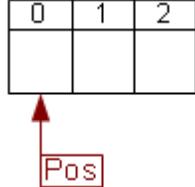
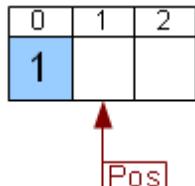
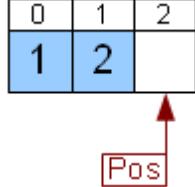
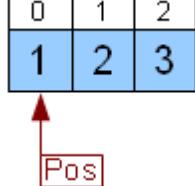
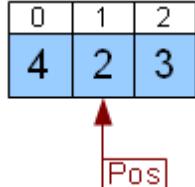
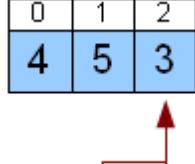
for the all levels of intermediate buffers: `IBBufSize`, `IBDataSizeEx`, `IBPosEx`, `FirstIBLevel`, `IData.IBLevels`

DEPRECATED: for the reduced buffer: `RBBufSize`, `RBDataSize`, `RBPos` --> see also: `IBValuesEx`

Detailed description

In this example we will assume a buffer size of 3 (just to make it easier to explain what's happening - DEWEsoft® will never choose such a low buffer size. In a real measurement it might be 100000).

In the *Buffer* column the small numbers (0...2) at the top are the indexes of the array slots. The larger rectangles show the data (sample number in this case): blue means that the slot already holds a valid data sample, white means that the slot is not valid yet (i.e. it holds no data).

Description	No of acquired samples	BufSize	Data Size	Pos	Buffer	Newest Idx	Oldest Idx
When the measurement starts, the buffer will be empty. The size of the buffer (<code>BufSize</code>) is 3 and will remain constant during measurement. The data size (<code>DataSize</code>) is 0 because we have not acquired data yet (i.e. the buffer is empty). <code>Pos</code> is 0: the next data sample will be written into the first slot of the array (which has index 0).	0	3	0	0		-	-
In the image on the right, you can see that the 1st sample is now stored in slot 0 of the buffer. <code>Pos</code> has been increased and now points to slot 1 (where the next sample will be stored). The data size is now 1.	1	3	1	1		0	0
The 2nd sample has been stored into slot 1 of the buffer. <code>Pos</code> has been increased and now points to slot 1. The data size is now 2.	2	3	2	2		1	0
The 3rd sample has been stored into slot 2 (the last slot) of the buffer. Now that the buffer is full, <code>Pos</code> has wrapped around and points to slot 0 again. The data size is now 3: the buffer is full.	3	3	3	0		2	0
The 4th sample will be written into slot 0 and will overwrite the 1st sample (the oldest sample in the buffer). <code>Pos</code> has been increased and will now points to slot 1. The data size is still 3: the buffer is full - this will never change again until the end of the measurement.	4	3	3	1		0	1
The 5th sample will be written into slot 1 and will overwrite the 2nd sample (the oldest sample in the buffer). <code>Pos</code> has been increased and will now points to slot 3. The data size is still 3: the buffer is full - this will never change again until the end of the measurement.	5	3	3	2		1	2

there are different ways to access the data of the buffers of a channel ([IChannel](#)):

direct buffer: [GetScaledData](#), [GetTSData](#), [GetUnscaledData](#), [IChannelConnection](#)

intermediate buffer/s: [IBValues](#), [IBValuesEx](#)

see also [T_ReducedRec](#)

Example For Buffer Index

When you want to use functions that access the buffer, you must always take care of 2 things:

- check that the buffer already contains the number of samples that you want to read:
i.e. when you want to read 10 samples from the direct buffer then check if `DBDataSize` ≥ 10 .
- handle the case when the buffer has wrapped around

Formula for index access

when you want to get the buffer index relative to the age of the data-sample, you can calculate it in one formula. We assume that `IndexByAge` is the relative index that you want to read - and of course it must be \geq the data:-

IndexByAge	Description
1	the newest data
2	the 2nd newest data
..	
DataSize	the oldest data

`BufIndex = (DBPos - IndexByAge + DBBufSize) mod DBBufSize`

Next, we will consider an alternative approach to the formula for 2 special cases:

Example: read the newest data

1. check that `DBDataSize` ≥ 1
2. since Pos always points to the next insert position, the newest data is always at `DBPos - 1`
so we set `PostoRead = DBPos - 1`
3. now we handle the special case when the buffer has just wrapped around: i.e. `DBPos = 0`
thus `PostoRead = -1` (and -1 is not a valid buffer index)
this is the case in the example above when *No of acquired samples* is 3
To correct the index we must simply add the buffer size `DBBufSize`:
`PostoRead = PostoRead + DBBufSize`
which gives the final index of 2 (-1 + 3)

Example: read the oldest data

1. check that `DBDataSize` ≥ 1
2. when the buffer has not wrapped around yet, the oldest data is at index 0
`if (DBDataSize < DBBufSize) then PostoRead = 0`
3. otherwise the oldest data is at `DBPos`:
`PostoRead = DBPos`

1.3.3 Sample Rates

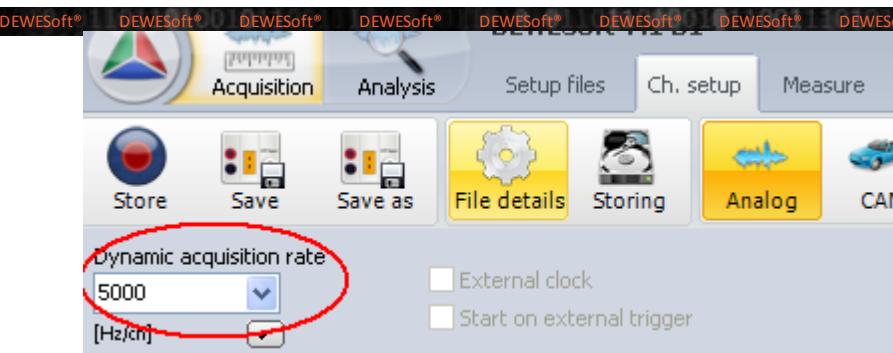
In DEWEsoft® there are several different settings for the sample rate.

The sample rate with which data is acquired can be different (see *dynamic acquisition rate*, *setup sample rate*, *reduced sample rate* below).

You can use the following members of the interface to get the information about the currently used sample rate: [IData](#), [SampleRate](#), [IData.SampleRateEx](#), [IData.Samples](#)

Dynamic Acquisition Rate

The most important sample rate is the *dynamic acquisition rate* that you can setup in the [Analog Channel setup](#):



This is the rate at which analog data will be acquired in *Measure mode*. Depending on this setting (and some other factors), DEWEsoft® will calculate the size of the buffers (see [The Buffer Structure](#)).

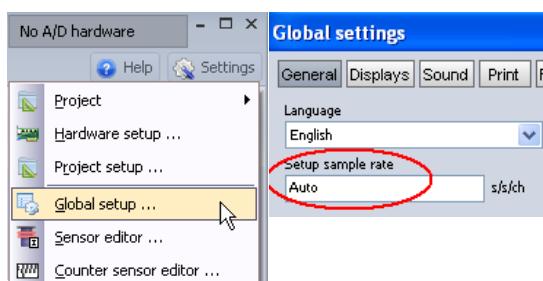
The *Dynamic acquisition rate* can be read/written via the [IApp](#) interface: see [IApp.MeasureSampleRate](#), [IApp.MeasureSampleRateEx](#)

Setup Sample Rate

DEWEsoft® will also show you live values (online data) when you are in *Channel setup mode*:



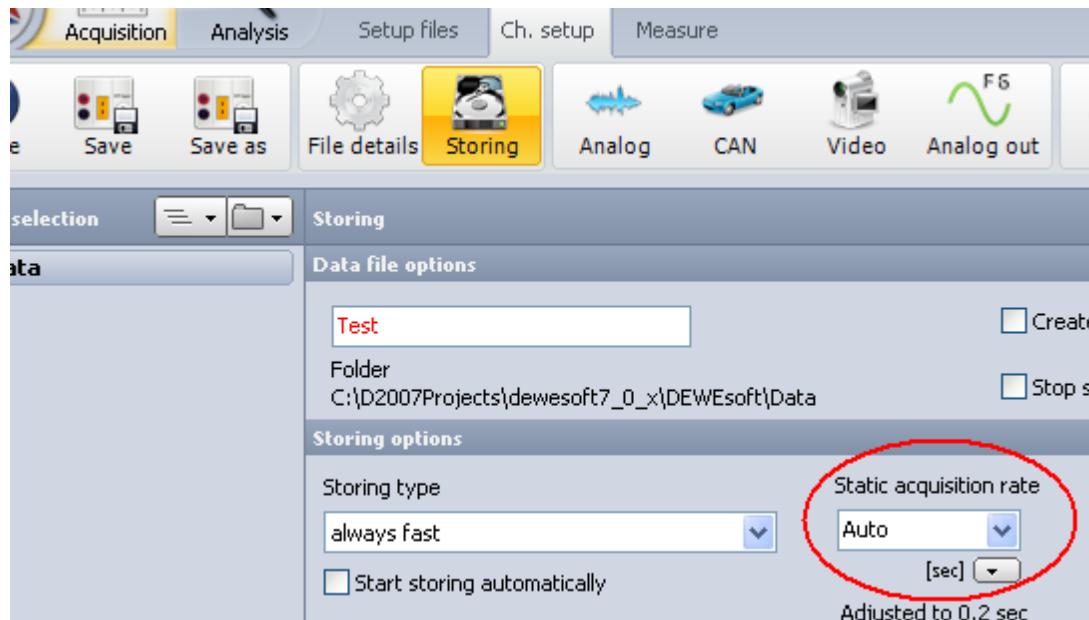
This preview data will be acquired with a slower sample rate than the *Dynamic acquisition rate*. The *setup sample rate* can be set in [Settings – Global Setup ... Setup sample rate](#).



The *Setup sample rate* can be read/written via the [IApp](#) interface: [IApp.SetupSampleRate](#)

Reduced Sample Rate

The *reduced sample rate* can be setup in the *Storing* options of *Channel setup*. The *reduced sample rate* is only relevant when the Storing type (see [IStoreEngine.StoreMode](#)) is set to: *always slow or fast on trigger, slow otherwise*.



The *reduced sample rate* can be read/written via the [IApp](#) interface: [IApp.ReducedRate](#)

Other Sample Rates

The sample rates above were all related to analogue devices.

[Plug-Ins](#) may also have their own sample rate that is not related to the sample rates mentioned above. E.g. if a Plug-In reads data from a serial device, it may have a much slower sample rate than the fast analogue channels.

1.3.4 Channels

The channel structure of **DEWEsoft®** is shown in the illustration below. The channels of the same kind are grouped together to a [ChannelGroup](#). Such a group can be e.g. for analog channels, for digital channels, for CAN data, etc. These groups are put together to the [ChannelGroups](#) which is an element of [Data](#).

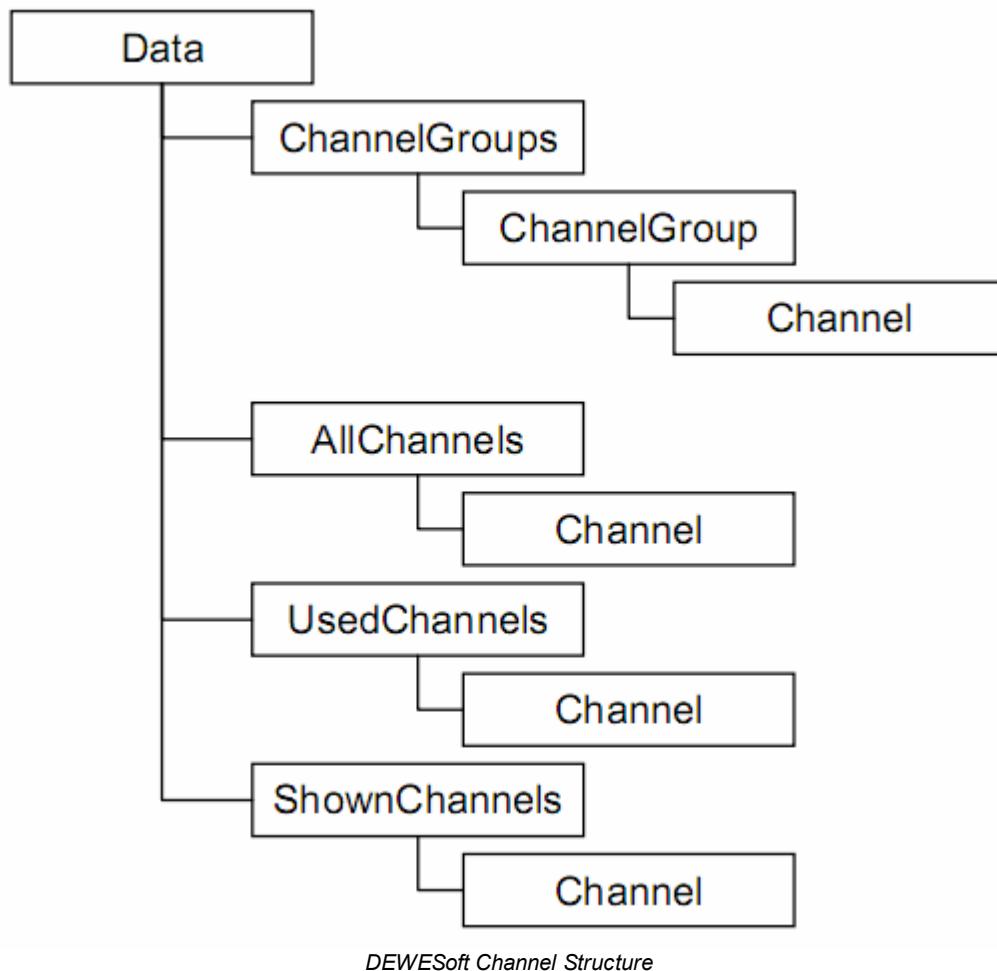
There are different types of channels:

- channels can hold synchronous, asynchronous or single: see [Synchronism](#)

- channels can contain different data types: see [Numeric Channels](#), [String Channels](#)
 - array channels can store several data points for each time-instant: see [Array Channels](#)
 - there are even channels that you can control (write values to) during measurement: [Control Channels](#)

The basic interface for accessing channels is: `IChannel`

Another way for accessing a channel would be via the list `IData.AllChannels` or `IData.UsedChannels`.



1.3.4.1 Channel Index

in *DEWEsoft®* each channel must have a unique index and this index must not change. The index is an array of integer numbers which can have a variable number of items.

Note that the channel names need not be unique: i.e. you can have several channels with the same name, but they all must have a different index.

Note: the [T_ChIndex](#) structure is deprecated since DEWEsoft® version 7 and should not be used anymore. The [T_ChIndex](#) does not include the `Index[0]` (for local or NET-option channels).

see also: [Channel Index Example](#), [IChannel.IndexEx](#), [IData.FindChannelByIndexEx](#)

Index[0]

Dec	Description
0	local system
1...n	remote measurement units that are connected via the DEWE-NET option to a master/view client or a master measurement unit

Index[1]

Dec	Description
1	Analog
100	Digital
200	Counter
1000	PAD
2000	CAN
4000	GPS
7000	Math
100000	Plugins
600000	Variables
700000	Video timestamps
800000	Import

Index[2]..Index[IndexLevel]

Note: the column `Index[0]` has been removed from the table below for readability.

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Channel type	Details	IndexLevel	Index1	Index2	Index3	Index4						
AI Analog Input		3	1	slot number (0...number of slots-1)	-	-						
		Example: Channel "AI 3" ? (3; 0, 1, 3)										
DI Digital Input	Port Channel	3	100	port number (0...number of ports-1)								
		Example: Channel "DI port 2" ? (3; 0, 100, 2)										
	Digital Channel	4	100	port number (0...number of ports-1)	bit number (0...bits per port-1)	-						
		Example: Channel "bit no 3 on DI port 2" ? (4; 0, 100, 2, 3)										
Counter	Master counter channel	3	200	slot number (0...number of slots-1)								
		Example: Channel "counter 0" ? (3; 0, 200, 0)										
	Additional counter channels	4	200	slot number (0...number of slots-1)	additional channel index (0...number of add. channels-1)	-						
		Example: Channel "counter 0" ? (4; 0, 200, 0, 0)										
PAD		5	1000	interface id (reserved for future use, always 0 now)	slot number (0...number of slots-1)	channel number in slot (0...number of ch. in slot-1)						
		Example: Channel "PAD slot 3/Ch 2" ? (5; 0, 1000, 0, 3, 2)										
CAN bus	Message channel	4	2000	port number (0...number of ports-1)	arbitration ID							
		Example: channel "RPM in Motor message on CAN port 0" ? (4; 0, 2000, 0, 33 [Motor message id])										
	Decoded value channel	5	2000	port number (0...number of ports-1)	arbitration ID	unique integer value within message						
		Example: channel "RPM in Motor message on CAN port 0" ? (5; 0, 2000, 0, 33 [Motor message id], 16 [start bit of RPM channel])										
GPS		3	4000	GPS channel: 0...X absolute 1...Y absolute 2...Z 3...Velocity 4...Velocity Z 5...Direction 6...Distance 7...used satellites 8...Current second 9...Mark input 10...PDOP								

1.3.4.1.1 Channel Index Example

see also: [Channel Index](#)

In the illustration below you can see a screenshot from the channel list in Measure Mode. We have 2 analogue channels and some channels from 2 plugins:



Illustration 9: Channel List in
Measure Mode

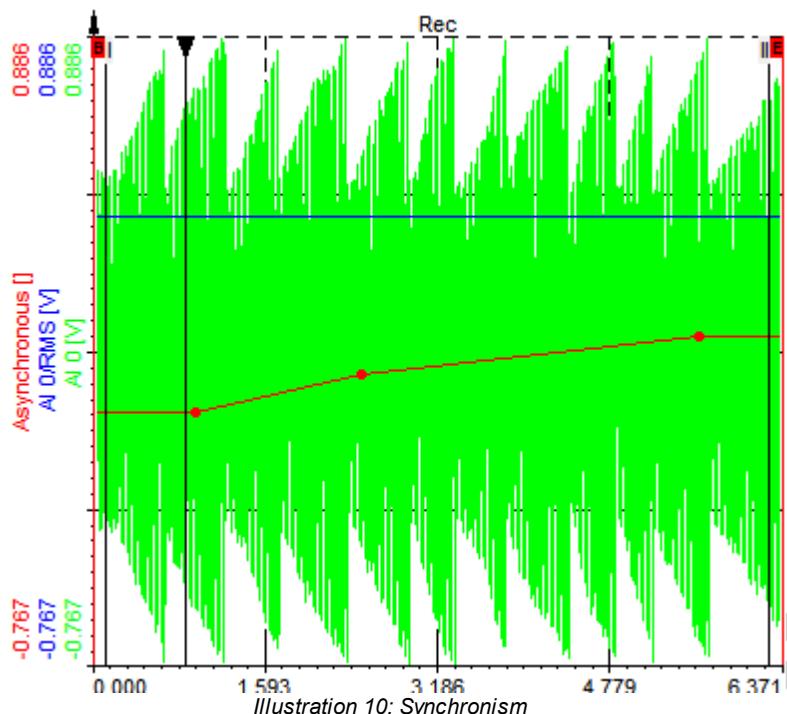
The channel indexes for the example above are shown and explained in the following table.

Group/Channel Name	Index	Description
AI	[0,1]	Group for the analogue channels. The first index is 0, because this channel is part of the local DEWEsoft® instance - see definition in Channel Index The 2nd index is 1 for all analogue channels - see definition in Channel Index
AI A0/0	[0,1,0]	The first analogue channel - belongs to the group above and since it is the first channel, it has the index 0 (3rd item in the array).
AI A0/2	[0,1,2]	The 3rd analogue channel - belongs to the group above and since it is the 3rd channel, it has the index 2 (3rd item in the array). note: the 2nd analogue channel has not been set to <i>Used</i> in the channel setup, so it will not show up in the channel list. also note: that the index of this channel is always - no matter if the 2nd analogue channel is set to <i>Used</i> or not.
Plugins	[0,100000]	Plugins group: automatically created by DEWEsoft®
KGG-Plugin	[0,100000,962870 596]	KGG-Plugin group: automatically created by DEWEsoft® for the KGG-Plugin.
Control Channels	[0,100000,962870 596,0]	1st group of the KGG-Plugin (the channels of the group are not shown in the illustration above).
Wheels	[0,100000,962870 596,1]	the Wheels group has been mounted by the KGG-Plugin. Delphi-code to mount this group: <pre>Ind := VarArrayCreate([0, 0], varInteger); // create array with one item Ind[0] := 1; // index of this group is 1 - 'Control Channels' group had 0 PluginGroup.AddIndexNameEx(GUIDToString(CLASS_Plugin), 1, Ind, 'Wheels');</pre>
Wheel1	[0,100000,962870 596,1,0]	group for the 1st wheel - this is the 1st subgroup of the Wheels group. Delphi-code to mount this group: <pre>Ind := VarArrayCreate([0, 1], varInteger); // create array with 2 items Ind[0] := 1; // 1 is the index of the parent group 'Wheels' Ind[1] := 0; // 0, because this is the very first subgroup of 'Wheels' PluginGroup.AddIndexNameEx(GUIDToString(CLASS_Plugin), 2, Ind, 'Wheel1');</pre>
W1_dist	[0,100000,962870 596,1,0,0]	1st channel of the Wheel1 group Delphi-code to mount this channel: <pre>Ind := VarArrayCreate([0, 2], varInteger); // create array with 3 items Ind[0] := 1; // 1 is the index of the parent group 'Wheels' Ind[1] := 0; // 0 is the index of the parent group 'Wheel1' Ind[2] := 0; // 0 because this is the first channel in the group 'Wheel1' NewChannel := PluginGroup.MountChannelEx(GUIDToString(CLASS_Plugin), 3, Ind);</pre>
W1_vel_f	[0,100000,962870 596,1,0,1]	2nd channel of the Wheel1 group Delphi-code to mount this channel: <pre>Ind := VarArrayCreate([0, 2], varInteger); // create array with 3 items Ind[0] := 1; // 1 is the index of the parent group 'Wheels' Ind[1] := 0; // 0 is the index of the parent group 'Wheel1' Ind[2] := 1; // 1 because this is the 2nd channel in the group 'Wheel1'</pre>

1.3.4.2 Synchronism

In DEWESoft® there are 3 fundamentally different types of channels with regard do synchronism: synchronous channels, asynchronous channels and single value channels.

The following screenshots show the 3 types: the channel *AI 0* (green) is a synchronous channel, the channel called *Asynchronous* (red) is an asynchronous channel and the channel called *AI 0/RMS* (blue) is a single value channel.



t = 2.474.8; AI 0 = 0.536 V

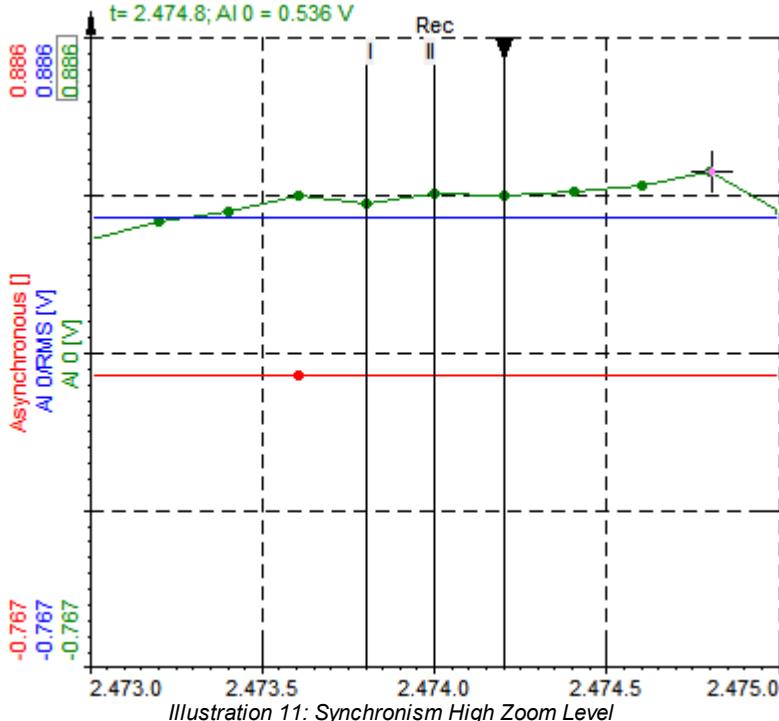


Illustration 11: Synchronism High Zoom Level

Synchronous channel

For *synchronous* channels the data are coordinated under the centralized control of a fixed-rate clock signal (which is called *Master Clock* in *DEWESoft®* speak): thus the data are equidistant.

When you look at the illustration *Synchronism High Zoom Level*, you can see that the time between the data points of the synchronous channel *AI 0* (shown in green) is always the same.

Asynchronous channel

In contrast to *synchronous* channels, the data of *asynchronous* channels are not coordinated to the Master Clock; the time-distance between the data points is arbitrary.

When you look at the illustration *Synchronism*, you can see that the time between the data points of the asynchronous channel *Asynchronous* (shown in red) is not equidistant - the data may arrive at arbitrary times.

Single Value channel

Single value channels have only one single value for the whole measurement.

When you look at the illustrations above, you cannot see a dedicated data point for the single value channel *AI 0/RMS* (shown in blue) at all. In this example the channel is a *Math* channel that calculates the RMS value of channel *AI 0* for the whole measurement; i.e. we only have exactly one value for this channel.

see also: [IChannel.SetIsSingleValue](#), [IChannel.SingleValue](#), [IChannel.Text](#) (for single value [String Channels](#))

1.3.4.3 Numeric Channels

The vast majority of channels in *DEWESoft®* are numeric channels. They can hold numeric data with different precision (e.g. a single Byte, or a floating point number - see [Data Types](#) for details). You can use those numeric channels in [Mathematics](#) to do calculations or display them directly in visual controls.

see also: [Data Types](#)

1.3.4.4 Textual Channels

In *DEWESoft®* there are 2 ways to store display textual data in a channel:

String Channels

A string channel is actually an array channel of data type `Byte` (see: [IChannel.DataType](#)) with the [IChannel.MType](#) set to character.

see also: [How to: Mount Dewesoft Channels - String Channel](#), [IChannel.AddAsStringString](#), [IChannel.SetStringChannel](#)

Text Channels

A text channel is a channel with data type [Text](#) (see: [IChannel.DataType](#)). It only stores one text item and thus behaves like a single value channel (see [Synchronism](#)).

see also: [How to: Mount Dewesoft Channels - Text Channel](#)

see also: [IChannel.Text](#)

1.3.4.5 Array Channels

Array channels can store several data points for each time-instant. You can use a 3D graph to display this data. In the example below, you can see an FFT channel in a 3D graph. The red arrow shows the direction of the time axis; i.e. for each time instant there is a complete array of values.

Array channels are always asynchronous (see also [Synchronism](#)).

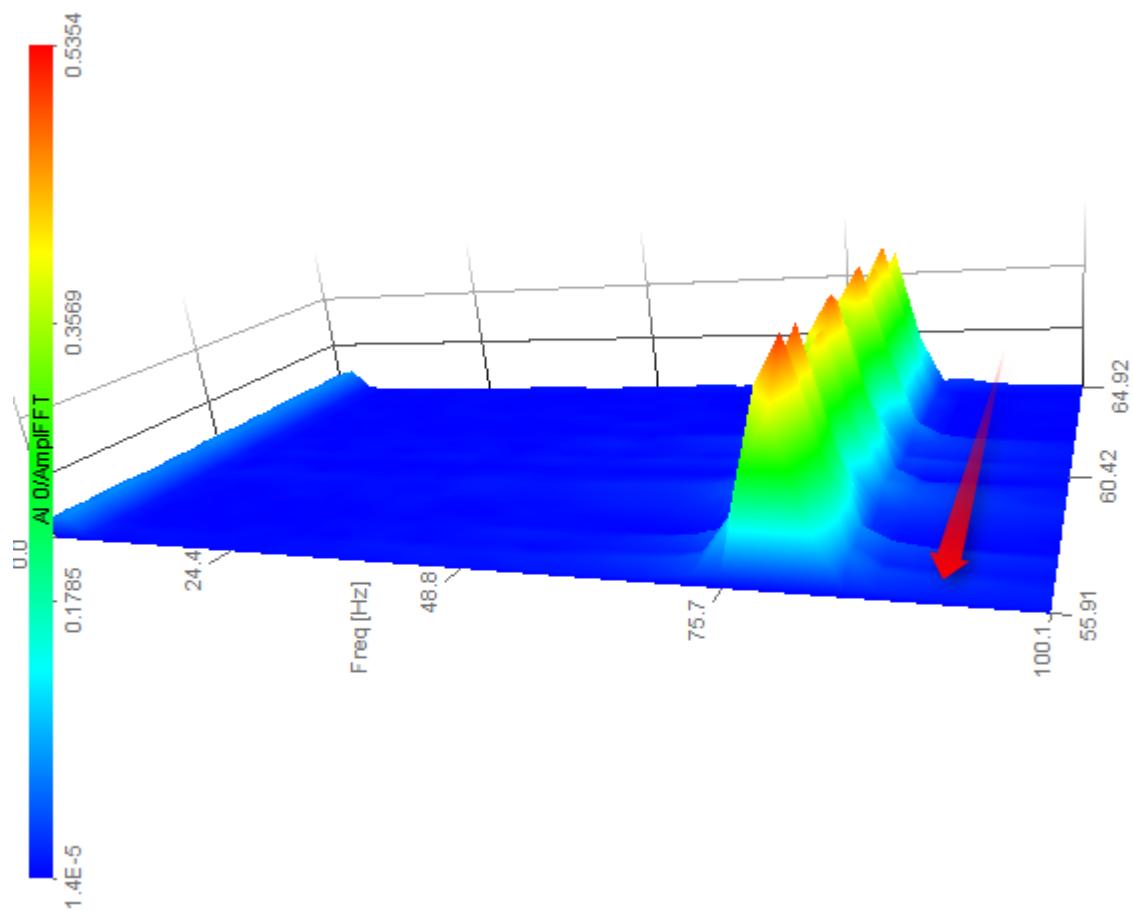


Illustration 12: Array Channel in a 3D display

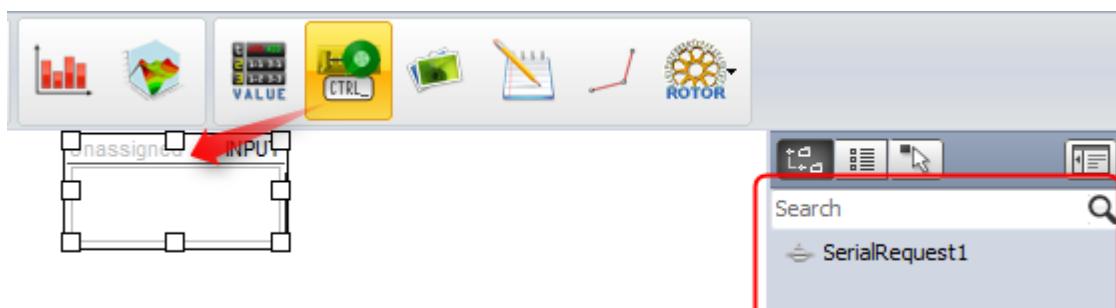
see also: [IChannel.ArrayChannel](#), [IChannel.ArrayInfo](#), [IChannel.ArraySize](#)

1.3.4.6 Control Channels

Control channels can be assigned to a visual control (e.g. a push-button) and then the user can change the value during measurement; i.e. data is written to the control channel.

In plugins you may check the values of such a control channel (in the [IPlugin.OnGetData](#) method) and act accordingly: e.g. reset some data when the user pushes a button.

In design mode, you can drag a control channel visual control to the measurement screen (note that only control channels will be available in the channel list for this visual control):



The screenshot shows the DEWEsoft Control Panel interface. At the top, there are tabs for Acquisition (selected), Analysis, Setup files, Ch. setup, Measure, and Design. Below the tabs is a toolbar with icons for Copy, Paste, Delete, Undelete, and various status indicators (265, battery level, signal levels). The main area displays a control element named "SerialRequest1" with the label "INPO". On the left, the "Control properties" panel is open, showing settings for "Transparent" and "Unified properties". Under "Display type", the "Control Channel" option is selected. A dropdown menu for "Control type" is open, with "Push button" highlighted by a red arrow. Other options in the dropdown include Input field, Push button, Switch, Turn knob, Horizontal slider, and Vertical slider.

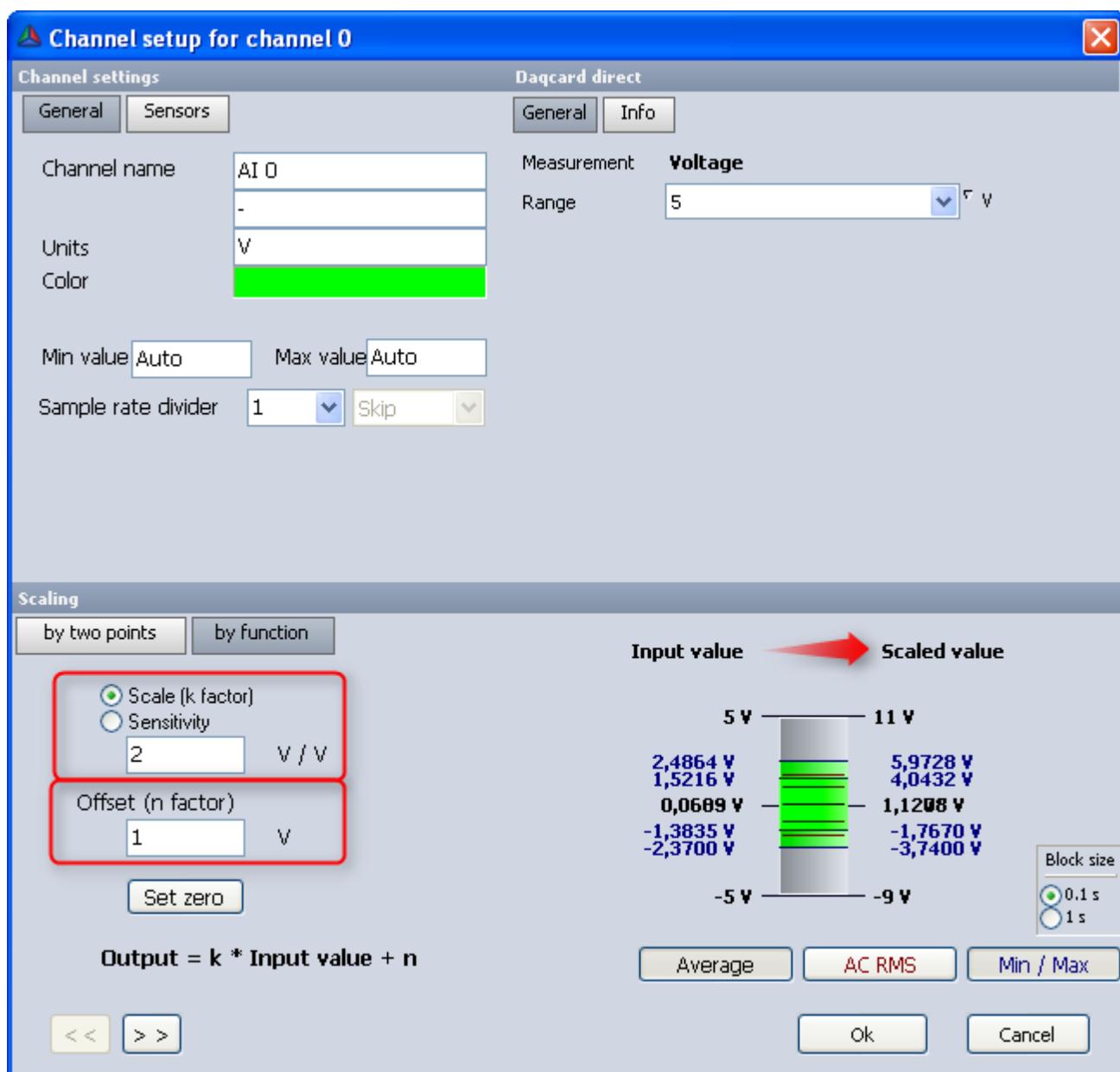
see also: [IChannel.IsControlChannel](#)

1.3.4.7 Scaling

Channels can easily be scaled in their setup dialog. You can enter a scale factor (see [IChannel.Scale](#)) and an offset (see [IChannel.Offset](#)).

In the image below you can see that we have used a scale factor of 2 and an offset of 1 V. On the right bottom you can see a live preview of the scaling.

The maximum input value of 5 V results in a scaled value of 11 V ($5 \times 2 = 10$, $10 + 1 = 11$)



see also: [IChannel.Scale](#), [IChannel.Offset](#), [IChannel.ScaleValue](#), [IChannel.GetScaledData](#), [IChannel.GetScaledDataEx](#), [IChannel.GetScaledDataEx1](#), [IChannel.GetUnscaledDataEx](#), [IChannel.GetUnscaledDataEx1](#), [IChannel.GetScaledDataEx](#)

1.3.5 Data Types

This is an overview of all possible data types for DEWESoft® channels.

DataType	Channel Datatype Value see IChannel.DataType	Storage size [Bytes]	Range
Byte	0	1	0 to 255
ShortInt	1	1	-127 to 127
Word	3	2	0 to 65,535
SmallInt	2	2	-32,768 to 32,767
Longword	8	4	0 to 4,294,967,295
Integer	4	4	-2,147,483,648 to 2,147,483,647
Int64	6	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Single	5	4	7 significant digits, exponent -38 to +38
Double	7	8	15 significant digits, exponent -308 to +308
ComplexSingle	9	8	complex number real and imaginary part are variables of type Single
ComplexDouble	10	16	complex number real and imaginary part are variables of type Double
Text	11	-	see also Textual Channels

see also: [IChannel.DataType](#)

1.3.5.1 DateTime

Delphi (and thus DEWESoft®) uses variables of type `Double` to represent data/time values.

The integral part of the `Double` value is the number of days that have passed since *30 Dec. 1899*. The fractional part of the `Double` value is the fraction of a 24 hour day that has elapsed.

Some examples:

Value	Date/Time
0	12/30/1899 12:00 am
2.75	01/01/1900 06:00 pm
1.25	12/29/1899 06:00 am
35065	01/01/1996 12:00 am

see also: [TDateTime](#)

1.3.5.2 TDateTime

`TDateTime` is just a type definition for a variable of type `Double` that holds specially encoded date and time information:
see [DateTime](#) for details.

1.3.5.3 HRESULT

The `HRESULT` (for result handle) is a way of returning success, warning, and error values. `HRESULTs` are really not handles to anything; they are only values with several fields encoded in the value. As per the COM specification, a result of zero indicates success and a nonzero result indicates failure.

see also: [MSDN - Error Handling in COM](#)

1.3.5.4 TColor

The type `TColor` in Delphi is a specific 4-byte number. The lower three bytes represent RGB (red, green, blue) color intensities for red, green, and blue, respectively.

Therefore, the color red can be defined as `TColor($0000FF)`.

see also [ColourCodes](#)

1.3.5.4.1 ColourCodes

The default colours of `DEWEsoft®` and their order are as follows:

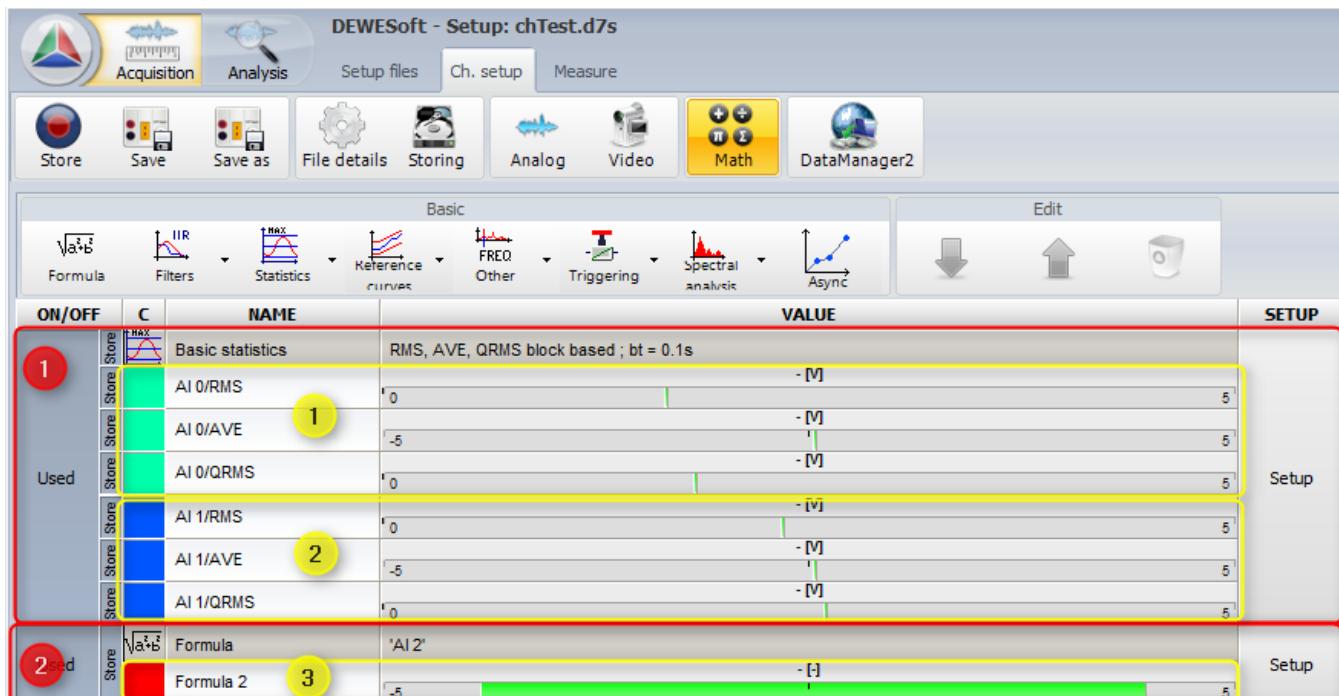
```
clLime, clAqua, clRed, clFuchsia, clBlue, clTeal, clOlive, clGreen, $00FF8080,  
$00FF80FF, $0080FF00, $00FFFF80, $00D6DAD6, $0081ABB1, $00CF63C7,  
$0065CD92, $0043E7EF, $005ED5D5, $00953AC9, $00F19A9C, $00AFE6A6, $00A0A0DEB,  
$0055CAB9, $0013E6E6, $0010CBCB, $00EA738A, $002E04EE, $004D5ED2, $009F9FA2,  
$00C67362, $00C67362, $00AA027C, $00AA027C, $00BDFFFED, clWhite, $005E0DE3,  
$008F8D61, $0021E4AA, $002C49D3, $00BC05B3, $002E03BE, $0093FDF0, $00026458,  
$00BC4151, $0040A293, $00656569, $00B37015, $0002DBCB, $0021FEF3, $0040B09A,  
$008CFFDF, $009A8BFE, $00716F4D, $005B49FE, $0033079A, $0001DA3D, $00020DD2,  
$0074852E, $003F41A0, $00727070, clYellow, clSilver, clActiveCaption, clBtnFace
```

see also: [TColor](#), [IChannel.MainDisplayColor](#), [ICustomExport2.SetChannelColor](#)

1.3.6 Mathematics

You can access the mathematic functions in DEWESoft® via [IApp.Math](#), which provides a list of math objects and these math objects ([IMathObject](#)) in turn provide access to their math module/s ([IMathModule](#)).

A math module has sets of input and output channels. Each math object can have 1 or more math modules: the math object has some properties that apply to all its modules.



The illustration above shows 2 math objects (the red areas marked with the red numbered circles):

(1) is a *Basic statistics* math object ([IMathObject](#)) which has 2 math modules (the yellow areas marked with the yellow numbered circles):

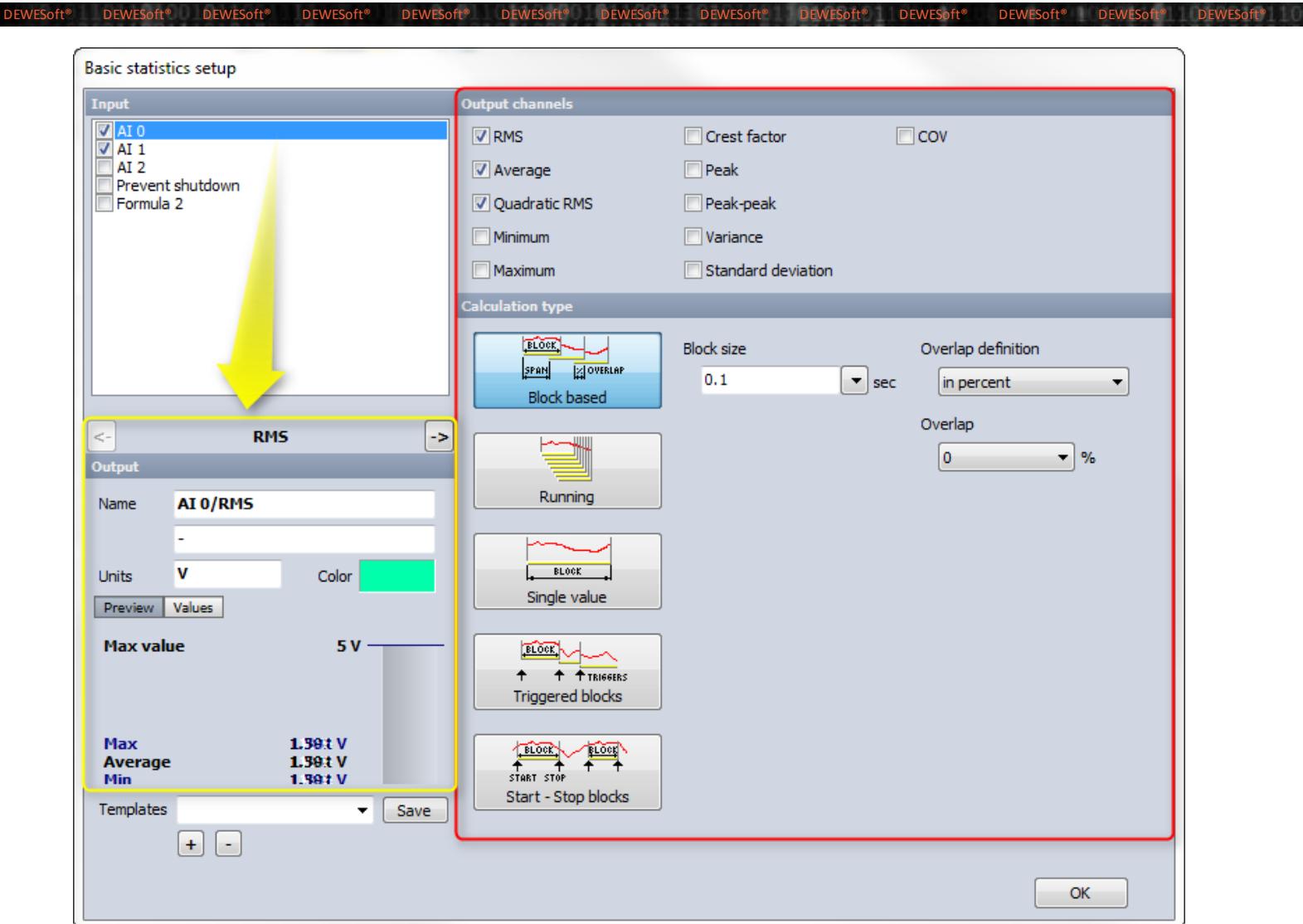
(1) first math module of the *Basic statistics* math object which has the channel *AI 0* as input and has 3 output channels *AI 0/RMS*, *AI 0/AVE*, *AI 0/QRMS*

(2) second math module of the *Basic statistics* math object which has the channel *AI 1* as input and has 3 output channels *AI 1/RMS*, *AI 1/AVE*, *AI 1/QRMS*

Note: actually the *Basic statistics* math object has always 11 output channels, but in this case only 3 of the are used (you can change this in the *Setup* of the math object)

(2) is a *Formula* math object ([IMathObject](#)). *Formula* math objects always have exactly one input channel, one output channel and one math module (see in (3) the illustration above).

Each math object ([IMathObject](#)) has it's own context ([IMathObject.MathObjContext](#)):



In the example above the the **Basic statistics** math object has some settings that are the same for each of it's math modules: in this case the desired output channels and all other settings in the red square.

The *Input* selection box at the left top shows a list of all channels that this math object supports. For each selected channel the math object will create a different math module.

The math module has a math object context ([IMathObject.MathObjContext](#)) in which it can store all relevant settings.

1.3.7 XML Setup

Since DEWEsoft® version 7 the channel setup data is stored in XML format which makes it easy for other applications to read, change or write DEWEsoft® channel setup files.

The setup data is stored in XML files. The XML file structure makes it easy to read the setup (also with 3rd party tools - XML editors).

There is a detailed documentation about the structure and format of these files available as download from our homepage:

<http://www.dewesoft.com/dewesoft7/developer-downloads> search for Dewesoft7XML (at the time of writing this documentation the current version is Dewesoft7XML103)

Note: you need to log in with your user-account before you can access these developer download files.

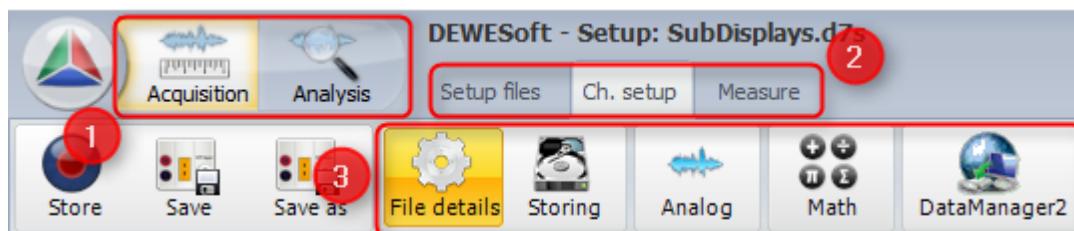
see also: [IChannel.GetChannelSetup](#), [IChannel.SetChannelSetup](#), [IChannel.UpdateXML](#)

1.3.8 GUI Navigation

The DEWEsoft® is very flexible and will always dynamically adjust to the given mode.

Main Menu Buttons

The following illustration shows some of the main GUI navigation elements of DEWEsoft® in channel setup:



1. Mode selection: The top level of GUI navigation consists of two modes:

[Measure Mode](#): in this mode DEWEsoft® will acquire measurement data and store the data in measurement files, which you can then analyse in *Analysis* mode

[Analysis Mode](#): in the *Analysis* mode you can do detailed analysis of DEWEsoft® measurement data (and also export this data to other file formats)

2. Main Toolbar: the available items of the main toolbar are dependant on the mode DEWEsoft® is running in ([Measure Mode/Analysis Mode](#)): see also [IApp.SetMainToolBar](#)

3. Main Menu Buttons: the available items of the main toolbar are dependant on the mode DEWEsoft® is running in ([Measure Mode/Analysis Mode](#)), on the currently selected item of the main toolbar ([IApp.SetMainToolBar](#)) and on the and on the hardware setup (e.g. *DataManger2* in the illustration above is a DEWEsoft® plugin that can be activated or deactivated in hardware setup)

Screens (aka. Instruments)

The following illustration shows some of the main GUI navigation elements of DEWEsoft® while measuring:



Compared to the *Main Menu Buttons* example above, there are no main menu items visible, but we see DEWESoft® measurement screens (aka. instruments)

1. **Mode selection:** The top level of GUI navigation consists of two modes:

[Measure Mode](#): in this mode DEWESoft® will acquire measurement data and store the data in measurement files, which you can then analyse in *Analysis mode*

[Analysis Mode](#): in the *Analysis mode* you can do detailed analysis of DEWESoft® measurement data (and also export this data to other file formats)

2. **Main Toolbar:** the available items of the main toolbar are dependant on the mode DEWESoft® is running in ([Measure Mode/Analysis Mode](#)): see also [IApp.SetMainToolBar](#)

3. **Screens (aka. Instruments):** the available screens can be customized by the user: see also [IApp.Screens](#)

4. **Subscreens:** the user can also define sub-screens: see [IApp.Screens](#)

1.3.8.1 Measure Mode

Measure (Acquisition) mode: in this mode DEWESoft® will acquire measurement data and store the data in measurement files, which you can then analyse in [Analysis Mode](#).

Calling [IApp.Measure](#) will put DEWESoft® in *Measure (Acquisition)* mode, which is the same as pressing the *Acquisition* button in the GUI:



Besides the general navigation ([IApp.SetMainToolBar](#)), there are 2 special functions for switching to setup and to the measurement instruments:

Setup

Calling [IApp.SetupScreen](#) will switch to the setup screen. DEWESoft® must be in *Measure* mode. This is the same as

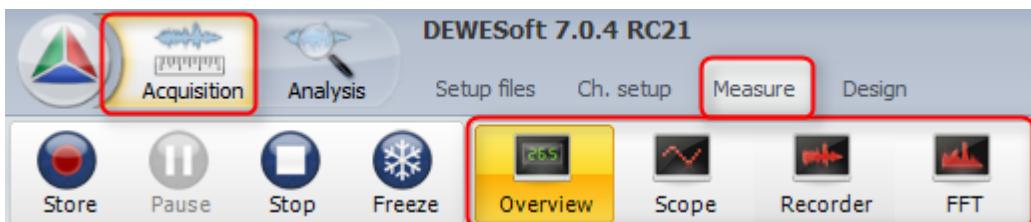
clicking the *Ch. setup* button:

A general way to activate a main toolbar item or a main menu button is to call `IApp.SetMainToolBar`.



Measurement Screens (Instruments)

Calling [IApp.GoToInstruments](#) will take you to the measurement instruments (aka. measurement screens - see [IApp.Screens](#)). Here you can take a look at the live data, start storing, etc.



1.3.8.2 Analysis Mode

Analysis mode: in the *Analysis* mode you can do detailed analysis of DEWEsoft® measurement data (and also export this data to other file formats)

Calling `IApp.Analyze` will put DEWEsoft® in *Analysis* mode, which is the same as pressing the *Analysis* button in the GUI:

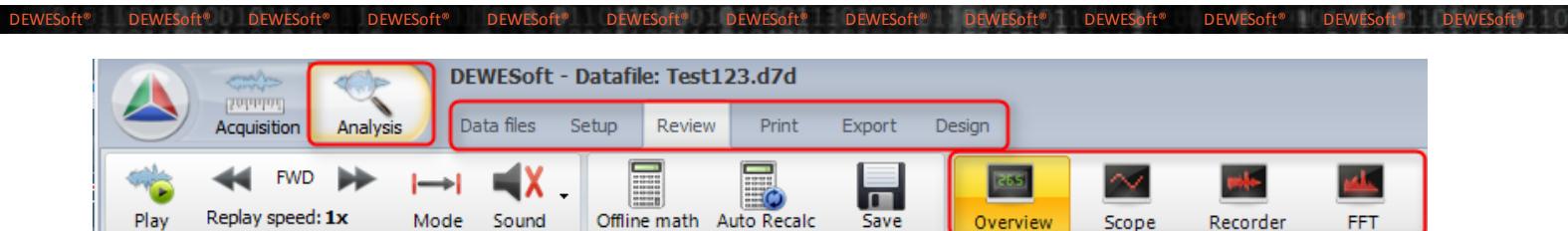


In *Analysis* mode the available menu items are different if you have already loaded a DEWEsoft® measurement file or not.

When no file is loaded:



After loading a file:



1.3.9 Terms

Describes some terms that are used throughout this documentation.

1.3.9.1 Calculation Delay

DEWEsoft® applies an internal calculation delay (default about 200ms - plugins may change that) to the external data, before the calculations are done.

This is important because you may have external data sources that take some time until they pass the data to *DEWEsoft®* (e.g. an Ethernet based device may send it's data in blocks every 100ms).

If the data are not received within the calculation delay, the last value of the channel will be used for the calculations. If the missing data are then later added to the channel, the calculations will already be done; i.e. the data in the direct buffer of the channel will be there, but the calculation may be "wrong". This can be fixed by simply recalculating the math channels in *Analyse* mode.

see also: [ICChannel.CalcDelay](#), [IData.MaxCalcDelay](#)

Calculation Delay Example

To illustrate the calculation delay, we use a simple plugin that will pretend a delay in the available data.

Source data

The plugin write the current time since the start of the measurement into an asynchronous channel whenever the [OnGetData](#) function is called . But after 3 seconds, the plugin will pause for a second: i.e. it will not write any data to the channel in this pause-time.

After 3.6 seconds of measurement the recorder graph in *DEWEsoft®* will look like this:

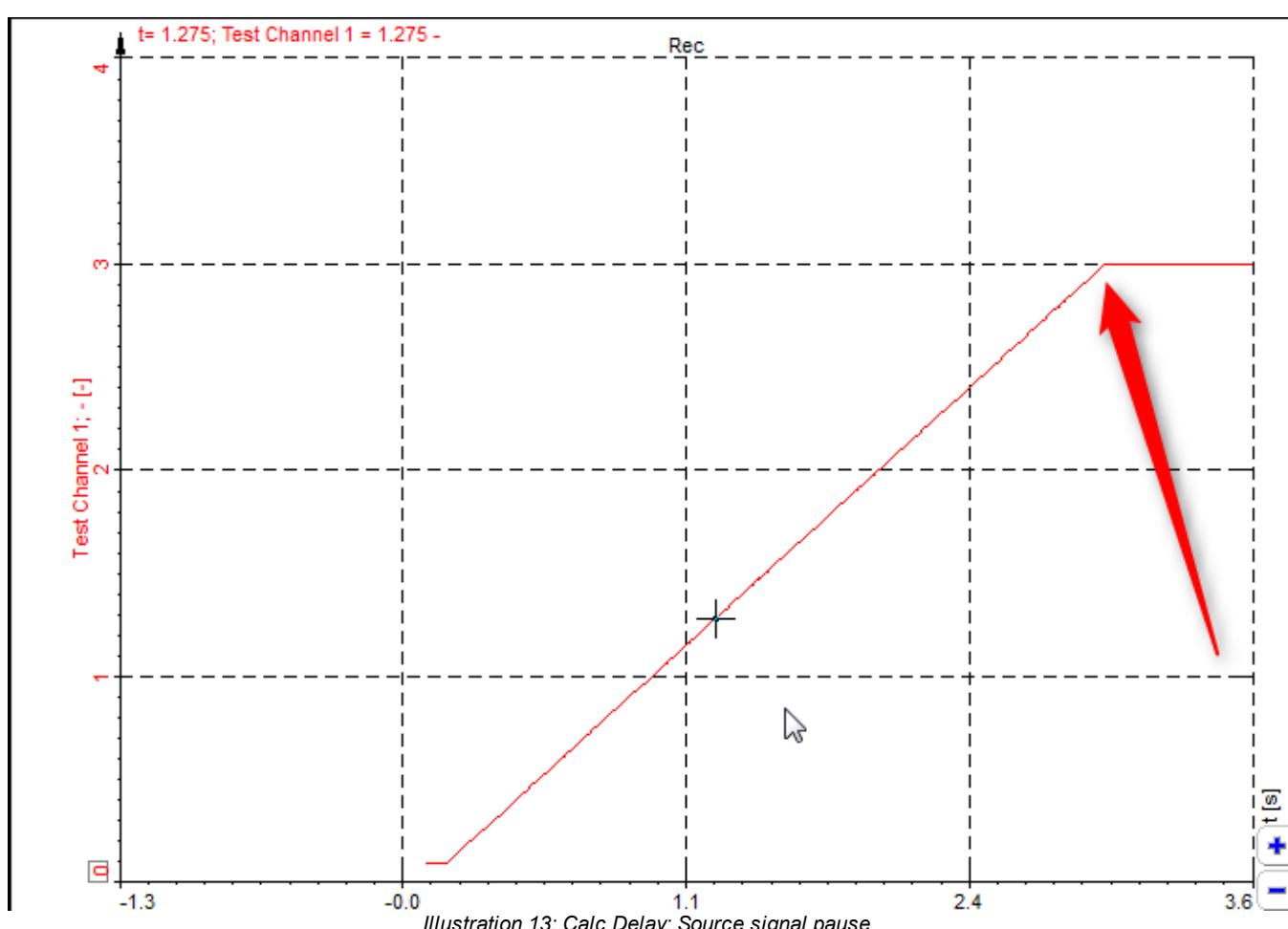


Illustration 13: Calc Delay: Source signal pause

When we zoom in on the region where the pause begins, you can clearly see that there are no data-points after the 3 second limit (the recorder just shows a flat line with the value of the last data point):

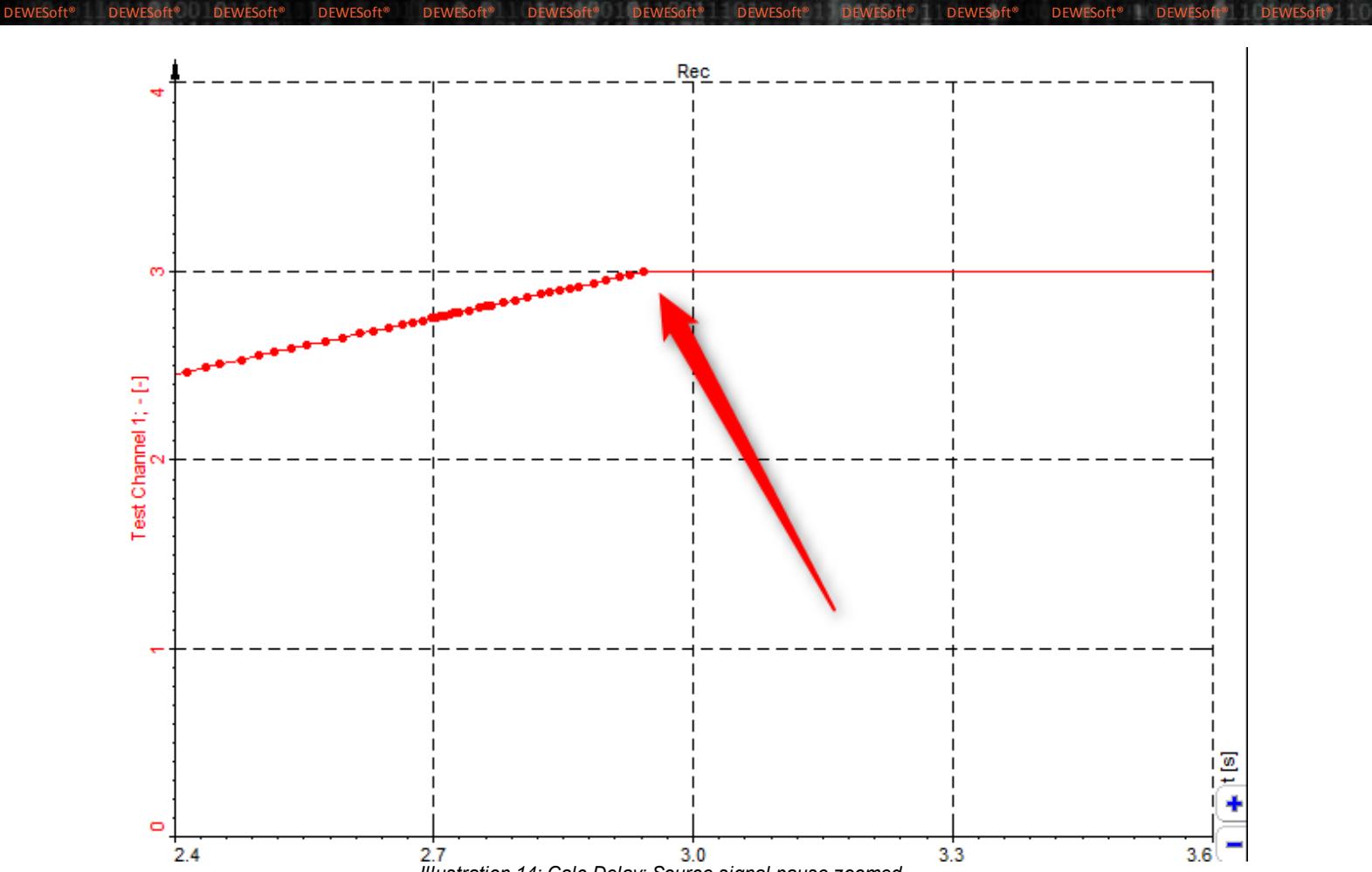
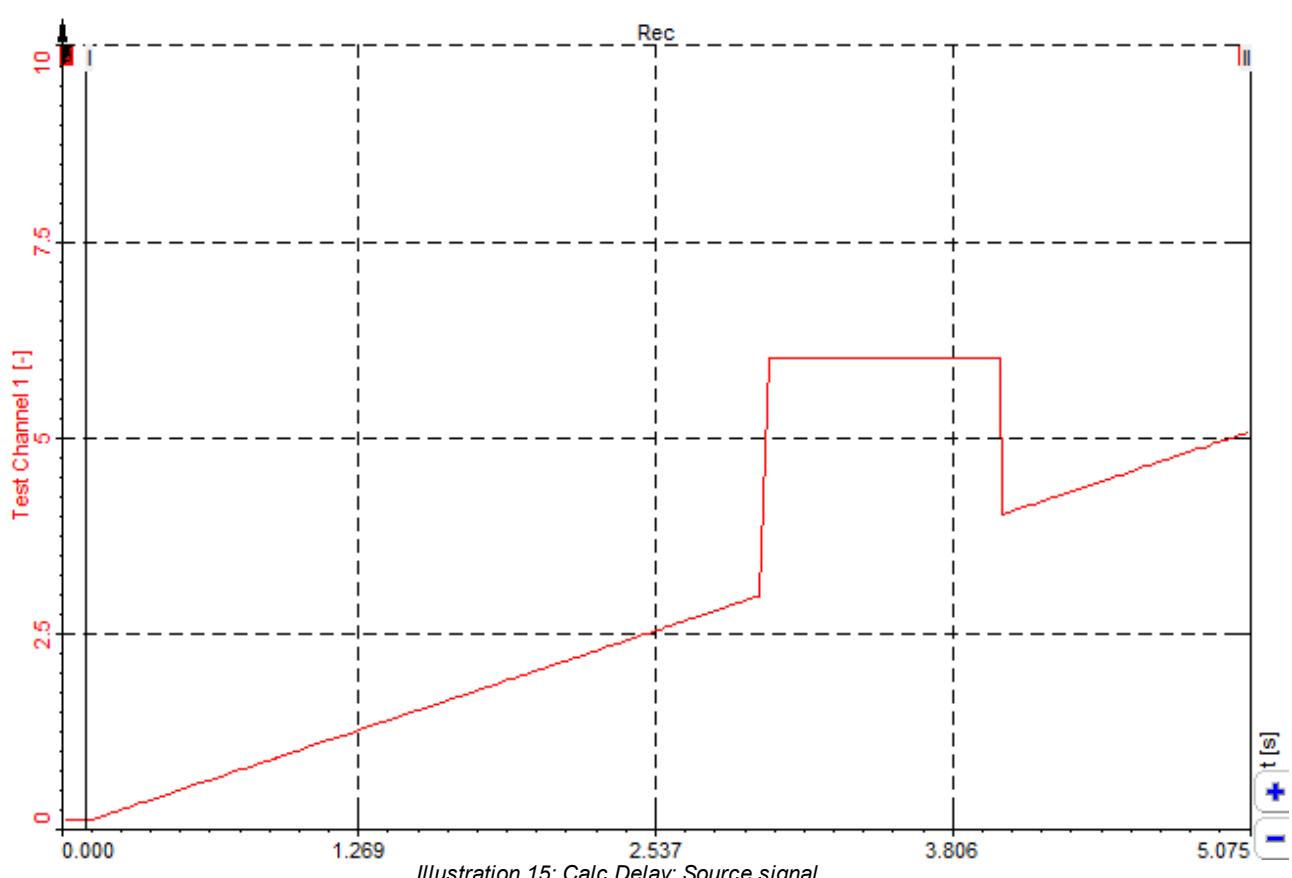


Illustration 14: Calc Delay: Source signal pause zoomed

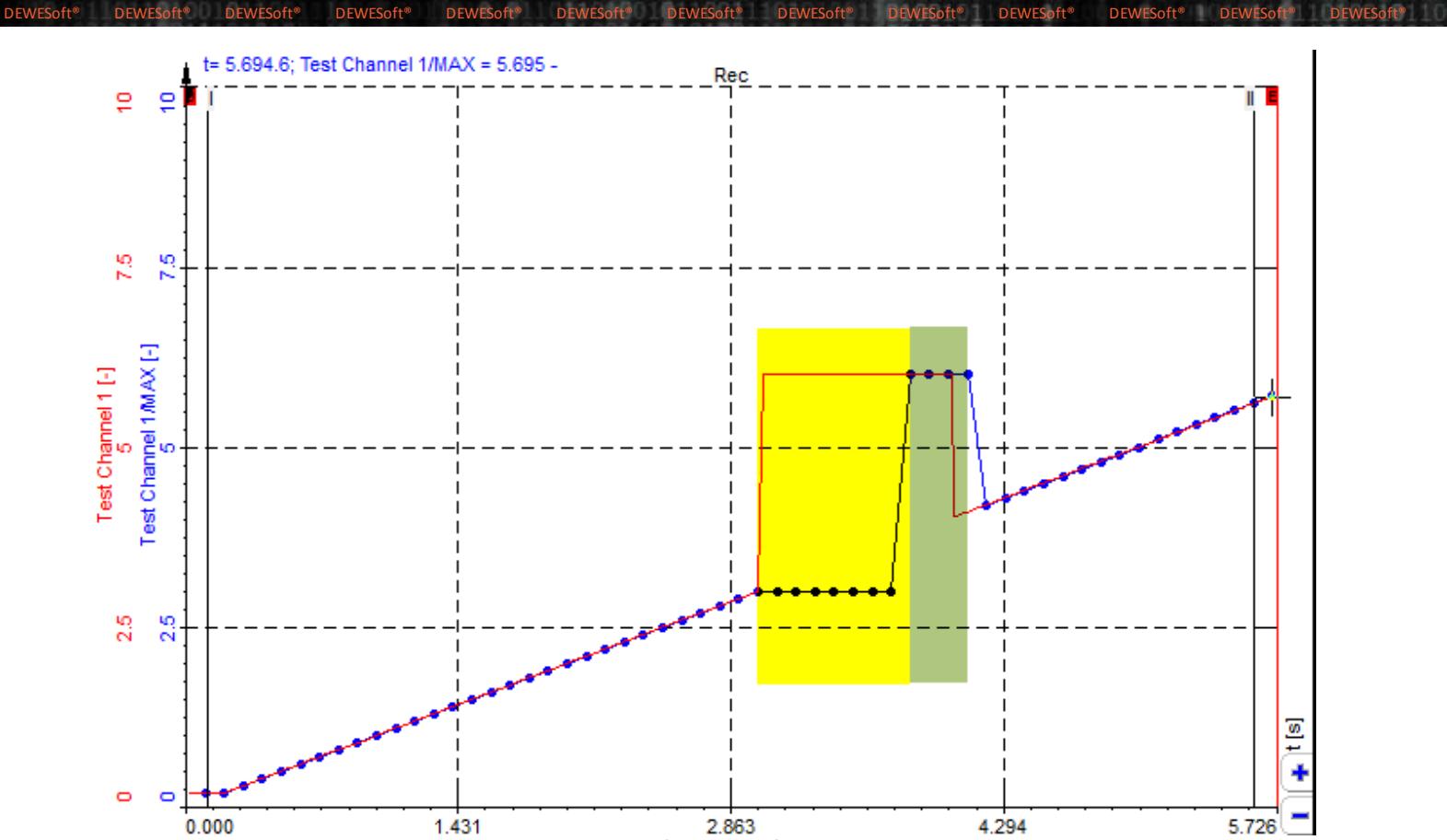
After 4 seconds it will fill up the data of the last second with 100 data-points of the fixed value 6, and then continue to insert the current measurement time. The complete source signal will now look like this:



Note, that the data between the 3rd and 4th second is now at value 6, because the missing data-points have been added later.

Math channel

Now we add a simple basic statistics Math channel that will calculate the *Maximum* of our source channel with a block-size of 0.1 seconds and we start storing the signal. The final data will look like this:



You can see that the *Maximum* channel (the blue signal) is wrong in the yellow area and then shows the correct values again in the green area. The time-span of the green area is exactly the calculation delay (which is about 200ms per default). Detailed explanation:

When the values in the yellow area have been created (live during measurement), the data of the source channel was not available, and thus the last valid data (the data points up to the first 3 seconds - where the maximum value was 3) had to be used for the calculation: and thus the calculated maximum in the yellow area shows a value of 3.

The first point of the blue statistics channel (which is at the time-instant 3.8 seconds) has been calculated 4 seconds after the start of the measurement and at this time the plugin has already filled up the missing data between the seconds 3 and 4: that's why the calculated values in the green area are correct.

If the calculation delay were 0 (i.e. we calculate the statistics right away), all values in between 3 and 4 seconds would be wrong (i.e. the green area would not exist and be part of the yellow area).

Recalculate

Since the raw data in the data file is correct we can just recalculate the mathematics in Analyse mode. e.g. open the channel setup of the statistics channel and also activate the *Minimum* calculation (in addition to the *Maximum* calculation) and then press the *Recalculate* button in *Review* mode. Now you can see that the calculated values are correct:

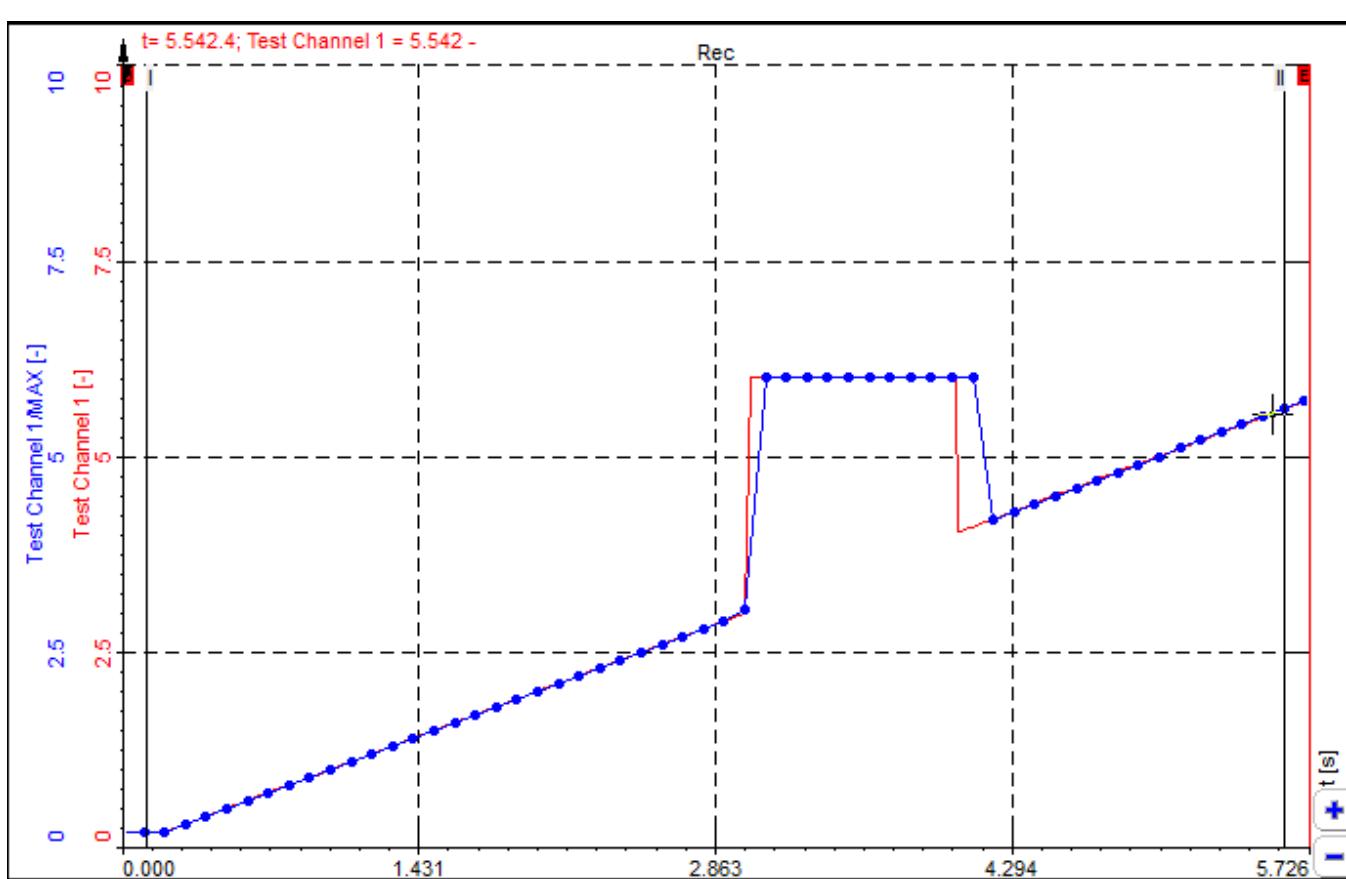


Illustration 17: Calc Delay: Recalculated

Plugin Set Maximum Calculation Delay

If your plugin expects such data pauses to be larger than the default 200ms (e.g. because your data source is slow or you have a bad connection to it) you should tell DEWEsoft® about it, so that the buffer size of the channels will be increased (to hold more data for longer calculation delay values):

Just return the calculation delay in seconds when the event `evGetMaxCalcDelay` is called: see [IPlugin4.OnEvent](#).

1.3.9.2 GUID

A globally unique identifier (*GUID*) is a practically unique identifier in computer software. The term GUID in this manual refers to the Microsoft implementation of the Universally unique identifier ([UUID](#)) standard.

Since DEWEsoft® add-ons are COM-components, they must be identified by *GUID*.

see also: [UUID](#), [Finding Plugins](#)

1.3.9.3 nil (aka. NULL, nothing)

`nil` is used in Delphi (and some other computer programming languages) to indicate the *null pointer*.

In other programming languages this may also be called `NULL` or `nothing`.

do not confuse with [Null](#) (mixed case).

1.3.9.4 Null

a special variable of type [Variant](#) which has an undefined value.

Do not confuse with *NULL* (all uppercase - see [nil](#))

1.3.9.5 UUID

A universally unique identifier (*UUID*) is an identifier standard used in software construction, standardized by the Open Software Foundation (*OSF*).

It is a 16-byte (128-bit) number which is practically unique in time and space which is represented by 32 hexadecimal digits; e.g.

3A5AEBF0-AA45-4249-A00C-038DF584C3F3

One widespread use of this standard is in Microsoft's globally unique identifiers ([GUIDs](#)) which are also used in DCOM programming.

see also: [GUID](#)

1.3.9.6 Variant

The *Variant* data type provides a flexible general purpose data type. It can hold different other data types (but not structured data and pointers).

see also: [Null](#)

1.3.10 Data Files

DEWEsoft® stores the measurement data in *DEWEsoft®* datafiles (file-extension: `.d7d`).

If you want to access the data from other applications, then you have these choices:

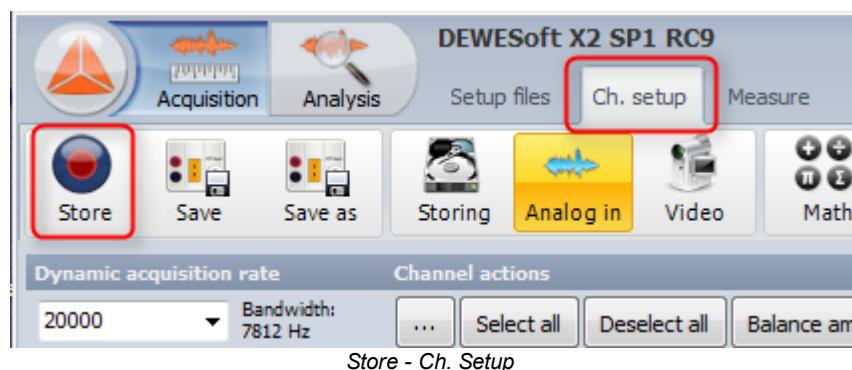
- you can load the data file in *DEWEsoft®* and export the data
- or you can use the `DWDataReaderLib.dll` to read the data-file directly
you can download the `DWDataReaderLib.dll` from our homepage: first log in, then go to <http://www.dewesoft.com/dewesoft7/developer-downloads> and click on the *Developers* item at the left - then search for *API Library*.

1.3.11 Start Events

There are some common pitfalls related to the start-events: [IPlugin2.OnStartAcq](#), [IPlugin2.OnStartStoring](#)

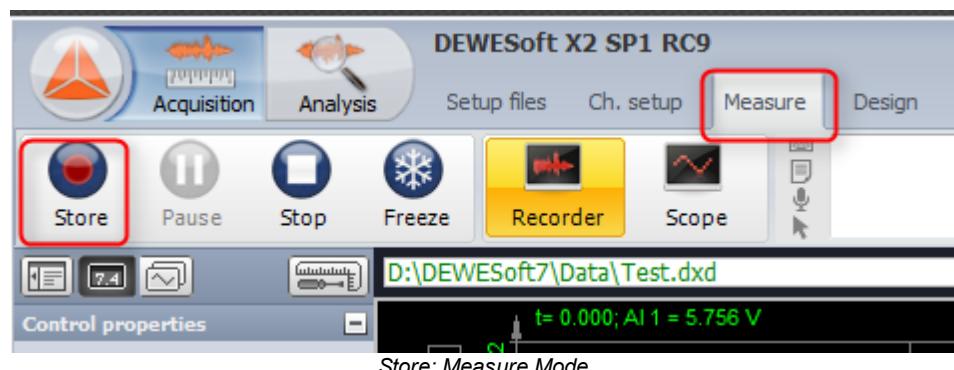
Note, that the order in which the events are called is NOT fixed. It does make a difference if you press the *Store* button on the *Ch. Setup* or in *Measure Mode*.

Store button in *Ch. Setup Mode*:



In this case [IPlugin2.OnStartStoring](#) is called before [IPlugin2.OnStartAcq](#). From now on [IPlugin2.OnGetData](#) will be called regularly.

Store button in *Measure Mode*:



In this case [IPlugin2.OnStartAcq](#) is called when the user clicks on *Measure* (to go from *Ch. Setup* to *Measure Mode*). Now on [IPlugin2.OnGetData](#) will already be called regularly. When the user then clicks the Store button [IPlugin2.OnStartStoring](#) is called.

Notes that [IData.StartStoreTimeUTC](#) and [IMasterClock.GetCurrentTime](#) should only be called in [IPlugin2.OnGetData](#). The following table shows example data for the results of these calls:

Notes: DEWEsoft® X1 was used for the test and the Dynamic Acquisition rate in the example is 5 kHz.

Example 1: no Timing Device

When the *Store* button in *Ch. Setup Mode* is used:

Event	IMasterClock.GetCurrentTime	IData_StartStoreTimeUTC	IData_GetSamplesAcquired	Time related to IData_GetSamplesAcquired
IPlugin2.OnStartStoring	0.00	-1	66101	13.22
IPlugin2.OnStartAcq	0.00	-1	0	0.00
IPlugin2.OnGetData	0.18	2015.03.03 12:30:07.865	889	0.18
IPlugin2.OnStopAcq	3.67	2015.03.03 12:30:07.865	18102	3.62
IPlugin2.OnStopStoring	0.00	2015.03.03 12:30:07.865	18102	3.62

When the *Store* button in *Measure Mode* is used:

Event	IMasterClock.GetCurrentTime	IData_StartStoreTimeUTC	IData_GetSamplesAcquired	Time related to IData_GetSamplesAcquired
IPlugin2.OnStartAcq	0.08	-1	0	0.00
IPlugin2.OnGetData	0.23	2015.03.03 12:25:16.815	1134	0.23
IPlugin2.OnStartStoring	2.95	2015.03.03 12:25:16.815	14556	2.91
IPlugin2.OnGetData	0.08	2015.03.03 12:25:19.645	418	0.08
IPlugin2.OnStopAcq	1.82	2015.03.03 12:25:19.645	8910	1.78
IPlugin2.OnStopStoring	0.00	2015.03.03 12:25:19.645	8910	1.78

Example 2: Timing Device (GPS)

When the *Store* button in *Ch. Setup Mode* is used:

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Event	IMasterClock.GetCurrentTime		IData.StartStoreTimeUTC		IData.GetSamplesAcquired		Time related to IData.GetSamplesAcquired					
IPlugin2.OnStartStoring		0.00	-1			116989		23.40				
IPlugin2.OnStartAcq		0.00	-1			0		0.00				
IPlugin2.OnGetData		0.32	2015.03.03 12:11:34.000			1525		0.30				
IPlugin2.OnStopAcq		4.28	2015.03.03 12:11:34.000			20917		4.18				
IPlugin2.OnStopStoring		0.00	2015.03.03 12:11:34.000			20917		4.18				

When the **Store** button in **Measure Mode** is used:

Event	IMasterClock.GetCurrentTime		IData.StartStoreTimeUTC		IData.GetSamplesAcquired		Time related to IData.GetSamplesAcquired					
IPlugin2.OnStartAcq		0.00	-1			0		0.00				
IPlugin2.OnGetData		0.31	2015.03.03 12:02:50.000			1525		0.30				
IPlugin2.OnStopAcq		5.56	2015.03.03 12:02:50.000			27253		5.45				
IPlugin2.OnStartStoring		0.00	-1			27253		5.45				
IPlugin2.OnStartAcq		0.00	-1			0		0.00				
IPlugin2.OnGetData		0.21	2015.03.03 12:02:58.000			949		0.19				
IPlugin2.OnStopAcq		5.80	2015.03.03 12:02:58.000			28597		5.72				
IPlugin2.OnStopStoring		0.00	2015.03.03 12:02:58.000			28597		5.72				

1.4 What's new in DEWESoft 7

This is a list of the major changes/ new features from *DEWESoft® 6* to *DEWESoft® 7*:

- *DEWESoft® 7* now supports several levels of intermediate buffers: see [The Buffer Structure](#)

- [IApp.Modules](#) is now deprecated and has been replaced by [IApp.AmplInterfaces](#)

If you have an existing automation application that worked with *DEWESoft® 6*, we can give you a Delphi tool that makes it easier for you to upgrade. The tool tries to keep as much compatible as possible with the old *DEWESoft® 6* [IApp.Modules](#). Please contact our support to get the tool and documentation.

- The channel setup is now stored in an XML file: [XML Setup](#)

- *DEWESoft® 7* supports new channel types:

- textual channels: [Textual Channels](#)

- control channels: [Control Channels](#)

- array channels: [Array Channels](#)

- The channel index structure has been expanded: [Channel Index](#)

- In *DEWESoft® 7* you can do offline mathematical calculations: [IOfflineCalc](#)

- Plugins: *DEWESoft® 7* has introduced a new plugin interface [IPlugin4](#) that can be expanded with arbitrary events in future versions. Whenever a plugin wants to use one of these new events it must only listen to this new event and not implement a new plugin interface (i.e. there will be no [IPlugin5](#) interface)

- new interface vor visual controls and math add-ons: [IDewePlugin.OnMessage](#)

- scaling with double precision is now possible: see [IChannel.ScaleValueDouble](#), [IChannel.Scale](#)

- multipass calculations are now supported :in the first run, the average is calculated which is needed by other formulas in the second run: see also [MathMultipassType](#)

- You can now get the timestamp of a video frame: [IVideoLoadEngine.GetFramesInfo](#)

- [IPluginChannelXMLHelper](#): convenience class for easy access of plugin channels

- support for input groups: [IInputGroup](#)

via [IInputGroup.Properties](#) the plugin can pass arbitrary name-value pairs to visual controls

1.5 Plugin Examples

This section shows simple examples of custom *DEWEsoft®* Plugins in different programming languages.

All examples will only do the minimal steps required to create a plugin, register it, so that it can be used in *DEWEsoft®* and add one single channel with some random data.

- [Delphi](#)
- [Visual Basic](#)
- [Visual C++](#)

1.5.1 Finding Plugins

Describes how *DEWEsoft®* finds plugins.

DEWEsoft® searches for plugins in the Windows registry, under the following key:

[HKEY_LOCAL_MACHINE\SOFTWARE\Devesoft\Plugins\

Note: 64bit Windows systems will automatically insert the `Wow6432Node` node (highlighted in blue below):

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Devesoft\Plugins\

The most important key is the [GUID](#), because it will be used to locate the COM object on the system:

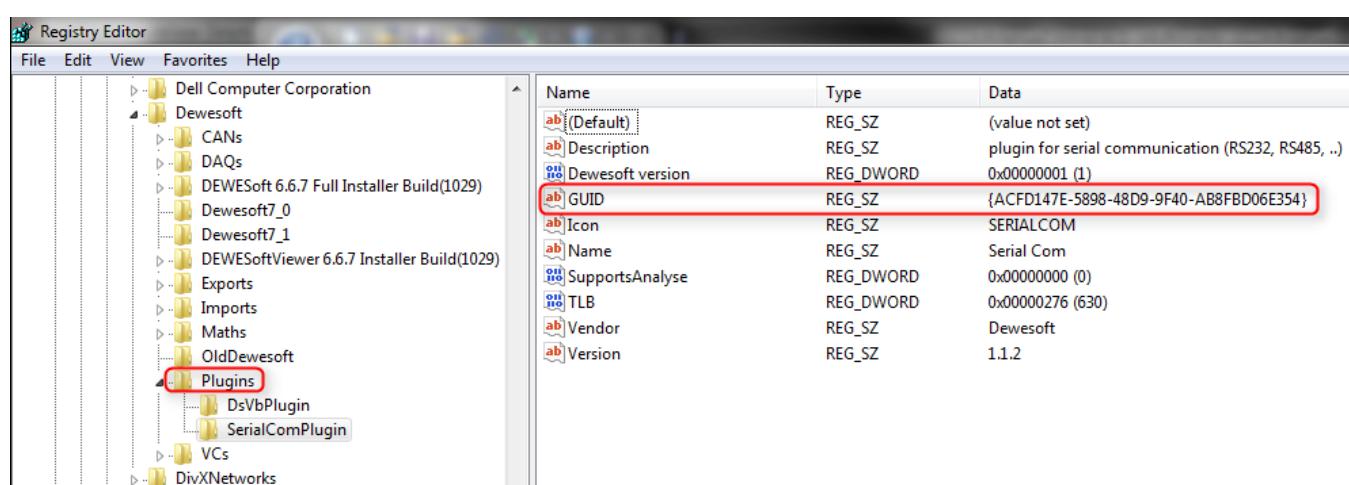


Illustration 18: *DEWEsoft™* Plugin Registry Entries

see also:

- [Delphi: Register Plugin](#)
- [Visual Basic: Register Assembly](#)

- [Visual C++: Register Plugin](#)
- [MSDN - CLSID Key](#)
- [MSDN - Registering COM Applications](#)

1.5.2 Delphi

This example shows how to build a simple plugin using the CodeGear™ Delphi® 2007 for Win32® IDE:

- [Delphi: Prepare Project](#)
- [Delphi: Implementation](#)
- [Delphi: Prepare for DCOM](#)
- [Delphi: Register Plugin](#)
- [Delphi: Test the Plugin](#)
- [Delphi: Sourcecode](#)

1.5.2.1 Delphi: Prepare Project

Create ActiveX Library

To create a new *ActiveX Library* project, select *File - New - Other...*:

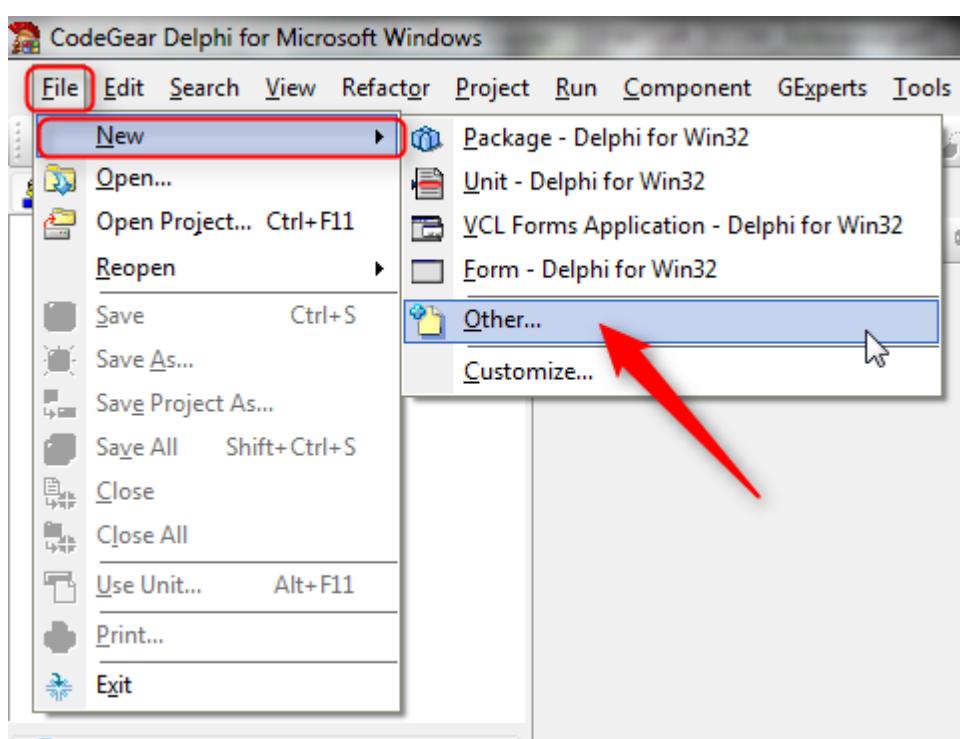


Illustration 19: Create Other...

Then select *ActiveX Library*:

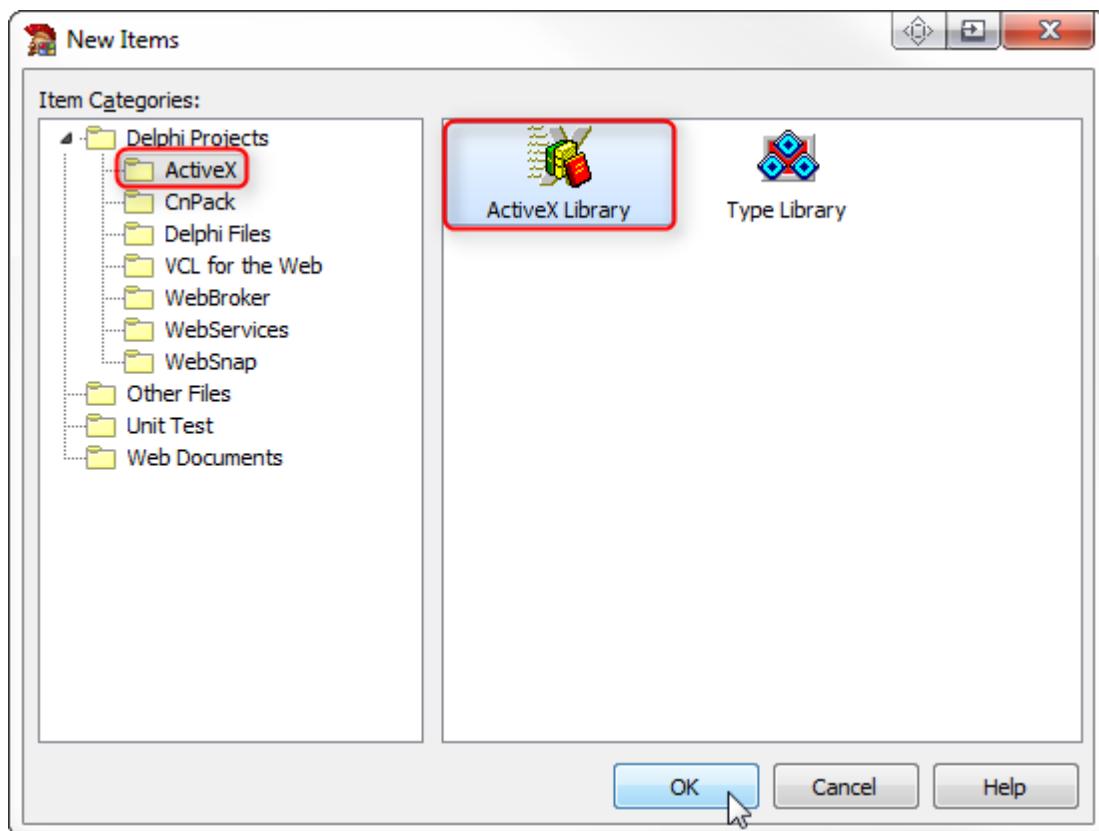


Illustration 20: ActiveX Library

Now we can save the new Delphi project (select *File - Save Project As...*):

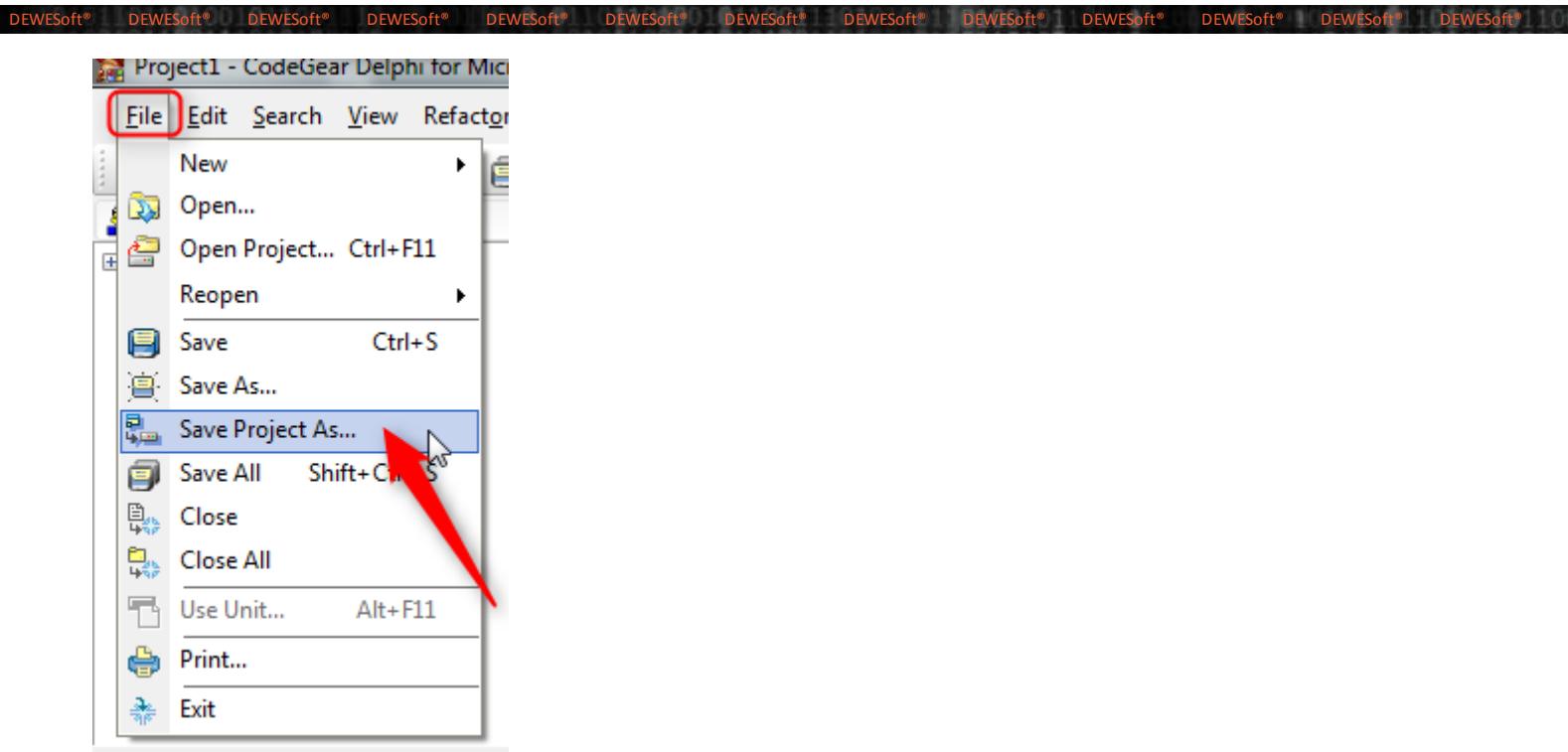


Illustration 21: Save Project as

and enter a meaningful name:

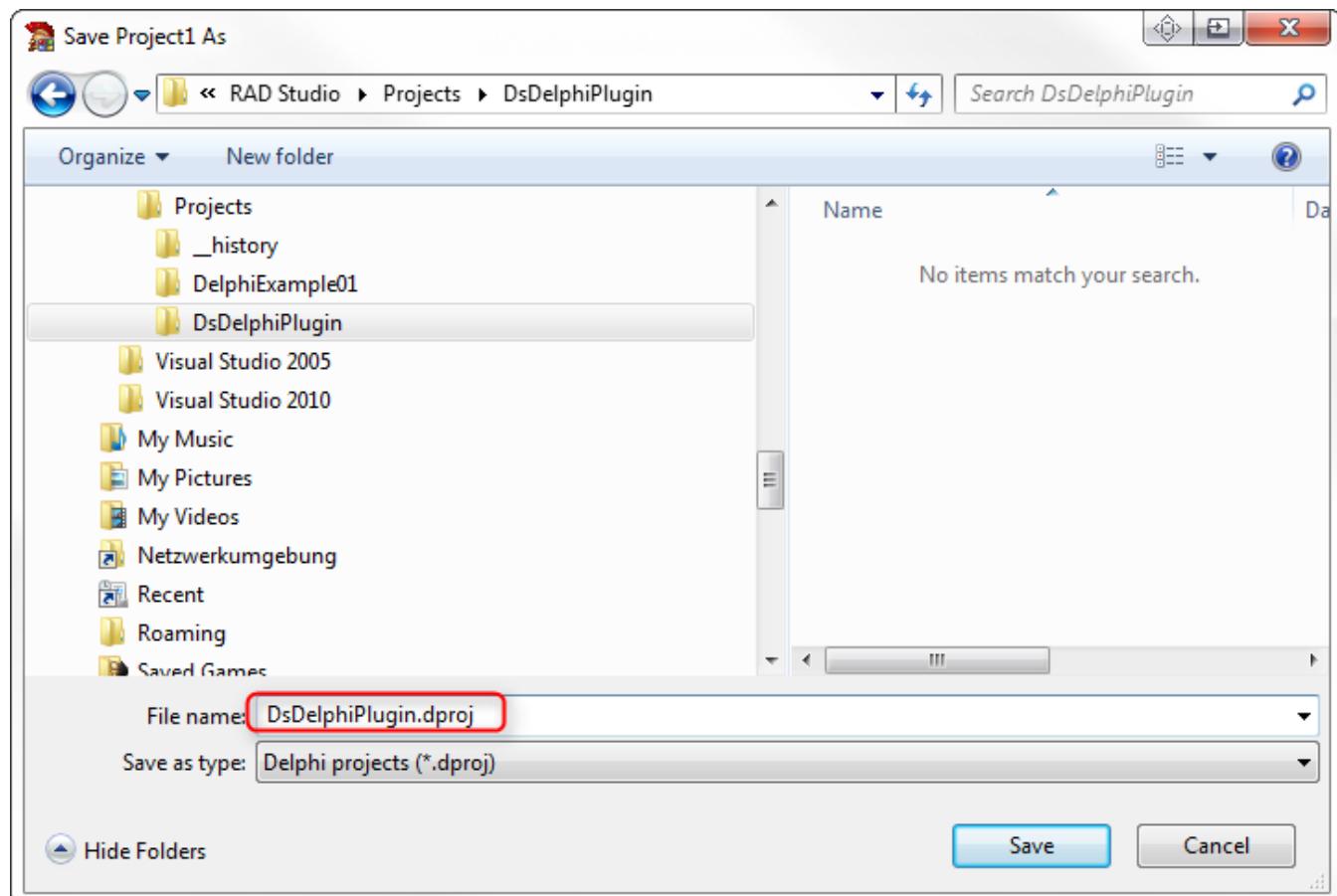


Illustration 22: Project Name

Create Type Library

Next, we need to add a type library. Select *File - New - Other...*:

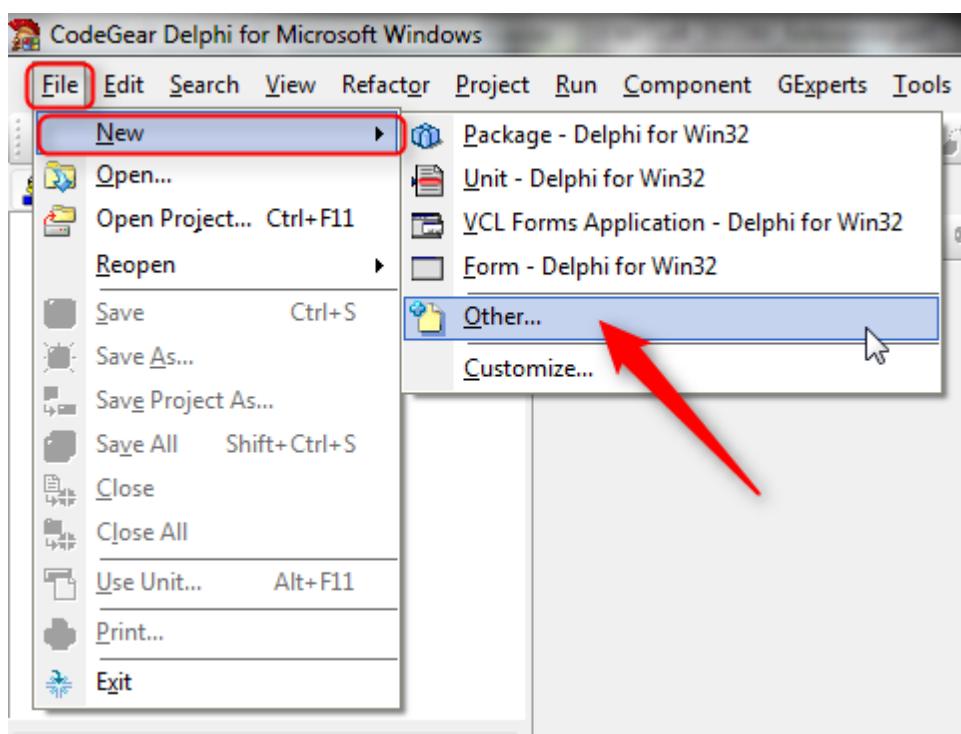


Illustration 23: Create Other...

and then select *Type Library*.

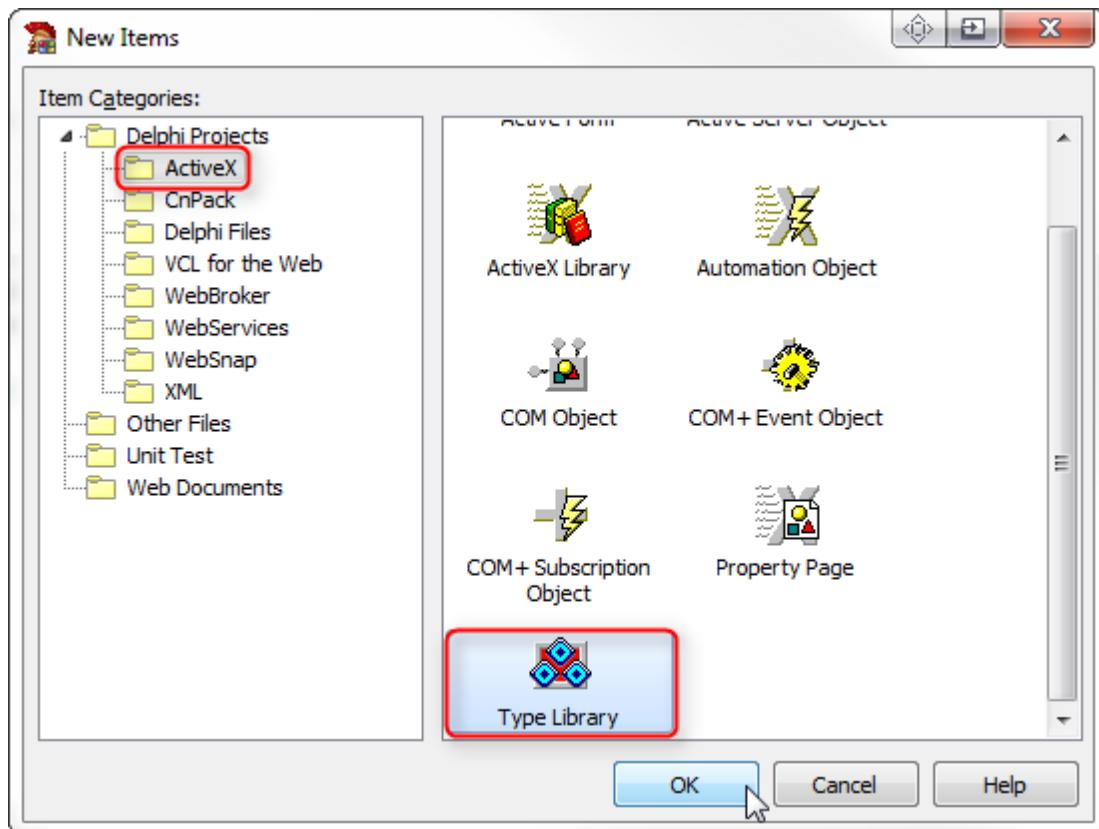


Illustration 24: Create Type Library

Per default, only the *OLE Automation* type library is used. We also need the *DEWEsoft®* type library. Right-click in the

main area of the *Uses* tab-sheet in the *Type library editor* window and select *Show All Type Libraries* from the pop-up menu.

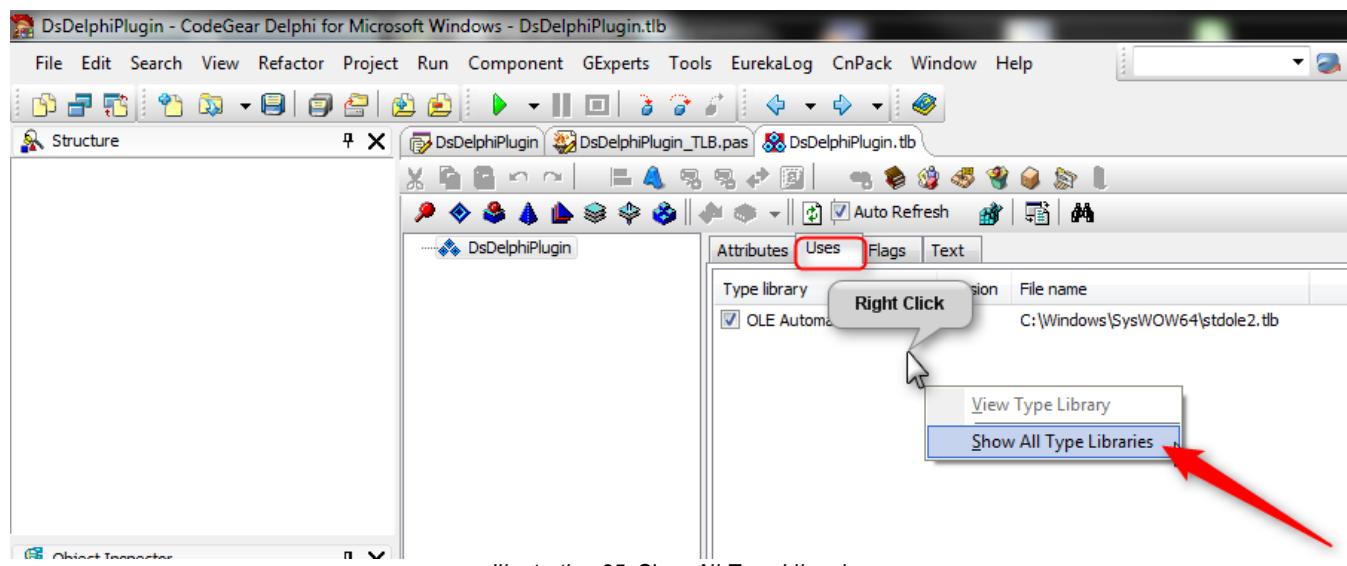


Illustration 25: Show All Type Libraries

then locate the *DEWEsoft Library* and check the box on the left to select it.

Note: the *DEWEsoft Library* will only show up if DEWEsoft® has been installed successfully on your system.

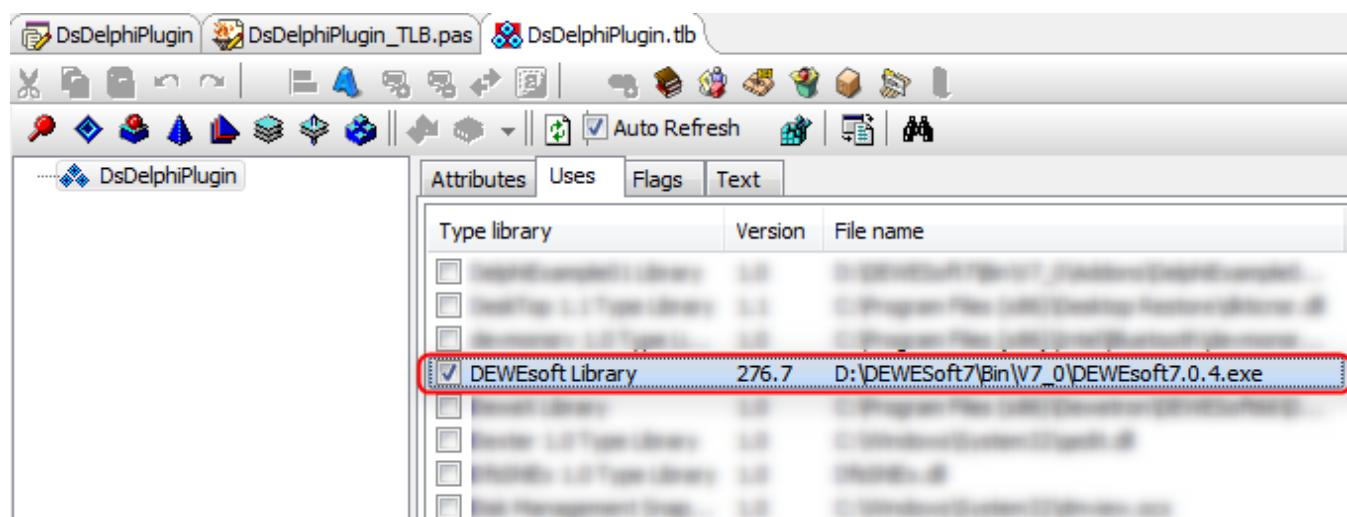


Illustration 26: DEWEsoft™ type library

Now you can right click again and click on *Show Selected* to hide the unused libraries:

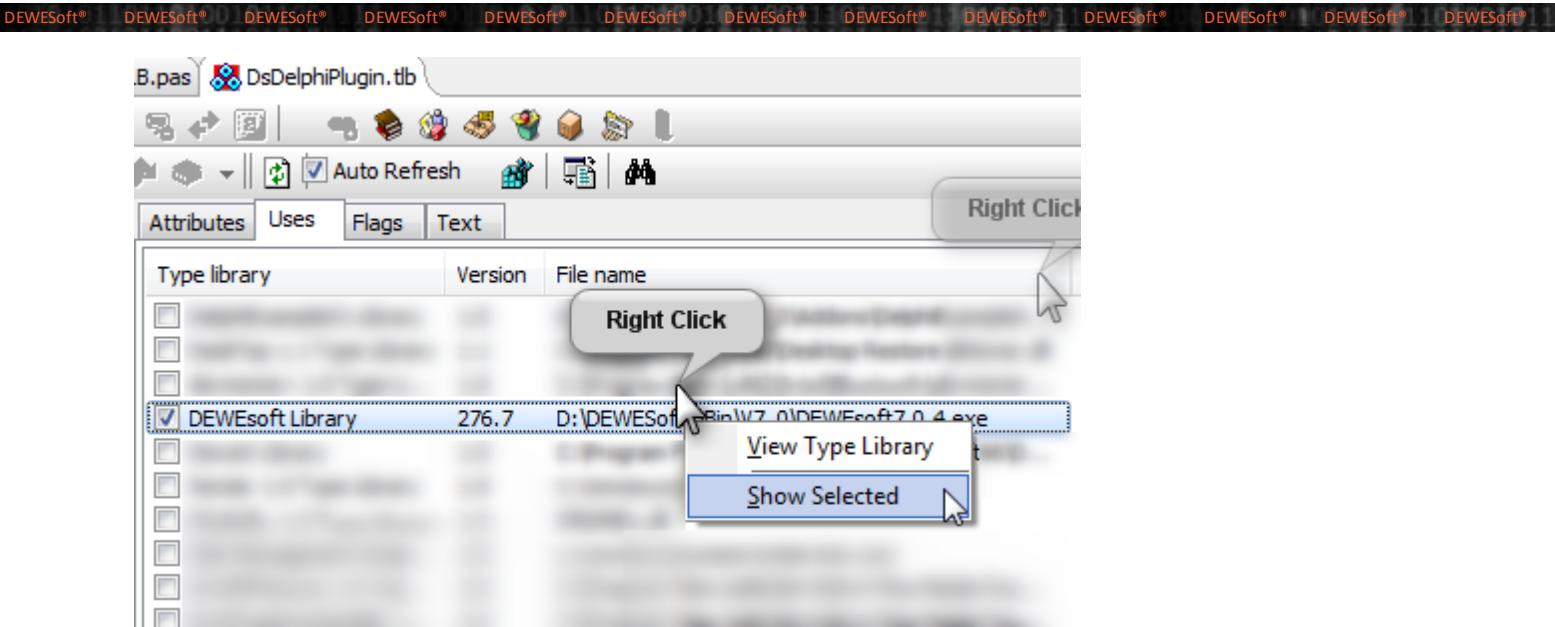


Illustration 27: Show selected

Only the *OLE Automation* and the *DEWESoft Library* should show up now:

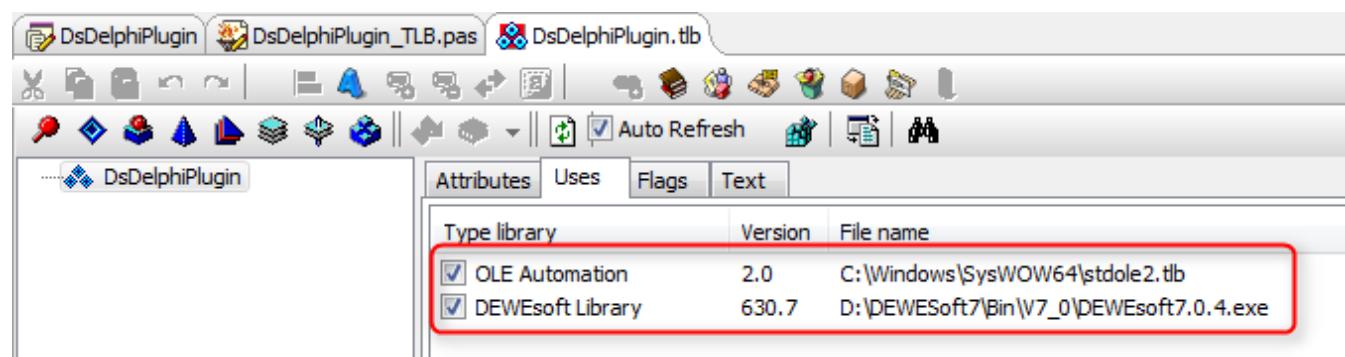


Illustration 28: Selected Type Libraries

Add new CoClass

Click the 3rd icon on the left of the *Type Library Editor* to add a new *CoClass* to our type library and rename it to *DelphiPlugin*:

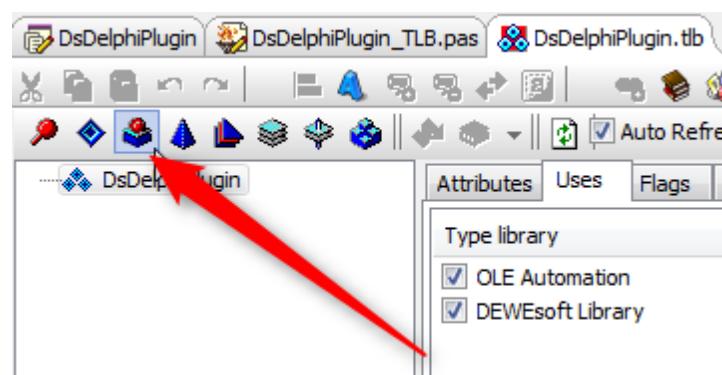


Illustration 29: New CoClass

Add IPlugin2 Interface

Open on the *Implements* tab-sheet and right-click in the main area. Select Insert Interface from the pop-up menu:

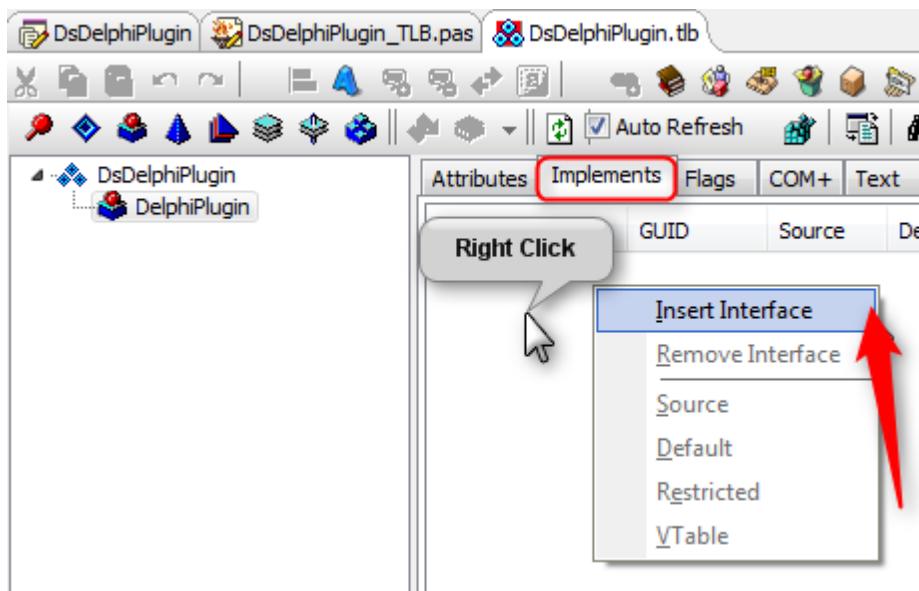


Illustration 30: Insert Interface

Select *IPlugin2* from the list of interfaces:

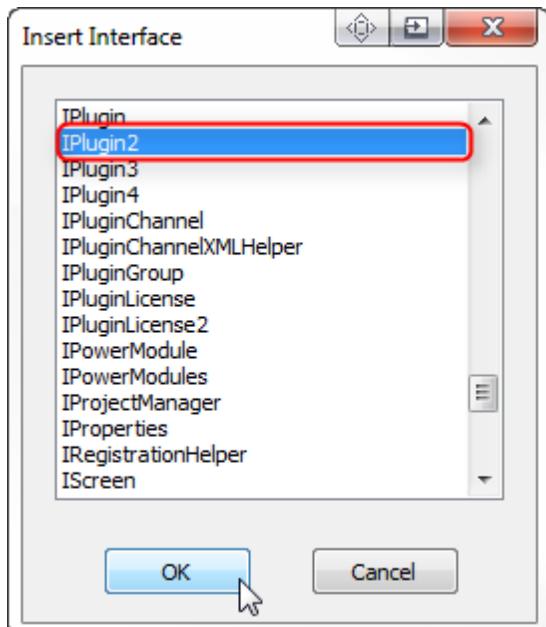


Illustration 31: Insert Interface IPlugin2

Click the *Refresh* button to make sure that the type library is up-to-date.

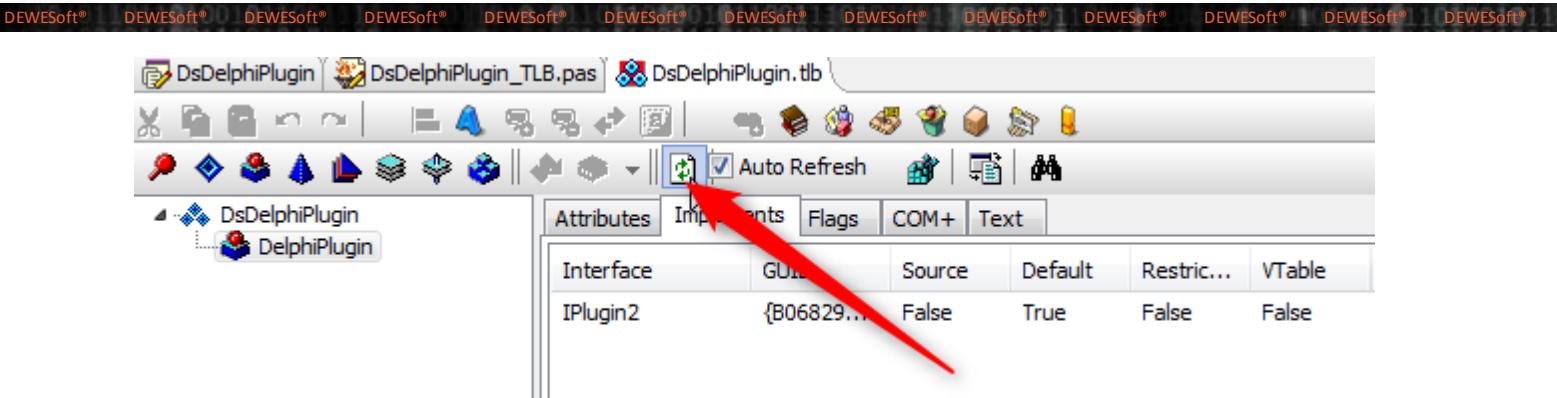


Illustration 32: Refresh

and click the Save all icon to make sure that all the work we did so far, is saved to disk:

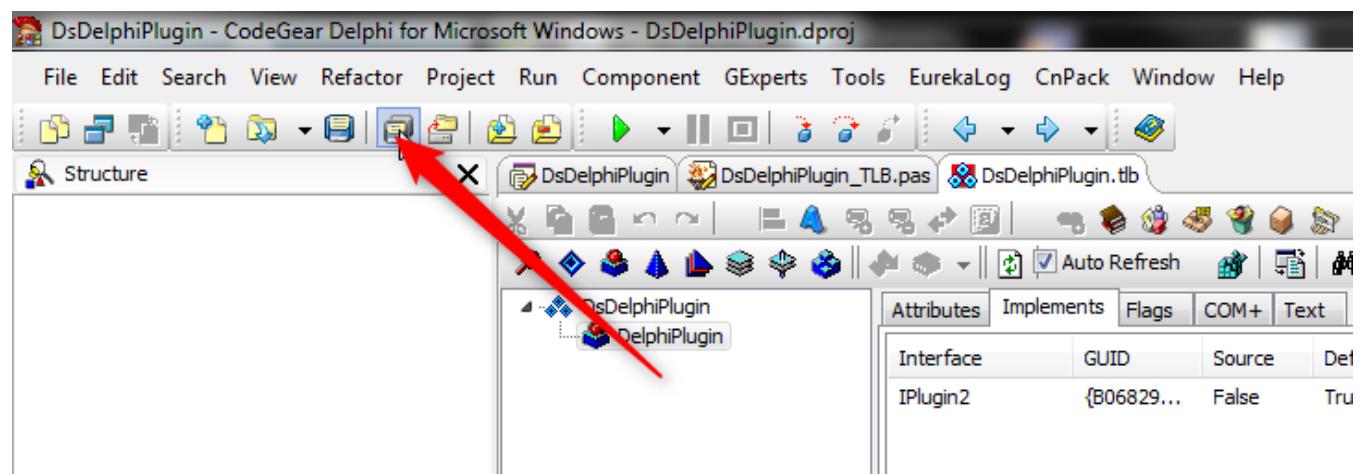


Illustration 33: Save all

Project Options

We will now adapt the project options so that we can easily start our plugin directly from the Delphi IDE.

Open [Project - Options...](#):

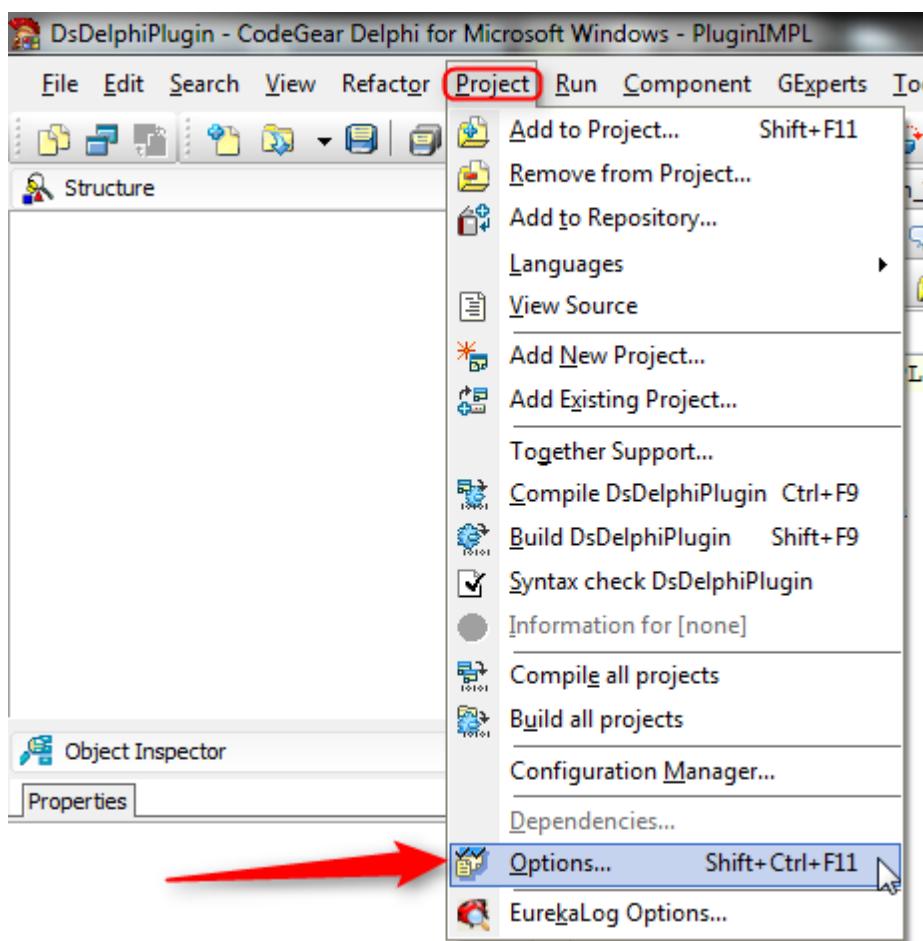


Illustration 34: Open Project Options

In the *Directories/Conditionals* section, set the *Output directory* to the Addons directory of your *DEWESoft®* installation, so that the .dll file will be directly created in the Addons directory of *DEWESoft®*.

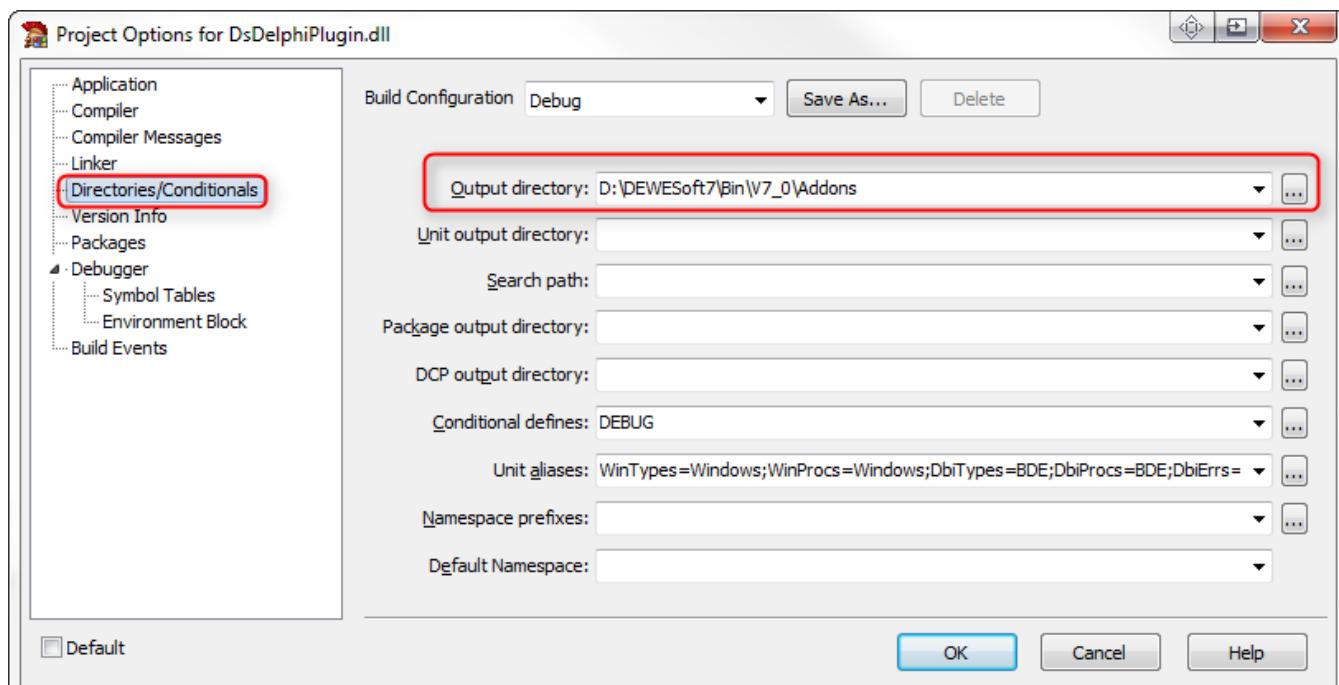


Illustration 35: Output directory

In the *Debugger* section, set the *Host application* to the DEWESoft® executable file, so that the IDE will start DEWESoft® automatically when you select *Run - Run* from the Delphi IDE menu.

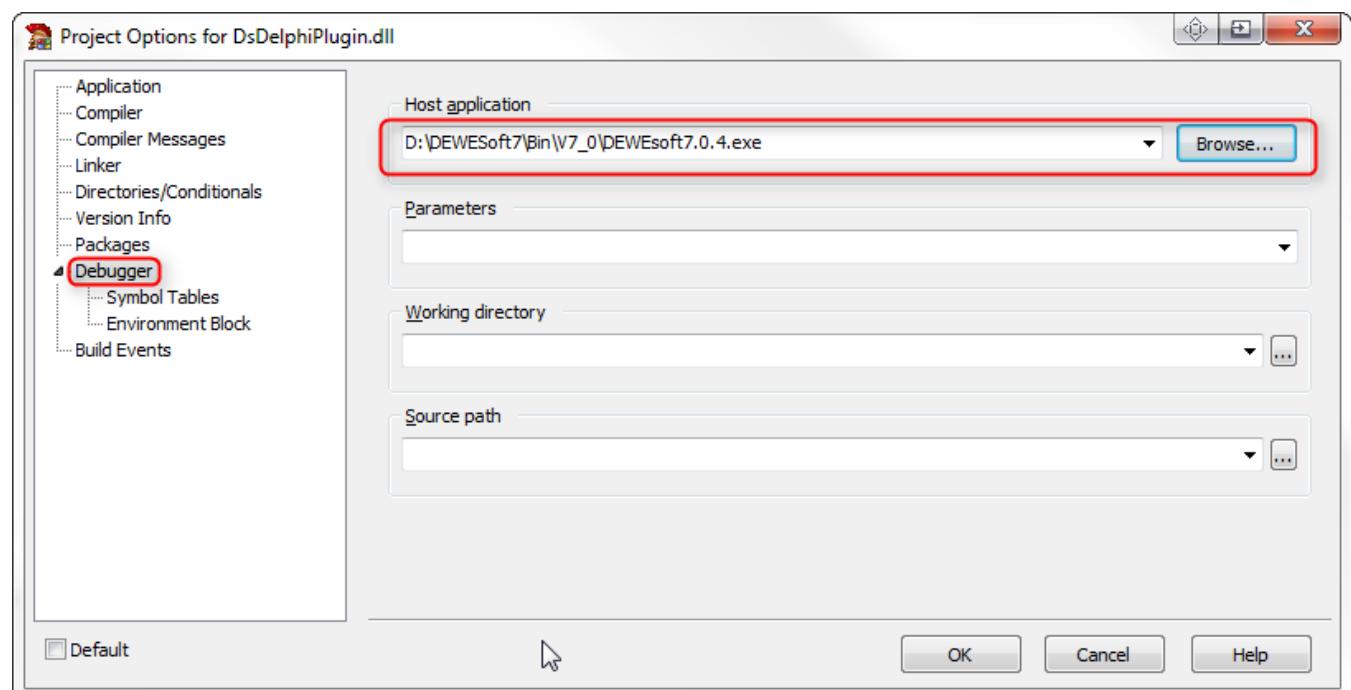


Illustration 36: Host Application

next: [Delphi: Implementation](#)

1.5.2.2 Delphi: Implementation

After you have prepared the Delphi project ([Delphi: Prepare Project](#)), we can start to implement the plugin class.

Create Delphi Unit for the Plugin

First we add a new *Unit*; select *File - New - Unit*:

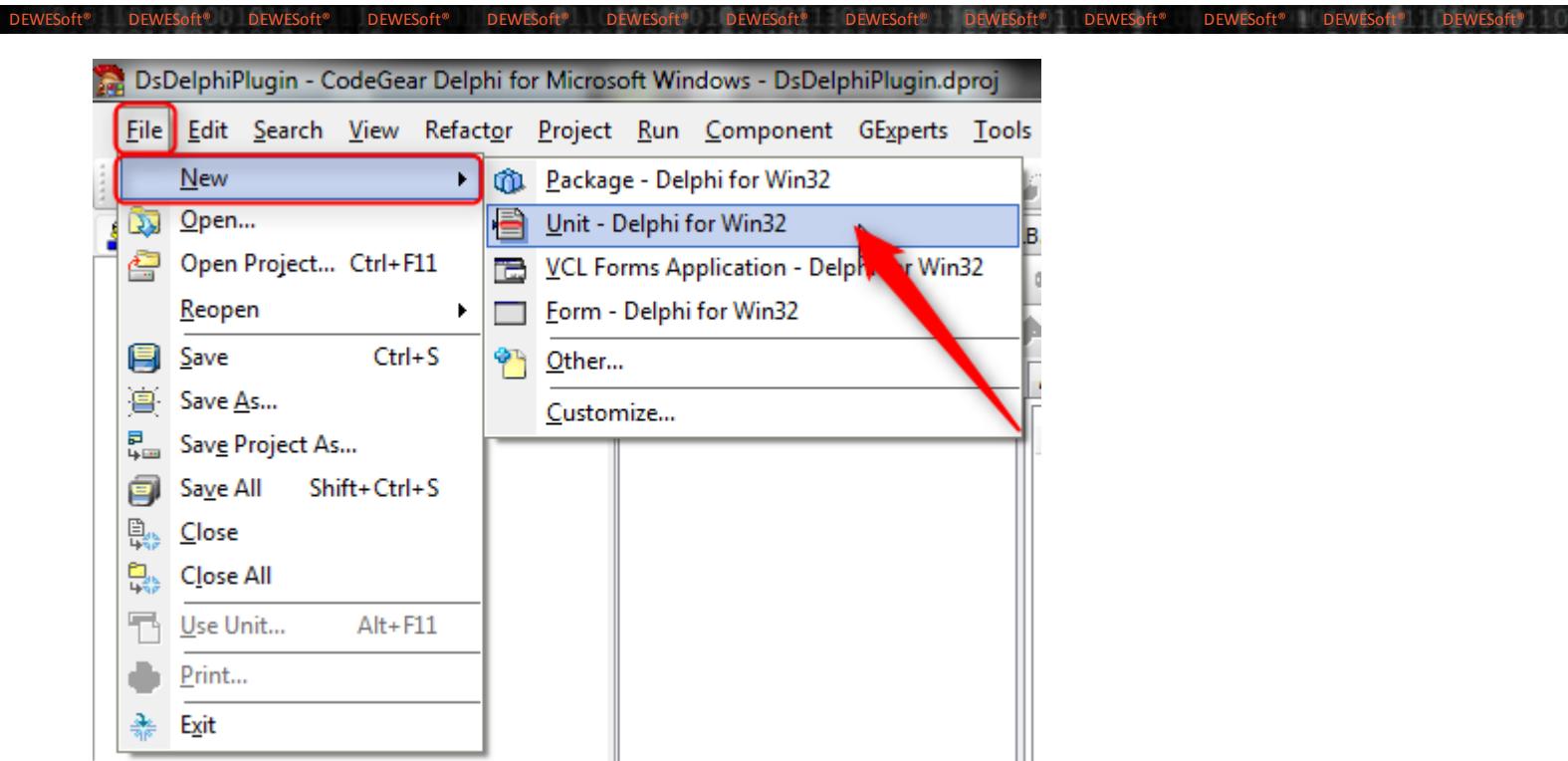


Illustration 37: New Unit

Enter a meaningful name for the *Unit*; e.g. `PluginIMPL`:

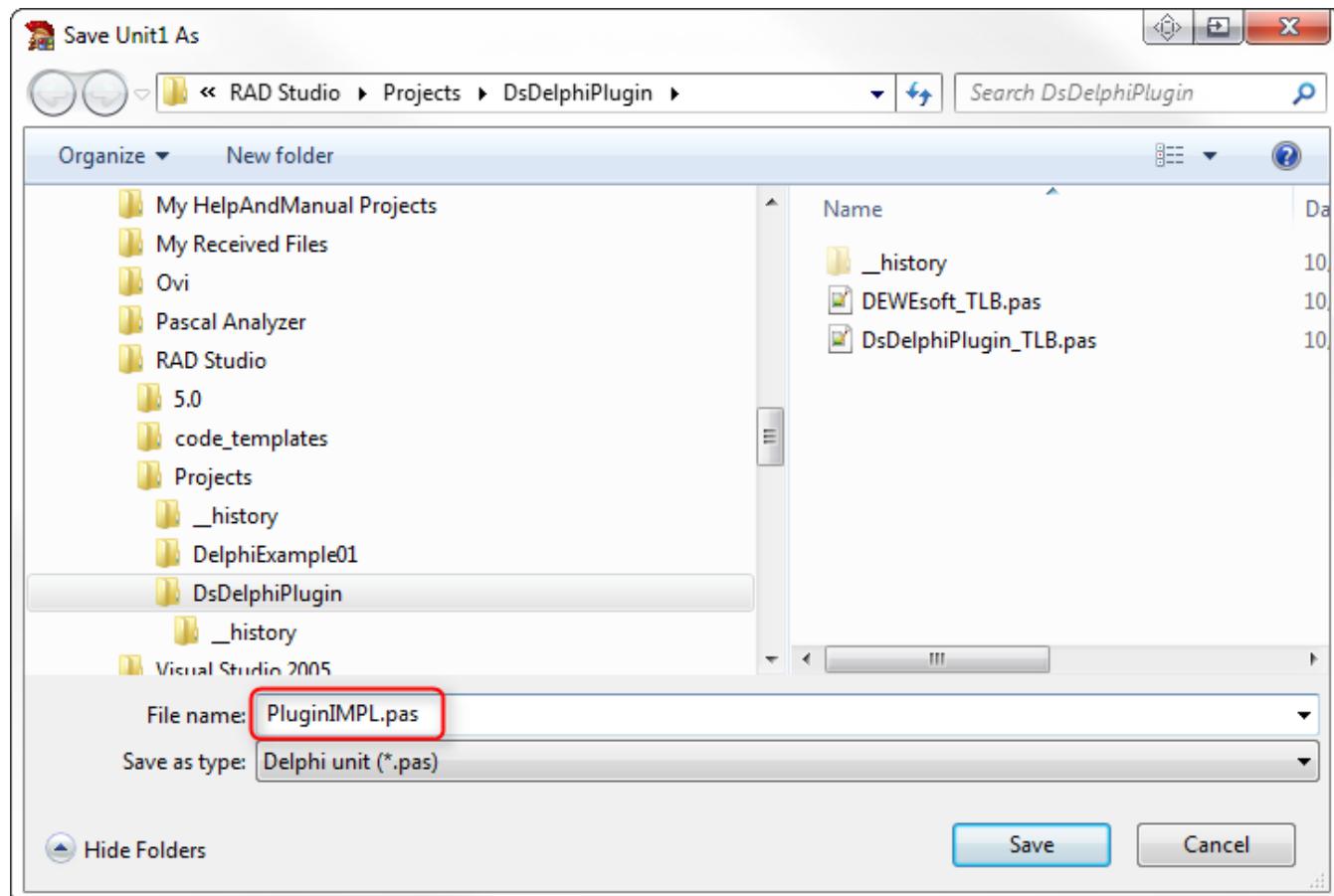


Illustration 38: Enter Unit Name

Add the Plugin class

We add a new *Class* definition to the *Unit*: we call it `TExamplePlugin` and the class derives from `TAutoObject` and implements the `IPlugin2` interface:

A screenshot of the Delphi IDE showing a unit named `PluginIMPL`. The code defines an interface with a type definition. The type definition includes `TExamplePlugin = class(TAutoObject, IPlugin2)`. Both `TAutoObject` and `IPlugin2` are underlined with red curly braces, indicating they are undeclared identifiers. A red rectangular box highlights this line of code.

```
1 unit PluginIMPL;
2
3 interface
4
5 type
6   TExamplePlugin = class(TAutoObject, IPlugin2)
7
8 end;
9
10 implementation
11
12 end.
```

Illustration 39: Type Definition

You can see that `TAutoObject` and `IPlugin2` have a red curly underline to indicate an error. When you hover the mouse over the tokens, you can see the error-message: *Undeclared identifier*. We just need to add the correct uses clause to the unit, so that the Delphi IDE can find the definitions of those tokens:

A screenshot of the Delphi IDE showing the same unit `PluginIMPL`. The `uses` clause has been added, containing `ComObj, DEWEsoft_TLB;`. This line is highlighted with a red rectangular box. The previously underlined identifiers now have no underlines, indicating they are now declared.

```
1 unit PluginIMPL;
2
3 interface
4
5 uses
6   ComObj, DEWEsoft_TLB;
7
8 type
9   TExamplePlugin = class(TAutoObject, IPlugin2)
10
11 end;
12
13 implementation
14
15 end.
```

Illustration 40: Interface Uses Clause

Now the red curly underlines are gone and we can try to compile the project; select *Project - Compile DsDelphiPlugin* (or press *Ctrl+F9*):

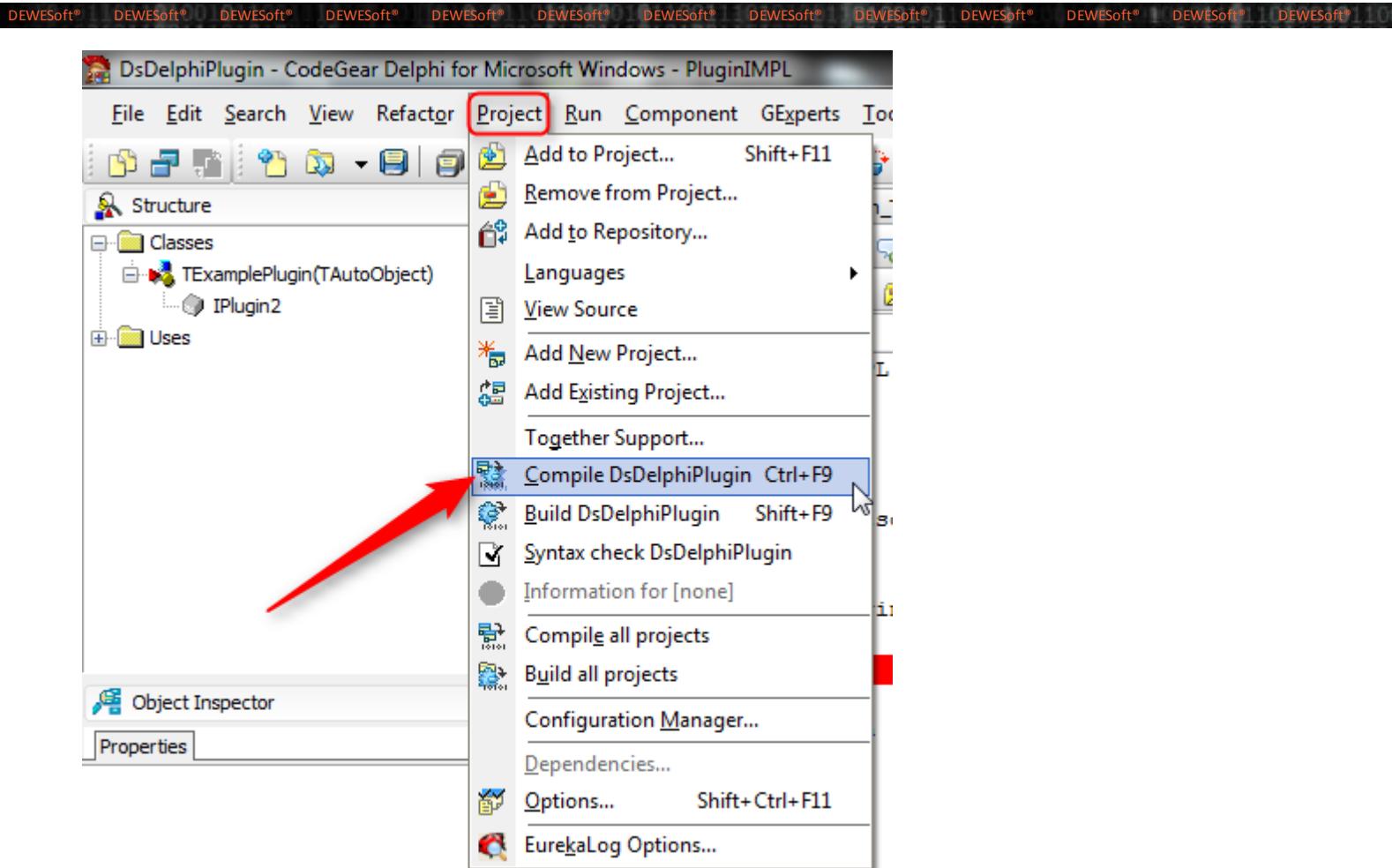


Illustration 41: Compile Project

You will see that there are a lot of errors regarding undeclared identifiers:

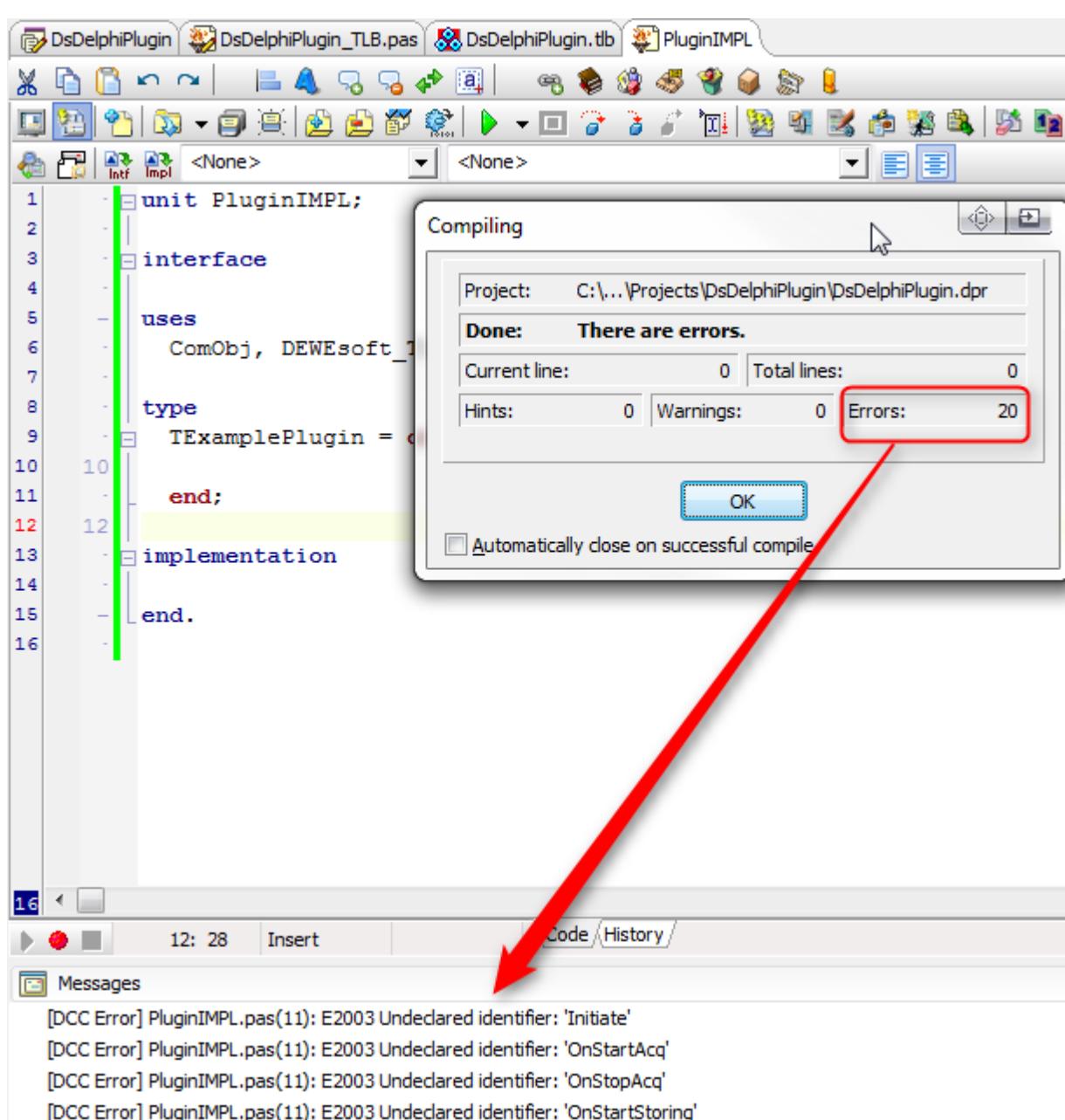
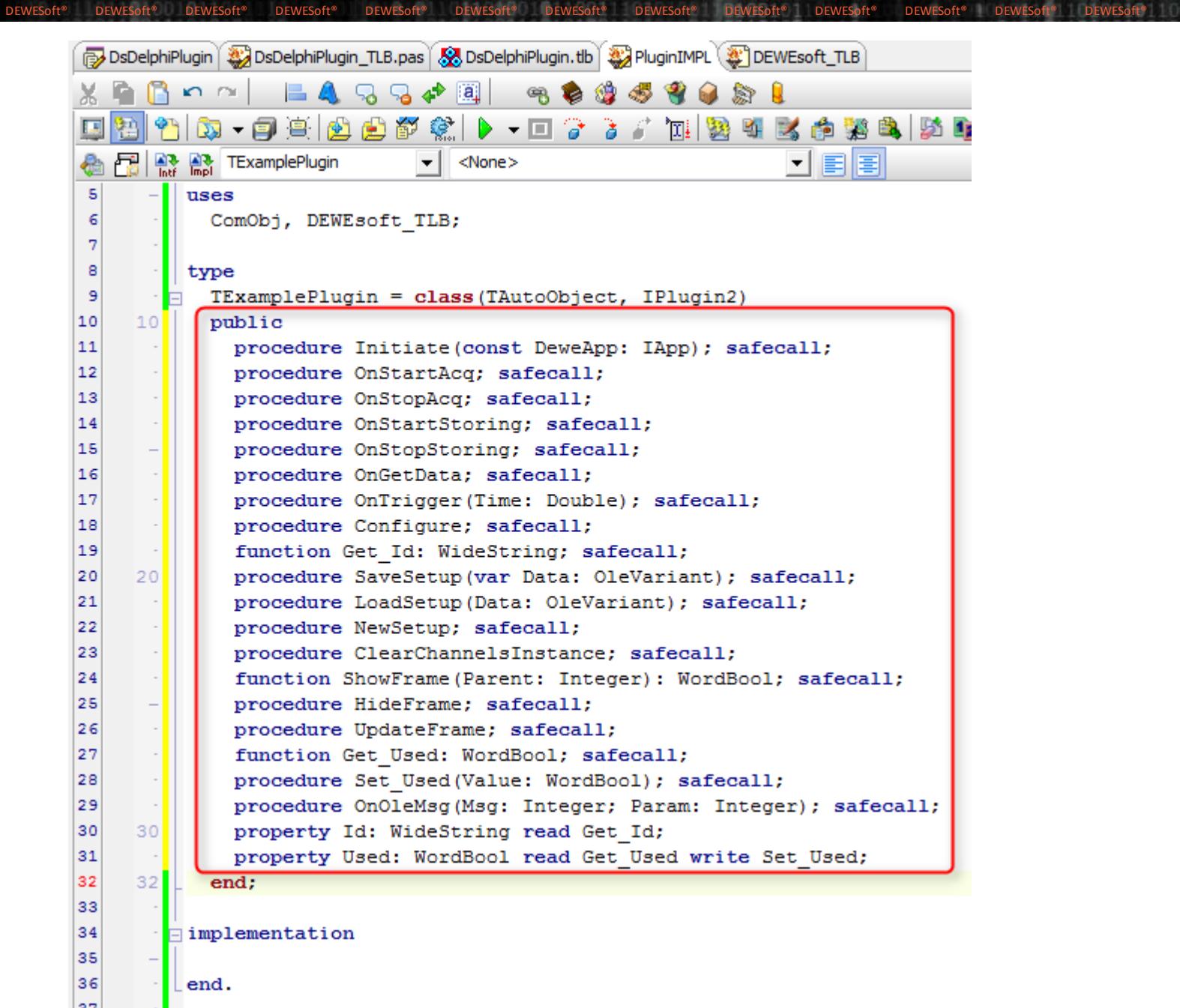


Illustration 42: Compilation Errors

The problem is, that the implementations of all methods of the `IPlugin2` interface are still missing. We can simply copy the method declarations from the `IPlugin2` definition in the `DEWEsoft_TLB.pas` file (you can hold the `Ctrl` key and then click on the `IPlugin2` token to get directly to the definition):



The screenshot shows the Delphi IDE interface with the following details:

- Title Bar:** Shows multiple tabs including "DsDelphiPlugin", "DsDelphiPlugin_TLB.pas", "DsDelphiPlugin.tlb", "PluginIMPL", and "DEWEsoft_TLB".
- Toolbar:** Standard Delphi toolbar with icons for file operations, selection, and code navigation.
- Status Bar:** Shows "TExamplePlugin" and "<None>".
- Code Editor:** Displays the Delphi code for `TExamplePlugin`. The code includes declarations for `uses`, `type`, and `implementation`. A red box highlights the `IPlugin2` interface declaration, which contains 29 methods listed from `Initiate` to `Set_Used`.

```

5   - uses
6     ComObj, DEWEsoft_TLB;
7
8   - type
9     TExamplePlugin = class(TAutoObject, IPlugin2)
10    public
11      procedure Initiate(const DeweApp: IApp); safecall;
12      procedure OnStartAcq; safecall;
13      procedure OnStopAcq; safecall;
14      procedure OnStartStoring; safecall;
15      procedure OnStopStoring; safecall;
16      procedure OnGetData; safecall;
17      procedure OnTrigger(Time: Double); safecall;
18      procedure Configure; safecall;
19      function Get_Id: WideString; safecall;
20      procedure SaveSetup(var Data: OleVariant); safecall;
21      procedure LoadSetup(Data: OleVariant); safecall;
22      procedure NewSetup; safecall;
23      procedure ClearChannelsInstance; safecall;
24      function ShowFrame(Parent: Integer): WordBool; safecall;
25      procedure HideFrame; safecall;
26      procedure UpdateFrame; safecall;
27      function Get_Used: WordBool; safecall;
28      procedure Set_Used(Value: WordBool); safecall;
29      procedure OnOleMsg(Msg: Integer; Param: Integer); safecall;
30      property Id: WideString read Get_Id;
31      property Used: WordBool read Get_Used write Set_Used;
32    end;
33
34  - implementation
35
36  - end.
37

```

Illustration 43: *IPlugin2* methods

We still need to create all the implementations for the declared functions. The easiest and fastest way to do that is to move the cursor to the class name (`TExamplePlugin`) and press `Ctrl-Alt-C` (or right-click and select `Complete class at cursor` from the pop-up menu):

```

procedure ClearChannelsInstance; safecall;
function ShowFrame(Parent: Integer): WordBool; safecall;
procedure HideFrame; safecall;
procedure UpdateFrame; safecall;
function Get_Used: WordBool; safecall;
procedure Set_Used(Value: WordBool); safecall;
procedure OnOleMsg(Msg: Integer; Param: Integer); safecall;
property Id: WideString read Get_Id;
property Used: WordBool read Get_Used write Set_Used;
end;

implementation

{ TExamplePlugin }

procedure TExamplePlugin.ClearChannelsInstance;
begin
end;

procedure TExamplePlugin.Configure;
begin
end;

function TExamplePlugin.Get_Id: WideString;
begin
end;

function TExamplePlugin.Get_Used: WordBool;
begin
end;

procedure TExamplePlugin.HideFrame;
begin
end;

```

Illustration 44: Class completion

You can see that the class-completion feature has added empty method implementations for all declared methods on the class. The next step is to implement some meaningful code to the most important functions.

Class implementation

Now we will start to implement the most important functions that are necessary to make our plugin work in *DEWESoft®*.

Keep reference for the [IApp](#) interface

After *DEWESoft®* starts the plugin, it will call the [Initiate](#) function and pass along the [IApp](#) interface. We will assign it to

a local variable called `FApp`. It is the main entry point for communicating with `DEWESoft®` and we will need it in most other methods that we implement.

```
TExamplePlugin = class(TAutoObject, IPlugin2)
private
    FApp: IApp;
public
    .
    .
    .
procedure TExamplePlugin.Initiate(const DweApp: IApp);
begin
    FApp := DweApp;
end;
```

Remember the Used-status

Next, we implement the [Used](#) status which tells us, if the user has set the plugin to *Used* or *Unused* in the hardware setup.

```
type
TExamplePlugin = class(TAutoObject, IPlugin2)
private
    FApp: IApp;
    FUsed: WordBool;
public
    .
    .
    .
function TExamplePlugin.Get_Used: WordBool;
begin
    Result := FUsed;
end;

procedure TExamplePlugin.Set_Used(Value: WordBool);
begin
    FUsed := Value;
end;
```

Implement channel setup

Next, we implement the [Configure\(\)](#) function. This is not really necessary, but we just want to check if the function is called:

```
procedure TExamplePlugin.Configure;
begin
    MessageDlg('Deplhi plugin could show a config dialog here or could use the ' +
               'ShowFrame function to embedd a frame.',
               mtInformation, [mbOK], 0);
end;
```

Since we do not show an embedded frame in the channel setup of `DEWESoft®`, we set the result of the [ShowFrame\(\)](#) function to `False` (in this example we simply show a message box via the [Configure\(\)](#) function above).

```
function TExamplePlugin.ShowFrame(Parent: Integer): WordBool;
begin
    Result := False;
end;
```

Add a plugin channel

Finally we will add a [channel](#) and fill it with some random data.

```

type
TExamplePlugin = class(TAutoObject, IPlugin2)
private
  FApp: IApp;
  FUsed: WordBool;
  FCh1: IChannel;
  procedure MountChannels();
public
  .
  .
procedure TExamplePlugin.MountChannels;
var
  PluginGroup: IPluginGroup;
begin
  PluginGroup := FApp.Data.Groups.Item[8] as IPluginGroup;
  FCh1 := PluginGroup.MountChannel(5, True, -1);
  FCh1.Name := 'Test Channel 1';
  FCh1.Used := True;
end;

procedure TExamplePlugin.LoadSetup(Data: OleVariant);
begin
  MountChannels();
end;

procedure TExamplePlugin.NewSetup;
begin
  MountChannels();
end;

procedure TExamplePlugin.OnGetData;
var
  Timestamp: Extended;
begin
  Timestamp := FApp.MasterClock.GetCurrentTime;
  FCh1.AddAsyncSingleSample(Random(), Timestamp);
end;

```

Now that the implementation is okay, we need to add some more information to make our class library work with *DCOM* and *DEWESoft®*: see [Delphi: Prepare for DCOM](#)

1.5.2.3 Delphi: Prepare for DCOM

Now that the plugin implementation is complete (see [Delphi: Implementation](#)), we need to add some more code, so that the class is recognized by *DCOM* and by *DEWESoft®*.

```
{
  The TPluginObjectFactory class will handle the DCOM registration and
  the registration for DEWESoft® to find the DCOM object.
```

TAutoObjectFactory creates instances of the TAutoObject class, and has

DEWEsoft®
methods that manage the functionality of all of the instances of the automation objects it creates.

Note: In Delphi applications, an Automation object is a COM object, which means that the module in which it is implemented must provide a factory object so that the system can create the Automation object. To add an Automation object factory to your program, create the factory object in the initialization section of the unit that defines the Automation object.

```

}

TPluginObjectFactory = class(TAutoObjectFactory)
public
    procedure UpdateRegistry(Register: Boolean); override;
end;

implementation

uses
    Dialogs, Registry, Windows, DsDelphiPlugin_TLB, ComServ;
.

.

procedure TPluginObjectFactory.UpdateRegistry(Register: Boolean);
var
    Reg: TRegistry;
const
    PLUGIN_KEY = 'Delphi_E01';
    PLUGIN_NAME = 'Delphi Example 01';
begin
    inherited UpdateRegistry(Register);

    {
        register/unregister the plugin dll to DEWEsoft®
    }
    if Register then
    begin
        Reg := TRegistry.Create(KEY_ALL_ACCESS);
        Reg.RootKey := HKEY_LOCAL_MACHINE;
        Reg.OpenKey('Software\Devesoft\Plugins\' + PLUGIN_KEY, True);
        Reg.WriteString('GUID', GUIDToString(CLASS_DelphiPlugin));
        Reg.WriteString('Name', PLUGIN_NAME);
        Reg.WriteString('Version', 'beta');
        Reg.WriteString('Description', 'Just an empty template...');

        Reg.WriteString('Vendor', 'Devesoft');
        Reg.WriteInteger('TLB', DEWEsoftMinorVersion);
        Reg.Free;
    end
    else
    begin
        Reg := TRegistry.Create(KEY_ALL_ACCESS);
        Reg.RootKey := HKEY_LOCAL_MACHINE;
        Reg.DeleteKey('Software\Devesoft\Plugins\' + PLUGIN_KEY);
        Reg.Free;
    end;
end;

initialization
    TPluginObjectFactory.Create(ComServer, TExamplePlugin, CLASS_DelphiPlugin,
        ciMultiInstance, tmApartment);

```

When **DEWEsoft®** starts up it will load all the .dll files it finds in the Addons folder. When our .dll file is loaded, the initialization section will be executed which creates an instance of our **TPluginObjectFactory** class.

Now that the source code is complete, we can build the project:

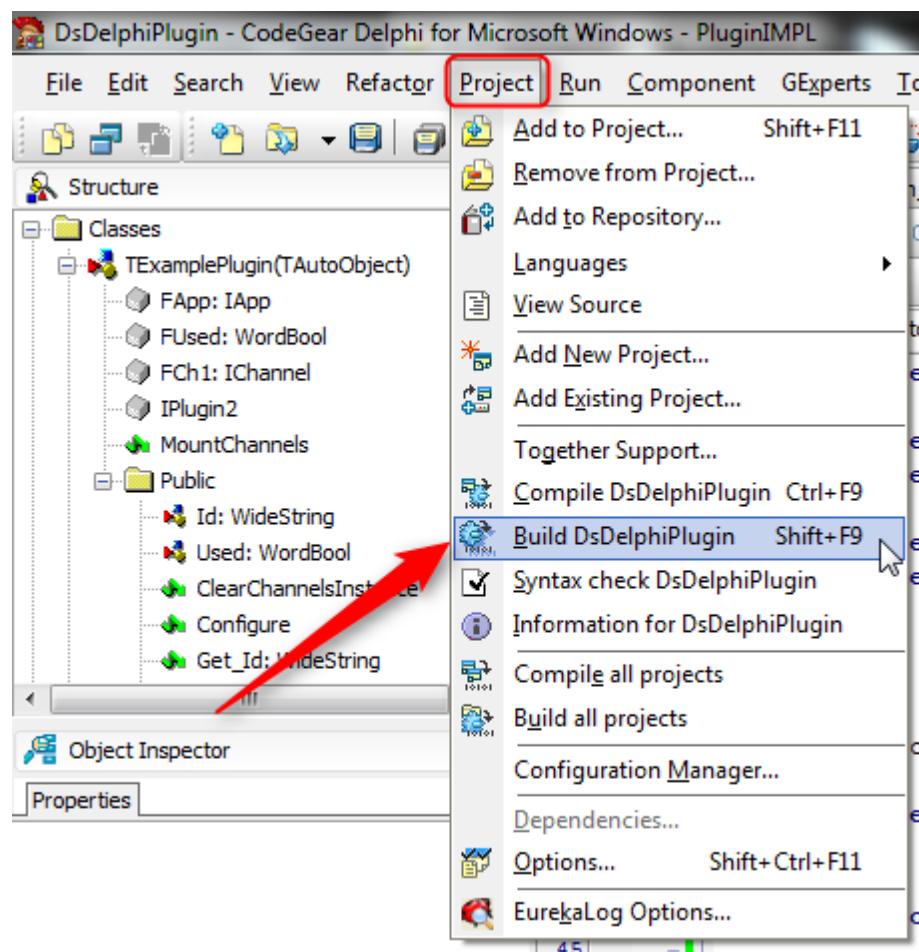


Illustration 45: Build Project

The Delphi build-process will create the plugin dll file directly in the Addons folder of **DEWEsoft®** (see [Delphi: Prepare Project](#) for details on how to specify the output directory):

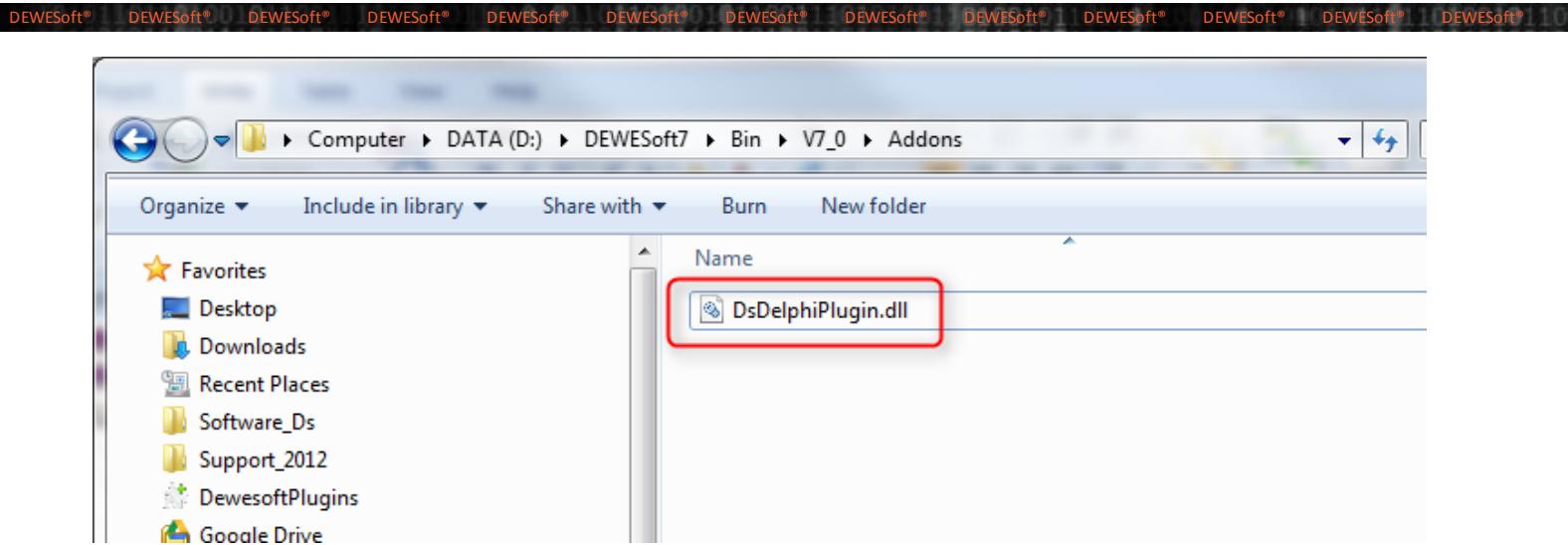


Illustration 46: Delphi Build Result

next: [Delphi: Register Assembly](#)

1.5.2.4 Delphi: Register Plugin

The plugin registration is done automatically when *DEWEsoft™* starts up. It will load the `.dll` file (since it is in the **Addons** folder), and the initialization section of our plugin (see [Delphi: Prepare for DCOM](#)) will create the `TPluginObjectFactory` class which does all the registration work.

Note: on Windows 7 it may be necessary that you start the plugin registration manually and then restart *DEWEsoft®*:

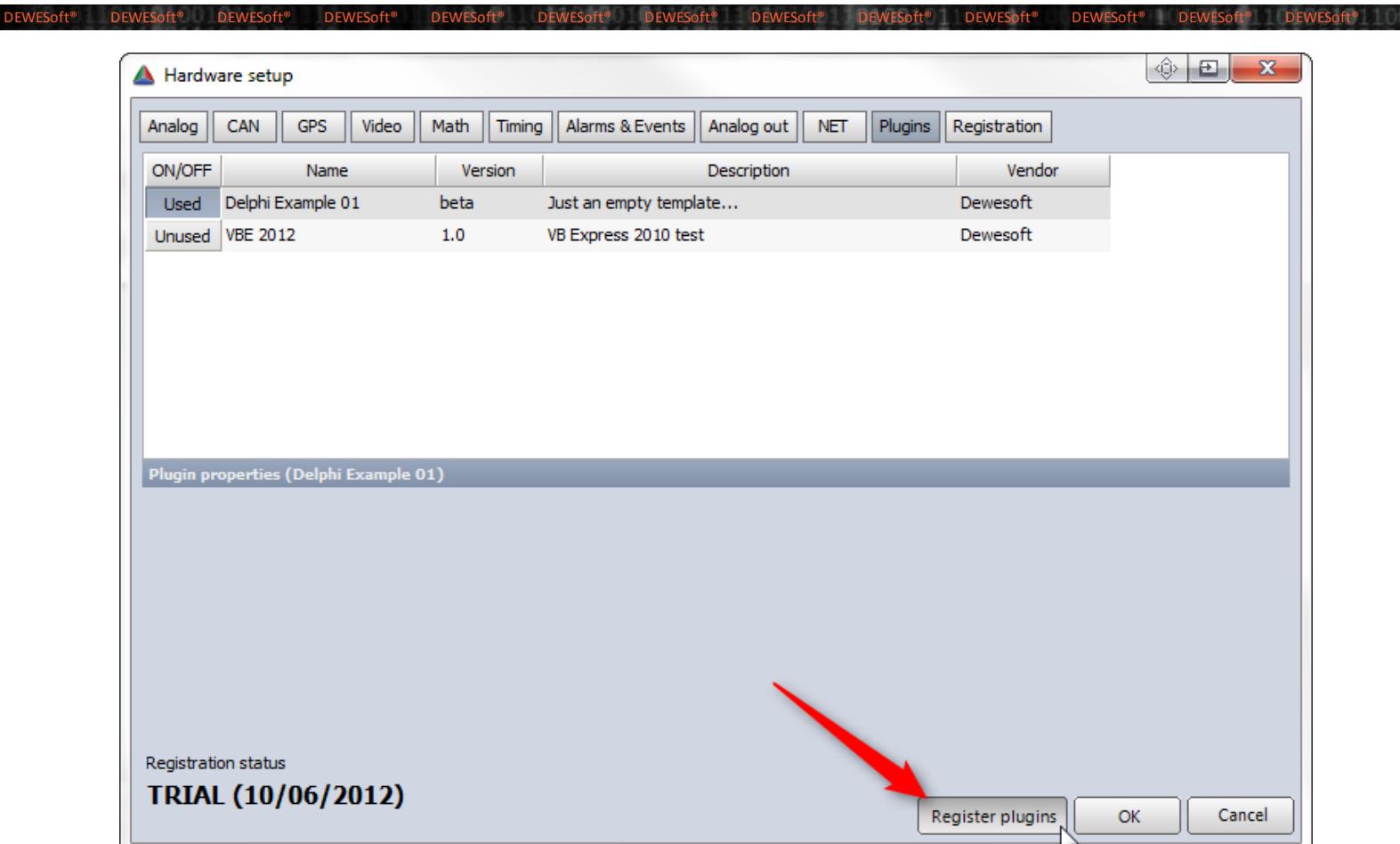


Illustration 47: Register Plugins

see also: [Finding Plugins](#)

That means: we can simply start *DEWESoft®* and test the plugin: [Delphi: Test the Plugin](#)

1.5.2.5 Delphi: Test the Plugin

After we have successfully registered the plugin (see [Delphi: Register Plugin](#)), we can finally start up *DEWESoft®* and test our Delphi plugin.

We can do this right from the Delphi IDE; press F9 or select *Run - Run* from the menu:

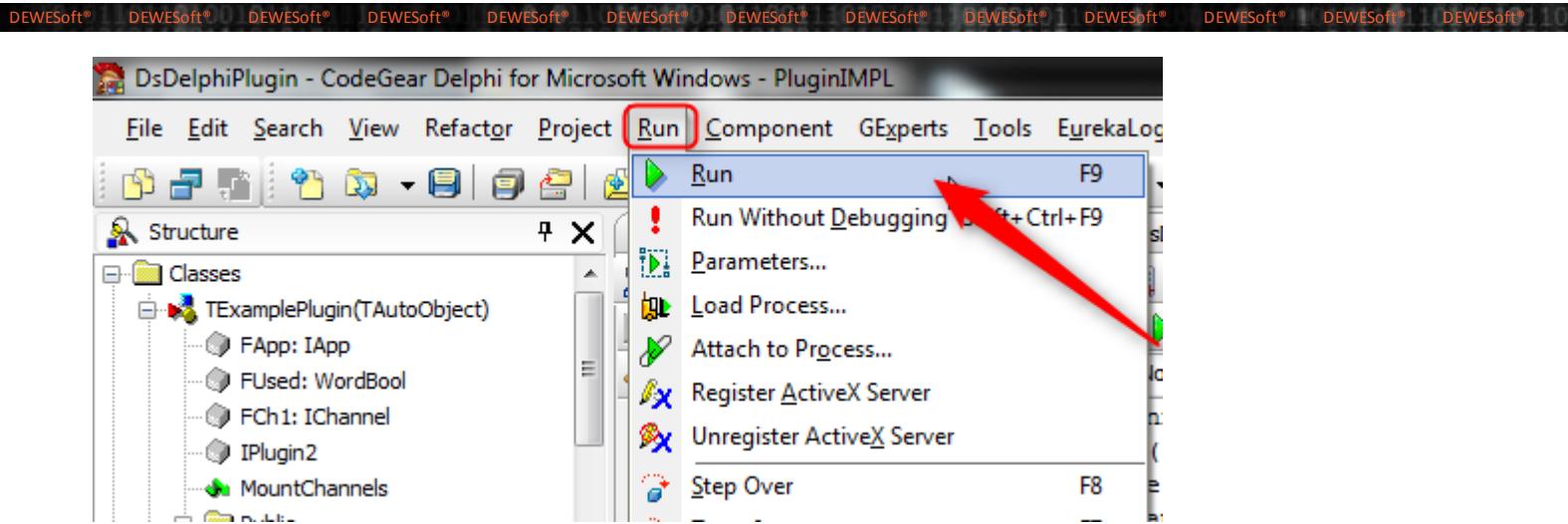


Illustration 48: Start DEWESoft™ from the Delphi GUI

After *DEWESoft®* has started, you can go to the *Plugins* tab sheet of the hardware setup, and the Plugin will already show up:

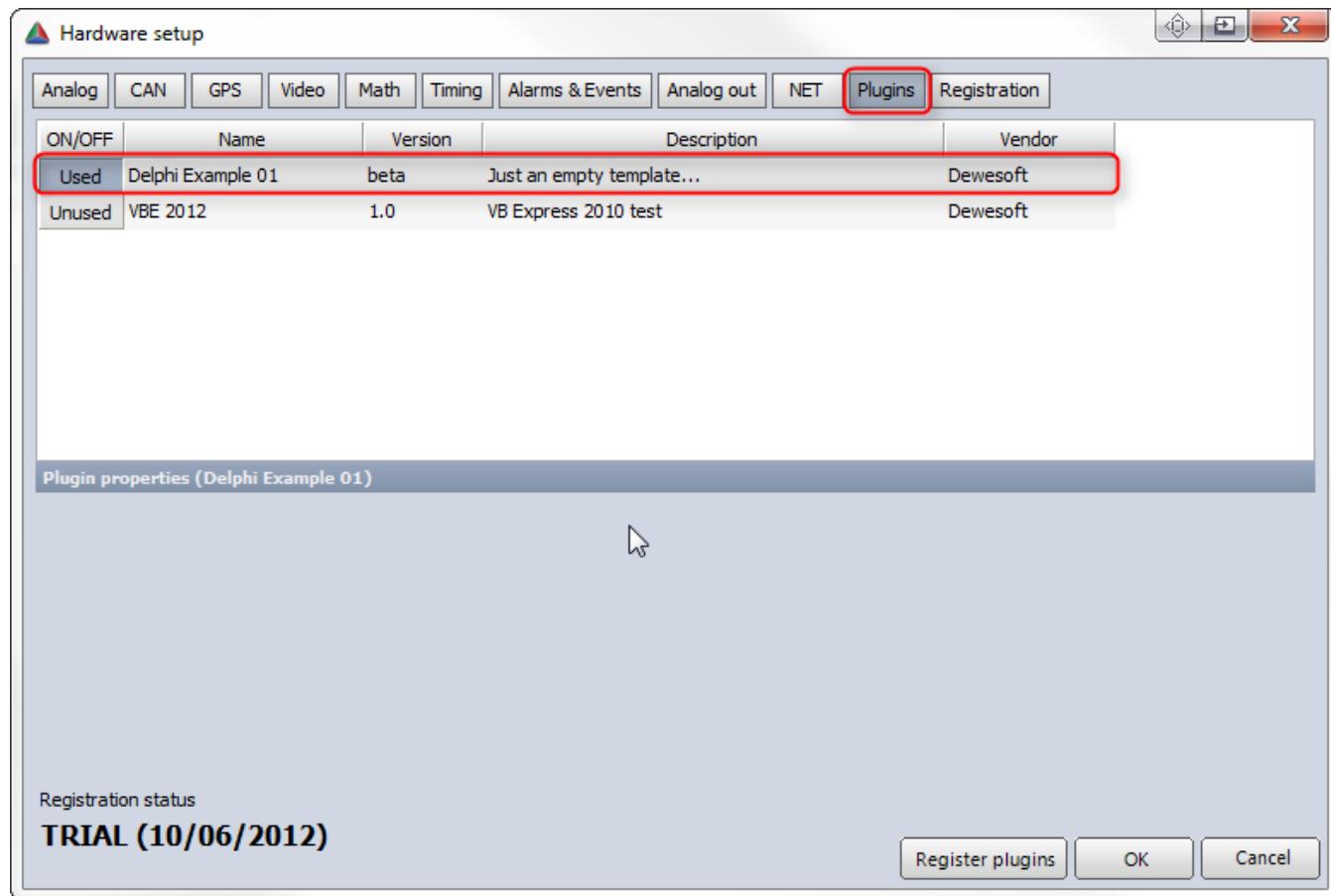


Illustration 49: Plugin in Hardware setup

Click the *Unused* button to set the plugin to *Used* and then you can already see the plugin in the channel setup:

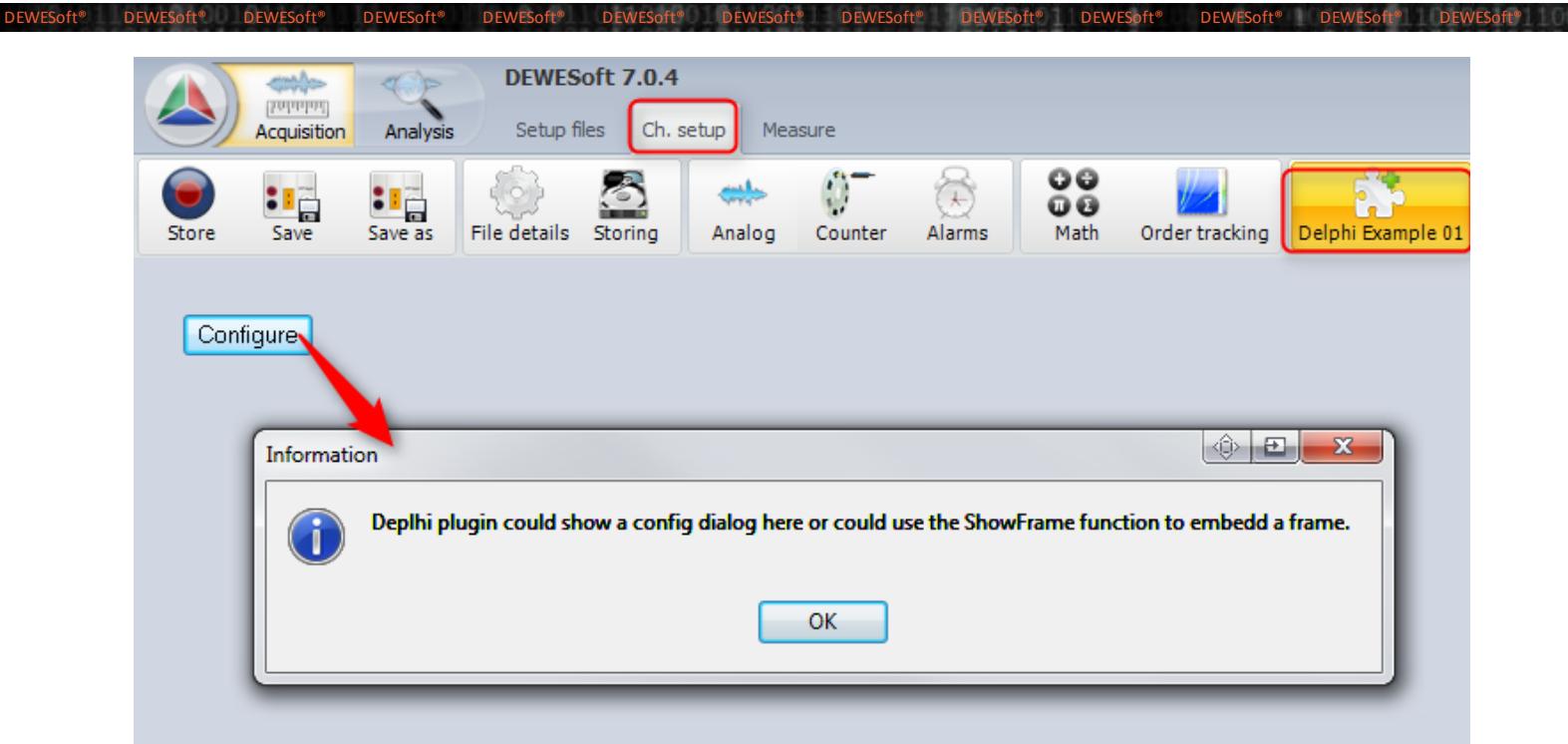


Illustration 50: Plugin - Channel setup

When you press the *Configure* button, the message box that we have defined in the `Configure()` function (see [Delphi: Implementation](#)) will show up.

When we switch to *Measure* mode, we can see that the channel (called *Test Channel 1*) that we have mounted (see function `MountChannels()` in [Delphi: Implementation](#)) shows up in the channel list (see red rectangle on the right side in the screenshot below) and that the random data is shown in the recorder:

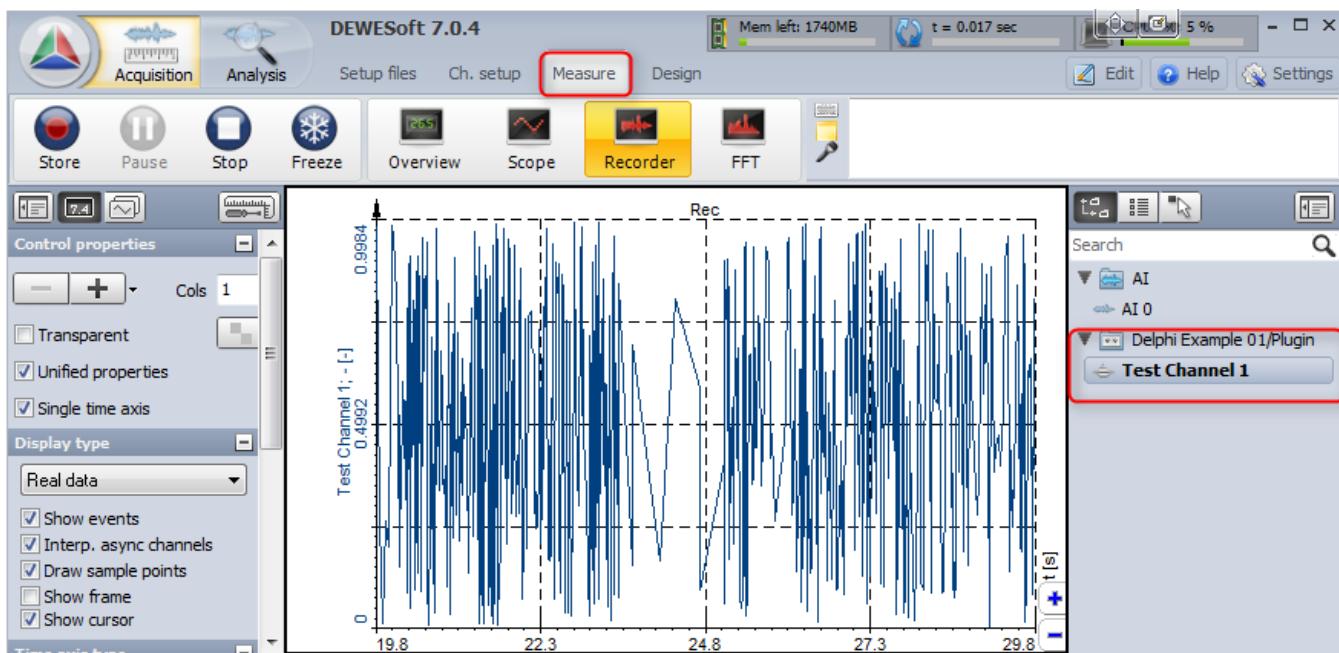


Illustration 51: Measure Mode

see also: [Delphi: Sourcecode](#)

1.5.2.6 Delphi: Sourcecode

The complete source code of our Delphi example plugin:

```
unit PluginIMPL;

interface

uses
  ComObj, DEWEsoft_TLB;

type
  TExamplePlugin = class(TAutoObject, IPlugin2)
  private
    FApp: IApp;
    FUsed: WordBool;
    FCh1: IChannel;
    procedure MountChannels();
  public
    procedure Initiate(const DeweApp: IApp); safecall;
    procedure OnStartAcq; safecall;
    procedure OnStopAcq; safecall;
    procedure OnStartStoring; safecall;
    procedure OnStopStoring; safecall;
    procedure OnGetData; safecall;
    procedure OnTrigger(Time: Double); safecall;
    procedure Configure; safecall;
    function Get_Id: WideString; safecall;
    procedure SaveSetup(var Data: OleVariant); safecall;
    procedure LoadSetup(Data: OleVariant); safecall;
    procedure NewSetup; safecall;
    procedure ClearChannelsInstance; safecall;
    function ShowFrame(Parent: Integer): WordBool; safecall;
    procedure HideFrame; safecall;
    procedure UpdateFrame; safecall;
    function Get_Used: WordBool; safecall;
    procedure Set_Used(Value: WordBool); safecall;
    procedure OnOleMsg(Msg: Integer; Param: Integer); safecall;
    property Id: WideString read Get_Id;
    property Used: WordBool read Get_Used write Set_Used;
  end;

TPluginObjectFactory = class(TAutoObjectFactory)
  public
    procedure UpdateRegistry(Register: Boolean); override;
  end;

implementation

uses
  Dialogs, Registry, Windows, DsDelphiPlugin_TLB, ComServ;

{ TExamplePlugin }

procedure TExamplePlugin.ClearChannelsInstance;
begin
end;
```

```
procedure TExamplePlugin.Configure;
begin
  MessageDlg('Delphi plugin could show a config dialog here or could use the '+'ShowFrame function to embed a frame.',
  mtInformation, [mbOK], 0);
end;

function TExamplePlugin.Get_Id: WideString;
begin
end;

function TExamplePlugin.Get_Used: WordBool;
begin
  Result := FUsed;
end;

procedure TExamplePlugin.Set_Used(Value: WordBool);
begin
  FUsed := Value;
end;

procedure TExamplePlugin.HideFrame;
begin
end;

procedure TExamplePlugin.Initiate(const DeweApp: IApp);
begin
  FApp := DeweApp;
end;

procedure TExamplePlugin.MountChannels;
var
  PluginGroup: IPluginGroup;
begin
  PluginGroup := FApp.Data.Groups.Item[8] as IPluginGroup;
  FCh1 := PluginGroup.MountChannel(5, True, -1);
  FCh1.Name := 'Test Channel 1';
  FCh1.Used := True;
end;

procedure TExamplePlugin.LoadSetup(Data: OleVariant);
begin
  MountChannels();
end;

procedure TExamplePlugin.NewSetup;
begin
  MountChannels();
end;

procedure TExamplePlugin.OnGetData;
var
  Timestamp: Extended;
begin
  Timestamp := FApp.MasterClock.GetCurrentTime;
  FCh1.AddAsyncSingleSample(Random(), Timestamp);
end;
```

```
procedure TExamplePlugin.OnOleMsg(Msg, Param: Integer);
begin
end;

procedure TExamplePlugin.OnStartAcq;
begin
end;

procedure TExamplePlugin.OnStartStoring;
begin
end;

procedure TExamplePlugin.OnStopAcq;
begin
end;

procedure TExamplePlugin.OnStopStoring;
begin
end;

procedure TExamplePlugin.OnTrigger(Time: Double);
begin
end;

procedure TExamplePlugin.SaveSetup(var Data: OleVariant);
begin
end;

function TExamplePlugin.ShowFrame(Parent: Integer): WordBool;
begin
    Result := False;
end;

procedure TExamplePlugin.UpdateFrame;
begin
end;

{ TPluginObjectFactory }

{ TPluginObjectFactory }

procedure TPluginObjectFactory.UpdateRegistry(Register: Boolean);
var
    Reg: TRegistry;
const
    PLUGIN_KEY = 'Delphi_E01';
    PLUGIN_NAME = 'Delphi Example 01';
begin
    inherited UpdateRegistry(Register);
    if Register then
        begin
            Reg := TRegistry.Create(KEY_ALL_ACCESS);
            Reg.RootKey := HKEY_LOCAL_MACHINE;
            Reg.OpenKey('Software\DEWESoft\Plugins\' + PLUGIN_KEY, True);
        end;
end;
```

1.5.3 Visual Basic

This example shows how to build a simple plugin using Microsoft Visual Basic 2012 Express.

- [Visual Basic: Prepare Project](#)
 - [Visual Basic: Implementation](#)
 - [Visual Basic: Prepare for DCOM](#)
 - [Visual Basic: Register Assembly](#)
 - [Visual Basic: Test the Plugin](#)
 - [Visual Basic: Troubleshooting](#)
 - [Visual Basic: Sourcecode](#)

1.5.3.1 Visual Basic: Prepare Project

Goto *File* and select *New Project...*

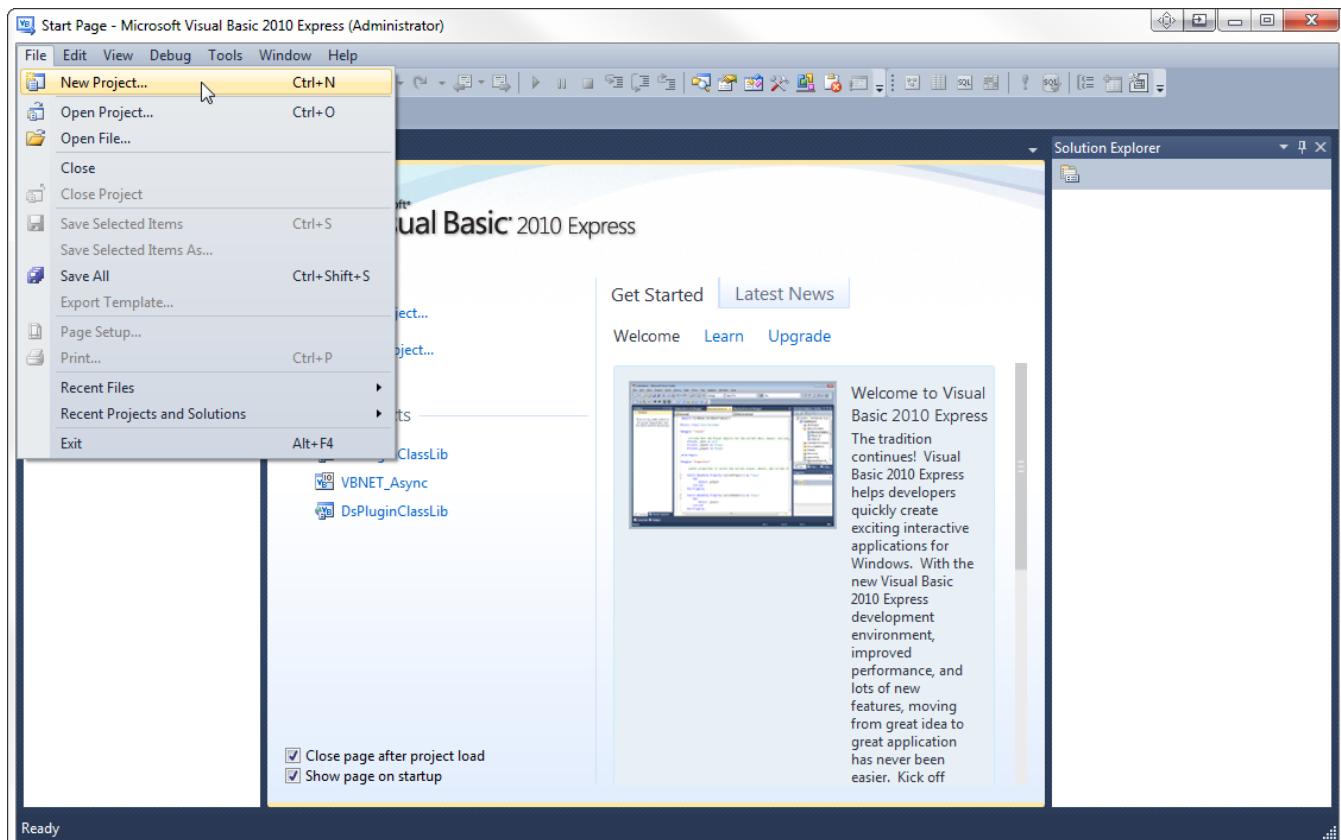


Illustration 52: Create Project

Select *Class Library* project and enter a name (e.g. *DsVbPlugin*):

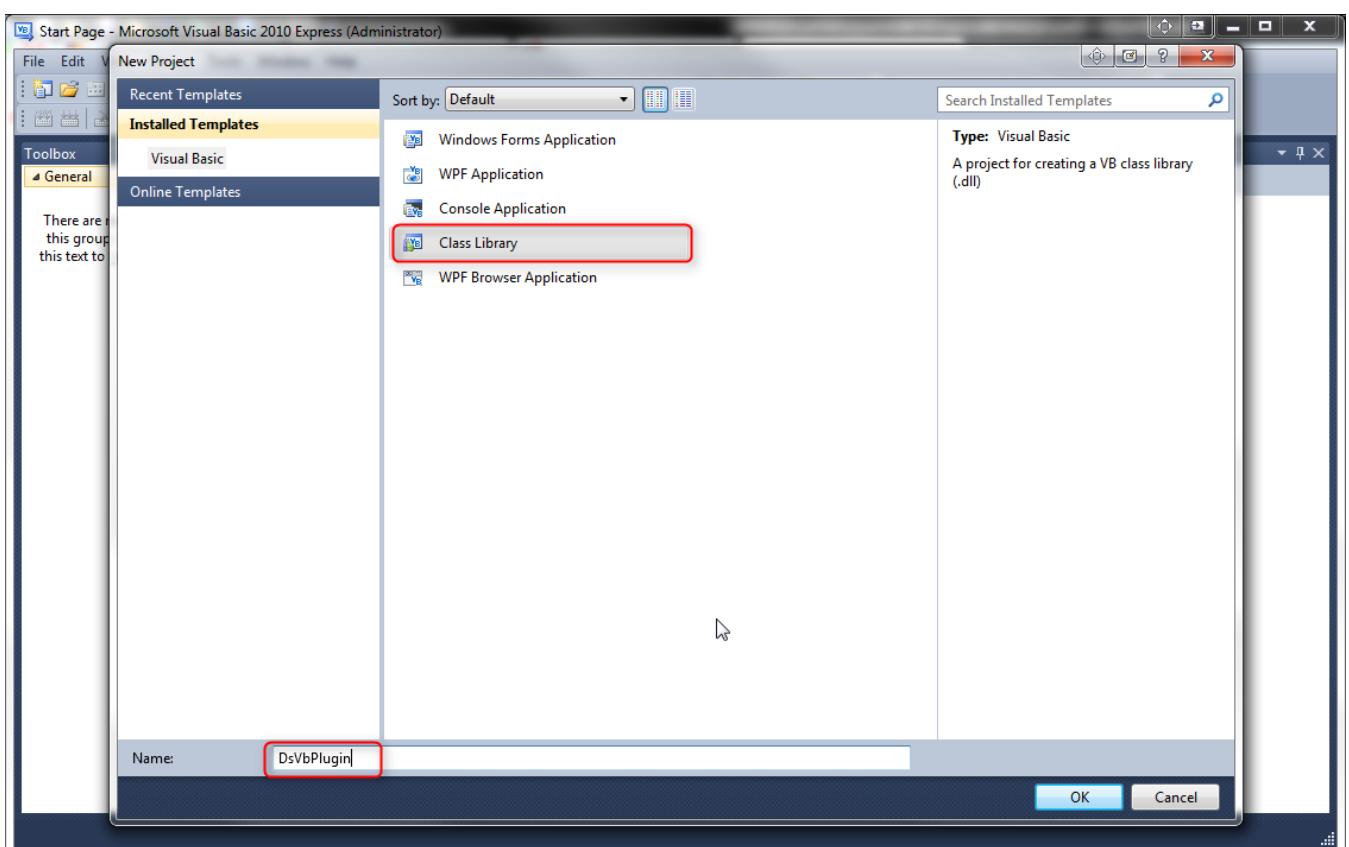


Illustration 53: Create Class Library

Now, go to [Project - DsVbPlugin Properties](#):

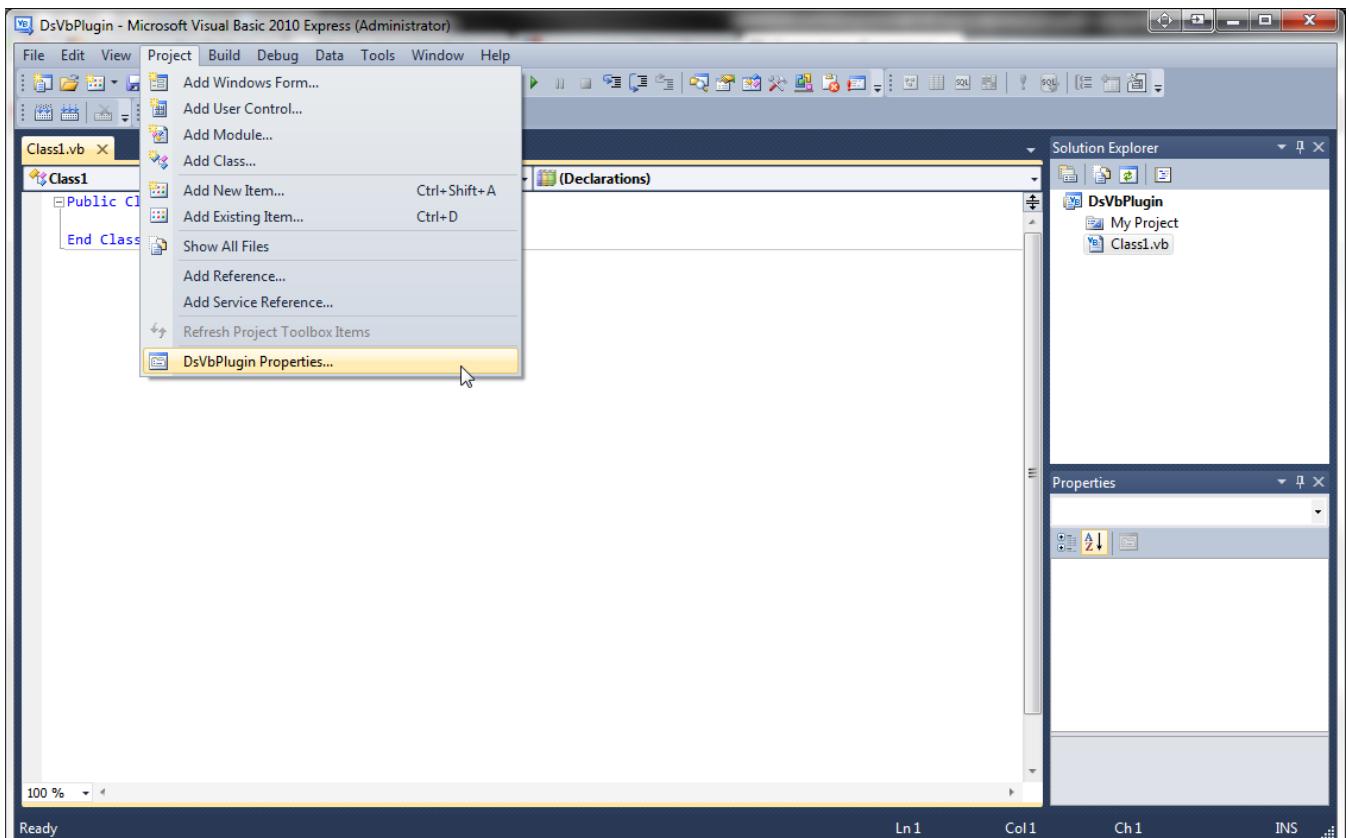


Illustration 54: Project Properties

In the *Application* section of the project properties, click the [Assembly Information...](#) button and make sure to check the [Make assembly COM-Visible](#) check-box.

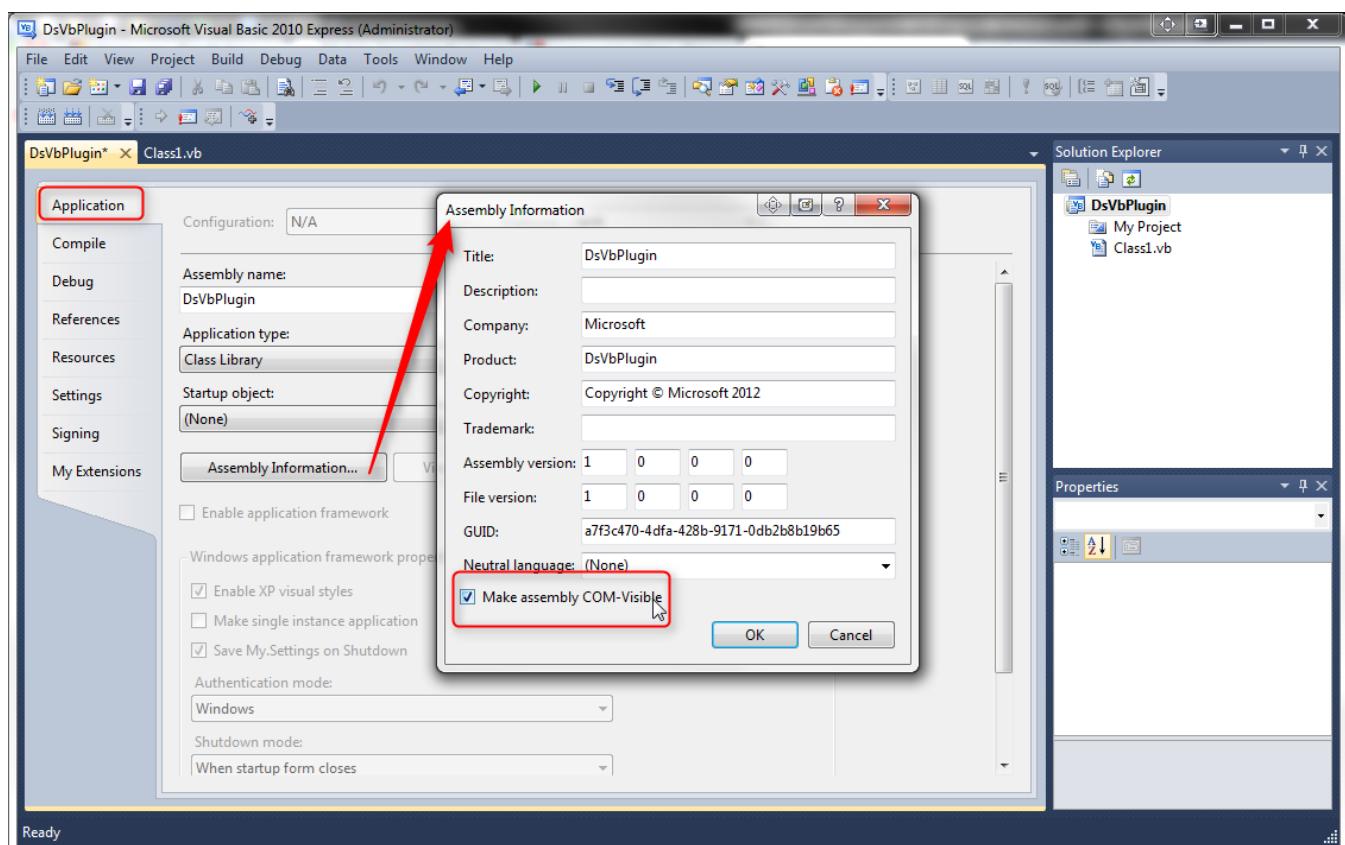


Illustration 55: Open Assembly Info

In the *References* section of the project properties, click on the arrow near the [Add...](#) button and select [Reference...](#):

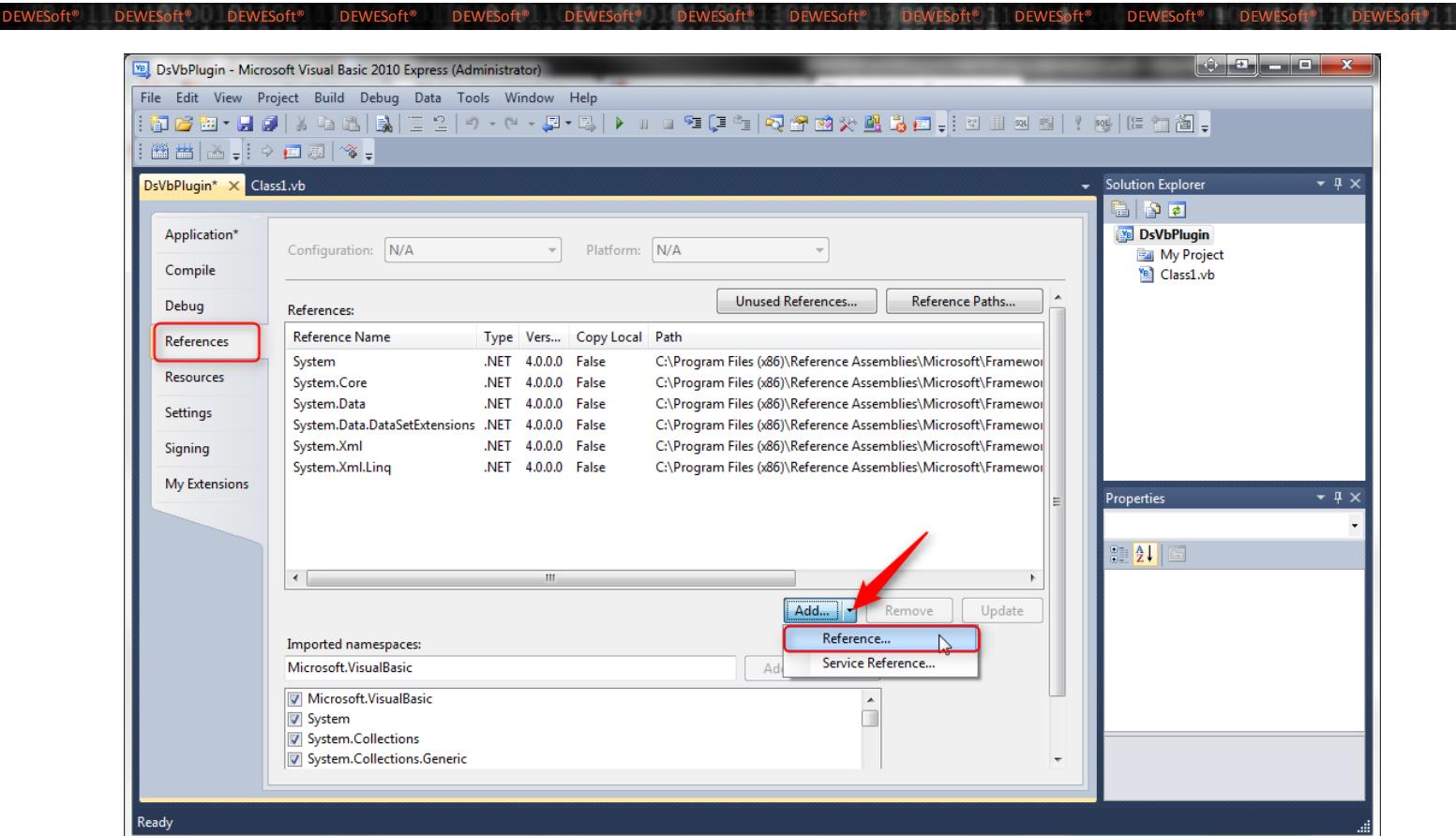


Illustration 56: Add Reference

Note: optionally you can now choose to [sign the assembly](#) to avoid a warning message when [registering the assembly](#).

In the *Add Reference* dialogue, goto the **COM** tab-sheet and select the *DEWESoft Library*:

NOTE: the library will only show up, if *DEWESoft®* has been installed correctly on your system!

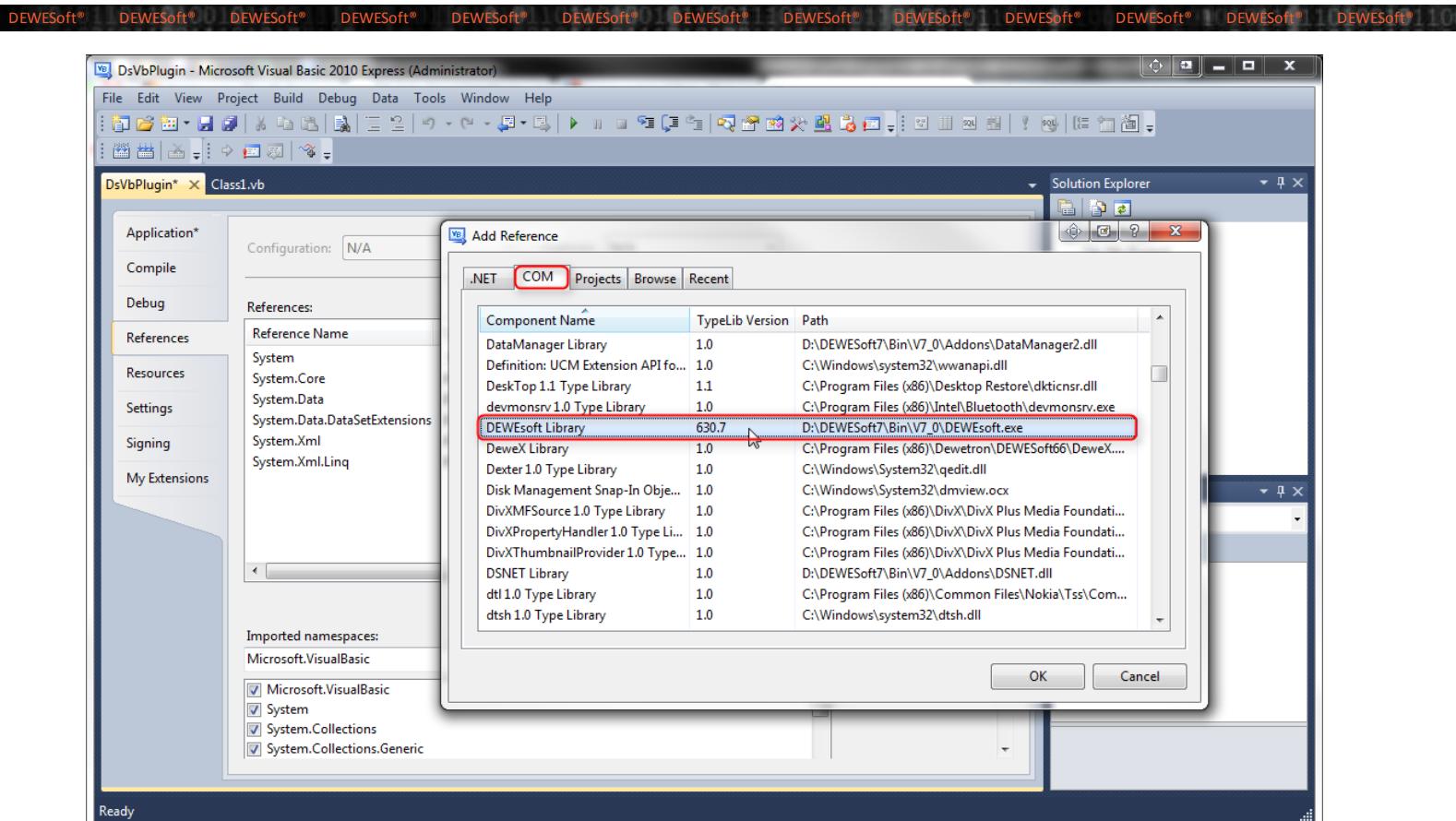


Illustration 57: Add reference to the DEWEsoft type library

Now it's time to save the project:

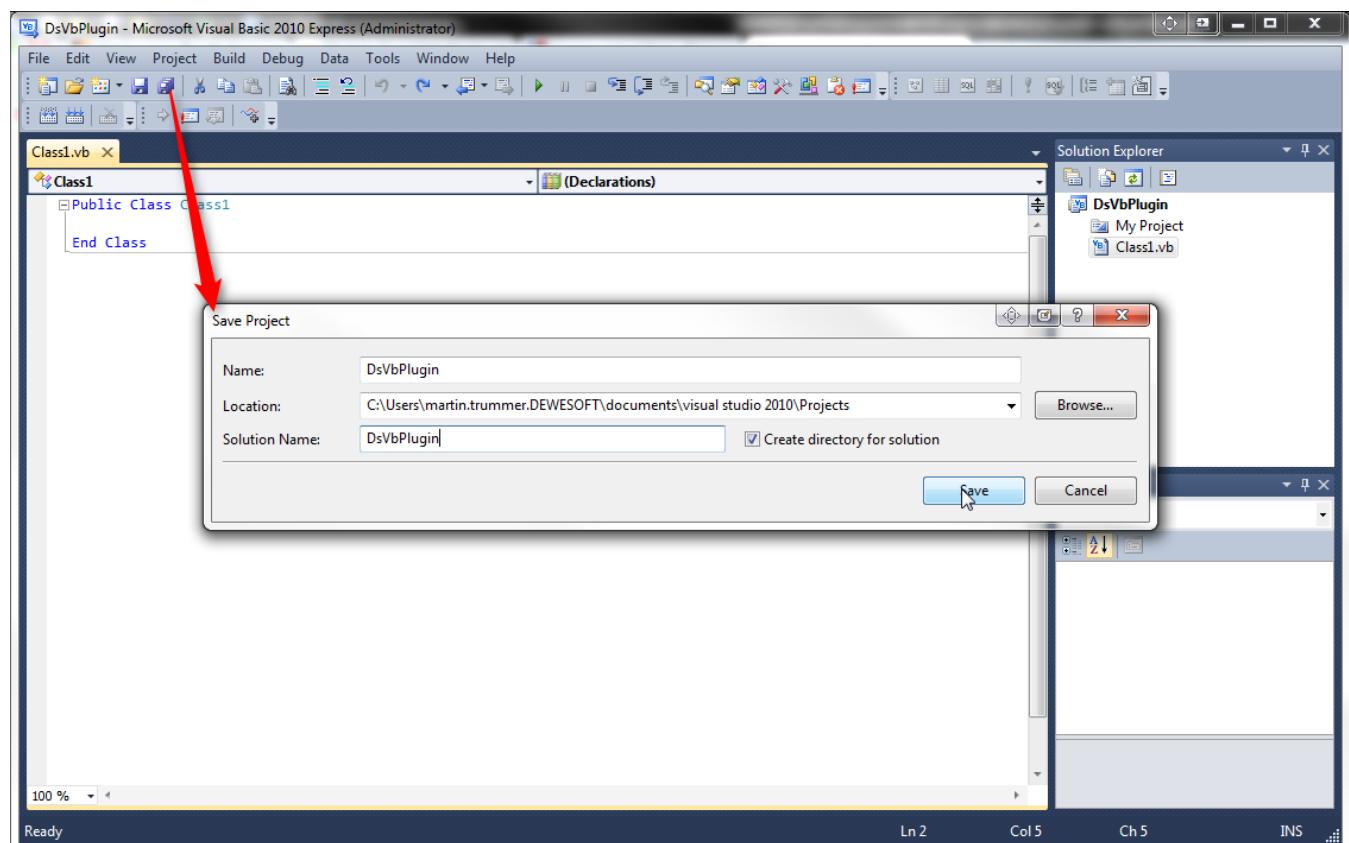


Illustration 58: Save the project

The project has been set-up correctly: it's now time to implement the plugin: see [Visual Basic: Implementation](#)

1.5.3.2 Visual Basic: Implementation

After you have prepared the Visual Basic project ([Visual Basic: Prepare Project](#)), we can start to implement the plugin class.

The first step is to rename the default class (*Class1*) to something more meaningful: e.g. *DsVbPluginMainClass*: and make it implement the *IPlugin2* interface from the *DEWEsoft* DCOM library.

Note: after you have added the *Implements* clause, the Visual Basic IDE will show a curled line under the token *IPlugin2* to indicate that *IPlugin2* is not defined. When you hover with the mouse over the token *IPlugin2*, the IDE will show you some suggestions how to fix the problem. In our case we just want to add an import statement referring to the *DEWEsoft* DCOM library.

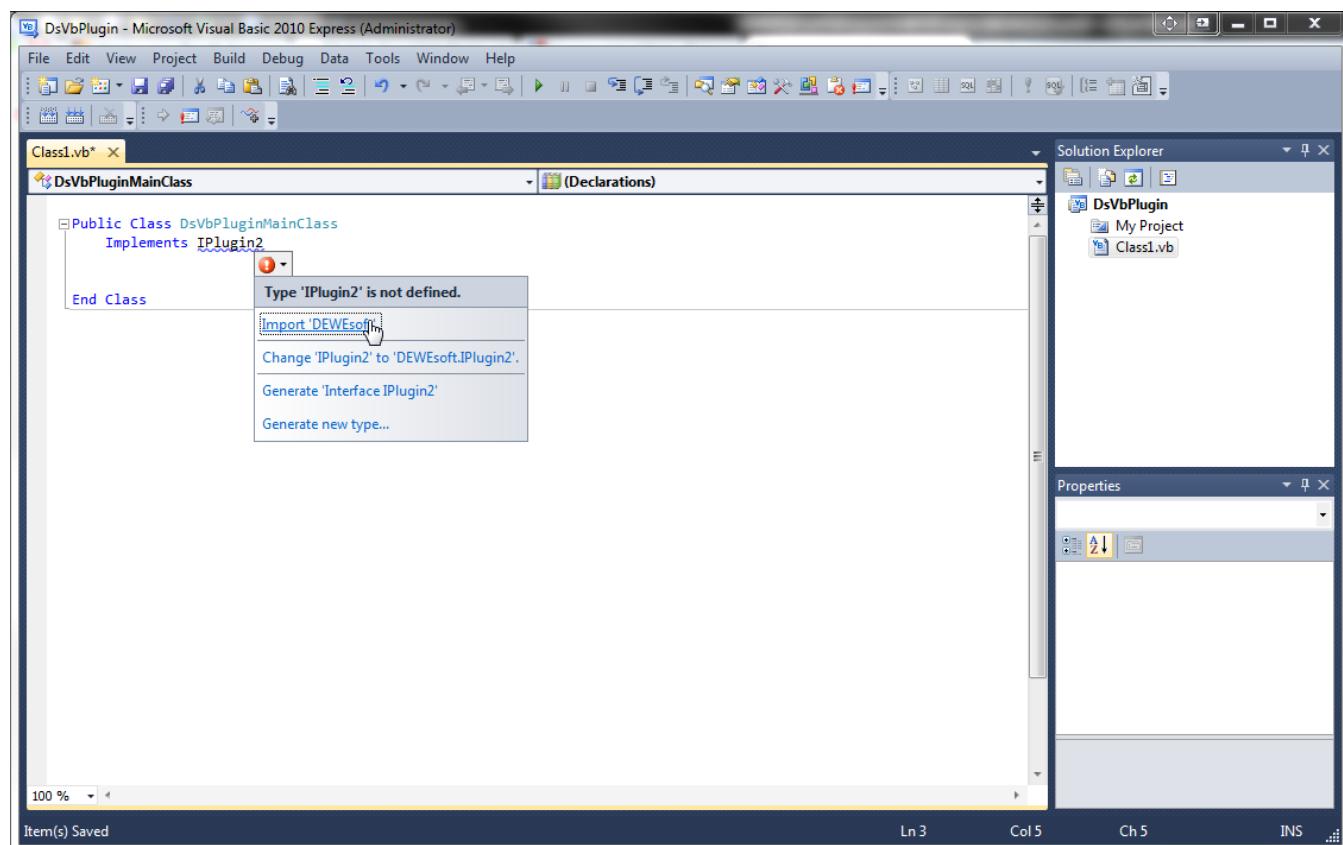


Illustration 59: Class declaration

You can see that the IDE has added an Imports statement automatically (1st line in the Illustration below). The token *IPlugin2* is still underlined, but this time, hovering over the token shows another message, which indicates that we now need to implement all methods and properties of this interface. Also in this case the Visual Basic IDE can help us to add default implementations for the methods and properties, which saves a lot of typing. Just move the cursor directly after the *IPlugin2* token and press the *Enter* key:

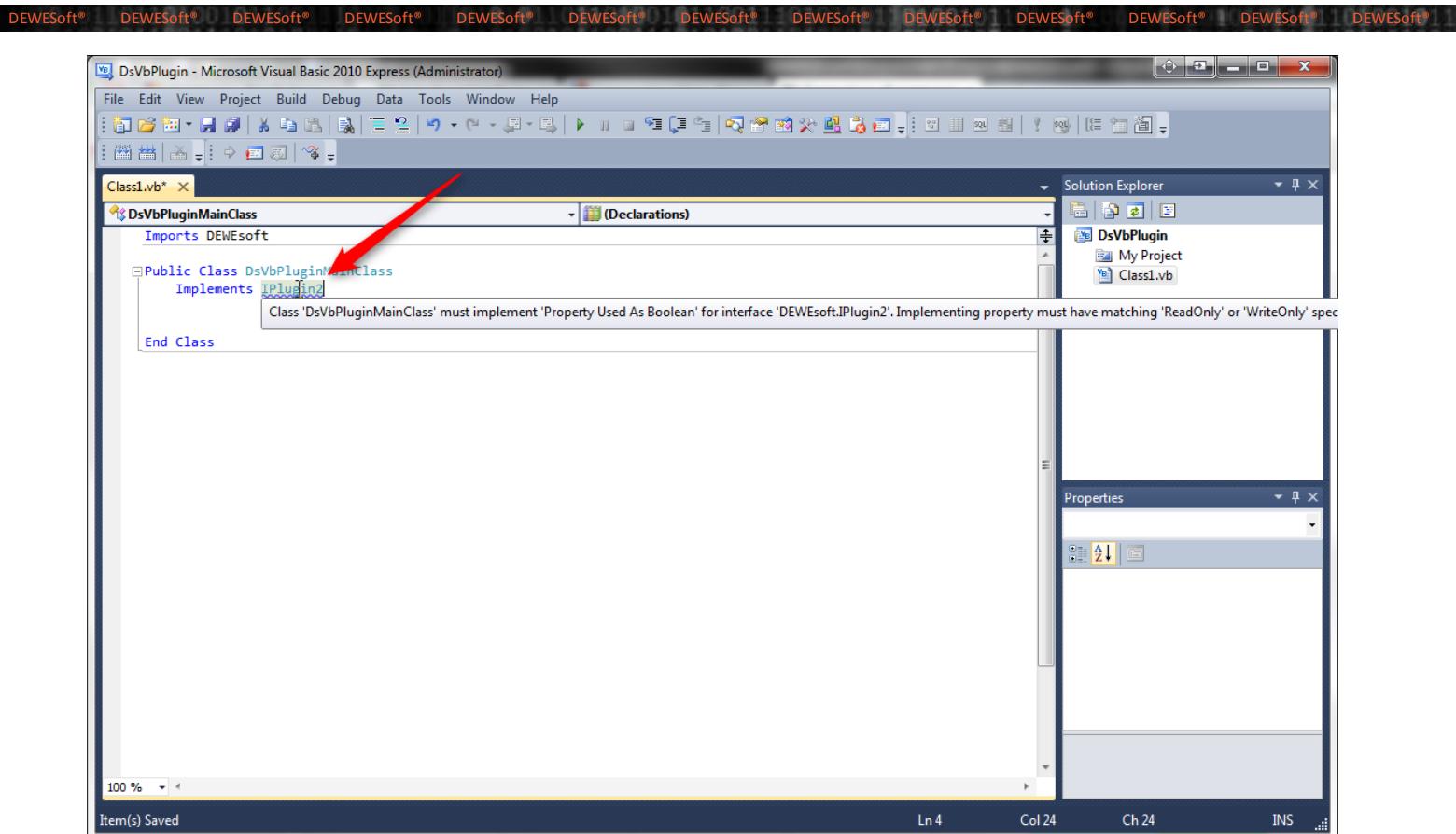


Illustration 60: create a dummy implementation of the interface

You can see that the IDE has created dummy implementations for all methods and properties of the interface `IPlugin2`:

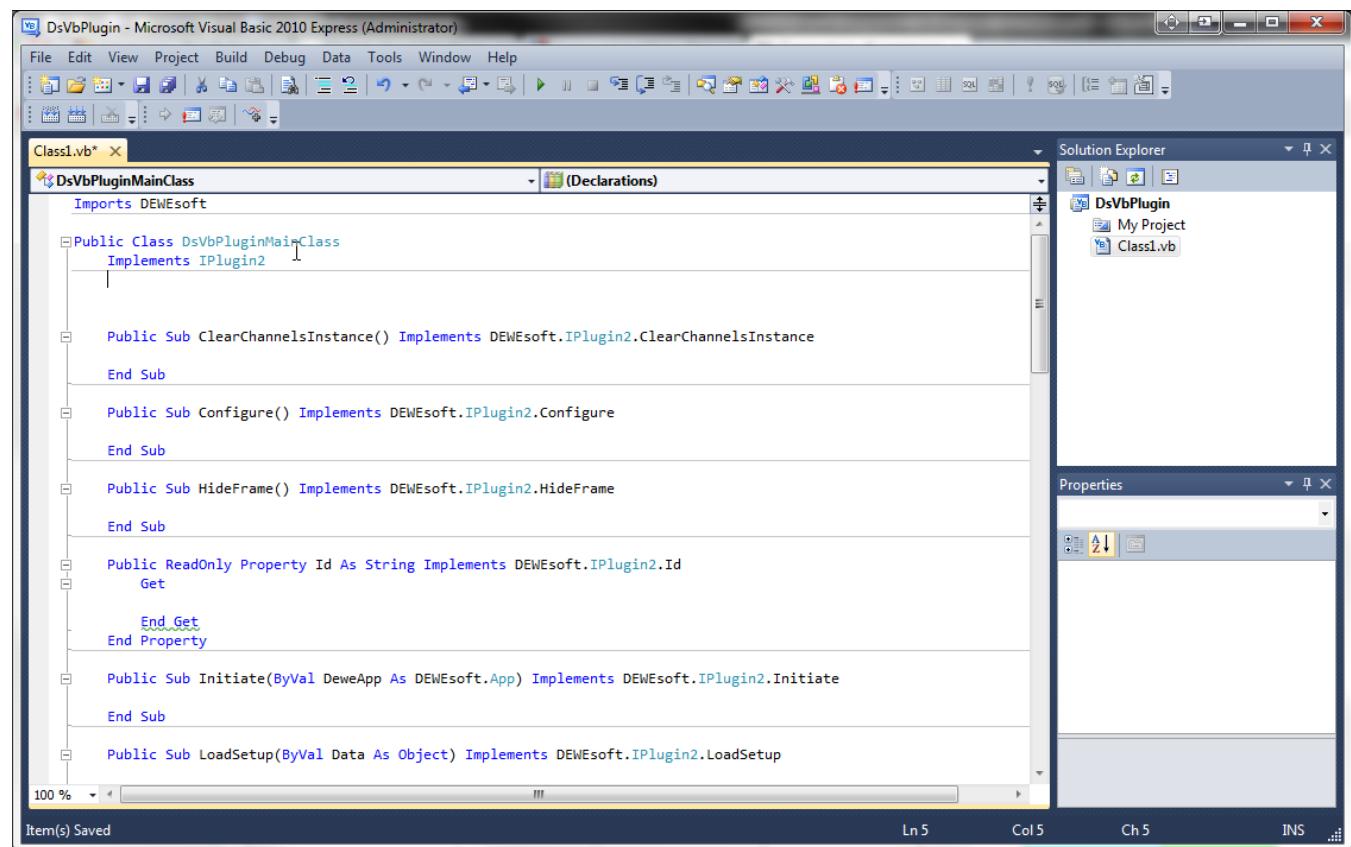


Illustration 61: dummy implementation

Keep reference for the IApp interface

After *DEWEsoft®* starts the plugin, it will call the [Initiate](#) function and pass along the [IApp](#) interface. We will assign it to a local variable called FApp. It is the main entry point for communicating with *DEWEsoft®* and we will need it in most other methods that we implement.

```
Private FApp As DEWEsoft.App

Public Sub Initiate(ByVal DeweApp As DEWEsoft.App) Implements DEWEsoft.IPlugin2.Initiate
    FApp = DeweApp
End Sub
```

Remember the Used-status

Next, we implement the [Property Used](#) which tells us, if the user has set the plugin to *Used* or *Unused* in the hardware setup.

```
Private FUsed As Boolean
Public Property Used As Boolean Implements DEWEsoft.IPlugin2.Used
    Get
        Return FUsed
    End Get
    Set(ByVal value As Boolean)
        value = FUsed
    End Set
End Property
```

Implement channel setup

Next, we implement the `Configure()` function. This is not really necessary, but we just want to check if the function is called:

```
Public Sub Configure() Implements DEWEsoft.IPlugin2.Configure
    ' will be executed when the user clicks on the Config button in
    ' the channel setup of the plugin
    ' you could show a configuration form to the user
    ' in this example we just show a message box
    MsgBox("plugin could show a config dialog")
End Sub
```

Since it is only possible for Delphi plugins to show an embedded frame in *DEWESoft®*, we must set the result of the `ShowFrame()` function to `False`. (we could show a configuration form via the `Configure()` function above.

```
    Public Function ShowFrame(ByVal Parent As Integer) As Boolean Implements DEWEsoft.IPlugin2.  
ShowFrame  
        Return False  
    End Function
```

Add a plugin channel

Finally we will add a [channel](#) and fill it with some random data.

```
Private Ch1 As DEWEsoft.IChannel

' this function will mount a channel for our plugin
' see also: ClearChannelsInstance()
```

Now that the implementation is okay, we need to add some more information to make our class library work with DCOM and DEWEsoft®: see [Visual Basic: Prepare for DCOM](#)

1.5.3.3 Visual Basic: Prepare for DCOM

Now that the plugin implementation is complete (see [Visual Basic: Implementation](#)), we need to add some more code, so that the class is recognized by *DCOM* and by *DEWEsoft®*.

For each plugin, we also need a unique [GUID](#).

While the full version of the *Visual Basic 2010 IDE* comes with an external tool called `guidgen.exe`, the Express edition lacks this program. However, you can simply use an online [GUID generator](#), like we did for this example:

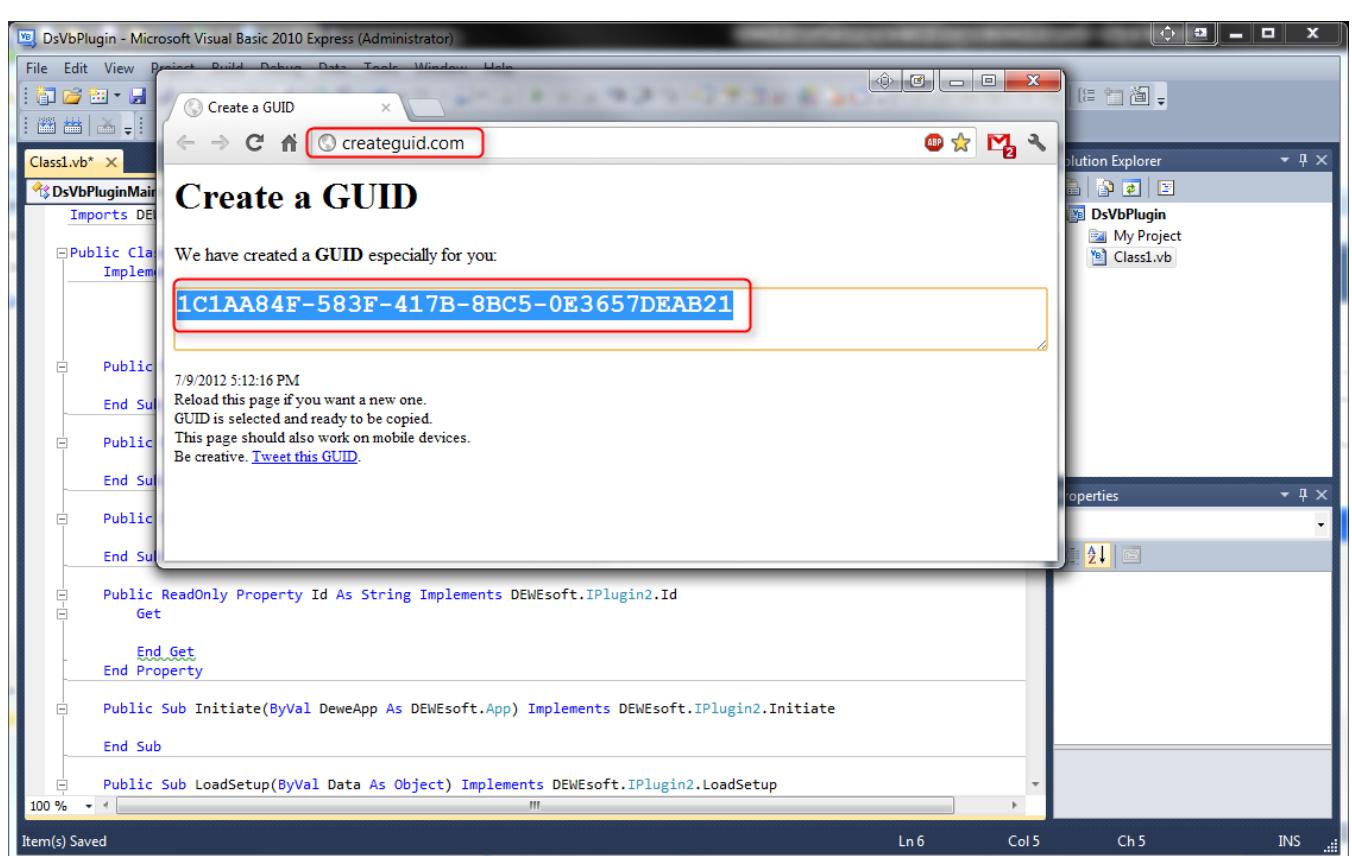


Illustration 62: Create GUID

Then just copy the [GUID](#) and use it in your code (see inline comments for details):

```

Imports DEWEsoft
Imports System.Runtime.InteropServices
Imports Microsoft.Win32

' The ComClassAttribute attribute instructs the compiler to add
' metadata that allows a class to be exposed as a COM object.
<ComClass(DsVbPluginMainClass.ClassId)>
Public Class DsVbPluginMainClass
    Implements IPPlugin2

    ' This GUIDs provides the COM identity for this class
    ' and its COM interface. If you change it, existing
    ' clients will no longer be able to access the class.
    Public Const ClassId As String = "1C1AA84F-583F-417B-8BC5-0E3657DEAB21"
    Public Const RegKeyForPlugin As String = "DsVbPlugin"

    ' A creatable COM class must have a Public Sub New()
    ' with no parameters, otherwise, the class will not be
    ' registered in the COM registry and cannot be created
    ' via CreateObject.
    Public Sub New()
        MyBase.New()
    End Sub

    ' this code will be executed during the registration process
    ' of the assembly (e.g. when "regasm.exe" is called)
    ' it will create the registry entries so that DEWEsoft® will
    ' find the plugin code
    ' see also UnregisterFunction() below
    <ComRegisterFunctionAttribute()> _
    Shared Sub RegisterFunction(ByVal t As Type)

```

Now that the source code is complete, we can build the project:

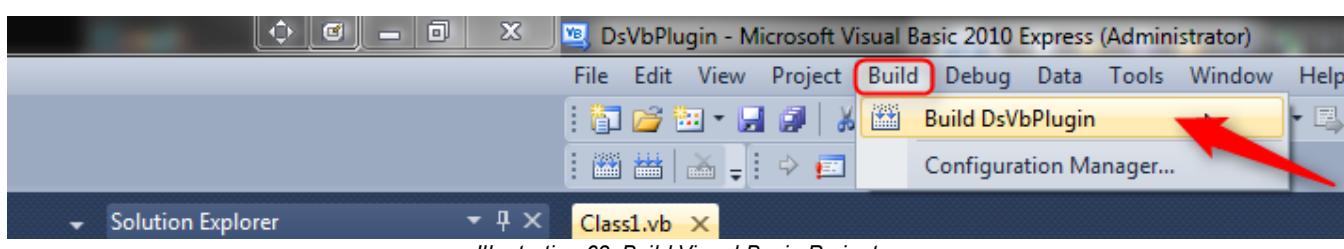


Illustration 63: Build Visual Basic Project

The Visual Basic IDE has now created a .dll file for our plugin:

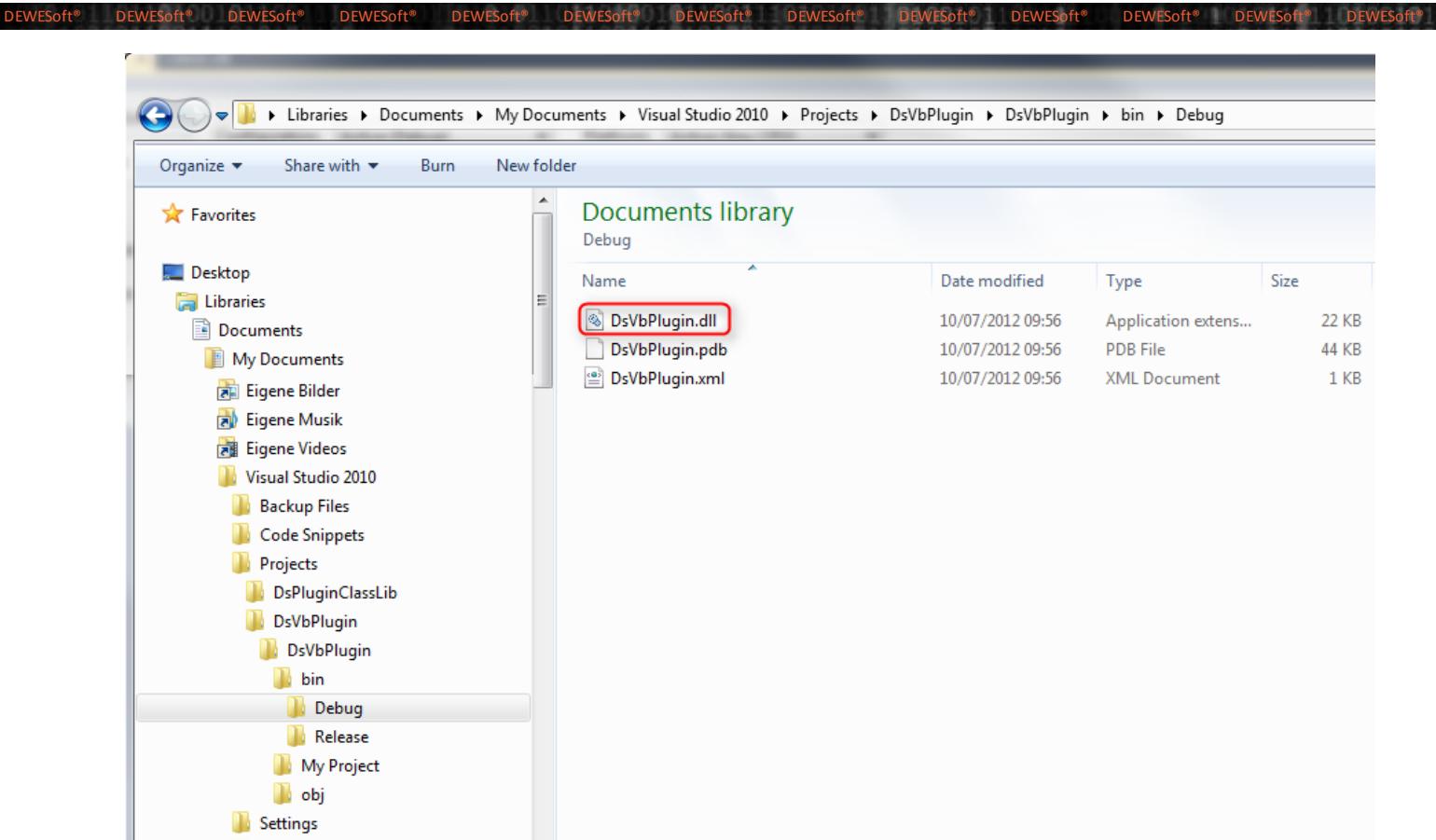


Illustration 64: Build Output

next: [Visual Basic: Register Assembly](#)

1.5.3.4 Visual Basic: Register Assembly

In order to use .NET Framework classes as COM objects, you must register the classes once with the `Regasm.exe` tool, which is included in the installation of the .NET framework.

We do this for your plugin. In the command line interpreter, we navigate to the `Addons` folder of **DEWESoft®** and start the assembly registration tool (note: the plugin library `DsVbPlugin.dll` must exist in the `Addons` directory - either copy it there, or configure your Visual Basic IDE to build it there).

```
D:\DEWESoft7\Bin\V7_0\Addons>C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.  
exe /codebase DsVbPlugin.dll
```

```
Microsoft (R) .NET Framework Assembly Registration Utility 4.0.30319.1  
Copyright (C) Microsoft Corporation 1998-2004. All rights reserved.
```

```
RegAsm : warning RA0000 : Registering an unsigned assembly with /codebase can cause  
your assembly to interfere with other applications that may be installed on the  
same computer. The /codebase switch is intended to be used only with signed  
assemblies. Please give your assembly a strong name and re-register it.
```

Types registered successfully

Note: you can ignore the warning about the unsigned assembly - it will still work. If you want to get rid of the warning, you must sign your assembly.

The Regasm.exe tool has registered the assembly for COM usage - in detail it has created some entries in the Windows registry, so that COM programs (i.e. DEWEISoft®) can use the COM classes we have defined.

```
[HKEY_CLASSES_ROOT\DsVbPlugin.DsVbPluginMainClass]
@="DsVbPlugin.DsVbPluginMainClass"

[HKEY_CLASSES_ROOT\DsVbPlugin.DsVbPluginMainClass\CLSID]
@="{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}"

[HKEY_CLASSES_ROOT\CLSID\{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}\InprocServer32]
@="DsVbPlugin.DsVbPluginMainClass"

[HKEY_CLASSES_ROOT\CLSID\{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}\InprocServer32]
@="mscoree.dll

"ThreadingModel"="Both"
"Class"="DsVbPlugin.DsVbPluginMainClass"
"Assembly"="DsVbPlugin, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"
"RuntimeVersion"="v4.0.30319"
"CodeBase"="file:///D:/DEWESoft7/Bin/V7_0/Addons/DsVbPlugin.dll"

[HKEY_CLASSES_ROOT\CLSID\{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}\InprocServer32
\1.0.0.0]
"Class"="DsVbPlugin.DsVbPluginMainClass"
"Assembly"="DsVbPlugin, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"
"RuntimeVersion"="v4.0.30319"
"CodeBase"="file:///D:/DEWESoft7/Bin/V7_0/Addons/DsVbPlugin.dll"

[HKEY_CLASSES_ROOT\CLSID\{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}\ProgId]
@="DsVbPlugin.DsVbPluginMainClass"

[HKEY_CLASSES_ROOT\CLSID\{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}\Implemented
Categories]\{62C8EE65-4FB4-45E7-B440-6E39B2CD8E2911}
```

During its execution, the Regasm.exe tool has also executed the RegisterFunction function with the ComRegisterFunctionAttribute attribute (see [Visual Basic: Prepare for DCOM](#)), which will be read by DEWEsoft® to find the plugin:

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\DeWeSoft\Plugins\DsVbPlugin]

```
"GUID"="{1C1AA84F-583F-417B-8BC5-0E3657DEAB21}"  
"Description"="VB Express 2010 test"  
"Name"="VBE 2012"  
"Version"="1.0"  
"Vendor"="Dewesoft"  
"TLB"=dword:00000005  
"NET"=dword:00000001
```

Note: that the registration information for *DEWESoft®* refers via the [GUID](#) (highlighted in green in the text above) to the COM object for our plugin.

Note: the key `Wow6432Node` (highlighted in blue) will be inserted on 64-bit Windows systems automatically - On 32-bit Windows systems the key `Wow6432Node` will not be present.

see also: [Finding Plugins](#)

That's it: we can now start *DEWESoft®* and test the plugin: [Visual Basic: Test the Plugin](#)

1.5.3.4.1 Visual Basic: Signing the assembly

To sign your assembly (give it a strong name), go to [Project - DsVbPlugin Properties](#):

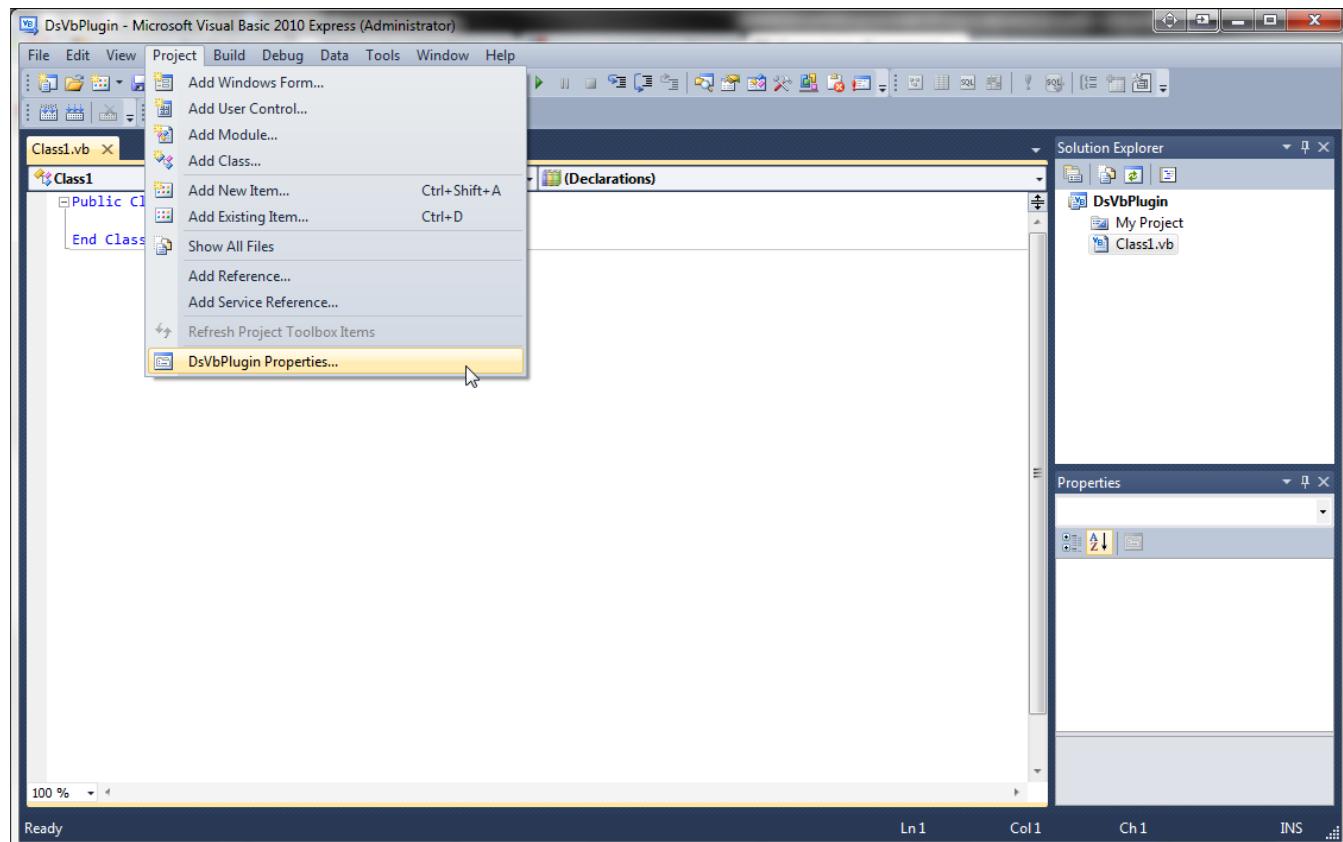


Illustration 65: Project Properties

Then select the ***Siging*** section on the left, check the ***Sign the assembly*** check-box and choose **<New...>** from the drop-down list named '*Choose a strong name key file*'. This will open the '*Create Strong Name Key*' dialogue. Now enter a name for your key file (e.g. **DsVbPluginKey**) and optionally enter a password.

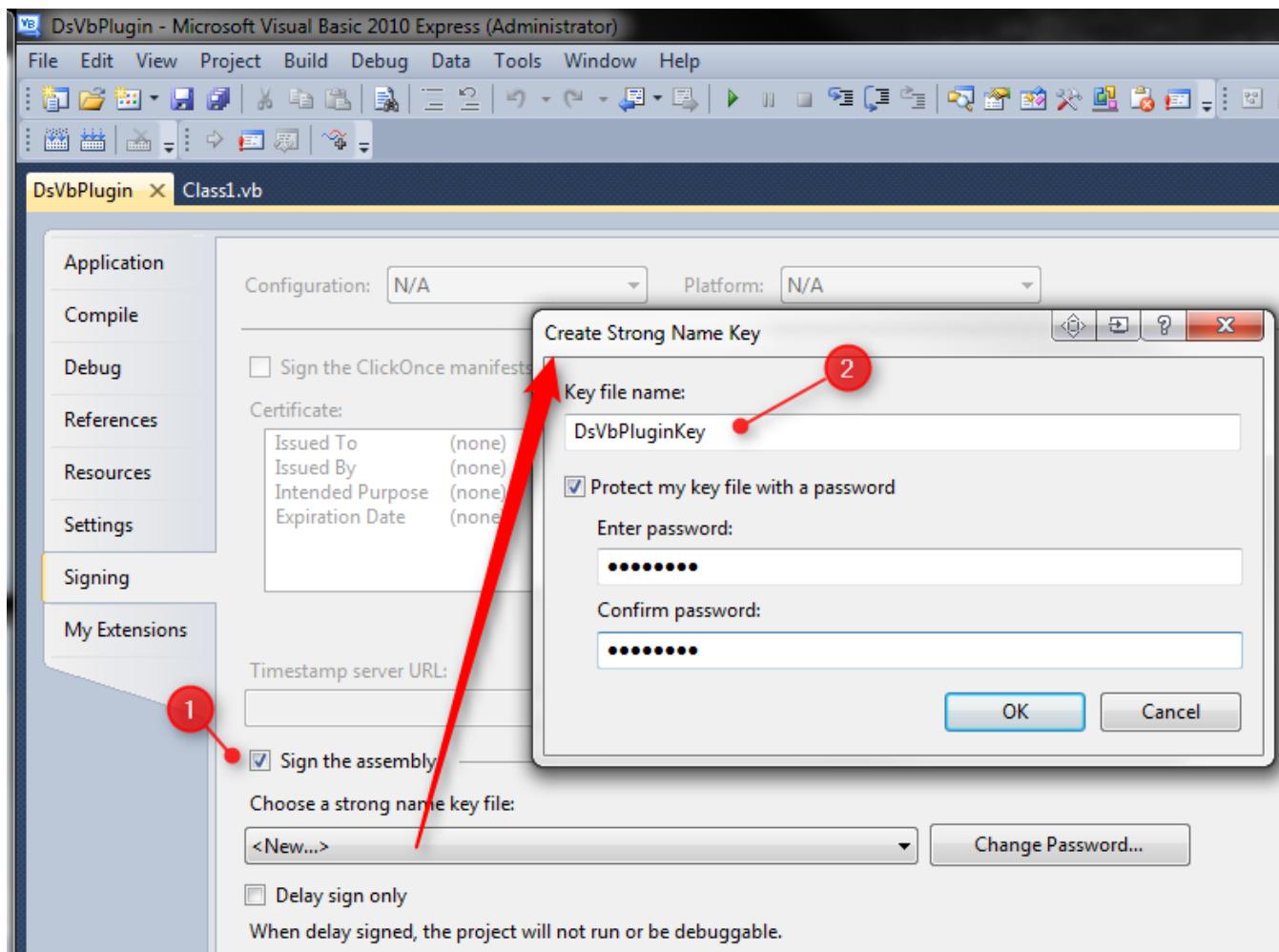


Illustration 66: Signing the assembly

Then click okay and build the solution. Now the registration process will work without any warning:

```
D:\DEWEsoft7\Bin\V7_0\Addons>C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe /codebase DsVbPlugin.dll
Microsoft (R) .NET Framework Assembly Registration Utility 4.0.30319.1
Copyright (C) Microsoft Corporation 1998-2004. All rights reserved.

Types registered successfully
```

D:\DEWEsoft7\Bin\V7_0\Addons>

see also: [Visual Basic: Register Assembly](#)

1.5.3.5 Visual Basic: Test the Plugin

After we have successfully registered the assembly (see [Visual Basic: Register Assembly](#)), we can finally start up **DEWESoft®** and test our Visual Basic plugin.

When you go to the *Plugins* tab sheet of the hardware setup, the Plugin will already show up:

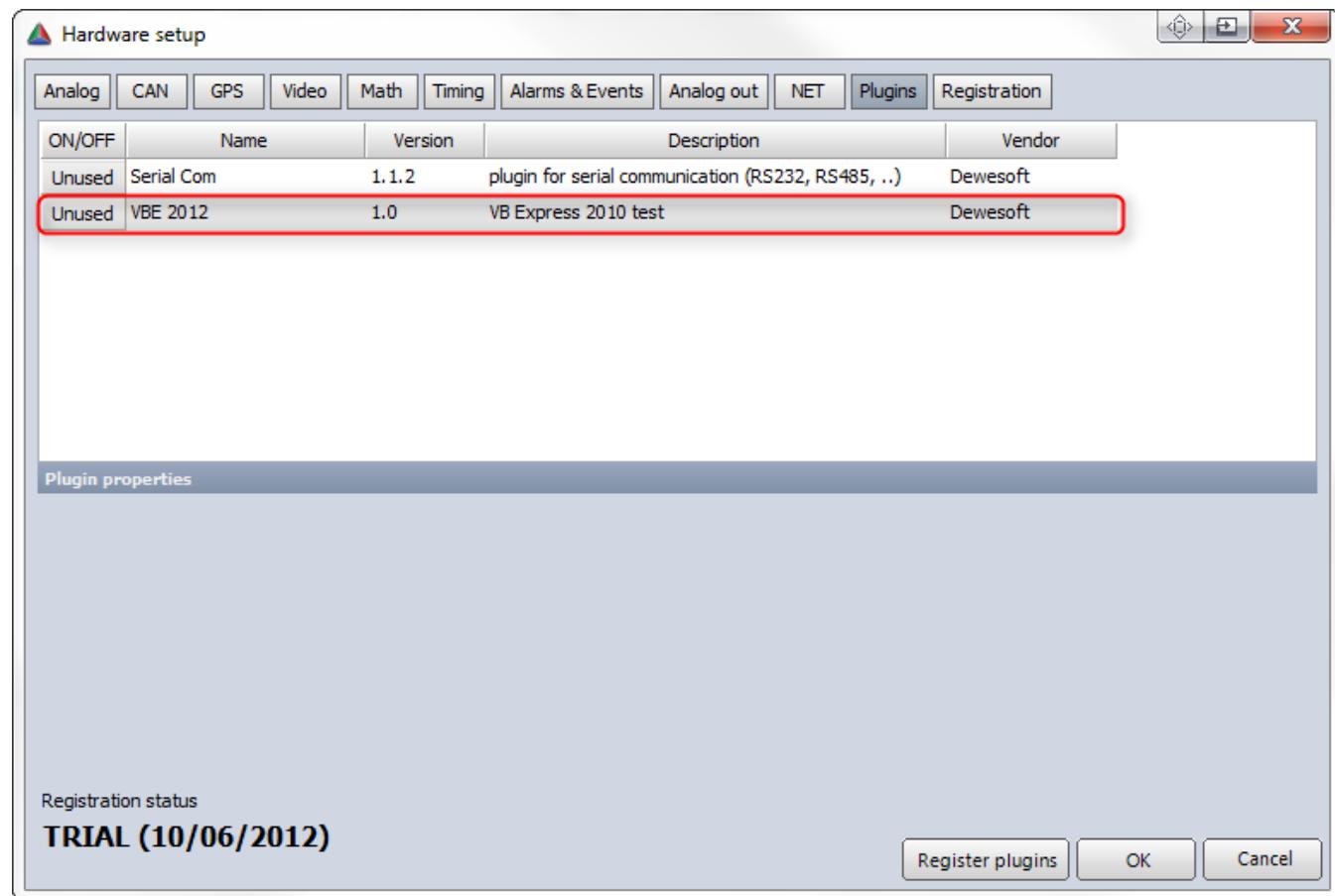


Illustration 67: Plugin shows up in DEWESoft™

Click the *Unused* button to set the plugin to *Used* and then you can already see the plugin in the channel setup:

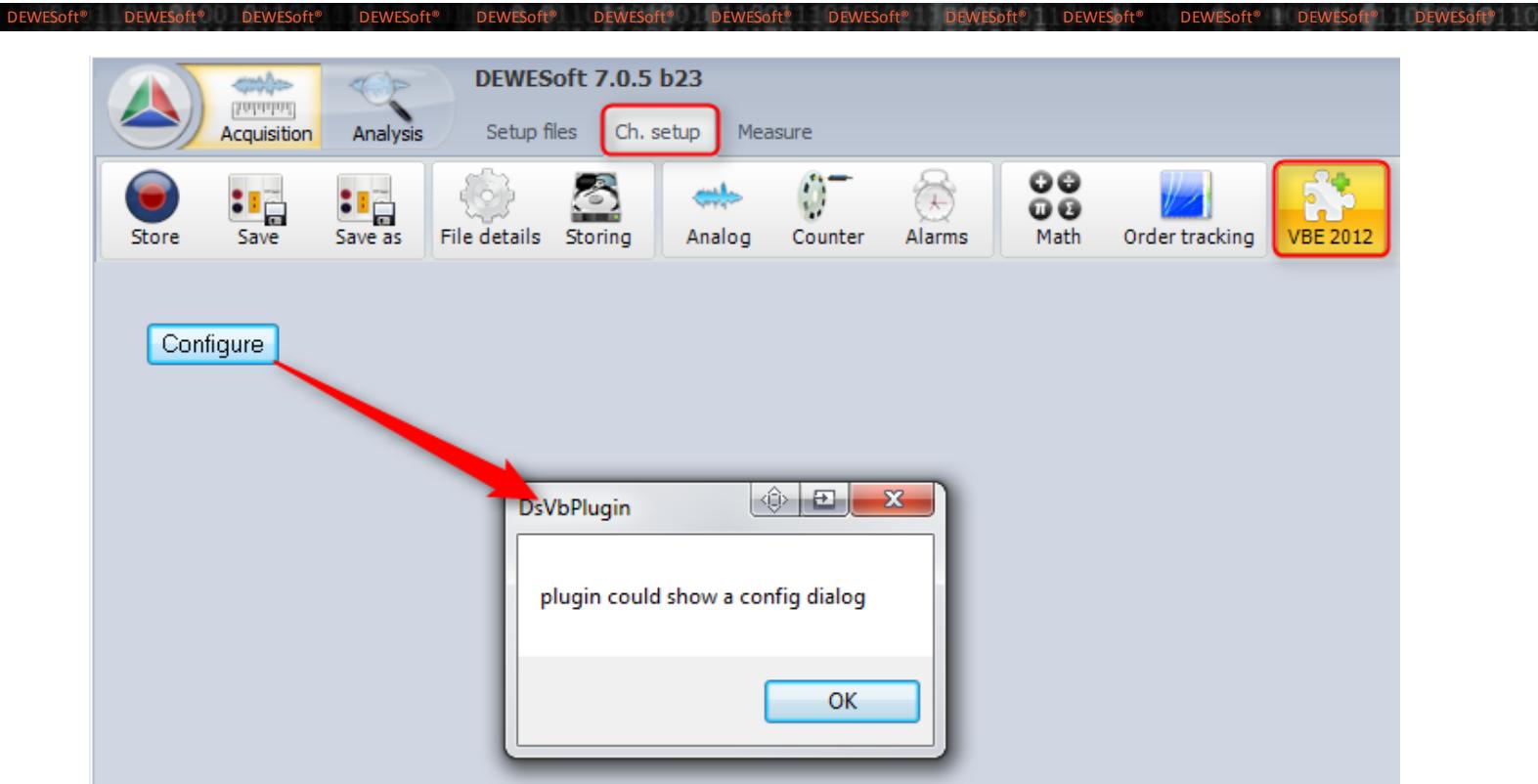


Illustration 68: Channel Setup

When you press the [Configure](#) button, the message box that we have defined in the `Configure()` function (see [Visual Basic: Implementation](#)) will show up.

When we switch to [Measure](#) mode, we can see that the channel (called *Test Channel 1*) that we have mounted (see `function MountChannels()` in [Visual Basic: Implementation](#)) shows up in the channel list (see red rectangle on the right side in the screenshot below) and that the random data is shown in the recorder:

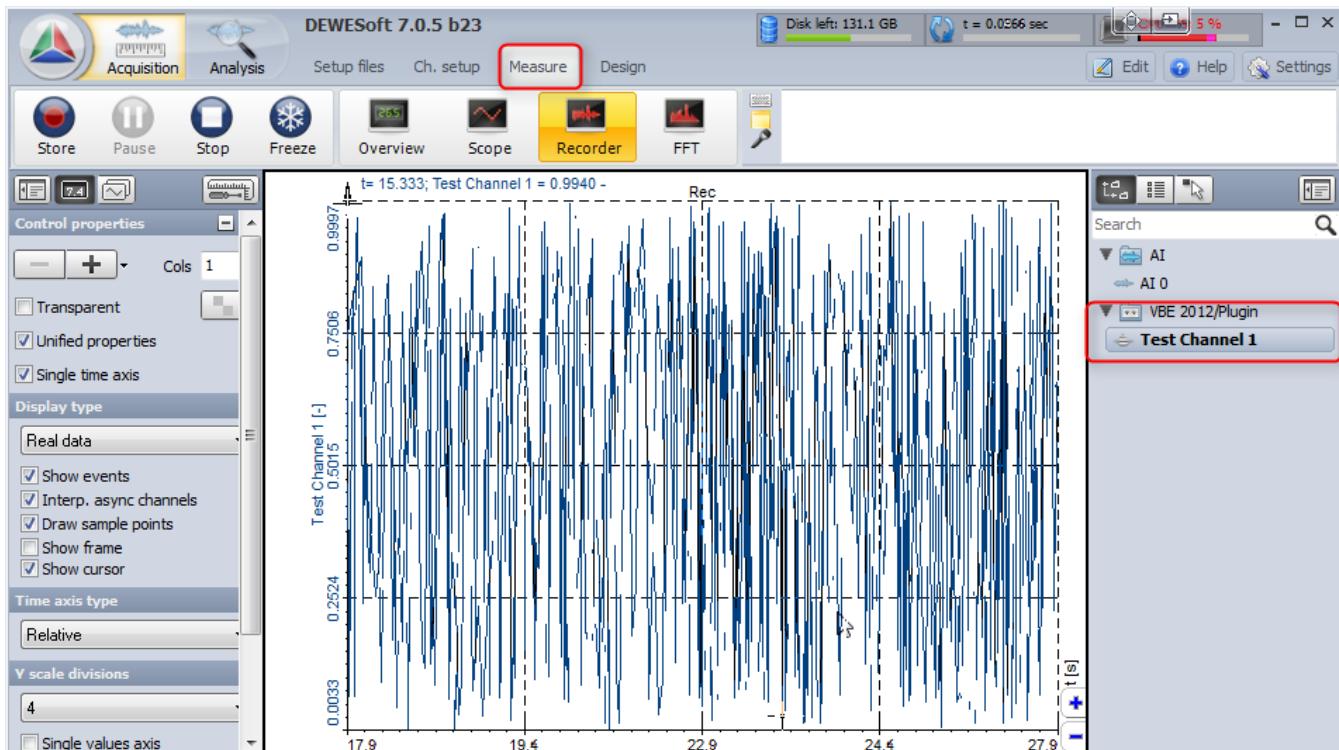


Illustration 69: Measure mode

see also: [Visual Basic: Troubleshooting](#), [Visual Basic: Sourcecode](#)

1.5.3.6 Visual Basic: Troubleshooting

The system cannot find the file specified.

The following error-message may occur when try to activate the plugin in hardware setup:

Error initializing VBE 2012 plugin: The system cannot find the file specified.

Note: VBE 2012 is the name of the plugin.

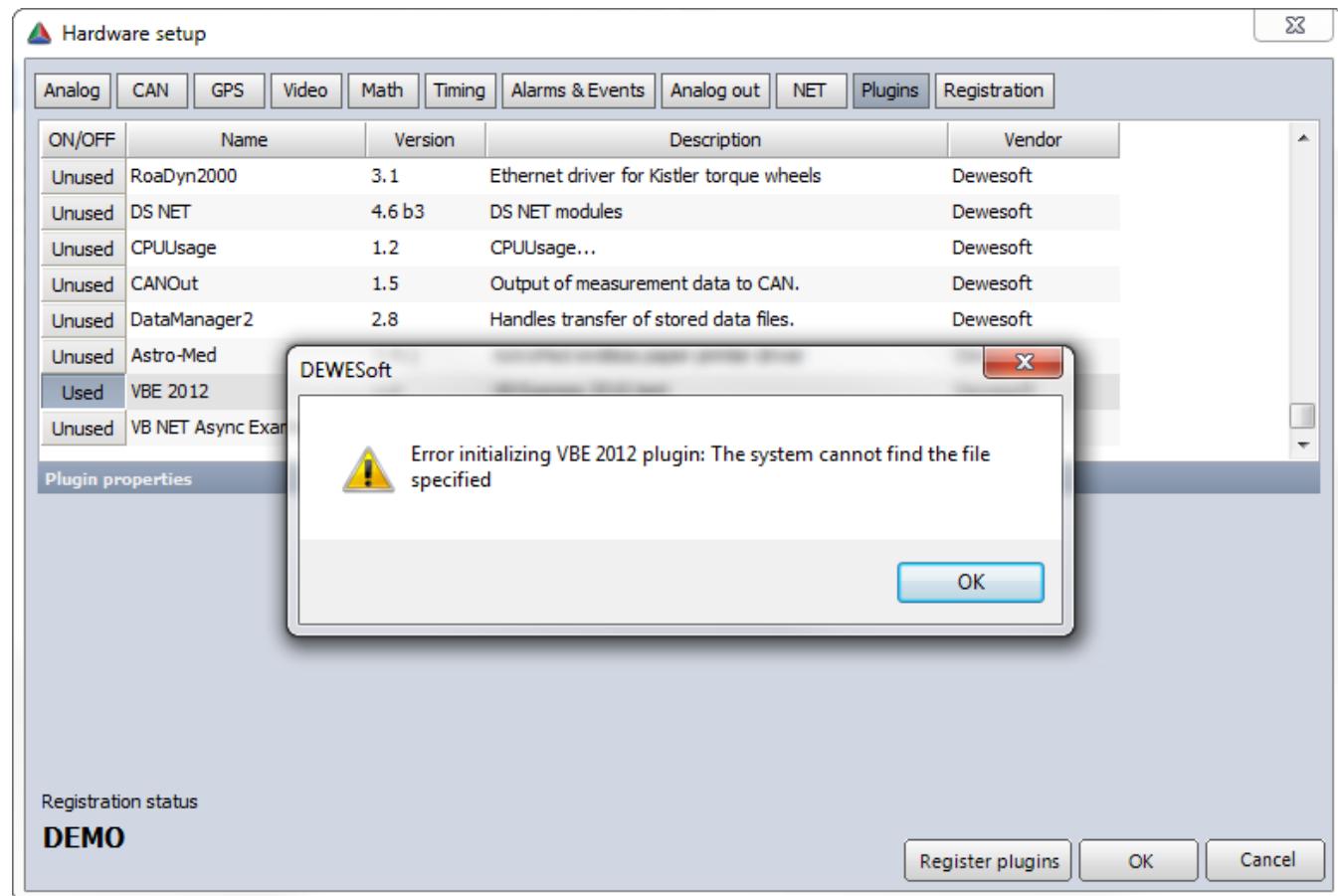


Illustration 70: Visual Basic - wrong registration

Possible causes of the error:

- check if you have used the /codebase switch for the Regasm.exe tool: see [Visual Basic: Register Assembly](#)
 - check if your plugin has dependencies on other libraries - if so, you may statically link those libraries to your plugin

1.5.3.7 Visual Basic: Sourcecode

The complete source code of our Visual Basic example plugin:

```

Imports DEWEsoft
Imports System.Runtime.InteropServices
Imports Microsoft.Win32

' The ComClassAttribute attribute instructs the compiler to add
' metadata that allows a class to be exposed as a COM object.
<ComClass(DsVbPluginMainClass.ClassId)>
Public Class DsVbPluginMainClass
    Implements IPPlugin2

#Region "DCOM"
    ' This GUIDs provides the COM identity for this class
    ' and its COM interface. If you change it, existing
    ' clients will no longer be able to access the class.
    Public Const ClassId As String = "1C1AA84F-583F-417B-8BC5-0E3657DEAB21"
    Public Const RegKeyForPlugin As String = "DsVbPlugin"

    ' A creatable COM class must have a Public Sub New()
    ' with no parameters, otherwise, the class will not be
    ' registered in the COM registry and cannot be created
    ' via CreateObject.
    Public Sub New()
        MyBase.New()
    End Sub

    ' this code will be executed during the registration process
    ' of the assembly (e.g. when "regasm.exe" is called)
    ' it will create the registry entries so that DEWEsoft™ will
    ' find the plugin code
    ' see also UnregisterFunction() below
    <ComRegisterFunctionAttribute()> _
    Shared Sub RegisterFunction(ByVal t As Type)
        Dim strId As String
        strId = "{" + t.GUID.ToString + "}"
        strId = UCASE(strId)
        Try
            Dim regHkLM As RegistryKey
            regHkLM = Registry.LocalMachine
            regHkLM.CreateSubKey("SOFTWARE\\Dewesoft\\Plugins\\" + RegKeyForPlugin)
            regHkLM.OpenSubKey("SOFTWARE\\Dewesoft\\Plugins\\" + RegKeyForPlugin)
            regHkLM.SetValue("GUID", strId)
            regHkLM.SetValue("Description", "VB Express 2010 test")
            regHkLM.SetValue("Name", "VBE 2012")
            regHkLM.SetValue("Version", "1.0")
            regHkLM.SetValue("Vendor", "Dewesoft")
            regHkLM.SetValue("TLB", 5)
            regHkLM.SetValue("NET", 1)

        Catch e As Exception
        End Try
    End Sub

    ' this code will be executed during the unregistration process
    ' of the assembly (e.g. when "regasm.exe /unregister" is called)
    ' it will delete all registry entries that RegisterFunction()
    ' has created
    ' see also RegisterFunction() above

```

```
<ComUnregisterFunctionAttribute()> _
Shared Sub UnregisterFunction(ByVal t As Type)
    Dim regHKLM As RegistryKey
    regHKLM = Registry.LocalMachine
    regHKLM.DeleteSubKey("SOFTWARE\\Dewesoft\\Plugins\\" + RegKeyForPlugin)
End Sub

Public ReadOnly Property Id As String Implements DEWEsoft.IPlugin2.Id
    Get
        Return ClassId
    End Get
End Property

#End Region

#Region "Variables"
Private FApp As DEWEsoft.App
Private FUsed As Boolean
Private Ch1 As DEWEsoft.IChannel
#End Region

' this function will mount a channel for our plugin
' see also: ClearChannelsInstance()
Private Sub MountChannels()
    Dim PluginGroup As DEWEsoft.IPluginGroup
    PluginGroup = FApp.Data.Groups(8)      ' channel group with index 8 is for plugins
    Err.Clear()
    On Error GoTo errorhandler

    ' data type 5 is Single precision
    Ch1 = PluginGroup.MountChannel(5, True, -1)
    Ch1.Name = "Test Channel 1"
    Ch1.Used = True

    PluginGroup = Nothing
    Exit Sub
End Sub

errorhandler:
    MsgBox("Error in MountChannels()")
End Sub

' ClearChannelsInstance will clear all references to the channels that
' the plugin has mounted in the MountChannel() method
Public Sub ClearChannelsInstance() Implements DEWEsoft.IPlugin2.ClearChannelsInstance
    Ch1 = Nothing

    GC.Collect()
    GC.WaitForPendingFinalizers()
End Sub

Public Sub LoadSetup(ByVal Data As Object) Implements DEWEsoft.IPlugin2.LoadSetup
    MountChannels()
End Sub

Public Sub NewSetup() Implements DEWEsoft.IPlugin2.NewSetup
    MountChannels()
End Sub

Public Sub OnGetData() Implements DEWEsoft.IPlugin2.OnGetData
    Dim Timestamp As Double
    Timestamp = FApp.MasterClock.GetCurrentTime
    Ch1.AddAsyncSingleSample(Rnd, Timestamp)
End Sub
```

```

Public Sub Configure() Implements DEWEsoft.IPlugin2.Configure
    ' will be executed when the user clicks on the Config button in
    ' the channel setup of the plugin
    ' you could show a configuration form to the user
    ' in this example we just show a message box
    MsgBox("plugin could show a config dialog")
End Sub

Public Sub HideFrame() Implements DEWEsoft.IPlugin2.HideFrame
End Sub

Public Sub Initiate(ByVal DeweApp As DEWEsoft.App) Implements DEWEsoft.IPlugin2.Initiate
    FApp = DeweApp
End Sub

Public Sub OnOleMsg(ByVal Msg As Integer, ByVal Param As Integer) Implements DEWEsoft.IPlugin2.OnOleMsg
End Sub

Public Sub OnStartAcq() Implements DEWEsoft.IPlugin2.OnStartAcq
End Sub

Public Sub OnStartStoring() Implements DEWEsoft.IPlugin2.OnStartStoring
End Sub

Public Sub OnStopAcq() Implements DEWEsoft.IPlugin2.OnStopAcq
End Sub

Public Sub OnStopStoring() Implements DEWEsoft.IPlugin2.OnStopStoring
End Sub

Public Sub OnTrigger(ByVal Time As Double) Implements DEWEsoft.IPlugin2.OnTrigger
End Sub

Public Sub SaveSetup(ByRef Data As Object) Implements DEWEsoft.IPlugin2.SaveSetup
End Sub

Public Function ShowFrame(ByVal Parent As Integer) As Boolean Implements DEWEsoft.IPlugin2.ShowFrame
    Return False
End Function

Public Sub UpdateFrame() Implements DEWEsoft.IPlugin2.UpdateFrame
End Sub

Public Property Used As Boolean Implements DEWEsoft.IPlugin2.Used
    Get
        Return FUsed
    End Get
    Set(ByVal value As Boolean)
        value = FUsed
    End Set
End Property

```

```
    End Property  
End Class
```

1.5.4 Visual C++

This example shows how to build a simple plugin using Microsoft Visual C++ 2010:

- [Visual C++: Prepare Project](#)
 - [Visual C++: Implementation](#)
 - [Visual C++: Prepare for DCOM](#)
 - [Visual C++: Register Plugin](#)
 - [Visual C++: Test the Plugin](#)
 - [Visual C++: Troubleshooting](#)
 - [Visual C++: Sourcecode](#)

1.5.4.1 Visual C++: Prepare Project

Create the project

Goto *File* and select *New - Project...*

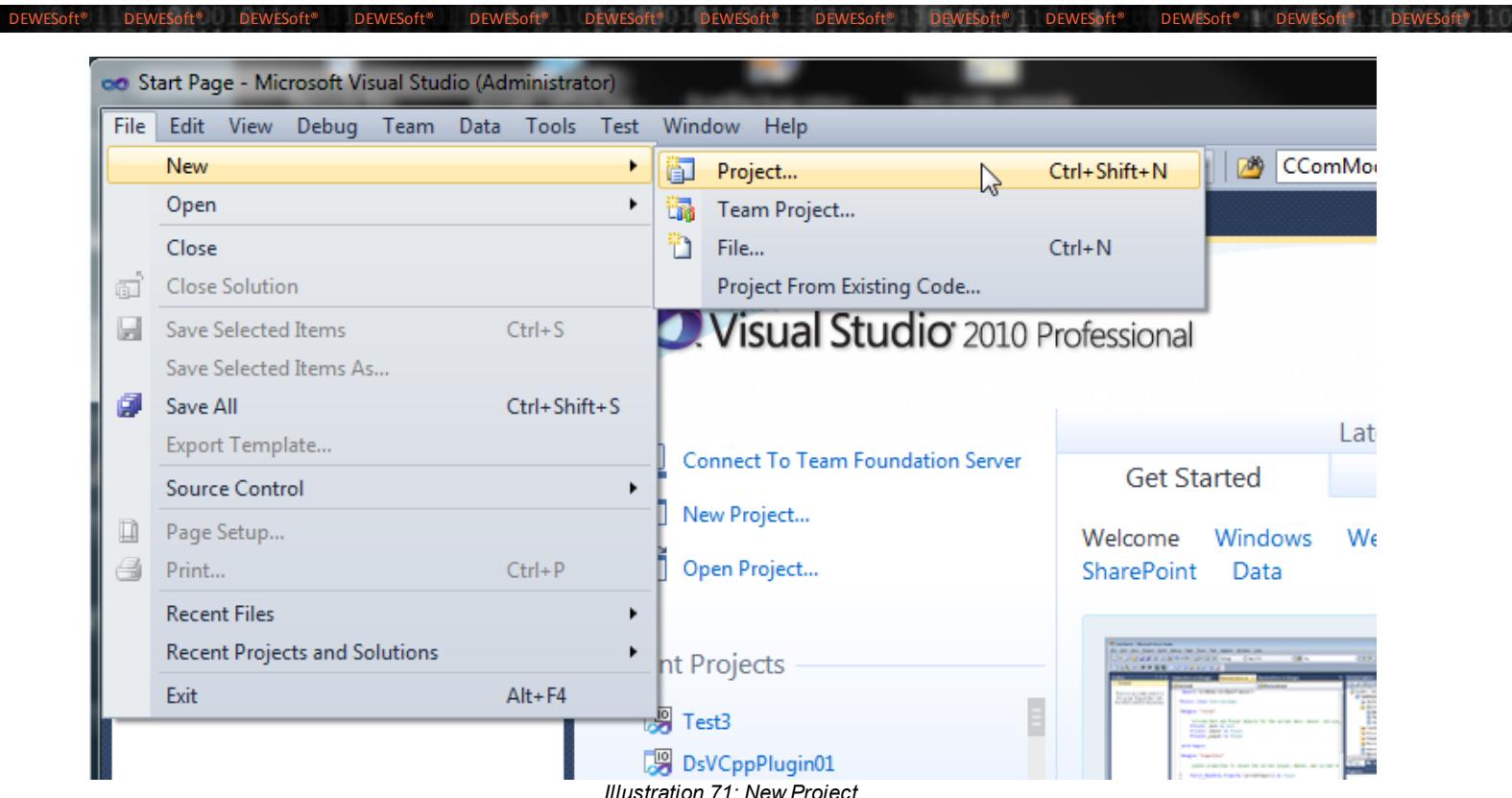


Illustration 71: New Project

Select [ATL Project](#) project and enter a name (e.g. *DsCppPlugin*):

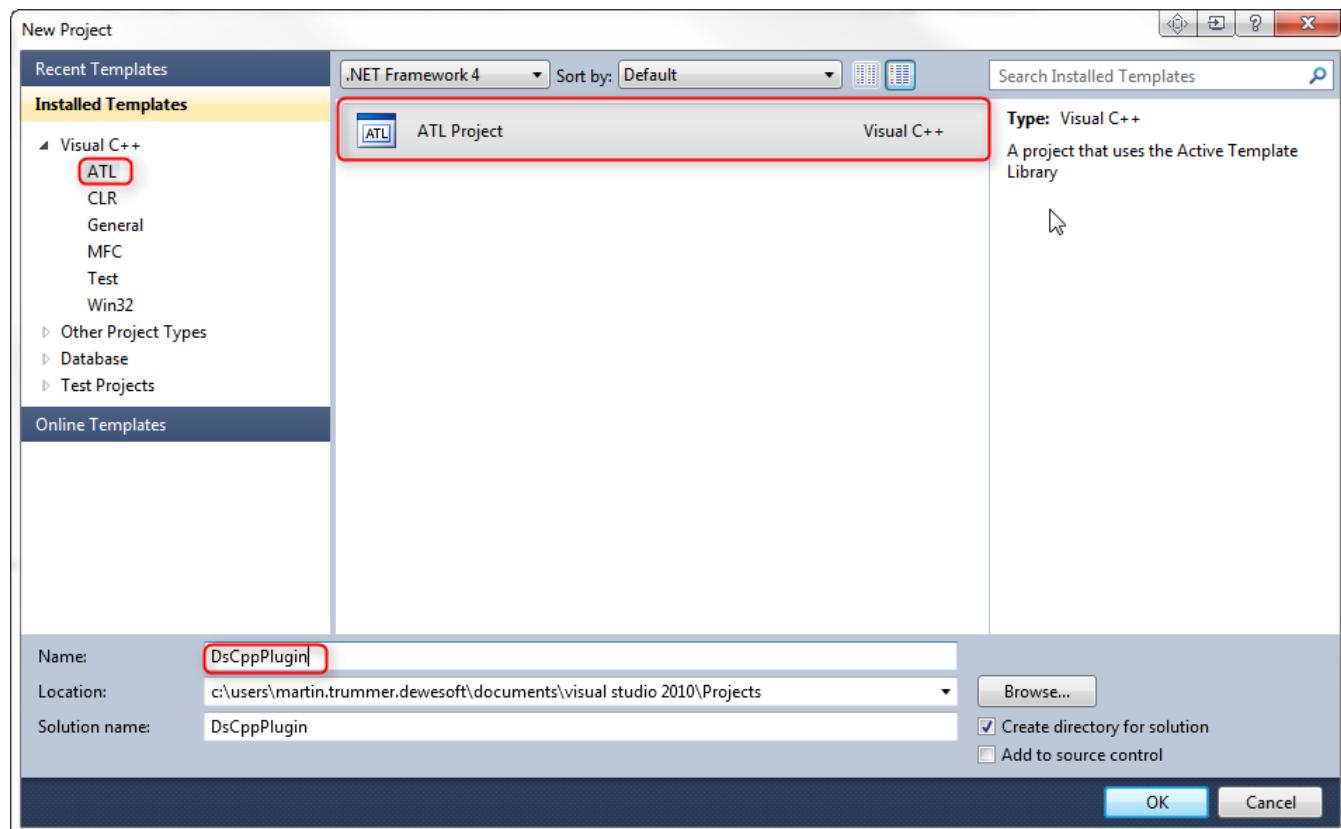


Illustration 72: Create ATL Project

Then click [Finish](#) in the *ATL Project Wizard* to create the project:

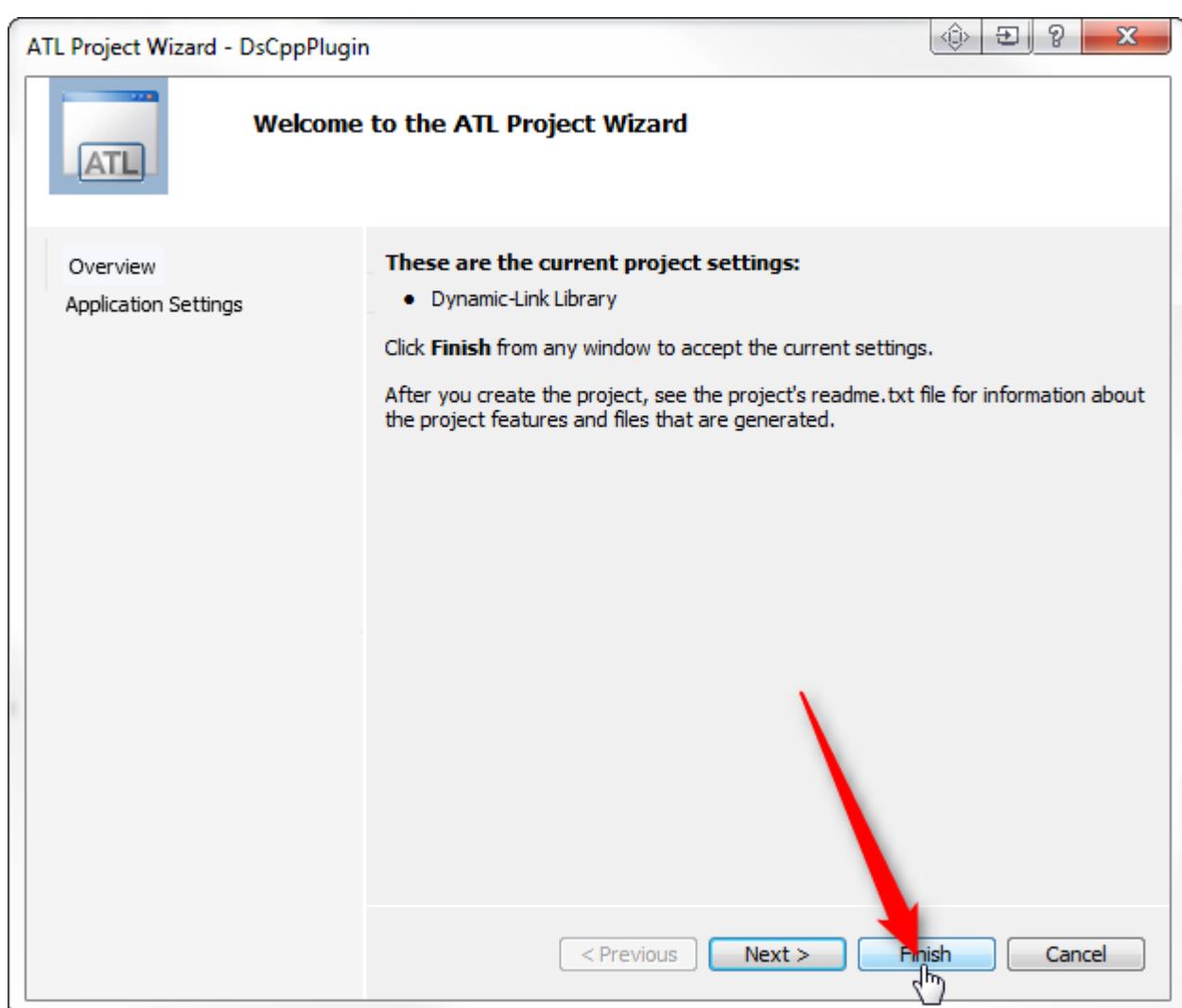


Illustration 73: Create Project

Now, go to *Project - Properties*:

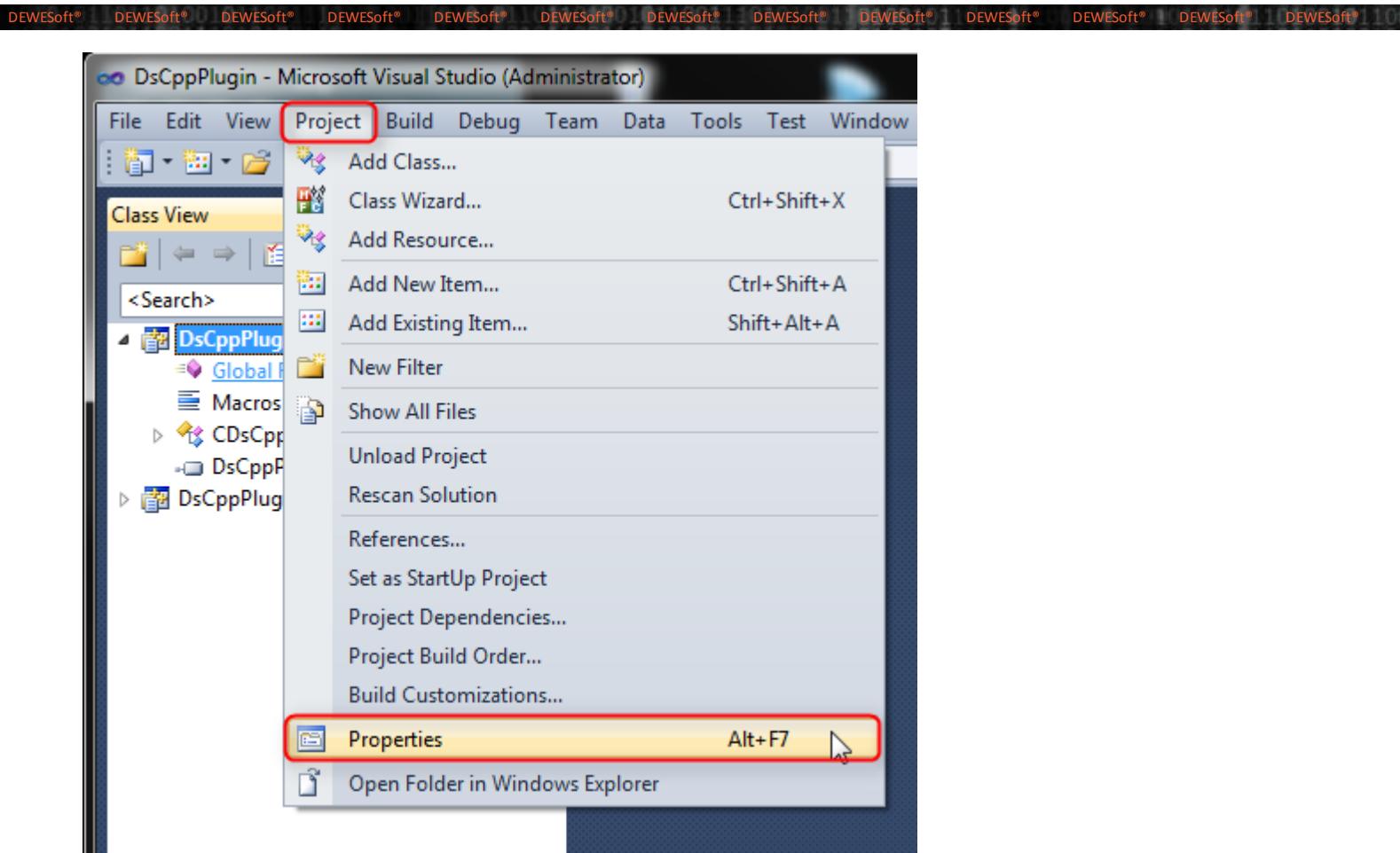


Illustration 74: Project Properties

In the *General* section of the project properties, change the *Output Directory* to the *Addons* folder of your DEWESoft® installation (e.g. D:\DEWESoft7\Bin\V7_0\Addons).

Note: don't forget the trailing backslash.

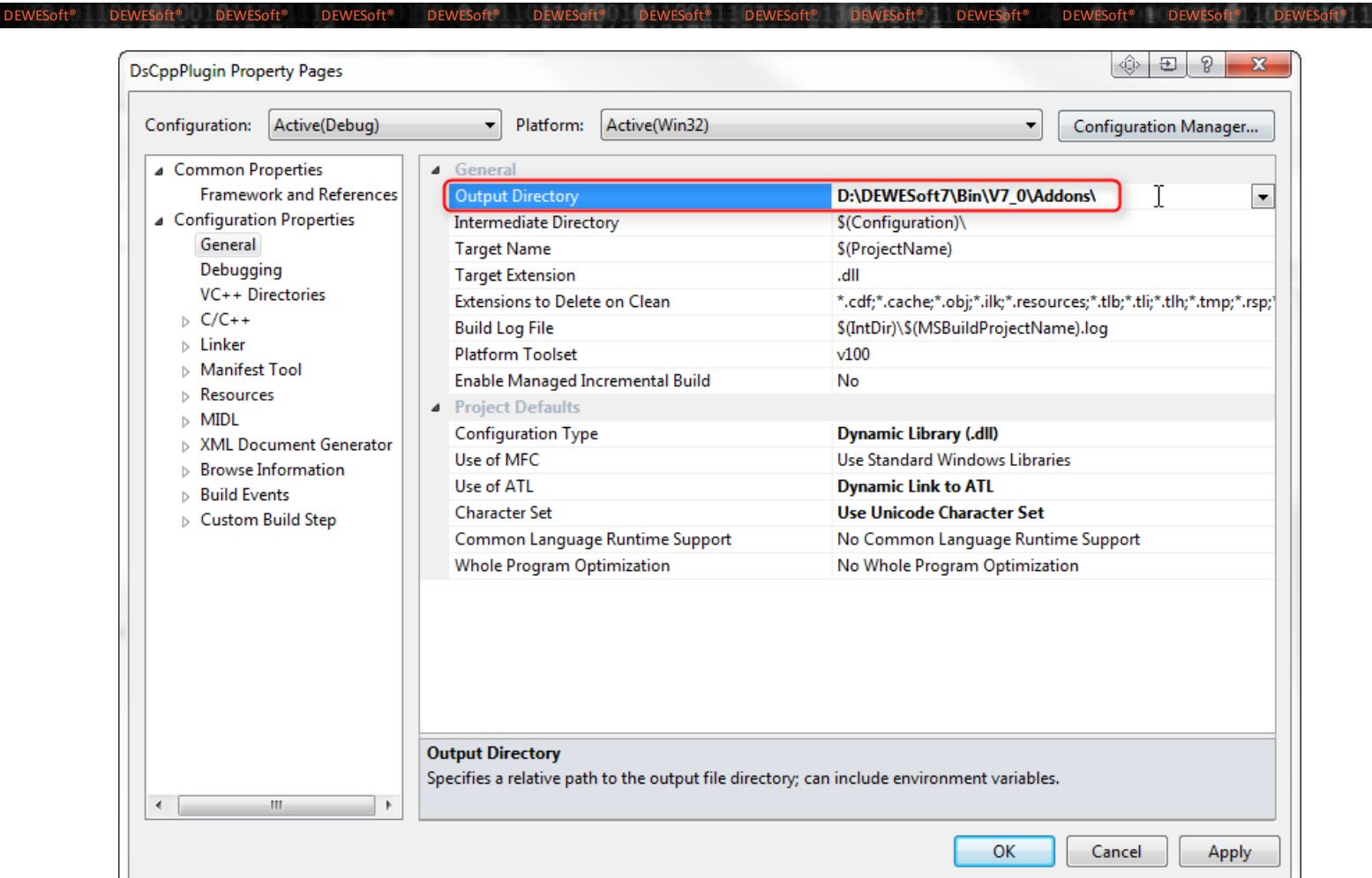


Illustration 75: Output Directory

The plugin implementation class

Select *Project - Add Class...*:

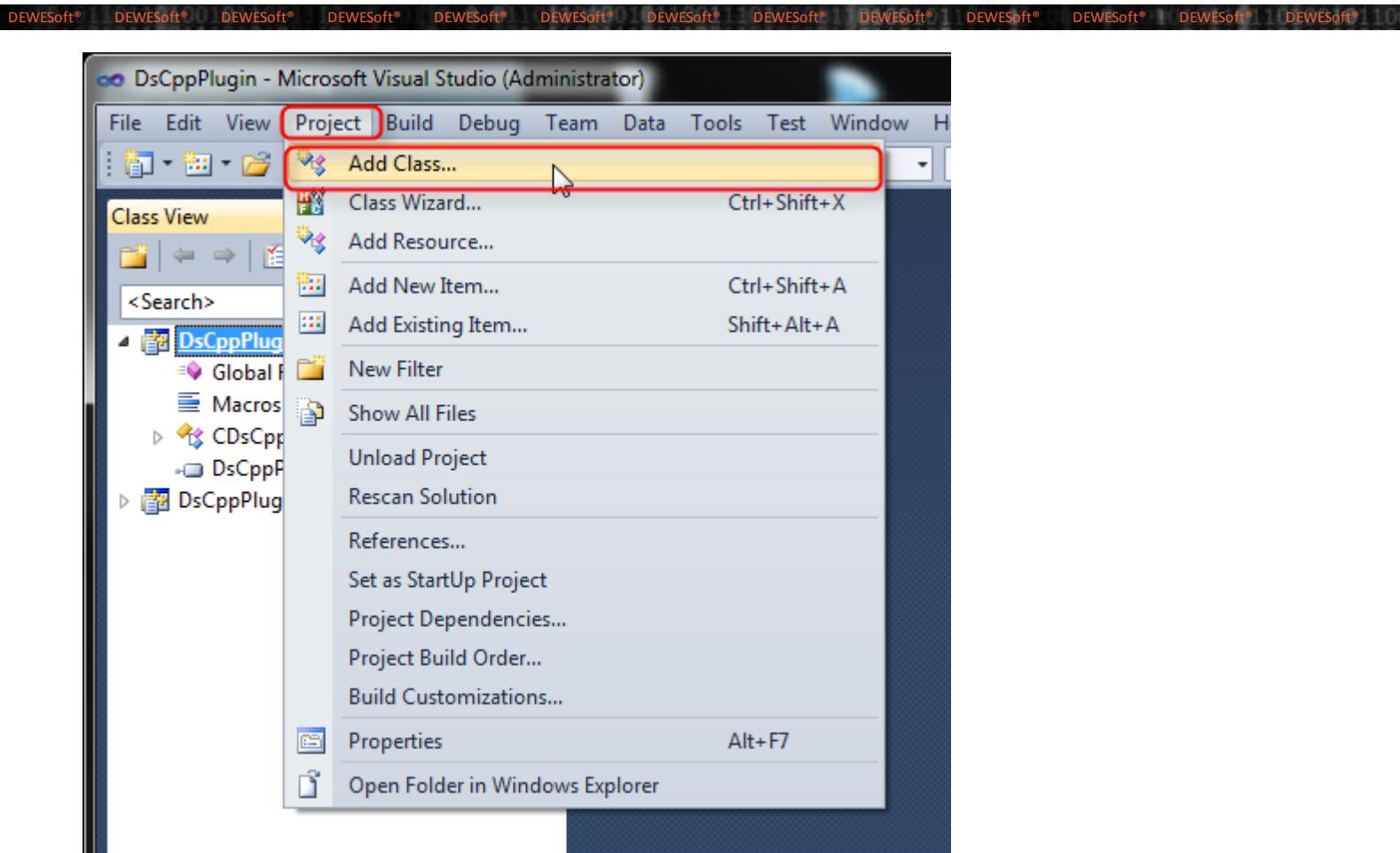


Illustration 76: Add Class

Then select *ATL Simple Object* and click the *Add* button:

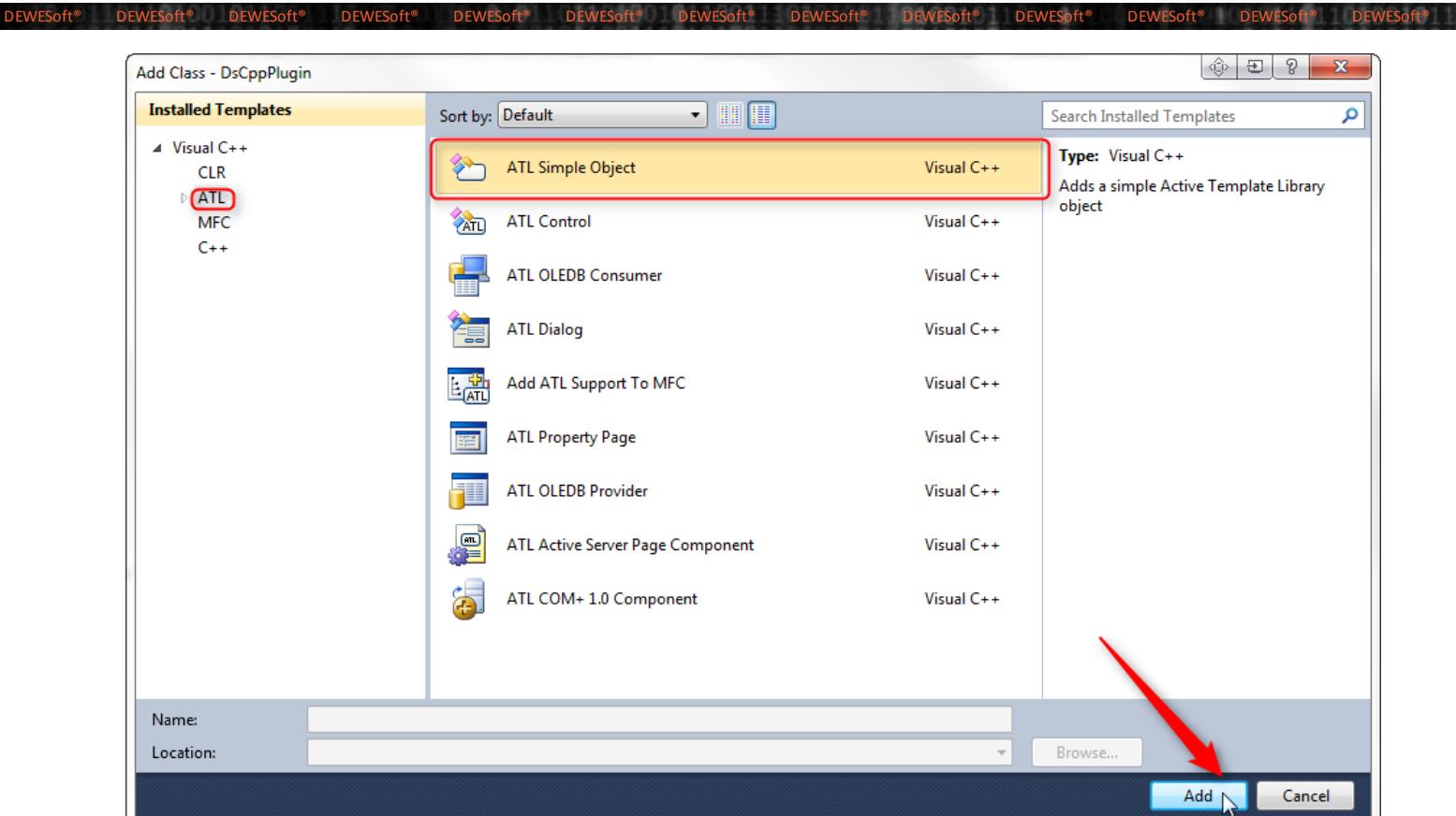


Illustration 77: ATL Simple Object

Give a meaningful name to the implementation class (e.g. `VCppPluginImpl`) and click the [Finish](#) button:

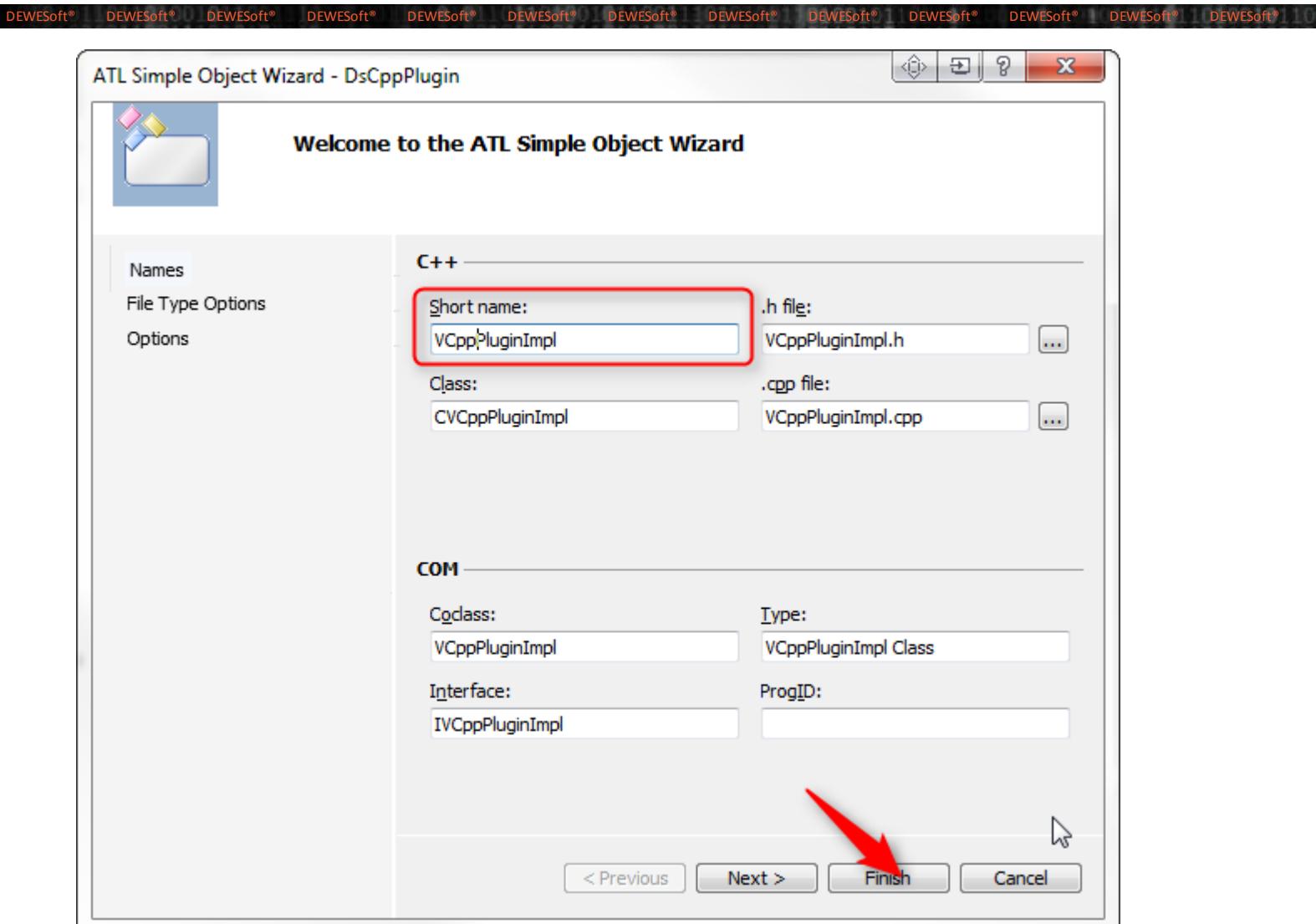


Illustration 78: ATL Simple Object Wizard

Next, we add the DEWESoft® DCOM interface [IPlugin2](#) to our plugin implementation class:

Right-click the class *CVCppPluginImpl* in the *Class View* and select *Add - Implement Interface...* from the pop-up menu.

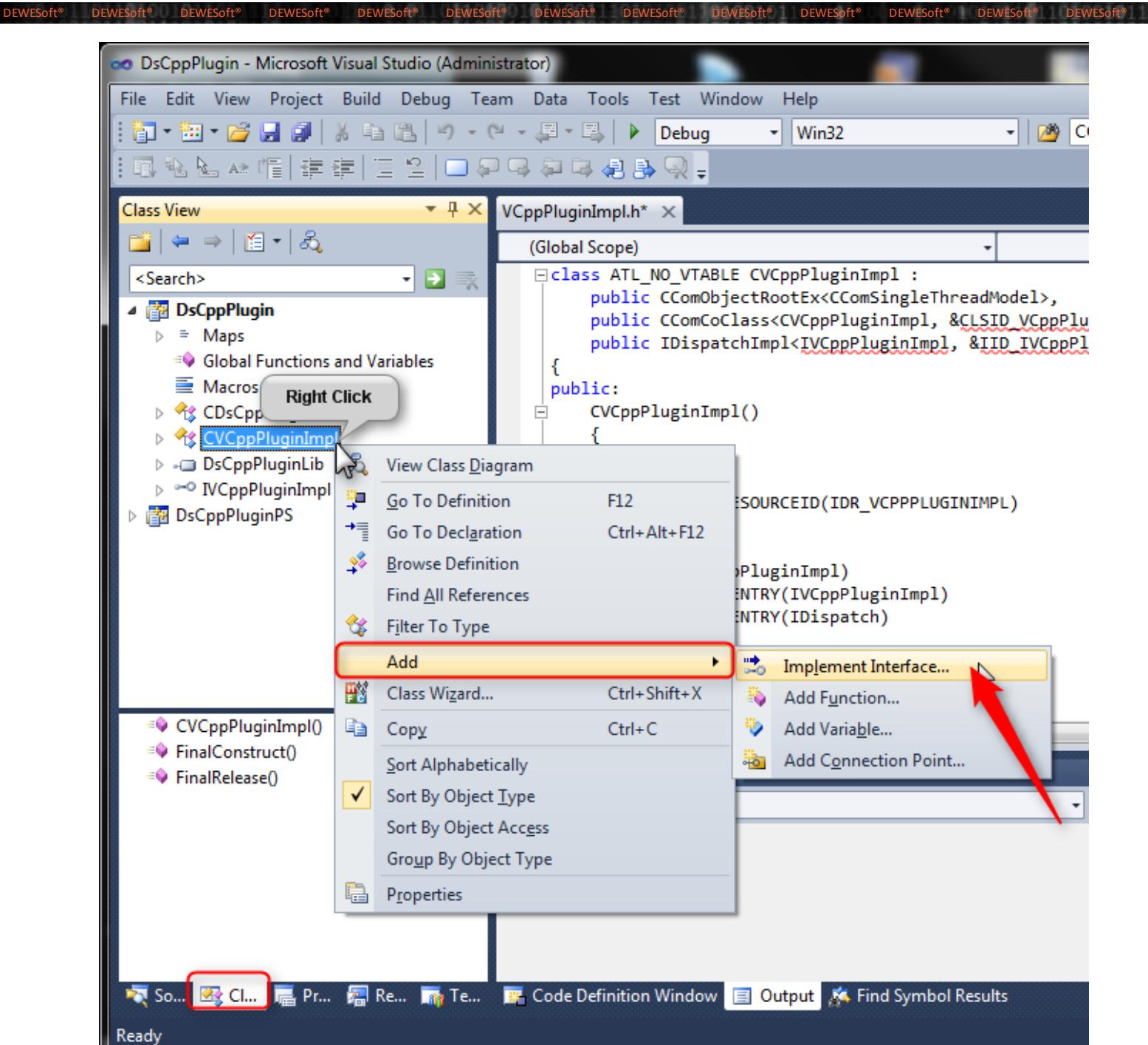


Illustration 79: Implement Interface

In the *Implement Interface Wizard*, select the radio group item *File* and enter the full path and filename of the DEWEsoft.exe file from your DEWEsoft® installation in the *Location* input field (e.g. D:\DEWEsoft7\Bin\V7_0\DEWEsoft.exe). Then select the interface which we want to implement (e.g. [IPlugin2](#)) and press the button with the right arrow to move it to the right list box:

Notes:

- do not select [IPlugin2](#) and [IPlugin3](#) at the same time (since [IPlugin2](#) is the parent-class of [IPlugin3](#), you should only select [IPlugin3](#)).
- [IPlugin4](#) does not inherit from [IPlugin2](#) and [IPlugin3](#) - so you can add it (e.g. choose [IPlugin2](#) and [IPlugin4](#) or [IPlugin3](#))

and [IPlugin4](#)).

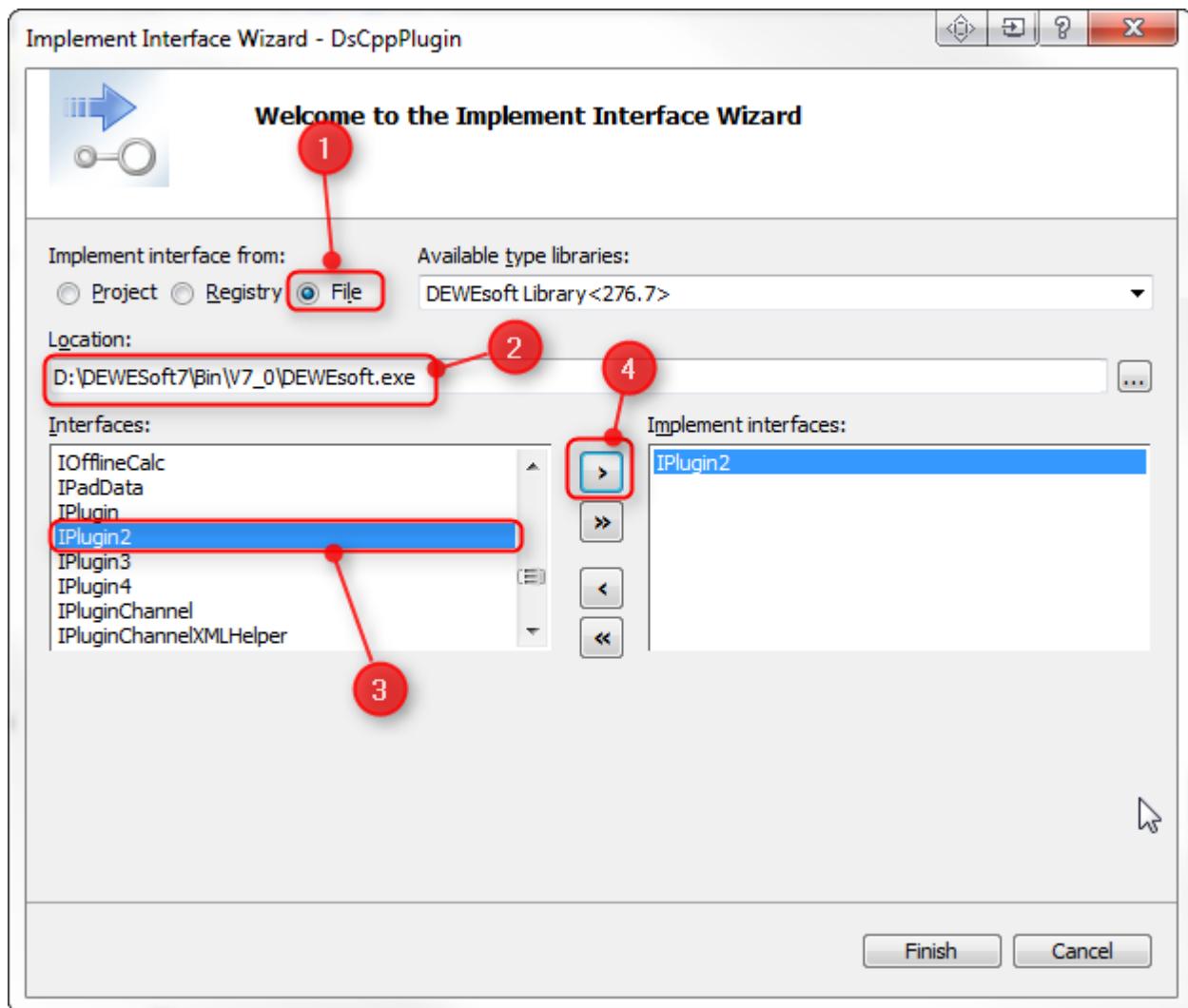


Illustration 80: Implement Interface Wizard

In the code editor you can now see that some identifiers show warnings (the red curly underlined tokens):

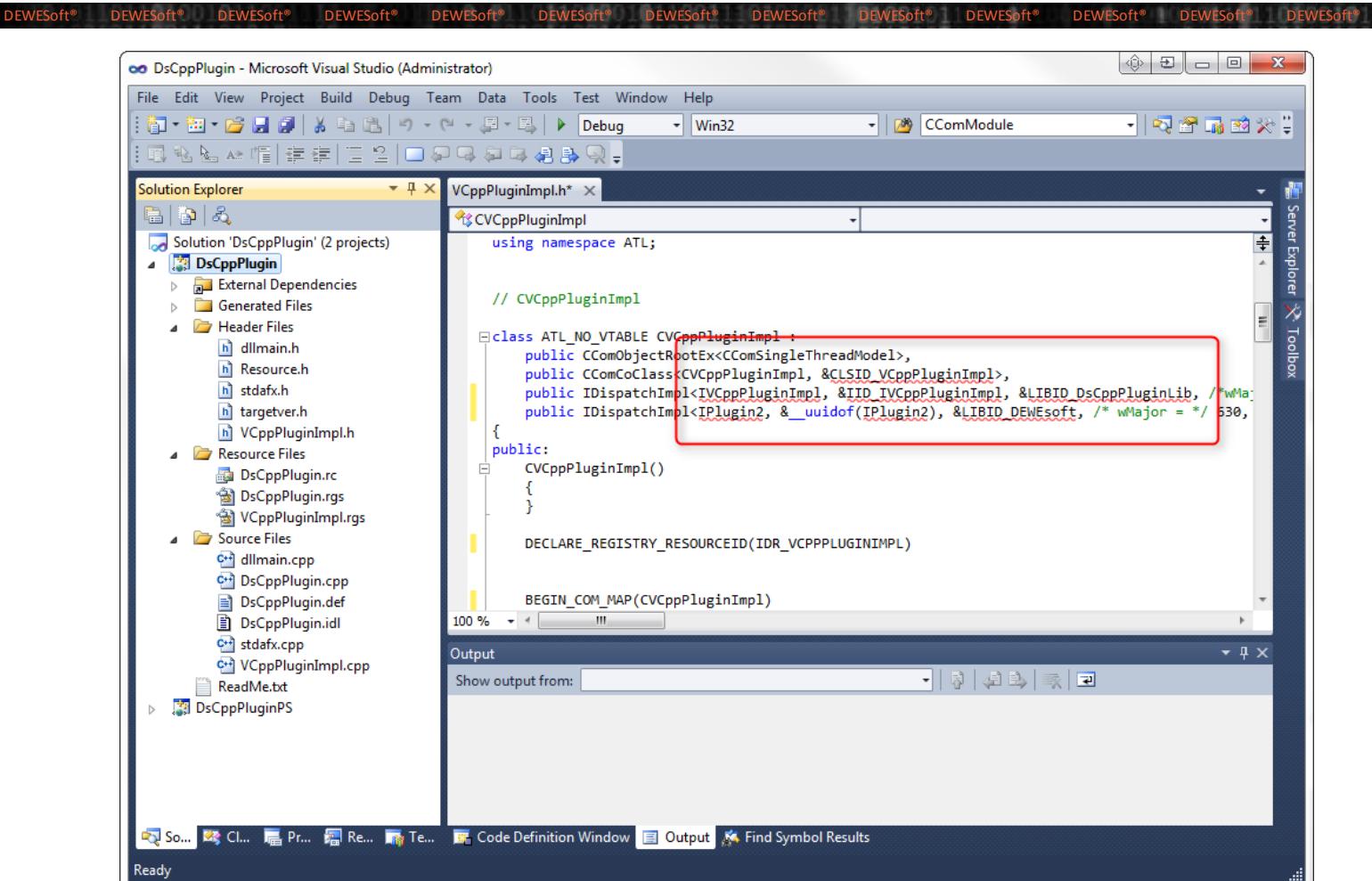


Illustration 81: Warnings

We should now start the build of the solution:

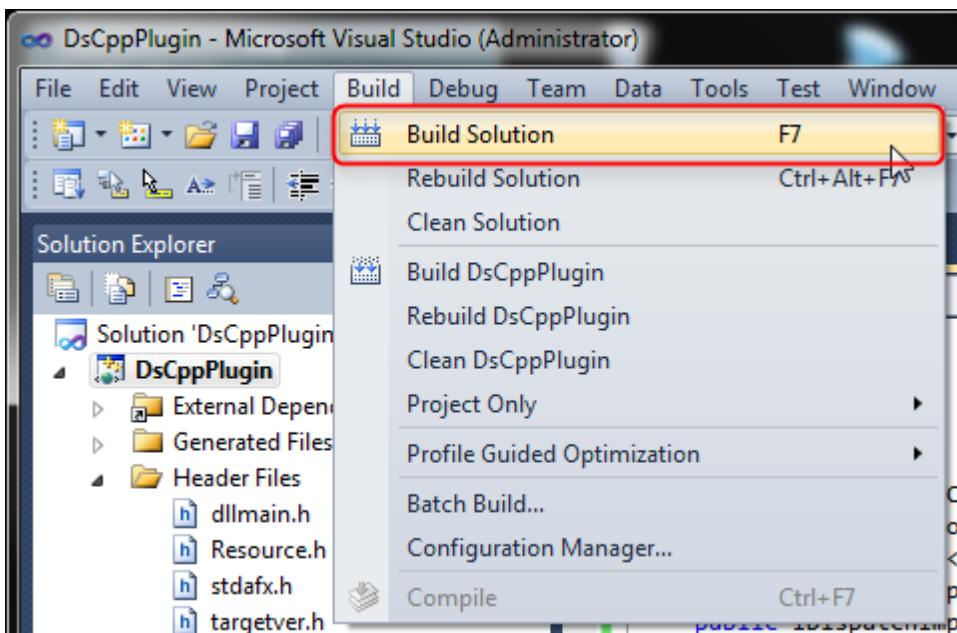
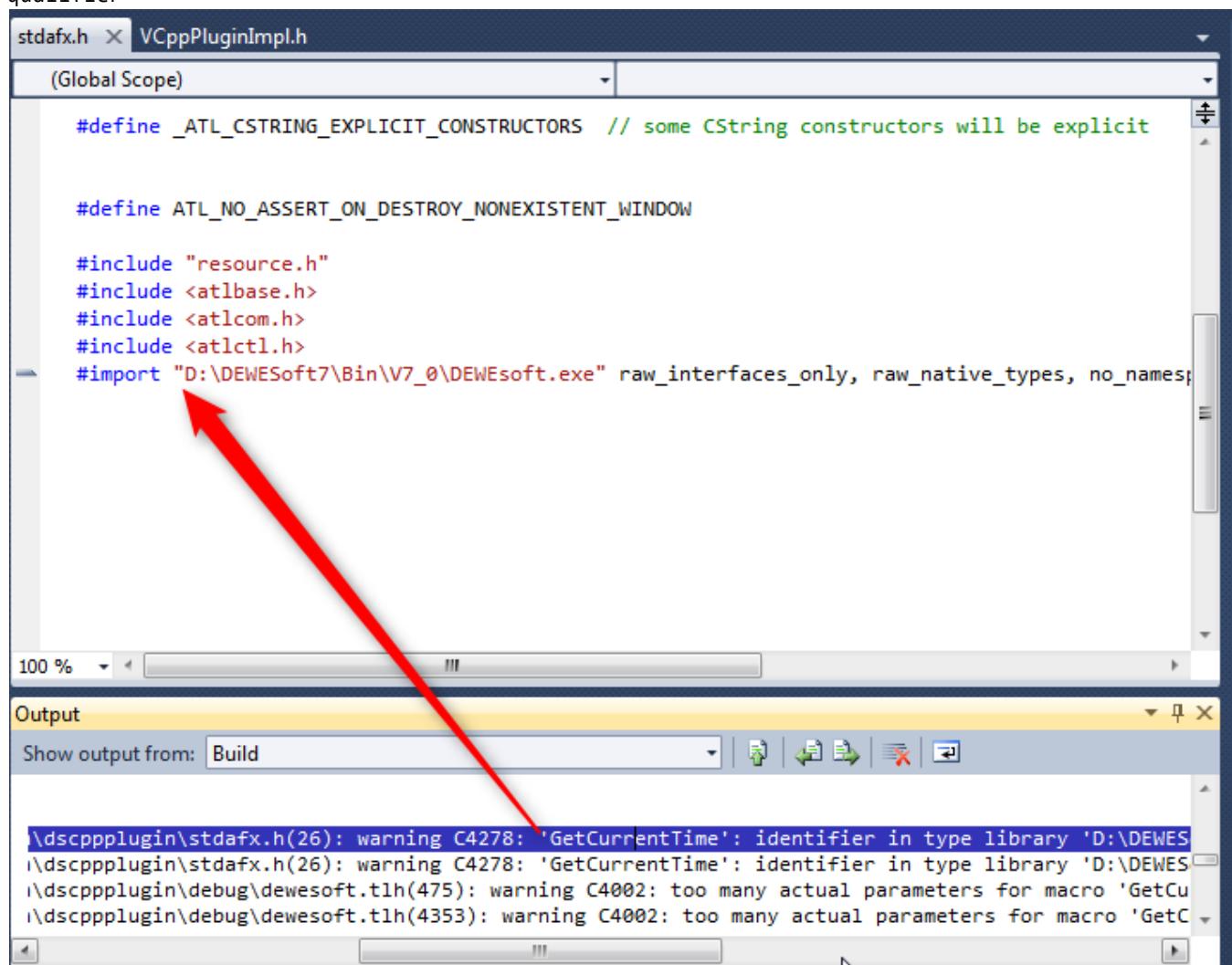


Illustration 82: Build Solution

Note: if you can still see the red curly underlines after you have built the solution, try to add a character somewhere to the source code, remove that character and then save the file: if that does also not help, restart the IDE.

The build output window will now show a warning message:

```
1>c:\users\xxx.xxx.dewesoft\documents\visual studio 2010
\projects\dscppplugin\dscppplugin\stdafx.h(26): warning C4278: 'GetCurrentTime': identifier in
type library 'D:\DEWEsoft7\Bin\V7_0\DEWEsoft.exe' is already a macro; use the 'rename'
qualifier
```



To get rid of this, we need to undefine the `GetCurrentTime` macro:

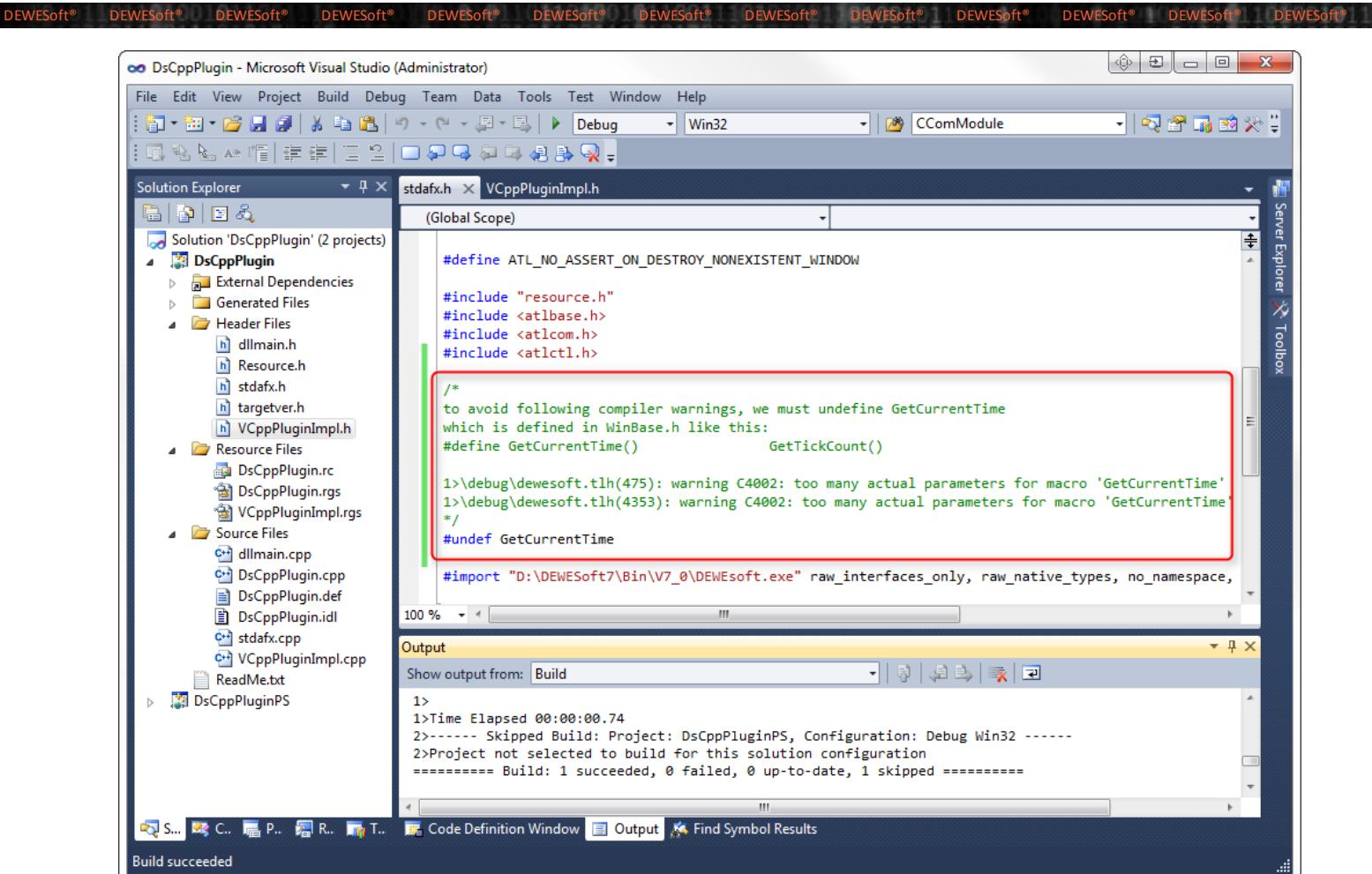


Illustration 84: Warning GetcurrentTime fixed

The project has been set-up correctly: it's now time to implement the plugin: see [Visual C++: Implementation](#)

1.5.4.2 Visual C++: Implementation

After you have prepared the C++ Basic project ([Visual C++: Prepare Project](#)), we can start to implement the plugin class.

Replace return codes

When we have added the interface to our class, the Visual Basic IDE has created default method implementations for all the functions of the interface, which return `E_NOTIMPL`. We need to change all those codes to `S_OK`.

Just open the Find and Replace dialogue by hitting `CTRL-H` (or goto: *Edit - Find and Replace - Quick replace*). In the *Find* *What* input field enter `E_NOTIMPL` and in the *Replace with* input field enter `S_OK`. Then click the *Replace All* button.

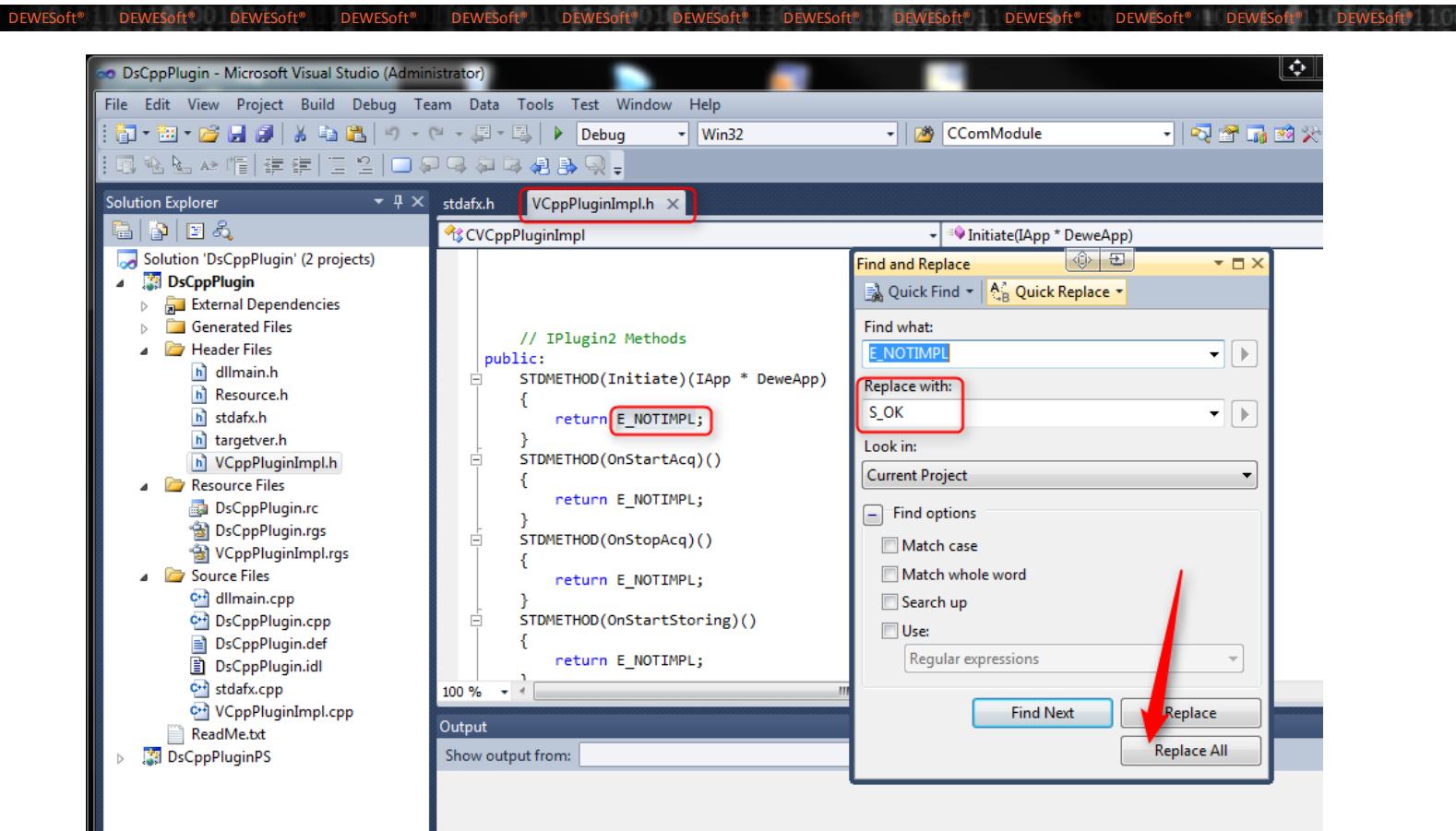


Illustration 85: Replace return codes

Now we will start to implement the most important functions that are necessary to make our plugin work in *DEWESoft®*.

Keep reference for the [IApp](#) interface

After *DEWESoft®* starts the plugin, it will call the [Initiate](#) function and pass along the [IApp](#) interface. We will assign it to a local variable called app. It is the main entry point for communicating with *DEWESoft®* and we will need it in most other methods that we implement.

Since we want to have all non-trivial method implementations in the `VCppPluginImpl.cpp` file, we replace the standard method implementation of the function with a function declaration in the `VCppPluginImpl.h` file - and we will also create a private variable `app` of type [IApp](#).

The screenshot shows the Microsoft Visual Studio interface with the title bar "DsCppPlugin - Microsoft Visual Studio (Administrator)". The menu bar includes File, Edit, View, Project, Build, Debug, Team, Data, Tools, Test, Window, Help. The toolbar has various icons for file operations. The status bar shows "Debug Win32". The Solution Explorer on the left lists the solution "DsCppPlugin" with two projects: "DsCppPlugin" and "Generated Files". Under "Header Files" for "DsCppPlugin", there is a file named "VCppPluginImpl.h". The code editor on the right contains the following code:

```

void FinalRelease()
{
}

private:
    IApp* app;

// IPlugin2 Methods

public:
    STDMETHOD(Initiate)(IApp * DeweApp);
    STDMETHOD(OnStartAcq)()
    {
        return S_OK;
    }
    STDMETHOD(OnStopAcq)
}

```

The "Initiate" method declaration is highlighted with a red box.

Illustration 86: *Initiate()* declaration

The implementation will be added in the `VCppPluginImpl.cpp` file:

The screenshot shows the Microsoft Visual Studio interface with the title bar "DsCppPlugin - Microsoft Visual Studio (Administrator)". The menu bar includes File, Edit, View, Project, Build, Debug, Team, Data, Tools, Test, Window, Help. The toolbar has various icons for file operations. The status bar shows "Debug Win32" and "E_NOD". The Solution Explorer on the left lists the solution "DsCppPlugin" with two projects: "DsCppPlugin" and "Generated Files". Under "Header Files" for "DsCppPlugin", there is a file named "VCppPluginImpl.h". The code editor on the right contains the following code:

```

// VCppPluginImpl.cpp : Implementation of CVCppPluginImpl

#include "stdafx.h"
#include "VCppPluginImpl.h"

// CVCppPluginImpl

STDMETHODIMP CVCppPluginImpl::Initiate(IApp * DeweApp)
{
    app = DeweApp;
    return S_OK;
}

```

The implementation of the "Initiate" method is highlighted with a red box.

Illustration 87: *Initiate()* implementation

Remember the Used-status

Next, we implement the property `Used` which tells us, if the user has set the plugin to *Used* or *Unused* in the hardware setup.

In the header file (`VCppPluginImpl.h`) we add a `used` variable and replace the function implementations of

`get_Used` and `put_Used` with their declarations:

```
private:
    IApp* app;
    bool used;

...
public:
...
    STDMETHOD(get_Used)(VARIANT_BOOL * Value);
    STDMETHOD(put_Used)(VARIANT_BOOL Value);
```

in the `VCppPluginImpl.cpp` file we add the corresponding implementations:

```
STDMETHODIMP CVCppPluginImpl::get_Used(VARIANT_BOOL * Value)
{
    *Value = used;
    return S_OK;
}
STDMETHODIMP CVCppPluginImpl::put_Used(VARIANT_BOOL Value)
{
    used = Value;
    return S_OK;
}
```

Implement channel setup

Next, we implement the [Configure\(\)](#) function. This is not really necessary, but we just want to check if the function is called:

In the header file (`VCppPluginImpl.h`) we replace the function implementations with the following declarations:

```
STDMETHOD(Configure)();
STDMETHOD>ShowFrame)(long Parent, VARIANT_BOOL * Value);
```

in the `VCppPluginImpl.cpp` file we add the corresponding implementations:

```
STDMETHODIMP CVCppPluginImpl::Configure()
{
    MessageBox(NULL, TEXT("config"), TEXT("Title"), MB_OK);
    return S_OK;
}

STDMETHODIMP CVCppPluginImpl::ShowFrame(long Parent, VARIANT_BOOL * Value)
{
    *Value = false;
    return S_OK;
}
```

Since it is only possible for Delphi plugins to show an embedded frame in *DEWEsoft®*, we must set the result of the [ShowFrame\(\)](#) function to `False`. (we could show a configuration form via the [Configure\(\)](#) function above.

Add a plugin channel

Finally we will add a [channel](#), a `MountChannels()` function to mount this channel and the `OnGetData()` function to fill it

In the header file (`VCppPluginImpl.h`) we add a `ch` variable, a declaration for the `MountChannels()` function and replace the function implementations with the following declarations:

private:

```
...  
IChannel* ch;  
void MountChannels();  
...  
  
STDMETHOD(OnGetData)();  
STDMETHOD(LoadSetup)(VARIANT Data);  
STDMETHOD(NewSetup)();  
STDMETHOD(ClearChannelsInstance)();
```

in the `VCppPluginImpl.cpp` file we add the corresponding implementations:

```

void CVCppPluginImpl::MountChannels()
{
    IData* data;
    IChannelGroups* groups;
    IChannelGroup* group;
    IPluginGroup* pluginGroup;

    app->get_Data(&data);
    data->get_Groups(&groups);
    groups->get_Item(8, &group);
    group->QueryInterface(IID_IPluginGroup, (void **) &pluginGroup);

    pluginGroup->MountChannel(5, TRUE, -1, &ch);
    _bstr_t s = "Test channel";
    ch->put_Name(s);
    ch->put_Used(TRUE);
}

STDMETHODIMP CVCppPluginImpl::LoadSetup(VARIANT Data)
{
    MountChannels();
    return S_OK;
}

STDMETHODIMP CVCppPluginImpl::NewSetup()
{
    MountChannels();
    return S_OK;
}

STDMETHODIMP CVCppPluginImpl::OnGetData()
{
    IMasterClock* mClock;
    double time;
    float value;

    app->get_MasterClock(&mClock);
    mClock->GetCurrentTime(&time);

    value = (float)(rand()) / RAND_MAX;
}

```

```

    if (time >= 0)
        ch->AddAsyncSingleSample(value, time);

    return S_OK;
}

STDMETHODIMP CVCppPluginImpl::ClearChannelsInstance()
{
    ch = NULL;
    return S_OK;
}

```

Now that the implementation is okay, we need to add some more information to make our class library work with *DCOM* and *DEWESoft®*: see [Visual C++: Prepare for DCOM](#)

1.5.4.3 Visual C++: Prepare for DCOM

Now that the plugin implementation is complete (see [Visual C++: Implementation](#)), we need to add some more code, so that the class is recognized by *DEWESoft®*.

We alter the file `DsCppPlugin.cpp` which already has the installation routines for the DLL .

We add a `#define` for the plugin registry key, a function `WriteDewesoftReg()` which will be called in the `DllRegisterServer()` function and in the `DllUnregisterServer()` function we delete the registry plugin key, when the plugin is uninstalled:

```

#define PLUGIN_KEY "SOFTWARE\$\ Dewesoft\$\ Plugins\$\ DsCppPlugin"

void WriteDewesoftReg()
{
    HKEY hKey;
    DWORD disposition;

    RegCreateKeyEx(HKEY_LOCAL_MACHINE, TEXT(PLUGIN_KEY), 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &hKey, &disposition);

    LPCTSTR RegName = TEXT("Ds C++ Plugin");
    LPCTSTR RegVendor = TEXT("Dewesoft");
    LPCTSTR RegVersion = TEXT("1.0.0");
    LPCTSTR RegDescription = TEXT("Test");
    LPOLESTR lpoleGuid;
    StringFromCLSID(CLSID_VCppPluginImpl, &lpoleGuid);

    RegSetValueEx(hKey, TEXT("Name"), 0, REG_SZ, (LPBYTE)RegName, _tcslen(RegName) * sizeof
(TCHAR));
    RegSetValueEx(hKey, TEXT("Vendor"), 0, REG_SZ, (BYTE*)RegVendor, _tcslen(RegVendor) * sizeof
(TCHAR));
    RegSetValueEx(hKey, TEXT("Version"), 0, REG_SZ, (BYTE*)RegVersion, _tcslen(RegVersion) * sizeof
(TCHAR));
    RegSetValueEx(hKey, TEXT("Description"), 0, REG_SZ, (BYTE*)RegDescription, _tcslen
(RegDescription) * sizeof(TCHAR));
    RegSetValueEx(hKey, TEXT("GUID"), 0, REG_SZ, (BYTE*)lpoleGuid, _tcslen(lpoleGuid) * sizeof
(TCHAR));

    RegCloseKey(hKey);
}

```

```
// DllRegisterServer - Adds entries to the system registry.
STDAPI DllRegisterServer(void)
{
    WriteDewesoftReg();

    // registers object, typelib and all interfaces in typelib
    HRESULT hr = _AtlModule.DllRegisterServer();
    return hr;
}

// DllUnregisterServer - Removes entries from the system registry.
STDAPI DllUnregisterServer(void)
{
    RegDeleteKey(HKEY_LOCAL_MACHINE, TEXT(PLUGIN_KEY));

    HRESULT hr = _AtlModule.DllUnregisterServer();
    return hr;
}
```

now we have finished all the source code and can build the the solution:

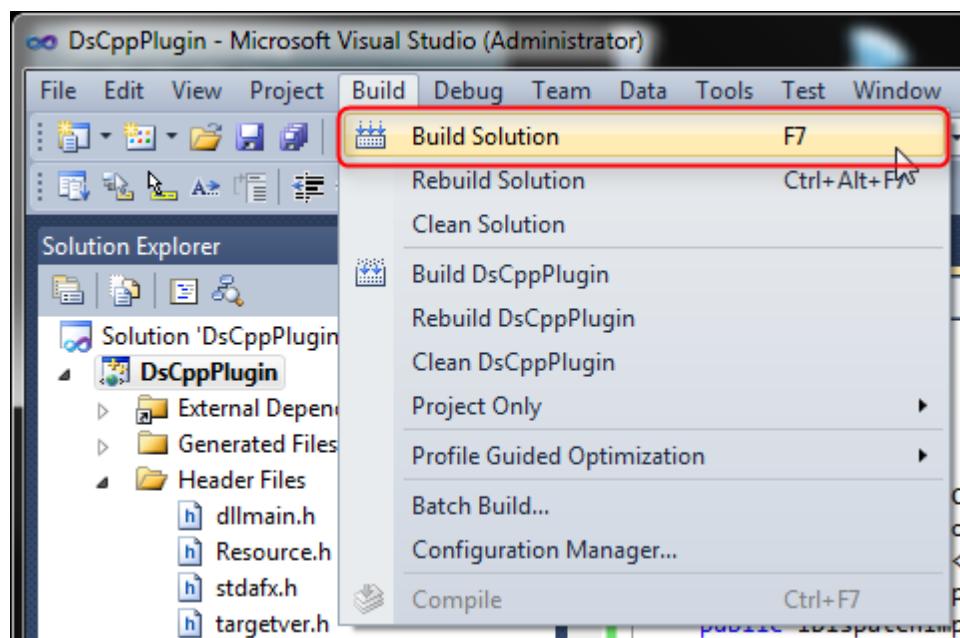


Illustration 88: Build Solution

Note: since we have changed the *Output Directory* in the project properties (see [Visual C++: Prepare Project](#)) the .dll will be created directly in the Addons folder of DEWESoft® (e.g. D:\DEWESoft7\Bin\V7_0\Addons):

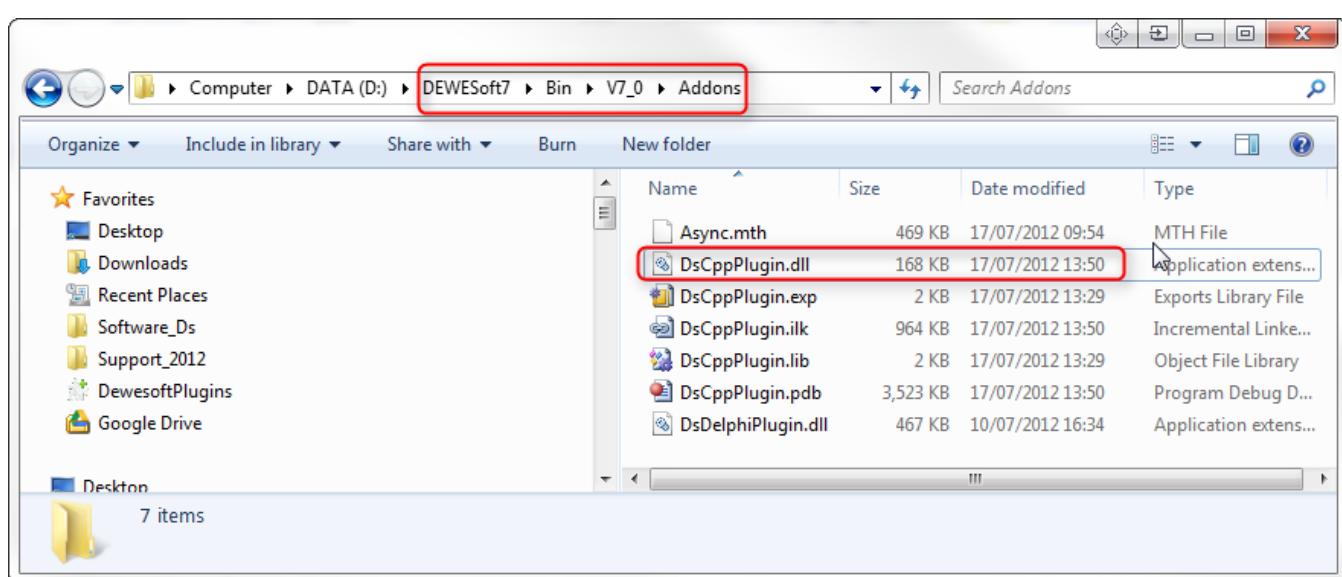


Illustration 89: Addons directory

Note: the Visual Studio IDE will also generate some other files (in addition to the `DsCppPlugin.dll`): `DsCppPlugin.exp`, etc. Those other files are not required for the execution of the plugin (i.e. you can delete them and the plugin will still work). Thus, when you want to use the plugin on another PC (than on your development PC), you just need to copy the `DsCppPlugin.dll` file to the target PC (and register it with the `regsvr32` tool - see [Visual C++: Register Plugin](#))

next: [Visual C++: Register Plugin](#)

1.5.4.4 Visual C++: Register Plugin

In order to use the DLL as COM object, you must do a registration process once with the `RegSvr32` tool:

In the Windows command line, go to the `Addons` folder of *DEWESoft®* (e.g. `D:\DEWESoft7\Bin\V7_0\Addons`) and call the `regsvr32` tool:

```
regsvr32 DsCppPlugin.dll
```

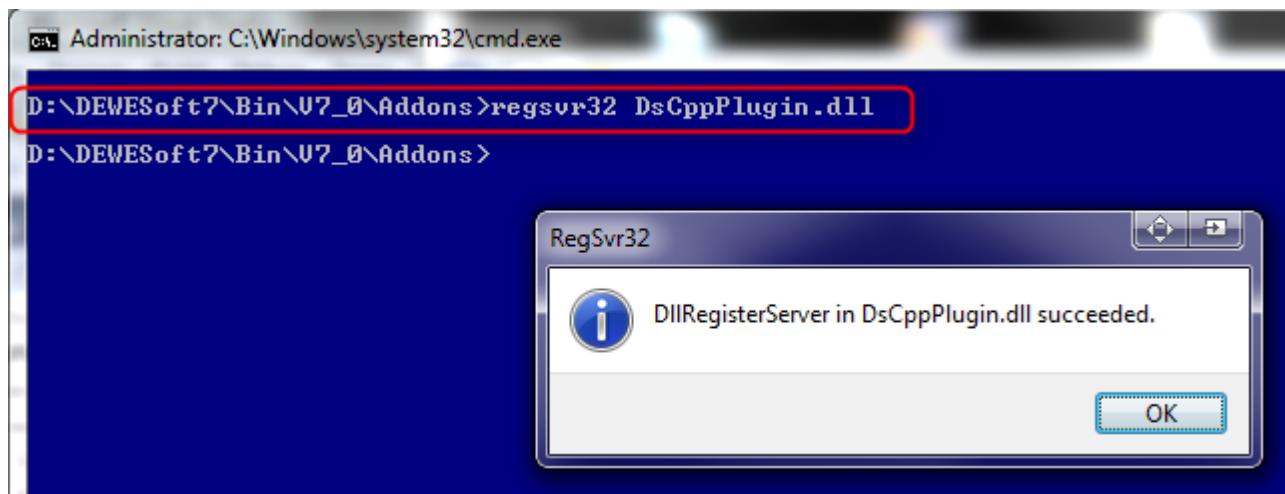


Illustration 90: Register Plugin

See also: [Visual C++: Troubleshooting](#)

The `regsvr32` tool will call the `DllRegisterServer()` function (see [Visual C++: Prepare for DCOM](#)) which will generate all required registry entries for COM and **DEWEsoft®**.

```
[HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{BE6EAD76-FDC6-455B-977A-268C879E4FF6}]
@="VCppPluginImpl Class"

[HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{BE6EAD76-FDC6-455B-977A-268C879E4FF6}\InprocServer32]
@="D:\DEWEsoft7\Bin\V7_0\Addons\DsCppPlugin.dll"
"ThreadingModel"="Apartment"

[HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{BE6EAD76-FDC6-455B-977A-268C879E4FF6}\Programmable]

[HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{BE6EAD76-FDC6-455B-977A-268C879E4FF6}\TypeLib]
@="{2500D9DB-B380-4FFF-963E-FFEEAC4D2FFA}"

[HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{BE6EAD76-FDC6-455B-977A-268C879E4FF6}\Version]
@="1.0"
```

Note: in the Visual Studio IDE you will find the registry information in the `VCppPluginImpl.rgs` file:

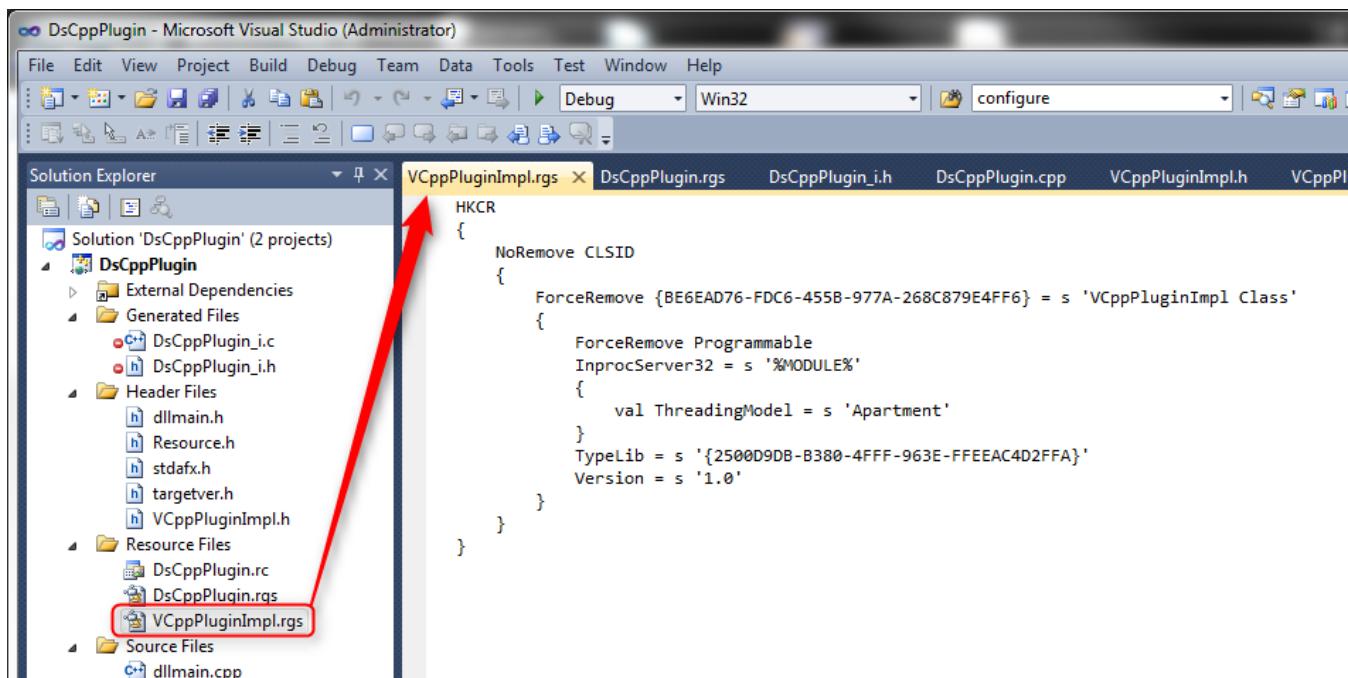


Illustration 91: *PluginImpl registration file*

During its execution, the `regsvr32` tool has also executed the `WriteDewesoftReg()` function (see [Visual C++: Troubleshooting](#)).

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\DeWeSoft\Plugins\DsCppPlugin]
"Name"="Ds Plugin 03"
"Vendor"="DeWeSoft"
"Version"="1.0.0"
"Description"="Test"
"GUID"="{BE6EAD76-FDC6-455B-977A-268C879E4FFF}"
```

Note: that the registration information for *DEWESoft®* refers via the [GUID](#) (highlighted in green in the text above) to the COM object for our plugin.

Note: the key `Wow6432Node` (highlighted in blue) will be inserted on 64-bit Windows systems automatically - On 32-bit Windows systems the key `Wow6432Node` will not be present.

see also: [Finding Plugins](#)

That's it: we can now start *DEWESoft®* and test the plugin: [Visual C++: Test the Plugin](#)

1.5.4.5 Visual C++: Test the Plugin

After we have successfully registered the plugin (see [Visual C++: Register Plugin](#)), we can finally start up *DEWESoft®* and test our Visual C++ plugin.

When you go to the *Plugins* tab sheet of the hardware setup, the plugin will already show up:

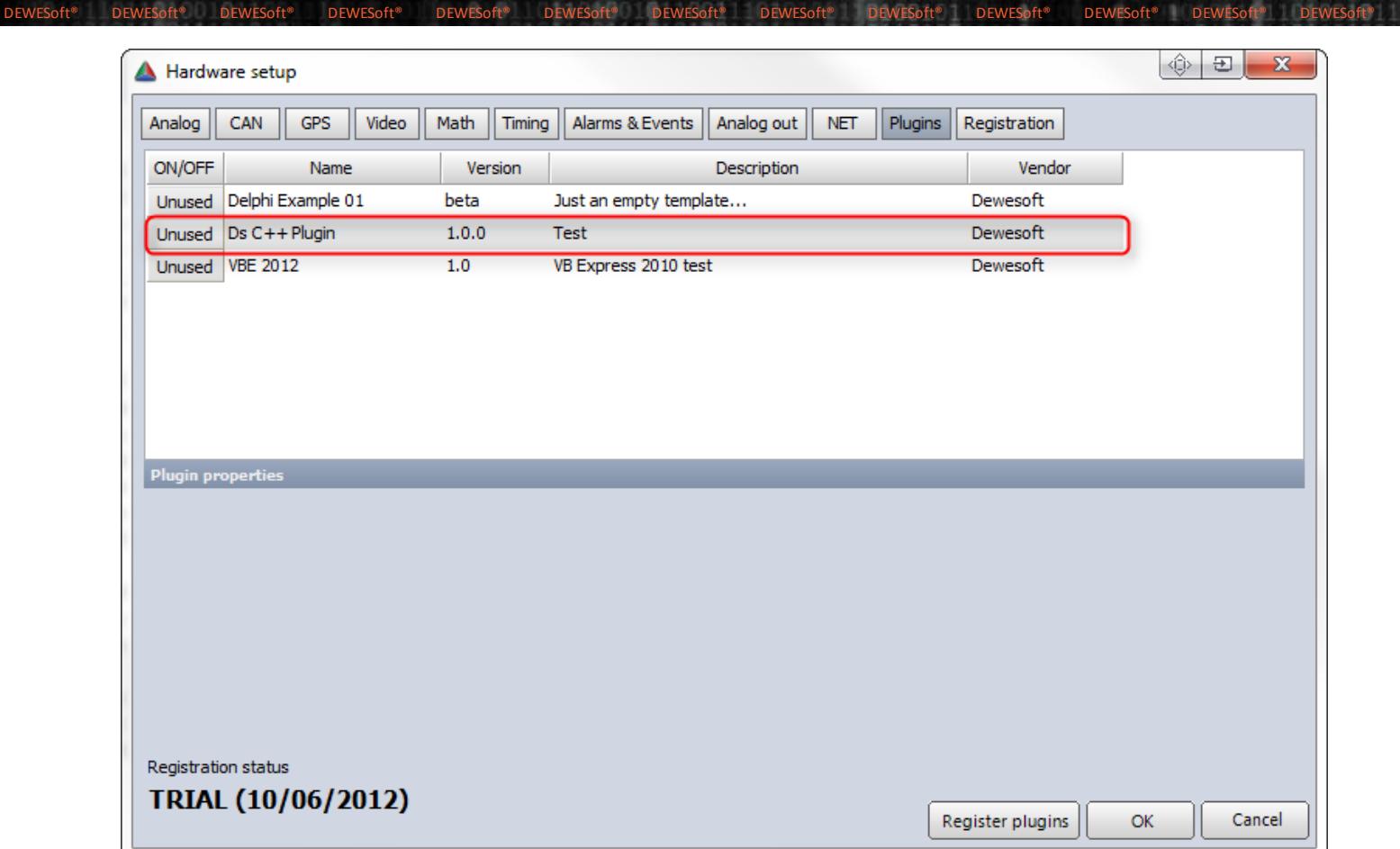


Illustration 92: Plugin shows up in DEWEsoft™

Click the *Unused* button to set the plugin to *Used* and then you can already see the plugin in the channel setup:



Illustration 93: Channel Setup

When you press the *Configure* button, the message box that we have defined in the `Configure()` function (see [Visual C++ +: Implementation](#)) will show up.

When we switch to *Measure* mode, we can see that the channel (called *Test Channel*) that we have mounted (see

function MountChannels() in [Visual C++: Implementation](#)) shows up in the channel list (see red rectangle on the right side in the screenshot below) and that the random data is shown in the recorder:

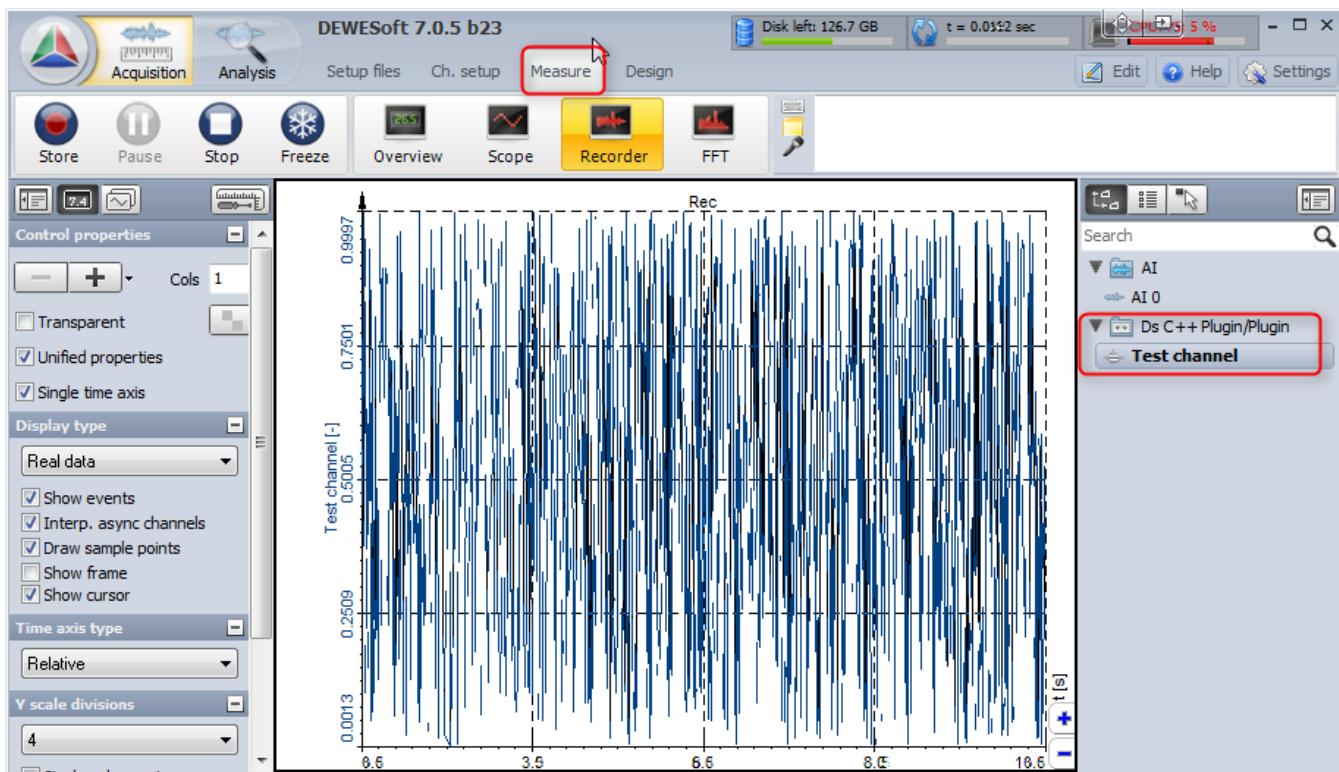


Illustration 94: Measure mode

see also: [Visual C++: Troubleshooting](#), [Visual C++: Sourcecode](#)

1.5.4.6 Visual C++: Troubleshooting

This chapter describes some frequent problems regarding C++ plugins ([Visual C++](#)).

not a valid .NET assembly

when you try to register the Visual C++ dll with RegAsm.exe you will get this error message:

```
D:\DEWEsoft7\Bin\V7_0\Addons>C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe /codebase DsCppPlugin.dll
Microsoft (R) .NET Framework Assembly Registration Utility 4.0.30319.1
Copyright (C) Microsoft Corporation 1998-2004. All rights reserved.

RegAsm : error RA0000 : Failed to load 'D:\DEWEsoft7\Bin\V7_0\Addons\DsCppPlugin.dll' because it is not a valid .NET assembly
```

The C++ dll that we have created is not a .NET assembly (in comparison to the Visual Basic example), so you must not use the RegAsm.exe tool to register the COM dll, but the regsvr32 tool instead: see [Visual C++: Register Plugin](#)

Redistributable

When copy the .dll file to another PC and start the registration tool, you may get the following error message:

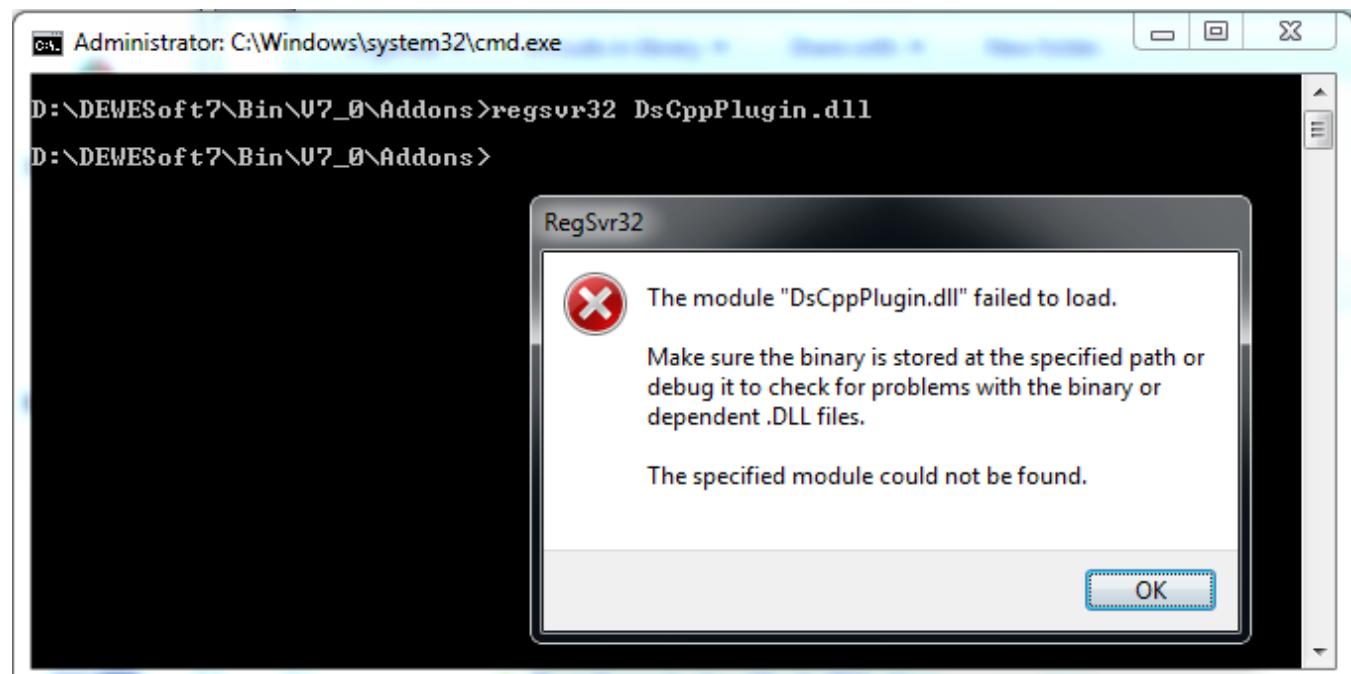


Illustration 95: RegSvr32 Error Message

In this case check the following:

- make sure that the correct version of the *Microsoft Visual C++ 2010 Redistributable Package* is installed on that PC
- make sure that you have built your project with the *Release configuration* settings (more details below)

Microsoft Visual C++ 2010 Redistributable Package

You can download the *Microsoft Visual C++ 2010 Redistributable Package* from the following URL:

<http://www.microsoft.com/en-us/download/search.aspx?q=Microsoft%20Visual%20C%2B%2B%202010%20Redistributable%20>

Also note, that the toolset can be specified in the project properties (hit ALT+F7 or goto *Project - DsCppPlugin properties...*):

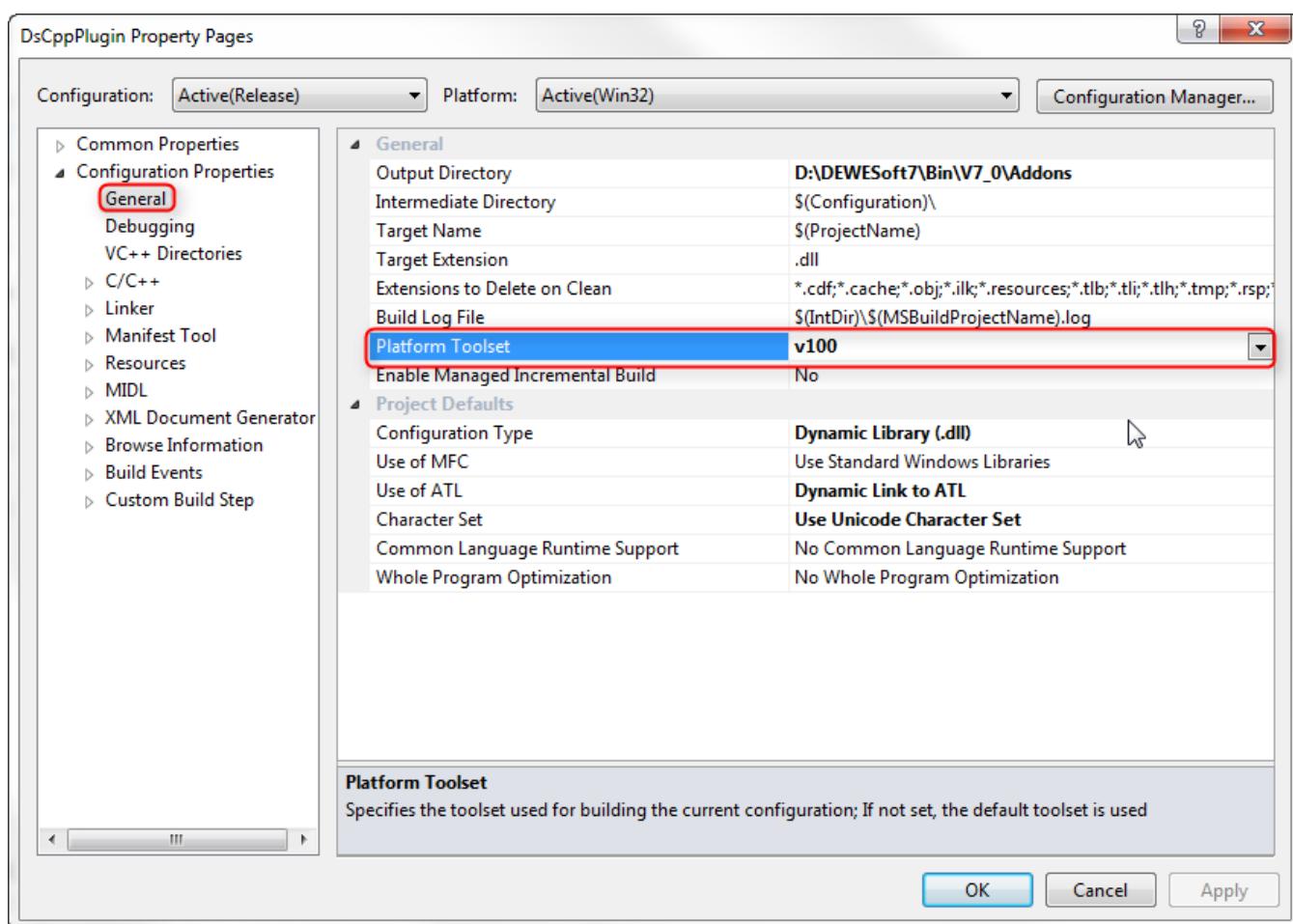


Illustration 96: Toolset

see also: [Visual C++: Register Plugin](#)

Release configuration

When you have tested your plugin and finally decide to run it on another computer, you must switch to the *Release configuration* and then build the project:

Open the project properties (hit ALT+F7 or goto *Project - DsCppPlugin properties...*) and then click on the *Configuration Manager...* button:

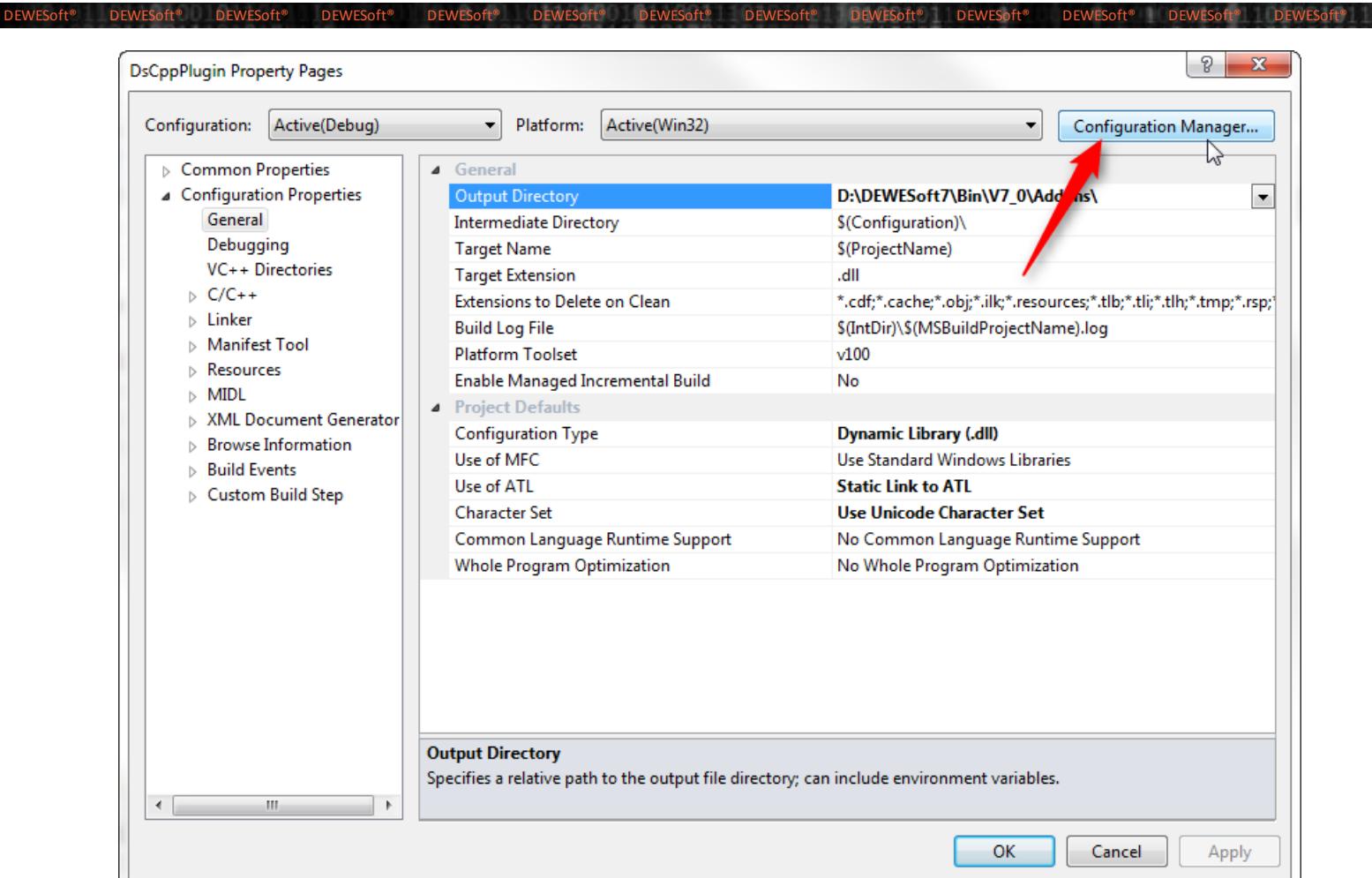


Illustration 97: Open Configuration Manager

Then select the [Release configuration](#) for all projects:

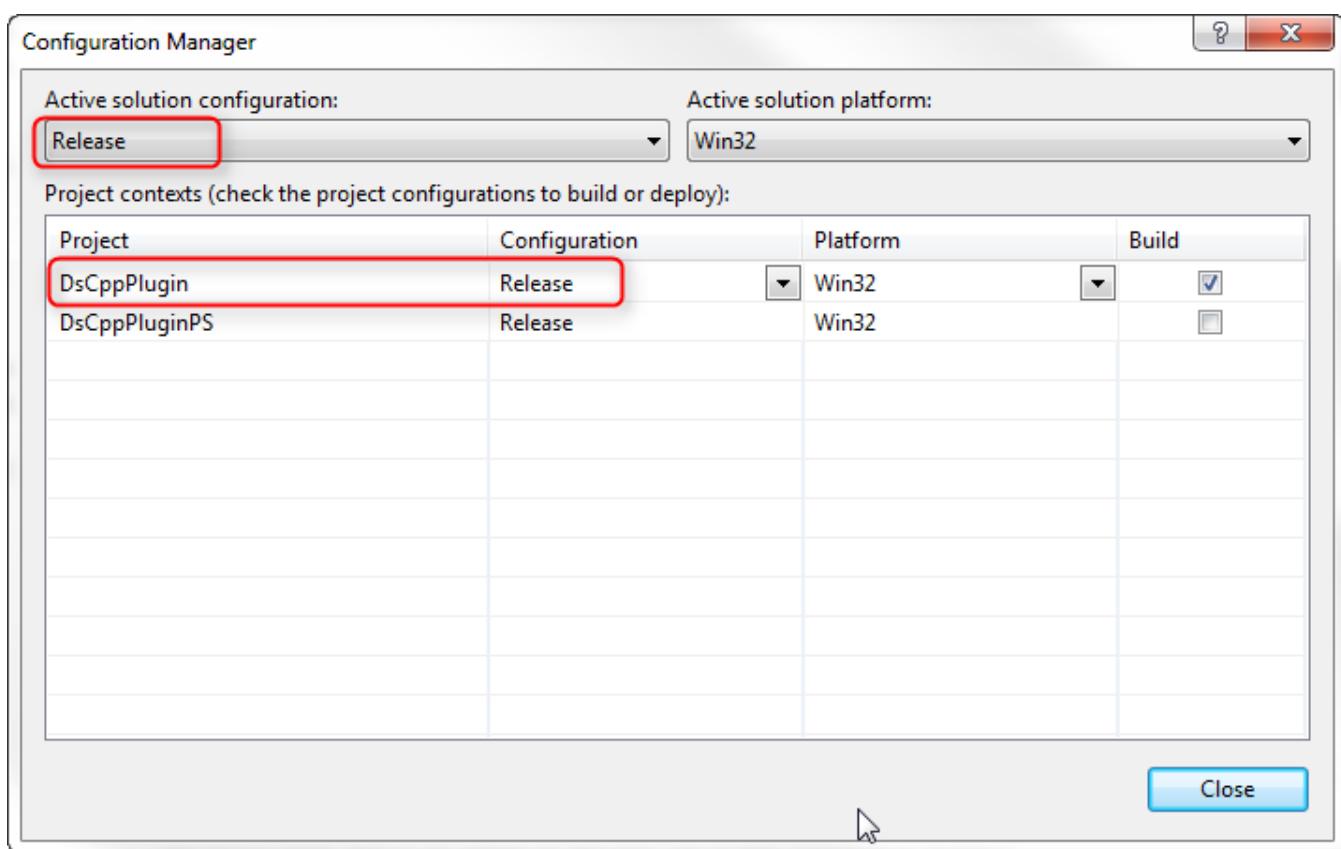


Illustration 98: Configuration Manager - Release Configuration

Now you can rebuild your project and distribute the generated .dll file to the target PC's (note that the generated .dll file of the *Release* build will be much smaller than in the the *Debug* build).

Other

If you still get the error your code may use other external dependencies. The tool `dumpbin.exe` which is included in the *Visual Studio* package may help you to find out what's missing:

to start `dumpbin.exe` go to: [Tools - Visual Studio Command Prompt](#):

e.g. use the tool to show the `msvcr` dependencies:

```
D:\DEWEsoft7\Bin\V7_0\Addons>dumpbin /ALL DsCppPlugin.dll | find /i "msvcr"
100057C0: C4 05 6D 61 6C 6C 6F 63 00 00 4D 53 56 43 52 31 -.malloc..MSVCR1
MSVCR100.dll
```

see also: [http://msdn.microsoft.com/en-us/library/c1h23y6c\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/c1h23y6c(v=vs.71).aspx)

Not Implemented

When you get a Not implemented error message when trying to active the plugin like this:

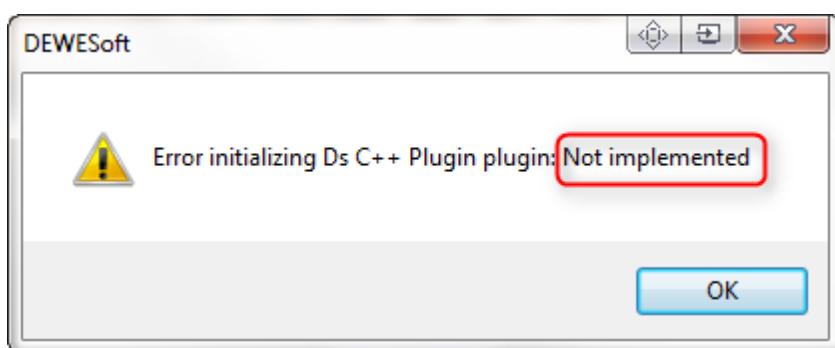


Illustration 99: Not Implemented

a possible problem is that you return the wrong error code `E_NOTIMPL` from a function:

e.g. the following return code would cause this error message:

```
STDMETHODIMP CVCppPluginImpl::Initiate(IApp * DeweApp)
{
    app = DeweApp;
    return E_NOTIMPL;
}
```

Just return S_OK in the function to solve the problem:

```
STDMETHODIMP CVCppPluginImpl::Initiate(IApp * DeweApp)
{
    app = DeweApp;
    return S_OK;
}
```

1.5.4.7 Visual C++: Sourcecode

VCppPluginImpl.h

```
// VCppPluginImpl.h : Declaration of the CVCppPluginImpl

#pragma once
#include "resource.h"           // main symbols

#include "DsCppPlugin_i.h"

#if defined(_WIN32_WCE) && !defined(_CE_DCOM) && !defined
(_CE_ALLOW_SINGLE_THREADED_OBJECTS_IN_MTA)
#error "Single-threaded COM objects are not properly supported on Windows CE platform, such as
the Windows Mobile platforms that do not include full DCOM support. Define
_CE_ALLOW_SINGLE_THREADED_OBJECTS_IN_MTA to force ATL to support creating single-thread COM
object's and allow use of it's single-threaded COM object implementations. The threading model
in your rgs file was set to 'Free' as that is the only threading model supported in non DCOM
Windows CE platforms."
#endif

using namespace ATL;
```

```

// CVCppPluginImpl

class ATL_NO_VTABLE CVCppPluginImpl :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CVCppPluginImpl, &CLSID_VCcppPluginImpl>,
    public IDispatchImpl<IVCcppPluginImpl, &IID_IVCcppPluginImpl, &LIBID_DsCppPluginLib, /
/*wMajor = */ 1, /*wMinor = */ 0>,
    public IDispatchImpl<IPlugin2, &__uuidof(IPlugin2), &LIBID_DEWEsoft, /* wMajor = */ 630,
/* wMinor = */ 7>
{
public:
    CVCppPluginImpl()
    {
    }

DECLARE_REGISTRY_RESOURCEID(IDR_VCPPPLUGINIMPL)

BEGIN_COM_MAP(CVCppPluginImpl)
    COM_INTERFACE_ENTRY(IVCcppPluginImpl)
    COM_INTERFACE_ENTRY2(IDispatch, IPlugin2)
    COM_INTERFACE_ENTRY(IPlugin2)
END_COM_MAP()

DECLARE_PROTECT_FINAL_CONSTRUCT()

HRESULT FinalConstruct()
{
    return S_OK;
}

void FinalRelease()
{
}

private:
    IApp* app;
    bool used;
    IChannel* ch;
    void MountChannels();

    // IPlugin2 Methods
public:
    STDMETHOD(Initiate)(IApp * DeweApp);
    STDMETHOD(OnStartAcq)()
    {
        return S_OK;
    }
    STDMETHOD(OnStopAcq)()
    {
        return S_OK;
    }
    STDMETHOD(OnStartStoring)()
    {
        return S_OK;
    }
    STDMETHOD(OnStopStoring)()

```

VCppPluginImpl.cpp

```
// VCppPluginImpl.cpp : Implementation of CVCppPluginImpl

#include "stdafx.h"
#include "VCppPluginImpl.h"

// CVCppPluginImpl

STDMETHODIMP CVCppPluginImpl::Initiate(IApp * DeweApp)
{
    app = DeweApp;
    return S_OK;
}

STDMETHODIMP CVCppPluginImpl::get_Used(VARIANT_BOOL * Value)
{
    *Value = used;
    return S_OK;
}
```



```
    ch->AddAsyncSingleSample(value, time);

    return S_OK;
}

STDMETHODIMP CVCppPluginImpl::ClearChannelsInstance()
{
    ch = NULL;
    return S_OK;
}
```

DsCppPlugin.cpp

// DsCppPlugin.cpp : Implementation of DLL Exports.

```
#include "stdafx.h"
#include "resource.h"
#include "DsCppPlugin_i.h"
#include "dllmain.h"

// Used to determine whether the DLL can be unloaded by OLE.
STDAPI DllCanUnloadNow(void)
{
    return _AtlModule.DllCanUnloadNow();
}

// Returns a class factory to create an object of the requested type.
STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{
    return _AtlModule.DllGetClassObject(rclsid, riid, ppv);
}

#define PLUGIN_KEY "SOFTWARE\\Dewesoft\\Plugins\\DsCppPlugin"

void WriteDewesoftReg()
{
    HKEY hKey;
    DWORD disposition;

    RegCreateKeyEx(HKEY_LOCAL_MACHINE, TEXT(PLUGIN_KEY), 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &hKey, &disposition);

    LPCTSTR RegName = TEXT("Ds C++ Plugin");
    LPCTSTR RegVendor = TEXT("Dewesoft");
    LPCTSTR RegVersion = TEXT("1.0.0");
    LPCTSTR RegDescription = TEXT("Test");
    LPOLESTR lpoleGuid;
    StringFromCLSID(CLSID_VCppPluginImpl, &lpoleGuid);

    RegSetValueEx(hKey, TEXT("Name"), 0, REG_SZ, (LPBYTE)RegName, _tcslen(RegName) * sizeof(TCHAR));
    RegSetValueEx(hKey, TEXT("Vendor"), 0, REG_SZ, (BYTE*)RegVendor, _tcslen(RegVendor) * sizeof(TCHAR));
    RegSetValueEx(hKey, TEXT("Version"), 0, REG_SZ, (BYTE*)RegVersion, _tcslen(RegVersion) * sizeof(TCHAR));
    RegSetValueEx(hKey, TEXT("Description"), 0, REG_SZ, (BYTE*)RegDescription, _tcslen(RegDescription) * sizeof(TCHAR));
    RegSetValueEx(hKey, TEXT("GUID"), 0, REG_SZ, (BYTE*)lpoleGuid, _tcslen(lpoleGuid) * sizeof(TCHAR));
}
```


1.6 How to

The *How to* section shows some examples of frequently used tasks.

The examples are written in *Delphi* but since the major point of the examples is to show the usage of the DEWEsoft® DCOM features, it should be easy for the reader to implement the same in other programming languages.

1.6.1 How To Mount Dewesoft Channels

Usually custom plugins process some data and want to add the data to custom DEWEsoft® channels, so that these channels can then be used by DEWEsoft®. The channels can be displayed directly in visual controls or may even be used by other features of DEWEsoft® (e.g. a custom plugin channel can be used as input to a standard DEWEsoft® math channel).

When you want to add custom channels to DEWEsoft® in your plugin, you must first ask DEWEsoft® to create such a channel for you: this process is called mounting a channel.

When to mount channels

Usually it's the best idea to mount your channels as soon as possible, because then you can always be sure that the channel is already assigned in your plugin and those channels can then already be used by other sections of the channel-setup: e.g. by the Math functions or even by other plugins.

The typical way to mount channels involves the following steps:

1) write a function named `MountChannels` in your plugin (the function name can be arbitrary, but it is a good idea to stick to this name)

```
TPluginImpl = class (...)  
private  
  procedure MountChannels();  
end;
```

Note

Note: the implementation of the `MountChannels` function will follow below.

2) call this function whenever the user loads a setup or creates a new setup

```
procedure TPluginImpl.LoadSetup(Data: OleVariant);  
begin  
  MountChannels();  
end;  
  
procedure TPluginImpl.NewSetup;  
begin  
  MountChannels();  
end;  
  
function TPluginImpl.UpdateXML(Setup: TXMLSetup; Node: IDOMNode;  
  DataFile: Boolean; XMLHelper: IPluginChannelXMLHelper): Boolean;  
begin
```

```
④ DEWEISoft® DEWEISoft® DEWEISoft® DEWEISoft®  
// ... read/write setup data here  
if not Setup.Write then  
    MountChannels();  
    Result := True;  
end;
```

Cleaning up

It is very important that you always clear all references to the channels (i.e. set your `IChannel` variables to `nil`) you have mounted when DEWESoft® calls the `IPlugin2.ClearChannelsInstance` function. After DEWESoft® has called the `IPlugin2.ClearChannelsInstance` function, it will immediately destroy (free the memory) of the channel-objects (which implement `IChannel`).

If you had not cleaned up and accessed those stale references after the [IPlugin2.ClearChannelsInstance](#) function call, then you would read/write an undefined memory area and thus you could make the application malfunction or crash!

Mounting Channels

To actually mount a custom channel in your plugin you must can use the `IPluginGroup.MountChannelEx` function.

We will now show a simple example code which will mount 2 channels and we will discuss the steps in detail.

The first thing we do, is to add an instance variable to our plugin implementation class for each of the channels, that we want to mount (`FChOutput1` and `FChOutput2` in the example below):

```
TPluginImpl = class (...)  
private  
    FChOutput1: IChannel;  
    FChOutput2: IChannel;  
    procedure MountChannels();  
    ...
```

In the `MountChannels` function, we first need to get a reference to the [IPluginGroup](#); so that we can call the `MountChannelEx` function for each of the 2 channels - for more details see the comments in the source code.

```

procedure TPluginImpl.MountChannels;
var
  PluginGroup: IPluginGroup;
  Ind: Variant;
begin
  // get a reference to the plugin group which is the group number 8: see IData.Groups
  PluginGroup := (FApp.Data.Groups[8] as IPluginGroup);

  // now we must create a variant array of integers which will define the channel index of our channels
  // in this simple example we only mount 2 channels on the same index level, so the variant array needs o
  Ind := VarArrayCreate([0, 0], varInteger);
  // the value of the channel index must be unique - for the 1st channel we choose the value 0
  Ind[0] := 0;
  // now we mount (create) the 1st channel
  FChOutput1 := IPluginGroup.MountChannelEx(
    GUIDToString(CLASS_TPluginExample), // we must provide the GUID of the plugin
    1, // the Ind array has only one value (0 in this case)
    Ind); // the Ind array
  FChOutput1.SetAsync(False); // this channel is a synchronous channel
  FChOutput1.SetDataType(5); // data-type Single
  FChOutput1.Used := True;
  FChOutput1.MainDisplayColor := clRed;

  // the value of the channel index must be unique - for the 2nd channel we choose the value 1
  Ind[0] := 1;

```

```
DEWEsoft®  
FChOutput2 := IPluginGroup.MountChannelEx(  
    GUIDToString(CLASS_TPluginExample), // we must provide the GUID of the plugin  
    1, // the Ind array has only one value (1 in this case)  
    Ind); // the Ind array  
FChOutput1.SetAsync(True); // this channel is an asynchronous channel  
FChOutput1.SetDataType(5); // data-type Single  
FChOutput1.Used := True;  
FChOutput1.MainDisplayColor := clBlue;  
end;
```

whenever you mount a channel make sure, that you clear the channel references when [ClearChannelsInstance](#) is called:

```
procedure TPluginImpl.ClearChannelsInstance;
begin
  FChOutput1 := nil;
  FChOutput2 := nil;
end;
```

That's it - now that the channels are mounted, you can add data to the channels: see [How to: Write Data To Channels](#) for details

Array Channel

To mount an array channel you must set `IChannel.ArrayChannel` to true and provide the `ArrayInfo`.

```

procedure TPluginImpl.MountChannels;
var
  PluginGroup: IPluginGroup;
  Ind: Variant;
begin
  PluginGroup := (FApp.Data.Groups[8] as IPluginGroup);
  Ind := VarArrayCreate([0, 0], varInteger);
  Ind[0] := 0;
  FChOutArrayData := IPluginGroup.MountChannelEx(GUIDToString(CLASS_TPluginExample), 1, Ind);
  FChOutArrayData.SetAsync(True);
  FChOutArrayData.SetDataType(5); // data-type Single
  FChOutArrayData.Used := True;
  FChOutArrayData.MainDisplayColor := clRed;
  // array setup
  FChOutArrayData.ArrayChannel := True;
  FChOutArrayData.ArrayInfo.DimCount := 1;
  FChOutArrayData.ArrayInfo.DimSizes[0] := 100; // dimension 1 of the array has 100 items
  FChOutArrayData.ArrayInfo.Init();
  FChOutArrayData.ArrayInfo.AxisDef[0].AxisType := atFloatLinearFunc;
  FChOutArrayData.ArrayInfo.AxisDef[0].StartValue := 0;
  FChOutArrayData.ArrayInfo.AxisDef[0].StepValue := 1;
  FChOutArrayData.ArrayInfo.AxisDef[0].Name := 'DataPoint Index';
  FChOutArrayData.ArrayInfo.AxisDef[0].Unit := '-';

```

see also: [IChannel](#), [ArrayChannel](#)

Single Value Channel

To make the mounted channel a single value channel, just call the `SetIsSingleValue` function:

```
procedure TPluginImpl.MountChannels;
var
  PluginGroup: IPluginGroup;
  Ind: Variant;
begin
  PluginGroup := (FApp.Data.Groups[8] as IPluginGroup);
```

Notes:

- make sure, NOT to call the [IChannel.SetAsync](#) function.[Textual Channels](#)
 - in the [IPlugin2.OnGetData](#) function you must set the value ONCE via [IChannel.SingleValue](#) (or [IChannel.Text](#) if it is a [Text channel](#)) .

String channel

to mount a [String Channel](#) you must call the `IChannel.SetAsStringChannel` function and specify the length of the string.

```
procedure TPluginImpl.MountChannels;
var
  PluginGroup: IPluginGroup;
  Ind: Variant;
begin
  PluginGroup := (FApp.Data.Groups[8] as IPluginGroup);
  Ind := VarArrayCreate([0, 0], varInteger);
  Ind[0] := 0;
  FChString := IPluginGroup.MountChannelEx(GUIDToString(CLASS_TPluginExample), 1, Ind);
  FChString.SetAsync(True);
  FChString.SetAsStringChannel(20); // set the size of the string channel to 20
  FChString.Used := True;
  FChString.MainDisplayColor := clRed;
end;
```

see also: [Textual Channel - String Channel](#)

Text channel

A [text channel](#) is basically just a single value channel with [data-type](#) `Text` (11).

```
procedure TPluginImpl.MountChannels;
var
  PluginGroup: IPluginGroup;
  Ind: Variant;
begin
  PluginGroup := (FApp.Data.Groups[8] as IPluginGroup);
  Ind := VarArrayCreate([0, 0], varInteger);
  Ind[0] := 0;
  FChSingleValue := IPluginGroup.MountChannelEx(GUIDToString(CLASS_TPluginExample), 1, Ind);
  FChSingleValue.SetIsSingleValue(True);
  FChSingleValue.SetDataType(5); // data-type Single
  FChSingleValue.Used := True;
  FChSingleValue.MainDisplayColor := clRed;
end;
```

see also: [Textual Channel - Text channel](#)

1.6.2 How To Write Data To Channels

This topic will explain how to write data to DEWESoft® channels.

Before you can use your custom in DEWESoft® channels in custom plugins, you must mount the channels: see [How to: Mount Dewesoft Channels](#).

Whenever the [IPlugin2.OnGetData](#) function of your plugin is called you can write data to the custom channels of your plugin and you can also read data from other channels.

DEWESoft® will make sure, that the data that you read are consistent during the [IPlugin2.OnGetData](#) call.

You should only do the minimal required work in this function and leave it as fast as possible: i.e. if you have time-consuming tasks to do, you should do that work in a separate thread and in the [IPlugin2.OnGetData](#) call, you just add the calculated results to the channels.

How to: Write Data to single value channels

Depending on the [data type](#) of your [single value channel](#), simply assign the single value to [IChannel.SingleValue](#), or [IChannel.Text](#) (if it is a [Text channel](#)).

How to: Write Data to asynchronous channels

Depending on the [data type](#) of your [asynchronous channel](#), call the corresponding function of the [IChannel](#) interface:

[AddAsyncData](#), [AddAsyncByteSample](#), [AddAsyncShortintSample](#), [AddAsyncSmallintSample](#), [AddAsyncIntegerSample](#),
[AddAsyncInt64Sample](#), [AddAsyncSingleSample](#), [AddAsyncDoubleSample](#), [AddAsyncString](#)

Async Example

In this simple example we will mount an [asynchronous channel](#) of [data type](#) `Int64` and add a simple counter which we will increment on every [OnGetData](#) call.

On the plugin class we add a variable for the channel (`FChASync`) and another variable for the current value of the counter (`FOnGetDataCounter`):

```
TWriteChPluginImpl = class (TPluginExampleImpl)
private
  FChASync: IChannel;
  FOnGetDataCounter: Int64;
  procedure MountChannels();
  ...

```

In the [OnStartAcq](#) and [OnStartStoring](#) function we will reset the counter (the counter should always start from 0 for every measurement):

```
procedure TWriteChPluginImpl.OnStartAcq;
begin
  FOnGetDataCounter := 0;
end;
```

```
DEWEsoft® DEWEsoft®
procedure TWriteChPluginImpl.OnStartStoring;
begin
  FOnGetDataCounter := 0;
end;
```

In the [OnGetData](#) function we will first calculate the time that has passed since the start of the measurement, then add the current counter value to the channel and then increase the counter:

```
procedure TWriteChPluginImpl.OnGetData;
var
  DeweTime: Double;
  Mid: Integer;
  Dir: Integer;
  AllSamples: Int64;
begin
  FApp.Data.GetSamplesAcquired(Mid, Dir);
  AllSamples := Int64(Mid) * FApp.Data.Samples + Dir;
  // AllSamples is the total amount of samples that DEWEsoft® has already
  // collected since the start of the measurement
  DeweTime := AllSamples/FApp.Data.SampleRateEx;

  FChAsync.AddAsyncInt64Sample(FOnGetDataCounter, DeweTime);
  FOnGetDataCounter := FOnGetDataCounter + 1;
end;
```

When we start the plugin and look at the data in the recorder we can see that the counter starts at 0 and increases by 1 every time that the [IPlugin2.OnGetData](#) function is called:

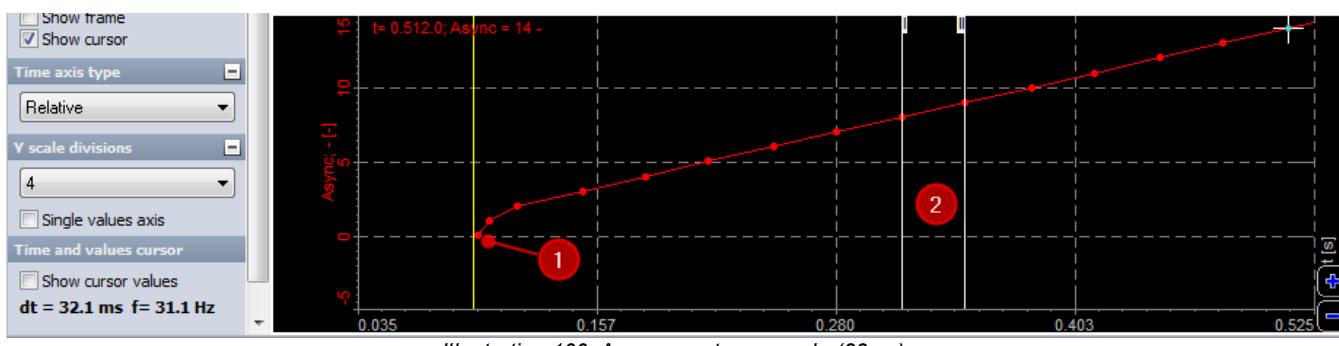


Illustration 100: Async counter example (33ms)

Thus, the time between the data points of our channel is the The [IPlugin2.OnGetData](#) function is called about every 33ms.

If you look at the start of the measurement, you can see that the first on [IPlugin2.OnGetData](#) call was executed about 100ms after the start of the acquisition and that the second call has been executed about 6ms after the first call. This should make it clear, that the time specified in the [IApp.TimerInterval](#) is just used as a timer interval - **NEVER rely on the fact that it is called exactly every 33ms!**

How to: Write Data to synchronous channels

Depending on the [data type](#) of your [synchronous channel](#), call the corresponding function of the [IChannel](#) interface:

see also: [AddByteSample](#), [AddWordSample](#), [AddShortintSample](#), [AddSmallintSample](#), [AddIntegerSample](#),
[AddIn64Sample](#), [AddSingleSample](#), [AddDoubleSample](#)

For best performance you can add a complete block of data sample (of [data type Single](#)) with only one DCOM call:

[AddSingleSamples](#)

Number of samples

When *DEWESoft®* calls the [OnGetData](#) function, you must add the correct number of samples to all the synchronous channels that you have mounted. The following code fragment shows how to calculate the number of expected samples:

```
procedure TWriteChPluginImpl.OnGetData;
var
  DeweTime: Double;
  Mid: Integer;
  Dir: Integer;
  AllSamples: Int64;
begin
  FApp.Data.GetSamplesAcquired(Mid, Dir);
  AllSamples := Int64(Mid) * FApp.Data.Samples + Dir;
  // AllSamples is the total amount of samples that DEWESoft® has already
  // collected since the start of the measurement
...

```

So: *DEWESoft®* expects that you have added exactly the number `AllSamples` samples to your synchronous channels. If you cannot add all the expected samples, you must inform *DEWESoft®* about it, by setting the [IChannel.CalcDelay](#) to the number of samples that you could not provide yet.

You must also take into account any sample rate divider ([IChannel.SRDiv](#)) that may be set on your channel: e.g. when [IChannel.SRDiv](#) is set to 2, then you must only fill every 2nd sample.

1.6.3 How To Find Channels

This topic will explain how to find *DEWESoft®* channels (i.e. get a reference to the [IChannel](#) interface of the channel); e.g. to read the data from a specific channel (see also: [How To Read Data From Channels](#)).

Channel lists

One way is to iterate over all channels that have been set to *Used* via [IData.UsedChannels](#). Note: If the status of used has changed (from *Used* to *Unused* or vice versa), then you must call [BuildChannelList](#) to update the [IData.UsedChannels](#) list.

Find functions

Another way is to call the [IData.FindChannelByIndexEx](#) function to search for a channel by its [Channel Index](#).

You could also use the [IData.FindChannel](#) function which searches for a channel by its name, but since several channels could have the same name, this is not recommended.

1.6.4 How To Read Data From Channels

This topic will explain how to read data from *DEWEsoft®* channels. Before you can read data from a channel, you must of course find the channel (i.e. get a reference to the [IChannel](#) interface): see [How To Find Channels](#)

Whenever the [IPlugin2.OnGetData](#) function of your plugin is called you can read data from all *DEWEsoft®* channels. *DEWEsoft®* will make sure, that the data that you read are consistent during the [IPlugin2.OnGetData](#) call.

How to: Read Data from single value channels

Depending on the [data type](#) of the [single value channel](#), simply read the single value from [IChannel.SingleValue](#), or [IChannel.Text](#) (if it is a [Text channel](#)) .

How to: Read Data from asynchronous channels

For [asynchronous channels](#), reading the data involves 2 things: reading the values of the channel and reading the timestamp information relating to those values.

The *DEWEsoft®* DCOM interface offers several methods how to read data from *DEWEsoft®* channels.

Method	Ease-of-use	Performance ¹⁾	Read Values	Read Timestamps
Read data via IChannelConnection	easy	good	GetDataValues or GetDataBlocks	GetTSValues or GetTSBlocks
Read data from direct buffer	medium	low	DBValues	DBTimeStamp
Read block from direct buffer	easy	good	GetScaledDataEx / GetUnscaledDataEx	GetTSDataEx
Read data via memory address ²⁾	difficult	best	GetDBAddress	GetTSAddress

1. in this context the performance is just coarse definition of how many DCOM calls are involved:

e.g. when you call read 100 samples value by value via [DBValues](#) you need 100 DCOM (1 for each value)

when you read the 100 samples at once via [GetScaledDataEx](#) then you only need 1 DCOM call (which will return an array of 100 samples)

less DCOM calls do of course mean better performance

2. only possible in custom plugins that run in the same address space as the *DEWEsoft®* application

How to: Read Data from synchronous channels

Reading the values of [synchronous channels](#) is the same as for asynchronous channels (see [How to: Read Data from asynchronous channels](#)) above. In this case you need not read any timestamps, because the data is synchronous anyway.

When you need to know the timestamp that corresponds to a certain sample, you can easily calculate it: just divide the absolute sample number by [IData.SampleRateEx](#).

2 Reference

the DCOM reference includes an extensive reference describing all the interfaces, methods, enumerations types and constants of DEWEsoft®'s DCOM interface.

2.1 Interfaces

This chapter contains a list of all interfaces for DEWEsoft® DCOM.

2.1.1 IAISetupScreen

This interface can be used to access the analogue channel setup:



Illustration 101: Analog channel setup

2.1.1.1 SetColumnVisible

```
procedure SetColumnVisible(ColNo: Integer; Visible: WordBool);
```

To show or hide column of the setup screen for analog channels specified by `ColNo`. (starting at index 1)

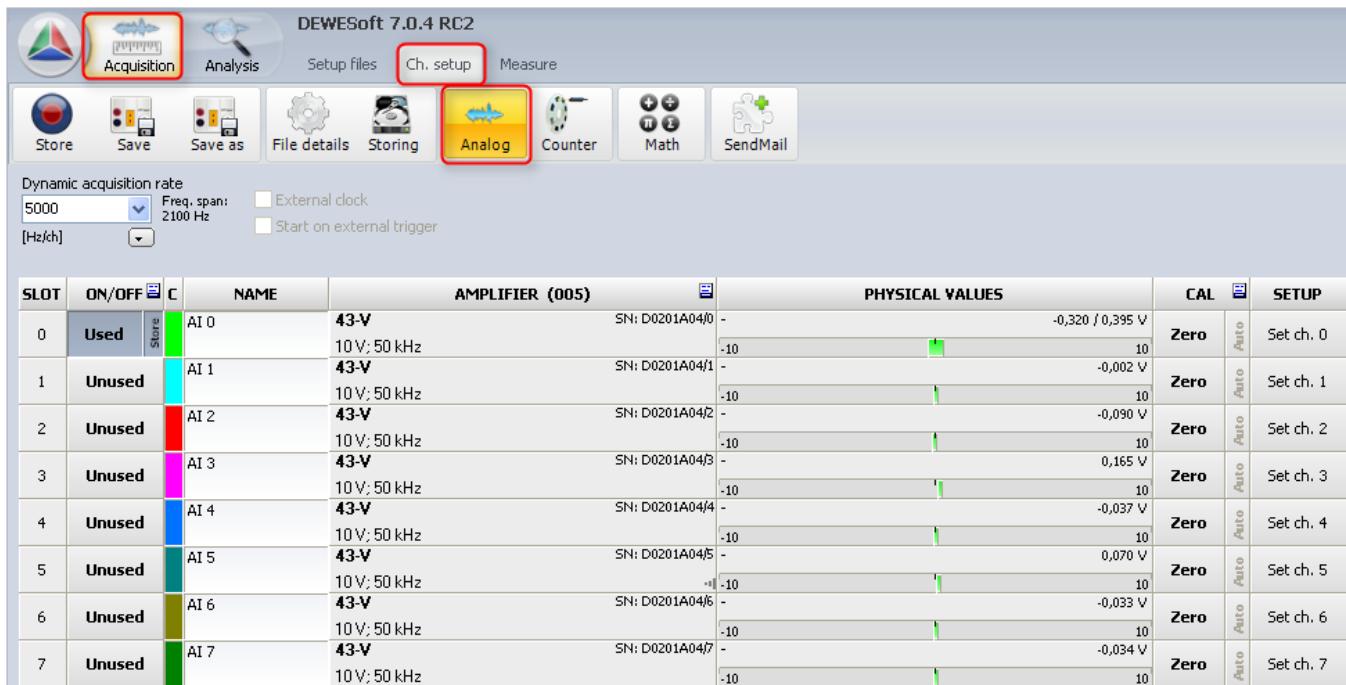


Illustration 102: Analog channel setup: all columns visible

When you call and then the corresponding columns (`SLOT` and `C`) will be hidden:



Illustration 103: Analog channel setup: columns 1 and 3 hidden

Interface: [IAISetupScreen](#)

Classifier	Name	Type	Description
	ColNo	Integer	the number of the column that will be set visible or hidden, starting with index 1
	Visible	WordBool	<i>TRUE</i> will show the column <i>FALSE</i> will hide the column

2.1.1.2 ShowChannelSetup

```
procedure ShowChannelSetup(ChNo: Integer);
```

To open the setup dialogue for the analog channel with the index ChNo (starting from 0). This is the same as clicking on the *Setup* button of this channel.

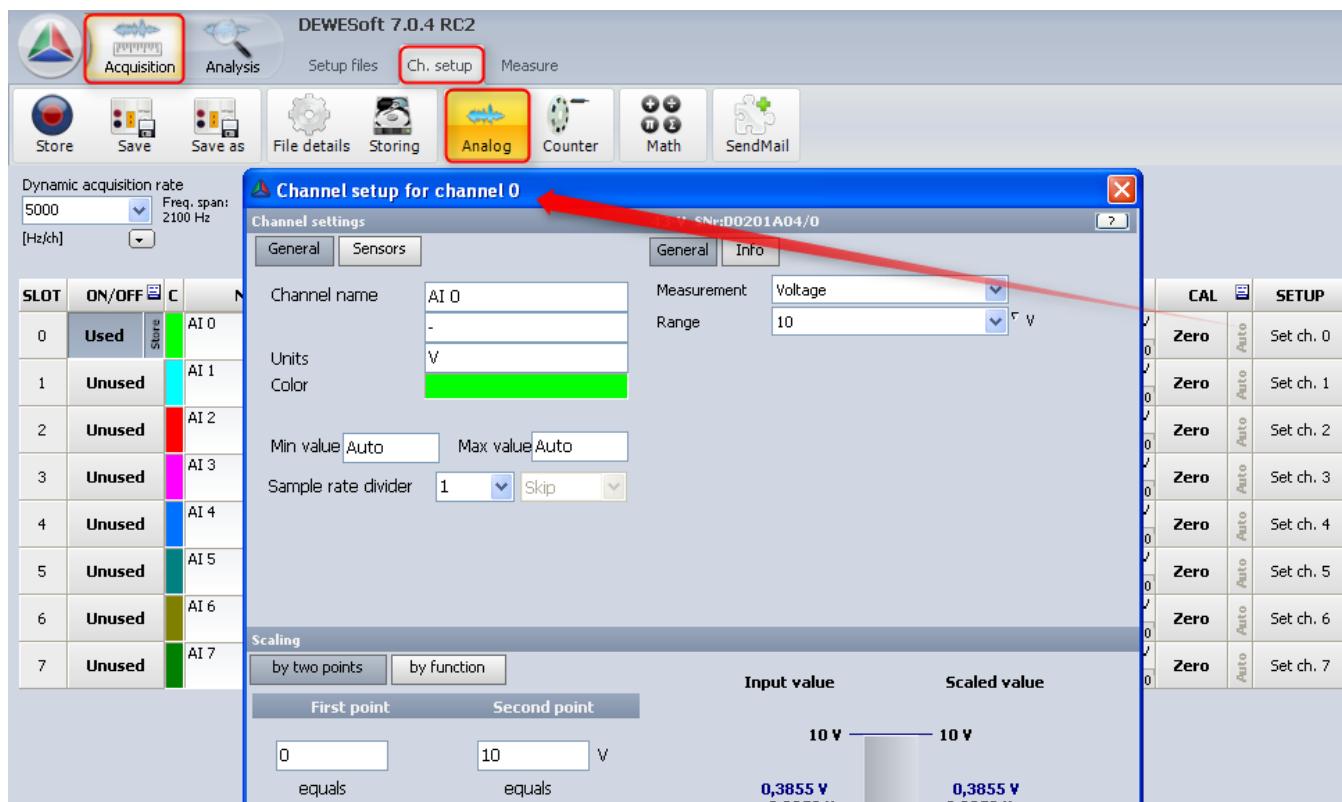


Illustration 104: Analog channel setup: Channel setup dialogue

Interface: [IAISetupScreen](#)

Classifier	Name	Type	Description
	ChNo	Integer	the index of the channel to open the setup dialogue for (starting with 0)

2.1.2 IAOChannel

This interface can be used to access the properties of an analogue output channel:

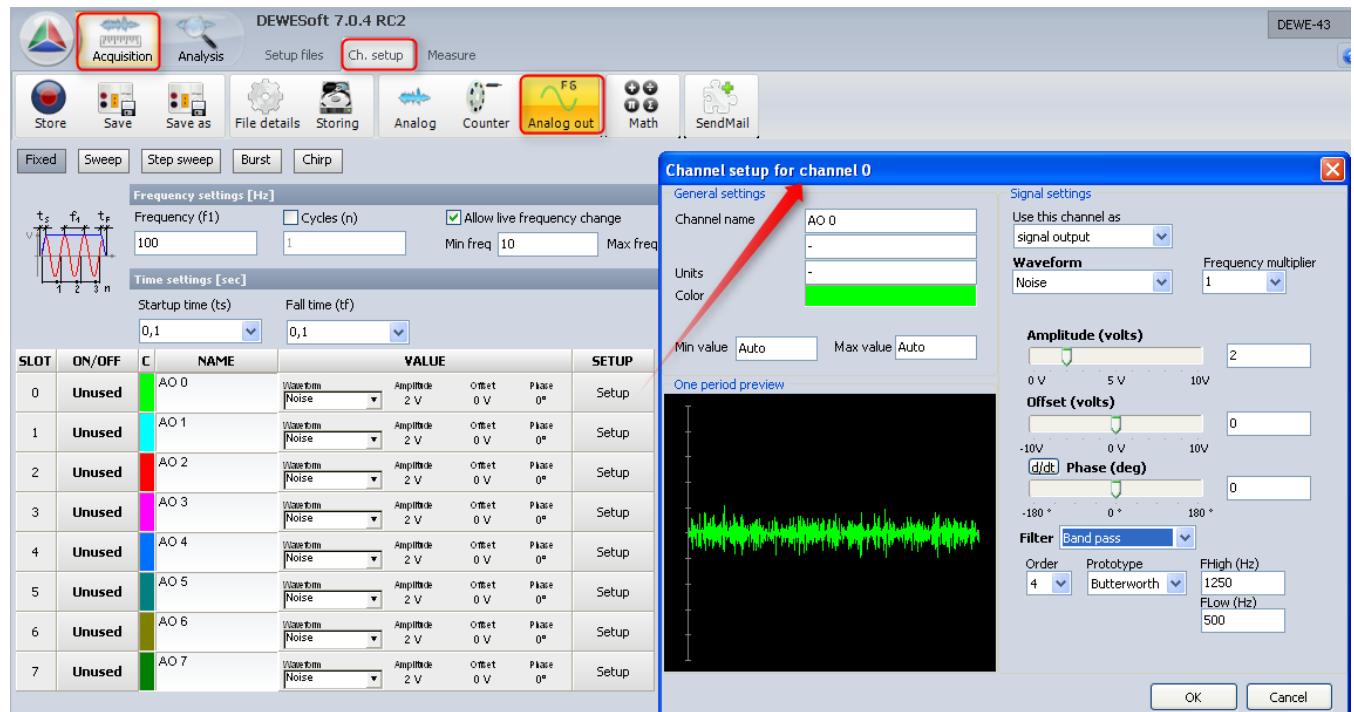


Illustration 105: Analog output channel setup

Note: the *Analog out* icon is only visible if the *Analog out* function has been activated in *Hardware setup*.

see also [IAOGroup.AOChannels](#)

2.1.2.1 Ampl

property Ampl: Single

The amplitude of the output signal in volts.

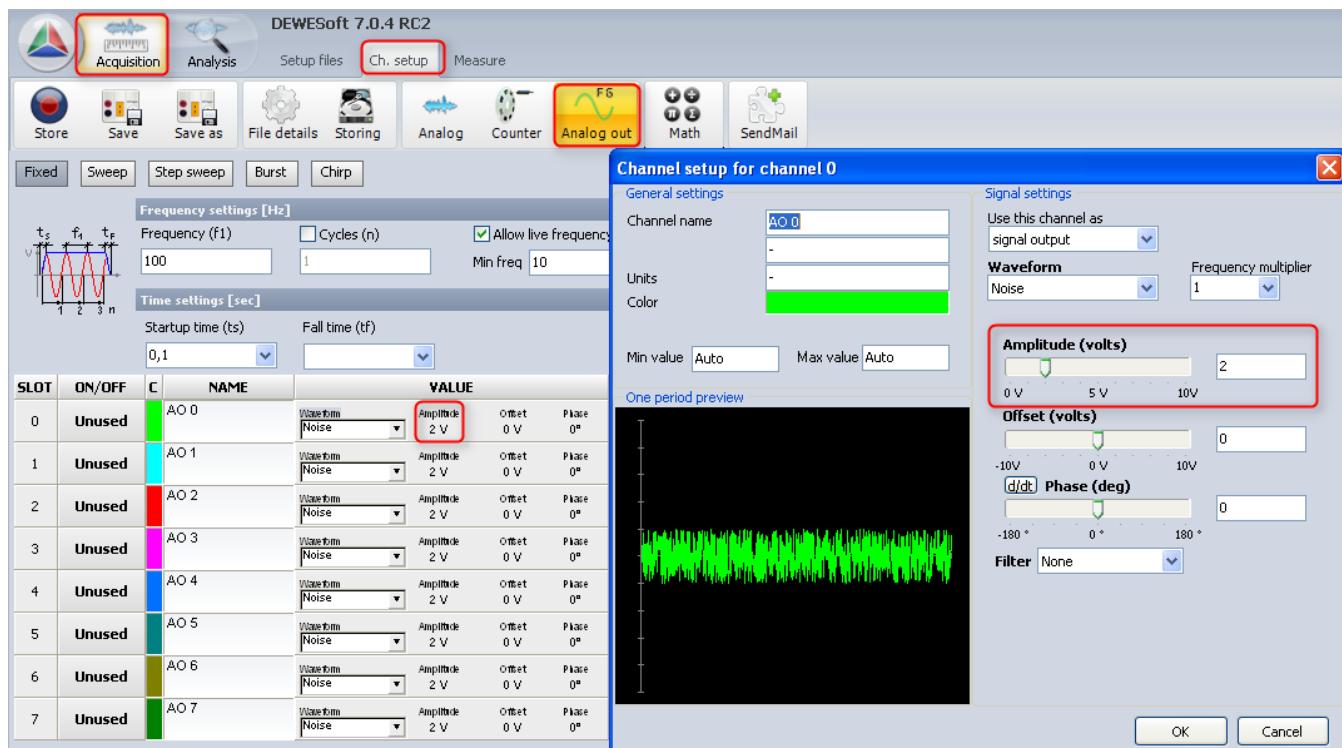


Illustration 106: Analog output channel setup: amplitude

Interface: [IAOChannel](#) read/write

2.1.2.2 FilterFreq1

property FilterFreq1: Single

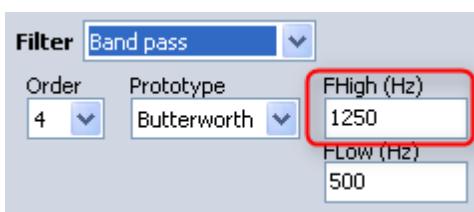
The first frequency of a filter.

If you have a low pass filter, it relates to *FHigh (Hz)*:



Illustration 107: Analog output channel setup: filter:
low pass

for a band pass filter it also relates to *FHigh (Hz)*:



*Illustration 108: Analog output channel setup:
filter: band pass*

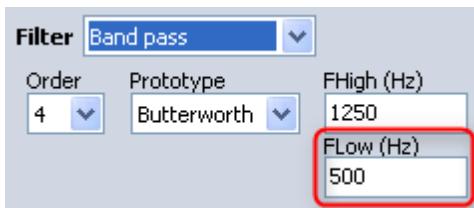
Interface: [IAOChannel](#) read/write

2.1.2.3 FilterFreq2

property FilterFreq2: Single

The second frequency of a filter.

for a band pass filter it also relates to F_{Low} (Hz):



*Illustration 109: Analog output channel setup:
filter: band pass*

Interface: [IAOChannel](#) read/write

2.1.2.4 FilterOrder

property FilterOrder: Integer

The Order of a low pass or band pass filter:



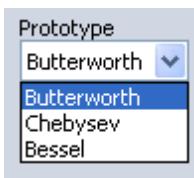
Illustration 110:
*Analog output
channel setup: filter:
band pass*

Interface: [IAOChannel](#)  read/write

2.1.2.5 FilterProtoType

property FilterProtoType: Integer

FilterProtoType refers to the filter's prototype which can be e.g. Butterworth, Chebyshev or Bessel.



*Illustration 111:
Analog output
channel setup:
filter: filter
prototype*

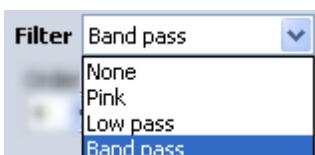
Interface: [IAOChannel](#)  read/write

2.1.2.6 FilterType

property FilterType: Integer

FilterType is the type of a filter which can be one of the following: *None, Pink, LowPass, Band pass*.

Only applicable if the [IAOChannel.WaveForm](#) is *Noise*.



*Illustration 112: Analog output
channel setup: filter: type*

Interface: [IAOChannel](#)  read/write

2.1.2.7 Offset

property Offset: Single

Offset specifies the offset value of the output signal in volts.

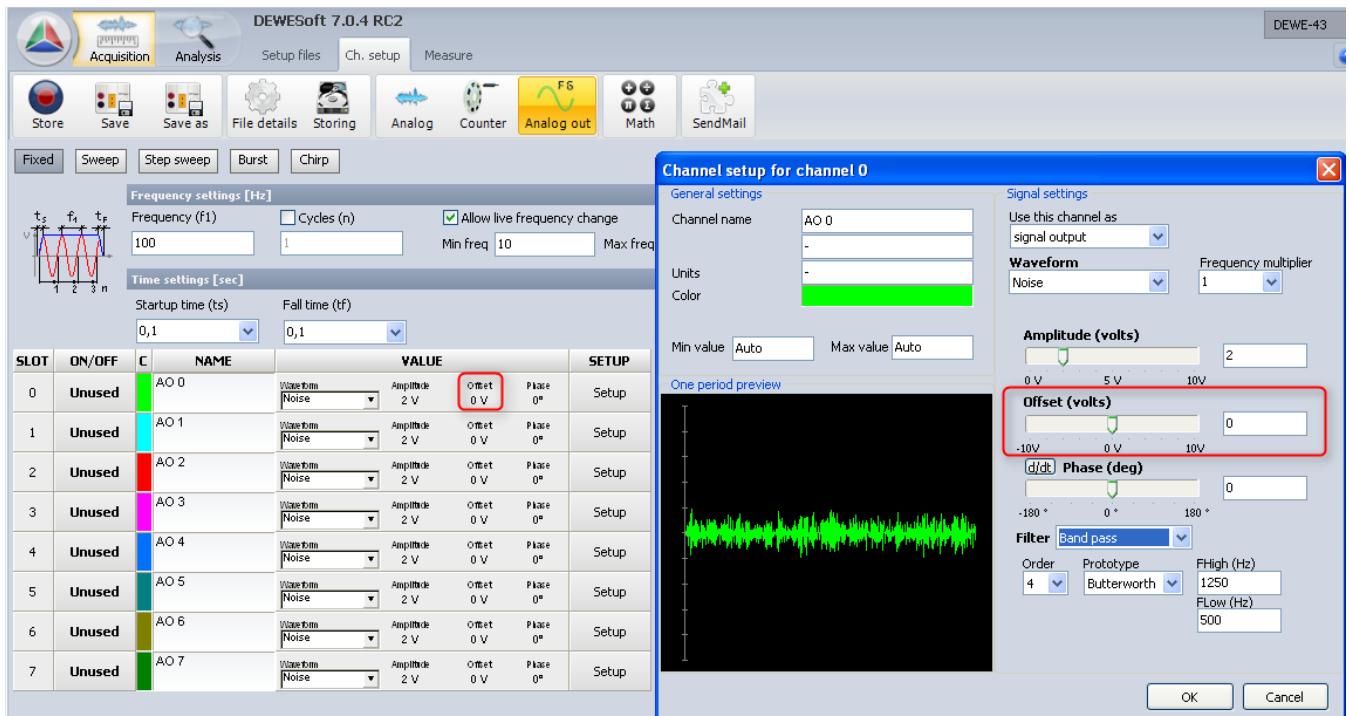


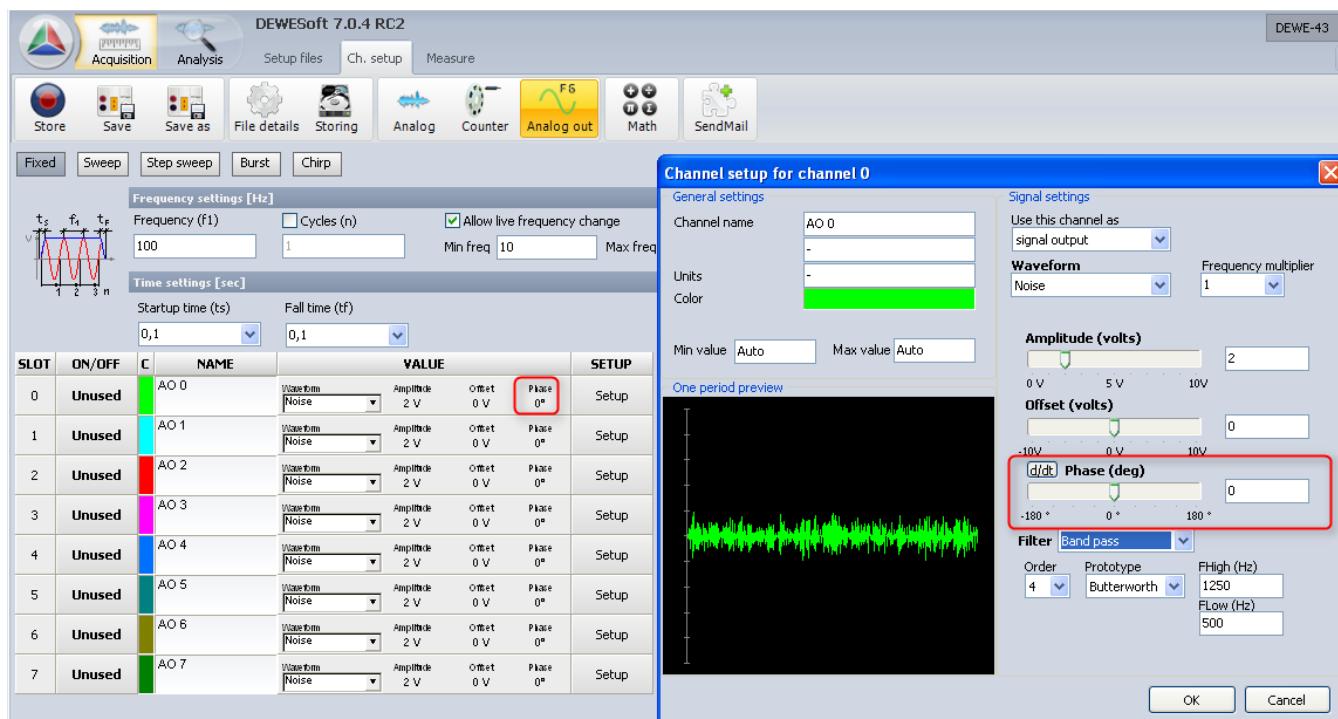
Illustration 113: Analog output channel setup: offset

Interface: [IAOChannel](#) read/write

2.1.2.8 Phase

property Phase: Single

Phase specifies the phase angle of the output signal in degrees:



Interface: [IAOChannel](#) read/write

2.1.2.9 Range

property Range: Integer

Range defines the voltage range of the output channel.

0...±10V

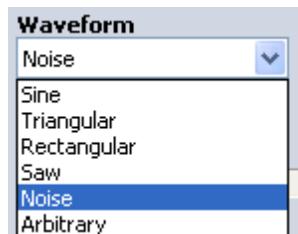
1...±1V

Interface: [IAOChannel](#) read/write

2.1.2.10 WaveForm

property WaveForm: [AOWaveForm](#)

The type of waveform to output: see also [AOWaveForm](#)



Interface: [IAOChannel](#) read/write

2.1.3 IAOGroup

Interface to access properties and channels of the analog output group:

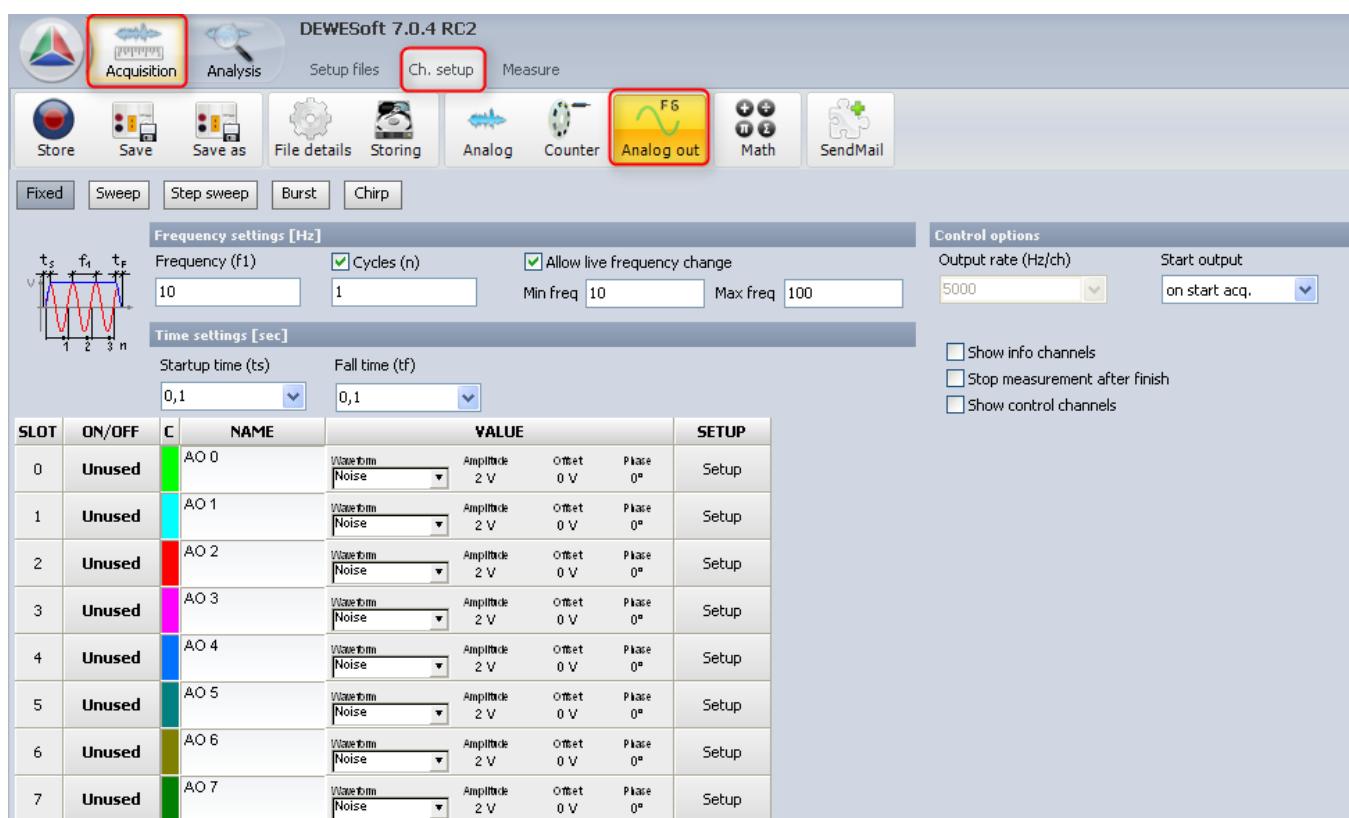


Illustration 114: Analog output channel setup

see also: [IApp.AOGroup](#), [IAOChannel](#)

2.1.3.1 AOChannels

property **AOChannels**: [IChannelList](#)

AOChannels provides a list of analog output channels (see: [IAOChannel](#)). It is of the type [IChannelList](#).

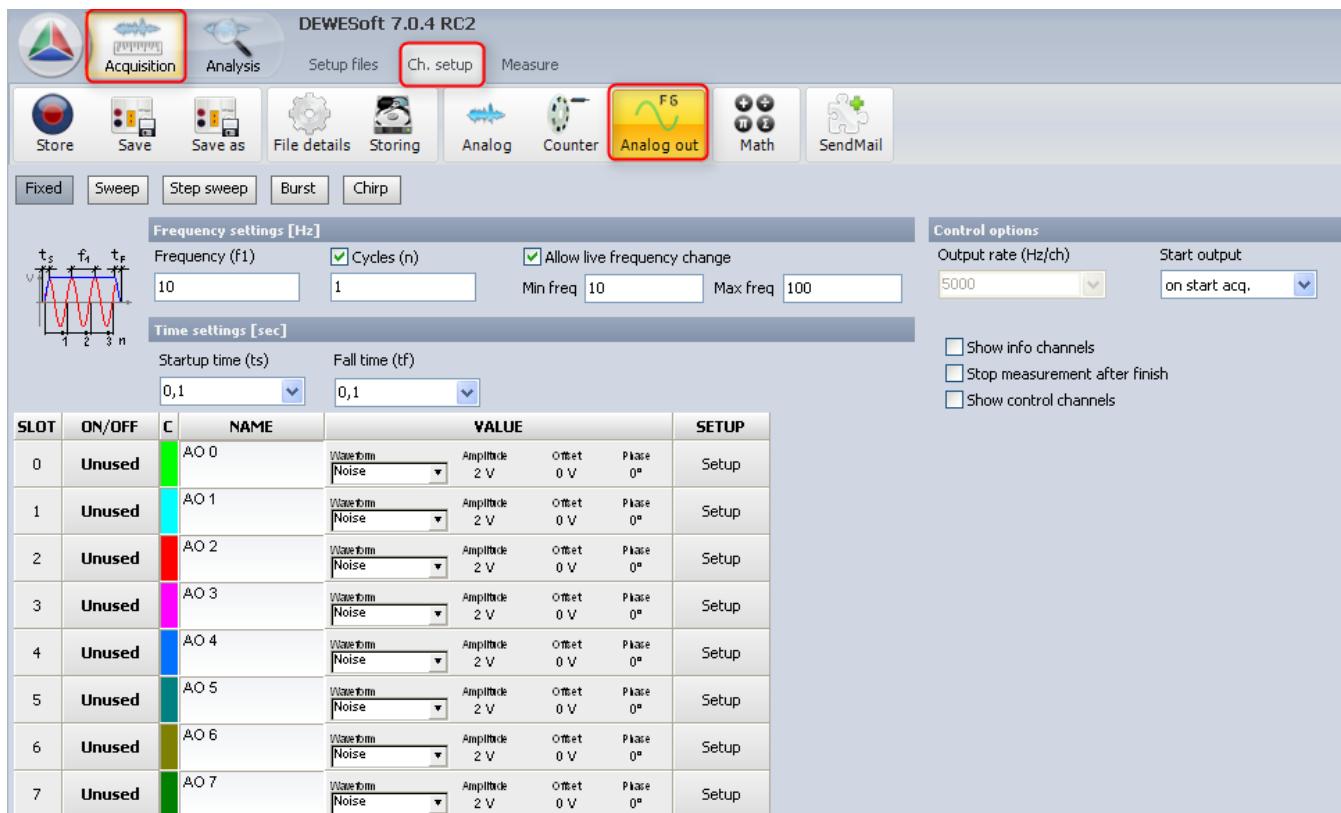


Illustration 115: Analog output channel setup: channel list

Interface: [IAOGroup](#)



2.1.3.2 AmplChangeFactor

property **AmplChangeFactor**: Single

AmplChangeFactor is the rate at which amplitude changes are done. Its unit is [V/s].

Note: the *Show control channels* check box must be activated.

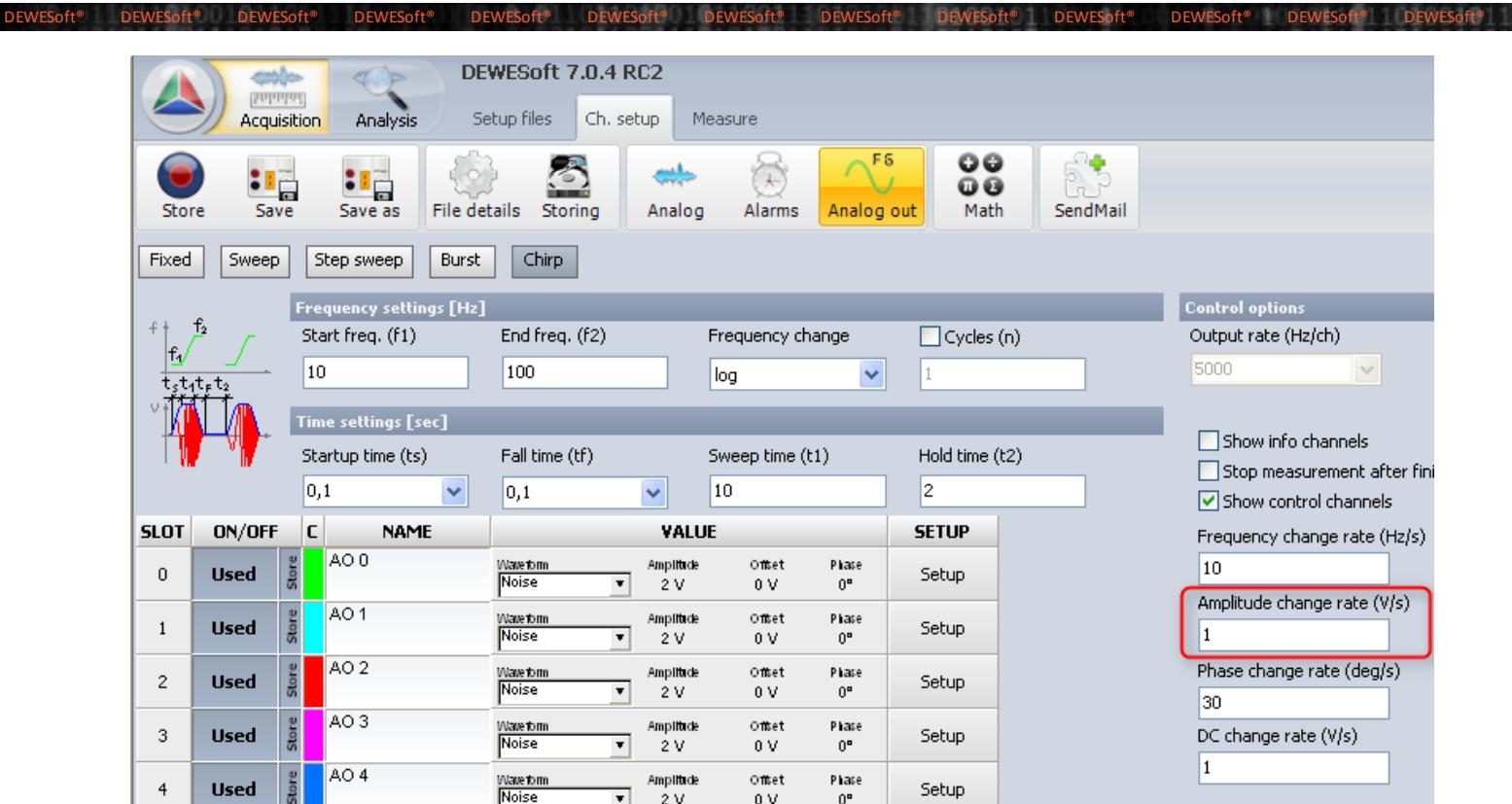


Illustration 116: Analog output channel setup: Amplitude change rate

The value can also be changed during measurement via a control channel:

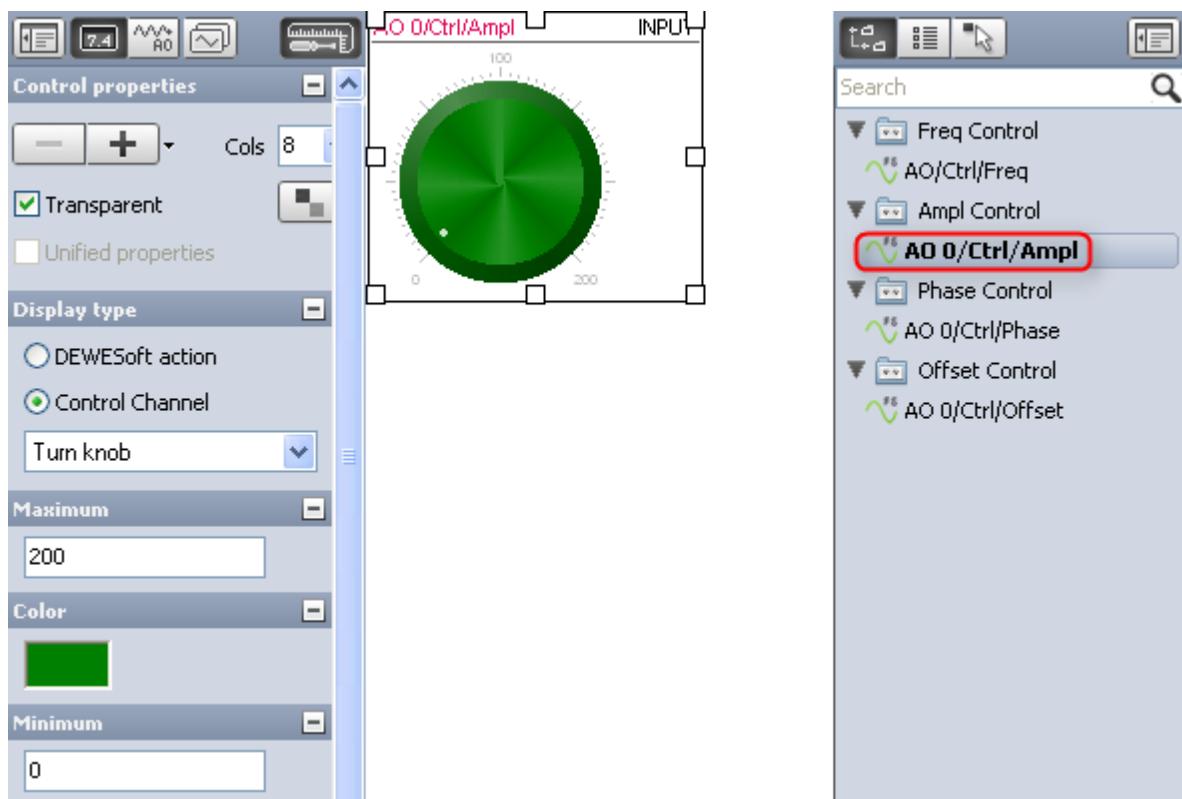


Illustration 117: Analog output channel setup: Amplitude change rate control channel

Interface: [IAOGroup](#)

read/write

2.1.3.3 ControlsClock

property ControlsClock: WordBool

ControlsClock is true, if the analog output-board is the master clock (dependant on the hardware setup: if no analogue input card is enabled).

Interface: [IAOGroup](#) read-only

2.1.3.4 DCChangeFactor

property DCChangeFactor: Single

DCChangeFactor is the rate at which DC value changes are done. Its unit is [V/s].

Note: the *Show control channels* check box must be activated (if you want to control it via a control channel).



Illustration 118: Analog output channel setup: DC change rate

The value can also be changed during measurement on the Analog output property page:



Illustration 119: Analog output property page

The value can also be changed during measurement via a control channel:

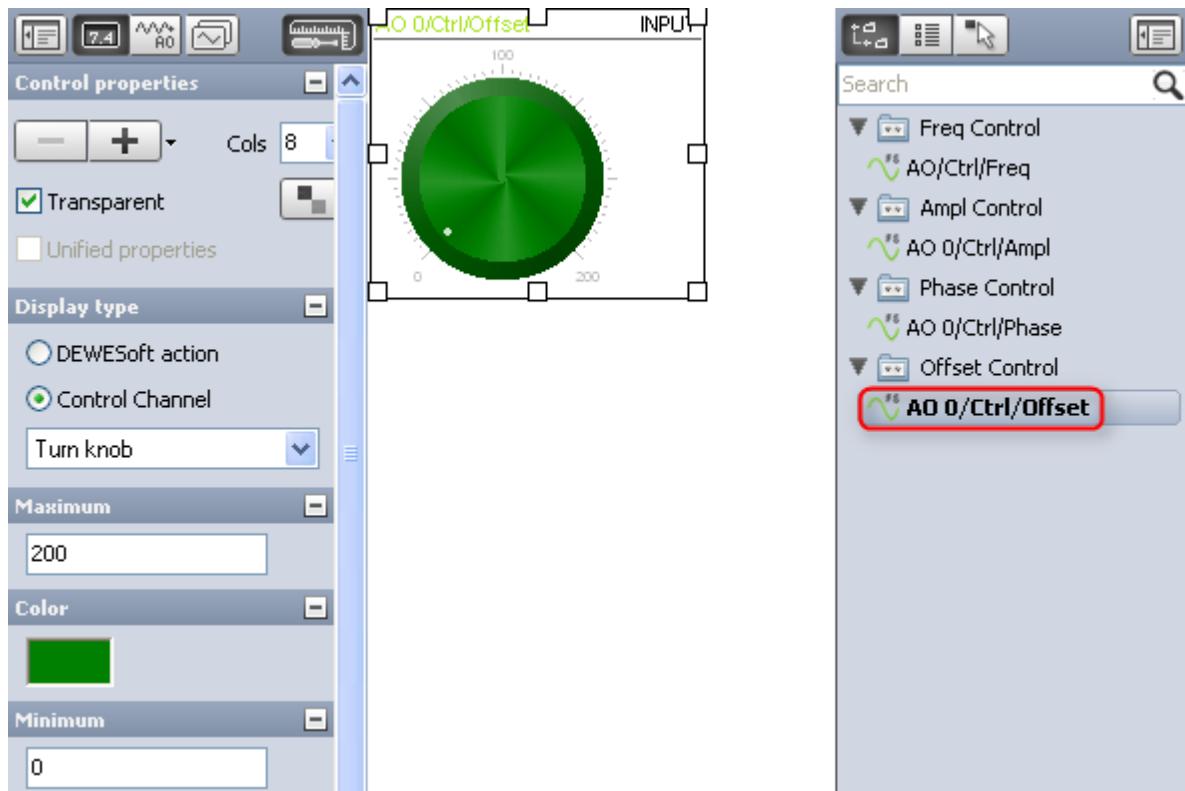


Illustration 120: Analog output channel setup: DC change rate control channel

Interface: [IAOGroup](#)



2.1.3.5 DeltaFreq

property DeltaFreq: Single

`DeltaFreq` is the frequency difference between subsequent steps of a *Step sweep* (see [OperationMode](#)) signal.



Illustration 121: Analog output channel setup: Steep sweep: Delta frequency

Interface: [IAOGroup](#)



2.1.3.6 Freq

property Freq: Single

`Freq` is the frequency of a signal in *Fixed* operation mode (see [OperationMode](#)).

Note: other operation modes use `StartFreq`, `StopFreq`

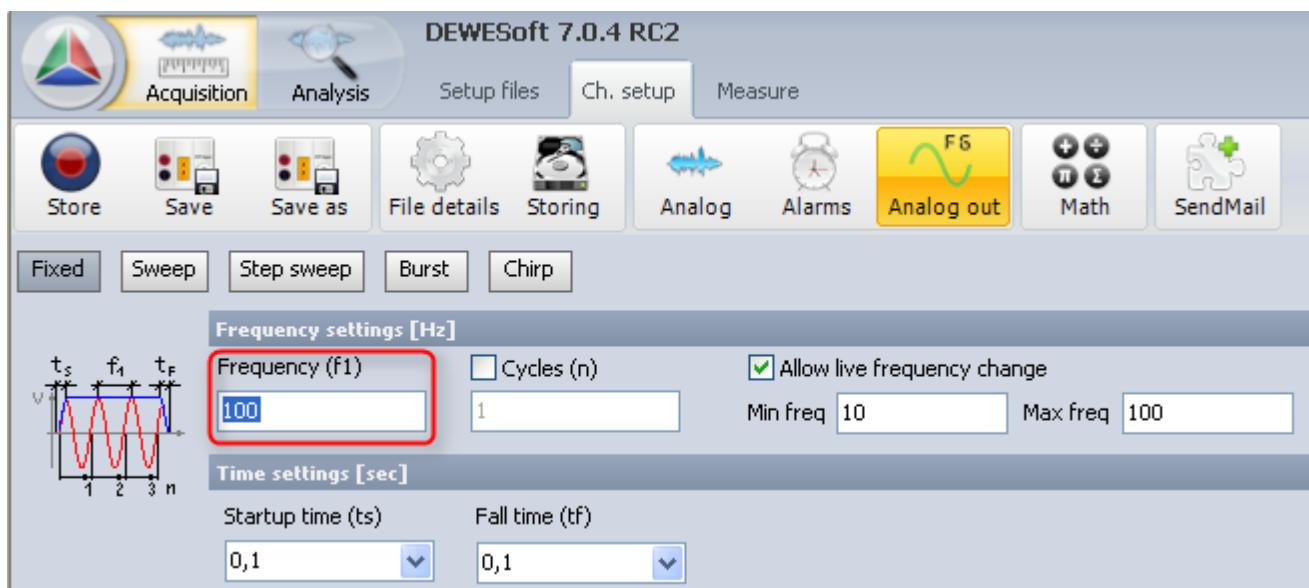


Illustration 122: Analog output channel setup: Fixed: Frequency

Interface: IAOGroup



2.1.3.7 FreqChangeFactor

property FreqChangeFactor: Single

FreqChangeFactor is the rate at which frequency changes are done. Its unit is [Hz/s].

Note: the *Show control channels* check box must be activated.

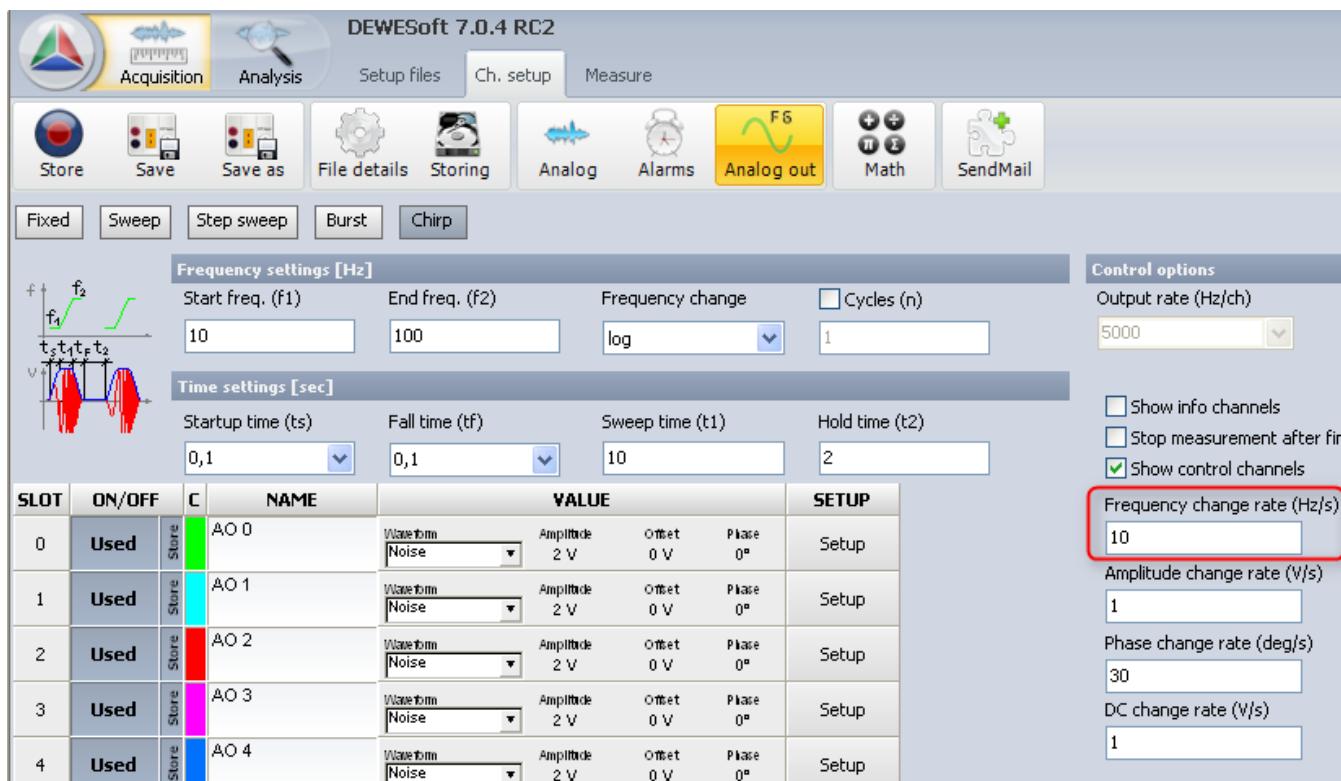


Illustration 123: Analog output channel setup: Frequency change rate

The value can also be changed during measurement via a control channel:

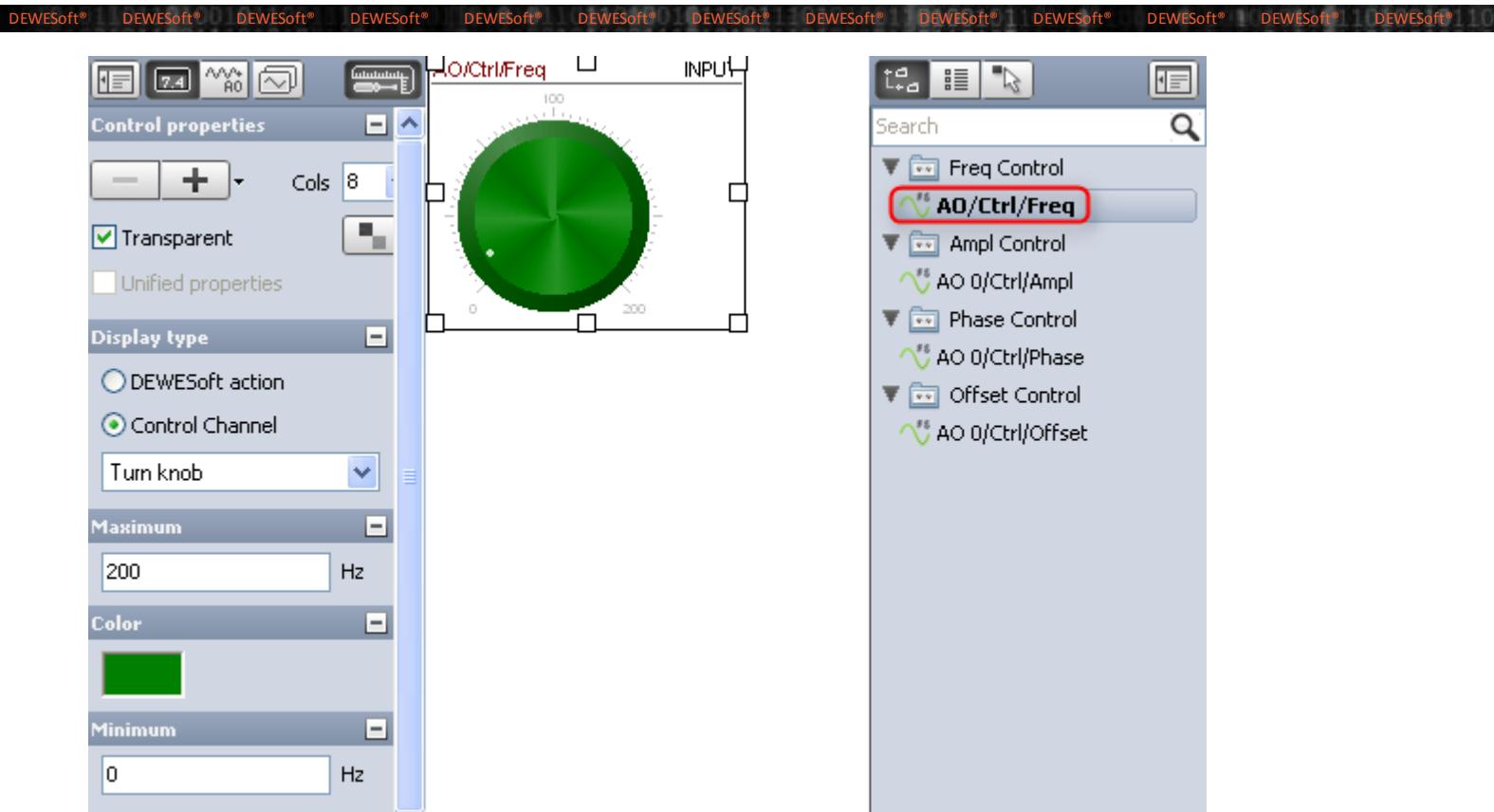


Illustration 124: Analog output channel setup: Frequency change rate control channel

Interface: [IAOGroup](#)

read/write

2.1.3.8 LogSweep

property LogSweep: WordBool

LogSweep defines whether the frequency change of a sweep is logarithmic (*TRUE*) or linear (*FALSE*).

This is only applicable for the operation modes *Sweep* and *Chirp* (see [OperationMode](#)).



Illustration 125: Analog output channel setup: Sweep: LogSweep

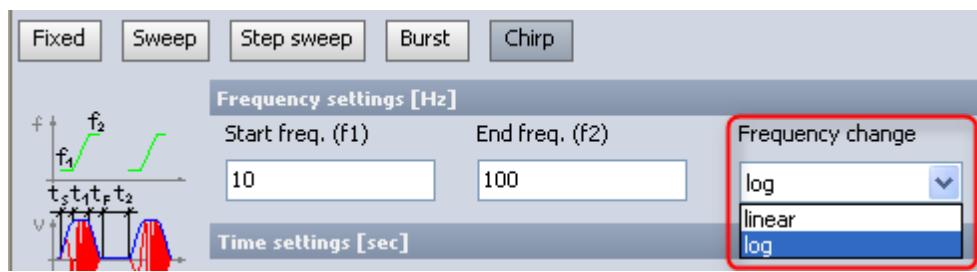


Illustration 126: Analog output channel setup: Chirp: LogSweep

Interface: [IAOGroup](#)

2.1.3.9 OperationMode

property OperationMode: [AOOperationMode](#)

OperationMode defines the mode of the analogue output function generator: see [AOOperationMode](#) for valid values.



Illustration 127: Analog output channel setup: Operation modes

Interface: [IAOGroup](#)

2.1.3.10 PhaseChangeFactor

property PhaseChangeFactor: Single

PhaseChangeFactor is the rate at which phase changes are done. Its unit is [°/s].

Note: the *Show control channels* check box must be activated.

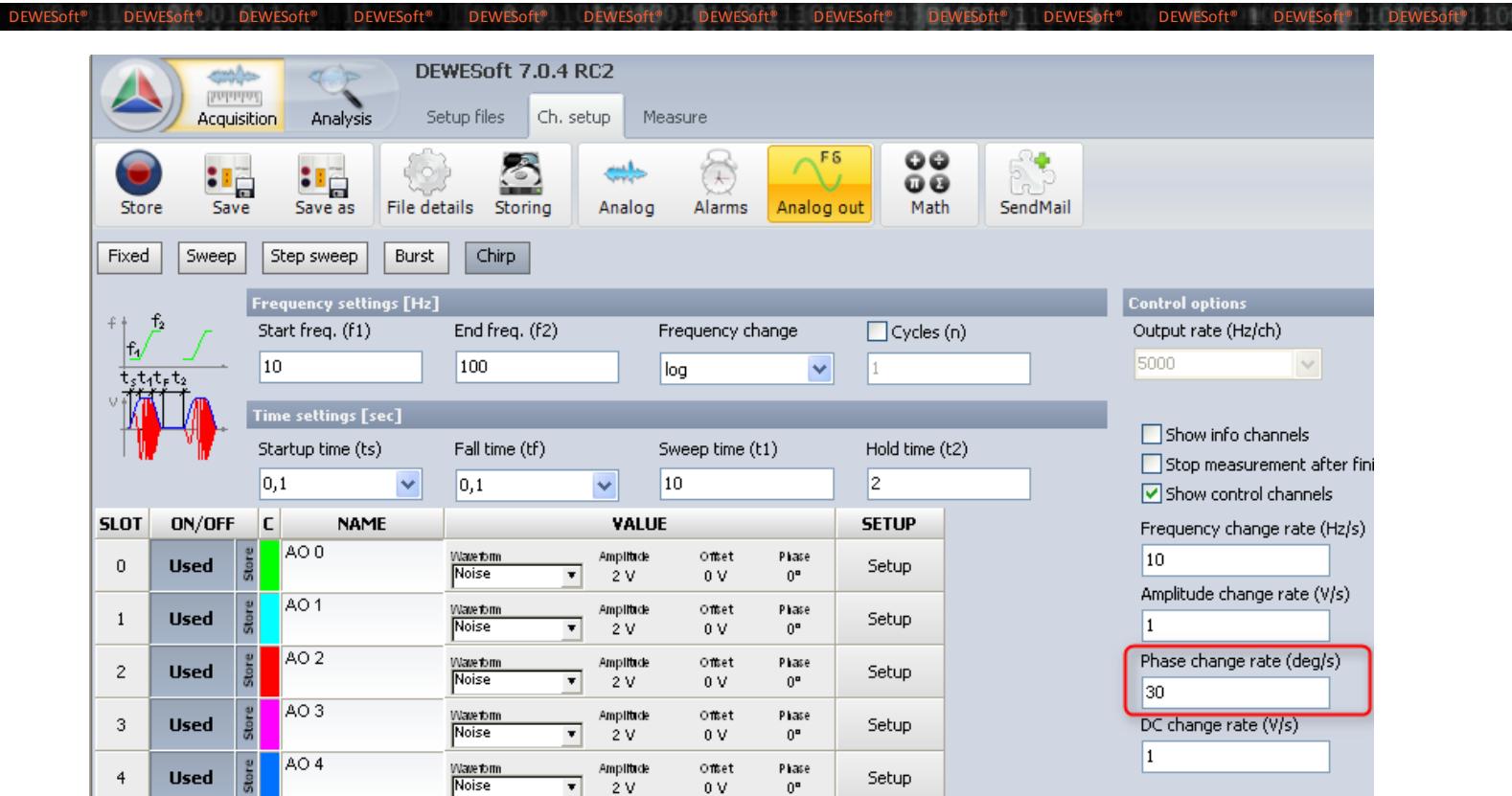


Illustration 128: Analog output channel setup: Phase change rate

The value can also be changed during measurement via a control channel:

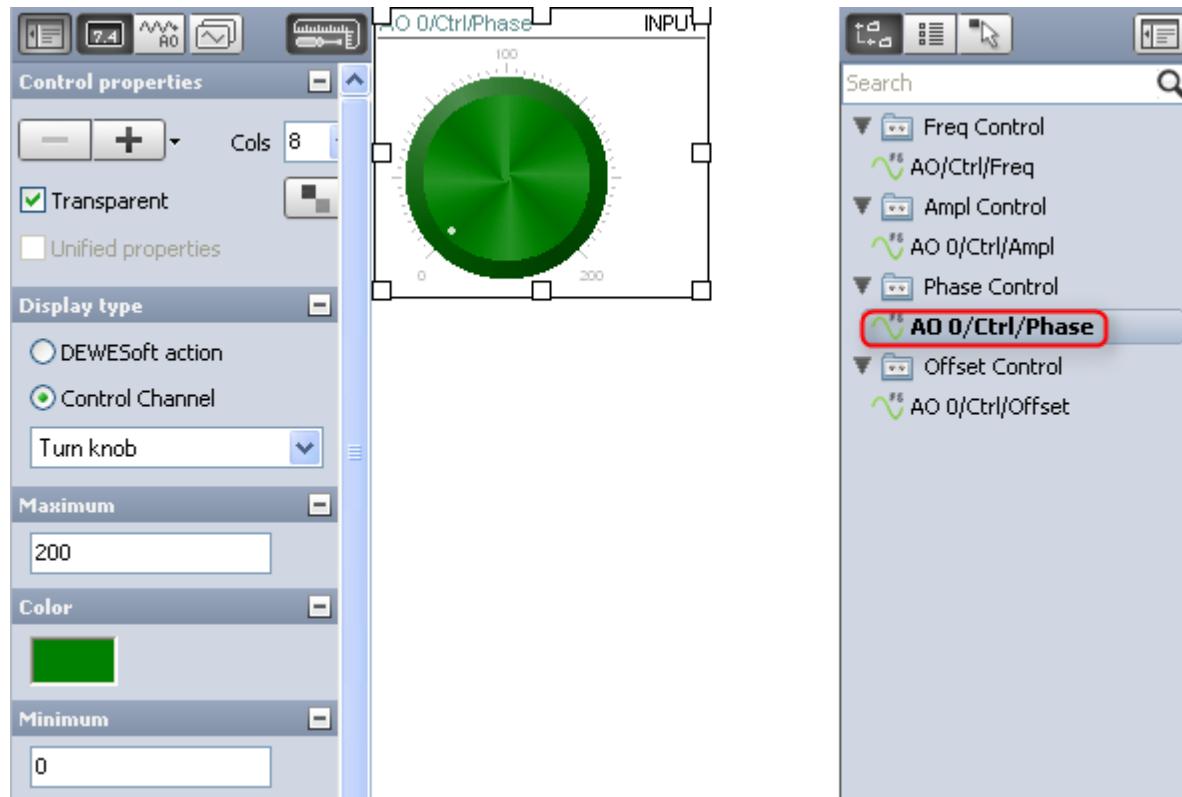


Illustration 129: Analog output channel setup: Phase change rate control channel

Interface: [IAOGroup](#)

read/write

2.1.3.11 SampleRate

property SampleRate: Integer

SampleRate is the sample rate for analog output signals in Hz/ch.

Why is the output rate deactivated? because I have no real hardware? - Yes (also if hard sync is enable).

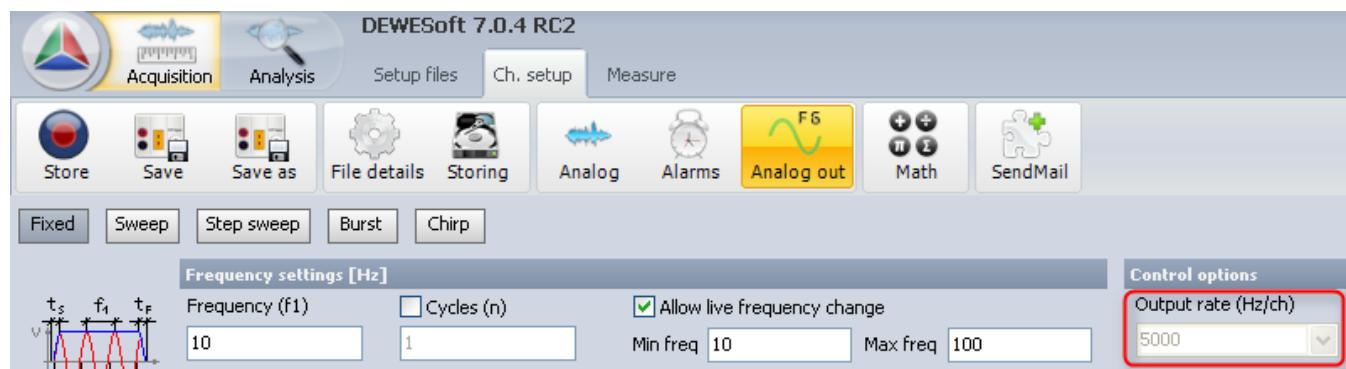


Illustration 130: Analog output channel setup: Output rate

Interface: [IAOGroup](#)

read/write

2.1.3.12 ShowInfoChannels

property ShowInfoChannels: WordBool

ShowInfoChannels defines whether informative channels should be available during measurement.:

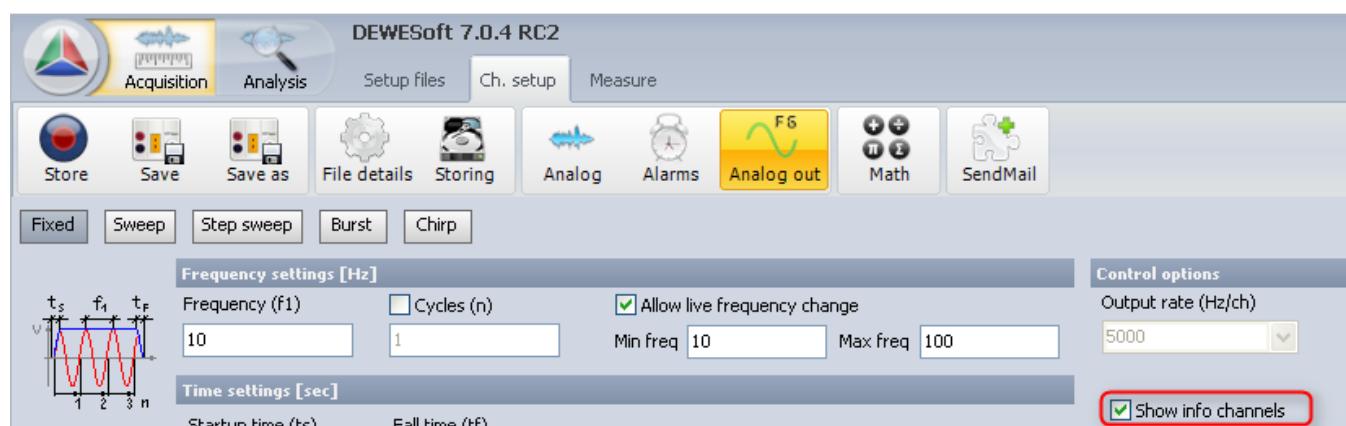


Illustration 131: Analog output channel setup: Show info channels

The following information channels are available: *Freq, Trig, Ampl, Phase, Offset*



Illustration 132: Analog output channel setup: Info channels

Interface: [IAOGroup](#)

read/write

2.1.3.13 StartFreq

property StartFreq: Single

StartFreq defines the start frequency of a signal. Applicable for the operation modes *Sweep*, *Step sweep* or *Chirp* (see [OperationMode](#)).

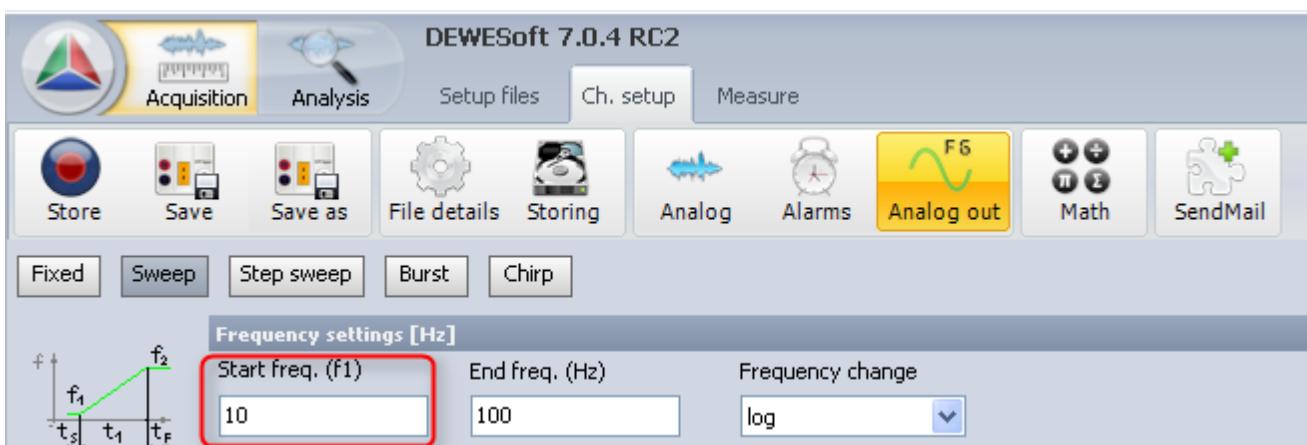


Illustration 133: Analog output channel setup: Start frequency

Interface: [IAOGroup](#)

read/write

2.1.3.14 StartTime

property StartTime: Single

`StartTime` defines the startup time of a signal.

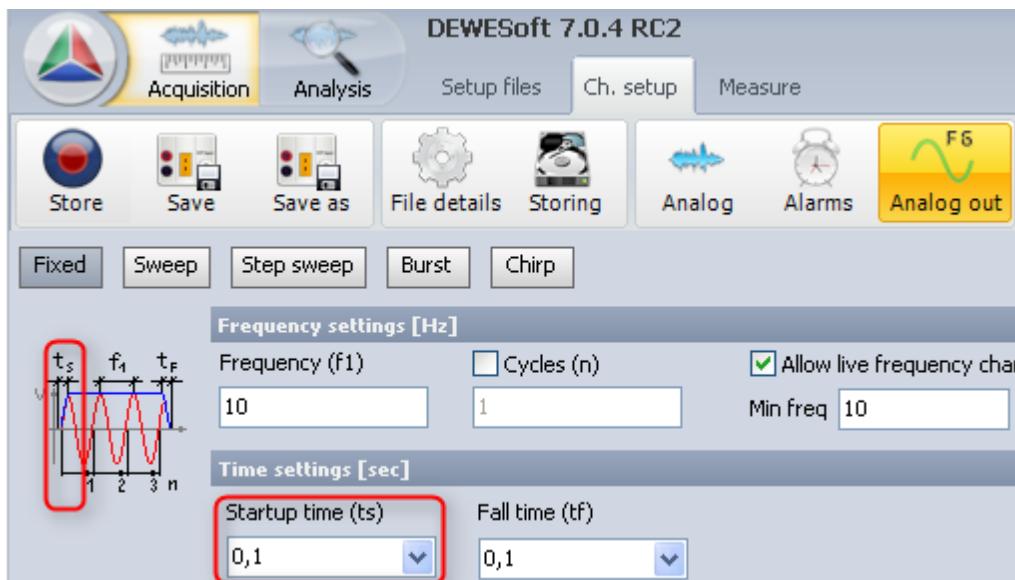


Illustration 134: Analog output channel setup: Startup time

Interface: [IAOGroup](#)



2.1.3.15 StopFreq

property StopFreq: Single

`StopFreq` is the end frequency of a signal. Applicable for the operation modes Sweep, Step sweep or Chirp (see [OperationMode](#)).

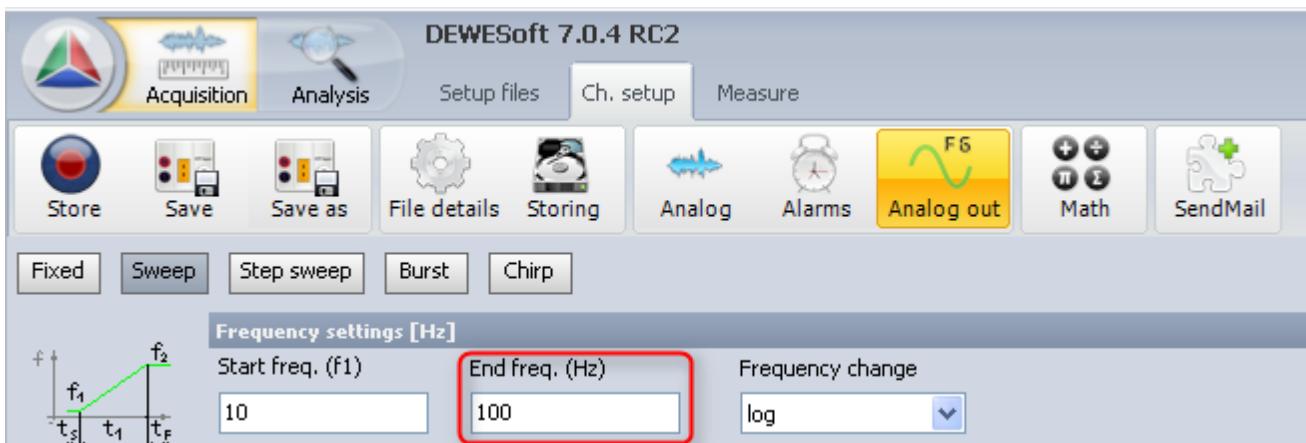


Illustration 135: Analog output channel setup: End frequency

Interface: [IAOGroup](#)



2.1.3.16 StopTime

property StopTime: Single

StopTime defines the fall time of a signal.

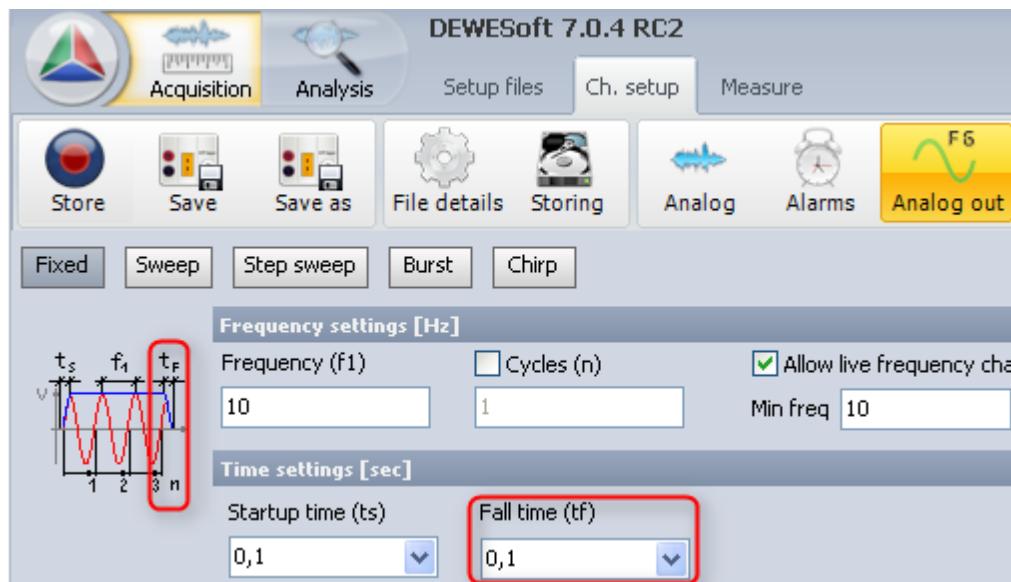


Illustration 136: Analog output channel setup: Fall time

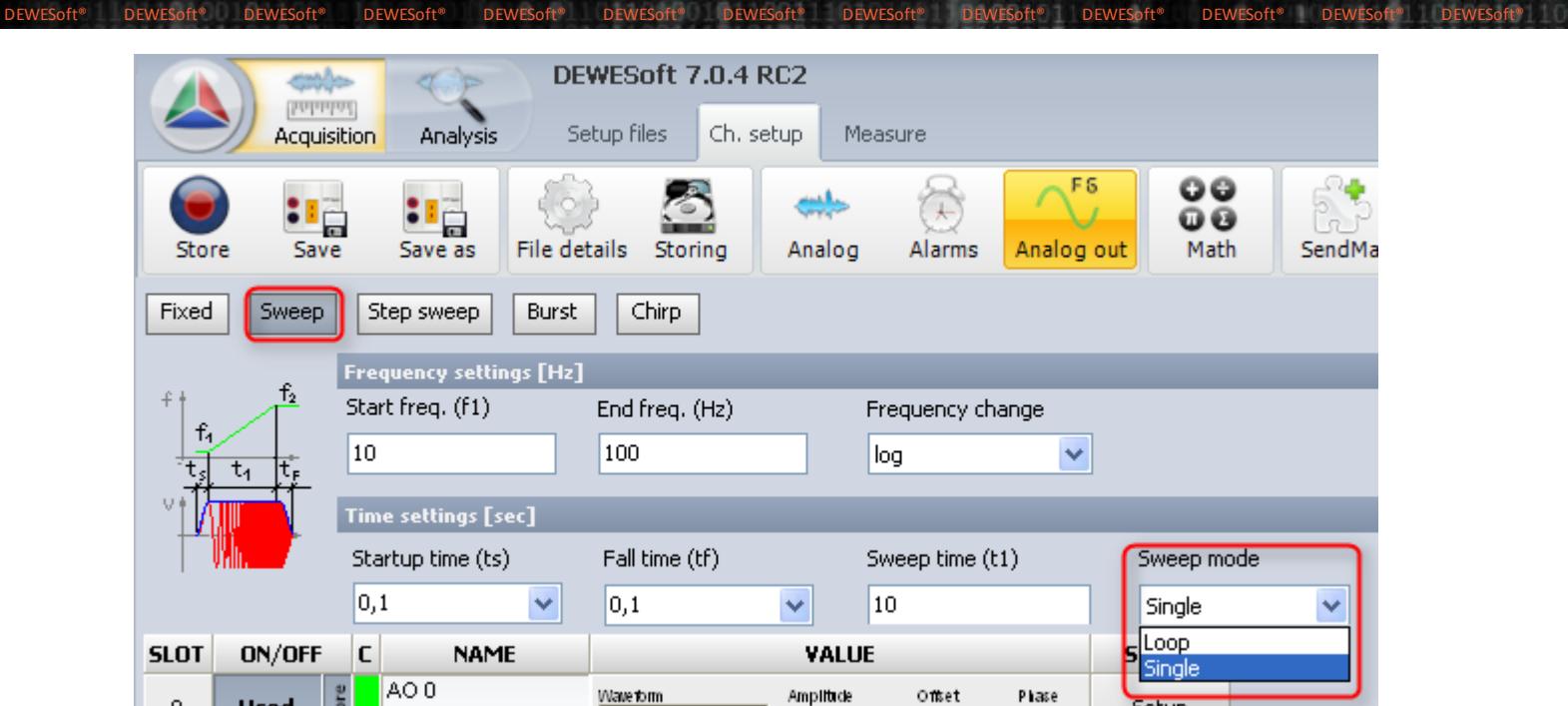
Interface: [IAOGroup](#) read/write

2.1.3.17 SweepMode

property SweepMode: AOSSweepMode

Defines if output signal is repeated in a *Loop* or only once (*Single*). This is only applicable for the operation mode *Sweep* (see [OperationMode](#)).

see also [AOSSweepMode](#)

Interface: [IAOGroup](#)

read/write

2.1.4 IAlarmCond

Defines an alarm condition.

see also: [IAlarms](#)

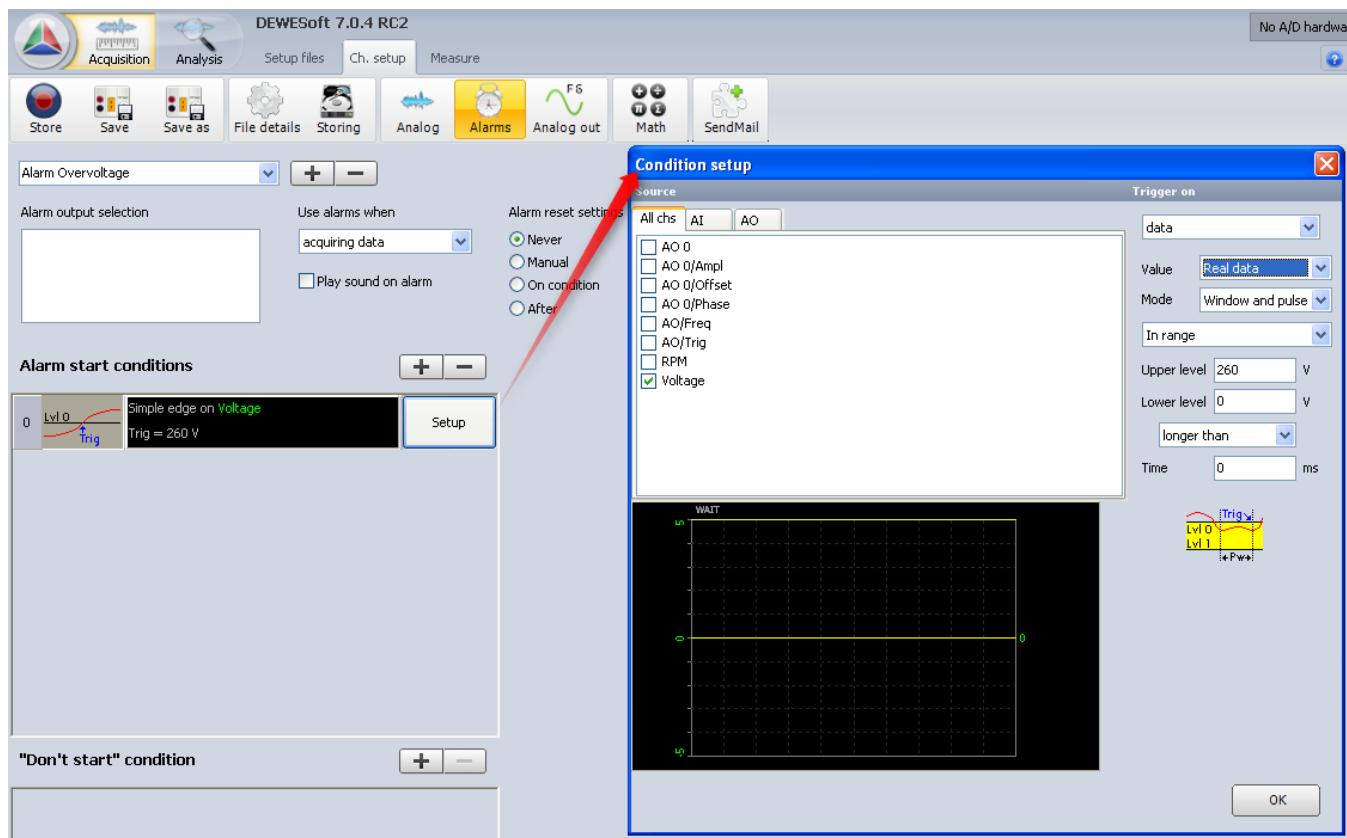


Illustration 138: Alarmcondition

2.1.4.1 Avail

property Avail: WordBool

for future use - not implemented yet

Interface: [IAlarmCond](#) read-only

2.1.4.2 CustomName

property CustomName: WideString

if CustomName is defined it will be the same as [Name](#).

Otherwise [Name](#) is automatically generated

see also: Name

Interface: [IAlarmCond](#) read/write

2.1.4.3 EndAlarm

```
procedure EndAlarm();
```

`EndAlarm` can be called in order to stop an alarm which has its `StopOption` set to *manual*.

Interface: [IAlarmCond](#)

2.1.4.4 Index

property Index: Integer

for future use - not implemented yet

2.1.4.5 Name

property Name: WideString

Interface: [WarmCond](#)

3.1.4.6 Status

property status: WordBool

`Status` is `TRUE` if an alarm condition is met and the alarm is currently still active.

All active alarms will be in this list: [IAlarms ActiveItem](#) (see also [IAlarms ActiveCount](#))

Interface: [IAlarmCond](#)

2.1.4.7 StopOption

property StopOption: Integer

StopOption defines how and whether an alarm condition is stopped. This can be one of the following:

- 0...never
- 1...manual
- 2...on stop condition
- 3...on time (see [StopTime](#))

see also [EndAlarm](#)

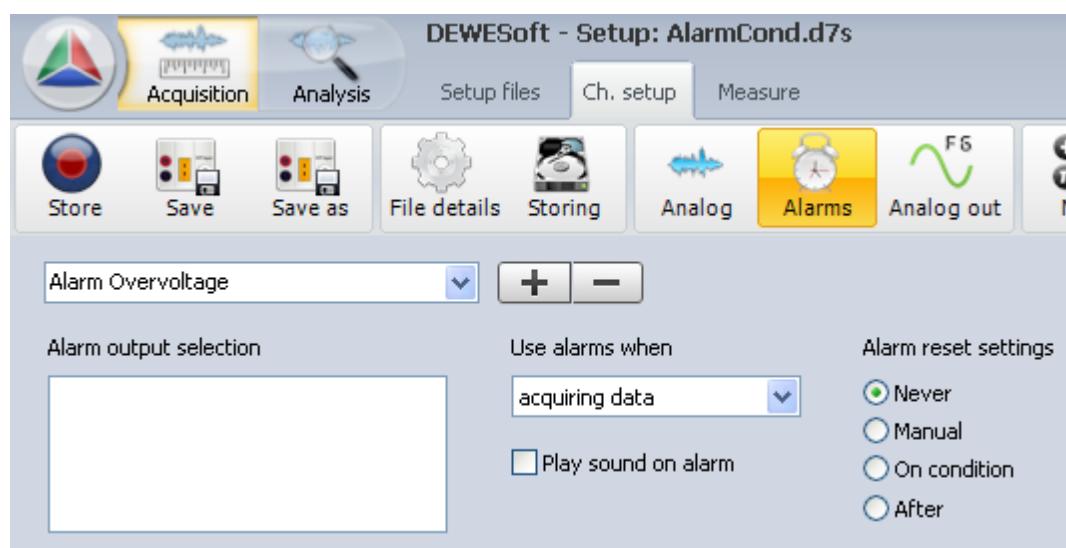


Illustration 139: Alarm condition stop option

Interface: [IAlamCond](#) read/write

2.1.4.8 StopTime

property StopTime: Single

StopTime is the time after which an alarm is stopped if the [StopOption](#) is set to 3 (on time). The unit of StopTime is seconds.

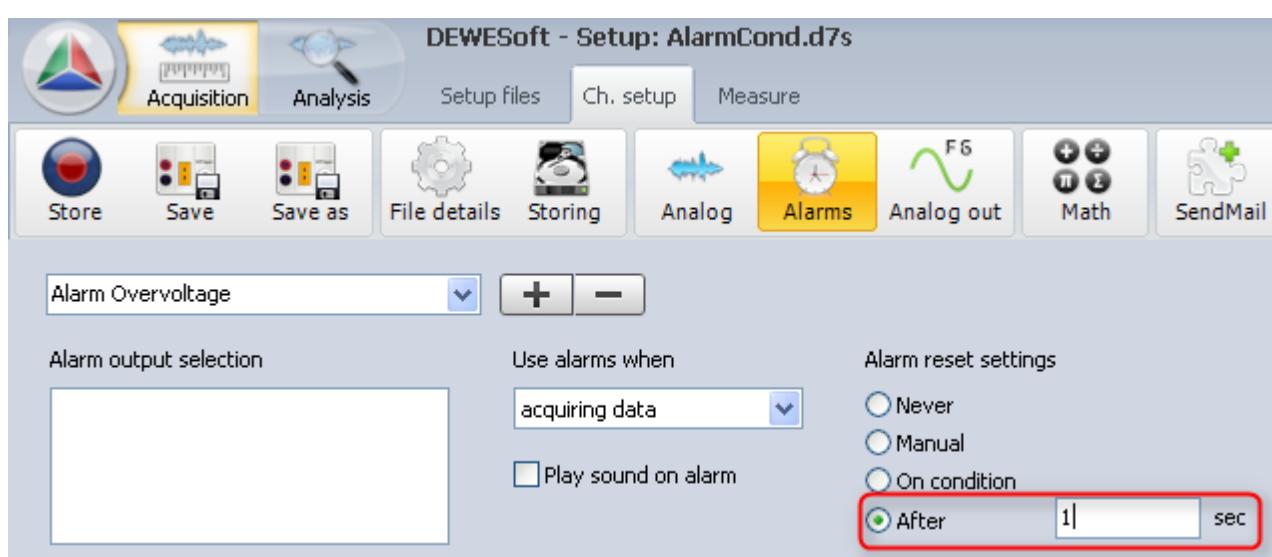


Illustration 140: Stop time

Interface: [IAlarmCond](#)  read/write

2.1.4.9 StopTrigger

property StopTrigger: ITrig

`StopTrigger` is the combination of trigger conditions deactivating an alarm. Only applicable if [StopOption](#) is set to 2 (*on stop condition*).

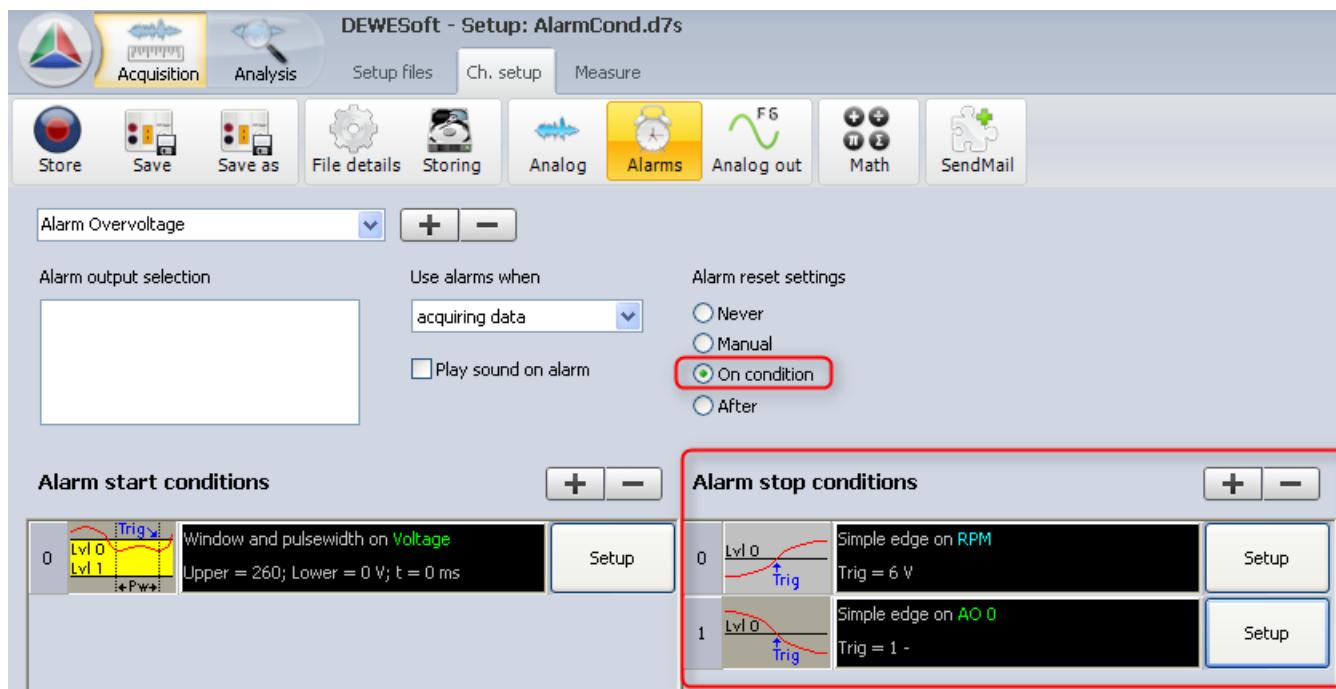


Illustration 141: Stop Trigger

Interface: [IAccount](#) read-only

2.1.4.10 Trigger

property Trigger: ITrigger

Trigger is the combination of trigger conditions activating an alarm.

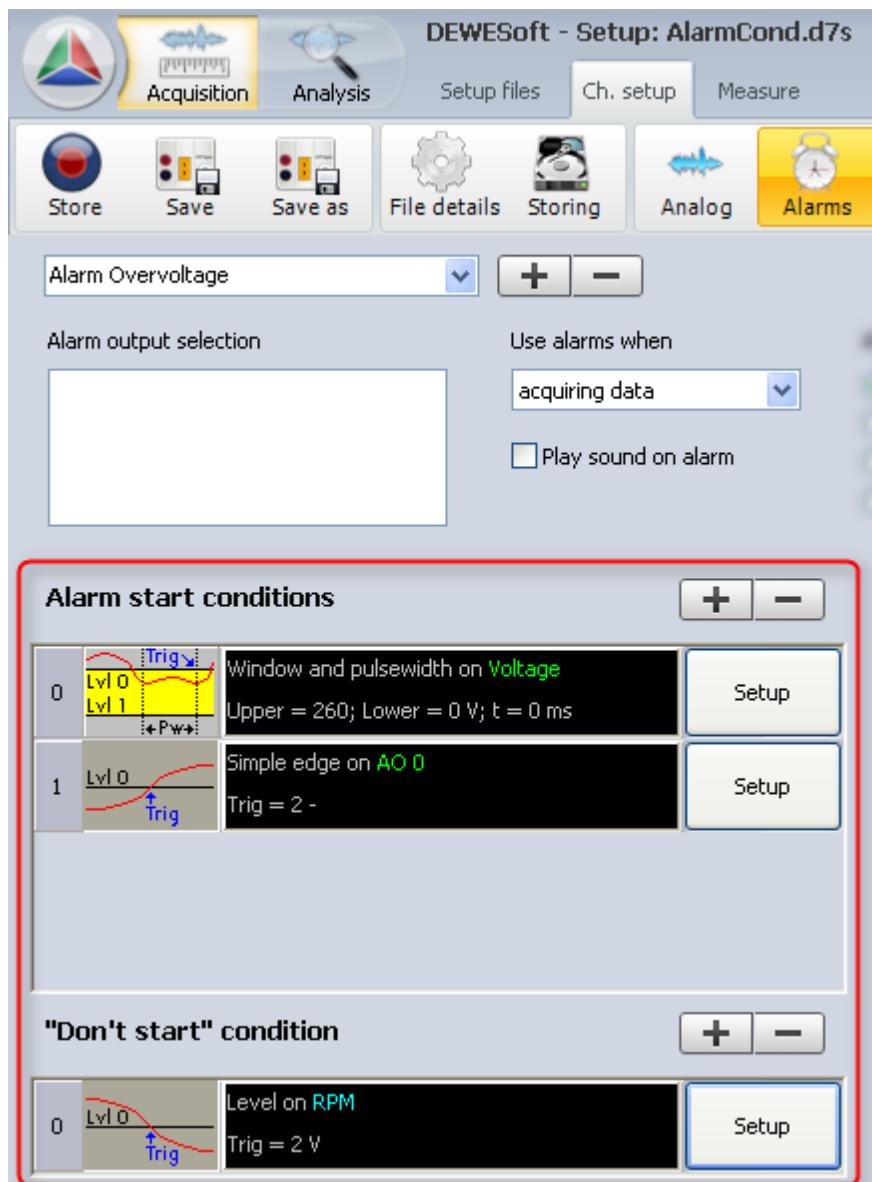


Illustration 142: Alarm Trigger

Interface: [IAlarmCond](#)



2.1.5 |Alarms

This interface gives you access to the list of all alarms (see [AlarmCond](#))

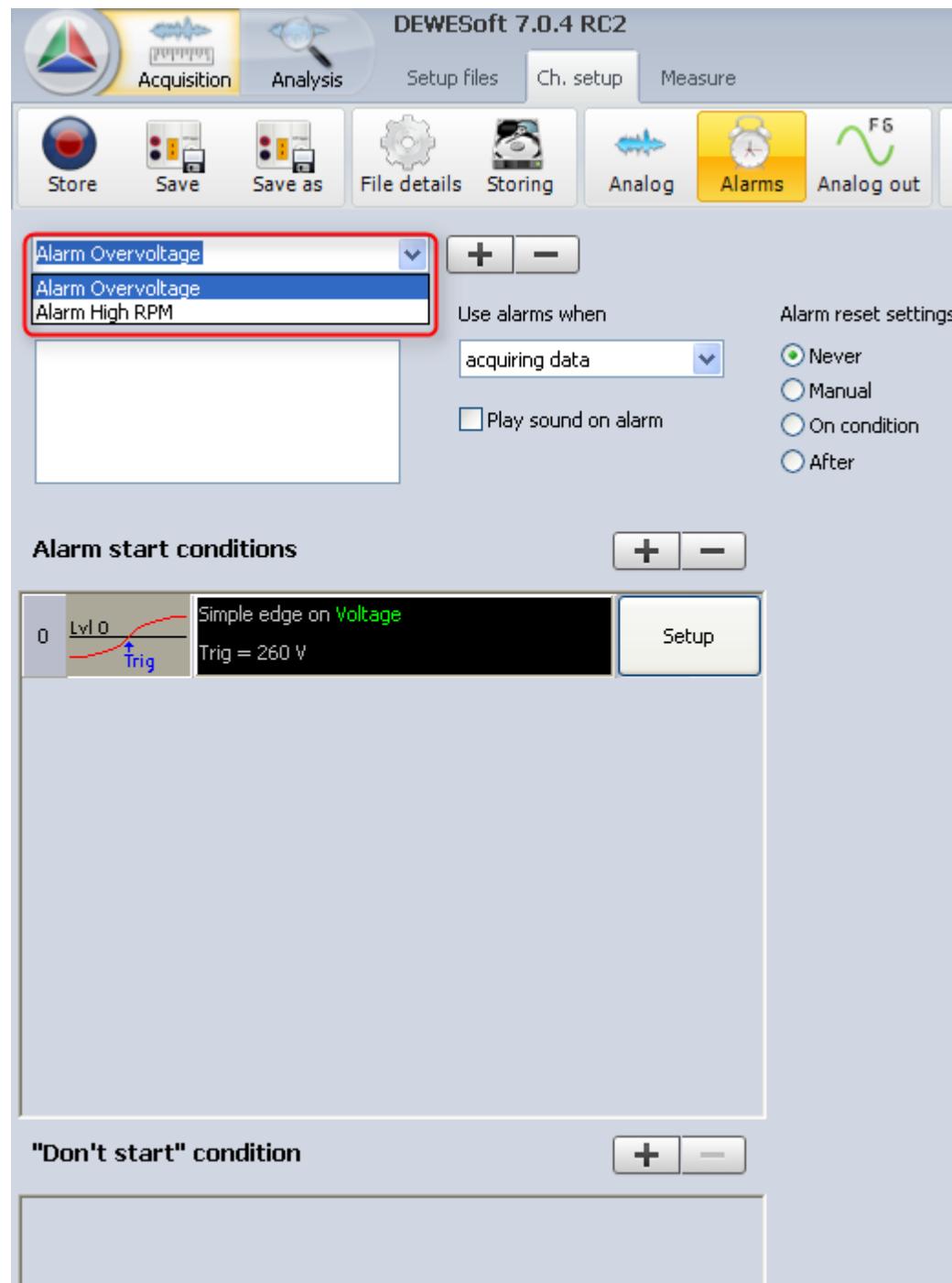


Illustration 143: Alarms

see also: [IApp.Altarms](#)

2.1.5.1 ActiveCount

property ActiveCount: Integer

ActiveCount is the number of alarm conditions ([IAlarmCond](#)) which are currently active ([IAlarmCond.Status](#) = TRUE - that means that the alarm condition has been met).

compare to: [Count](#)

see also: [IAlarmCond.Status](#)

Interface: [IAlarms](#)  read-only

2.1.5.2 ActiveItem

property ActiveItem[I: Integer]: [IAlarmCond](#)

ActiveItem[I] is the active alarm condition (see [IAlarmCond](#)) at the index I. I is in the range of 0...[ActiveCount](#)-1

The ActiveItem list is consistent only during measurement.

compare to: [Item](#)

see also: [IAlarmCond.Status](#)

Interface: [IAlarms](#)  read-only

2.1.5.3 Add

function Add(): [IAlarmCond](#);

will add a new alarm condition to the [ActiveItem](#) list and return this new condition.

Interface: [IAlarms](#)

Classifier	Name	Type	Description
-	RESULT	IAlarmCond	the new alarm condition that has just been added

2.1.5.4 Count

property Count: Integer

Count is the number of all alarm conditions which have been configured.

Compare to: [ActiveCount](#).

Interface: [IAlarms](#)  read-only

2.1.5.5 Item

property Item[I: Integer]: IAlarCond

Item[I] is the alarm condition at index I. I is in the range of 0...Count-1.

Compare to: [ActiveItem](#)

Interface: [IAlarms](#)  read-only

2.1.5.6 Remove

```
procedure Remove(Index: Integer);
```

will remove the alarm condition at the given Index. The index refers to the [Item](#) list.

Interface: [IAlarms](#)

Classifier	Name	Type	Description
	Index	Integer	The index of the alarm condition that will be removed. The index refers to the Item list.

2.1.6 IImplChain

provides access to all amplifiers ([IAmplifier](#)) of the channel

see also: [IAmIChainList](#), example in [IAmIInterfaces](#)

2.1.6.1 Count

property Count: Integer

the number of [IAmplifier](#) objects in this list

see also: example in [IAmIInterfaces](#)

Interface: [IAmpChain](#)

2.1.6.2 IOControl

```
function IOControl  
(IOCode: Integer; InParam: OleVariant; var OutParam: OleVariant): Integer;
```

to read/write properties of the amplifier interface: see [IO Codes for Chain properties](#) for valid IO codes and parameters

Interface: [IAmplChain](#), example in [IAmplInterfaces](#)

Classifier	Name	Type	Description
	IOCode	Integer	unique identification of the desired parameter or action: see IO Codes for Chain properties
	InParam	OleVariant	input parameter - type and meaning are dependant on the IO Code
var	OutParam	OleVariant	output parameter - type and meaning are dependant on the IO Code
-	RESULT	Integer	< 0 means that an error has occurred: see IOCodes

2.1.6.3 Item

```
property Item[Index: Integer]: IAmplifier
```

the [IAmplifier](#) object at the given Index.

see also: example in [IAmplInterfaces](#)

Interface: [IAmplChain](#)  read-only

2.1.7 IAmplChainList

provides access to the amplifier chain list

see also: [IAmplInterface.ChainList](#) example in [IAmplInterfaces](#)

2.1.7.1 Count

```
property Count: Integer
```

the number of [IAmplChain](#) objects in this list

see also: example in [IAmplInterfaces](#)

Interface: [IAmplChainList](#)  read-only

2.1.7.2 Item

property Item[Index: Integer]: IAmplChain

provides access to the the [IAmplChain](#) object at the specified Index.

see also: [IAmIChain](#), example in [IAmIInterfaces](#)

Interface: [IAmpICChainList](#)  read-only

2.1.8 IAmplInterface

provides access to the amplifier interface

see also: example in [IAmP1Interfaces](#)

2.1.8.1 ChainList

property ChainList: IAmplChainList

provides access to the amplifier chain list

see also: [IAmpICChainList](#), example in [IA](#)

3.1.8.3 IOControl

```
function IOControl  
(IOCode: Integer; InParam: OleVariant; var Param3: OleVariant): Integer;
```

to read/write properties of the amplifier interface; see [IOCodes for IAmpInterface](#) for valid IO codes and parameters.

Interface: [IAmperInterface](#) example in [IAmperInterfaces](#)

Classifier	Name	Type	Description
	IOCode	Integer	unique identification of the desired parameter or action: see IOCodes for IAmPIInterface
	InParam	OleVariant	input parameter - type and meaning are dependant on the IO Code
var	Param3	OleVariant	output parameter - type and meaning are dependant on the IO Code
-	RESULT	Integer	< 0 means that an error has occurred: see IOCodes

2.1.8.3 SubInterface

property SubInterface: [IAmplInterface](#)

not implemented yet

Interface: [IAmplInterface](#)  read-only

2.1.9 IAmplInterfaces

The amplifier interfaces provide access to the amplifiers of analog input channels. Each analog channel can have several chained amplifiers.

see: [IApp.AmplInterfaces](#)

The amplifier interfaces are best explained with an example. In this case we will use a DEWE-43 as analog device (with one MSI-BR-ACC adapter) plus 2 EPAD modules which are connected via COM port to the measurement PC.

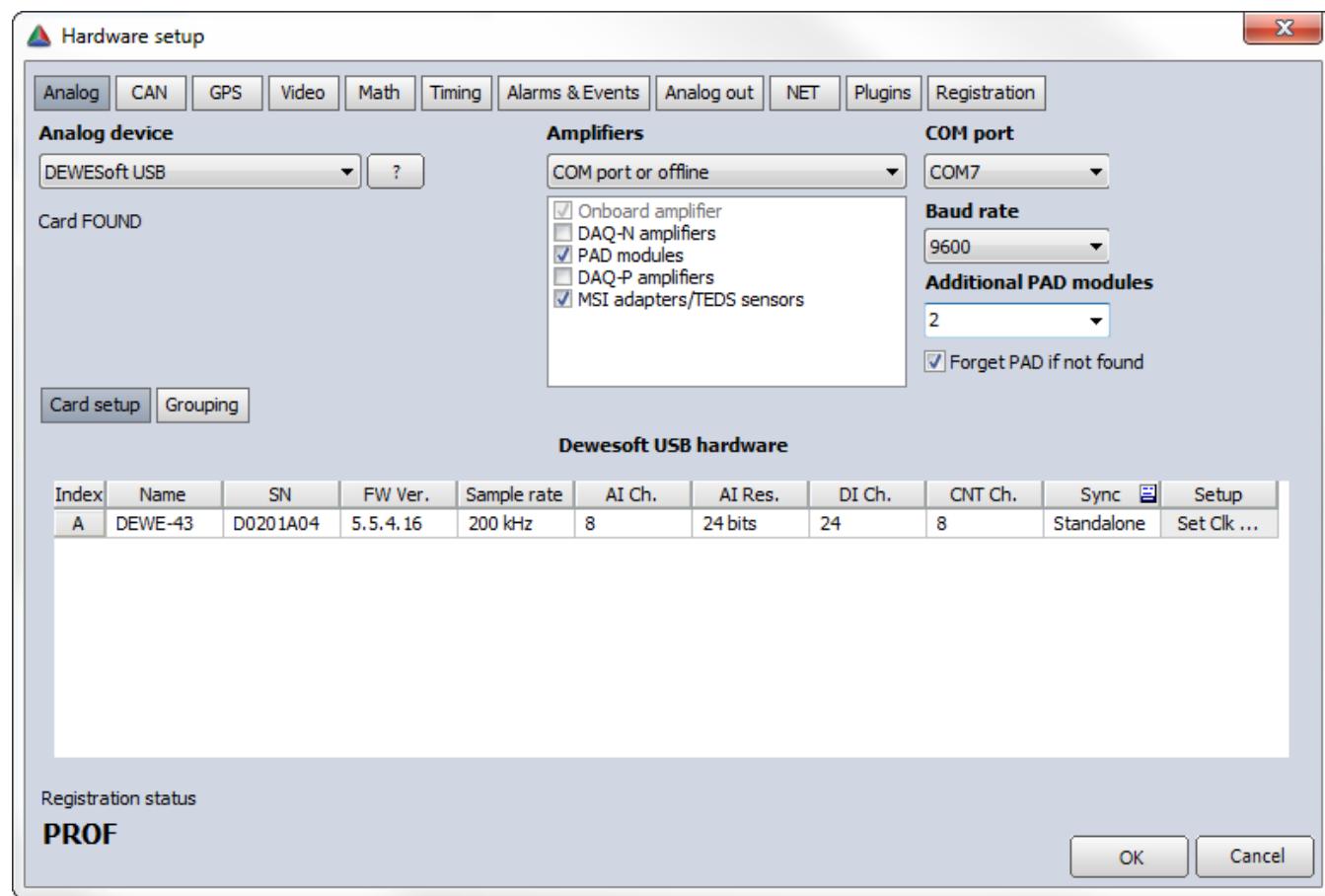


Illustration 144: Amplifier Example Hw Setup

The channel setup look like this:

SLOT	ON/OFF	C	NAME	AMPLIFIER (008)	PHYSICAL VALUES	CAL	SETUP
0	Used	Store	AI 0	MSI-BR-ACC (43-V)	SN: 372659 -0.0 V 10000 mV 1E4	Zero	Auto Set ch. 0
1	Used	Store	AI 1	43-V	SN: D0201A04 -17.4 mV/V 1000 mV/V; Exc 5V -2000 2000	Zero	Auto Set ch. 1
2	Used	Store	AI 2	43-V	SN: D0201A04 -0.091 V 10V; 50 kHz -10 10	Zero	Auto Set ch. 2
3	Used	Store	AI 3	43-V	SN: D0201A04 0.162 V 10V; 50 kHz -10 10	Zero	Auto Set ch. 3
4	Used	Store	AI 4	43-V	SN: D0201A04 -0.037 V 10V; 50 kHz -10 10	Zero	Auto Set ch. 4
5	Used	Store	AI 5	43-V	SN: D0201A04 0.067 V 10V; 50 kHz -10 10	Zero	Auto Set ch. 5
6	Unused	Store	AI 6	43-V	SN: D0201A04 -0.032 V 10V; 50 kHz -10 10	Zero	Auto Set ch. 6
7	Unused	Store	AI 7	43-V	SN: D0201A04 -0.034 V 10V; 50 kHz -10 10	Zero	Auto Set ch. 7
8	8 chnl			PAD-TH8-P	SN: 321317 28 1372 1372 1372 TC K: -270..1372 °C	NA	Set PAD 8
9	8 chnl			PAD-V8-P	SN: 317528 -0.1 0 0 -0.1 500 mV -0.1 -0.2 -0.1 0	NA	Set PAD 9

Illustration 145: Amplifier Example Channel Setup

1) the SLOT number of the analog channels

2) the first 8 channels belong to the DEWE-43

3) you can see that an MSI-BR-ACC adapter is connected to the first analog channel of the DEWE-43

4) the last 2 analog channels correspond to the 2 connected PAD modules

Navigation

you always start from [IAmplInterfaces](#) which gives you access to all amplifier interfaces. Currently DEWEsoft® supports only one amplifier interface, thus the [IAmplInterface](#) has only one member named: [MainInterface](#) which is of type [IAmplInterface](#).

The [IAmplInterface](#) has a function [IOControl](#) that let's you read/write properties of the amplifier interface (see also: [IOCodes](#) for [IAmplInterface](#)).

For example, a call to [IAmplInterface.IOControl](#) with input parameter `intDaqMaxChannels`, will return 8 in our case (the number of analogue channels in the hardware setup: see (1) in the illustration *Amplifier Example Channel Setup*)

Moreover it has a member [ChainList](#) (of type [IAmplChainList](#)): each member of this list corresponds to a slot in the analog channel setup:

For example [IAmplInterfaces.MainInterface.ChainList.Item\[0\]](#) is the [IAmplChain](#) object which corresponds to our first analogue channel called `AI 0`. Each of these [IAmplChain](#) objects can have 1 or more amplifiers.

The number of [IAmplifier](#) objects in the [IAmplChain](#) is already determined by the hardware setup:

i.e. If we had connected a DEWE-43 only (without MSI adapters, etc. - no PAD modules, etc.) then there would be only one amplifier in the list - the DEWE-43 amplifier itself.

In our case, the analog channels of the DEWE-43 will have 3 [IAmplifier](#) objects in the [IAmplChain](#) list (it is always the same number and the same order as specified in the *Amplifiers* section of the *Analog* tab-sheet in the hardware setup), so:

[IApp.AmplInterfaces.MainInterface.ChainList.Item](#) [0] . [Item](#) [0] is the [IAmplifier](#) of the DEWE-43 amplifier

[IApp.AmplInterfaces.MainInterface.ChainList.Item](#) [0] . [Item](#) [1] is the [IAmplifier](#) of the PAD modules amplifier (which is not used in our case, because the addresses of the PAD modules are set to be after the analog channels of the DEWE-43 at slots 8 and 9)

[IApp.AmplInterfaces.MainInterface.ChainList.Item](#) [0] . [Item](#) [2] is the [IAmplifier](#) of the MSI-BR-ACC adapter

The channels that correspond to the PAD modules are kind of special. they always have 2 amplifiers: the first one is for the analog input which is not used and the second one is for the PAD module itself.

Example for channel AI 0

Channel *AI 0* is the first analog channel of the DEWE-43 and we have connected an MSI-BR-ACC adapter to it: since we have also activated the PAD modules, we will have 3 amplifiers in [IAmplChain](#).

To get the module name of the DEWE-43 amplifier, we must call [IApp.AmplInterfaces.MainInterface.ChainList.Item](#) [0] .

[Item](#) [0] . [IOControl](#) with the *IOCode* `amplGetModuleName`.: the result is 43-V.

To get the module name of the MSI-BR-ACC amplifier, we must call [IApp.AmplInterfaces.MainInterface.ChainList.Item](#) [0] .

[Item](#) [2] . [IOControl](#) with the *IOCode* `amplGetModuleName`.: the result is MSI-BR-ACC.

2.1.9.1 MainInterface

property MainInterface: [IAmplInterface](#)

provides access to the main amplifier interface

see also: [IAmplInterface](#), example in [IAmplInterfaces](#)

Interface: [IAmplInterfaces](#)  read-only

2.1.10 IAmplifier

provides access to an amplifier

see also: [IAmIChain](#), example in [IAmIInterfaces](#)

2.1.10.1 IOControl

```
function IOControl  
(IOCode: Integer; InParam: OleVariant; var OutParam: OleVariant): Integer;
```

to read/write properties of the amplifier interface: see [IO Codes for Amplifier properties](#), [IO Codes for Amplifier commands](#) for valid IO codes and parameters

Interface: [IAmplifier](#), example in [IAmplInterfaces](#)

Classifier	Name	Type	Description
	IOCode	Integer	unique identification of the desired parameter or action: IO Codes for Amplifier properties , IO Codes for Amplifier commands
	InParam	OleVariant	input parameter - type and meaning are dependant on the IO Code
var	OutParam	OleVariant	output parameter - type and meaning are dependant on the IO Code
-	RESULT	Integer	< 0 means that an error has occurred: see IOCodes

2.1.11 |App

this interface refers to the the DEWEsoft application. It is the entry point and most important interface.

2.1.11.1 AISetupScreen

property AISetupScreen: TAISetupScreen;

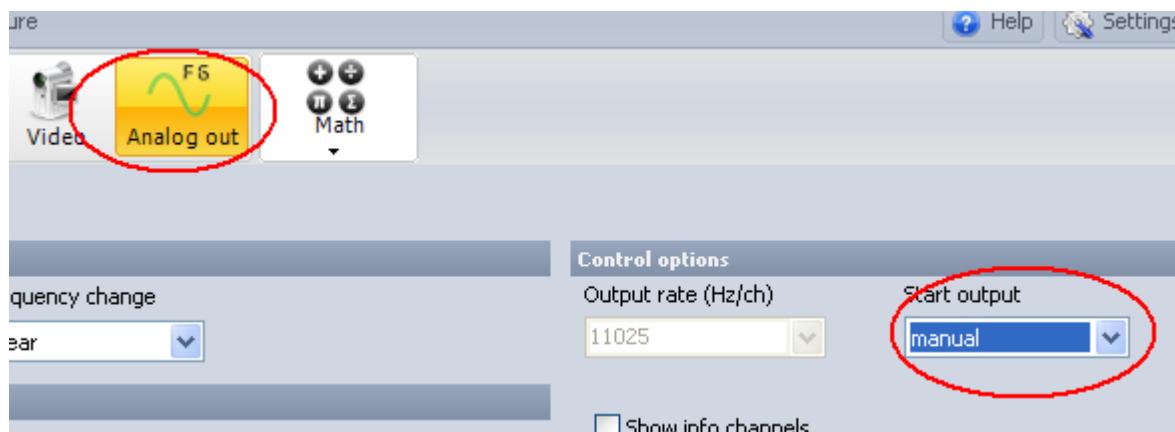
See [\AISetupScreen](#)

Interface: IApp

2.1.11.2 AOGetManualAvail

```
function AOGetManualAvail(): WordBool;
```

Whether the manual start/stop feature of analog output is available in the current setup.



see also: [AOSetManual](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
-	RESULT	WordBool	Whether the manual start/stop feature of analog output is available in the current setup.

2.1.11.3 AOGroup

```
property AOGroup: IAOGroup
```

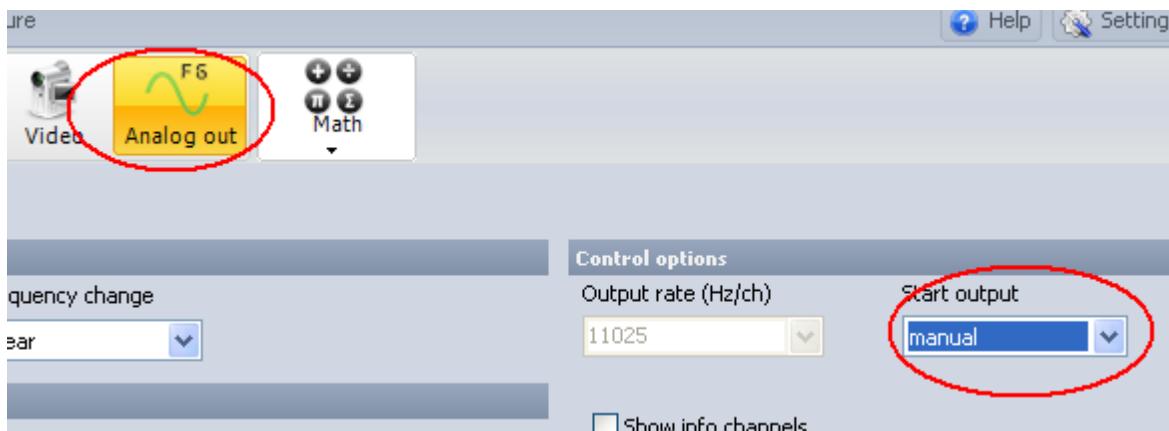
See [IAOGroup](#)

Interface: [IApp](#) read-only

2.1.11.4 AOSetManual

```
procedure AOSetManual();
```

If the function generator is set to *manual* output (see also [AOGetManualAvail](#)), calling this function will start the output:



This is the same as if the user would press the *Manual* button to start the function generator (shown below):



see also: [AISetupScreen](#)

Interface: IApp

2.1.11.5 Acquiring

property Acquiring: WordBook

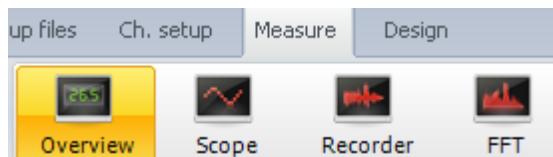
Whether DEWEsoft® is currently acquiring data (If this property is True, DEWEsoft® is not necessarily storing)

Interface: [!App](#)  read-only

2.1.11.6 ActiveScreen

property ActiveScreen: Integer

Returns the number of the active screen (aka. instrument):



The number can be one of the following:

The index of the first screen is 0.

e.g. (default setup)

0 refers to [Overview](#)

1 refers to [Scope](#)

2 refers to [Recorder](#)

3 refers to [FFT](#)

note that the user can completely customize the screens (aka. instruments). he could delete/insert/rename screens:
Thus only the number is important, not the display name.

see also: [GUI Navigation](#), [SetInstrument](#)

Interface: [IApp](#) read-only

2.1.11.7 ActualRunMode

property ActualRunMode: Integer

Returns the mode in which the application is actually running:

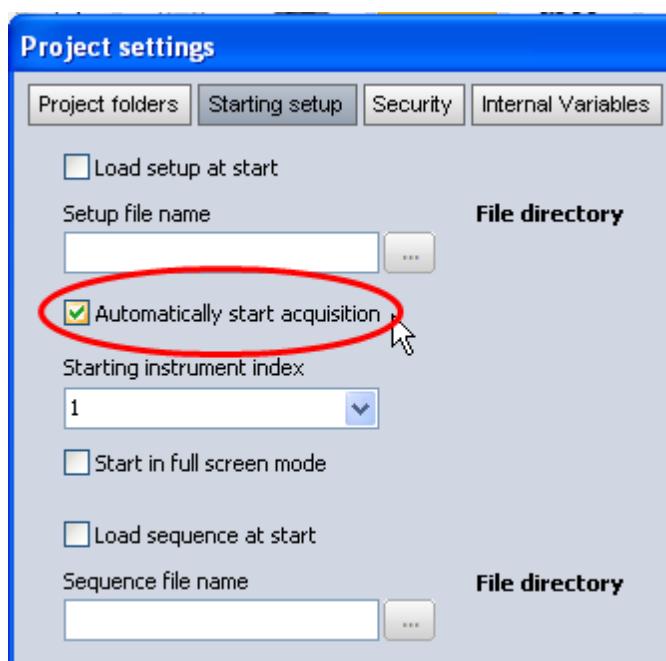
0...none (only during startup)

1...measure/acquisition ([Measure](#) mode)

2...analyse ([Analysis](#) mode)



At startup ActualRunMode will be 0 in most cases (except when *Automatically start acquisition* is checked in the [Starting setup](#) of the Project settings):



If the application is in [Measure](#) mode, `ActualRunMode` is 1.

If it is in [Analysis](#) mode, ActualRunMode will be 2.

The application can be set back to `ActualRunMode` of 0 by calling `Stop`.

Interface: IApp

2.1.11.8 Alarms

property Alarms: IAlarms

See [IAlarms](#)

Interface: [IApp](#) read-only

2.1.11.9 AlwaysEnableTrigger

```
property AlwaysEnableTrigger: WordBool
```

to also activate the trigger, even if store is not activated (e.g. used for database storing of Power module)

Interface: IApp

2.1.11.10 AmplInterfaces

property AmplInterfaces: [IAmplInterfaces](#)

see [IAmplInterfaces](#)

Interface: [IApp](#)  read-only

2.1.11.11 Analyze

procedure Analyze();

Switches Dewesoft to Analyze mode like pressing *Analysis* button on the screen:



see also [Measure](#), [ActualRunMode](#), [GUI Navigation](#)

Interface: [IApp](#)

2.1.11.12 AveragedCPB

property AveragedCPB: [IAveragedFFT](#)

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

now array channels are used for CPB

Interface: [IApp](#)  read-only

2.1.11.13 AveragedFFT

property AveragedFFT: [IAveragedFFT](#)

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

now array channels are used for FFT

Interface: [IApp](#)  read-only

2.1.11.14 CAN

property CAN: ICAN

See ICAN

Interface: [IApp](#) read-only

2.1.11.15 CalcScopeTrig

```
function CalcScopeTrig(): WordBool;
```

² See also the discussion of the relationship between the two in the previous section.

Interface: IApp

Classifier	Name	Type	Description
-	<i>RESULT</i>	WordBool	false, if no trigger is found true, if a trigger is found - then the trigger data will be loaded into the buffer

2.1.11.16 ChangeComPort

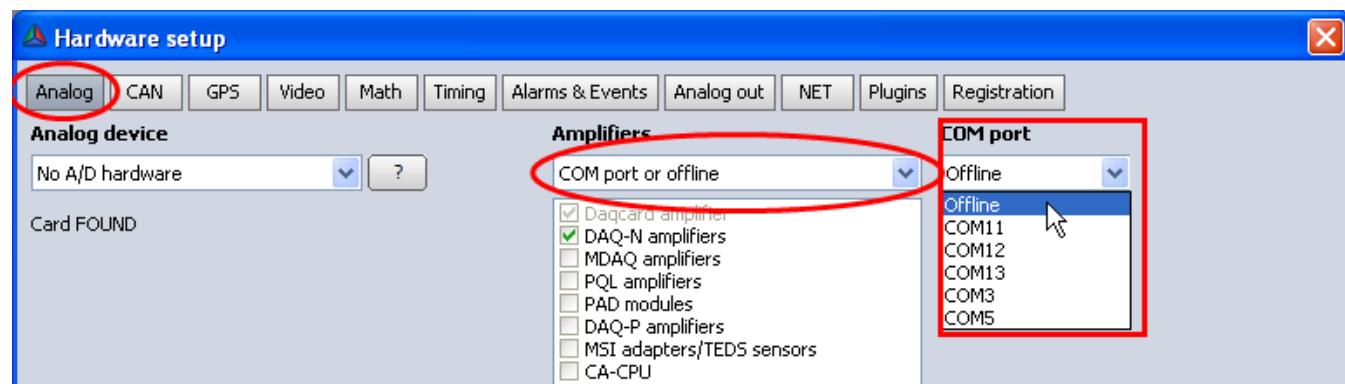
```
procedure ChangeComPort(ComPort: Integer);
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

now we have the amplifier interface is used instead - see [IAmplInterfaces](#)

Changes the COM port which is used for communication to the analog devices.

This is not possible in [Analysis](#) mode and only if the measurement is stopped.



Interface: [IApp](#)

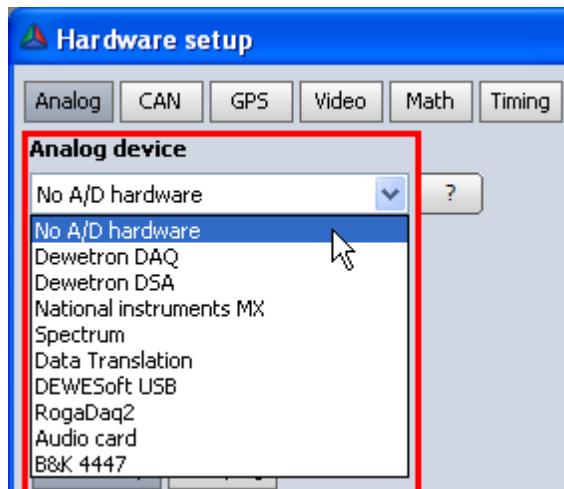
DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	ComPort	Integer									

2.1.11.17 ChangeDaqType

```
procedure ChangeDaqType(DaqType: Integer);
```

Changes the device type for the analog acquisition.

This is the same as changing the *Analog device* in the Hardware setup:



Interface: [IApp](#)

Classifier	Name	Type	Description
	DaqType	Integer	<p>the Id of the analog device:</p> <p>0...No A/D 1...SpectrumDewetron DAQ 2...Dewetron DSA 3...Spectrum 4...National Instruments 5...National Instruments MX 6...National Instruments DSA 7...Data Translation 8...Microstar DAP (</p> <p>DAQ classes:</p> <p>0 - test OK 5 spectrumM2i - ?? 2 TDeweDsa OK 4 TDTDaq, ?? 1 TNiDaqMX, ?? 3 TRtDaq3, ?? 6 TDWUSBDaq ??</p> <p>Global constants!? --> this is correct</p> <p>daqTypeSpectrumM2i = 1; daqTypeDeweDSA = 2; daqTypeDT = 3; daqTypeNiDaqMX = 4; daqTypeDeweDAQ = 5; daqTypeDeweUSB = 6;</p>

2.1.11.18 ConfigMode

property ConfigMode: WordBool

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Interface: [IApp](#) read-only

2.1.11.19 Daq

property Daq: IDaq

see IDaq

Interface: [IApp](#) read-only

2.1.11.20 DagGroup

property DaqGroup: IDaqGroup

see [IDaqGroup](#)

Interface: IApp

2.1.11.21 Data

property Data: IData

see [IData](#)

Interface: IApp read-only

2.1.11.22 DataLost

property DataLost: WordBool

is true when loss of data has occurred during measurement.

Interface: [IApp](#) read-only

2.1.11.23 DisableKeyboardShortcuts

```
property DisableKeyboardShortcuts: WordBool
```

can be used to disable keyboard shortcuts in DEWEsoft®.

e.g. *Control+F* switches to full screen mode

Interface: [IApp](#)  read/write

2.1.11.24 DisableStoring

```
property DisableStoring: WordBool
```

To hide the *Store* button.:.



see also: [StartStoring](#), [PauseStoring](#), [ResumeStoring](#), [Stop](#)

Interface: [IApp](#)  read/write

2.1.11.25 Enabled

```
property Enabled: WordBool
```

Whether DEWEsoft® is enabled or disabled.

If Enabled is set to False, it is not possible to manipulate DEWEsoft's controls by the mouse.

Interface: [IApp](#)  read/write

2.1.11.26 EventList

```
property EventList: IEventList
```

see: [IEventList](#)

Interface: [IApp](#)  read-only

2.1.11.27 ExecuteModulesFunction

```
procedure ExecuteModulesFunction(Func: ModulesFunction  
; Group: Byte; Value: Integer);
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Can be used to execute a specific function of a module when DEWEsoft is in [Measure](#) mode.

see also [Modules Function](#)

Interface: IApp

Classifier	Name	Type	Description
	Func	ModulesFunction	see table below for a detailed explanation of the available module functions
	Group	Byte	the meaning is dependant on the ModulesFunction
	Value	Integer	the meaning is dependant on the ModulesFunction

Enumerations for the `ModulesFunction` parameter:

Dec	Hex	Name	Description
0	0x00	mfBridgeZeroAll	available when not measuring
1	0x01	mfAmplZeroAll	available when not measuring
2	0x02	mfShortOnOffAll	available during measuring
3	0x03	mfShuntOnOffAll	available during measuring
4	0x04	mfShuntCalCheckAll	available when not measuring
5	0x05	mfChargeBZeroAll	available during measuring
6	0x06	mfFreqAFindTriggerAll	available when not measuring
7	0x07	mfDAQP_MDAQ_SetPowerOnDefaultAll	available when not measuring
8	0x08	mfDAQP_ResetPowerOnDefaultAll	available when not measuring
9	0x09	mfMDAQ_CalFileClearAll	available when not measuring
10	0x0A	mfBridgeZeroByGroupID	available when not measuring Group:
11	0x0B	mfAmplZeroByGroupID	available when not measuring
12	0x0C	mfShuntCalCheckByGroupID	available when not measuring Group:
13	0x0D	mfSetHighestRangeAll	available when not measuring
14	0x0E	mfSetBestRangeAll	available when not measuring
15	0x0F	mfSetHighestFilterAll	available when not measuring
16	0x10	mfSetFilter40PerOfSRAll	available when not measuring
17	0x11	mfSetFilterCustomAll	available when not measuring Value:
18	0x12	mfSetHighestRangeByGroupID	available when not measuring Group:
19	0x13	mfSetBestRangeByGroupID	available when not measuring Group:
20	0x14	mfSetHighestFilterByGroupID	available when not measuring Group:
21	0x15	mfSetFilter40PerOfSRByGroupID	available when not measuring Group:
22	0x16	mfSetFilterCustomByGroupID	available when not measuring Group: Value:
23	0x17	mfRescanDAQAt	available when not measuring Value:
24	0x18	mfRescanMDAQAt	available when not measuring Value:
25	0x19	mfRescanPADAt	available when not measuring Value:
26	0x1A	mfRescanDAQPMDAQAll	available when not measuring
27	0x1B	mfRescanPADAll	available when not measuring
28	0x1C	mfRescanMSIAAt	available when not measuring Value:
29	0x1D	mfRescanMSIAll	available when not measuring
30	0x1E	mfRescanDLRAll	available when not measuring
31	0x1F	mfResetFirstScan	available when not measuring

2.1.11.28 ExportData

```
procedure ExportData  
  (ExportType: Integer; TimeAxis: Integer; const FileName: WideString);
```

This procedure can be used to export DEWEsoft measurement data to other data formats (e.g. to Matlab, plain text, ...). It is a simpler version of [ExportDataEx](#).

For detailed description please see: [ExportDataEx](#).

Interface: IApp

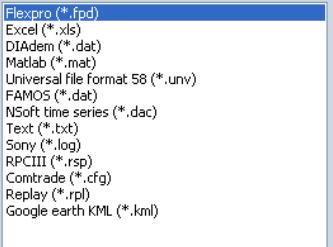
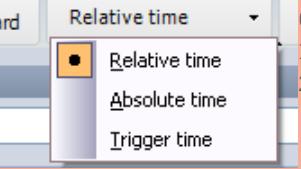
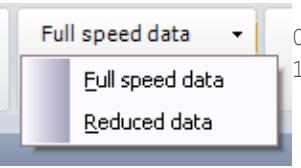
2.1.11.29 ExportDataEx

```
procedure ExportDataEx  
(ExportType: Integer; TimeAxis: Integer; ExportDataType: Integer; ExportOptions: In-  
teger; const FileName: WideString);
```

This procedure can be used to export DEWEsoft® measurement data to other data formats (e.g. to Matlab, plain text, ...). Please note that the file must be open in Analysis mode to call this action.

see also: [ExportData](#)

Interface: IApp

Class Identifier	Name	Type	Description
	ExportType	Integer	<p>Export type defines the type of the export to be used. The list of exports is enumerated in the same way as the drop down list in the export is shown.</p>  <p>0...Flexpro (*.fpd) 1...Excel (.xls) 2...DIAdem (*.dat) 3...Matlab (*.mat) 4...Universal File Format 58 (*.unv) 5...FAMOS (*.dat) 6...NSoft time series (*.dac) 7...Text (*.txt) 8...Sony (*.log) 9...RPCIII (*.rsp) 10...Comptade (*.cfg) 11...CAN Messages (*.csv) only available if there is CAN data stored in the measurement file 12...I-File</p> <p>Custom exports must be enumerated with negative values starting with -2. The custom exports depends on the installed exported (.exp) files. In our case from the screenshot above it would be: -2 ... Replay -3 ... Google earth KML</p>
	TimeAxis	Integer	<p>Time axis defines the how the time will be exported: The property corresponds to this selection:</p>  <p>0...relative 1...absolute 2...from trigger</p>
	ExportDataType	Integer	<p>Defines which data is to be exported: The property corresponds to this selection:</p>  <p>0..full speed 1..reduced (with speed defined in the reduced rate)</p>
	ExportOptions	Integer	<p>Defines which data will be exported at reduced rate. The property is important only if reduced data is exported. The property corresponds to this selection: The value is a set from: 1 ...min 2 ...max 4 ...average 8 ...rms</p> <p>So if we want to export min and max, we need to enter: 1+2=3. If we want to export min and RMS, we need to enter 1+8=9.</p> 
const	FileName	Widestring	<p>Specifies the path and name of the target-file. If the file extension is not defined, the default file extension will be taken. If the file path is not defined (just the file name is entered), then the default export folder will be taken. If not even a file name is defined (field is blank), then the loaded Dewesoft file name will be taken.</p>

2.1.11.30 FileNameSettings

property FileNameSettings: IFileNameSettings

See [IFileNameSettings](#)

Interface: [!App](#) read-only

2.1.11.31 FirstScanDonePercent

```
function FirstScanDonePercent(): Single;
```

returns the percentage of the first scan for amplifiers.

when a setup is loaded, the first scan of amplifiers will be done. During this time DEWEsoft® will show a progress bar at the top:

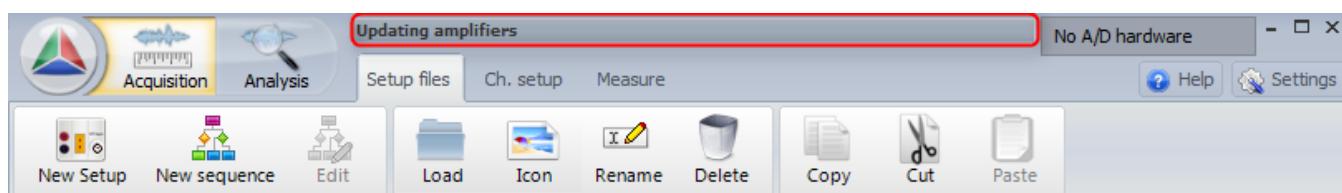


Illustration 146: First scan progress bar

Interface: [IApp](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	Single	the percentage of the first scan

2.1.11.32 FixedExternalClock

property FixedExternalClock: WordBool

FALSE: variable external clock: related to revolutions per second

TRUE: fixed variable clock: e.g. from a function generator

Interface: [IApp](#) read/write

2.1.11.33 GetInterfaceVersion

```
procedure GetInterfaceVersion  
  (var Major: Integer; var Minor: Integer; var Revision: Integer);
```

returns information about the DCOM interface

see also: [GetDewesoftVersion](#)

Classifier	Name	Type	Description
var	Major	Integer	This is the major version number of DEWEsoft® (same as GetDewesoftVersion parameter Super) e.g. for version "DEWEsoft X1 SP3 b223" the minor version umber is 7. Note: the X in DEWEsoft X is not the roman number 10 - it is actually a DEWEsoft 7 version.
var	Minor	Integer	Minor version number (same as see GetDewesoftVersion parameter Major) e.g. for version "DEWEsoft X1 SP3 b223" the minor version umber is 1.
var	Revision	Integer	this is the revision of the DCOM interface (version of the Type Library)

The Revision may be important when you want your code to be compatible with older DEWEsoft® versions.

e.g. The function [GetDewesoftVersion](#) has been added in Revision 79. If your DCOM plugin or application is run against a DEWEsoft® version lower than this Revision, it means that the function had not existed yet. And if you call the function, you will get an error. To avoid this, you can check the Revision:

```
DeweApp.GetInterfaceVersion(Super, Major, Revision);  
if Revision >= 79 then  
begin  
  DeweApp.GetDewesoftVersion(Super, Major, Minor, Build, State);  
  ...
```

2.1.11.34 GetDewesoftVersion

```
procedure GetDewesoftVersion(var Super: Integer; var Major: Integer; var Minor: Integer; var Build: Integer; var State: Shortint);
```

To find out the DEWEsoft® version number.

see also: [GetInterfaceVersion](#)

Example: "DEWEsoft X1 SP3 b223"

Interface: [IApp](#)

Classifier	Name	Type	Description
var	Super	Integer	Super version number. e.g. for version "DEWESoft X1 SP3 b223" the minor version umber is 7. Note: the X in DEWESoft X is not the roman number 10 - it is actually a DEWESoft 7 version.
var	Major	Integer	Major version number. e.g. for version "DEWESoft X1 SP3 b223" the minor version umber is 1.
var	Minor	Integer	The minor version number is the number of the Service Pack e.g. for version "DEWESoft X1 SP3 b223" the minor version umber is 3.
var	Build	Integer	The build number e.g. for version "DEWESoft X1 SP3 b223" the minor version umber is 223.
var	State	ShortInt	The state of the application: 0...Release version 1...Alpha version 2...Beta version 3...Release Candiate version

2.1.11.35 GetSpecDir

```
function GetSpecDir(DirType: SpecDirectory): WideString;
```

can be used to get the directories currently used by DEWESoft® (e.g the directory where the log files, data files etc. are stored)

see also: [SpecDirectory](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
	DirType	SpecDirectory	specifies which directory should be returned: see SpecDirectory for details
-	RESULT	WideString	the full path to the requested directory: e.g. sdSystemDir may return: D:\DEWESoft7\System\V7_0

2.1.11.36 GlobalHeader

```
property GlobalHeader: IGlobalHeader
```

Returns the global data header of the current DEWESoft® project. See [IGlobalHeader](#) for details.

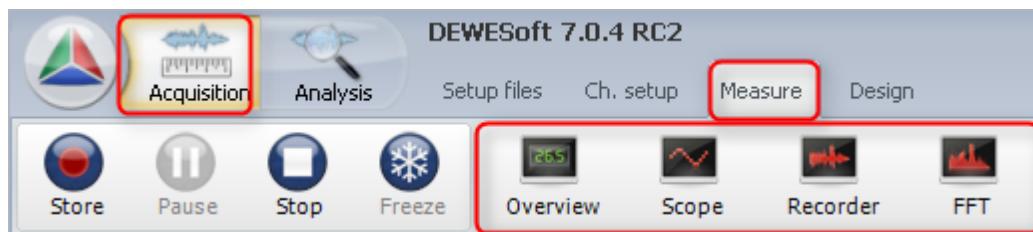
Interface: [IApp](#)  read-only

2.1.11.37 GoToInstruments

```
procedure GoToInstruments();
```

When DEWEsoft® is in *Acquisition mode*, you can call this function to go to *Measure mode*.

In the image below you can see a screenshot of DEWEsoft® after calling `GoToInstruments()`. You can see that we are now in *Measure mode* and you can see the default instruments (*Overview*, *Scope*, *Recorder* and *FFT*)



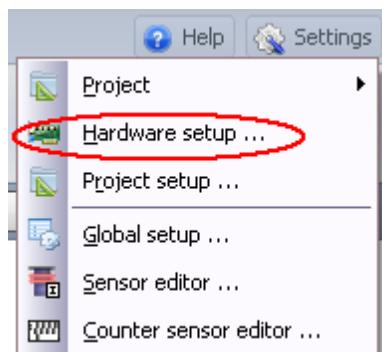
see also: [Measure Mode](#), [SetMainToolBar](#), [SetInstrument](#), [SetScreenIndex](#)

Interface: [IApp](#)

2.1.11.38 HardwareSetup

```
procedure HardwareSetup(Plugins: WordBool);
```

Will the dialog window of the hardware setup, like clicking [System - Harware setup](#)



see also: [UpdateHardwareSetup](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
	Plugins	WordBool	When this parameter is set to true, the Windows registry will be searched for plug-ins and all the plug-ins will be reinitialized (same behaviour as on the startup of DEWEsoft®). Otherwise the Hardware setup will just be shown like clicking System - Harware setup

2.1.11.43 Init

```
procedure Init();
```

Init starts the initialization of DEWEsoft®.

This must be called by automation applications right at the start.

Note: It is possible to set the [IniFileDir](#) before calling this function.

Interface: [IApp](#)

2.1.11.44 InitScopeTrig

```
procedure InitScopeTrig(Start: T\_RecordPosition; Stop: T\_RecordPosition);
```

should be called after setting the scope's trigger condition.

Interface: [IApp](#)

Classifier	Name	Type	Description
	Start	T_RecordPosition	start position
	Stop	T_RecordPosition	end position

2.1.11.45 IsAcqRunning

```
property IsAcqRunning: WordBool
```

Is TRUE when the data acquisition is running. This can be in Ch. Setup, Measure and Design mode:



Interface: [IApp](#) read-only

2.1.11.46 IsSetupMode

property IsSetupMode: WordBool

`IsSetupMode` is *TRUE* if DEWEsoft® is currently in the channel setup mode, *FALSE* otherwise.

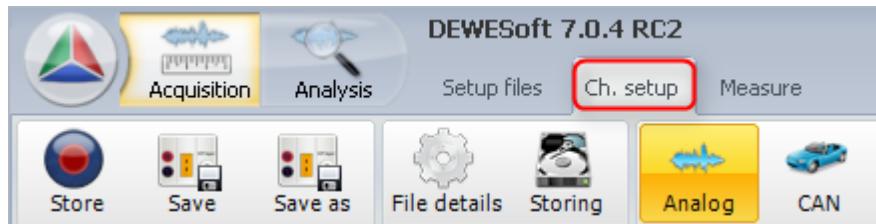


Illustration 148: Channel Setup

Interface: IApp  read-only

2.1.11.47 LastKey

property LastKey: Integer

the keycode of the last shortcut key that was pressed

Interface: [IApp](#) read-only

2.1.11.48 Left

property Left: Integer

Left is the distance between the left border of the DEWEsoft® application window and the left border of the screen expressed in pixels.

see also: [Top](#), [Height](#), [Width](#)

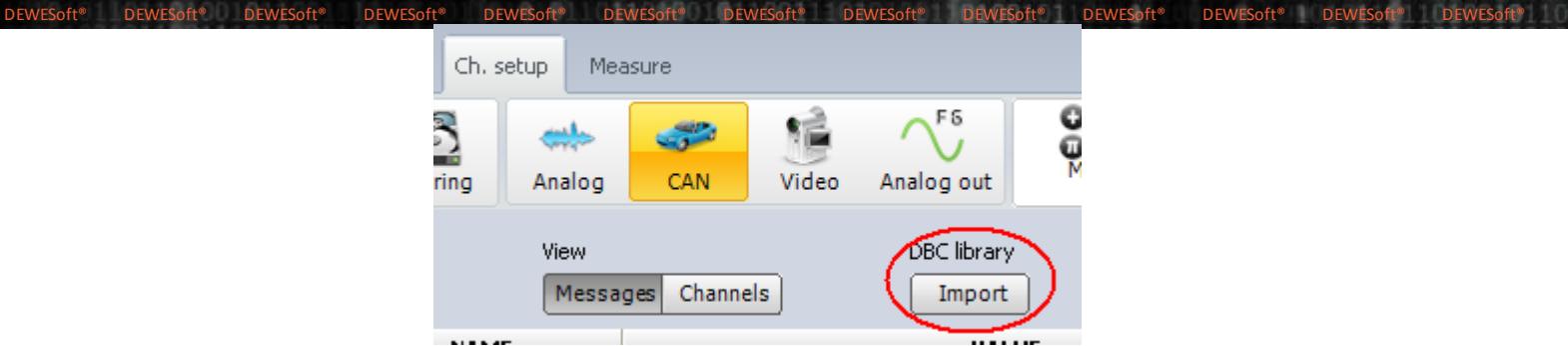
Interface: `IApp` read/write

2.1.11.49 LoadDBC

```
procedure LoadDBC(PortNo: Integer; const FileName: WideString);
```

This action will load the DEWEsoft®DBC file `FileName` for the CAN port specified by `PortNo` (first port is 0). It is the same action as this one found in CAN setup:

see also: [ICANPort](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
	PortNo	Integer	is the port-number to which the dbc-file is applied (first port is 0)
const	FileName	WideString	is the name of the dbc-file including its path. If the DBC-file resides in DEWEsoft's Setup-directory (see also SpecDirectory), the file-name without the whole path would be sufficient.

2.1.11.50 LoadDisplaySetup

```
procedure LoadDisplaySetup(const FileName: WideString);
```

LoadDisplaySetup allows loading a display setup as it can be done by the menu item [File - Load Display Setup](#):

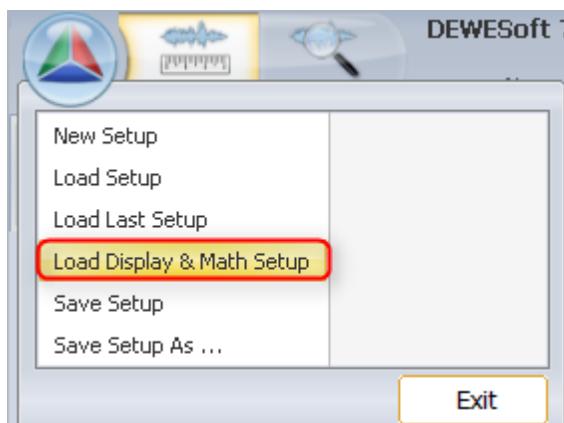


Illustration 149: Load Display Setup

Interface: [IApp](#)

Classifier	Name	Type	Description
const	FileName	WideString	is the file name including the path of the DEWEsoft® setup file which should be loaded.

2.1.11.51 LoadEngine

```
property LoadEngine: ILoadEngine
```

`LoadEngine` has to be used for handling data files acquired by DEWEsoft®. See [ILoadEngine](#) for detailed information.

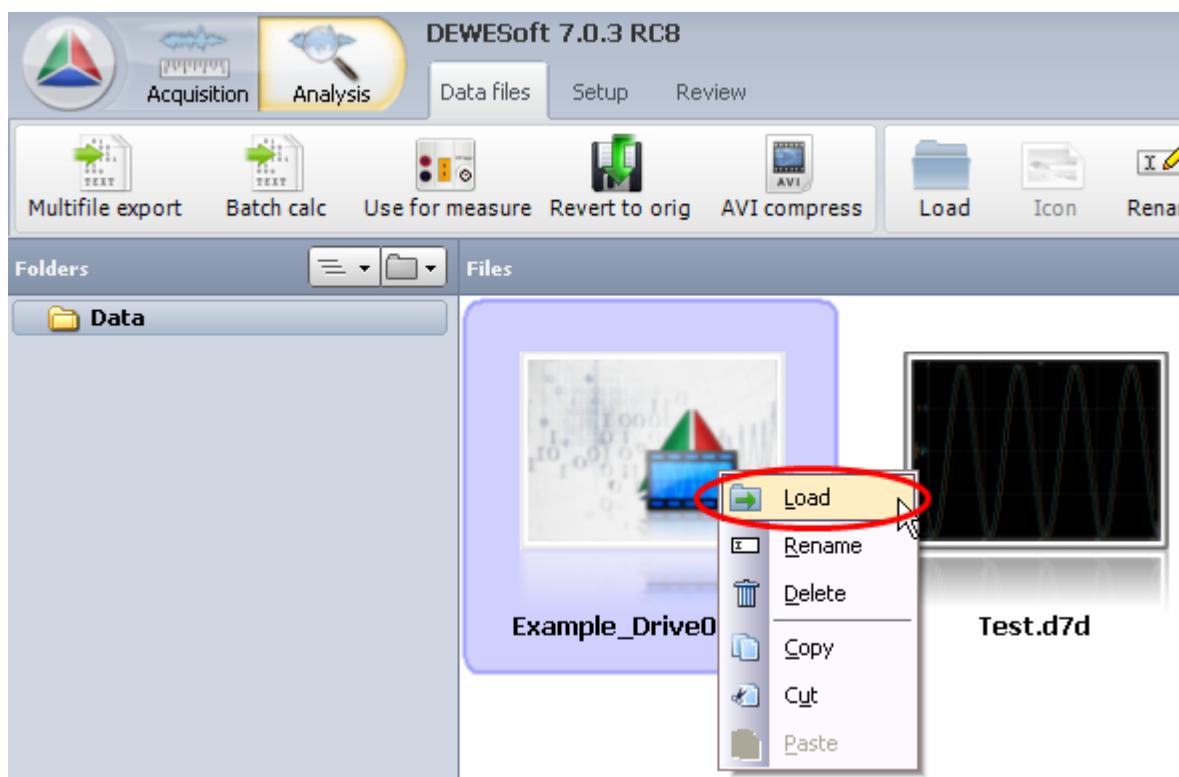
Interface: [!App](#) read-only

2.1.11.52 LoadFile

```
procedure LoadFile(const FileName: WideString);
```

This action will load a Dewesoft data file. If the file name is wrong or if it is not defined, it will show the dialog to choose the file manually. If the path is not defined, it will try to load the data from the main data folder of the currently selected project.

This function corresponds to loading a data file in Analysis mode:



Interface: IApp

Classifier	Name	Type	Description
const	FileName	WideString	the name and path of the file to load. If the file name is wrong or if it is not defined, it will show the dialog to choose the file manually. If the path is not defined, it will try to load the data from the main data folder of the currently selected project.

2.1.11.53 LoadModuleSetup

```
procedure LoadModuleSetup(const FileName: WideString);
```

LoadModuleSetup loads the setup of modules only (no screens, no triggers, etc.) from a DEWESoft® setup file (.d7s); i.e. only the channel groups AI and PAD are loaded.

usually you should use [LoadSetup](#) instead.

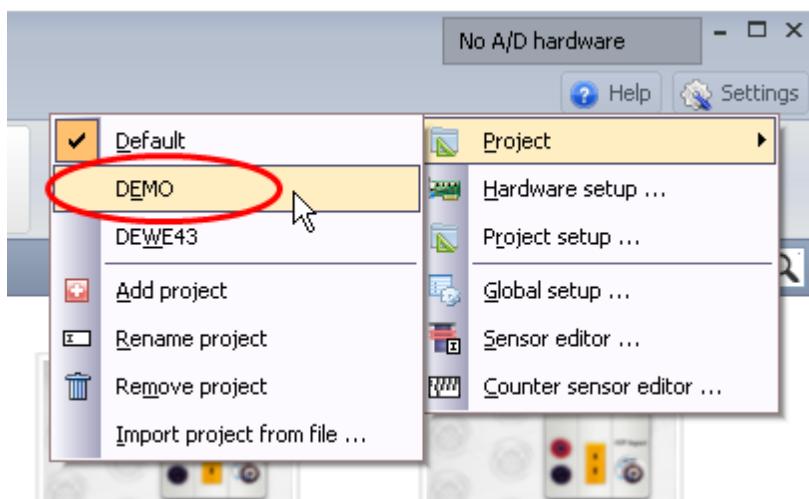
Interface: [IApp](#)

Classifier	Name	Type	Description
const	FileName	WideString	is the file name of the setup file including its path

2.1.11.54 LoadProject

```
procedure LoadProject(const Name: WideString);
```

This action will open the project file. This function corresponds to selecting the project directly:



Interface: [IApp](#)

Classifier	Name	Type	Description
const	Name	WideString	The name of the project (without path or extension)

2.1.11.55 LoadSequence

```
procedure LoadSequence(const FileName: WideString);
```

To load a sequence file specified by `FileName`

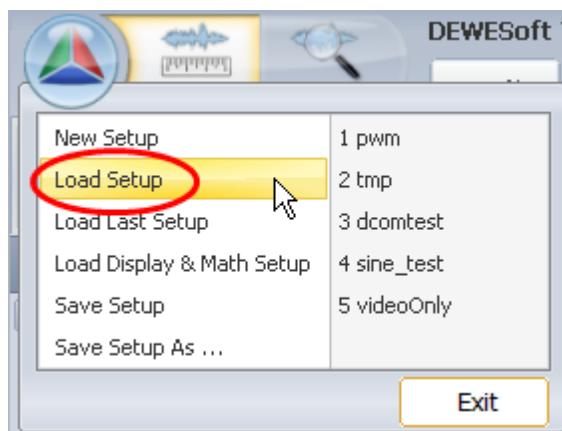
Interface: [IApp](#)

Classifier	Name	Type	Description
const	FileName	WideString	The filename, path and extension of the Sequence file. If the path is missing, the DEWEsoft® Setup-directory (see <code>sdSetupDataDir</code> in SpecDirectory) will be used. If the extension is missing, <code>d7t</code> will be used.

2.1.11.56 LoadSetup

```
procedure LoadSetup(const FileName: WideString);
```

To load a setup file specified by `FileName`. This function corresponds to clicking [Load Setup](#) from the main menu:



see also: [NewSetup](#), [SaveSetup](#), [MenuClick](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
const	FileName	WideString	The filename, path and extension of the Setup file. If the path is missing, the DEWEsoft Setup-directory (see <code>sdSetupDataDir</code> in SpecDirectory) will be used. If the extension is missing, <code>d7s</code> will be used.

2.1.11.57 LoadSetupFromXML

```
procedure LoadSetupFromXML(const XML: WideString);
```

will load the XML-channel setup which is stored in the parameter `XML`

see also: [SaveSetupToXML](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
const	XML	WideString	the XML-channel setup

2.1.11.58 MainDataDir

```
property MainDataDir: WideString
```

MainDataDir is the path of the directory where measurement data is stored. This is the same as [GetSpecDir](#)(sdDataDir).

e.g. D:\DEWEsoft7\Data\

See also: [GetSpecDir](#), [SpecDirectory](#)

Interface: [IApp](#)  read-only

2.1.11.59 MainWindowHandle

```
property MainWindowHandle: Integer
```

Handle of the main DEWEsoft® form.

Interface: [IApp](#)  read-only

2.1.11.60 MainWndMessage

```
procedure MainWndMessage(Msg: Integer; WParam: Integer; Wait: WordBool);
```

If it is necessary to send a message to DEWEsoft® from any thread other than the main-thread of a plug-in, MainWindowMessage has to be used.

Finally, Msg and Param will be passed to the [IPlugin2.OnOleMsg](#) method of all active plugins.

see also: [IPlugin2.OnOleMsg](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
	Msg	Integer	are Integer values specifying the message in order to be able to send the appropriate message when IPlugin2.OnOleMsg is called within the plug-in.
	WParam	Integer	
	Wait	WordBool	specifies whether the thread from which MainWndMessage was called will wait or continue execution.

2.1.11.61 ManualStart

```
procedure ManualStart();
```

ManualStart causes a manual start trigger, when DEWEsoft® is in measure mode, a trigger is available (see

(Illustration below) and armed:

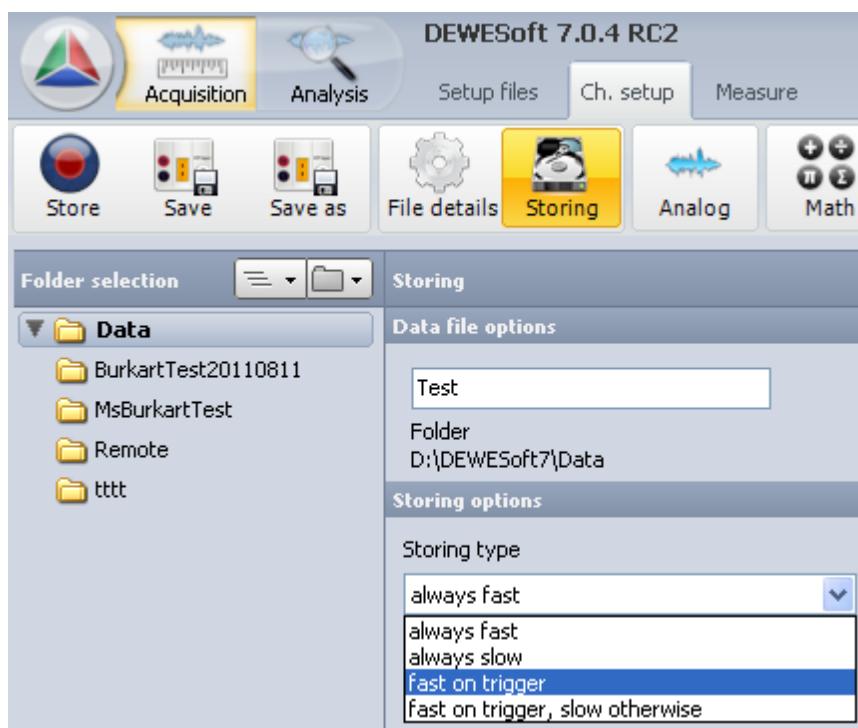


Illustration 150: Fast On Trigger

This function corresponds to clicking the trigger button as shown in the illustration below:



Illustration 151: Manual Trigger Start

see also: [ManualStop](#)

Interface: [IApp](#)

2.1.11.62 ManualStop

```
procedure ManualStop();
```

ManualStop causes a manual stop of a trigger, when DEWEsoft® is in measure mode, a trigger is available (see illustration below) and currently active.

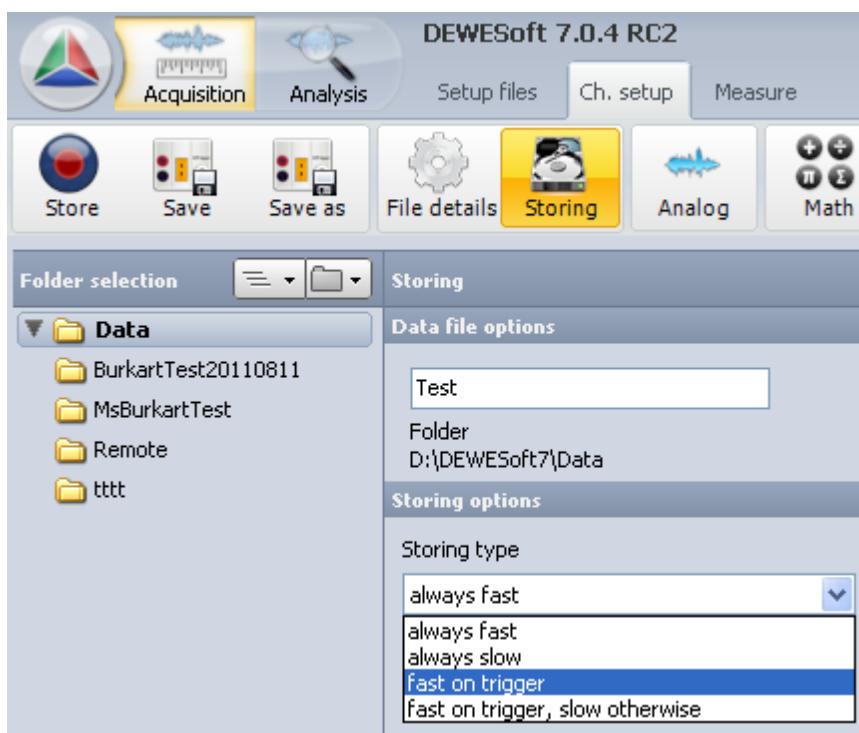


Illustration 152: Fast On Trigger

This function corresponds to clicking the active trigger button as shown in the illustration below:



Illustration 153: Manual Trigger Stop

see also: [ManualStart](#)

Interface: IApp

2.1.11.63 MasterClock

property MasterClock: IMasterClock

Returns the current time [IMasterClock](#) object which can be used to get the current time in seconds (see [IMasterClock.GetCurrentTime](#)) during measurement. This information can be used by plug-ins to timestamp asynchronous data.

see also: [IMasterClock](#), [IPlugin3.ProvidesClock](#), [IPlugin3.OnGetClock](#)

Interface: IApp read-only

2.1.11.64 Math

property Math: IMath

provides access to the [IMath](#) interface. See [IMath](#) for details.

Interface: [IApp](#) read-only

2.1.11.65 Measure

```
procedure Measure();
```

This action will put DEWEsoft® in *Measure (Acquisition)* mode. It is the opposite action from *Analysis* (see [Analyze](#)) and has the same effect as clicking on the [Acquisition](#) button:



see also: [Analyze](#), [ActualRunMode](#), [GUI Navigation](#)

Interface: `IApp`

2.1.11.66 MeasureSampleRate

```
property MeasureSampleRate: Integer
```

see [MeasureSampleRateEx](#)

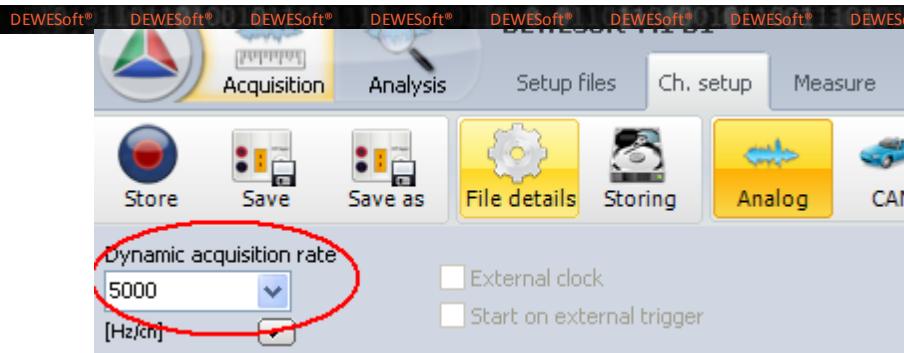
Interface: IApp

2.1.11.67 MeasureSampleRateEx

property MeasureSampleRateEx: Double

The sample rate that will be used in Measure mode for the data acquisition

IMPORTANT: DEWEsoft® must be running in *Acquisition* mode (see [Measure](#)) and you must be in the channel setup screen (see [SetupScreen](#)) as shown in the next screenshot (otherwise the command is ignored):



compare to [IData.SampleRate](#), [IData.SampleRateEx](#)

see also: [Sample Rates](#), [MeasureSampleRate](#)

Interface: [IApp](#) read/write

2.1.11.68 MenuClick

```
procedure MenuClick(Item: MenuItems);
```

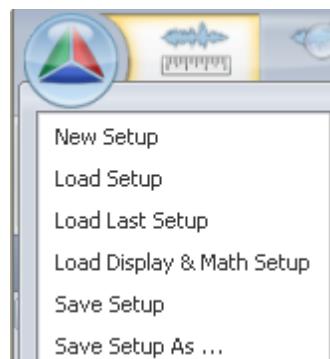
MenuClick causes the menu item (see image below) defined by Item to be clicked.

E.g. if Item is ItemLoadSetup it is the same as clicking the *Load Setup* menu item from the main menu (see image below).

see also: [LoadSetup](#), [SaveSetup](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
	Item	MenuItems	see MenuItems



2.1.11.69 Modules

```
property Modules: IModules
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Modules provides access to the [IModules](#) interface.

See [IModules](#).

Interface: IApp read-only

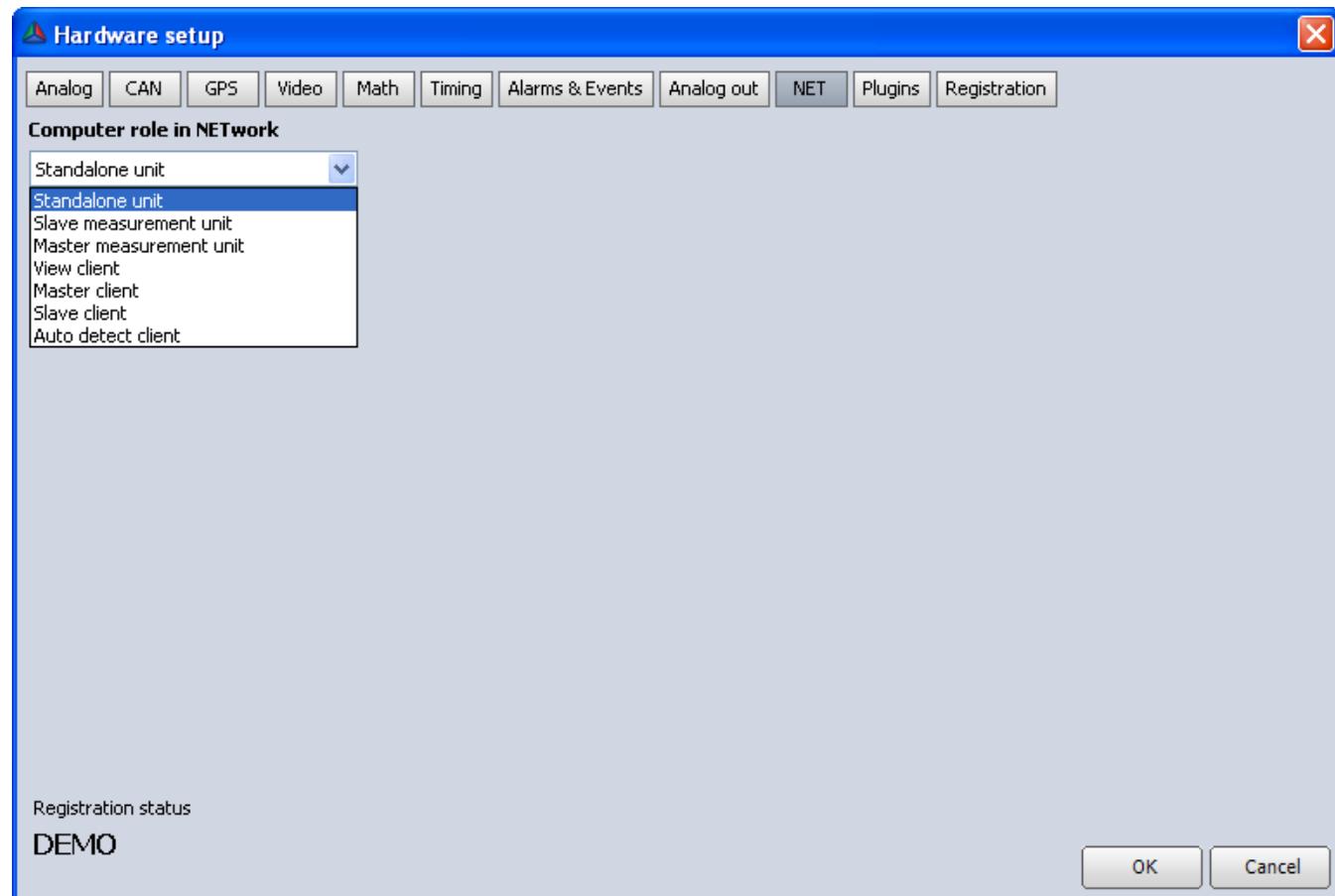
2.1.11.70 NETMode

property NETMode: Integer

returns the mode of the DEWESoft® NET option (for distributed data acquisition):

- 0...Standalone unit (NET option is disabled)
 - 1...Slave measurement unit = Slave client
 - 2...Master measurement unit
 - 3...View client
 - 4...Master client
 - 5...Auto detect client

see also: [RemoteControlled](#)



Interface: [IApp](#) read-only

2.1.11.71 NewSetup

```
procedure NewSetup();
```

This action will open a new setup. It is equal to the Dewesoft button – [New setup](#) menu item:



see also: [LoadSetup](#), [SaveSetup](#)

Interface: [IApp](#)

2.1.11.72 NotifyTrackingChanged

```
procedure NotifyTrackingChanged(Tracking: WordBool; TimeDiff: Double);
```

only valid for plugins that provide the master clock

Interface: [IApp](#)

Classifier	Name	Type	Description
	Tracking	WordBool	false: lost clock information
	TimeDiff	Double	only valid if tracking is true - difference between real time (e.g. GPS) and current device time

2.1.11.73 OfflineCalc

```
property OfflineCalc: IOfflineCalc
```

provides access to the [IOfflineCalc](#) interface

Interface: [IApp](#) read-only

2.1.11.74 Parent

```
property Parent: Integer
```

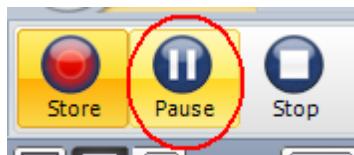
Parent provides a reference to the window handle of DEWEsoft®'s parent form.

Interface: [IApp](#) read/write use only in automation applications

2.1.11.75 PauseStoring

```
procedure PauseStoring();
```

This action will pause storing. It is equivalent to the [Pause](#) button and works in conjunction with [ResumeStoring](#). Dewesoft must be in [Measure](#) mode and should be storing the data that this command is valid.



see also: [StartStoring](#), [ResumeStoring](#), [Stop](#), [DisableStoring](#)

Interface: [IApp](#)

2.1.11.76 PowerModules

```
property PowerModules: IPowerModules
```

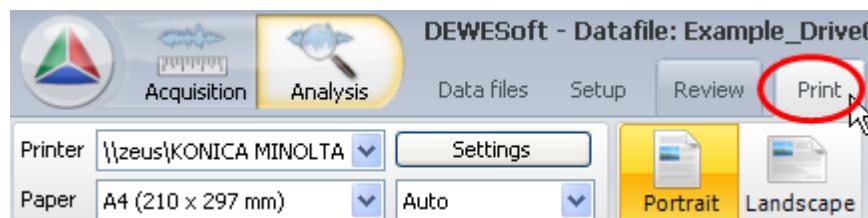
provides access to the [IPowerModules](#) interface

Interface: [IApp](#) read-only

2.1.11.77 PrintScreen

```
procedure PrintScreen(ShowDialog: WordBool);
```

Will make a screen dump to the printer. It is the same as the Analysis Print command:



This procedure can only be used when the application is in Analysis mode (see [Analyze](#)): then the Print button would be shown). By default this procedure prints the screen on the default printer.

Interface: [IApp](#)

Classifier	Name	Type	Description
	ShowDialog	WordBool	defines whether the printing dialog is shown or not

2.1.11.78 ProjectManager

property ProjectManager: [IProjectManager](#)

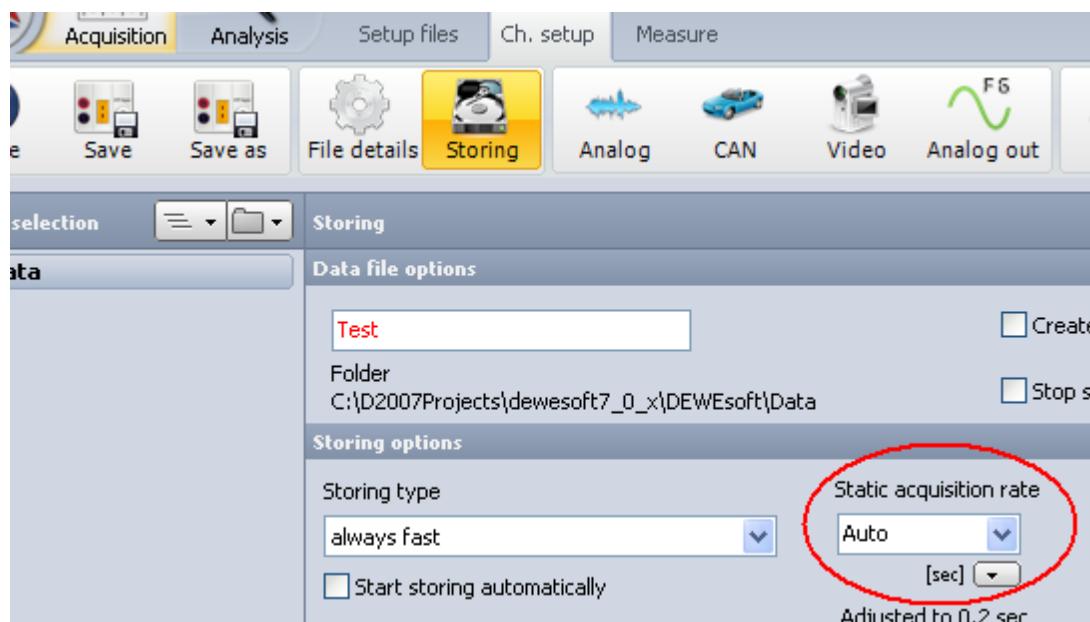
provides access to the project manager interface

Interface: [IApp](#) read-only

2.1.11.79 ReducedRate

property ReducedRate: Single

This action will set the reduced rate of the data. You must be on the [Ch. setup](#) screen of the Acquisition mode ([Measure](#)) that this command is valid. It is the same as the *Static acquisition rate* input field:



see also: [Sample Rates](#), [IStoreEngine.StoreMode](#)

Interface: [IApp](#) read/write

2.1.11.80 RegType

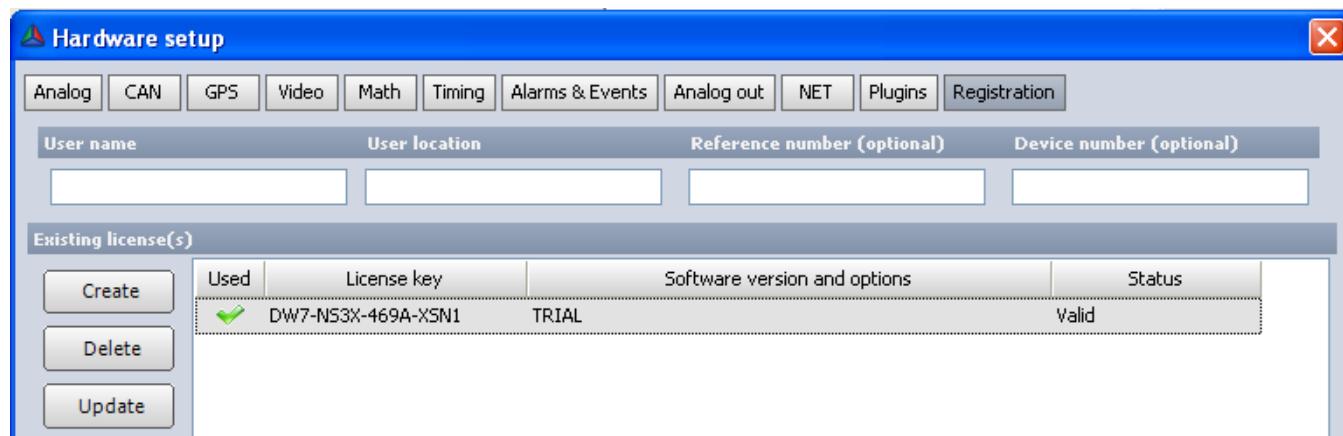
property RegType: Integer

Information about the registration type (DEWEsoft® license)

- 4...registration not found
- 3...registration not supported
- 2...trial license has expired

- 1...demo license
 - 0... evaluation license
 - 1... DEWESoft® lite edition
 - 2... DEWESoft® standard edition
 - 3... DEWESoft® professional edition
 - 4... DEWESoft® DSA edition
 - 5... DEWESoft® enterprise edition

Interface: IApp  read-only



2.1.11.81 RemoteControlled

property RemoteControlled: WordBool

Is TRUE, if DEWEsoft® is controlled by another remote DEWEsoft® instance.

see also: [NETMode](#)

Interface: IApp

2.1.11.82 ResumeStoring

```
procedure ResumeStoring();
```

This action will resume storing if paused (see [PauseStoring](#)) . Dewesoft must be in [Measure](#) mode, and storing must be paused, that this command is valid.

It is equivalent to the [Resume](#) button and works in conjunction with [PauseStoring](#):



see also: [StartStoring](#), [PauseStoring](#), [Stop](#), [DisableStoring](#)

Interface: [IApp](#)

2.1.11.83 SaveSetup

```
procedure SaveSetup(const FileName: WideString);
```

This action will save the setup under the defined file name (this corresponds to the *Save as* button in the following image).

If the file name is empty, it will store the currently loaded setup (this corresponds to the *Save* button in the following image):



see also: [LoadSetup](#), [NewSetup](#), [MenuClick](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
const	FileName	WideString	is the file name of the setup file to store to including its path and extension. If the filename is empty, the currently loaded setup is used. If the path is missing, the DEWEsoft Setup-directory is used. If the extension is missing, .d7s will be used.

2.1.11.84 SaveSetupToXML

```
procedure SaveSetupToXML(out XML: WideString);
```

will return the current setup in XML format

see also: [LoadSetupFromXML](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
out	XML	WideString	the current setup in XML format

2.1.11.85 Screens

property Screens: [IScreens](#)

Screens allows handling the different screens (aka. Instruments) of DEWESoft® (see [GUI Navigation](#) for details).

Moreover, each of these screens can have sub-screens.

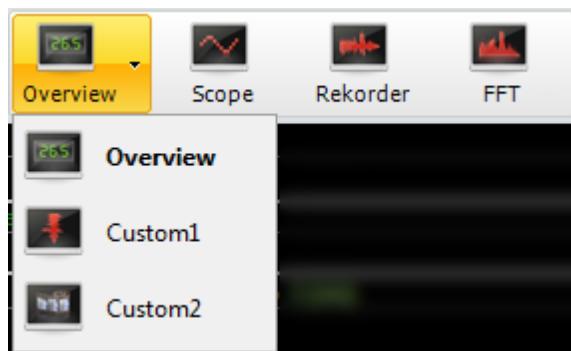


Illustration 154: Screens

In the Illustration above you can see 6 screens in total:

- *Overview*: which has 2 sub-screens
 - *Custom1*
 - *Custom2*
- *Scope*
- *Recorder*
- *FFT*

To navigate between the main screens (aka. Instruments) use the function [SetInstrument](#): e.g. [SetInstrument](#) (2) will activate the main screen called *Scope*) to navigate to a sub-screen you must first navigate to the main screen and then to the sub-screen. e.g. to navigate to the screen *Custom1*, you would first call [SetInstrument](#) (2) and then [SetScreenIndex](#) (1).

When you now want to go back to the *Overview* screen you have to call [SetScreenIndex](#) (0).

See also: [GUI Navigation](#), [SetInstrument](#), [ActiveScreen](#), [IScreens](#), [IScreen](#)

Interface: [IApp](#) read-only

2.1.11.86 SendCommand

```
function SendCommand(const Cmd: WideString; Timeout: Integer): WideString;
```

DEPRECATED - use amplifier interface

To send a command to a hardware module via the (internal) module bus. Returns the answer of the device.

SendCommand sends a command string to a hardware module via the (internal) module bus and returns its answer.

Interface: [IApp](#)

Classifier	Name	Type	Description
const	Cmd	WideString	the command string to send to the module
	Timeout	Integer	specifies how long to wait for an answer of the module. Its unit is ms.
-	RESULT	WideString	the answer of the device

2.1.11.87 SendKey

```
procedure SendKey(Key: LongWord);
```

This action will send the key stroke to Dewesoft.

Interface: [IApp](#)

Classifier	Name	Type	Description
	Key	LongWord	is the key code according to the character map.

2.1.11.88 SetFullScreen

```
procedure SetFullScreen(Full: WordBool);
```

SetFullScreen switches between full screen display mode and standard display mode, similar to pressing *CTRL+F*.

Interface: [IApp](#)

Classifier	Name	Type	Description
	Full	WordBool	defines the mode to switch to. If Full is True, the window switches to full screen. If Full is False, the window switches to normal display mode.

2.1.11.89 SetHeaderData

```
procedure SetHeaderData(const Caption: WideString; const Header: WideString);
```

`SetHeaderData` allows changing the data of the header of a measurement file.

see also: [IGlobalHeader](#), [IGHObject](#)

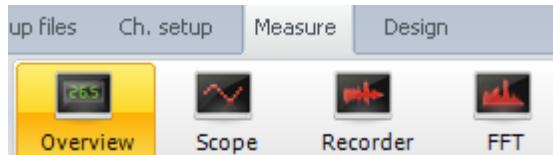
Interface: IApp

Classifier	Name	Type	Description
const	Caption	WideString	denotes which field of the header to change
const	Header	WideString	is the text being set to the specified field

2.1.11.90 SetInstrument

```
procedure SetInstrument(Id: Integer);
```

Will change the shown display. The `Id` parameter defines the main index of the screen. This command is valid only in [Measure mode](#):



Instruments may have subitems: see [SetScreenIndex](#)

see also: [GUI Navigation](#), [ActiveScreen](#), [IApp.Screens](#)

Interface: [IApp](#)

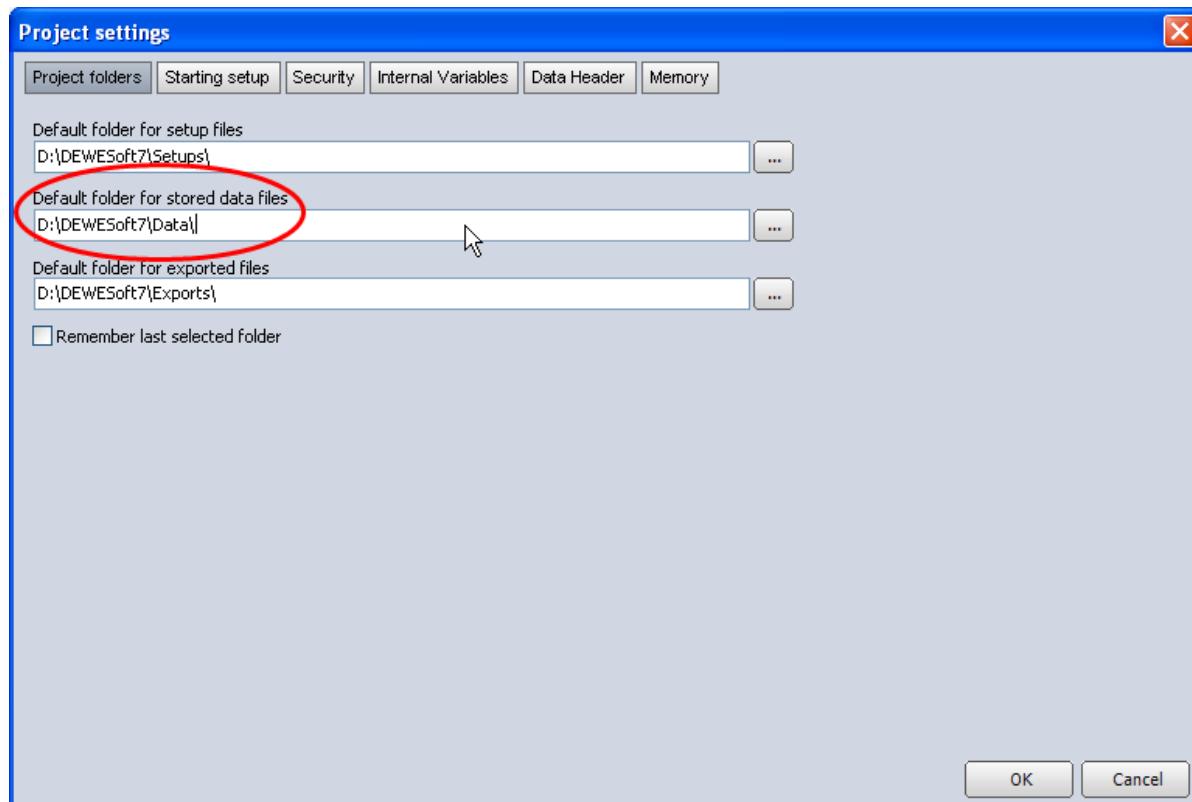
Classifier	Name	Type	Description
	Id	Integer	<p>The index of the first screen is 0. e.g. (default setup)</p> <p>0 refers to Overview</p> <p>1 refers to Scope</p> <p>2 refers to Recorder</p> <p>3 refers to FFT</p> <p>note that the user can completely customize the screens (aka. instruments). he could delete/insert/rename screens: Thus only the number is important, not the display name.</p>

2.1.11.91 SetMainDataDir

```
procedure SetMainDataDir(const DataDir: WideString);
```

This action sets the main folder for DEWEsoft data.

This is similar to setting it via [Settings - Project setup ...](#) manually.



Interface: [IApp](#)

Classifier	Name	Type	Description
const	DataDir	WideString	the path where the data files are stored

2.1.11.92 SetMainToolBar

```
procedure SetMainToolBar(const TabName: WideString; const ButtonName: WideString);
```

This action is very important. It sets DEWEsoft® to a certain mode.

see also: [GUI Navigation](#), [SetScreenIndex](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
const	TabName	WideString	<p>defines the name of the main tool bar. Here the name of a main tool bar must be entered (like Ch. Setup, Measure, Print and so on).</p> <p>Please note that the available tabs are dependant on the mode, that DEWESoft® is working in (Measure or Analysis mode):</p> 
const	ButtonName	WideString	<p>defines which main menu button will be selected. Here it is again important to take care which buttons are available in which menu:</p>  <p>Note: you must always use the English names for the buttons, even if you have chosen another display language.</p>

2.1.11.93 SetRemoteMode

```
procedure SetRemoteMode(Remote: WordBool);
```

DEPRECATED

Interface: [IApp](#)

Classifier	Name	Type	Description
	Remote	WordBool	-

2.1.11.94 SetScopeParams

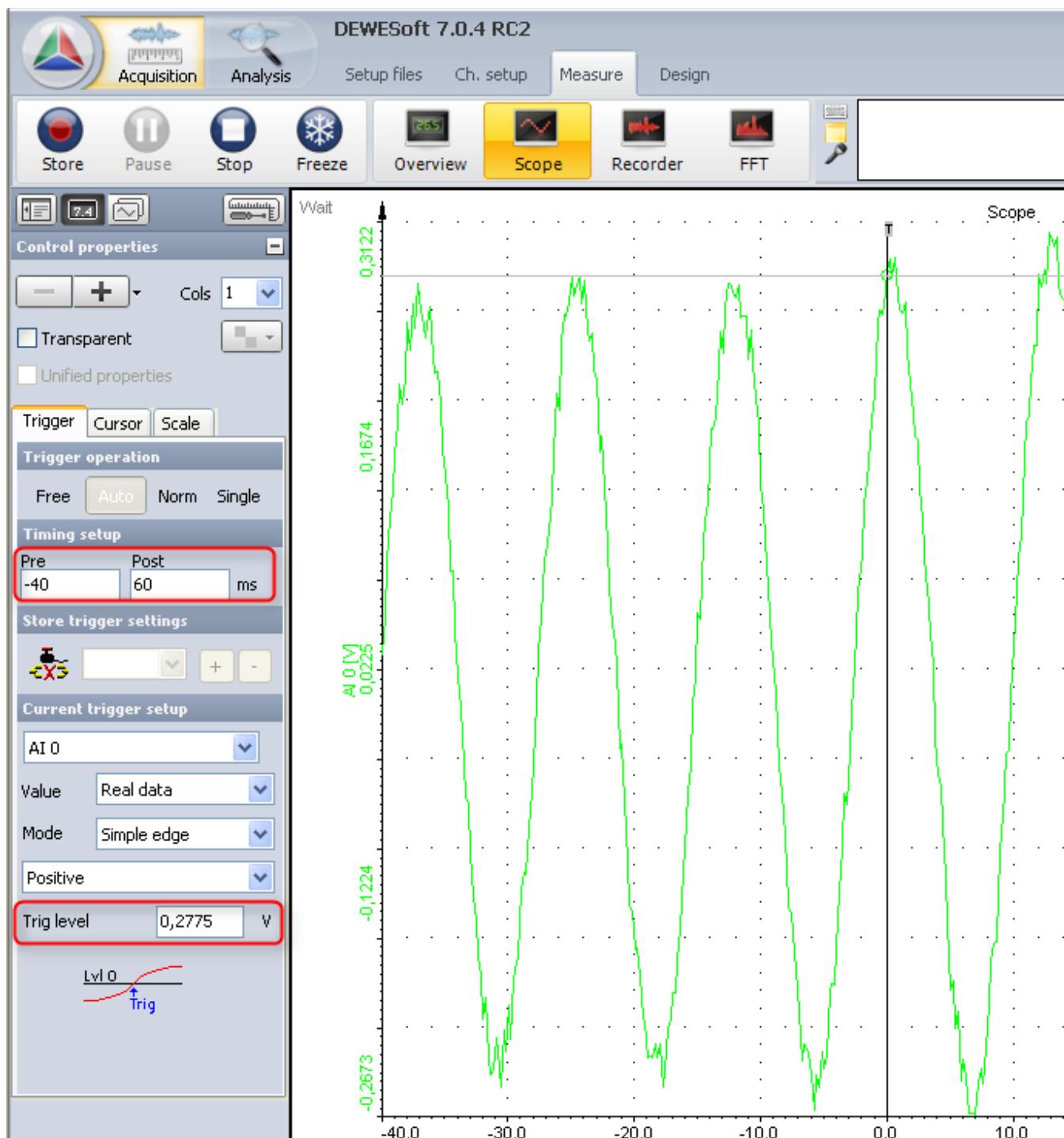
```
function SetScopeParams(PreTime: Double; PostTime: Double; const Channel: IChannel;
; Level: Single): Integer;
```

To set the pre- and the post-trigger time, the trigger channel and the trigger level of the scope. Returns the number of samples from the whole shot.

see also: [SetScopeUsed](#)

Interface: [IApp](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	PreTime	Double	is the pre-trigger time in ms								
	PostTime	Double	is the post-trigger time in ms								
const	Channel	IChannel	the channel that used for triggering								
	Level	Single	specifies the trigger level								
-	RESULT	Integer	Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com								



2.1.11.95 SetScopeUsed

```
procedure SetScopeUsed(Value: WordBool);
```

`SetScopeUsed` activates or deactivates the scope trigger.

see also: [SetScopeParams](#)

Interface: IApp

Classifier	Name	Type	Description
	Value	WordBool	denotes whether the scope trigger is used or not. True will activate the scope trigger and, False will deactivate it.

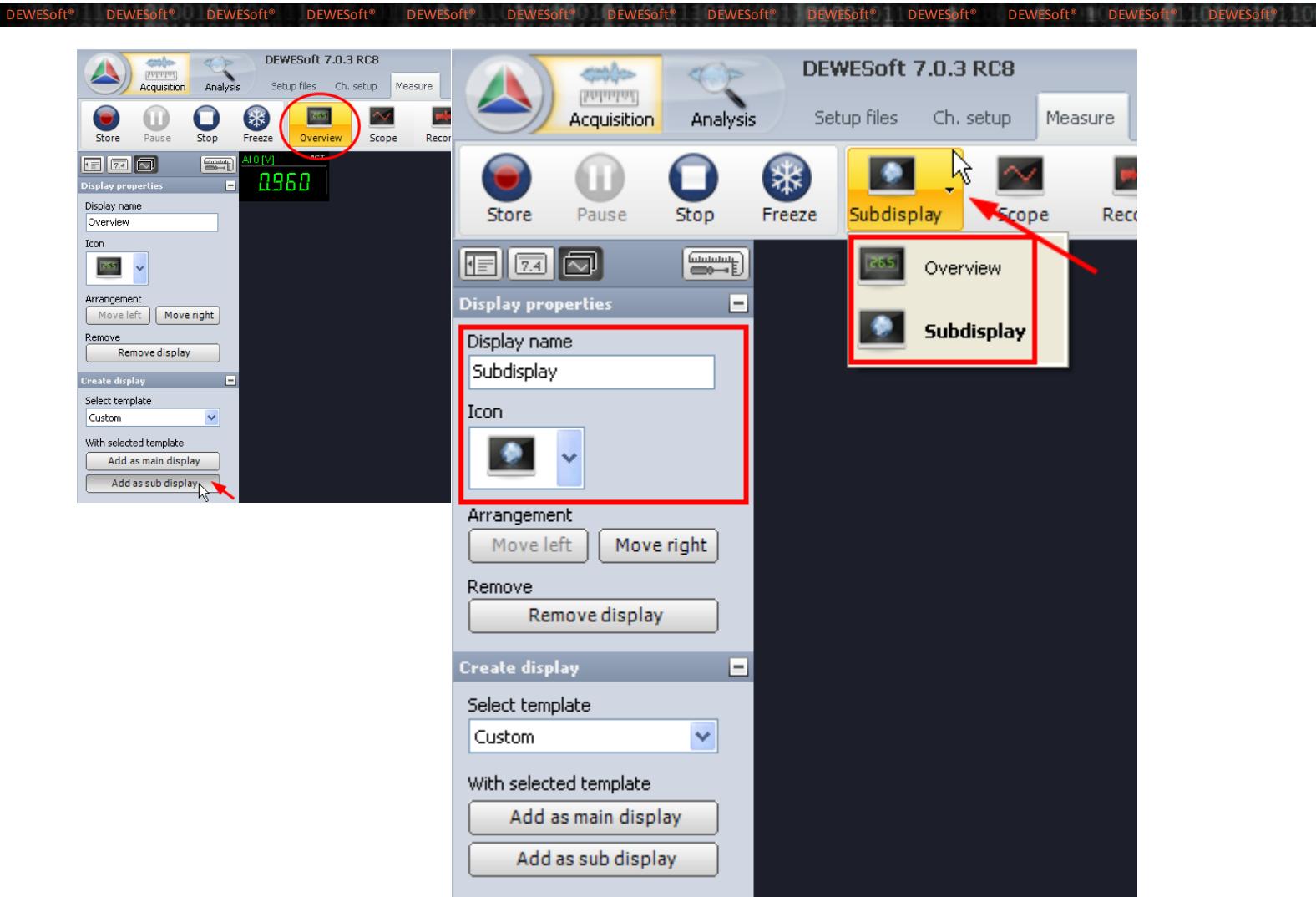
2.1.11.96 SetScreenIndex

```
procedure SetScreenIndex(Index: Integer);
```

This action changes the given screen (aka. sub display). If the main display has several screens, this action can change between them.

Make sure to call [IApp.SetInstrument](#) first.

You can add a sub display by clicking on the [Add as sub display button](#). The image below shows 2 sub-displays called *Overview* and *Subdisplay*.



see also: [SetMainToolBar](#), [Screens](#), [GoToInstruments](#), [GUI Navigation](#)

Interface: [IApp](#)

Classifier	Name	Type	Description
	Index	Integer	<p>The index of the screen. The screens are indexed starting at 0 up to NumberOfScreens-1.</p> <p>IApp.ActiveScreen</p> <p>In the example above, the sub-displays called <i>Overview</i> has the index 0 and the subdisplay called <i>Subscreen</i> has the index 1.</p>

2.1.11.97 SetStoreMode

```
procedure SetStoreMode (Mode: Integer);
```

This procedure can set the store mode.

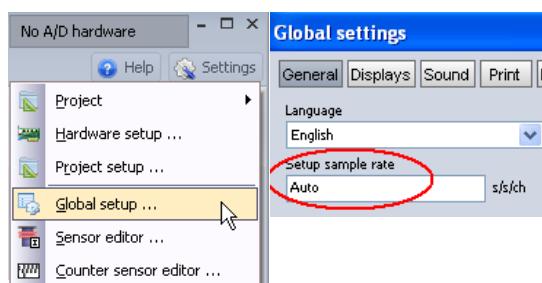
Interface: [IApp](#)

Classifier	Name	Type	Description
	Mode	Integer	<p>Storing type</p> <p>always fast always slow Fast on trigger Fast on trigger, slow otherwise</p> <p>0 ... always fast; 1 ... always slow; 2 ... fast on trigger; 3 ... fast on trigger, slow otherwise</p>

2.1.11.98 SetupSampleRate

```
property SetupSampleRate: Integer
```

This property changes the setup sample rate. It is similar to the [Settings – Global Setup ... Setup sample rate](#) input. DEWESoft® must NOT be in acquisition mode when this parameter is set.



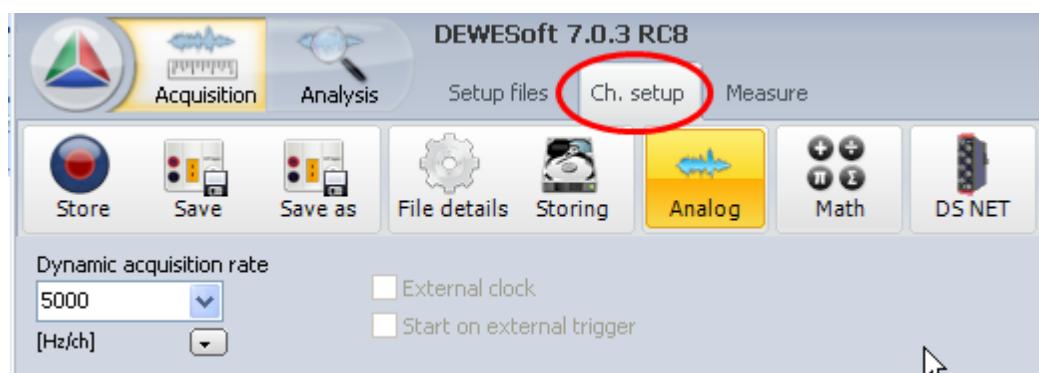
see also: [Sample Rates](#)

Interface: [IApp](#) read/write

2.1.11.99 SetupScreen

```
procedure SetupScreen();
```

Will switch to the setup screen. DEWESoft® must be in [Measure](#) mode. This is the same as clicking the [Ch. setup](#) button:



see also: [GoToInstruments](#), [Measure Mode](#), [GUI Navigation](#)

Interface: [IApp](#)

2.1.11.100 ShowCaptionBar

```
procedure ShowCaptionBar();
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Interface: [IApp](#)

2.1.11.101 ShowInstrumentsInFullScreen

```
property ShowInstrumentsInFullScreen: WordBool
```

if the Instruments toolbar should be shown in full-screen or not.

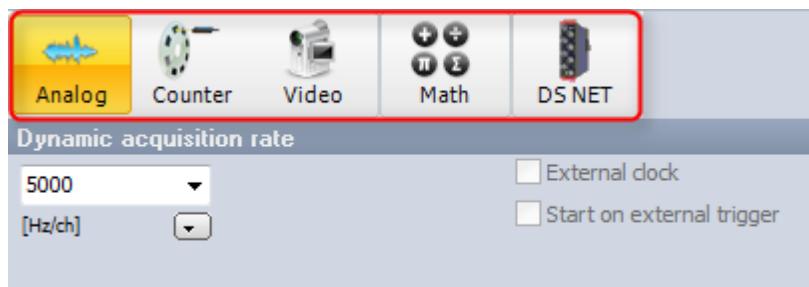


Illustration 155: Instruments shown in full screen mode

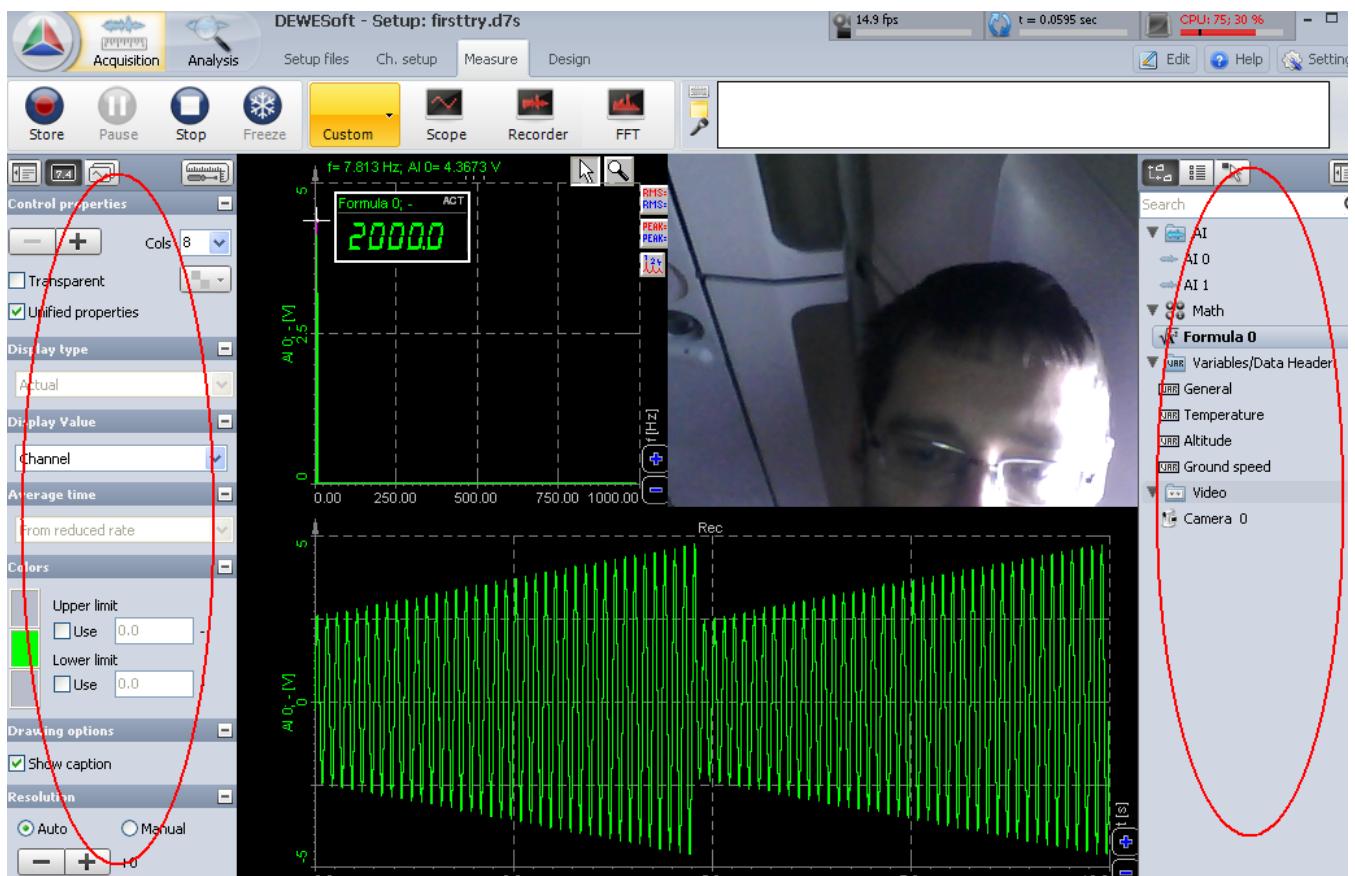
see also: [SetFullScreen](#)

Interface: [IApp](#) read/write

2.1.11.102 ShowPropertyFrame

property ShowPropertyFrame: WordBool

Will show or hide the property frame and channel list while measuring:



see also> [Measure](#)

Interface: [IApp](#) read/write

2.1.11.103 ShowSROptions

```
property ShowSROptions: WordBool
```

This action will show or hide the sample rate panel in [Ch. Setup – Storing](#) and [Ch. Setup – Analog](#) tabs:



see also: [Measure](#), [SetupScreen](#)

Interface: [IApp](#) read/write

2.1.11.104 ShowSensorEditor

```
procedure ShowSensorEditor();
```

`ShowSensorEditor` opens the sensor editor window. This is the same as clicking on the [Sensor editor...](#) menu item.

Interface: [IApp](#)



2.1.11.105 ShowStoreOptions

property ShowStoreOptions: WordBool

This action will show or hide the storing options in *Ch. Setup – Storing* and *Ch. Setup – file details*. It is important if we want the user to be able to change the header entries, but not the file name:



Illustration 156: hidden store options

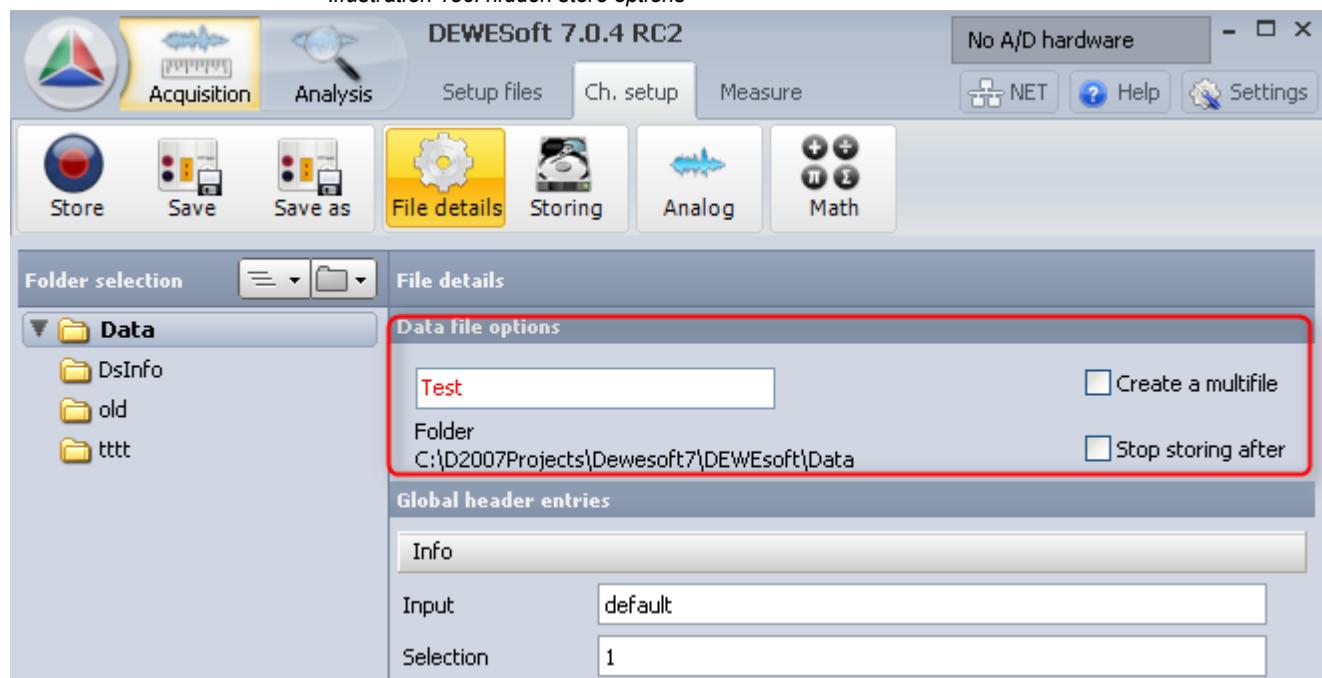


Illustration 157: store options visible

see also: [Measure](#)

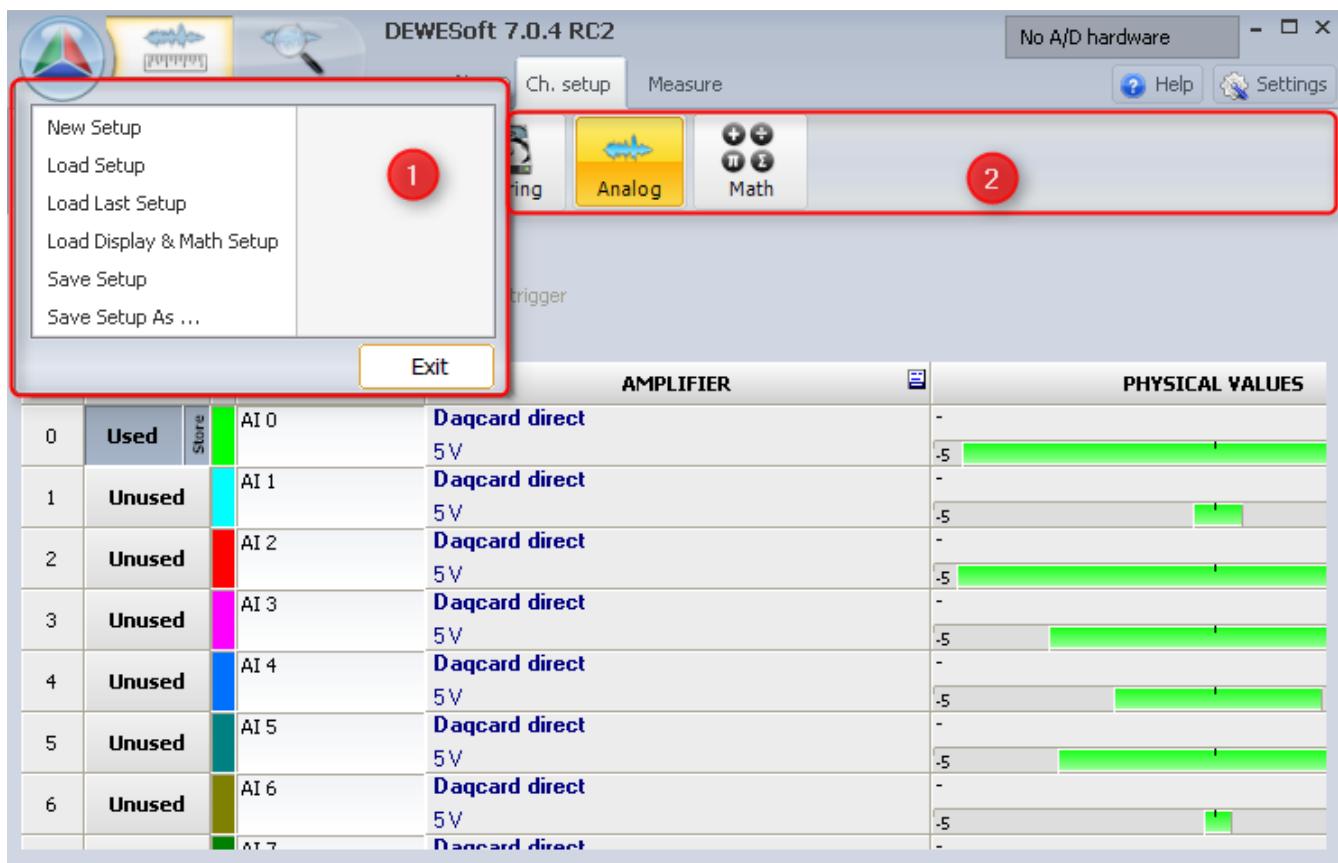
Interface: `IApp` read/write

2.1.11.106 ShowStyle

```
property ShowStyle: Integer
```

ShowStyle is for changing the appearance of the menu bar.

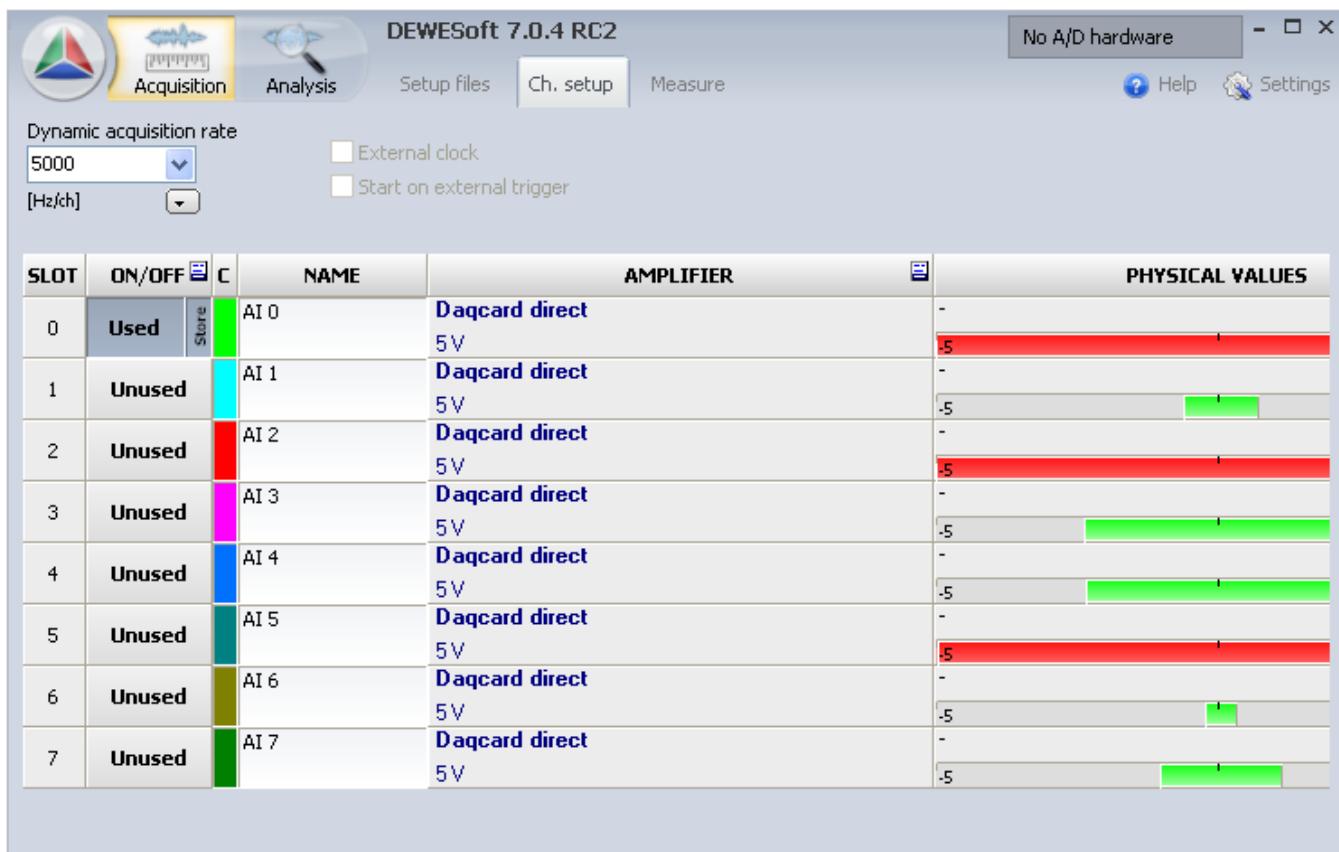
2... shows the menu bar (2) including the drop- down menu (1) bar (shows menu and buttons).



1...shows the menu bar but hides the drop-down menu bar (shows only buttons)



0...hides the whole menu bar (hides menu and buttons)



Interface: [IApp](#) read/write

2.1.11.107 Start

```
function Start(): WordBool;
```

Will switch to [Measure](#) mode and start the data acquisition.

Interface: [IApp](#)

Classifier	Name	Type	Description
-	RESULT	WordBool	true, if the acquisition has been started successfully

2.1.11.108 StartModuleScan

```
procedure StartModuleScan();
```

StartModuleScan allows to manually start the scanning of the modules after having stoped it by [StopModuleScan](#).

When the module scan is active the amplifier number (see image below) will change continuously and also the scan icon will move from channel to channel.



Illustration 158: active module scan

Interface: [IApp](#)

2.1.11.109 StartStoring

```
procedure StartStoring(const FileName: WideString);
```

This action will start storing the data (or arm the trigger, if one is set).

Dewesoft must be in [Measure](#) mode that this command is working. It is the same as the [Store](#) button:



see also: [PauseStoring](#), [ResumeStoring](#), [Stop](#), [DisableStoring](#)

Interface: IApp

Classifier	Name	Type	Description
const	FileName	WideString	is the name of the file where data is stored to including its path and its extension. If the <code>FileName</code> does not include a path, DEWEsoft's datat directory (SetMainDataDir) will be used.

2.1.11.110 StayOnTop

property StayOnTop: WordBool

If the property `StayOnTop` is true, then DEWESoft will be always on top even if another application has the focus.

Interface: IApp

2.1.11.111 Stop

```
procedure Stop();
```

Will stop storing measurement data. This function is similar to clicking the *Stop* button:



see also: [StartStoring](#), [PauseStoring](#), [ResumeStoring](#), [DisableStoring](#)

Interface: IApp

2.1.11.112 StopModuleScan

```
procedure StopModuleScan();
```

`StopModuleScan` allows you to stop the scanning of the modules

Use [StartModuleScan](#) to restart scanning again

When the module scan is stopped the amplifier number (see image below) will not change and also the scan icon will not move any longer from channel to channel

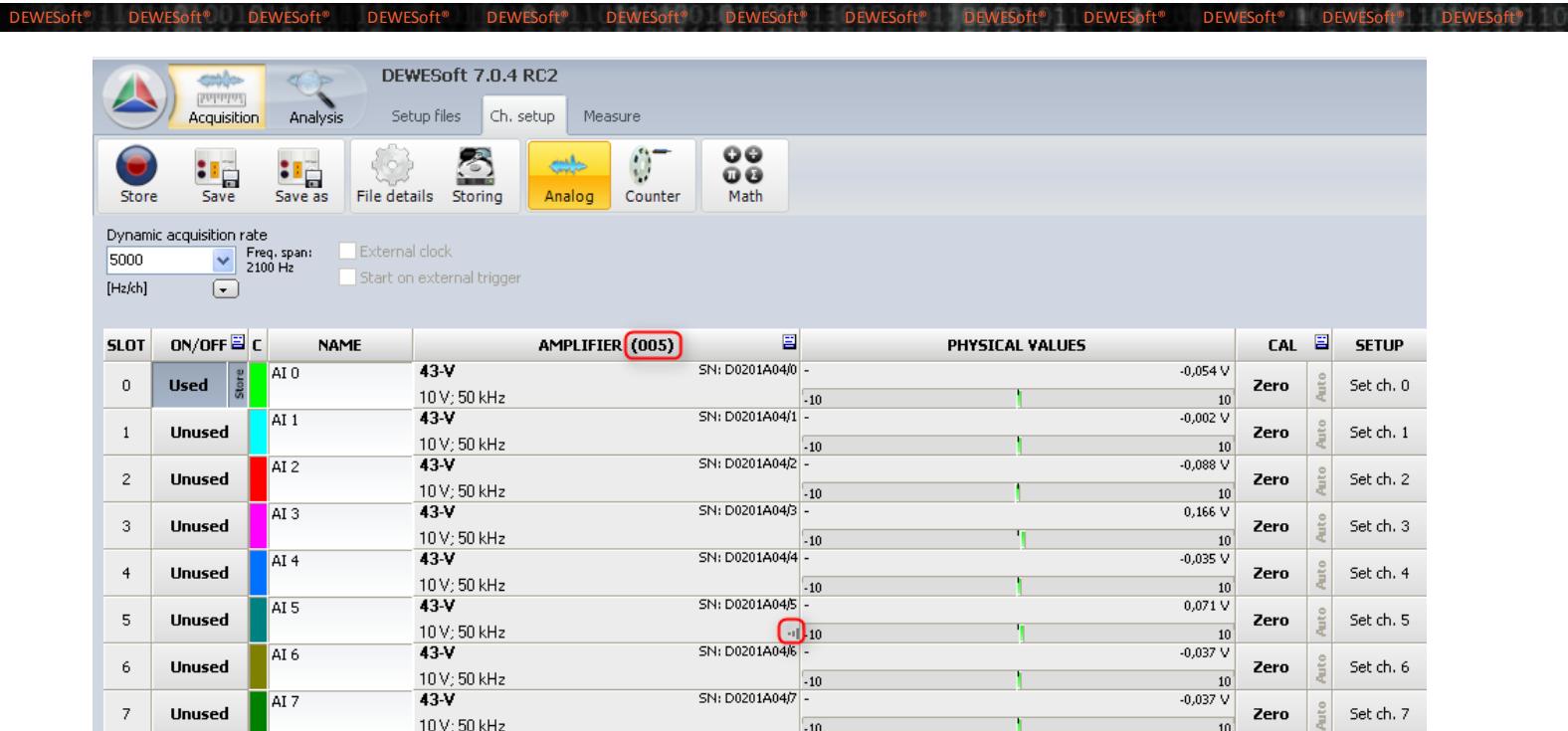


Illustration 159: active module scan

Interface: [IApp](#)

2.1.11.113 StoreEngine

```
property StoreEngine: IStoreEngine
```

StoreEngine has to be used for handling all data storage. See [IStoreEngine](#) for detailed information

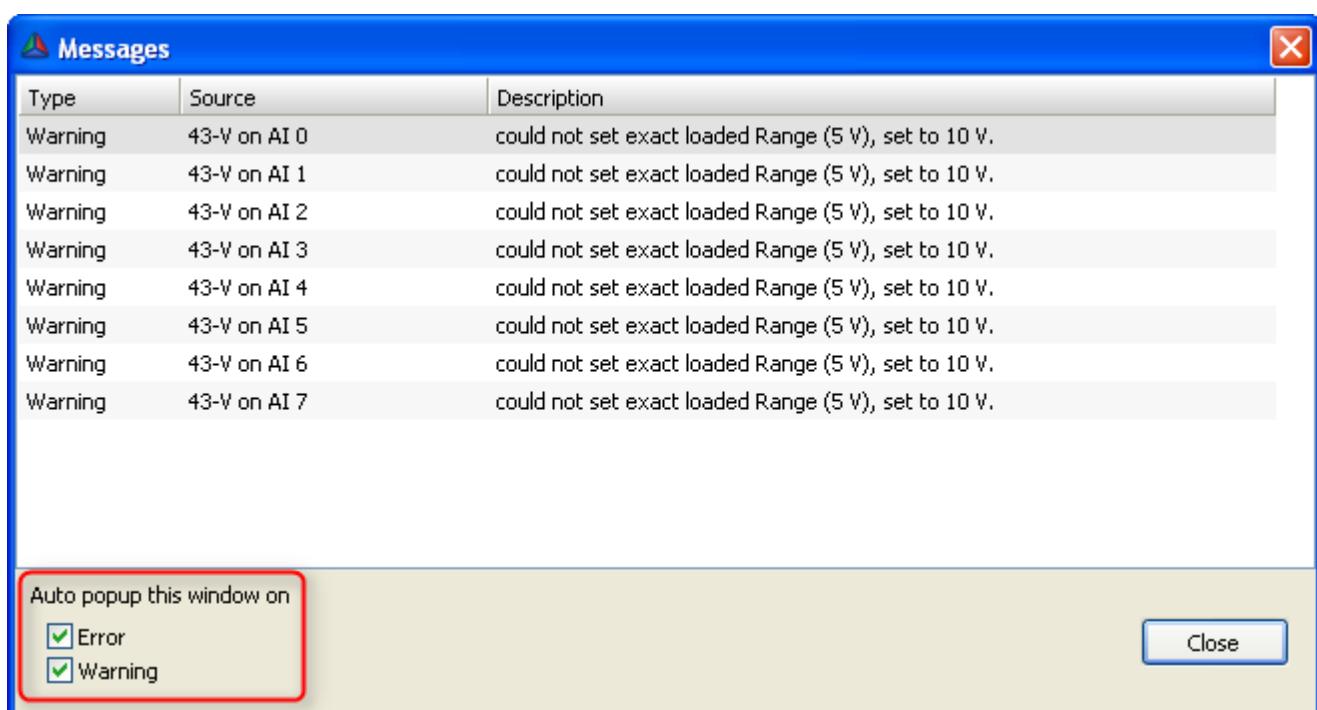
Interface: [IApp](#) read-only

2.1.11.114 SuppressMessages

```
property SuppressMessages: WordBool
```

relates to all message-boxes

if set to TRUE setup error messages will be suppressed. That means, that the messages dialog will not be shown automatically, even if there are error or warning messages and you have activated the auto popup function.



Interface: IApp

2.1.11.115 TimerInterval

property TimerInterval: Integer

The timer interval for DEWEsoft® acquisition in ms. The default is 33 ms.

For plugins it means the time between the calls to the `IPlugin.OnGetData`.

see also [the Async Example in How To Write Data To Channels](#) which clearly shows that this timer interval is not absolutely precise: i.e. do not rely on the fact that it is called exactly every 33ms!

This can also be changed in the *Global Settings* dialogue:

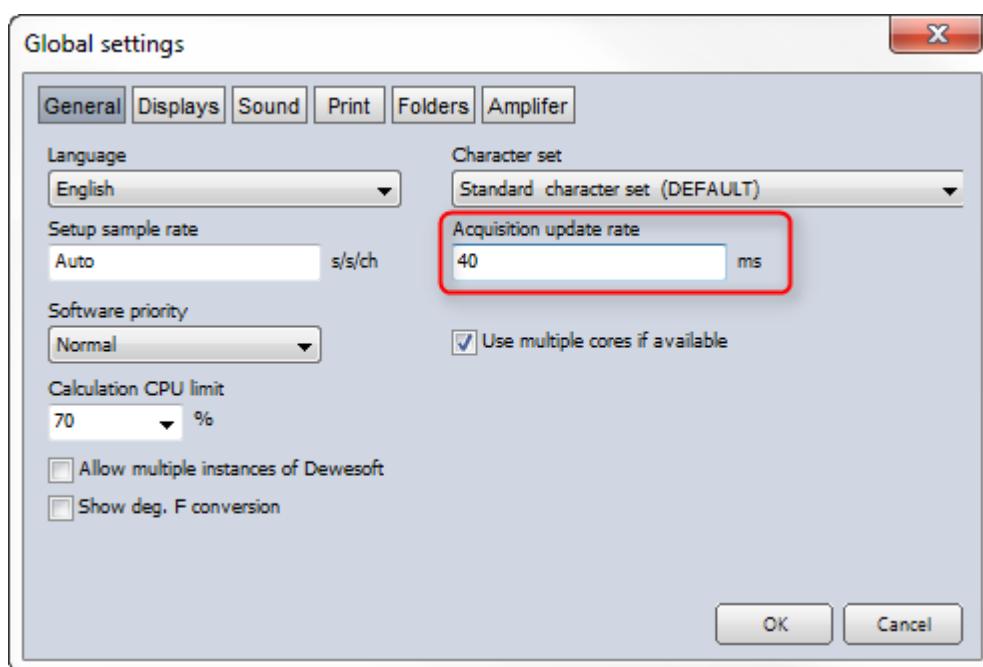


Illustration 160: Timer Interval

Interface: [IApp](#) read/write

2.1.11.116 Timing

property Timing: [ITiming](#)provides access to the [ITiming](#) interfacesee also: [ITiming](#)Interface: [IApp](#) read-only

2.1.11.117 Top

property Top: Integer

Top is the distance between the top border of the DEWESoft® application window and the top border of the screen expressed in pixels.

see also: [Left](#), [Height](#), [Width](#)Interface: [IApp](#) read/write

2.1.11.122 UsedSetupfile

property UsedSetupfile: WideString

UsedSetupfile is the file name of the setup file currently used, including its path and file extension (.d7s). E.g. this could be D:\DEWESoft7\Setups\Default.d7s.

see also: [GetSpecDir](#), sdSetupDataDir in [SpecDirectory](#)

Interface: [IApp](#)  read-only

2.1.11.123 UserInterface

property UserInterface: [IUserInterface](#)

provides access to the [IUserInterface](#) interface

see also: [IUserInterface](#)

Interface: [IApp](#)  read-only

2.1.11.124 Version

property Version: WideString

Version is the number of the currently used DEWESoft® version. E.g. 7.0.4 RC2

Interface: [IApp](#)  read-only

2.1.11.125 Video

property Video: [IVideo](#)

provides access to the video interface

Interface: [IApp](#)  read-only

2.1.11.126 Visible

property Visible: WordBool

Visible defines whether the DEWESoft® application window should be visible or not.

Interface: [IApp](#)  read/write

2.1.11.127 Width

property Width: Integer

Width is the width of the DEWESoft® application window expressed in pixels.

see also: [Height](#), [Top](#), [Left](#),

Interface: [IApp](#)  read/write

2.1.11.128 WriteErrorLog

procedure WriteErrorLog(const Str: WideString);

will write a log message to the applications error log file:

The logfile is usually located here: D:\DEWESoft7\System\V7_0\Logs\ErrorLog.txt

see also: [GetSpecDir](#), sdLogDir in [SpecDirectory](#)

Interface: [IApp](#)

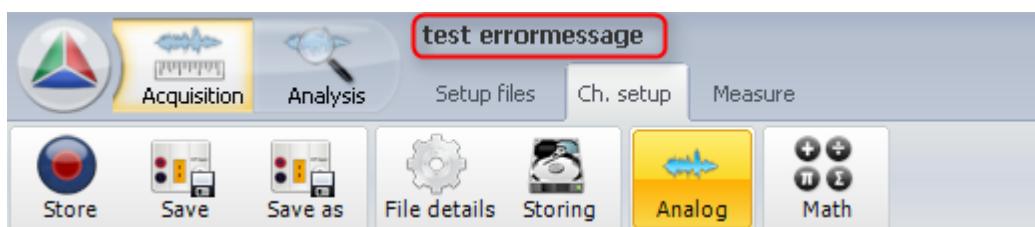
Classifier	Name	Type	Description
const	Str	WideString	the log message to write

2.1.11.129 WriteErrorMessage

procedure WriteErrorMessage(const ErrorMsg: WideString);

only for internal/debugging use

WriteErrorMessage writes an error message to the caption of the title bar of DEWESoft®. The purpose of this procedure is mainly for debugging applications.



Interface: [IApp](#)

Classifier	Name	Type	Description
const	ErrorMsg	WideString	the message text

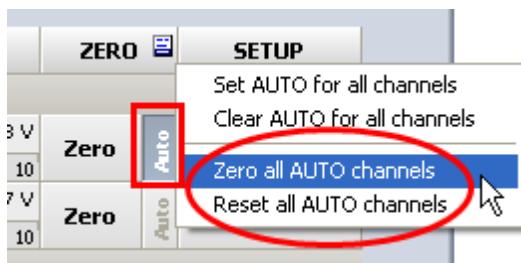
2.1.11.130 ZeroAllAutoChannels

```
procedure ZeroAllAutoChannels(Zero: WordBool);
```

Will zero all channels having `AutoZero` set to `True`. The parameter `Zero` determines whether the offset will be set or cleared.

When the parameter `Zero` is `true`, it is the same, as clicking the *Zero all AUTO channels* pop-up menu entry.

When the parameter `Zero` is `false`, it is the same, as clicking the *Reset all AUTO channels* pop-up menu entry.



In the [Measure](#) mode the Zero button is also available, but only when Dewesoft is not storing data (see [Stop](#)):



If a plugin (see [Plug-Ins](#)) needs to support zeroing in [Measure](#) mode, it must return `TRUE` in the `evEnableZero` event (see [OnEvent](#)) and do the zeroing when the `evOnSetZero` event (see [OnEvent](#)) is called: i.e. change the offset of the channel by calculating the average of the last x-samples.

Interface: [IApp](#)

Classifier	Name	Type	Description
	Zero	WordBool	If <code>Zero</code> is set to <code>True</code> , the function sets the offset to perform zeroing. If <code>Zero</code> is <code>False</code> , it clears the offset from the selected channels

2.1.12 IArrayInfo

information for array channels

see also: [IChannel.ArrayInfo](#), [Array Channels](#)

2.1.12.1 AxisDef

property AxisDef[Index: Integer]: IAxisDef

axis definitions for the array channel

see also: [IAxisDef](#)

Interface: [IArrayInfo](#)



read-only

2.1.12.2 ColorArr

property ColorArr[Ind: Integer]: Integer;

only relevant if `ItemChannels` is `TRUE`.

Interface: [IArrayInfo](#)

re

read/write

2.1.12.3 DimCount

property DimCount: Integer

number of dimensions of this array channel

must be called before `Init`

Interface: [IArrayInfo](#)

50

read/write

2.1.12.4 DimSizes

property DimSizes[Index: Integer]: Integer

number of data points in this array - e.g. lines for FFT

this is the same as `IAxisDef.Size`

must be called before `Init`

Interface: [IArrayInfo](#)

2.1.12.5 Init

```
procedure Init();
```

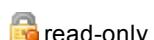
should be called after [DimCount](#) and [DimSizes](#) have been set.

Interface: [IArrayInfo](#)

2.1.12.6 ItemChannels

```
property ItemChannels: WordBool
```

when this property is TRUE it means that each element in the array is a channel ([IChannel](#)). Then we have one element in [ColorArr](#) and [NameArr](#) for each channel.

Interface: [IArrayInfo](#)

2.1.12.7 NameArr

```
property NameArr[Ind: Integer]: WideString
```

only relevant if [ItemChannels](#) is TRUE.

Interface: [IArrayInfo](#)

2.1.12.8 SyncSource

```
property SyncSource: ISyncSource
```

can be used to use a fixed sample rate also for asynchronous array channels.

see also: [Array Channels](#)

Interface: [IArrayInfo](#)

2.1.13 IAveragedFFT

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

2.1.13.1 AveCount

property AveCount: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

AveCount is the number of averages which will be used for the calculation.

Interface: [IAveragedFFT](#) read/write

2.1.13.2 AverageType

property AverageType: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`AverageType` defines the type of averaging which should be applied to the calculation.

Interface: [AveragedFFT](#)  read/write

2.1.13.3 CalculateFromPos

```
function CalculateFromPos(Mid: Integer; Dir: Integer): WordBool;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

`CalculateFromPos` calculates the octave values according to the previously set properties. The function returns `True` if enough data was available for the calculation; `False` otherwise.

Interface: [IAveragedFFT](#)

Classifier	Name	Type	Description
	Mid	Integer	The intermediate buffer position
	Dir	Integer	The direct buffer position
-	<i>RESULT</i>	WordBool	The function returns <code>True</code> if enough data was available for the calculation; <code>False</code> otherwise.

2.1.13.4 GetCPBData

```
procedure GetCPBData  
(ChNo: Integer; OctaveDivider: Integer; Weighting: Integer; out BandCount: Integer;  
  out Data: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Interface: [IAveragedFFT](#)

Classifier	Name	Type	Description
	ChNo	Integer	
	OctaveDivider	Integer	
	Weighting	Integer	
out	BandCount	Integer	
out	Data	OleVariant	

2.1.13.5 GetCPBXData

```
procedure GetCPBXData
(ChNo: Integer; OctaveDivider: Integer; Weighting: Integer; out BandCount: Integer;
 out Data: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Interface: [IAveragedFFT](#)

Classifier	Name	Type	Description
	ChNo	Integer	
	OctaveDivider	Integer	
	Weighting	Integer	
out	BandCount	Integer	
out	Data	OleVariant	

2.1.13.6 GetChannels

```
procedure GetChannels(out Channels: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

GetChannels provides a list of all channels used in [IAveragedFFT](#).

Interface: [IAveragedFFT](#)

Classifier	Name	Type	Description
out	Channels	OleVariant	contains an array of channels of the type IChannel

2.1.13.7 GetData

```
procedure GetData
(ChNo: Integer; OctaveDivider: Integer; Weighting: Integer; out BandCount: Integer;
 out Data: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

To retrieve the calculated EFT/CPB data.

Interface: IAveragedFFT

Classifier	Name	Type	Description
	ChNo	Integer	
	OctaveDivider	Integer	
	Weighting	Integer	
out	BandCount	Integer	
out	Data	OleVariant	

2.1.13.8 GetFFTData

```
procedure GetFFTData  
(ChNo: Integer; Weighting: Integer; DCCutoff: Integer; out Data: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Interface: [IAveragedFFT](#)

Classifier	Name	Type	Description
	ChNo	Integer	
	Weighting	Integer	
	DCCutoff	Integer	
out	Data	OleVariant	

2.1.13.9 Lines

property Lines: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

Lines defines the number of FFT lines used for the calculation.

Interface: [IAveragedFFT](#)  read/write

2.1.13.10 Overlap

property Overlap: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

The amount of overlap for averaging in percent.

Interface: [IAveragedFFT](#) read/write

2.1.13.11 Window

property Window: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

The window type for averaging.

Interface: [IAveragedFFT](#) read/write

2.1.14 IAxisDef

An Axis definition of an array channel.

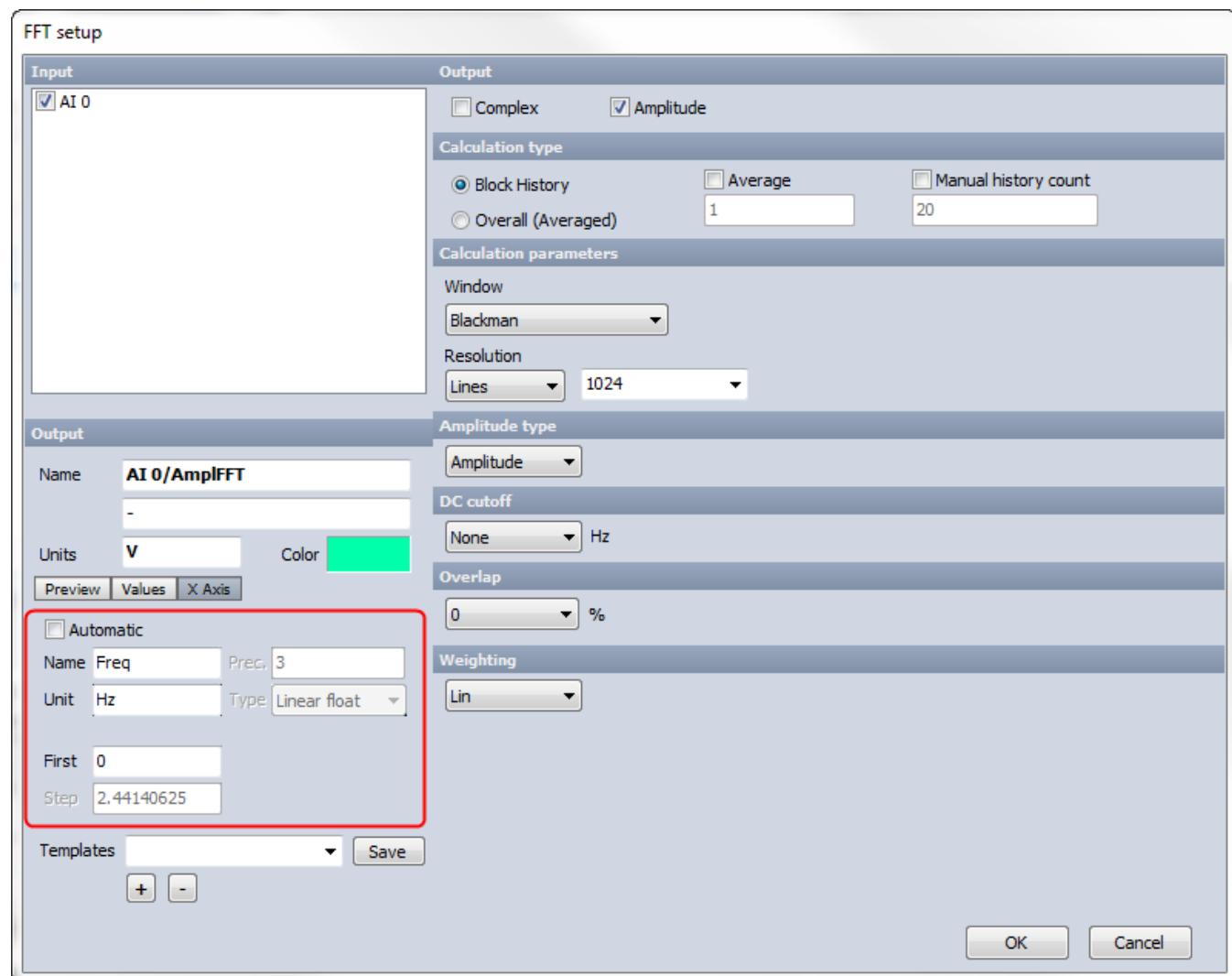


Illustration 161: Axis definition

2.1.14.6 Size

property Size: Integer

the size of the axis

Interface: [IAxisDef](#)



2.1.14.7 StartValue

property StartValue: Double

this is only relevant if the [AxisType](#) is atFloatLinearFunc.

the start value of the axis: usually the axis will start at value 0.

Interface: [IAxisDef](#)



2.1.14.8 StepValue

property StepValue: Double

this is only relevant if the [AxisType](#) is atFloatLinearFunc.

the value between steps on the axis.

Interface: [IAxisDef](#)



2.1.14.9 StringValues

property StringValues[Index: Integer]: WideString

string value for the axis (when [AxisType](#) is atString)

Interface: [IAxisDef](#)



2.1.14.10 _Unit

property _Unit: WideString

the measurement unit to show for the axis (e.g. Hz, V, ..)

Interface: [IAxisDef](#)



2.1.15 ICAN

Interface to access CAN ports.

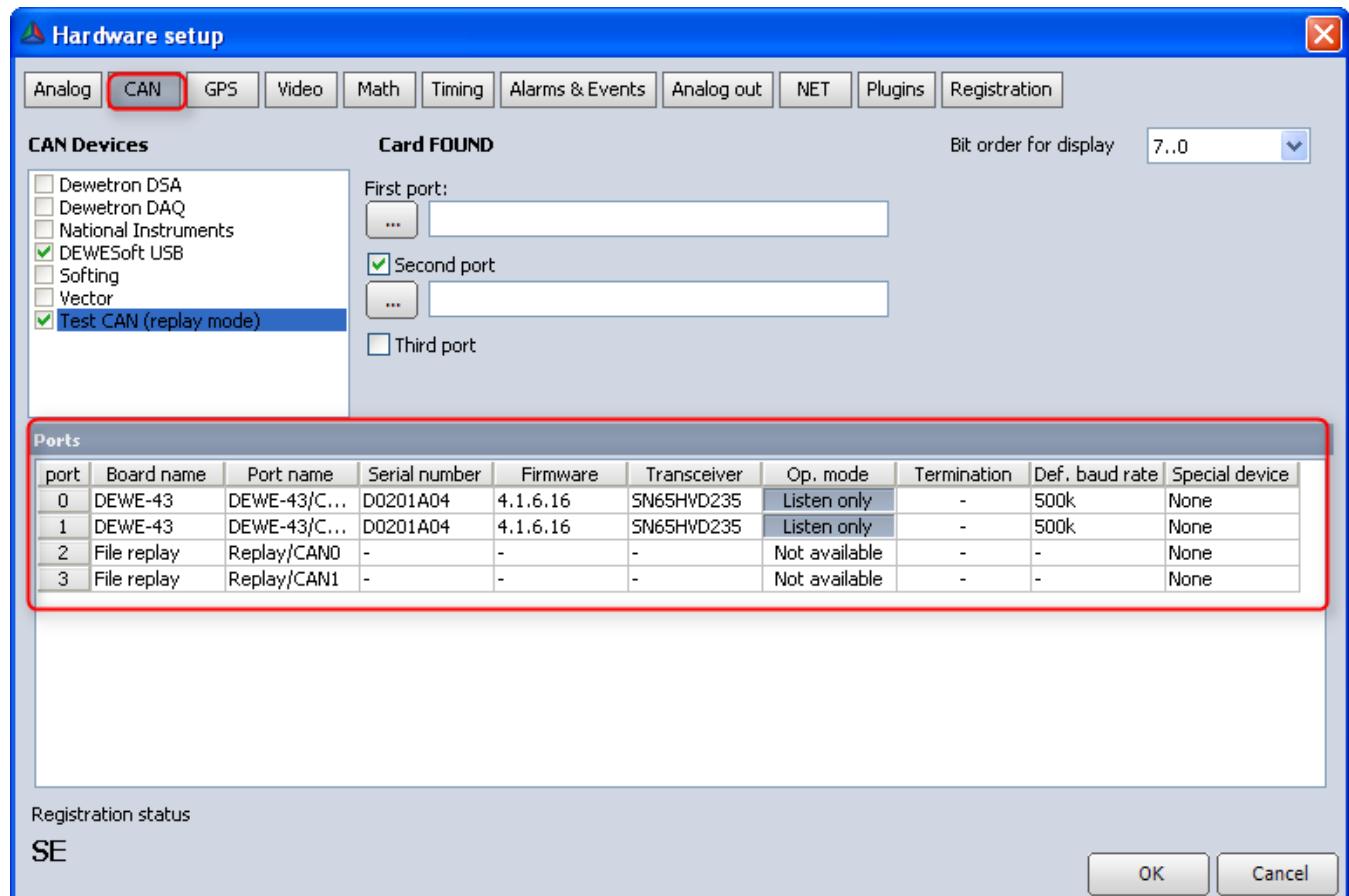


Illustration 162: CAN Hardware setup

see also: [IApp.CAN](#)

2.1.15.1 Count

property Count: Integer

The number of available CAN ports (see [ICANPort](#)) in the [Item](#) list.

see [Item](#)

Interface: [ICAN](#) read-only

2.1.15.2 Item

property Item[Index: Integer]: ICANPort

Item[I] is the CAN port (see [ICANPort](#)) at index I. I is in the range of 0...[Count](#)-1.

see also: [Count](#)

Interface: [ICAN](#)  read-only

2.1.15.3 SupportsOutput

property SupportsOutput: WordBool

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Interface: [ICAN](#)  read-only

2.1.16 ICANContext

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.16.1 GetClock

function GetClock(): Double;

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANContext](#)

Classifier	Name	Type	Description
-	RESULT	Double	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.16.2 GetClockOffset

function GetClockOffset(): Double;

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANContext](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description										
-	RESULT	Double	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>										

2.1.16.3 PortCount

property PortCount: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANContext](#)  read-only

2.1.16.4 Ports

property Ports[Index: Integer]: ICANPortContext

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANContext](#)  read-only

2.1.17 ICANMsg

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.17.1 AddData

procedure AddData(Data: OleVariant; TimeStamp: Double);

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANMsg](#)

Classifier	Name	Type	Description
	Data	OleVariant	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>
	TimeStamp	Double	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>

2.1.18 ICANPort

Interface to access a single CAN port.

see also: [ICAN](#), [IApp.LoadDBC](#)

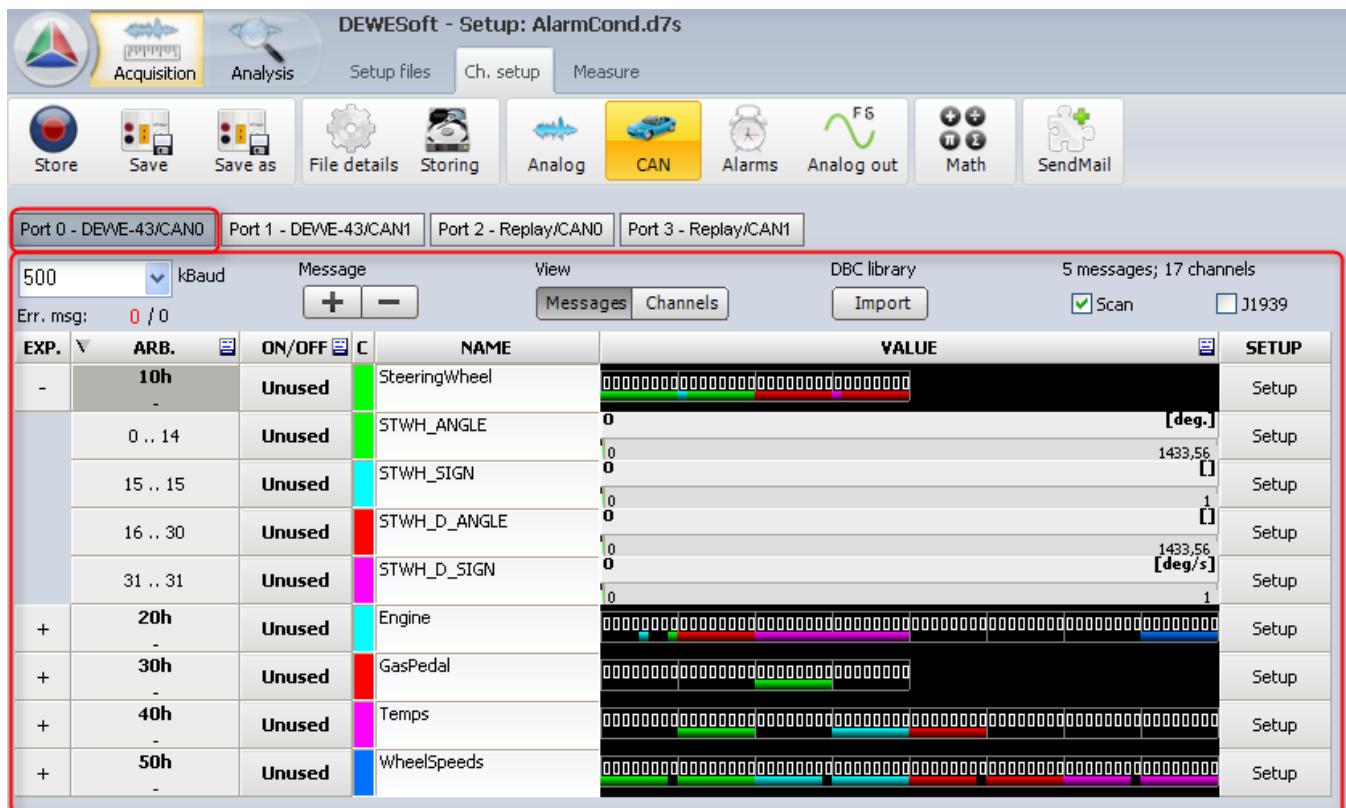


Illustration 163: CAN port

2.1.18.1 Capture

```
procedure Capture(Status: WordBool);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPort](#)

Classifier	Name	Type	Description
	Status	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.18.2 EnableOutput

```
procedure EnableOutput(Enable: WordBool);
```

`EnableOutput` enables or disables a [CAN-port](#) for output.

Interface: [ICANPort](#)

Classifier	Name	Type	Description
	Enable	WordBool	enables or disables the CAN-port for output

2.1.18.3 EndRead

```
procedure EndRead()
```

`EndRead` must be called after reading of data ([ReadMessage](#)) from the [CAN-port](#).

see [ReadMessage](#)

Interface: [ICANPort](#)

2.1.18.4 GetBaudRate

```
function GetBaudRate(): Integer;
```

`GetBaudRate` returns the Baud rate of the CAN-port.

In the example below, the function would return the Integer value `500000`.

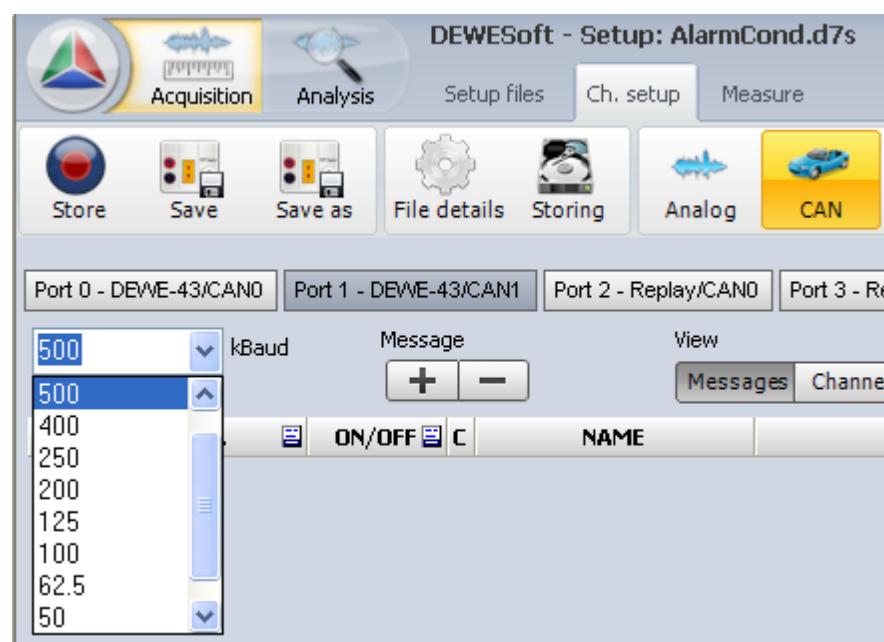


Illustration 164: CAN Baud rate

see also: [GetBaudRateList](#), [SetBaudRate](#)

Interface: [ICANPort](#)

Classifier	Name	Type	Description
-	RESULT	Integer	the current Baud rate of the CAN-port

2.1.18.5 GetBaudRateList

```
function GetBaudRateList(): OleVariant;
```

GetBaudRateList returns an array of available Baud rates in kBaud of the [CAN-port](#). The data type of the array elements is String.

In the example below the array would contain the Strings: '1000', '500', '400', '250', '200', '125', '100', '62.5', '50', '31.25' (Note: '1000' and '31.25' are not visible in the screenshot below)

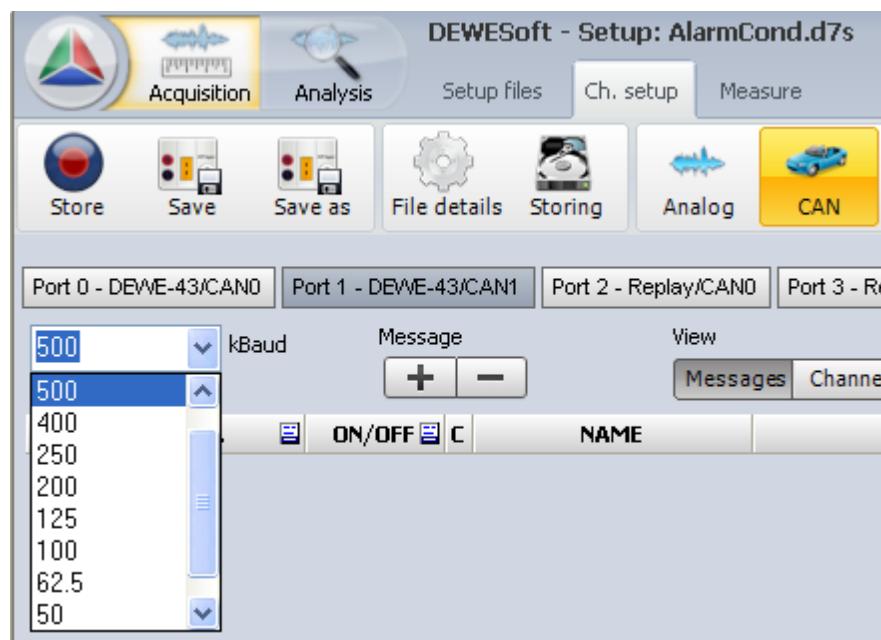


Illustration 165: CAN Baud rate

see also: [GetBaudRate](#), [SetBaudRate](#)

Interface: [ICANPort](#)

Classifier	Name	Type	Description
-	RESULT	OleVariant	an array of available Baud rates in kBaud of the CAN-port

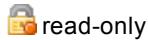
2.1.18.6 MessageCount

property MessageCount: Integer

The number of messages available.

see also: [ReadMessage](#)

Interface: [ICANPort](#)



2.1.18.7 ReadMessage

```
function ReadMessage
(var TimeStamp: Double; var ArbId: Integer; var DataLo: Integer; var DataHi: Integer;
r): WordBool;
```

Read a message from the [CAN-port](#).

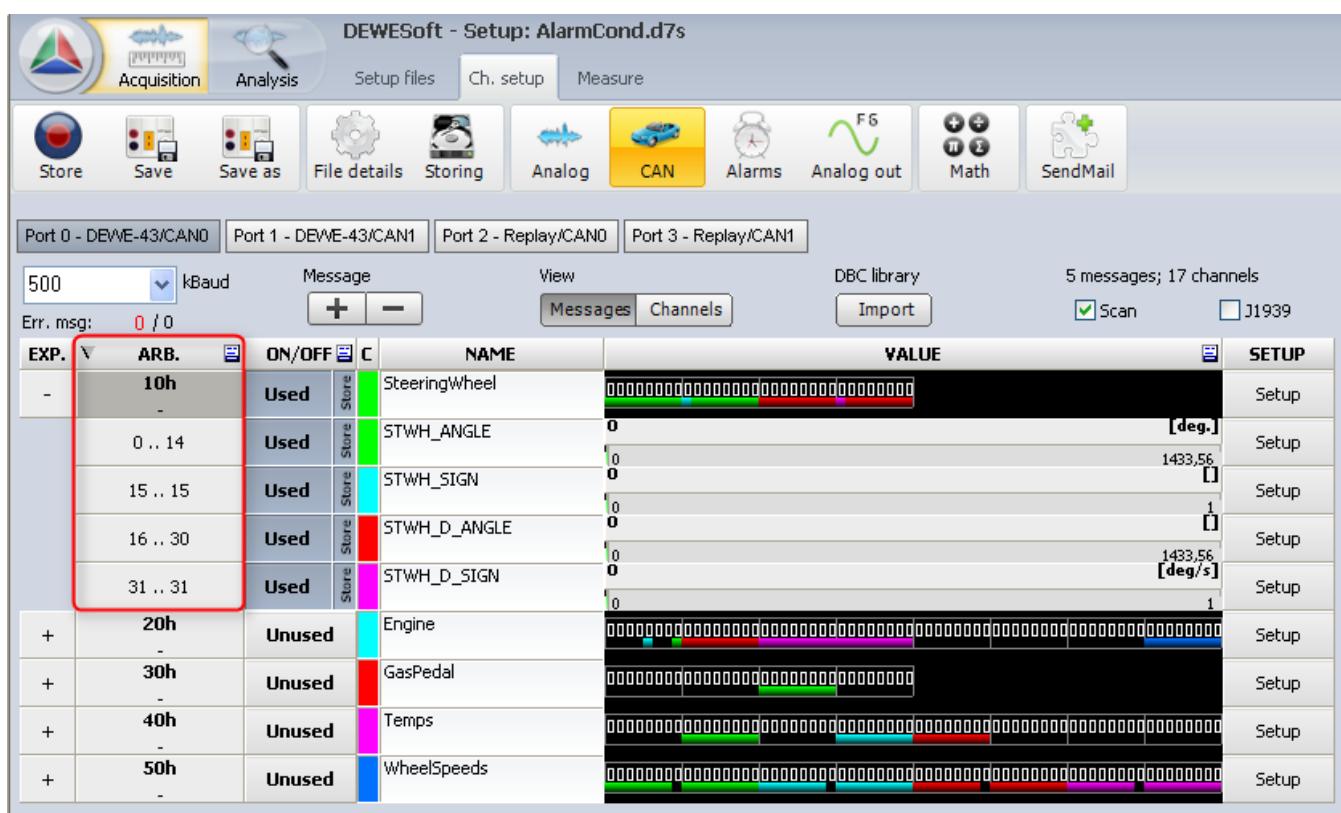
Note: If this function is used e.g. within a plug-in, it should be used within the procedure `OnGetData()` and in conjunction with [StartRead](#) and [EndRead](#). It should be called as often as defined by [MessageCount](#). So, the code could look somehow like the following:

```
procedure Plugin.OnGetData;
var
  i, ArbId, DataLo, DataHi: Integer;
  TimeStamp: Double;
begin
  MyCanPort.StartRead;
  try
    for i:= 0 to MyCanPort.MessageCount - 1 do
      begin
        ReadMessage(TimeStamp, ArbId, DataLo, DataHi);
        //do something with TimeStamp, ArbId, DataLo and DataHi.
      end;
  finally
    MyCanPort.EndRead;
  end;
end;
```

Interface: [ICANPort](#)

Classifier	Name	Type	Description
var	TimeStamp	Double	the timestamp of the read message
var	ArbId	Integer	the arbitration ID of the CAN message: see illustration below
var	DataLo	Integer	Low data of the CAN message
var	DataHi	Integer	Low data of the CAN message
-	RESULT	WordBool	<i>TRUE</i> if the message has been successfully read

ArbId is the Arbitration ID of the CAN Message:



2.1.18.8 SendFrame

```
function SendFrame(Extended: WordBool; ArbId: Integer; Data: T\_CANFrame;
; Size: Integer): WordBool;
```

Send a message to the [CAN-port](#).

see also: [EnableOutput](#)

Interface: [ICANPort](#)

Classifier	Name	Type	Description
	Extended	WordBool	specifies whether extended identifiers are used or not
	ArbId	Integer	the arbitration ID is the identifier of the CAN-message
	Data	T_CANFrame	the data of the message
	Size	Integer	the size of the message
-	<i>RESULT</i>	WordBool	<i>TRUE</i> , if the message has been send successfully

2.1.18.9 SetBaudRate

```
procedure SetBaudRate(BaudRate: Integer);
```

SetBaudRate sets the Baud rate of a [CAN-port](#).

see also: [GetBaudRate](#), [GetBaudRateList](#)

Interface: [ICANPort](#)

Classifier	Name	Type	Description
	BaudRate	Integer	the Baud rate to set (e.g. the Integer value of 125000 will set a Baud rate of 125k)

2.1.18.10 StartRead

```
procedure StartRead();
```

StartRead must be called before reading of data ([ReadMessage](#)) from the [CAN-port](#).

see [ReadMessage](#)

Interface: [ICANPort](#)

2.1.18.11 TotalErrMsgCount

```
property TotalErrMsgCount: Integer
```

the number of error CAN messages that have been received.

see also: [TotalMsgCount](#)

Interface: [ICANPort](#)



2.1.18.12 TotalMsgCount

```
property TotalMsgCount: Integer
```

the total number of CAN messages that have been received

see also: [TotalErrMsgCount](#)

Interface: [ICANPort](#)



2.1.19 ICANPortContext

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.19.1 BaudRate

property BaudRate: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)



2.1.19.2 Captured

property Captured: WordBool

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)



2.1.19.3 GetMsg

function GetMsg(ArbId: Integer; Extended: WordBool; DLC: Integer): ICANMsg;

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)

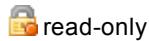
Classifier	Name	Type	Description
	ArbId	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
	Extended	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
	DLC	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	ICANMsg	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.19.4 ListenOnly

```
property ListenOnly: WordBool
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)



2.1.19.5 SetErrMsgCount

```
procedure SetErrMsgCount(Value: Integer);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)

Classifier	Name	Type	Description
	Value	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.19.6 SetTotalMsgCount

```
procedure SetTotalMsgCount(Value: Integer);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)

Classifier	Name	Type	Description
	Value	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.19.7 Termination

```
property Termination: WordBool
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)



2.1.19.8 Used

property Used: WordBool

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICANPortContext](#)

 read-only

2.1.20 ICNTGroup

group for counter channels

2.1.20.1 Count

property Count: Integer

Count is the number of items in the [Item](#) list

Interface: [ICNTGroup](#)

 read-only

2.1.20.2 Item

property Item[Index: Integer]: [IChannel](#)

Item[I] is the counter channel at index I. I is in the range of 0...[Count](#)-1.

see also: [IChannel](#)

Interface: [ICNTGroup](#)

 read-only

2.1.21 ICamera

interface to access a video camera

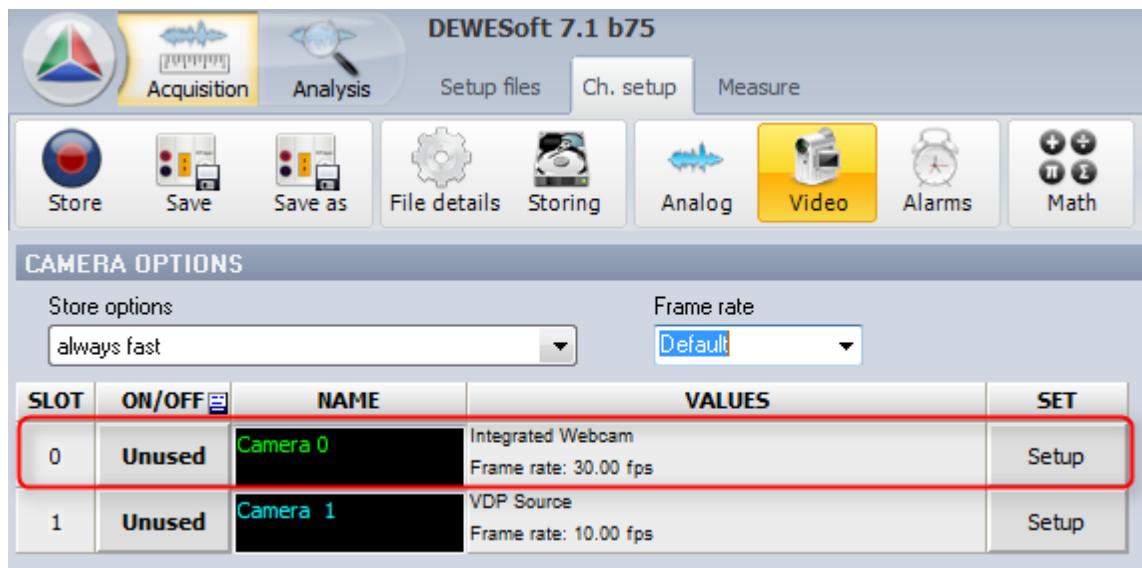


Illustration 166: ICamera

see also: [IVideo.Cameras](#)

2.1.21.1 FrameBufSize

property FrameBufSize: Integer

the size of the frame buffer - analogue to the `DBBufSize` described in [The Buffer Structure](#)

see also: [FrameContentSize](#), [FramePos](#), [FrameList](#)

Interface: [ICamera](#)



2.1.21.2 FrameContentSize

property FrameContentSize: Integer

the amount of data in the frame buffer - analogue to the `DBDataSize` described in [The Buffer Structure](#)

see also: [FrameBufSize](#), [FramePos](#), [FrameList](#)

Interface: [ICamera](#)



2.1.21.3 FrameList

property FrameList[Index: Integer]: [IVideoFrame](#)

a list of the video frames

see also: [FrameBufSize](#), [FrameContentSize](#), [FramePos](#)

Interface: [ICamera](#)  read-only

2.1.21.4 FramePos

property FramePos: Integer

the current position in the frame buffer - analogue to the `DBOPos` described in [The Buffer Structure](#)

see also: [FrameBufSize](#), [FrameContentSize](#), [FrameList](#)

Interface: [ICamera](#)  read-only

2.1.21.5 FrameSizeInBytes

property FrameSizeInBytes: Integer

the size of a single frame in Bytes

Interface: [ICamera](#)  read-only

2.1.21.6 GetBitmapInfoHeader

function GetBitmapInfoHeader(): OleVariant;

standard windows structure describing the bitmap

Interface: [ICamera](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	OleVariant	an array of Bytes

2.1.21.7 Name

property Name: WideString

then name of the camera - can be changed in channel setup

Interface: [ICamera](#)

2.1.21.8 Used

property Used: WordBool

if the camera is set to Used or Unused in channel setup

Interface: [ICamera](#)  read-only

2122 | Channel

TChannel represents a DEWESoft® data channel

see [Channels](#) for more details

see also: [IPluginChannel](#)

2.1.22.1 AbsMax

property AbsMax: Double

the absolute maximum value of this channel for the current measurement

see also: [AbsMin](#)

Interface: [IChannel](#)

21222 AbsMin

property AbsMin: Double

the absolute minimum value of this channel for the current measurement

see also: [AbsMax](#)

Interface: [IChannel](#)

2.1.22.3 AddAsyncByteSample

```
procedure AddAsyncByteSample(Value: Byte; TimeStamp: Double);
```

AddAsyncByteSample adds an asynchronous data sample of the type Byte and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Byte	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.4 AddAsyncData

```
procedure AddAsyncData(Data: OleVariant; TimeStamp: Double);
```

AddAsyncData adds an asynchronous data sample of an arbitrary type and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Data	OleVariant	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.5 AddAsyncDoubleSample

```
procedure AddAsyncDoubleSample(Value: Double; TimeStamp: Double);
```

AddAsyncDoubleSample adds an asynchronous data sample of the type Double and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Double	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.6 AddAsyncInt64Sample

```
procedure AddAsyncInt64Sample(Value: Largeuint; TimeStamp: Double);
```

`AddAsyncInt64Sample` adds an asynchronous data sample of the type `Largeuint` and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: IChannel

Classifier	Name	Type	Description
	Value	Largeuint	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.7 AddAsyncIntegerSample

```
procedure AddAsyncIntegerSample(Value: Integer; TimeStamp: Double);
```

`AddAsyncIntegerSample` adds an asynchronous data sample of the type `Integer` and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: IChannel

Classifier	Name	Type	Description
	Value	Integer	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.8 AddAsyncShortintSample

```
procedure AddAsyncShortintSample(Value: Shortint; TimeStamp: Double);
```

`AddAsyncShortintSample` adds an asynchronous data sample of the type `Shortint` and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: IChannel

Classifier	Name	Type	Description
	Value	Shortint	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.9 AddAsyncSingleSample

```
procedure AddAsyncSingleSample(Value: Single; TimeStamp: Double);
```

AddAsyncSingleSample adds an asynchronous data sample of the type `Single` and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Single	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.10 AddAsyncSmallintSample

```
procedure AddAsyncSmallintSample(Value: Smallint; TimeStamp: Double);
```

AddAsyncSmallintSample adds an asynchronous data sample of the type `Smallint` and its corresponding timestamp to an asynchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Smallint	the value of the sample that will be added
	TimeStamp	Double	the timestamp of the sample to be added

2.1.22.11 AddAsyncString

```
procedure AddAsyncString(const Value: WideString; TimeStamp: Double);
```

AddAsyncString adds an asynchronous data sample of the type `WideString` and its corresponding timestamp to an asynchronous channel.

Can only be used for string channels (see [String Channels](#)); i.e. when you have called [SetAsStringChannel](#) before.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	
Classifier	Name	Type	Description										
const	Value	WideString	the string-value of the sample that will be added Note: If the string is longer than the <code>Size</code> specified when calling SetAsStringChannel , it will be truncated.										
	TimeStamp	Double	the timestamp of the sample to be added										

2.1.22.12 AddByteSample

```
procedure AddByteSample(Value: Byte);
```

`AddByteSample` adds a synchronous data sample of the type `Byte` to a synchronous channel.

see also: [How to: Write Data To Channels - Async](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Byte	the value of the sample that will be added

2.1.22.13 AddData

```
procedure AddData(Data: OleVariant);
```

`AddData` adds a synchronous data sample of an arbitrary type to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Data	OleVariant	the value of the sample that will be added

2.1.22.14 AddDoubleSample

```
procedure AddDoubleSample(Value: Double);
```

`AddDoubleSample` adds a synchronous data sample of the type `Double` to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Double	the value of the sample that will be added

2.1.22.15 AddIn64Sample

```
procedure AddIn64Sample(Value: Largeuint);
```

AddIn64Sample adds a synchronous data sample of the type Largeuint to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Largeuint	the value of the sample that will be added

2.1.22.16 AddIntegerSample

```
procedure AddIntegerSample(Value: Integer);
```

AddIntegerSample adds a synchronous data sample of the type Integer to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Integer	the value of the sample that will be added

2.1.22.17 AddIntegerSampleWithCalc

```
procedure AddIntegerSampleWithCalc(Value: Integer);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.22.18 AddShortintSample

```
procedure AddShortintSample(Value: Shortint);
```

AddShortintSample adds a synchronous data sample of the type Shortint to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Shortint	the value of the sample that will be added

2.1.22.19 AddShortintSampleWithCalc

```
procedure AddShortintSampleWithCalc(Value: Shortint);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Shortint	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.22.20 AddSingleSample

```
procedure AddSingleSample(Value: Single);
```

`AddSingleSample` adds a synchronous data sample of the type `Single` to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: IChannel

Classifier	Name	Type	Description
	Value	Single	the value of the sample that will be added

2.1.22.21 AddSingleSamples

```
procedure AddSingleSamples  
(Count: Integer; Data: OleVariant; Timestamps: OleVariant);
```

`AddSingleSamples` adds `Count` synchronous data samples of the type `Single` to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
	Count	Integer	the number of elements (Single values) in the Data array and in the Timestamps array									
	Data	OleVariant	the values of the samples that will be added - must be an array of Single values with exactly Count elements									
	Timestamps	OleVariant	must be an array of Double values with exactly Count elements - or NULL for synchronous channels									

2.1.22.22 AddSmallintSample

```
procedure AddSmallintSample(Value: Smallint);
```

AddSmallintSample adds a synchronous data sample of the type Smallint to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Smallint	the value of the sample that will be added

2.1.22.23 AddSmallintSampleWithCalc

```
procedure AddSmallintSampleWithCalc(Value: Smallint);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Smallint	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.22.24 AddWordSample

```
procedure AddWordSample(Value: Word);
```

AddWordSample adds a synchronous data sample of the type Word to a synchronous channel.

see also: [How to: Write Data To Channels - Sync](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Word	the value of the sample that will be added

2.1.22.25 ArrayChannel

property ArrayChannel: WordBool

`True` if this channel is an array channel, `False` if it is a standard channel.

see also: [How to: Mount Dewesoft Channels](#)

[ArrayInfo](#), [ArraySize](#)

Interface: [IChannel](#)  read/write

2.1.22.26 ArrayInfo

property **ArrayInfo:** **IArrayInfo**

provides access to the [IArrayInfo](#) interface. Only relevant if [ArrayChannel](#) is TRUE.

see also: [Array Channels](#), [ArrayChannel](#), [ArraySize](#), [IArrayInfo](#)

Interface: [IChannel](#)

2.1.22.27 ArraySize

property ArraySize: Integer

e.g. when you have an array with

see also: [ArrayChannel](#), [ArrayInfo](#)

2.1.22.28 Async

property Async: WordBool

`Async` is `True` when the channel is asynchronous. `False` for synchronous channels.

see also: [SetAsync](#), [Synchronism](#)

Interface: [IChannel](#)

2.1.22.29 BitCount

property BitCount: Integer

the number of bits used for one data sample of this channel

e.g. for 24-bit card: [Bytes](#) may be 4 (32 bits) and bit count is 24.

Interface: [IChannel](#)  read/write

2.1.22.30 Bytes

property Bytes: Integer

`Bytes` is the number of bytes used for one data sample of the specific channel. The number of bytes depends on the [DataType](#) of the channel.

see also: [Data Types](#)

Interface: [IChannel](#)  read-only

2.1.22.31 CalcDelay

property CalcDelay: Integer

the calculation delay in samples.

Synchronous channels

Synchronous channels must always add exactly the number of required samples (as specified by [IData.GetSamplesAcquired](#)) in every [OnGetData](#) cycle.

If it is not possible (yet) to add all samples to the channel (i.e. because it has not received the data from the device yet, or maybe it needs to gather several samples before it can do some calculation), then the `CalcDelay` of the channel must be set to the number of samples that are missing (i.e. have not been added yet).

Asynchronous channels

For asynchronous channels the value of the `CalcDelay` needs not be set. Since DEWEsoft® knows also the time-stamps of the samples in asynchronous channels, it can calculate the `CalcDelay` automatically.

see also: [Calculation Delay](#), [IData.MaxCalcDelay](#)

Interface: [IChannel](#)  read/write

2.1.22.32 CalcSRDiv

property CalcSRDiv: Integer

this is the sample rate divider that is currently used (e.g. in channel setup it is always 1, during measurement the value of SRDiv is used).

Interface: [IChannel](#)  read-only

2.1.22.33 ChNo

property ChNo: WideString

This is set programmatically and cannot be changed by user (compare to: [Name](#), [Description](#))

Default example values:

analogue channel: 'AI 0'

math channel: '*Math 0 (Formula)*'

DS-NET plugin channel: '*Channel 0*'

Interface: [IChannel](#)  read-only

2.1.22.34 ChangeThreshold

property ChangeThreshold: Single

only useful used for tabular displays with the display option '*Print on value change only*'. Only if the value of the given channel (drop-down box: *Channel name*) changes for more than the specified threshold, the value will be printed to the tabular display.

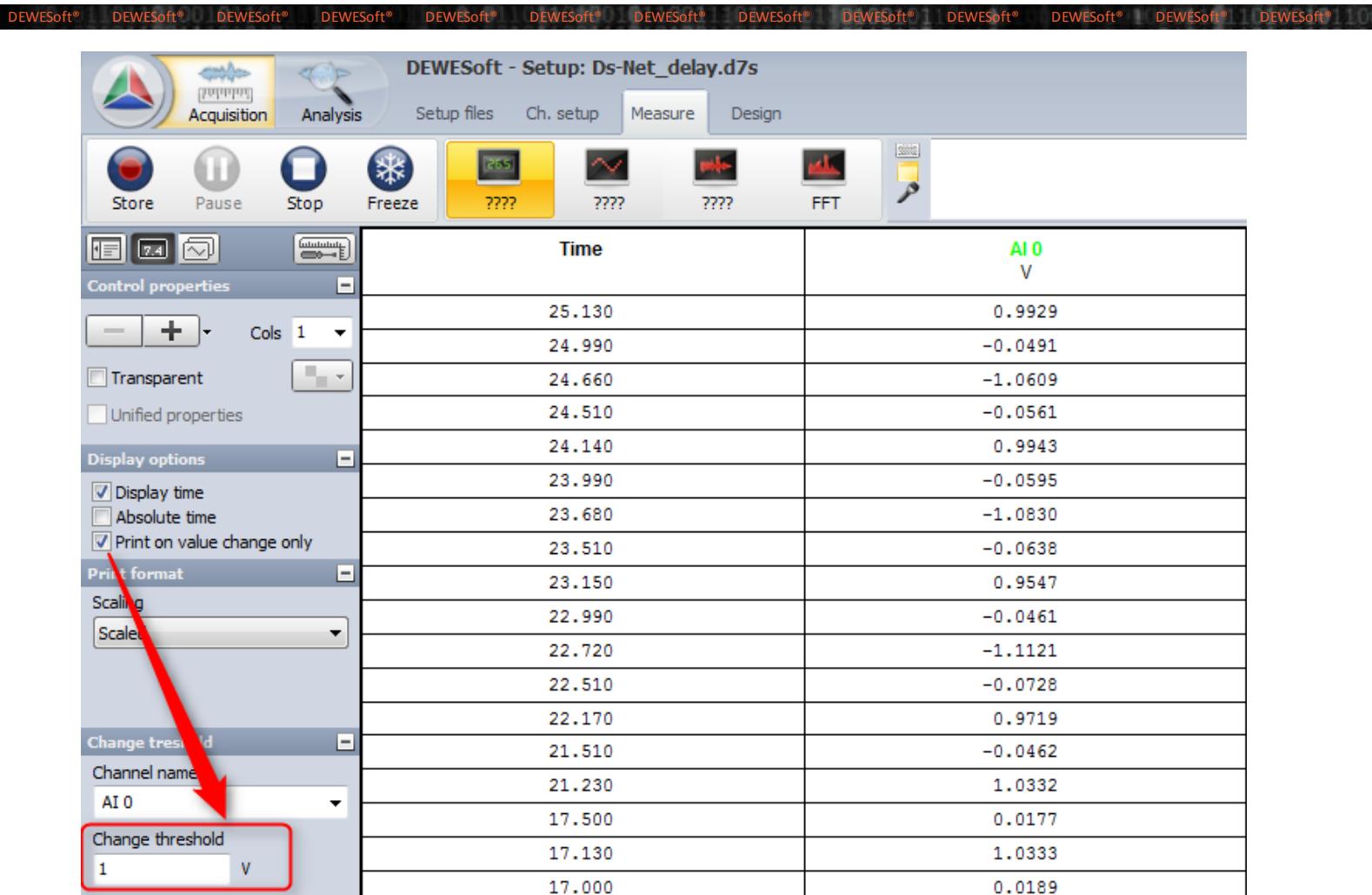


Illustration 167: ChangeThreshold

Interface: [IChannel](#)

read/write

2.1.22.35 ControlChannelFlags

property ControlChannelFlags: Integer

additional information about the control channel

see: [ControlChFlags](#)Interface: [IChannel](#)

read/write

2.1.22.36 ControlChannelState

property ControlChannelState: Integer

0.. control channel is enabled

1.. control channel is disabled (visual control disabled - cannot be changed)

Interface: [IChannel](#)



2.1.22.37 CreateConnection

```
function CreateConnection(): IChannelConnection;
```

`CreateConnection` creates a connection to this channel in order to access its data.

see [IChannelConnection](#) for detailed information

Interface: [IChannel](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	IChannelConnection	the connection to this channel in order to access its data.

2.1.22.38 DBBufSize

property DBBufSize: Integer

`DBBufSize` is the size of the direct buffer; see [The Buffer Structure](#) for more information.

see also: [The Buffer Structure](#), [DBDataSize](#), [DBPos](#)

Interface: [IChannel](#)



212239 DBDataSize

property DBDataSize: Integer;

`DBDataSize` is the size of valid data inside of the direct buffer; see [The Buffer Structure](#) for more information.

see also: [The Buffer Structure](#), [DBBufSize](#), [DBPos](#)

Interface: [IChannel](#)



212240 DBPos

property DBPos: Integer;

`DBPos` is the next position within the direct buffer to be written to.

see also: [The Buffer Structure](#), [DBBufSize](#), [DBDataSize](#)

Interface: [IChannel](#)

2.1.22.41 DBTimeStamp

property DBTimeStamp[Index: Integer]: Double

DBTimeStamp [Index] is the timestamp data of asynchronous data in the direct buffer.

Index is a value from 0...[DBBufferSize](#)-1

see also: [The Buffer Structure](#), [DBBufferSize](#), [DBValues](#), [DBValuesDouble](#)

Interface: [IChannel](#)

2.1.22.42 DBValues

property DBValues[Index: Integer]: Single

DBValues [Index] are the values of the (asynchronous or synchronous or Single value) data in the direct buffer in single precision.

Index is a value from 0...[DBBufferSize](#)-1

see also: [The Buffer Structure](#), [DBBufferSize](#), [DBTimeStamp](#), [DBValuesDouble](#)

Interface: [IChannel](#)

2.1.22.43 DBValuesDouble

property DBValuesDouble[Index: Integer]: Double

DBValues [Index] are the values of the asynchronous data in the direct buffer in double precision.

Index is a value from 0...[DBBufferSize](#)-1

see also: [The Buffer Structure](#), [DBBufferSize](#), [DBTimeStamp](#), [DBValues](#)

Interface: [IChannel](#)

2.1.22.44 DStartDataAvail

property DStartDataAvail: Integer

used in analyze mode. It describes the amount of data (from the data file) which is currently loaded in memory.

Via DCOM you can only access the data that is currently in memory.

To get the absolute number of samples relative to the start of the measurement file:

* [IData_Samples](#)

(there is no Dir for the start)

see also: [DStopDataAvail](#)

Interface: [IChannel](#)



2.1.22.45 DStopDataAvail

property DStopDataAvail: Integer

used in analyze mode. It describes the amount of data (from the data file) which is currently loaded in memory.

Via DCOM you can only access the data that is currently in memory.

To get the absolute number of samples relative to the start of the measurement file:

* [IData_Samples](#)

+ [DStopDataAvailDir](#)

see also: [DStartDataAvail](#), [DStopDataAvailDir](#)

Interface: [IChannel](#)



2.1.22.46 DStopDataAvailDir

property DStopDataAvailDir: Integer

used in analyze mode. It describes the amount of data (from the data file) which is currently loaded in memory.

Via DCOM you can only access the data that is currently in memory.

To get the absolute number of samples relative to the start of the measurement file:

* [IData_Samples](#)

+ [DStopDataAvailDir](#)

see also: [DStartDataAvail](#), [DStopDataAvail](#)

Interface: [IChannel](#)



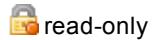
2.1.22.47 DataType

property DataType: Integer

DataType is the data type of data samples in this channel. See [Data Types](#) for more details.

see [Data Types](#), [SetDataType](#) [SetAsStringChannel](#)

Interface: [IChannel](#)

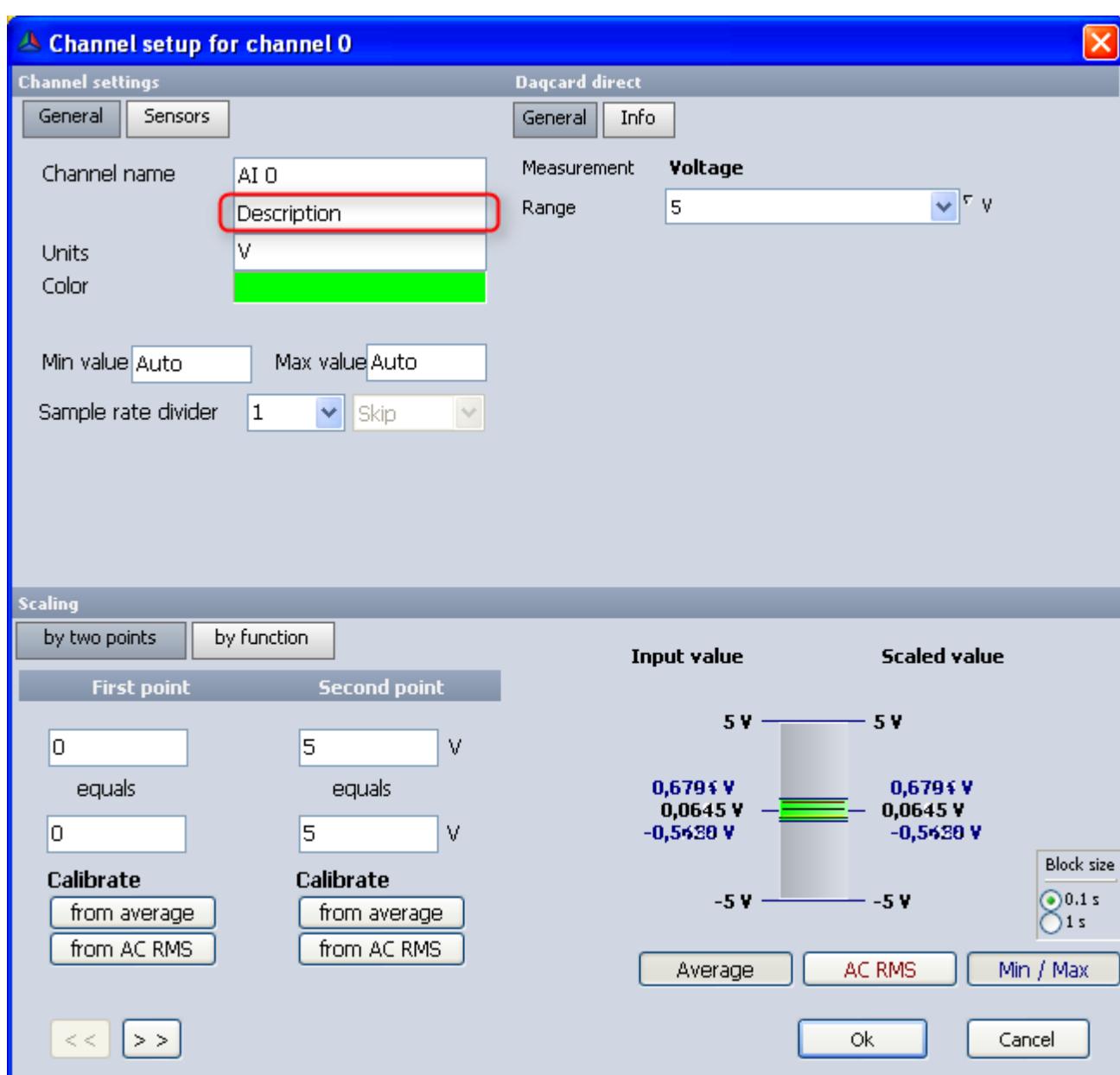


2.1.22.48 Description

property Description: WideString

Description is a text comment to a channel. Description is equal to [Measurement](#).

You can enter the description of a channel in the channel setup dialog (see example screenshot for an analogue channel in the image below).



see also: [Measurement](#), [Name](#)

Interface: [IChannel](#)

2.1.22.49 DiscreteList

property DiscreteList: IDiscreteList

A list of discrete display values. see [IDiscreteList](#) for details.

see also: [IDiscreteList](#)

Interface: [IChannel](#)

2.1.22.50 ExpectedAsyncRate

property ExpectedAsyncRate: Single

the expected sample rate is only relevant for asynchronous channels.

Make sure to set this value correctly, because the size of the channel's buffer is calculated based on this value. If you set the value too low, you may loose data.

Interface: [IChannel](#)  read/write

2.1.22.51 ExportOrder

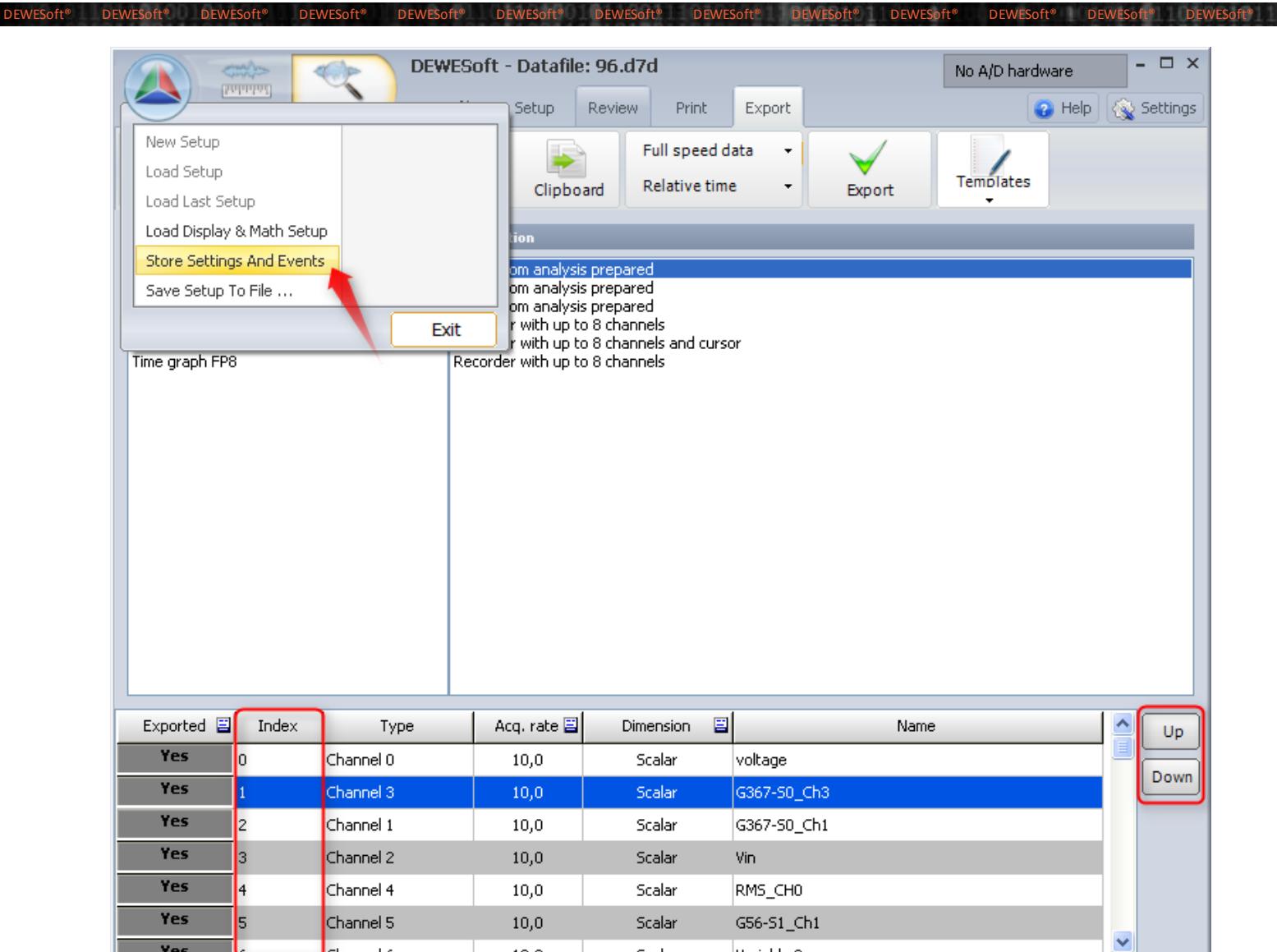
property ExportOrder: Integer

`ExportOrder` allows specifying the order in which the channels are exported. The default value is `-1` meaning not ordered.

Channels having the default `ExportOrder` set to `-1` will always be listed after any channel having a customized `ExportOrder`.

On the [Export](#) tab-sheet (in *Analyse* mode) you can change the export order (see buttons [Up/Down](#) in the screenshot below).

When you click [Store Settings and Events](#), these settings can be stored in the current data file.



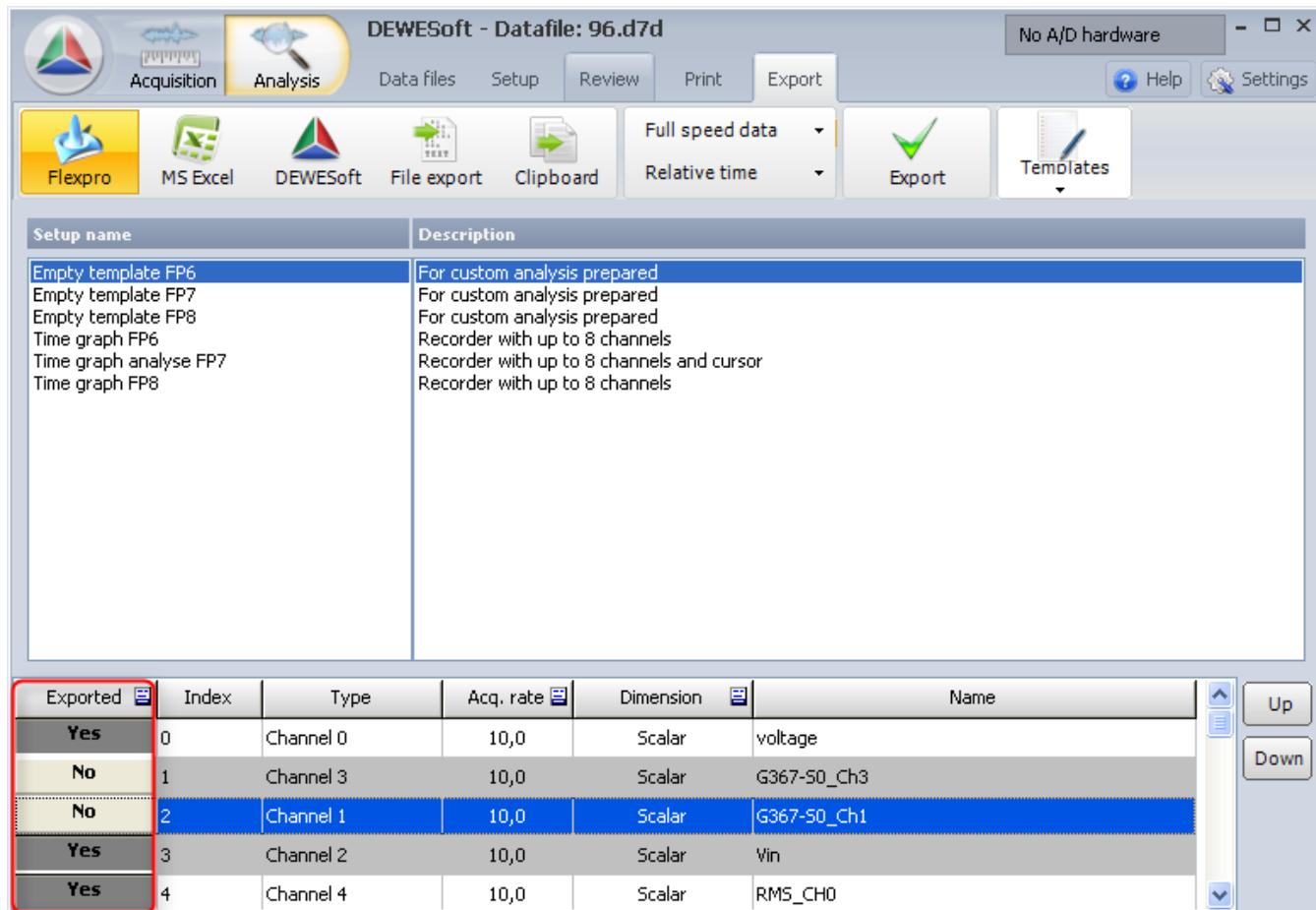
see also: [Exported](#)

Interface: [IChannel](#) read/write

2.1.22.52 Exported

property Exported: WordBool

If Exported is TRUE the channel will be exported, otherwise not.



see also: [ExportOrder](#)

Interface: [IChannel](#)

read/write

2.1.22.53 FastCalc

procedure FastCalc();

Under development! This feature is currently under development and should not be used yet.

Interface: [IChannel](#)

2.1.22.54 FastCalcInt32

```
procedure FastCalcInt32(Min: Integer; Max: Integer; Ave: Double; Rms: Double);
```

Under development! This feature is currently under development and should not be used yet.

Interface: IChannel

Classifier	Name	Type	Description
	Min	Integer	
	Max	Integer	
	Ave	Double	
	Rms	Double	

2.1.22.55 FirstIBLevel

```
property FirstIBLevel: Integer
```

the first level of intermediate buffers: this may be higher than 0, e.g. for slow async channels

see also: [The Buffer Structure](#), [IBBufSize](#), [IBDataSize](#), [IBDataSizeEx](#), [IBPos](#), [IBPosEx](#), [FirstIBLevel](#)

Interface: [IChannel](#)



212256 FirstX

property Firstx: Double

This is a project specific feature

see also: [SecondX](#)



212257 GetChannelSetup

```
procedure GetChannelSetup(out Data: Olevariant);
```

Returns the channel setup for this channel as an array of bytes

see also: XML_Setup_SetChannelSetup

Interface: [IChannel](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
out	Data	OleVariant	the channel setup for this channel as an array of bytes. For details on the XML format see: XML Setup .									

Example output (note: the output has been formatted to be readable in this documentation - the original string does not include any characters for indentation):

```
<Channel>
  <MeasuredValue>VOLTAGE</MeasuredValue>
  <Range>5 V</Range>
  <OutputChannel>
    <DisplayColor>#00FF00</DisplayColor>
    <Name>AI 0</Name>
    <Unit>V</Unit>
    <Used>True</Used>
    <Index>1;0</Index>
    <SIUnit>
      <Label>V</Label>
      <Length>2</Length>
      <Mass>1</Mass>
      <Time>-3</Time>
      <Current>-1</Current>
      <Scale>1</Scale>
      <Ref>1</Ref>
      <LogRef>1</LogRef>
    </SIUnit>
  </OutputChannel>
</Channel>
```

Delphi example code to convert the OleVariant to String:

```
function ConvertChSetup(VariantChSetup: OleVariant): String;
var
  ChSetupString: String;
  ByteArray: array of byte;
begin
  ByteArray := VariantChSetup;
  SetString(Result, PAnsiChar(@ByteArray[0]), Length(ByteArray));
end;
```

2.1.22.58 GetDBAddress

```
function GetDBAddress(): Integer;
```

Returns the pointer-address of direct buffer.

only usefull for plugins - faster than calling functions through DCOM

see also: [The Buffer Structure](#), [DBBufSize](#), [DBDataSize](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
-	RESULT	Integer	the index of the next position in direct buffer to be written to

2.1.22.59 GetIBValues

```
procedure GetIBValues
(Pos: Integer; out Min: Single; out Max: Single; out Ave: Single; out Rms: Single;);
```

GetIBValues reads the intermediate buffer values Min, Max, Ave and Rms at the position specified by Pos. The property [IBPos](#) can be read to determine the current position of the intermediate buffer.

This procedure is implemented for automation applications developed in LabVIEW because, due to its data type, the property [IBValues](#) is not accessible by LabVIEW.

In programming environments other than LabVIEW the property [IBValues](#) can be used instead of GetIBValues.

see also: [The Buffer Structure](#), [IBValues](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Pos	Integer	the position within the intermediate buffer to read from note that the position is relative to the buffer data (see also The Buffer Structure - especially Example for buffer index): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize
out	Min	Single	minimum value
out	Max	Single	maximum value
out	Ave	Single	average value
out	Rms	Single	RMS (root mean square) value

2.1.22.60 GetIndex1

```
procedure GetIndex1
(out IndexLevel: Integer; out I1: Integer; out I2: Integer; out I3: Integer; out I4 : Integer; out I5: Integer);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

USE [IChannel.Get_IndexEx](#) INSTEAD - see also [Channel Index](#).

Similar to reading the property [Index](#). The property Index cannot be read by LabVIEW, so this procedure must be used instead.

see also: [Index](#), [T_ChIndex](#)

Interface: [IChannel](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
out	IndexLevel	Integer	defines the number of used index levels for specifying a channel - this depends on the type of channel (for a detailed description see T_ChIndex), e.g. if IndexLevel=1 then only I1 will be used (I2, etc. will be ignored)									
out	I1	Integer	the 1st index value									
out	I2	Integer	the 2nd index value									
out	I3	Integer	the 3rd index value									
out	I4	Integer	the 4th index value									
out	I5	Integer	the 5th index value									

2.1.22.61 GetOfflineStatus

```
function GetOfflineStatus(): Integer;
```

returns the current offline status of the channel.

Possible values:

0...online channel

1...offline channel

2...offline calculated (not store)

3...calculated and stored

11...calculatig offline channel in analyse

Interface: [IChannel](#)

Classifier	Name	Type	Description
-	RESULT	Integer	the current offline status of the channel

2.1.22.62 GetRBValues

```
procedure GetRBValues
(Pos: Integer; out Min: Single; out Max: Single; out Ave: Single; out Rms: Single);
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

use [IBValuesEx](#) instead

Interface: [IChannel](#)

2.1.22.63 GetScaledData

```
function GetScaledData(): OleVariant;
```

Returns all available scaled data (see [Scaling](#)) from the direct buffer (see [The Buffer Structure](#)).

The return value is an OleVariant containing an array of Single values:

Double values which are higher/lower than the maximum/minimum Single range will be truncated to the maximum/minimum of type Single.

For Complex numbers only the real part is used.

During measurement, the size of the array returned will increase with the number of acquired samples (`DBDataSize`).

When reloading acquired data from a DEWEsoft® data file in conjunction with [ILoadEngine.ReloadBlock](#), the returned array will have the size of [IData.Samples](#).

see also: [How To Read Data From Channels](#), [Scaling](#), [The Buffer Structure](#), [Data Types](#), [Scale](#), [Offset](#), [GetScaledDataEx](#), [GetScaledDataEx1](#), [GetTSData](#), [GetUnscaledData](#)

Interface: IChannel

Classifier	Name	Type	Description
-	<i>RESULT</i>	OleVariant	all available scaled data (see Scaling) from the direct buffer

2.1.22.64 GetScaledDataEx

```
function GetScaledDataEx(Start: Integer; Count: Integer): OleVariant;
```

Returns scaled data (see [Scaling](#)) from the direct buffer (see [The Buffer Structure](#)).

`GetScaledDataEx` is an extended version of [GetScaledData](#), which allows you to specify the start position (`Start`) and the desired amount of data (`Count`).

The return value is an OleVariant containing an array of Single values:

Double values which are higher/lower than the maximum/minimum Single range will be truncated to the maximum/minimum of type Single.

For Complex numbers only the real part is used.

During measurement, the size of the array returned will increase with the number of acquired samples (`DBDataSize`).

When reloading acquired data from a DEWEsoft® data file in conjunction with [ILoadEngine.ReloadBlock](#), the returned array will have the size of [IData.Samples](#).

see also: [How To Read Data From Channels](#), [Scaling](#), [The Buffer Structure](#), [Data Types](#), [Scale](#), [Offset](#), [GetScaledData](#), [GetScaledDataEx](#), [GetScaledDataEx1](#), [GetUnscaledData](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Start	Integer	the position within the direct buffer from where to start retrieving data note that the position is relative to the buffer data (see also The Buffer Structure - especially Example for buffer index): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize
	Count	Integer	the desired amount of data
-	RESULT	OleVariant	the scaled data (see Scaling) from the direct buffer

2.1.22.65 GetScaledDataEx1

```
procedure GetScaledDataEx1(Start: Integer; Count: Integer; out Data: OleVariant);
```

Same as [GetScaledDataEx](#), but implemented as a procedure instead of a function.

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Start	Integer	defines the position within the direct buffer from where to start retrieving data note that the position is relative to the buffer data (see also The Buffer Structure - especially Example for buffer index): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize
	Count	Integer	the desired amount of data
out	Data	OleVariant	the scaled data (see Scaling) from the direct buffer

2.1.22.66 GetTSAddress

```
function GetTSAddress(): Integer;
```

Returns the pointer-address of timestamp buffer.

only usefull for plugins - faster than calling functions through DCOM

see also: [GetDBAddress](#), [IncDBSamples](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
-	RESULT	Integer	the pointer-address of timestamp buffer.

2.1.22.67 GetTSData

```
function GetTSDData(): OleVariant;
```

`GetTSDData` retrieves the timestamp data of asynchronous channels (see [Synchronism](#)).

This function should be used in conjunction with [GetScaledData](#) for asynchronous channels.

The returned `OleVariant` contains an array of timestamp values of the data type `Double` (see [DateTime](#)). The size of the array will be equal to the size of the array returned by `GetScaledData`.

see also: [The Buffer Structure](#), [How To Read Data From Channels](#), [Data Types](#), [GetScaledData](#), [GetUnscaledData](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	OleVariant	an array of timestamp values of the data type Double (see DateTime)

2.1.22.68 GetTSDDataEx

```
function GetTSDDataEx(Start: Integer; Count: Integer): OleVariant;
```

The structure of the model is shown in Figure 1. The main components of the model are the state-space, the state transition function, and the observation function.

This file is available from the [JSTOR Digital Library](#).

see also: [The Buffer Structure](#), [Data Types](#), [GetTSBData](#)

2.1.22.69 GetTSDDataEx1

```
procedure GetTSDDataEx1(Start: Integer; Count: Integer; out Data: OleVariant);
```

Same as [GetTSDDataEx](#), but implemented as a procedure instead of a function.

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Start	Integer	the position within the direct buffer from where to start retrieving data note that the position is relative to the buffer data (see also The Buffer Structure - especially Example for buffer index): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize
	Count	Integer	the desired amount of timestamps
out	Data	OleVariant	timestamp data of asynchronous channels

2.1.22.70 GetUnscaledData

```
function GetUnscaledData(): OleVariant;
```

Returns all available unscaled data (see [Scaling](#)) from the direct buffer (see [The Buffer Structure](#)).

The return value is an `OleVariant` containing an array of data points. The type of the data points in the array is dependant on the channels [DataType](#).

During measurement, the size of the array returned will increase with the number of acquired samples ([DBDataSize](#)).

When reloading acquired data from a DEWEsoft® data file in conjunction with [ILoadEngine.ReloadBlock](#), the returned array will have the size of [IData.Samples](#).

see also: [How To Read Data From Channels](#), [Scaling](#), [The Buffer Structure](#), [Data Types](#), [GetUnscaledDataEx](#), [GetUnscaledDataEx1](#), [GetScaledDataEx](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
-	RESULT	OleVariant	all available unscaled data (see Scaling) from the direct buffer (see The Buffer Structure)

2.1.22.71 GetUnscaledDataEx

```
function GetUnscaledDataEx(Start: Integer; Count: Integer): OleVariant;
```

Returns unscaled data (see [Scaling](#)) from the direct buffer (see [The Buffer Structure](#)).

`GetUnscaledDataEx` is an extended version of [GetUnscaledData](#), which allows you to specify the start position (`Start`) and the desired amount of data (`Count`).

The return value is an `OleVariant` containing an array of data points. The type of the data points in the array is dependant on the channels `DataType`.

During measurement, the size of the array returned will increase with the number of acquired samples ([DBDataSize](#)).

When reloading acquired data from a DEWEsoft® data file in conjunction with [ILoadEngine.ReloadBlock](#), the returned array will have the size of `IData.Samples`.

see also: [Scaling](#), [The Buffer Structure](#), [Data Types](#), [GetUnscaledData](#), [GetUnscaledDataEx1](#), [GetScaledDataEx1](#)

Interface: IChannel

Classifier	Name	Type	Description
	Start	Integer	<p>the position within the direct buffer from where to start retrieving data</p> <p>note that the position is relative to the buffer data (see also The Buffer Structure - especially Example for buffer index): before you try to read data, make sure, that there is already enough data to read: i.e. check the <code>DataSize</code></p>
	Count	Integer	the desired amount of timestamps
-	<i>RESULT</i>	OleVariant	the unscaled data (see Scaling) from the direct buffer (see The Buffer Structure)

2.1.22.72 GetUnscaledDataEx1

```
procedure GetUnscaledDataEx1(Start: Integer; Count: Integer; out Data: OleVariant);
```

Same as [GetUnscaledDataEx](#), but implemented as a procedure instead of a function.

Interface: IChannel

Classifier	Name	Type	Description
	Start	Integer	<p>the position within the direct buffer from where to start retrieving data</p> <p>note that the position is relative to the buffer data (see also The Buffer Structure - especially Example for buffer index): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize</p>
	Count	Integer	the desired amount of timestamps
out	Data	OleVariant	the unscaled data (see Scaling) from the direct buffer (see The Buffer Structure)

2.1.22.73 GetValueAtAbsPos

```
function GetValueAtAbsPos  
(Pos: Integer; var SeekPos: Integer; Interpolate: WordBool): Single;
```

Note: unfortunately the name of this function is misleading: actually it should be called `GetRelValue()`, because it will return a value relative to the given buffer index.

This function will return the value at the position (related to the direct buffer - see also [The Buffer Structure](#)) or the

interpolated value.

If it is an asynchronous channel you can choose if you want to get the interpolated value or the nearest matching value, by setting the `Interpolate` variable to TRUE or FALSE.

see also: [GetValueAtAbsPosDouble](#), [Data Types](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Pos	Integer	the position of the value that should be read -1...the newest data in the buffer (<code>BufferSize</code> must of course be ≥ 1) - <code>BufferSize</code> ...the oldest data in the buffer
var	SeekPos	Integer	not used if it is a Single Value Channel or a synchronous channel (see Synchronism)
	Interpolate	WordBool	only relevant for asynchronous channels (see Synchronism): TRUE...returns an interpolated value FALSE...returns the nearest matching value
-	RESULT	Single	the value at the absolute position (related to the direct buffer - see also The Buffer Structure) or the interpolated value.

2.1.22.74 GetValueAtAbsPosDouble

```
function GetValueAtAbsPosDouble  
(Pos: Integer; var SeekPos: Integer; Interpolate: WordBool): Double;
```

the same as [GetValueAtAbsPos](#) but returns a value of data type Double.

see also: [GetValueAtAbsPos](#), [Data Types](#)

Interface: [IChannel](#)

2.1.22.75 Group

```
property Group: IChannelGroup
```

Group the channel group that this channel belongs to: see [IData.Groups](#) for details.

see also: [IChannelGroup](#), [IData.Groups](#)

Interface: [IChannel](#)



2.1.22.76 IBBufSize

property IBBufSize: Integer

`IBBufSize` is the size of the intermediate buffer (for all levels the same)

see also: [The Buffer Structure](#), [IBDataS](#)

2.1.22.77 IBDatasize

property IBDatasize: Integer

`IBBufSize` is the size of the data for the first level of the intermediate buffers.

see also: [The Buffer Structure](#), [IBufSize](#)

2-1-22-78 IBDataSizeEx

property IBDatasizeEx[Level: Integer]: Integer

`TBDataSizeEx` is the size of the data for the intermediate buffer at level `Level`.

`TBDataSizeEx[FirstTBLlevel]` is the same as `TBDataSize`

see also: [The Buffer Structure](#), [IBBufSize](#), [IBDataSize](#), [IBPos](#), [IBPosEx](#), [FirstIBIlevel](#), [IData](#), [IBIlevels](#)

Interface: [IChannel](#)

2-1-22-79 IBPos

property IBPos: Integer

`IBPos` is the next position for the intermediate buffer at level `Level`, where data should be written to.

see also: [The Buffer Structure](#), [IBBufSize](#), [IBDataSize](#), [IBDataSizeEx](#), [IBPosEx](#)

Interface: [IChannel](#)

2.1.22.80 IBPosEx

```
property IBPosEx[Level: Integer]: Integer
```

IBPosEx is the next position for the intermediate buffer at level Level where data should be written to.

IBPosEx[FirstIBLevel] is the same as [IBPos](#).

see also: [The Buffer Structure](#), [IBBufSize](#), [IBDataSize](#), [IBDataSizeEx](#), [IBPos](#), [FirstIBLevel](#), [IData.IBLevels](#)

Interface: [IChannel](#)



2.1.22.81 IBValues

```
property IBValues[Index: Integer]: T\_ReducedRec
```

IBValues [Index] is a record of the first level intermediate buffer (see [The Buffer Structure](#)) values at the given Index.

note that the position is relative to the buffer data (see also [The Buffer Structure](#) - especially [Example for buffer index](#)): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize. The returned record contains the values Min, Max, Ave and RMS.

The property IBValues does not work in conjunction with LabVIEW. It is recommended to use the procedure [GetIBValues](#) instead.

Note: the intermediate buffer is always synchronous - there are no timestamps to read

see also: [T_ReducedRec](#), [The Buffer Structure](#), [IBBufSize](#), [IBDataSize](#), [IBPos](#)

Interface: [IChannel](#)



2.1.22.82 IBValuesEx

```
property IBValuesEx[Level: Integer; Index: Integer]: T\_ReducedRec
```

IBValues [Index] is a record of the intermediate buffer at level Level (see [The Buffer Structure](#)) values at the given Index.

note that the position is relative to the buffer data (see also [The Buffer Structure](#) - especially [Example for buffer index](#)): before you try to read data, make sure, that there is already enough data to read: i.e. check the DataSize. The returned record contains the values Min, Max, Ave and RMS.

IBValuesEx[FirstIBLevel, Index] is the same as [IBValues](#)[Index].

see also: [T_ReducedRec](#), [The Buffer Structure](#), [IBValues](#), [IBBufSize](#), [IBDataSize](#), [IBPosEx](#)

Interface: [IChannel](#)



2.1.22.83 IncDBSamples

```
procedure IncDBSamples(Count: Integer);
```

To increment the position of the direct buffer [DBPos](#) (and may increase [DBDataSize](#)).

You must only call this after you have added data directly to the direct buffer via [GetDBAddress](#) (and maybe [GetTSAAddress](#))

see also: [DBPos](#), [DBDataSize](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Count	Integer	the amount of samples the direct buffer position will be increased.

2.1.22.84 Index

```
property Index: T\_ChIndex
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

USE [IChannel.IndexEx](#) INSTEAD - see also [Channel Index](#).

Index is a record of index values that uniquely identifies a channel.

The property **Index** is not available in LabVIEW. It is recommended to use [GetIndex1](#) instead.

see also: [T_ChIndex](#), [GetIndex1](#)

Interface: [IChannel](#)



2.1.22.85 IndexEx

```
property IndexEx: OleVariant
```

Returns the channel index (see [Channel Index](#)). This is an array of type [Integer](#).

In Delphi you can use the `Low()` and `High()` functions to get the boundaries of the array.

Interface: [IChannel](#)



2.1.22.86 IsControlChannel

property IsControlChannel: WordBool

TRUE if this channel is a control channel (see [Control Channels](#)), FALSE otherwise.

Interface: [IChannel](#)  read/write

2.1.22.87 IsSingleValue

property IsSingleValue: WordBool

TRUE if this channel is a single value channel (see [Single Value Channel](#)), FALSE otherwise.

Interface: [IChannel](#)  read-only

2.1.22.88 LogicalIndex

property LogicalIndex: Largeuint

stored to setup - but not used by DEWESoft® - free to be used by plugins

Interface: [IChannel](#)  read/write

2.1.22.89 LogicalName

property LogicalName: WideString

stored to setup - but not used by DEWESoft® - free to be used by plugins

Interface: [IChannel](#)  read/write

2.1.22.90 MType

property MType: Integer

the MType defines how to interpret the data (how to convert the numbers to string) (see also: [DataType](#)):

- 0...general
- 1...GPS_X
- 2...GPS_Y
- 3...GPS direction
- 4...Absolute time

5...character: see [String Channels](#), [SetAsStringChannel](#)

6 ...relativ time

7 ...ip number

8 ...variable array

9...MAC address

Interface: IChannel

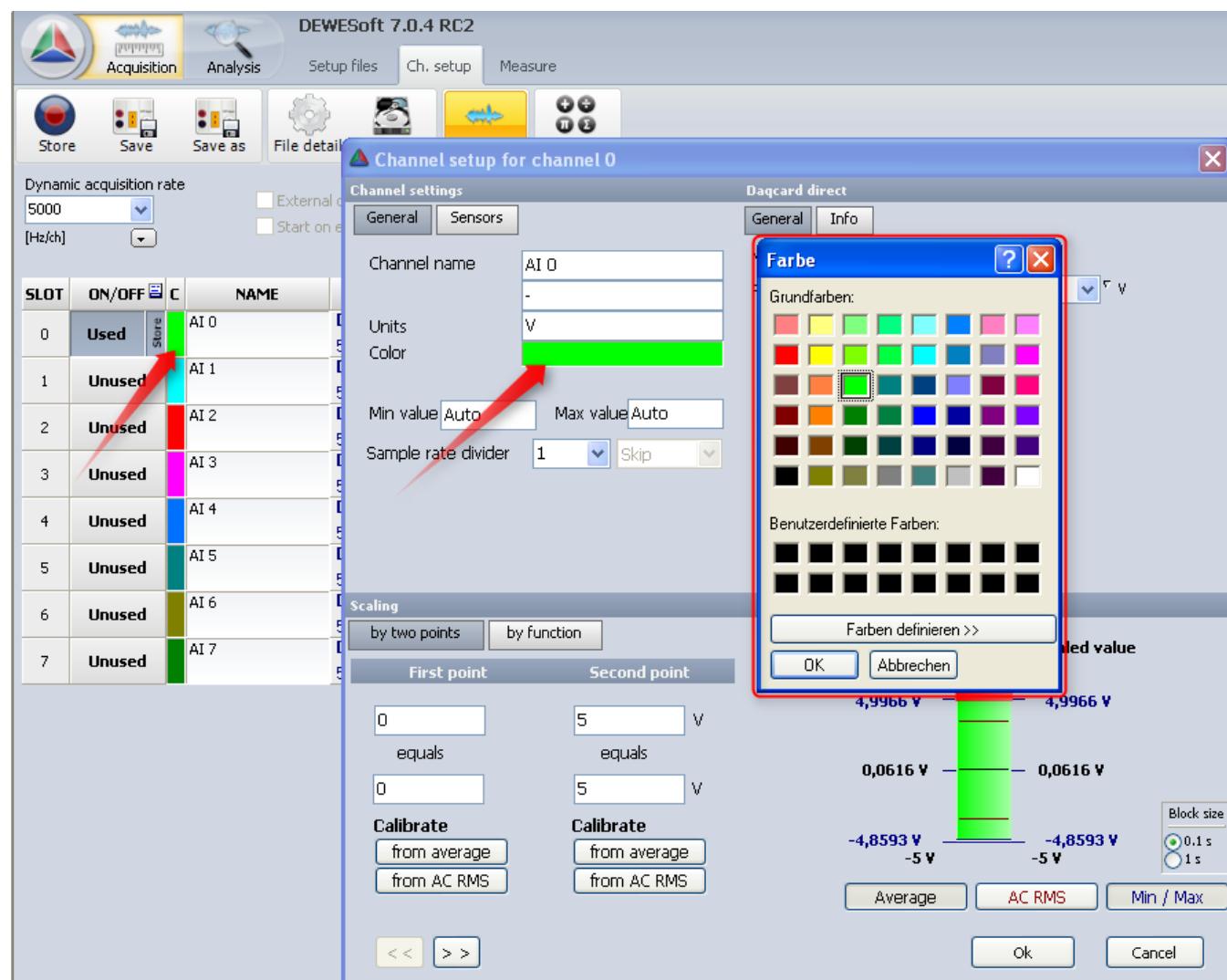


2.1.22.91 MainDisplayColor

property MainDisplayColor: Integer

`MainDisplayColor` specifies the colour of a channel (see also [ColourCodes](#)).

The default colour can be changed in the channel setup by clicking on the colour indicator in the channel setup grid or in the channel setup dialogue:



Interface: [IChannel](#)

2.1.22.92 Measurement

property Measurement: WideString

The description of the channel. Same as [Description](#).

see also: [Description](#), [Name](#)

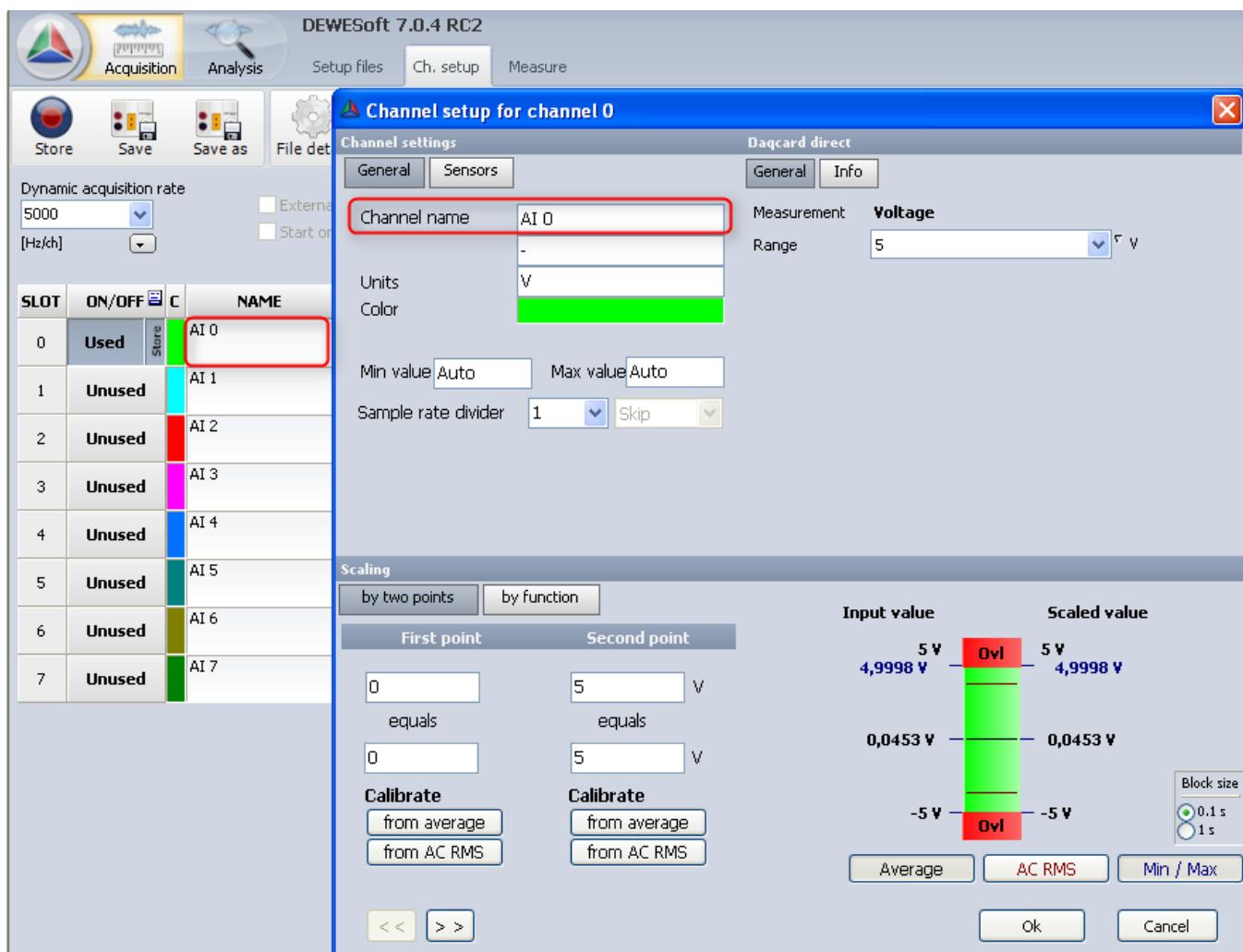
Interface: [IChannel](#)

2.1.22.93 Name

property Name: WideString

Name is the name of a channel.

The name can be changed in the channel setup grid or in the channel setup dialogue:



see also: [Description](#), [Measurement](#)

Interface: [IChannel](#)

2.1.22.94 Offset

property Offset: Double

The offset for scaling the channel channel: see [Scaling](#) for details.

see also: [Scaling](#), [Scale](#), [GetScaledData](#), [GetScaledDataEx](#), [GetScaledDataEx1](#)

Interface: **IChannel**

2.1.22.95 RBBufSize

property RBBufSize: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`BBBbufSize` is the size of the reduced buffer.

see also: [The Buffer Structure](#), [RBDataSize](#), [RBPos](#), [RBValues](#)

Interface: [IChannel](#)

212296 RBDataSize

property BBDataSize: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`RBDataSize` is the size of the data within the reduced buffer.

see also: The Buffer Structure, RBBufSize, RBPos, RBValues

Interface: [IChannel](#)

212297 RBPos

property BBPos: Integer;

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`RBPos` is the next position in the reduced buffer to be written to.

see also: [The Buffer Structure](#), [RBBufSize](#), [RBBufSize](#), [RBValues](#)

Interface: [IChannel](#) read-only

2.1.22.98 RBValues

property RBValues[Index: Integer]: T ReducedRec

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

use [IBValuesEx](#) instead

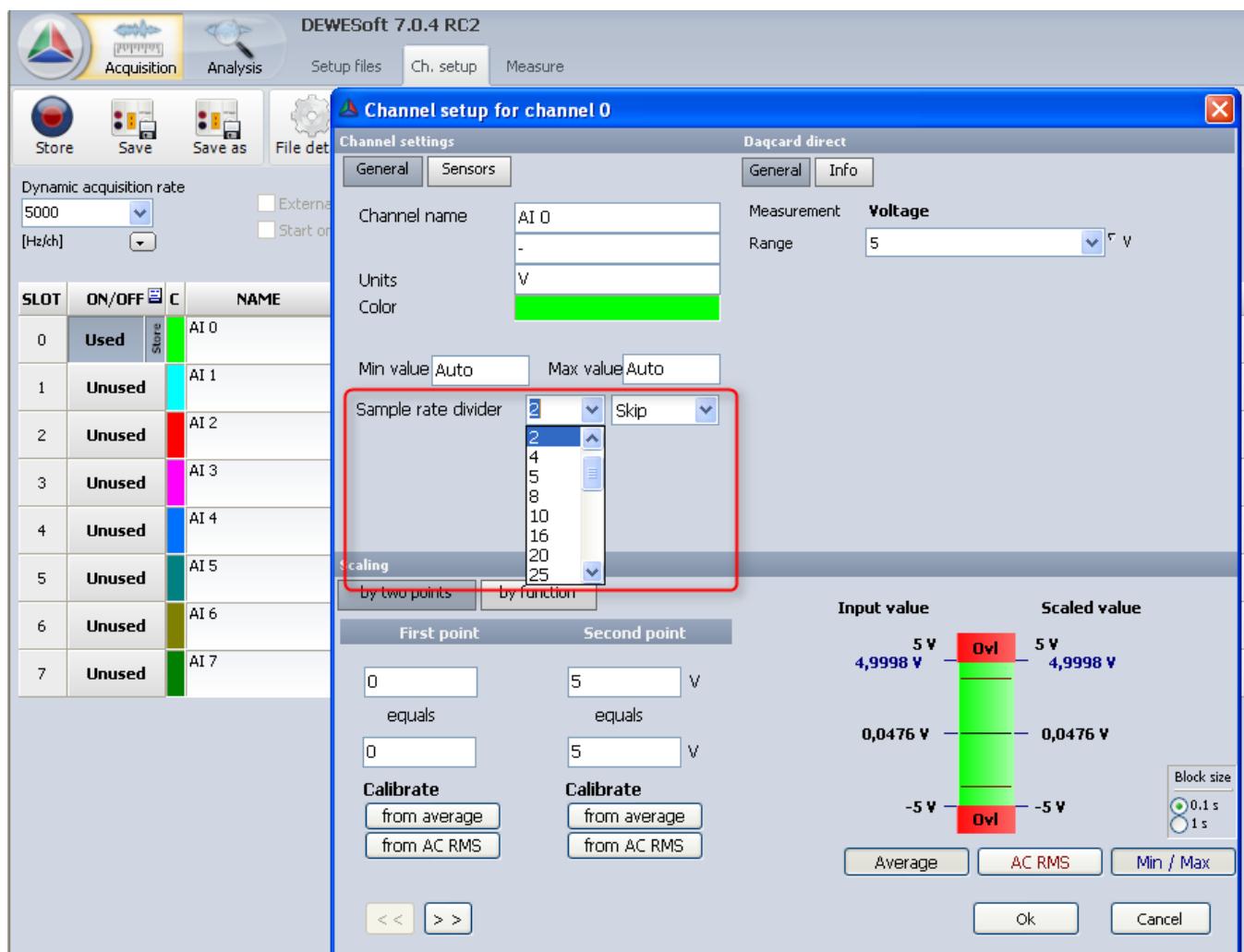
see also: [T ReducedRec](#), [The Buffer Structure](#), [GetRBValues](#), [RBBufSize](#), [RBDataSize](#), [RBPos](#)

Interface: [IChannel](#)

2.1.22.99 SRD iv

property SRDiv: Integer

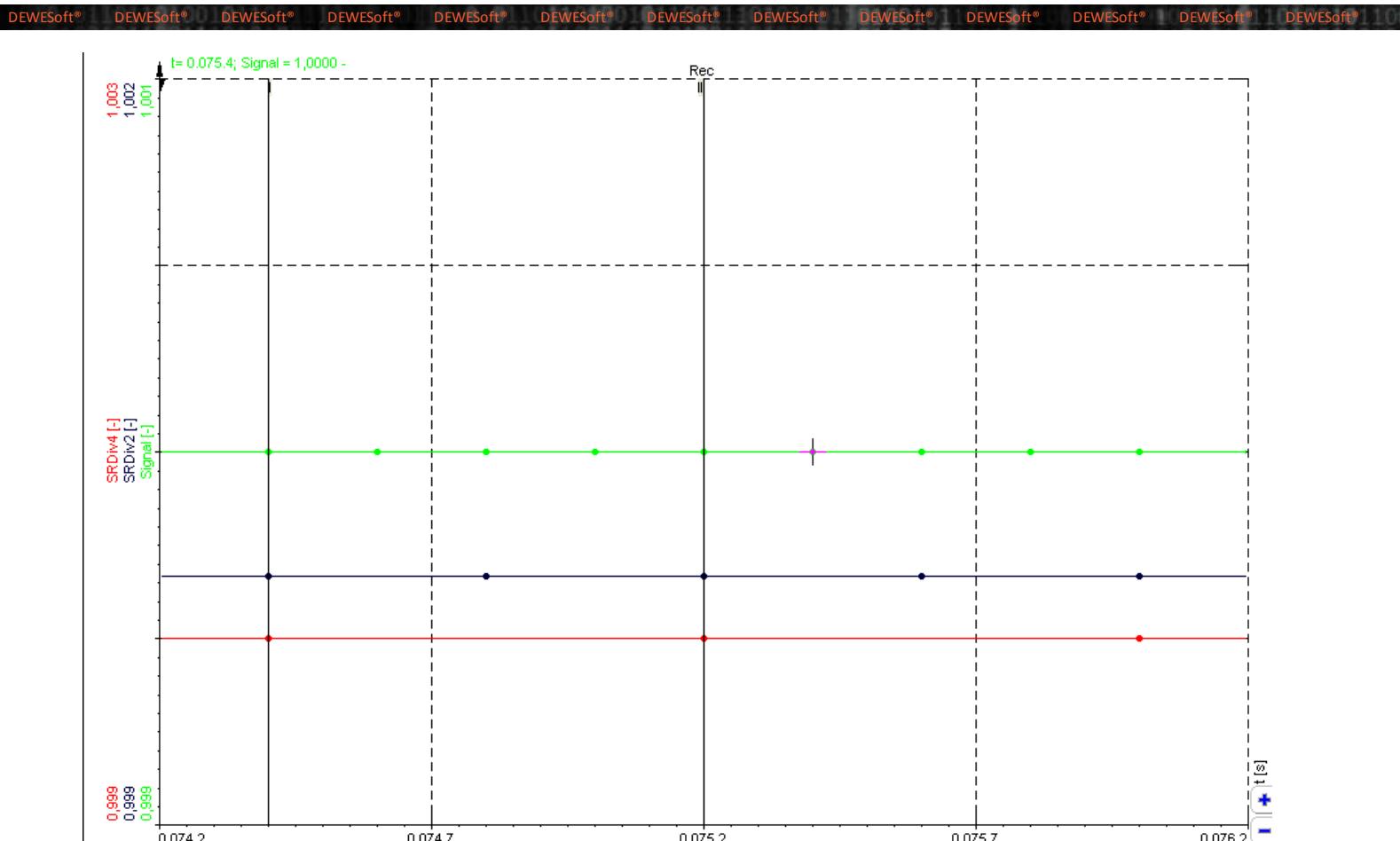
SRDiv is the sample rate divider set for the channel. SRDiv can be changed in the channel setup dialogue:



In the illustration below you can see 3 channels: the green channel `signal` is the source signal.

The blue channel `SRDiv2` shows the same signal with a sample rate divider set to 2: you can see that it has only half the data points of the source signal.

The red channel `SRDiv4` shows also the same signal with sample rate divider set to 4: it has only quarter of the data points of the source signal.



see also: [SetSRDiv](#), [SetSRDivType](#), [CalcSRDiv](#)

Interface: [IChannel](#)



2.1.22.100 SRDivType

property SRDivType: [TSRDivType](#)

the type of the sample rate divider

see also: [TSRDivType](#)

Interface: [IChannel](#)



2.1.22.101 Scale

property Scale: Double

Scale is the scale factor of the channel. See [Scaling](#) for more details.

see also: [Scaling](#), [Offset](#), [GetScaledData](#), [GetScaledDataEx](#), [GetScaledDataEx1](#)

Interface: [IChannel](#)  read/write



2.1.22.102 ScaleValue

```
function ScaleValue(Value: Single): Single;
```

`ScaleValue` returns the scaled value of the given unscaled `Value`.

Classifier	Name	Type	Description
	Value	Single	the unscaled value which should be scaled
-	RESULT	Single	the scaled value of the given unscaled Value

2.1.22.103 ScaleValueDouble

```
function ScaleValueDouble(value: Double): Double;
```

the same as [ScaleValue](#), but the result is of data type Double (see [Data Types](#))

see also: [Scaling](#), [ScaleValue](#), [Scale](#), [Offset](#), [GetScaledData](#), [GetScaledDataEx](#), [GetScaledDataEx1](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Value	Double	The input value
-	RESULT	Double	The scaled output value

2.1.22.104 Scale

property Scale : Double

is the same as: [Scale](#)

Interface: IChannel

2122105 SecondX

property `SecondX`: Double

This is a project specific feature.

see also: [FirstX](#)

Interface: [IChannel](#)

2.1.22.106 SelectorIndex

property SelectorIndex: OleVariant

used to override the index for the channel-tree view display

Interface: [IChannel](#)

2.1.22.107 SelectorIndexLevel

property SelectorIndexLevel: Integer

the index level of the selector: see [SelectorIndex](#)

Interface: [IChannel](#)

2.1.22.108 SelectorIndexStartLevel

property SelectorIndexStartLevel: Integer

the start level of the selector: see [SelectorIndex](#)

Interface: [IChannel](#)

2.1.22.109 SetAsStringChannel

procedure SetAsStringChannel(Size: Integer);

will mark this channel as string channel (internal data type is array of bytes).

see also: [String Channels](#), [AddAsyncString](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Size	Integer	this is the size in characters of the values that you can store in the string channel. Longer strings will be truncated. You should choose the lowest possible size to reduce the file size of the data file.

2.1.22.110 SetAsync

```
procedure SetAsync(Async: WordBool);
```

if `Async` is `True`, the channel will be asynchronous, if it is `False` it will be synchronous.

see also: [Async, Synchronism](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Async	WordBool	if <code>Async</code> is <code>True</code> , the channel will be asynchronous, if it is <code>False</code> it will be synchronous.

2.1.22.111 SetChannelSetup

```
procedure SetChannelSetup(Data: OleVariant);
```

will set the channel setup. see [GetChannelSetup](#) for more details.

see also: [XML Setup, GetChannelSetup](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	Data	OleVariant	the new channel setup - must be an array of bytes. For details on the XML format see: XML Setup .

2.1.22.112 SetDataType

```
procedure SetDataType(ADataType: Integer);
```

Will set the data type for the channel. See [Data Types](#) for more details

see [Data Types, DataType, SetAsStringChannel](#)

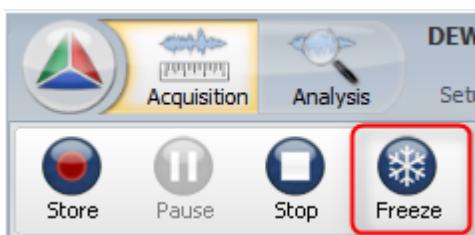
Interface: [IChannel](#)

Classifier	Name	Type	Description
	ADataType	Integer	See Data Types for details

2.1.22.113 SetFreezeMode

```
procedure SetFreezeMode(Freeze: WordBool);
```

visual controls can call this function to read the data from the freeze buffer instead of the direct buffer when the application is in freeze mode



see also: [IData.FreezeMode](#)

Interface: IChannel

Classifier	Name	Type	Description
	Freeze	WordBool	activate (TRUE) or deactivate (FALSE) the freeze mode

2.1.22.114 SetIsSingleValue

```
procedure SetIsSingleValue(Value: WordBool);
```

will setup this channel to be a single value channel (see [Single Value Channel](#)).

Single value channels must always be mounted as synchronous channels ([Async](#) must be FALSE, see also [Synchronism](#)).).

see also: [How to: Mount Dewesoft Channels](#), [Single Value Channel](#), [SingleValue](#), [Text](#) (for single value [Text channels](#))

Interface: IChannel

Classifier	Name	Type	Description
	Value	WordBool	set the channel to be a Single Value Channel (TRUE) or not (FALSE)

2.1.22.115 SetSRDiv

```
procedure SetSRDiv(SRDiv: Integer);
```

`SetSRDiv` allows setting the s

Interface: IChannel			
Classifier	Name	Type	Description
	SRDiv	Integer	the sample rate divider to be set

2.1.22.116 SetSRDivType

```
procedure SetSRDivType(AType: TSRDivType);
```

`SetSRDivType` allows setting the type of sample rate divider. Only relevant if the sample rate divider value (see [SRDiv](#)) is ≥ 1 .



see also: [TSRDivType](#), [SRDiv](#), [SetSRDiv](#)

Interface: [IChannel](#)

Classifier	Name	Type	Description
	AType	TSRDivType	the type of sample rate divider: see TSRDivType for details.

2.1.22.117 Settings

property Settings: WideString

Just used in the channel list (ch. setup)

An arbitrary string where channels can store some additional settings. Different channel types may store different settings.

an analog channel may store the measurement range: 'Daggard direct (2.5 V)', 'Daggard direct (5 V)',

a math channels stores its formula: $3 \cdot 1^x + 0.1 + 1$

Interface: [IChannel](#)



2.1.22.118 ShowChannelSetup

```
procedure ShowChannelSetup();
```

will open the channel.

2.1.22.119 Shown

property `Shown: WordBool`

`Shown` defines whether a channel is shown or not. A way to hide special channels from the user.

see also: [IData_ShownChannels](#), [Used](#), [Stored](#)

Interface: [IChannel](#)  read/write

2.1.22.120 SingleValue

property `SingleValue: Double`

the single value of a numeric [Single Value Channel](#).

for single value [String Channels](#) you must use: [Text](#)

see also: [How to: Mount Dewesoft Channels](#), [Single Value Channel](#), [SetIsSingleValue](#), [Text](#)

Interface: [IChannel](#)  read/write

2.1.22.121 Stored

property `Stored: WordBool`

if the channel data will be stored in the DEWESoft® data file (TRUE) or not (FALSE).

In the illustration below you can see that channel 1 will not be stored in the data file. It can still be seen during measurement and it can be used in Math formulas.



One use-case for this would be to reduce the size of the data files, e.g. you could set the `Stored` property of the measurement channel (which may have a high sample rate) to `FALSE`, so that the raw-data is not stored the data file, but you could still use it in Math channels (e.g. to calculate some block based statistics, like Min, Max, Avg, RMS, etc.) and only store the calculated (reduced) data samples.

You can of course not store an unused (see [Used](#)) channel. That's why the `Store` button does not even show up for channels that are unused (see illustration above).

see also: [Used](#)

Interface: [IChannel](#) read/write

2.1.22.122 Tag

property Tag: Integer

an arbitrary integer number that might freely be used by plugins or automation applications.

Interface: [IChannel](#) read/write

2.1.22.123 Text

property Text: WideString

the single value for single value [String Channels](#). (for data-type = 11)

see also: [How to: Mount Dewesoft Channels](#), [String Channels](#), [Single Value Channel](#), [SetIsSingleValue](#), [SetAsStringChannel](#)

Interface: [IChannel](#)  read/write

2.1.22.124 Typical.MaxValue

property Typical.MaxValue: Single

Typical.MaxValue is the maximum value of the y-axis of a graph displaying a certain channel.

If [UserScaleMax](#) has been set, it will be used, otherwise the value is 5.

see also: [UserScaleMax](#)

Interface: [IChannel](#)  read-only

2.1.22.125 Typical.MinValue

property Typical.MinValue: Single

Typical.MinValue is the minimum value of the y-axis of a graph displaying a certain channel.

If [UserScaleMin](#) has been set, it will be used, otherwise the value is -5.

see also: [UserScaleMin](#)

Interface: [IChannel](#)  read-only

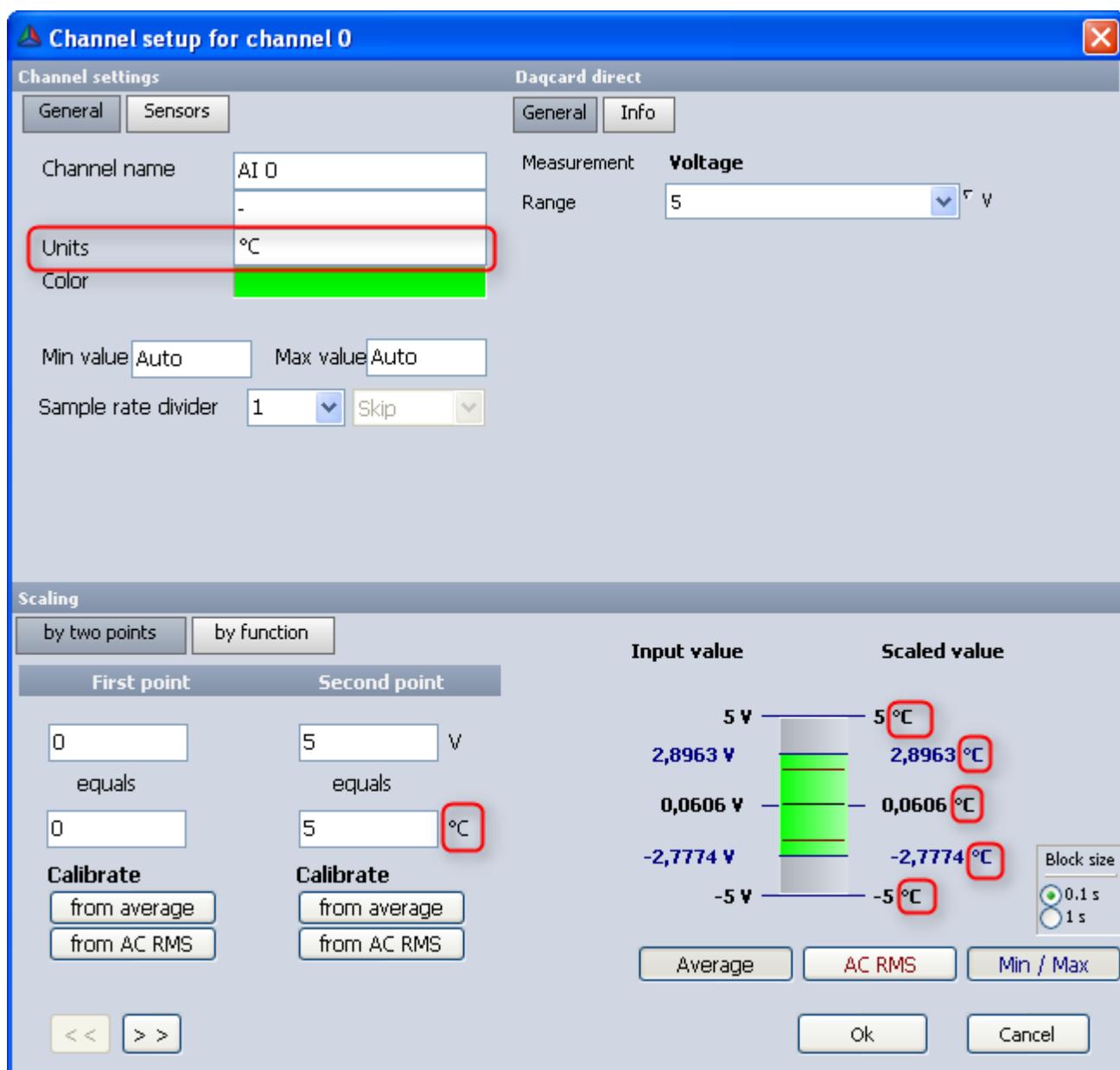
2.1.22.126 Unit_

property Unit_: WideString

Unit_ defines the measurement unit of a channel.

Note: the underscore at the end is because Unit is a keyword in Delphi and must not be used as a variable name.

The unit of a channel can be changed in the channel setup dialogue:

Interface: [IChannel](#)

read/write

2.1.22.127 UpdateXML

```
procedure UpdateXML(DOMDocument: OleVariant; DOMNode: OleVariant; Write: WordBool);
```

will read (when `Write=FALSE`) or write (when `Write=TRUE`) the channel properties from/to the `DOMNode` of the XML setup document `DOMDocument`.

see also: [IPlugin4.OnEvent](#) (for event Id `evOnUpdateXML`)

Interface: [IChannel](#)

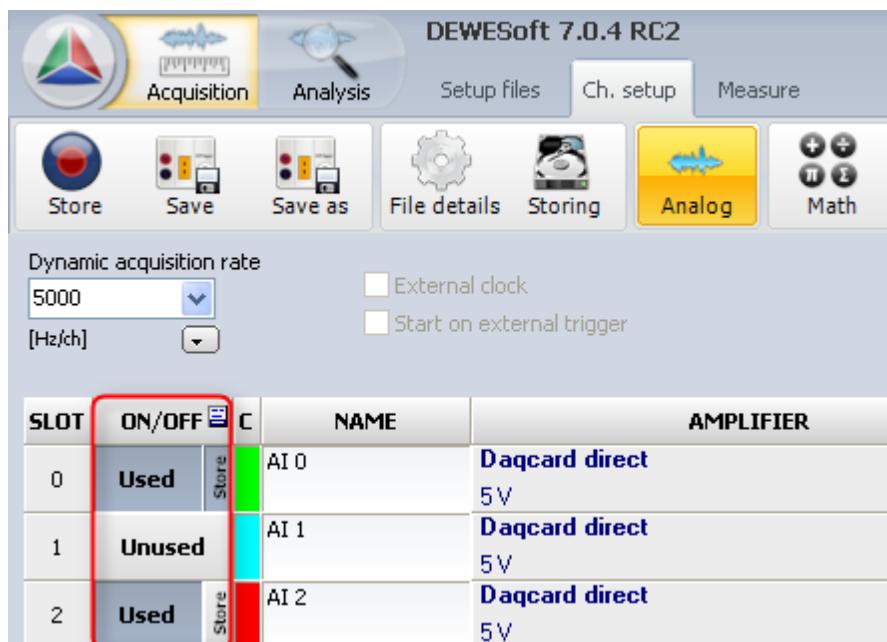
DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
	DOMDocument	OleVariant	the XML setup document									
	DOMNode	OleVariant	the <i>DOMNode</i> where to write the channel setup to									
	Write	WordBool	TRUE to write the channel setup, FALSE to read it									

2.1.22.128 Used

property Used: WordBool

`Used` specifies if the channel is used or not, corresponding to the *Used* button of the channel setup.

Only used channels will show up during measurement and can be used in Math channels.



see also: [Stored](#), [Shown](#), [IData.UsedChannels](#)

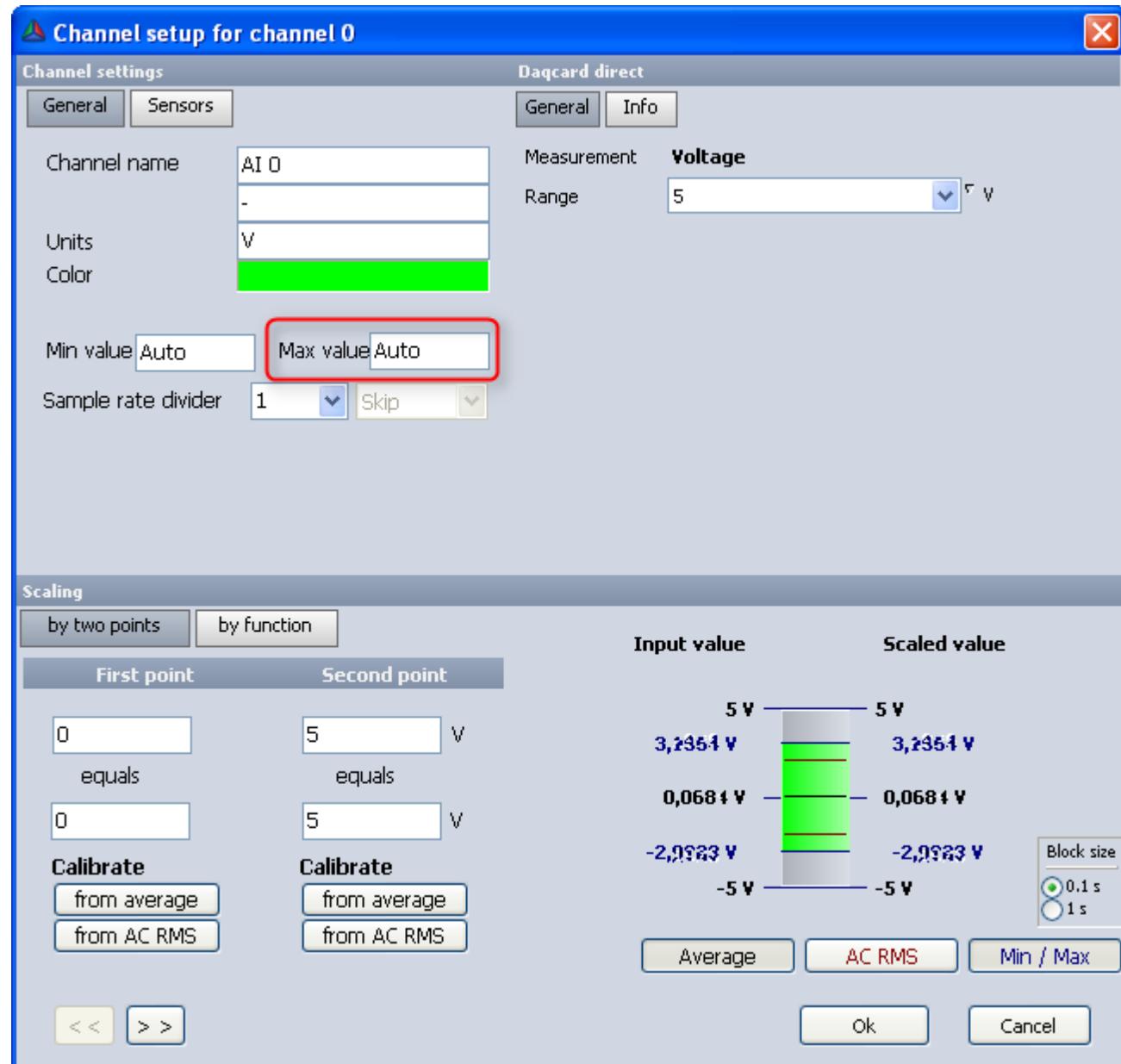
Interface: [IChannel](#)

read/write

2.1.22.129 UserScaleMax

property UserScaleMax: Single

UserScaleMax offers the ability to change the default maximum value of the scaling of the y-axis of a channel.



see also: [Typical.MaxValue](#), [UserScaleMin](#), [Typical.MinValue](#)

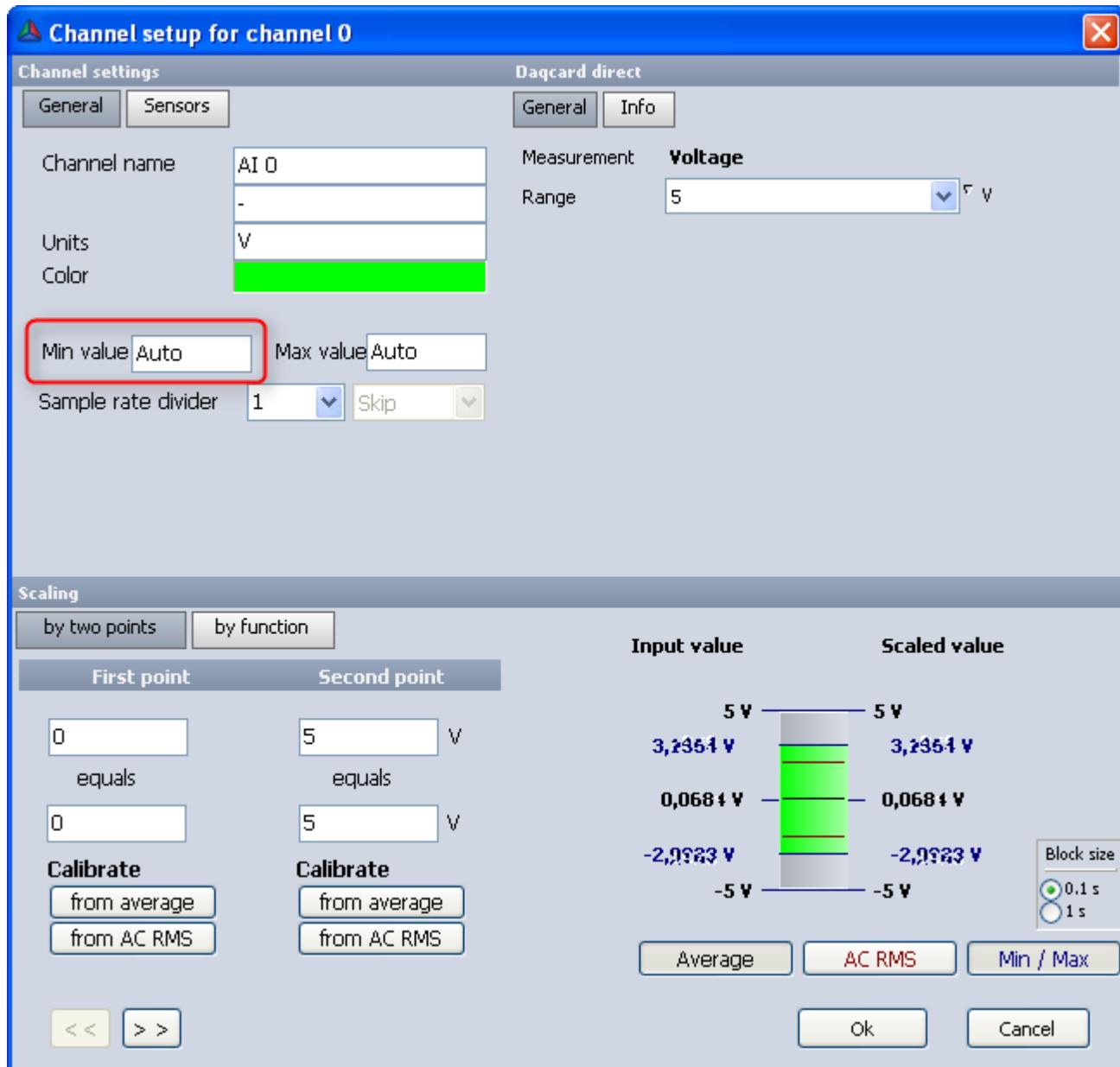
Interface: [IChannel](#)



2.1.22.130 UserScaleMin

property UserScaleMin: Single

UserScaleMin offers the ability to change the default minimum value of the scaling of the y-axis of a channel.



see also: [TypicalMinValue](#), [UserScaleMax](#), [TypicalMaxValue](#)

Interface: [IChannel](#)

read/write

is a convenient way to read the data from a channel without the need to think about the wrap-around of the data-buffer (see [The Buffer Structure](#)).

Use [IChannel.CreateConnection](#) to create a channel connection. Make sure to set the correct connection type: [AType](#).

see also: [IChannel.CreateConnection](#)

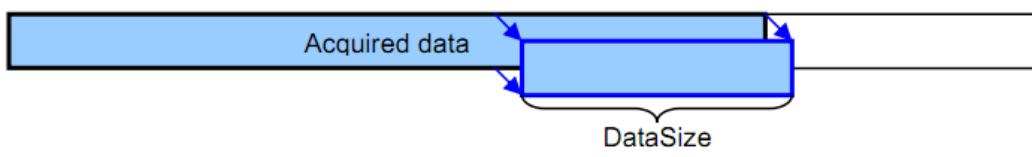
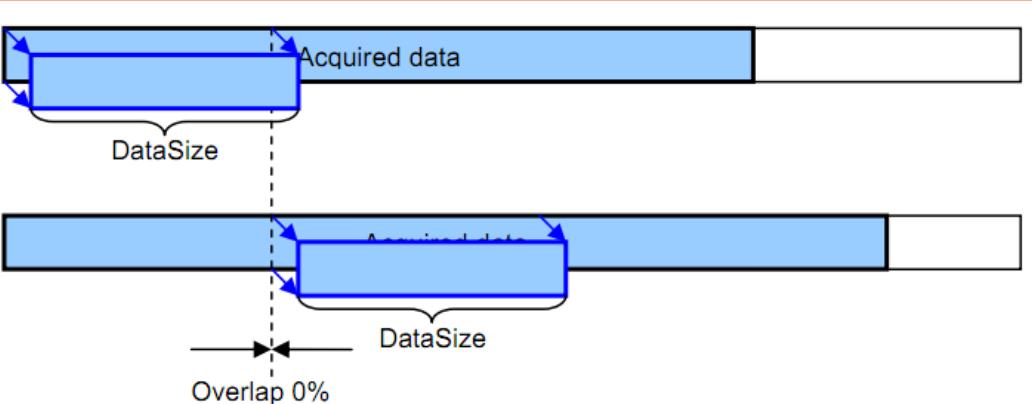
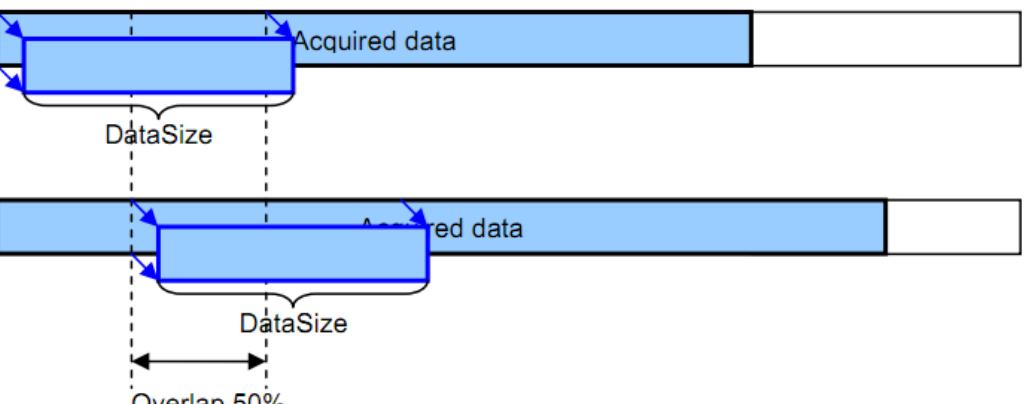
2.1.23.1 AType

property AType: ConnTypes

AType defines the type of the channel connection and therefore alters the behavior of the following functions:

- [GetDataBlocks](#), [GetDataBlocks1](#)
 - [GetDataValues](#), [GetDataValues1](#)
 - [GetTSBlocks](#), [GetTSBlocks1](#)
 - [GetTSValues](#), [GetTSValues1](#)

Values of ConnTypes

De c	He x	Name	Description
0	0x00	ctLast	<p>for reading the last BlockSize samples of the data (see also ctNew below). In the example below you can see that 2 subsequent calls to GetDataBlocks may return overlapping data.</p> 
1	0x01	ctOverl ap	<p>for reading overlapping blocks of data (with BlockSize samples). The overlap-percentage can be specified via Overlap.</p> <p>Example with 0% overlap:</p>  <p>Example with 50% overlap:</p> 
2	0x02	ctTrigg er	<p>for reading the data which is displayed in the Scope-screen on triggers. You must use IApp.SetScopeParams before, in order to setup the trigger of the scope and then activate the scope trigger by calling IApp.SetScopeUsed (TRUE).</p>
3	0x03	ctNew	<p>for reading a certain amount of the last data (specified by BlockSize or NumValues) only if such an amount of new data (not read yet) is available. Compare to ctLast above.</p>

see also: [ConnTypes](#), [IChannel](#)Interface: [IChannelConnection](#)

2.1.23.2 BlockSize

property BlockSize: Integer

`BlockSize` specifies the size of the data block(s) to read using [GetDataBlocks](#) or [GetDataBlocks1](#).

Interface: [IChannelConnection](#)



2.1.23.3 Channel

property Channel: IChannel

Channel the channel for this channel connection

1.4.5. $\langle \Omega \rangle = \langle S \rangle = 0$



2.1.23.4 GetDataBlocks

```
function GetDataBlocks(NumBlocks: Integer): OleVariant;
```

`GetDataBlocks` returns a certain number of data blocks from the direct buffer (see [The Buffer Structure](#)).

The return value is an `OleVariant` containing an array of values of the type `Single`. The size of the array will be `NumBlocks` times the size of one data block. The size of the data blocks and the position within the direct buffer from where data is retrieved are dependant on the property `AType` and `BlockSize`. See `AType` for detailed information.

Interface: IChannelConnection

Classifier	Name	Type	Description
	NumBlocks	Integer	defines the number of data blocks to read
-	<i>RESULT</i>	OleVariant	The return value is an OleVariant containing an array of values of the type Single . The size of the array will be NumBlocks times the size of one data block. The size of the data blocks and the position within the direct buffer from where data is retrieved are dependant on the property AType and BlockSize . See AType for detailed information.

2.1.23.5 GetDataBlocks1

```
procedure GetDataBlocks1(NumBlocks: Integer; out Data: OleVariant);
```

Same as `GetDataBlocks` implemented as a procedure.

Interface: IChannelConnection

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name		Type		Description							
	NumBlocks		Integer		see GetDataBlocks							
out	Data		OleVariant		see return value of GetDataBlocks							

2.1.23.6 GetDataValues

```
function GetDataValues(NumValues: Integer): OleVariant;
```

GetDataValues returns the specified amount of data values from the direct buffer.

The return value is an OleVariant containing an array of values of the type Single. The size of the array will be equal to NumValues.

If GetDataValues is used, [AType](#) would be set to ctLast in most cases.

If [AType](#) is set to ctNew, data will be returned only if the specified amount of new data (not read yet) is available.

If GetDataValues is used for reading asynchronous data (see [Synchronism](#)), the corresponding timestamps can be read by [GetTSValues](#). If those two functions are used, [GetTSValues](#) has to be called before GetDataValues is called.

Interface: [IChannelConnection](#)

Classifier	Name	Type	Description
	NumValues	Integer	specifies the amount of data values to retrieve from the direct buffer.
-	RESULT	OleVariant	The return value is an OleVariant containing an array of values of the type Single. The size of the array will be equal to NumValues.

2.1.23.7 GetDataValues1

```
procedure GetDataValues1(NumValues: Integer; out Data: OleVariant);
```

Same as [GetDataValues](#) implemented as a procedure.

Interface: [IChannelConnection](#)

Classifier	Name	Type	Description
	NumValues	Integer	see GetDataValues
out	Data	OleVariant	same as the return value of GetDataValues

2.1.23.8 GetTSBlocks

```
function GetTSBlocks(NumBlocks: Integer): OleVariant;
```

GetTSBlocks has to be used in conjunction with [GetDataBlocks](#) in case of asynchronous channels (see [Synchronism](#)).

When those two functions are used, GetTSBlocks has to be called before [GetDataBlocks](#).

The return value is an OleVariant containing an array of timestamp values (see [DateTime](#)) of the type Double.

Interface: [IChannelConnection](#)

Classifier	Name	Type	Description
	NumBlocks	Integer	specifies the number of blocks to read.
-	RESULT	OleVariant	The return value is an OleVariant containing an array of timestamp values (see DateTime) of the type Double.

2.1.23.9 GetTSBlocks1

```
procedure GetTSBlocks1(NumBlocks: Integer; out Data: OleVariant);
```

Same as [GetTSBlocks](#) implemented as a procedure.

Interface: [IChannelConnection](#)

Classifier	Name	Type	Description
	NumBlocks	Integer	see GetTSBlocks
out	Data	OleVariant	same as the return value of GetTSBlocks

2.1.23.10 GetTSValues

```
function GetTSValues(NumValues: Integer): OleVariant;
```

GetTSValues returns a certain amount (specified by NumValues) of timestamps (see [DateTime](#)) from the direct buffer (see [The Buffer Structure](#)) starting at the position where timestamp data has not been read yet.

The return value is an OleVariant containing an array of timestamp values (see [DateTime](#)) of the type Double. The size of the array will be equal to NumValues.

If GetTSValues is used, [AType](#) would be set to `ctLast` in most cases.

If [AType](#) is set to `ctNew`, data will be returned only if the specified amount of new data (not read yet) is available.

GetTSValues is intended to be used in conjunction with [GetDataValues](#). When both of these functions are used, GetTSValues has to be called before [GetDataValues](#).

Interface: [IChannelConnection](#)

Classifier	Name	Type	Description
	NumValues	Integer	specifies the amount of timestamps to retrieve from the direct buffer.
-	RESULT	OleVariant	The return value is an OleVariant containing an array of timestamp values (see DateTime) of the type Double. The size of the array will be equal to NumValues.

2.1.23.11 GetTSValues1

```
procedure GetTSValues1(NumValues: Integer; out Data: OleVariant);
```

Same as [GetTSValues](#), but implemented as a procedure.

Interface: [IChannelConnection](#)

Classifier	Name	Type	Description
	NumValues	Integer	see GetTSValues
out	Data	OleVariant	same as the return value of GetTSValues

2.1.23.12 NumBlocks

```
property NumBlocks: Integer
```

NumBlocks is the number of available data blocks of the size specified by [BlockSize](#) within the direct buffer (see [The Buffer Structure](#)) that have not been not read yet.

see also: [AType](#), [BlockSize](#), [NumValues](#)

Interface: [IChannelConnection](#)  read-only

2.1.23.13 NumValues

```
property NumValues: Integer
```

NumValues is the number of available values within the direct buffer (see [The Buffer Structure](#)) that have not been not read yet.

see also: [AType](#), [BlockSize](#), [NumBlocks](#)

Interface: [IChannelConnection](#)  read-only

2.1.23.14 Overlap

```
property Overlap: Integer
```

The percentage of how much the data blocks to read overlap each other. Only relevant if [AType](#) is `ctOverlap`.

see also: [AType](#) (`ctOverlap`)

Interface: [IChannelConnection](#)  read/write

2.1.23.15 Reset

```
procedure Reset();
```

To reset the channel connection at the current data (compare to [Start](#)).

In the case of multiple connections, calling [IData.StartDataSync](#) before resetting the connections and calling [IData.EndDataSync](#) after having reset each connection is necessary for synchronization.

Remark: Reset is also called by `IChannel.CreateConnection`

Interface: [IChannelConnection](#)

2.1.23.16 Start

```
procedure Start();
```

Start resets the channel connection to the start of the acquisition.

Interface: [IChannelConnection](#)

2.1.24 IChannelGroup

a group of channels (see [IChannel](#)): See [IData.Groups](#) for a list of different groups.

child-interfaces: [IChannelGroup2](#), [IPluginGroup](#), [ImportGroup](#)

see also: [IChannelGroup2](#), [IChannelGroup](#), [IDataGroups](#), [IChannelGroups](#)

2.1.24.1 ExportRate

property ExportRate: Integer

`ExportRate` defines the sample rate for exporting asynchronous data (currently only supported by DIAdem). This property can be set separately for each group.

Interface: **IChannelGroup** read/write

2.1.24.2 GetIndexName

```
function GetIndexName(Index: T ChIndex): WideString;
```

Returns the name of the channel group specified by `Index`.

see also: [T_ChIndex](#), [Name](#), [IChannelGroup2.GetIndexNameShort](#)

Interface: [IChannelGroup](#)

Classifier	Name	Type	Description
	Index	T_ChIndex	the index of the channel group
-	RESULT	WideString	the name of the channel group specified by Index

2.1.24.3 Name

property Name: WideString

The name of the channel group.

see also: [GetIndexName](#), [IChannelGroup2.GetIndexNameShort](#)

Interface: [IChannelGroup](#)  read-only

2.1.25 IChannelGroup2

An extended version of [IChannelGroup](#)

The parent interface is: [IChannelGroup](#)

2.1.25.1 GetIndexNameShort

function GetIndexNameShort(Index: [T_ChIndex](#)): WideString;

GetIndexNameShort returns the name of the index in an abbreviated version.

see also: [T_ChIndex](#), [IChannelGroup.Name](#)

Interface: [IChannelGroup2](#)

Classifier	Name	Type	Description
	Index	T_ChIndex	the channel index (see T_ChIndex)
-	RESULT	WideString	the name of the index in an abbreviated version.

2.1.26 IChannelGroups

a list of channel group ([IChannelGroup](#)) objects.

see also: [IData.Groups](#)

2.1.26.1 Count

property Count: Integer

`Count` indicates the number of channel groups in the [Item](#) list.

Interface: [IChannelGroups](#)



2.1.26.2 Item

property Item[Index: Integer]: IChannelGroup;

Item[I] is the channel group at index I. I is in the range of 0..Count-1.

These items are of the type

see also: [IChannelGroup](#)

2.1.27 IChannell ist

Is a container for [IChannel](#) elements

see also: [IChannel](#), [IAQGroup](#), [AQChannels](#), [IData_UsedChannels](#), [IData_ShownChannels](#), [IData_AllChannels](#)

21271 Count

property Count: Integer;

`Count` is the number of channels in the [Item](#) list.

Interface: [IChannelList](#)

2.1.27.2 Item

property Item[Index: Integer]: IChannel

Item[I] is the channel at index I. I is in the range of 0...Count-1.

see also: [IChannel](#)

Interface: [IChannelList](#)

2.1.28 IChannelListEx

extended channel list interface

see also: [IChannelList](#)

2.1.28.1 AddCh

```
procedure AddCh(const Ch: IChannel);
```

will add the given channel Ch to the channel list.

see also: [IChannel](#)

Interface: [IChannelListEx](#)

Classifier	Name	Type	Description
const	Ch	IChannel	the channel that will be added to the channel list

2.1.28.2 Clear

```
procedure Clear();
```

will clear the channel list (remove all channels)

Interface: [IChannelListEx](#)

2.1.28.3 SetCh

```
procedure SetCh(Index: Integer; const Ch: IChannel);
```

will set the item at the given Index of the list to the channel Ch.

Interface: [IChannelListEx](#)

Classifier	Name	Type	Description
	Index	Integer	the index of the item in the list
const	Ch	IChannel	the channel instance

2.1.29 ICntChannel

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

2.1.29.1 AdvCntMode

property AdvCntMode: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#) read-only

2.1.29.2 BaseMode

property BaseMode: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.3 CanAutoCalculate

property CanAutoCalculate: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.4 CardChannelIO

```
property CardChannel10: IChannel
```

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.5 CardChannel1

property CardChannel1: IChannel

For internal use only! This feature must only be used by DEWEsoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.6 CntAux

property CntAux: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.7 CntAuxInv

property CntAuxInv: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.8 CntDoManualReset

property CntDoManualReset: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.9 CntEncoderMode

property CntEncoderMode: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.10 CntEncoderZero

property CntEncoderZero: WordBool;

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#) read-only

2.1.29.11 CntEventWithZero

property CntEventWithZero: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#) read-only

2.1.29.12 CntFilter

property CntFilter: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.13 CntGate

property CntGate: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.14 CntGateInv

property CntGateInv: WordBool;

For internal use only! This feature must only be used by DEWEsoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.15 CntMode

property CntMode: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.16 CntnewValueUpdateMode

property CntnewValueUpdateMode: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.17 CntPair

property CntPair: ICntChannel

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.18 CntResetOnStartMeasure

property CntResetOnStartMeasure: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.19 CntSignalZero

property CntSignalZero: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.20 CntSource

property CntSource: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.21 CntSourceInv

property CntSourceInv: WordBool

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.22 CntUpDownMode

property CntUpDownMode: Integer

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)

2.1.29.23 DIChannels

property DIChannels[Index: Integer]: IChannel;

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.24 TrigLevels

```
property TrigLevels[Index: Integer]: IDigitalTrigLevel
```

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.29.25 TrigLevelsCombined

```
property TrigLevelsCombined: WordBool
```

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [ICntChannel](#)  read-only

2.1.30 ICustomDAQ

interface for custom DAQ devices that will then show up in the *Analog* section of the hardware setup.

2.1.30.1 CheckSampleRate

```
function CheckSampleRate
(WantedRate: Integer; NumChannels: Integer; ChList: OleVariant; var AdjustedRate: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description										
	WantedRate	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com										
	NumChannels	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com										
	ChList	OleVariant	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com										
var	AdjustedRate	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com										
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success										

2.1.30.2 GetBitResolution

```
function GetBitResolution(ChIndex: Integer; var Bits: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
	ChIndex	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
var	Bits	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.30.3 GetBufferSize

```
function GetBufferSize(out Value: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
out	Value	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com									
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success									

2.1.30.4 GetCNTBitResolution

```
function GetCNTBitResolution(ChIndex: Integer; var Bits: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
	ChIndex	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
var	Bits	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.30.5 GetCardName

```
function GetCardName(var Name: WideString): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
var	Name	WideString	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.30.6 GetChannelGain

```
function GetChannelGain(out Gain: Single): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
out	Gain	Single	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	<u>HResult</u>	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.7 GetCurrentTime

```
function GetcurrentTime(var Time: Largeuint): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
var	Time	Largeuint	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	<u>HResult</u>	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.8 GetDWTypeLibVersion

```
function GetDWTypeLibVersion(var Version: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
var	Version	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.9 GetData

```
function GetData(var Samples: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
var	Samples	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com									
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success									

2.1.30.10 GetDeviceCode

```
function GetDeviceCode(out Code: WideString; out Verify: WideString): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
out	Code	WideString	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
out	Verify	WideString	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.30.11 GetDeviceType

```
function GetDeviceType(): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.30.12 GetMinMax

```
function GetMinMax(ChnIndex: Integer; var Min: Single; var Max: Single): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	
	Classifier	Name	Type	Description										
		ChnIndex	Integer	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>										
var		Min	Single	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>										
var		Max	Single	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>										
-		RESULT	<u>HResult</u>	negative values indicate an error, others ($>=0$) indicate success										

2.1.30.13 GetOptionName

```
function GetOptionName(Index: Integer; var Name: WideString): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
	Index	Integer	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>
var	Name	WideString	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>
-	RESULT	<u>HResult</u>	negative values indicate an error, others ($>=0$) indicate success

2.1.30.14 GetOptionsCount

```
function GetOptionsCount(var Cnt: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
var	Cnt	Integer	<i>Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com</i>
-	RESULT	<u>HResult</u>	negative values indicate an error, others ($>=0$) indicate success

2.1.30.15 GetSampleRates

```
function GetSampleRates(var List: OleVariant): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
var	List	OleVariant	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.16 Get_CardFound

```
function Get_CardFound(out Value: WordBool): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
out	Value	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.17 Get_NumCNTChannels

```
function Get_NumCNTChannels(out Value: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
out	Value	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.18 Get NumChannels

```
function Get_NumChannels(out Value: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
out	Value	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	<u>HResult</u>	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.19 HideSetupFrame

```
function HideSetupFrame(): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.30.20 SetApp

```
function SetApp(const App: IApp): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [IApp](#)

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
const	App	IApp	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.21 ShowSetupFrame

```
function ShowSetupFrame(AppHandle: Integer; Handle: Integer): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
	AppHandle	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
	Handle	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.22 StartAcq

```
function StartAcq(SampleRate: Integer; SetupMode: WordBool): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
	SampleRate	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
	SetupMode	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.30.23 StopAcq

```
function StopAcq(): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ](#)

Classifier	Name	Type	Description
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.31 ICustomDAQ2

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.31.1 OnMessage

```
function OnMessage  
(Msg: Integer; InParam: OleVariant; var OutParam: OleVariant): HResult;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ICustomDAQ2](#)

Classifier	Name	Type	Description
	Msg	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
	InParam	OleVariant	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
var	OutParam	OleVariant	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
-	RESULT	HResult	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.32 ICustomExport

child interface: [ICustomExport2](#)

see also: [ICustomExport3](#)

2.1.32.1 EndChannel

```
function EndChannel() : HRESULT;
```

Ends the export of a channel when the `ExportType` is set to `etChannelBased`.

see also: StartChannel EndValue

Interface: [ICustomExport](#)  use only for custom export add-ons

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success								

2.1.32.2 EndDataFolder

```
function EndDataFolder(): HResult;
```

`EndDataFolder` is called at the end of each data folder. A data folder is a section of time where data is stored: in other words the time between a start and a stop trigger.

see also: [StartDataFolder](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.3 EndExport

```
function EndExport(): HResult;
```

`EndExport` is called at the very end of the export. E.g. DLLs used during export can be released here.

see also: [StartExport](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.4 EndHeader

```
function EndHeader(): HResult;
```

`EndHeader` is called at the end of writing the header information of an export and before the export of the measurement data is started.

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.5 EndInfo

```
function EndInfo(): HResult;
```

`EndInfo` is called after writing the general information to the export file (functions [WriteInfoDate](#), [WriteInfoInteger](#), [WriteInfoSingle](#), [WriteInfoString](#)).

In most cases `EndInfo` will remain empty.

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
-	<i>RESULT</i>	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.6 EndValue

```
function EndValue(): HRESULT;
```

`EndValue` is called, when the `ExportType` is `etValueBased`, after one value of each channel is exported.

see also: [StartValue](#), [EndChannel](#)

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.32.7 Get AbsoluteTime

```
function Get_AbsoluteTime(out Value: WordBool) : HResult;
```

should return the value that has been set by `Set_AbsoluteTime`

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	WordBool	the value that has been set by Set_AbsoluteTime
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.32.8 Get_DataCount

```
function Get_DataCount(out Value: Largeuint): HResult;
```

Value specifies the number of samples.

This property is set by DEWEsoft®. For channel-based export the data count is different for each channel. For value-based export the data count is equal for each channel.

Some export add-ons need to know this information at the beginning of the export.

see also: [Set_DataCount](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	Largeuint	the number of samples
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.9 Get_ExportType

```
function Get_ExportType(out Value: ExportTypes): HResult;
```

ExportType specifies whether the export is value-based or channel-based (see [ExportTypes](#) for details).

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	ExportTypes	the export type
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.10 Get_Extension

```
function Get_Extension(out Value: WideString): HResult;
```

Extension specifies the file extension denoting the file type, e.g. ".txt" for a text file.

see also: [Get_FileName](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	WideString	the file extension denoting the file type, e.g. ".txt" for a text file
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.11 Get FileName

```
function Get_FileName(out Value: WideString): HResult;
```

Value is the name of the export file entered by the user before performing the export. This property is set by DEWESoft®.

see also: [Set_FileName](#), [Get_Extension](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	WideString	the name of the export file entered by the user before performing the export
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others ($>= 0$) indicate success

2.1.32.12 Get SupportsAsync

```
function Get_SupportsAsync(out Value: WordBool): HResult;
```

`Value` specifies whether the custom export supports asynchronous data (see [Synchronism](#)) or not. This is relevant only for channel-based export.

If Value is TRUE, asynchronous channels will be exported as such. If Value is FALSE, asynchronous channels will be exported as synchronous channels and gaps between values will be interpolated.

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	WordBool	If Value is TRUE, asynchronous channels will be exported as such. If Value is FALSE, asynchronous channels will be exported as synchronous channels and gaps between values will be interpolated.
-	<i>RESULT</i>	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.13 Get SupportsSRDiv

```
function Get_SupportsSRDiv(out Value: WordBool): HResult;
```

should set `Value` to `TRUE` if the custom export add-on supports a sample rate divider. `FALSE` otherwise.

This property is relevant only for channel-based export. If Value is FALSE, missing samples will be inserted if necessary.

Interface: [ICustomExport](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
out	Value	WordBool	should set Value to TRUE if the custom export add-on supports a sample rate divider, FALSE otherwise									
-	<i>RESULT</i>	HResult	negative values indicate an error, others ($>=0$) indicate success									

2.1.32.14 Get_TimeIncrease

```
function Get_TimeIncrease(out Value: Double): HResult;
```

Value defines the interval between two subsequent values of synchronous data (see [Synchronism](#)).

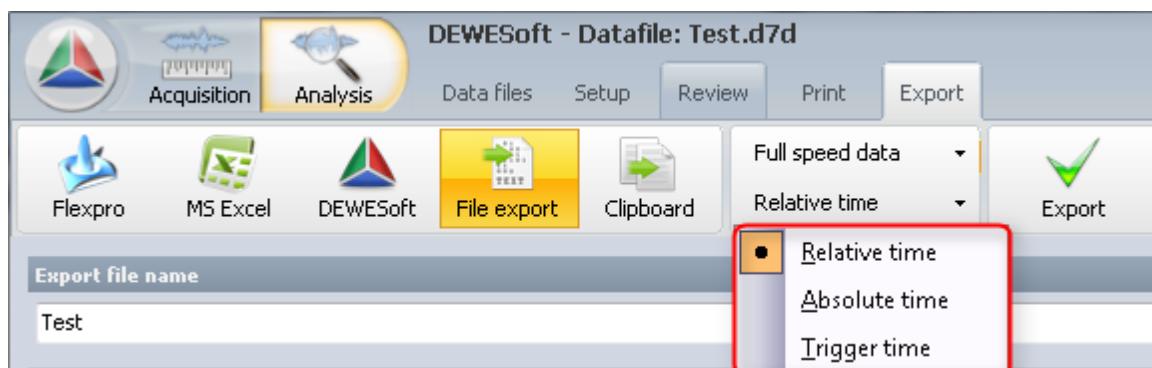
Interface: [ICustomExport](#)

Classifier	Name	Type	Description
out	Value	Double	the interval between two subsequent values of synchronous data
-	<i>RESULT</i>	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.15 Set_AbsoluteTime

```
function Set_AbsoluteTime(Value: WordBool): HResult;
```

DEWEsoft® will call this function when the export is being initialized to tell the custom export add-on if it should use absolute time (Value is TRUE) or not (Value is FALSE).



Note: Trigger time is not supported yet

see also: [Get_AbsoluteTime](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
	Value	WordBool	the custom export add-on should use absolute time if Value is TRUE, otherwise relative time should be used.
-	<i>RESULT</i>	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.16 Set_DataCount

```
function Set_DataCount(Value: Largeuint) : HResult;
```

for details see: [Get DataCount](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
	Value	Largeuint	the number of samples
-	RESULT	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.32.17 Set FileName

```
function Set FileName(const Value: WideString): HResult;
```

for details see: [Get FileName](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
const	Value	WideString	the file name
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.18 Set TimeIncrease

```
function Set TimeIncrease(Value: Double): HResult;
```

see: [Get TimeIncrease](#)

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
	Value	Double	time in seconds
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.19 StartAbsValue

```
function StartAbsValue(DateTime: TDateTime): HResult;
```

`StartAbsValue` is called when the [ExportType](#) is `etValueBased` and the time axis is set to absolute (see parameter `TimeAxis` in [IApp.ExportDataEx](#)) before one value of each channel is exported.

see also: [DateTime](#), [ExportTypes](#)

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
	DateTime	TDateTime	is the time stamp in absolute time
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.20 StartChannel

```
function StartChannel
(const FieldName: WideString; const FieldUnit: WideString; Async: WordBool):
HResult;
```

StartChannel is called when the [ExportType](#) is etChannelBased every time the export of a new channel is started.

see also: [EndChannel](#)

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
const	FieldName	WideString	is the name and the comment of the channel. E.g. this could be "AI 0 - MyComment" for an analogue channel.
const	FieldUnit	WideString	is the unit of the channel. E.g. this could be "m/s ² " or just "-", if no unit was specified for the channel
	Async	WordBool	specifies whether the channel contains asynchronous data or not (see also: Synchronism).
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.21 StartDataField

```
function StartDataField
(const FieldName: WideString; const FieldUnit: WideString; ExportRate: Integer):
HResult;
```

StartDataField is called for each channel for exporting the name, the unit and the rate for each channel.

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
const	FieldName	WideString	is the name and the comment of the channel. E.g. this could be "AI 0 - MyComment" for an analogue channel.
const	FieldUnit	WideString	is the unit of the channel. E.g. this could be "m/s ² " or just "-", if no unit was specified for the channel
	ExportRate	Integer	the acquisition rate of a channel
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.22 StartDataFolder

```
function StartDataFolder(const FolderName: WideString; AbsTime: TDateTime): HResult  
;
```

`StartDataFolder` is called when the export of channel-specific data starts. A data folder is a section of time where data is stored; in other words the time between a start and a stop trigger.

see also: [EndDataFolder](#), [DateTime](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
const	FolderName	WideString	the name of the data folder. E.g. this can be " <i>Data1</i> "
	AbsTime	TDateTime	the absolute start time of the folder
-	<i>RESULT</i>	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.23 StartExport

```
function StartExport(const App: IApp): HResult;
```

When `StartExport` is called, DLLs for exporting should be initialized if necessary (depending on the type of export) and the file to which data will be exported should be opened here.

see also: [EndExport](#), [IApp](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
const	App	IApp	a reference to the DEWEsoft® instance for accessing its functions and properties
-	<i>RESULT</i>	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.32.24 StartInfo

```
function StartInfo(const Info: WideString): HResult;
```

`StartInfo` is called when the writing of general information of the exported data begins. Then the following functions can be called: `WriteInfoDate`, `WriteInfoInteger`, `WriteInfoSingle`, `WriteInfoString`.

Interface: [ICustomExport](#) use only for custom export add-ons

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
const	Info	WideString	Info can contain any general header information.									
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success									

2.1.32.25 StartTimeField

```
function StartTimeField(const FieldName: WideString; const FieldUnit: WideString): HResult;
```

StartTimeField is called for exporting the name of the time channel and its unit.

Interface: [ICustomExport](#)

Classifier	Name	Type	Description
const	FieldName	WideString	is the name of the time channel. In most cases this will be just "Time".
const	FieldUnit	WideString	the time unit. E.g. this could be "s" for seconds
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.26 StartValue

```
function StartValue(TimeValue: Double): HResult;
```

StartValue is called, when the [ExportType](#) is etValueBased, and the time axis is set to relative (see parameter TimeAxis in [IApp.ExportDataEx](#)), before one value of each channel is exported.

see also: [EndValue](#), [WriteValue](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	TimeValue	Double	is the time value for the following set of values to export
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.27 WriteAsyncValue

```
function WriteAsyncValue(TimeStamp: Double; Value: Single): HResult;
```

WriteAsyncValues is called for each asynchronous value when the [ExportType](#) is etChannelBased.

see also: [WriteValue](#), [ICustomExport2.WriteAsyncDoubleValue](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description										
	TimeStamp	Double	the time that Value corresponds to										
	Value	Single	is the data value to export.										
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success										

2.1.32.28 WriteInfoDate

```
function WriteInfoDate(const Description: WideString; Value: TDateTime): HResult;
```

WriteInfoDate writes date information to the header of the export file.

see also: [DateTime](#)

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
const	Description	WideString	a string describing the value. E.g. this string could be something like " <i>Start time</i> "
	Value	TDateTime	the date-time value
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.29 WriteInfoInteger

```
function WriteInfoInteger(const Description: WideString; Param: Integer): HResult;
```

WriteInfoInteger writes a value of type Integer to the header of the export file.

Interface: [ICustomExport](#)  use only for custom export add-ons

Classifier	Name	Type	Description
const	Description	WideString	a string describing the value. E.g. this string could be something like " <i>Sample rate</i> "
	Param	Integer	the Integer value
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.32.30 WriteInfoSingle

```
function WriteInfoSingle(const Description: WideString; Value: Single): HResult;
```

WriteInfoSingle writes a value of type Single to the header of the export file.

Interface: [ICustomExport](#)  use only for custom export add-ons

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
const	Description	WideString	a string describing the value. E.g. this string could be something like " <i>Factor</i> "								
	Value	Single	the <i>Single</i> value								
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success								

2.1.32.31 WriteInfoString

```
function WriteInfoString(const Description: WideString; const Value: WideString) : HResult;
```

WriteInfoString writes a value of type String to the header of the export file.

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
const	Description	WideString	a string describing the value. E.g. this string could be something like " <i>Description</i> "
const	Value	WideString	the text value
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.32.32 WriteValue

```
function WriteValue(Value: Single) : HResult;
```

WriteValues is called when the [ExportType](#) is etValueBased, or for synchronous values when the [ExportType](#) is etChannelBased.

see also: [WriteAsyncValue](#), [ICustomExport2.WriteDoubleValue](#)

Interface: [ICustomExport](#) use only for custom export add-ons

Classifier	Name	Type	Description
	Value	Single	is the data value to export
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.33 ICustomExport2

parent interface: [ICustomExport](#)

see also: [ICustomExport3](#)

2.1.33.1 GetDWTypeLibVersion

```
function GetDWTypeLibVersion(out Value: Integer): HResult;
```

Value should be set to the expected DEWEsoft® type library version (DEWEsoft® minor version).

see also: [IApp.GetInterfaceVersion](#)

Interface: [ICustomExport2](#) use only for custom export add-ons

Classifier	Name	Type	Description
out	Value	Integer	should be set to the expected DEWEsoft® type library version (DEWEsoft® minor version)
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.2 Get SupportsDouble

```
function Get_SupportsDouble(out Value: WordBool): HResult;
```

If measurement values of data type Double (see also: [Data Types](#)) are

Interface: **ICustomExport2** uses only for custom export add one

Classifier	Name	Type	Description
out	Value	WordBool	If measurement values of data type Double are supported (TRUE) or not (FALSE).
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.3 SetAbsMax

```
function SetAbsMax(AbsMax: Single): HResult;
```

`AbsMax` is the absolute maximum of the measurement. This can be useful in conjunction with [SetAbsMin](#) if the export add-on needs to do some scaling.

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	AbsMax	Single	the absolute maximum of the measurement data
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.4 SetAbsMin

```
function SetAbsMin(AbsMin: Single): HResult;
```

AbsMin is the absolute minimum of the measurement. This can be useful in conjunction with [SetAbsMax](#) if the export add-on needs to do some scaling.

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	AbsMin	Single	the absolute minimum of the measurement data
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.33.5 SetApp

```
function SetApp(const App: IApp): HResult;
```

Provides a reference to [IApp](#)

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
const	App	IApp	Provides a reference to IApp
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.33.6 SetChannel

```
function SetChannel(const Ch: IChannel): HResult;
```

Provides a reference to the channel (see [IChannel](#)).

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
const	Ch	IChannel	a reference to the channel
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.33.7 SetChannelColor

```
function SetChannelColor(Color: Integer): HResult;
```

Sets the color of the channel.

see also [ColourCodes](#), [IChannel.MainDisplayColor](#)

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	Color	Integer	the color of the channel
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.8 SetDoubleFloat

```
function SetDoubleFloat(DoubleFloat: WordBool): HResult;
```

DoubleFloat defines if a channel contains data of the type Double (**TRUE**) or Single (**FALSE**): see also: [Data Types](#)

see also: [Get_SupportsDouble](#), [WriteAsncDoubleValue](#), [WriteDoubleValue](#)

Interface: `ICustomExport2` use only for custom export add-ons

Classifier	Name	Type	Description
	DoubleFloat	WordBool	DoubleFloat defines if a channel contains data of the type Double (TRUE) or Single (FALSE)
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.9 SetRangeMax

```
function SetRangeMax(Value: Single): HResult;
```

Value is the maximum measurement range.

see also: [SetRangeMin](#)

Interface: **ICustomExport2**  use only for custom export add-ons

Classifier	Name	Type	Description
	Value	Single	the maximum measurement range
-	RESULT	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.10 SetRangeMin

```
function SetRangeMin(Value: Single): HResult;
```

Value is the minimum measurement range.

see also: [SetRangeMax](#)

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	Value	Single	is the minimum measurement range
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.33.11 SetTrigOffset

```
function SetTrigOffset(TrigTime: Double): HResult;
```

TrigTime is the pre-time of a trigger event. It is not overlapping with previous trigger events.

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	TrigTime	Double	the pre-time of a trigger event
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

2.1.33.12 StartEvents

```
function StartEvents(): HResult;
```

StartEvents is called before the events are exported.

see also: [StopEvents](#)

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
-	RESULT	HResult	negative values indicate an error, others (≥ 0) indicate success

```
function StopEvents(): HResult;
```

`StopEvents` is called after the events have been exported.

see also: [StartEvents](#)

Interface: [ICustomExport2](#) use only for custom export add-ons

Classifier	Name	Type	Description
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.14 WriteAsyncDoubleValue

```
function WriteAsyncDoubleValue(TimeStamp: Double; Value: Double): HResult;
```

`WriteAsyncDoubleValue` will be called for each asynchronous value of data type `Double` (see also: [Data Types](#)) when the `ExportType` is `etChannelBased`.

`WriteAsyncDoubleValue` is similar to [ICustomExport.WriteAsyncValue](#).

see also: [Get_SupportsDouble](#), [SetDoubleFloat](#), [WriteDoubleValue](#)

Interface: **ICustomExport2**  use only for custom export add-ons

Classifier	Name	Type	Description
	TimeStamp	Double	the time that Value corresponds to
	Value	Double	is the data value to export.
-	<i>RESULT</i>	<u>HResult</u>	negative values indicate an error, others (≥ 0) indicate success

2.1.33.15 WriteDoubleValue

```
function WriteDoubleValue(Value: Double): HResult;
```

is called for each value of data type Double (see also: [Data Types](#)) when the [ExportType](#) is `etValueBased`, or for synchronous values of data type Double when the [ExportType](#) is `etChannelBased`.

is similar to [ICustomExport.WriteValue](#).

see also: [Get](#) [SupportsDouble](#), [SetDoubleFloat](#), [WriteAsVncDoubleValue](#)

Interface: **ICustomExport2** use only for custom export add-ons

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	Value	Double	is the data value to export								
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success								

2.1.33.16 WriteEvent

```
function WriteEvent
(EventType: Integer; const EventTypeString: WideString; Time: Double; const Comment
: WideString): HResult;
```

WriteEvent is called for each event of the exported data.

see also: [EventType](#)

Interface: [ICustomExport2](#)  use only for custom export add-ons

Classifier	Name	Type	Description
	EventType	Integer	EventType defines the type of event: see EventType
const	EventTypeString	WideString	a string defining the event type
	Time	Double	the time when the event occurred
const	Comment	WideString	an optional remark associated with the event.
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success

2.1.34 ICustomExport3

see also: [ICustomExport](#), [ICustomExport2](#)

2.1.34.1 OnEvent

```
function OnEvent(EventIDs: CustomExpEventIDs
; InParam: OleVariant; var OutParam: OleVariant): HResult;
```

OnEvent can be called for any event IDs specified in: [CustomExpEventIDs](#)

Interface: [ICustomExport3](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description										
	EventIDs	CustomExpEventIDs	the event ID: see CustomExpEventIDs										
	InParam	OleVariant	input parameter for the event - can have different meanings and data types dependant on the EventIDs (see CustomExpEventIDs for details)										
var	OutParam	OleVariant	output parameter for the event - can have different meanings and data types dependant on the EventIDs (see CustomExpEventIDs for details)										
-	RESULT	HResult	negative values indicate an error, others ($>=0$) indicate success										

2.1.35 IDIChannel

sepecial interface for digital input channels:

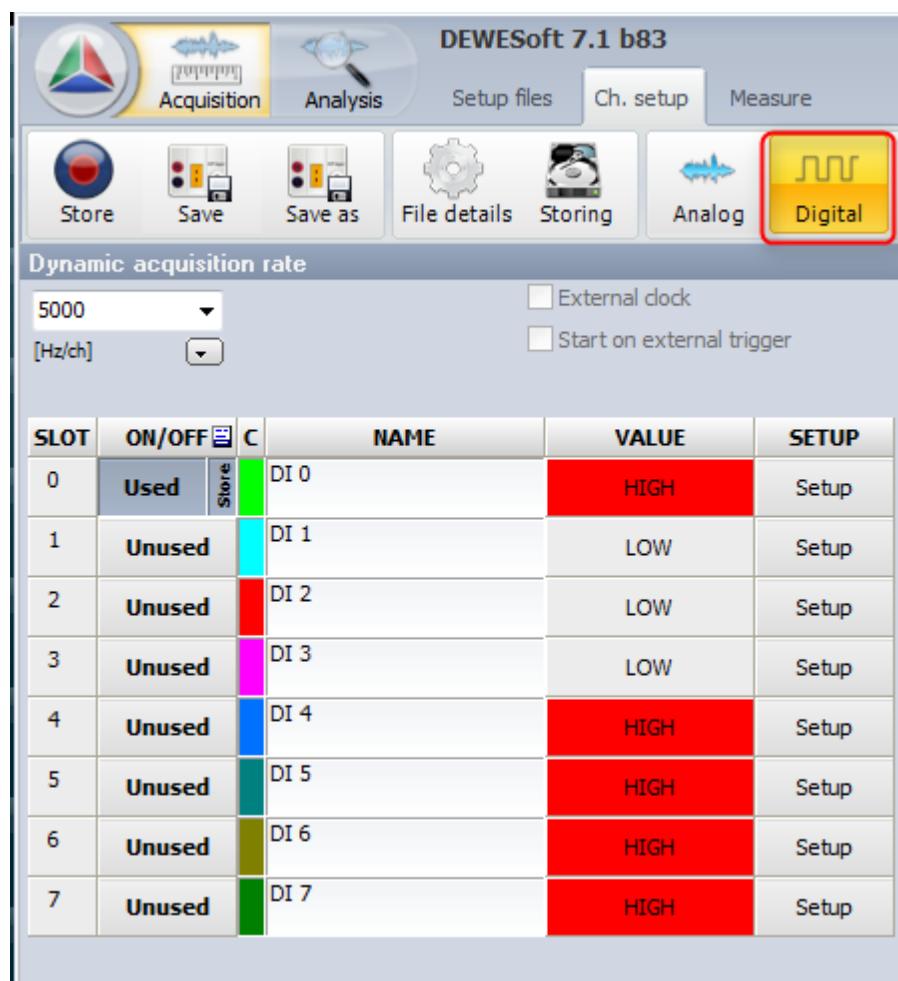


Illustration 168: Digital Input Channels

see also: [IDIGroup](#), [IDIPort](#)

2.1.35.1 DIFilter

property DIFilter: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDIClass](#)  read-only

2.1.35.2 DIInvert

property DIInvert: WordBool

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDIClass](#)  read-only

2.1.35.3 TrigLevels

property TrigLevels: [IDigitalTrigLevel](#)

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDIClass](#)  read-only

2.1.36 IDIGroup

channel group for digital input ports (aka. channels):

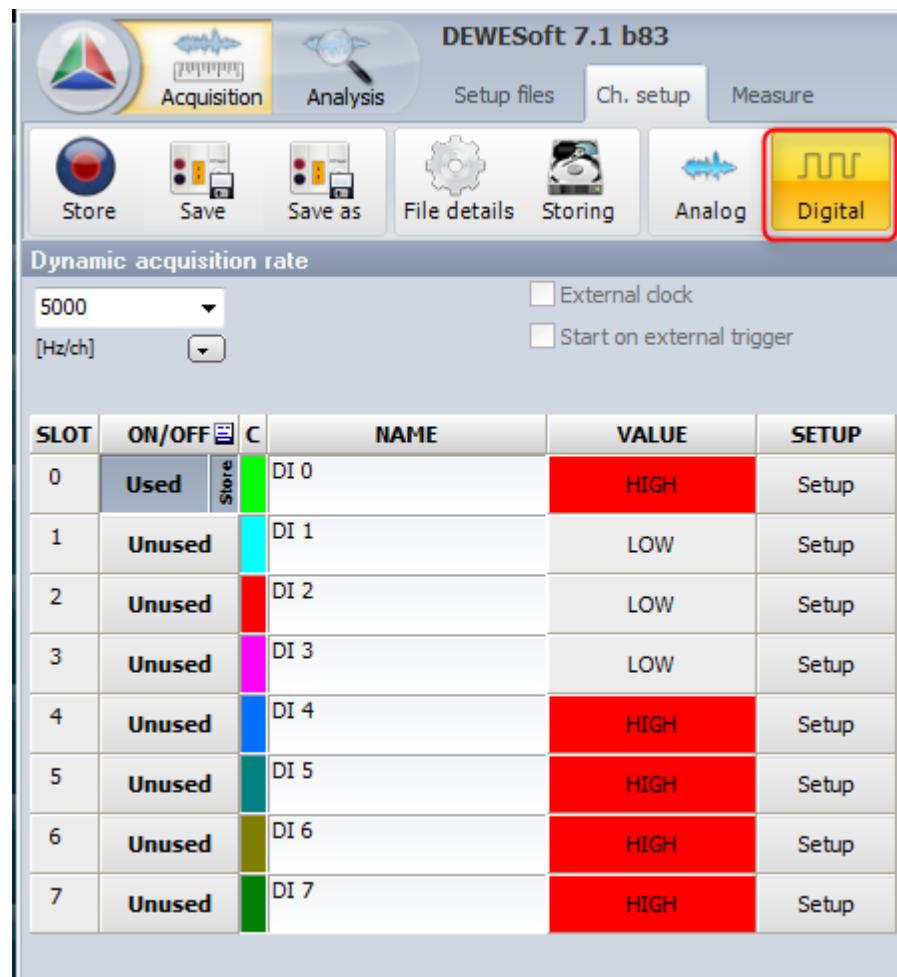


Illustration 169: Digital Input Channels

You can get a reference to this group via [IData.Groups](#) [7].

2.1.36.1 Count

property Count: Integer

Count is the number of channels in the [Item](#) list.

Interface: [IDIGroup](#) read-only

2.1.36.2 Item

```
property Item[Index: Integer]: IDIPort
```

Item[I] is the digital input port at index I. I is in the range of 0...[Count](#)-1.

Interface: [IDIGroup](#)  read-only

2.1.37 IDIPort

see also: [IDIGroup](#)

2.1.37.1 ApplyDBBuf

```
procedure ApplyDBBuf();
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDIPort](#)

2.1.38 IDaq

interface to access the analog measurement devices

see: [IApp.Daq](#)

2.1.38.1 CanAutoCalculate

```
property CanAutoCalculate: WordBool
```

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [IDaq](#)  read-only

2.1.38.2 CardCount

```
property CardCount: Integer
```

the number of analog cards that are activated in hardware setup.

Interface: `IDao`  read-only

2.1.38.3 DagType

property DagType: Integer

the type of the analog device
daqTypeSpectrumM2i = 1;
daqTypeDeweDSA = 2;
daqTypeDT = 3;
daqTypeNiDaqMX = 4;
daqTypeDeweDAQ = 5;
daqTypeDeweUSB = 6;

Interface: [IDaq](#) read-only

2.1.38.4 DataLost

property DataLost: WordBool

If a data lost condition has occurred since the start of the measurement.

Interface: **IDaq** read/write

2.1.38.5 GetCNTTrgLevel

```
function GetCNTTrgLevel  
(CNTNr: Integer; PinType: Byte; out TrgLevel: Integer; out ReTrgLevel: Integer;  
  Coupling: Byte; out SupportLevel: Byte): Integer;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: SetCNTTrgl, eye

Interface: [IDao](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
	CNTNr	Integer										
	PinType	Byte										
out	TrgLevel	Integer										
out	ReTrgLevel	Integer										
out	Coupling	Byte										
out	SupportLevel	Byte										
-	RESULT	Integer										

2.1.38.6 GetDITrgLevel

```
function GetDITrgLevel
  (PinNr: Integer; out TrgLevel: Integer; out ReTrgLevel: Integer; out Coupling: Byte
  ; out SupportLevel: Byte): Integer;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [SetDITrgLevel](#)

Interface: [IDaq](#)

Classifier	Name	Type	Description
	PinNr	Integer	
out	TrgLevel	Integer	
out	ReTrgLevel	Integer	
out	Coupling	Byte	
out	SupportLevel	Byte	
-	RESULT	Integer	

2.1.38.7 GetDeviceCode

```
function GetDeviceCode(Index: Integer): WideString;
```

The device code of the device at the given Index.

e.g. a DEWE-43 will return it's serial number: e.g. D0201A04

Interface: [IDaq](#)

Classifier	Name	Type	Description
	Index	Integer	the index of the device when several devices are used
-	RESULT	WideString	The device code of the device at the given Index.

2.1.38.8 GetDeviceInfo

```
function GetDeviceInfo(Index: Integer): DaqDeviceInfo;
```

returns a device information record about the analog measurement card with the given Index.

see also: [DaqDeviceInfo](#), [CardCount](#)

Interface: [IDaq](#)

Classifier	Name	Type	Description
	Index	Integer	the Index of the analog measurement card
-	RESULT	DaqDeviceInfo	a device information record about the analog measurement card with the given Index

2.1.38.9 IOControl

```
function IOControl
(Msg: Integer; InParam: OleVariant; out OutParam: OleVariant): Integer;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDaq](#)

Classifier	Name	Type	Description
	Msg	Integer	
	InParam	OleVariant	
out	OutParam	OleVariant	
-	RESULT	Integer	

2.1.38.10 SetCNTTrgLevel

```
function SetCNTTrgLevel
(CNTNr: Integer; PinType: Byte; TrgLevel: Integer; ReTrgLevel: Integer; Coupling: Byte): Integer;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [GetCNTTrgLevel](#)

Interface: [IDaq](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	CNTNr	Integer									
	PinType	Byte									
	TrgLevel	Integer									
	ReTrgLevel	Integer									
	Coupling	Byte									
-	RESULT	Integer									

2.1.38.11 SetDITrgLevel

```
function SetDITrgLevel  
(PinNr: Integer; TrgLevel: Integer; ReTrgLevel: Integer; Coupling: Byte): Integer;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [GetDITrgLevel](#)

Interface: [IDaq](#)

Classifier	Name	Type	Description
	PinNr	Integer	
	TrgLevel	Integer	
	ReTrgLevel	Integer	
	Coupling	Byte	
-	RESULT	Integer	

2.1.38.12 SetDeviceCalDate

```
function SetDeviceCalDate  
(Index: Integer; const CalDate: WideString; const Pwd: WideString): Integer;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDaq](#)

Classifier	Name	Type	Description
	Index	Integer	
const	CalDate	WideString	
const	Pwd	WideString	
-	RESULT	Integer	

2.1.39.5 CustomSensorOffset

```
property CustomSensorOffset: Single
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDaqChannel](#)  read/write

2.1.39.6 CustomSensorScale

```
property CustomSensorScale: Single
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDaqChannel](#)  read/write

2.1.39.7 GetBoardOpt

```
function GetBoardOpt(Index: Integer): Word;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDaqChannel](#)

Classifier	Name	Type	Description
	Index	Integer	
-	RESULT	Word	

2.1.39.8 GetSensor

```
function GetSensor(): WideString;
```

Returns the name of the selected sensor.

see also: [SetSensor](#)

Interface: [IDaqChannel](#)

Classifier	Name	Type	Description
-	RESULT	WideString	the name of the selected sensor.

2.1.39.9 GetSensorType

```
function GetSensorType(): WideString;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDaqChannel](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	

2.1.39.10 ModuleGain

property ModuleGain: Double

`ModuleGain` is the gain value set at a module.

Interface: [IDaqChannel](#)  read-only

2.1.39.11 ModuleOffset

property ModuleOffset: Double

`ModuleOffset` is the offset value set at a module.

Interface: [IDaqChannel](#)  read-only

2.1.39.12 ModuleType

property ModuleType: Integer

The type number of the module.

2.1.39.13 SetBoardOpt

```
procedure SetBoardOpt(Index: Integer; Used: Word);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [GetBoardOpt](#)

Interface: [IDaqChannel](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	Index	Integer									
	Used	Word									

2.1.39.14 SetCardGain

```
procedure SetCardGain(Gain: Single);
```

SetCardGain sets the gain of a DAQ card.

see also: [CardGain](#)

Interface: [IDaqChannel](#)

Classifier	Name	Type	Description
	Gain	Single	the new gain value to be set

2.1.39.15 SetSensor

```
function SetSensor(const SensorName: WideString): WordBool;
```

will set the sensor by name.

see also: [GetSensor](#)

Interface: [IDaqChannel](#)

Classifier	Name	Type	Description
const	SensorName	WideString	name of the sensor
-	RESULT	WordBool	TRUE if the sensor has been set, FALSE otherwise

2.1.40 IDaqData

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see also: [IModule.DaqData](#)

2.1.40.1 Address

```
property Address: Smallint
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Address defines the address of the DAQ-module.

Interface: [IDaqData](#)



2.1.40.2 CopyToString

```
function CopyToString(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`CopyToString` copies the module parameters to a string.

Interface: [IDaqData](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	

2.1.40.3 CopyUnitToString

```
function CopyUnitToString(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`CopyUnitToString` copies the unit of measurement to a string.

Interface: [IDaqData](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	

2.1.40.4 CurrentSource

property CurrentSource: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

`CurrentSource` defines the current source of ACC modules.

Interface: [IDagData](#)



2.1.40.5 DaqNames

```
property DaqNNames [ANameCode: Smallint]: WideString
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`DaqNames` provides the name of a DAQN-module referenced by `ANameCode`.

Interface: [IDaqData](#)  read-only

2.1.40.6 DaqNNamesCount

property DaqNNamesCount: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`DaqNNamesCount` is the number of available DAQN-modules.

Interface: [IDaqData](#)  read-only

2.1.40.7 FREQAInputCoupling

property FREQAInputCoupling: Shortint

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`FREQAInputCoupling` defines the input coupling-type of a FREQ-A-module.

Interface: [IDaqData](#)  read/write

2.1.40.8 FREQAOutputFilter

property FREQAOutputFilter: Shortint

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`FREQAOutputFilter` defines the output filter-type of a FREQ-A-module.

Interface: [IDaqData](#)  read/write

2.1.40.9 FREQATriggerLevel

property FREQATriggerLevel: Single

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`FREQATriggerLevel` defines the trigger level of a FREQ-A-module.

Interface: [IDaqData](#)  read/write

2.1.40.10 FilterCode

property FilterCode: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`FilterCode` defines the filter of a DAQ-module.

Interface: [IDaqData](#)  read/write

2.1.40.11 Filters

property Filters[AFilterCode: Byte]: WideString;

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`Filters` provides the name of a filter corresponding to `AFilterCode`.

Interface: [IDagData](#)

2.1.40.12 FiltersCount

property FiltersCount: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`FiltersCount` is the number of filters available for DAQ-modules

Interface: [IDagData](#)

2.1.40.13 HighpassType

property HighpassType: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`HighpassType` defines the type of the highpass-filter set at a DAQ-module. Possible values are dependent on the module type.

Interface: `IDagData`

2.1.40.14 ICPInput

property ICPInput: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

ICPInput defines whether a charge-module is set to ICP or to charge. The meaning is dependant on the module-type.

Interface: [IDaqData](#)  read/write

2.1.40.15 ModuleAmpl

function ModuleAmpl(): Single;

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

ModuleAmpl returns the module scale factor.

Interface: [IDaqData](#)

Classifier	Name	Type	Description
-	RESULT	Single	

2.1.40.16 ModuleError

property ModuleError: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

ModuleError is the error code of a module. Refer to the technical reference of the modules for detailed information.

Interface: [IDaqData](#)  read/write

2.1.40.17 ModuleOffset

function ModuleOffset(): Single;

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

ModuleOffset returns the module offset factor.

Interface: [IDaqData](#)

Classifier	Name	Type	Description
-	RESULT	Single	

2.1.40.18 ModuleType

property ModuleType: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

The type of a module.

Type	Description
2	DAQP-Bridge
5	DAQP-Charge
8	DAQP-Freq
9	DAQP-Acc
23	DAQP-Charge-A
24	DAQP-Bridge-A
26	DAQP-Freq-A
27	DAQP-ACC-A
30	DAQP-Charge-B
31	DAQP-Bridge-B
34	DAQP-V-A
35	DAQP-V-B
36	MDAQ

Interface: [IDaqData](#)



2.1.40.19 Name

property Name: WideString

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Name provides the name of a module.

Interface: [IDaqData](#)



2.1.40.20 Overflow

property Overflow: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Overflow will be 1 if an overflow has occurred.

Interface: [IDaqData](#)  read/write

2.1.40.21 RangeCode

property RangeCode: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`RangeCode` is the code denoting the measurement range of the module. Refer to the technical reference of the modules for detailed information.

Interface: [IDaqData](#)  read/write

2.1.40.22 Ranges

property Ranges[ARangeCode: Integer]: WideString

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`Ranges` allows to retrieve the range settings as a string according to the range code. `ARangeCode` must be in the range of 0... [RangesCount](#)-1.

Interface: [IDaqData](#)  read-only

2.1.40.23 RangesCount

property RangesCount: Integer

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`RangesCount` is the number of valid ranges of a module.

Interface: [IDaqData](#)  read-only

2.1.40.24 Remote

property Remote: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`Remote` denotes whether the module is set to local or remote mode. Refer to the technical reference of the modules for detailed information.



2.1.40.25 ShortCopyToString

```
function ShortCopyToString(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

`ShortCopyToString` copies the module parameters in a short version (only the values) to a string.

Interface: [IDaqData](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	

2.1.40.26 ThermLinearize

```
function ThermLinearize(InputVoltage: Double): Double;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Calculates the thermal linearized value of the input voltage.

Interface: [IDaqData](#)

Classifier	Name	Type	Description
	InputVoltage	Double	
-	RESULT	Double	

2.1.40.27 VRANGE

property VRage: Byte

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Is only used with DAQP-FREQ-module to indicate the trigger voltage input range.

Interface: [IDagData](#)



2.1.41 IDagGroup

is the group of analog input channels

see [IApp DagGroup](#)

2.1.41.1 Count

property Count: Integer

Count is the number of DAQ channels (type: [IDaqChannel](#)) in the Item list.

Interface: [IDaqGroup](#)  read-only

2.1.41.2 Item

property Item[Index: Integer]: [IDaqChannel](#)

Item[I] is the DAQ channel at index I. I is in the range of 0...Count-1.

see also: [IDaqChannel](#)

Interface: [IDaqGroup](#)  read-only

2.1.42 IData

provides the measurement data and corresponding methods and properties

see: [IApp.Data](#)

2.1.42.1 ActiveChannels

property ActiveChannels: [IChannelList](#)

Do not use ActiveChannels because it is implemented for internal use only. Use [AllChannels](#) or [UsedChannels](#) instead!

see also: [AllChannels](#), [UsedChannels](#), [IChannelList](#)

Interface: [IData](#)  read-only

2.1.42.2 AllChannels

property AllChannels: [IChannelList](#)

AllChannels is a list of all available channels according to the hardware setup.

Call [BuildChannelList](#) before reading the property to update it, if some channels have been set to used or not before.

see also: [UsedChannels](#), [ShownChannels](#)

Interface: [IData](#)  read-only

2.1.42.3 AnalyseMode

property AnalyseMode: WordBool

TRUE when you are in Analysis Mode

see also: [MeasureMode](#)

Interface: [IData](#) read-only

2.1.42.4 ApplyChannels

```
procedure ApplyChannels();
```

`ApplyChannels` has to be called after channels of a plug-in have been mounted or unmounted by the plugin.

Note: after the following functions DEWEsoft® will call this function automatically:

- [IPlugin2.NewSetup](#)
 - [IPlugin2.LoadSetup](#)
 - [IPlugin4.OnEvent](#) for event evOnUpdateXML (see also [EventIDs](#)).

see also: [AllChannels](#), [UsedChannels](#), [ShownChannels](#)

Interface: IData

2.1.42.5 BuildChannelList

```
procedure BuildChannelList();
```

`BuildChannelList` updates the properties `AllChannels`, `UsedChannels`, `ShownChannels`.

This procedure should be called before reading one of the properties [AllChannels](#), [UsedChannels](#) and [ShownChannels](#) after some channels have been set to used or unused ([IChannel.Used](#)).

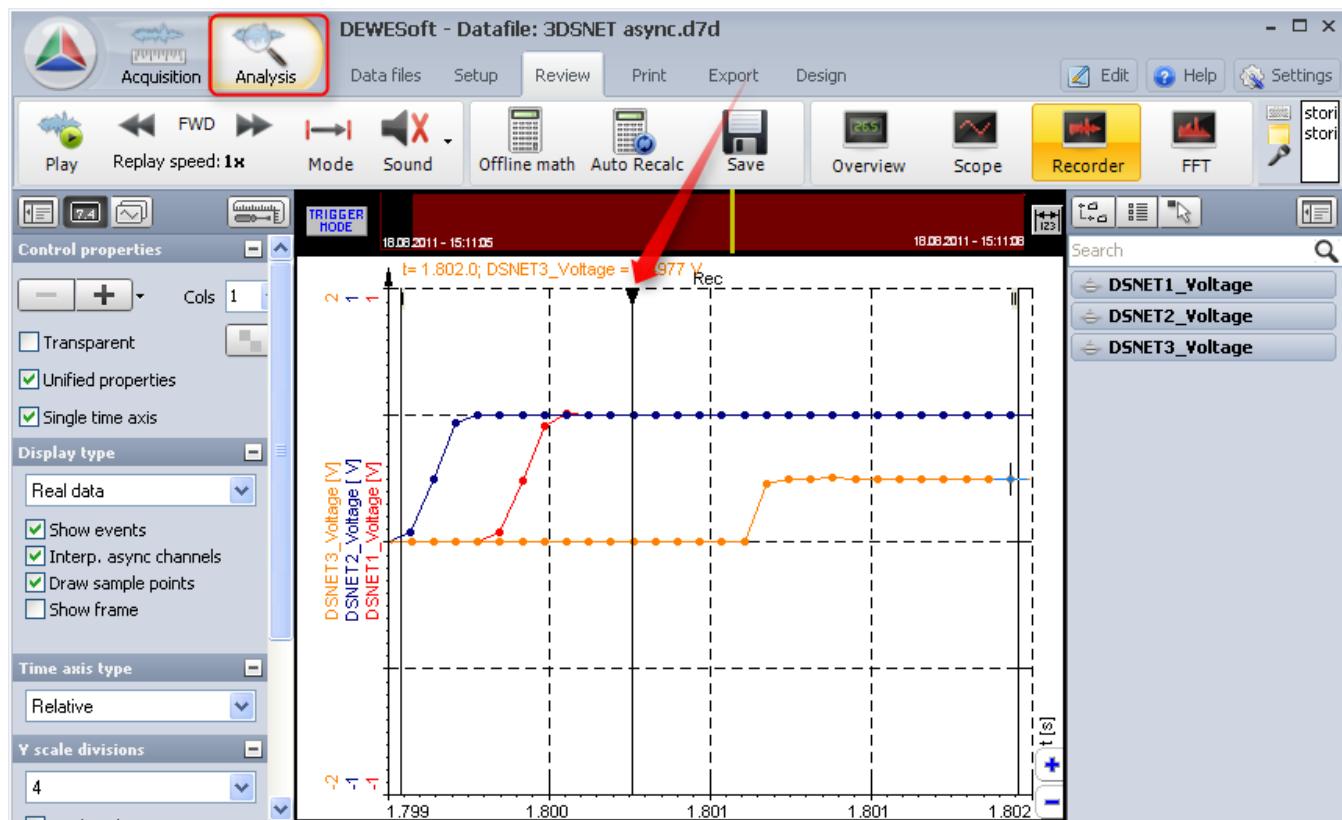
see also: [AllChannels](#), [UsedChannels](#), [ShownChannels](#)

Interface: IData

2.1.42.6 CurrentPos

property CurrentPos: T RecordPosition

CurrentPos is used in analyze mode of DEWEsoft®. It corresponds to the position of the black (or yellow) cursor as shown in the image below.



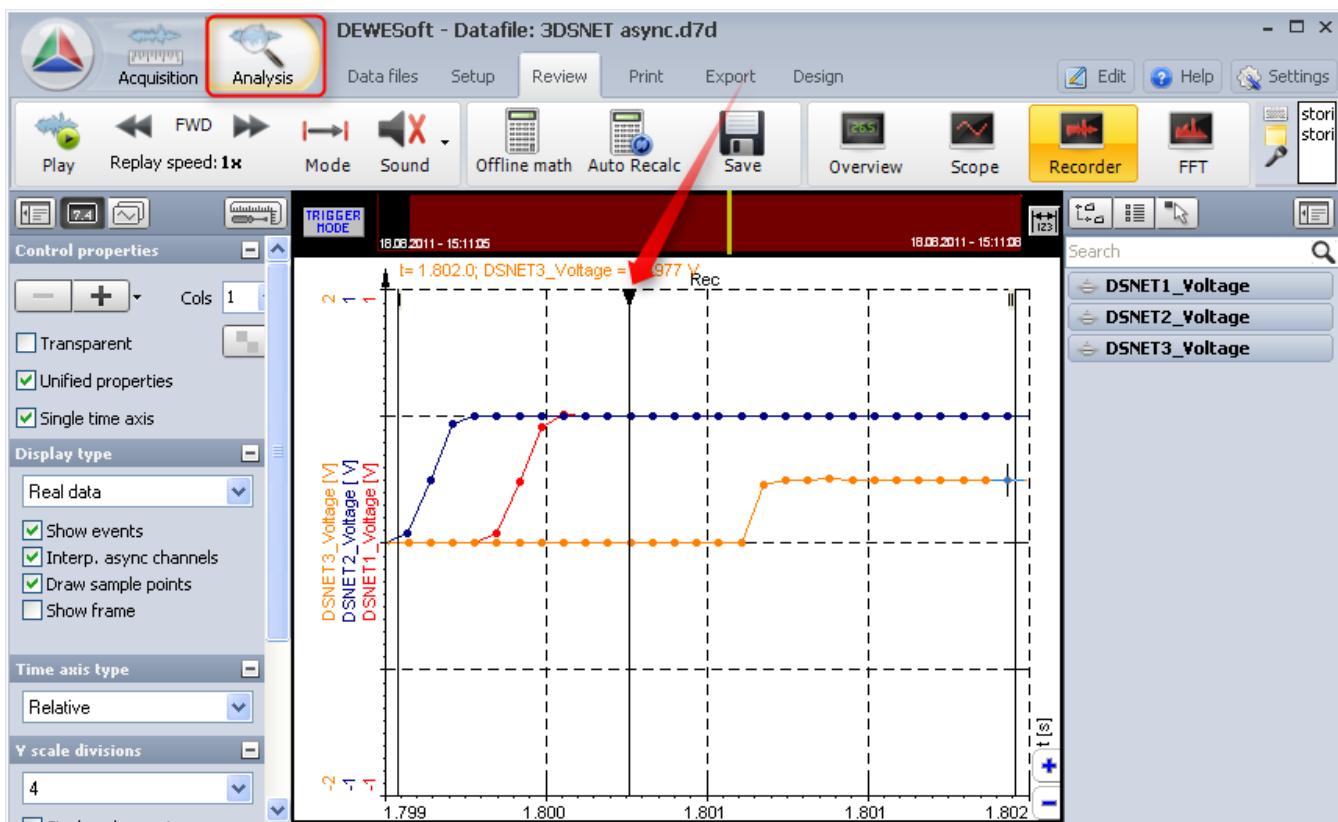
see also: [CurrentPosD](#), [StartStamp](#), [StartStampD](#), [EndStamp](#), [EndStampD](#), [T RecordPosition](#)

Interface: [IData](#)

2.1.42.7 CurrentPosD

property CurrentPosD: Double

CurrentPosD is used in analyze mode of DEWEsoft®. It corresponds to the time in seconds for the black (or yellow) cursor as shown in the image below.



see also: [CurrentPos](#), [StartStamp](#), [StartStampD](#), [EndStamp](#), [EndStampD](#), [T](#), [RecordPosition](#)

Interface: `IData`

2.1.42.8 EndDataSync

```
procedure EndDataSync();
```

Used together with [StartDataSync](#) and [IChannelConnection](#) for synchronous manipulation of multiple channels.

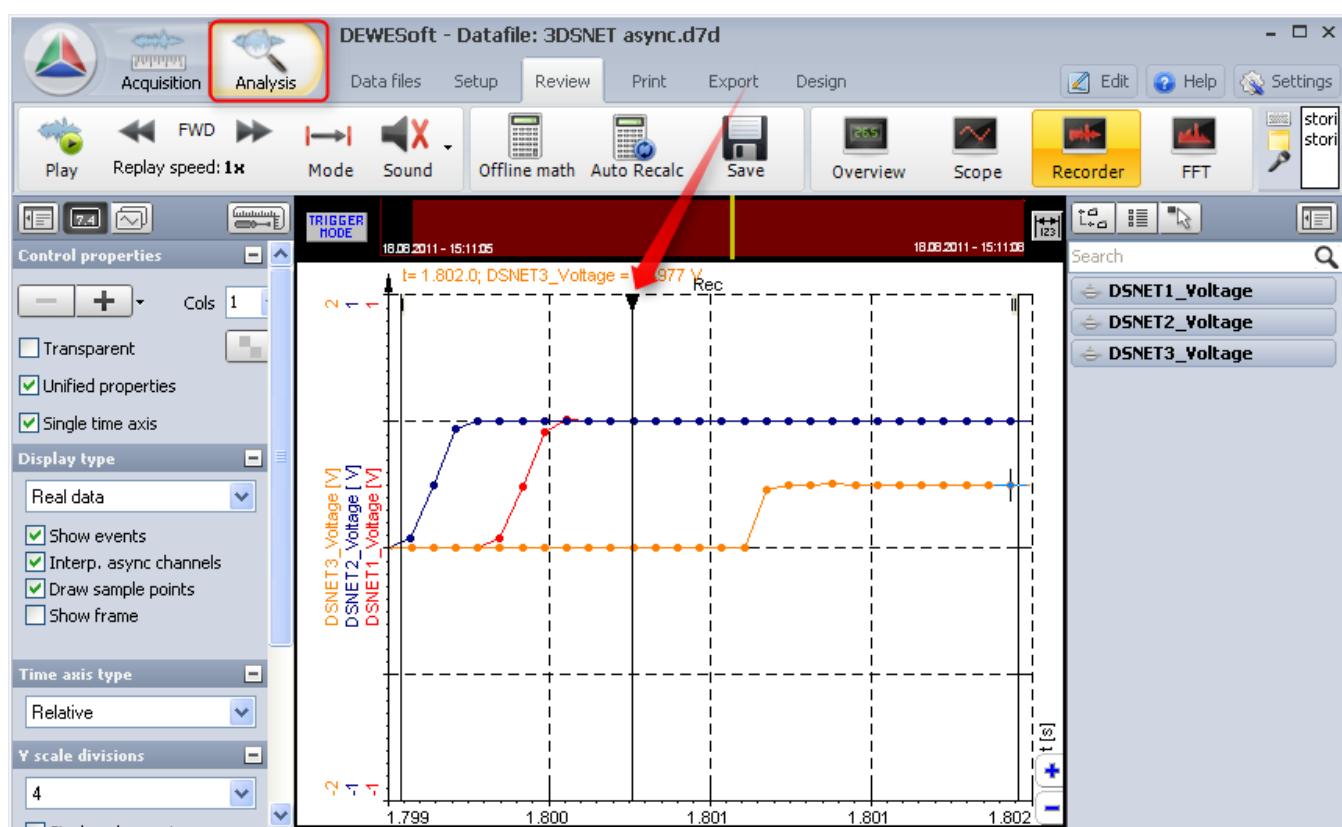
see also: [StartDataSync](#), [IChannelConnection](#)

Interface: IData

2.1.42.9 EndStamp

property EndStamp: T RecordPosition

EndStamp is used in analyze mode of DEWEsoft®. It corresponds to the last possible position of the black (or yellow) cursor as shown in the image below.



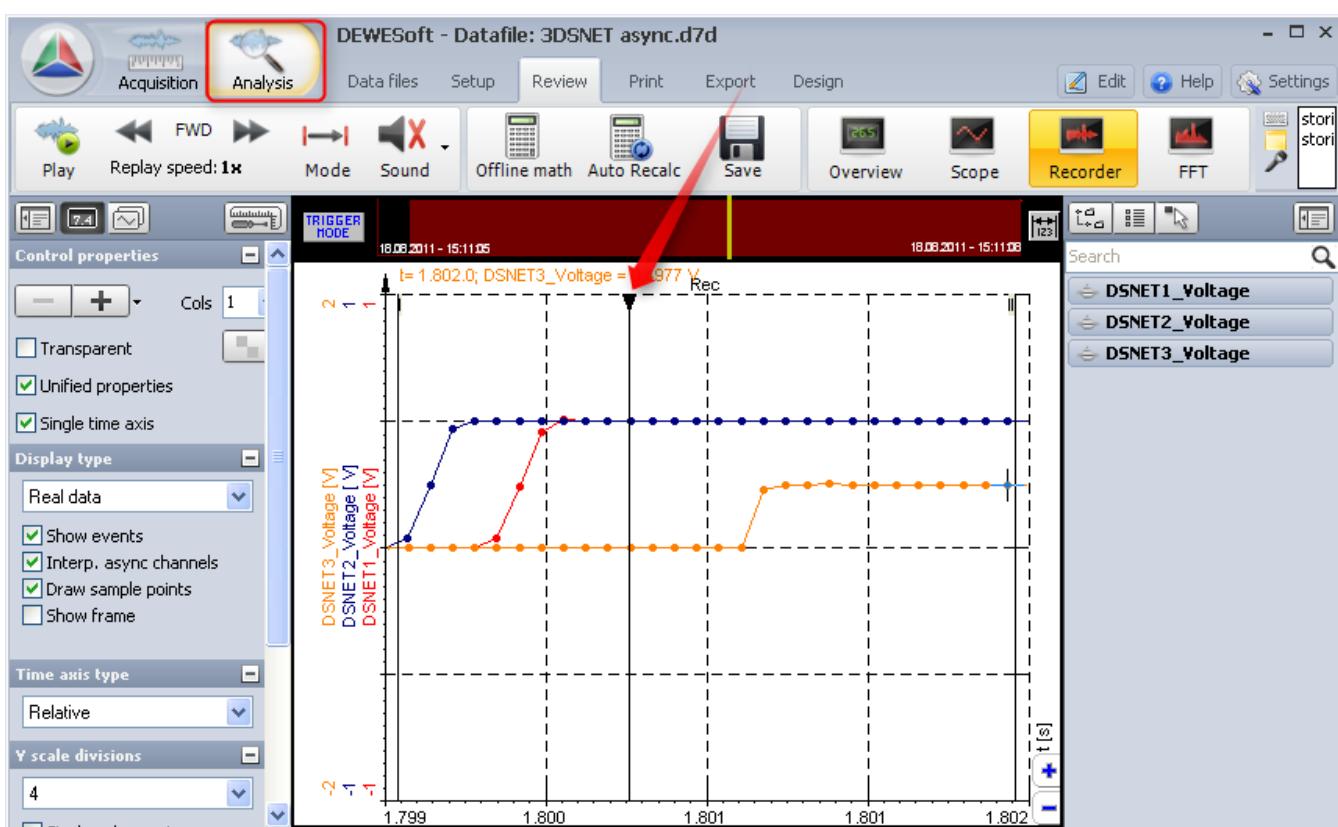
see also: [CurrentPos](#), [CurrentPosD](#), [StartStamp](#), [StartStampD](#), [EndStampD](#), [T_RecordPosition](#), [ILoadEngine.ShrinkFile](#)

Interface: [IData](#) read/write

2.1.42.10 EndStampD

property EndStampD: Double

EndStampD is used in analyze mode of DEWESoft®. It corresponds to the last possible time in seconds for the black (or yellow) cursor as shown in the image below.



see also: [CurrentPos](#), [CurrentPosD](#), [StartStamp](#), [StartStampD](#), [EndStamp](#), [T RecordPosition](#)

Interface: [IData](#) read/write

2.1.42.11 ExternalClock

property ExternalClock: Integer

`ExternalClock` is the external clock definition in clocks per revolution.

Interface: [IData](#) read-only

2.1.42.12 ExternalTrigger

property ExternalTrigger: WordBool

analog (&cnt, etc.) - ch.setup: external clock & externaltrigger: just the value of the checkbox

is TRUE, when an external trigger event has occurred.

Interface: [IData](#)  read/write

2.1.42.13 FindChannel

```
function FindChannel(const Name: WideString): IChannel;
```

To find a channel by its name. Note: several channels can have the same name, thus it is recommended use the [FindChannelByIndexEx](#) function which searches by channel index.

see also: [How To Find Channels](#), [FindChannelByIndexEx](#)

Interface: [IData](#)

Classifier	Name	Type	Description
const	Name	WideString	the name of the requested channel. The name must be exact, the function is case-sensitive.
-	RESULT	IChannel	the found channel or <code>nil</code>

2.1.42.14 FindChannelByIndex

```
function FindChannelByIndex(Index: T\_ChIndex): IChannel;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

USE [IData.FindChannelByIndexEx](#) **INSTEAD** - see also [Channel Index](#).

`FindChannelByIndex` allows finding a channel by its indexes (see [T_ChIndex](#) for more details)

`FindChannelByIndex` does not work in conjunction with LabVIEW. It is recommended to use [FindChannelByIndex1](#) instead.

see also: [FindChannelByIndex1](#), [T_ChIndex](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	Index	T_ChIndex	record with the index definition
-	RESULT	IChannel	the found channel or <code>nil</code>

2.1.42.15 FindChannelByIndex1

```
function FindChannelByIndex1
(IndexLevel: Integer; I1: Integer; I2: Integer; I3: Integer; I4: Integer; I5: Integer): IChannel;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

USE [IData.FindChannelByIndexEx](#) **INSTEAD** - see also [Channel Index](#).

To find a channel by its indexes. This function is similar to [FindChannelByIndex](#). It is implemented because [FindChannelByIndex](#) does not work in conjunction with LabVIEW.

see [T ChIndex](#) for more details on Indexes.

see also: [FindChannelByIndex](#), [T](#), [ChIndex](#), [IChannel](#).[Index](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	IndexLevel	Integer	defines the number of used indexes for specifying a channel. e.g. if IndexLevel=1 then only I1 will be used (I2, etc. will be ignored).
	I1	Integer	the 1st index value
	I2	Integer	the 2nd index value
	I3	Integer	the 3rd index value
	I4	Integer	the 4th index value
	I5	Integer	the 5th index value
-	RESULT	IChannel	the found channel or nil

2.1.42.16 FindChannelByIndexEx

```
function FindChannelByIndexEx(Index: OleVariant): IChannel;
```

To find a channel by its index (see [Channel Index](#)).

214217 FirstTimeStamp

property FirstTimeStamp: T_RecordPosition;

the position of the first timestamp (in Analyse mode it can be different from 0)

see also: [T RecordPosition](#)

Interface: [IData](#) read-only

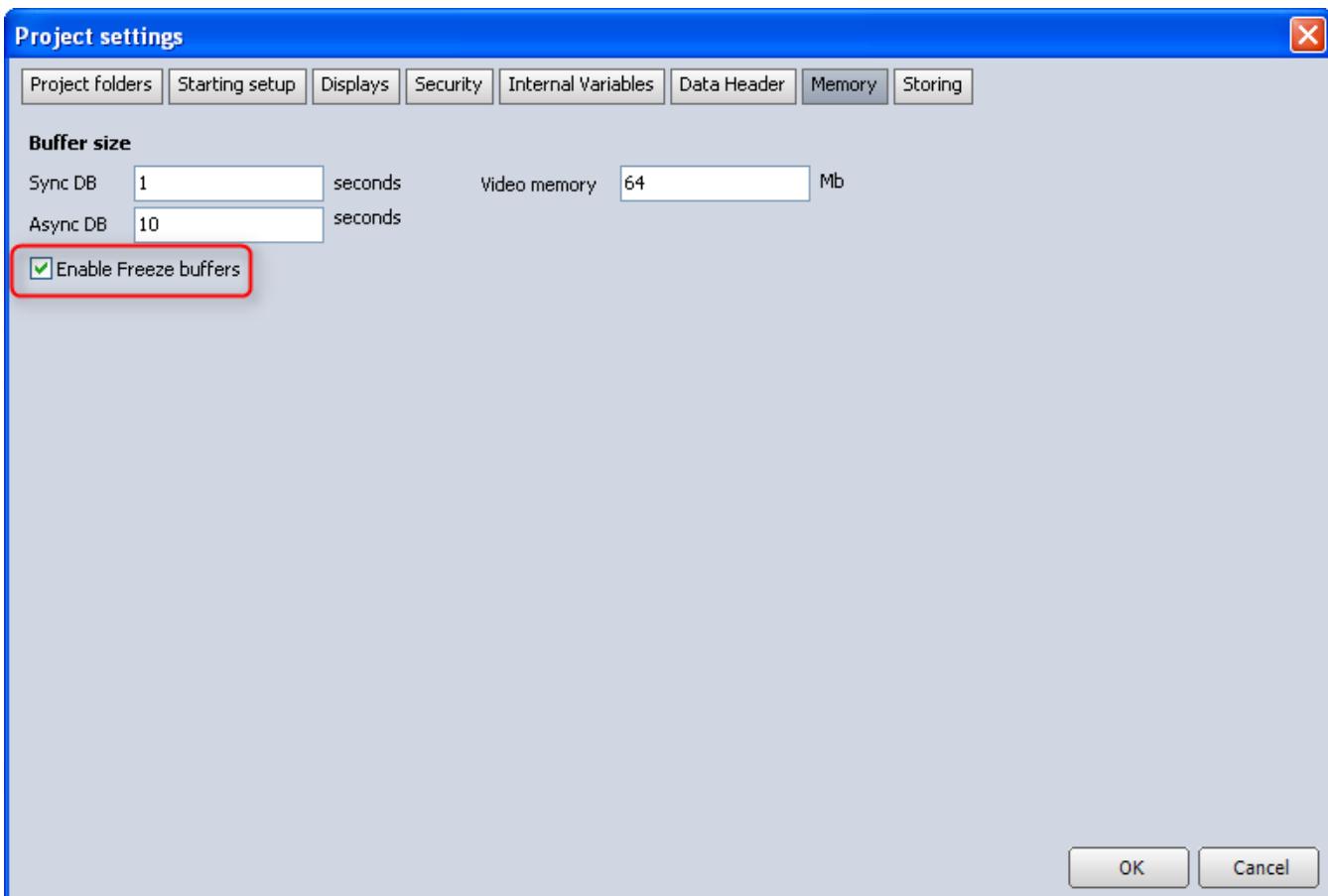
2.1.42.18 FreezeMode

property FreezeMode: WordBool

Will activate or deactivate the *freeze mode*:



Note: the *Freeze* button will only then be enabled when the *Enable Freeze buffers* check box is activated in the *Project settings*.



see also: [IChannel.SetFreezeMode](#)

Interface: [IData](#) read-only

2.1.42.19 GetIndexName

```
function GetIndexName(Index: T\_ChIndex): WideString;
```

GetIndexName returns the identifier specified by Index.

GetIndexName does not work in conjunction with LabVIEW. It is recommended to use [GetIndexName1](#) instead.

see also: [GetIndexName1](#), [GetIndexNameShort](#), [GetIndexNameShort1](#), [T_ChIndex](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	Index	T_ChIndex	the index for which you want to get the name
-	RESULT	WideString	the identifier specified by Index.

2.1.42.20 GetIndexName1

```
function GetIndexName1
(IndexLevel: Integer; I1: Integer; I2: Integer; I3: Integer; I4: Integer; I5: Integer): WideString;
```

This function is similar to [GetIndexName](#), but uses separate integer variables instead of the [T_ChIndex](#) record, so that it also works with LabView.

see also: [GetIndexName](#), [GetIndexNameShort](#), [GetIndexNameShort1](#), [T_ChIndex](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	IndexLevel	Integer	defines the number of used index levels for specifying a channel - this depends on the type of channel (for a detailed description see T_ChIndex), e.g. if IndexLevel=1 then only I1 will be used (I2, etc. will be ignored)
-	I1	Integer	the 1st index value
-	I2	Integer	the 2nd index value
-	I3	Integer	the 3rd index value
-	I4	Integer	the 4th index value
-	I5	Integer	the 5th index value
-	RESULT	WideString	the identifier specified by the function parameters

2.1.42.21 GetIndexNameShort

```
function GetIndexNameShort(ChIndex: T\_ChIndex): WideString;
```

Returns the short identifier of the given index ChIndex.

GetIndexName does not work in conjunction with LabVIEW. It is recommended to use [GetIndexNameShort1](#) instead.

see also: [GetIndexName](#), [GetIndexName1](#), [GetIndexNameShort1](#), [T_ChIndex](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	ChIndex	T_ChIndex	the index for which you want to get the short identifier
-	RESULT	WideString	the short identifier specified by Index.

2.1.42.22 GetIndexNameShort1

```
function GetIndexNameShort1
(IndexLevel: Integer; I1: Integer; I2: Integer; I3: Integer; I4: Integer; I5: Integer): WideString;
```

Returns the short identifier of a certain index. This function is similar to [GetIndexNameShort](#). GetIndexNameShort1 is implemented because [GetIndexNameShort](#) dose not work in conjunction with LabVIEW.

see also: [GetIndexName](#), [GetIndexName1](#), [GetIndexNameShort](#), [T_ChIndex](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	IndexLevel	Integer	defines the number of used index levels for specifying a channel - this depends on the type of channel (for a detailed description see T_ChIndex), e.g. if IndexLevel=1 then only I1 will be used (I2, etc. will be ignored)
	I1	Integer	the 1st index value
	I2	Integer	the 2nd index value
	I3	Integer	the 3rd index value
	I4	Integer	the 4th index value
	I5	Integer	the 5th index value
-	RESULT	WideString	the short identifier specified by the function parameters

2.1.42.23 GetSamplesAcquired

```
procedure GetSamplesAcquired(out Mid: Integer; out Dir: Integer);
```

GetSamplesAcquired returns information about the amount of data acquired.

The total amount of acquired samples can be calculated like this:

TotalNumberOfSamples = Mid * [Samples](#) + Dir

From this you can then also calculate the total time that has passed like this:

SecondsTillStart = TotalNumberOfSamples / [SampleRate](#)

2.1.42.25 IBAbsMidRate

```
property IBAbsMidRate[Level: Integer]: Integer
```

This is a project specific feature.

Interface: [IData](#)  read-only

2.1.42.26 IBAbsRate

```
property IBAbsRate[Level: Integer]: Integer
```

the absolute sample rate of the intermediate buffer up to the given Level.

Interface: [IData](#)  read-only

2.1.42.27 IBLevels

```
property IBLevels: Integer
```

the number of intermediate buffer levels

see: [The Buffer Structure](#)

Interface: [IData](#)  read-only

2.1.42.28 IBRate

```
property IBRate[Level: Integer]: Integer
```

the sample rate of the intermediate buffer at the given Level.

Interface: [IData](#)  read-only

2.1.42.29 InputGroups

```
property InputGroups: IInputGroups
```

a list of input groups. To find a specific input group, you would iterate over all input groups and compare the [Guid](#).

see also: [IInputGroups](#)

Interface: [IData](#)  read-only

2.1.42.30 MRealTimeStamp

property MRealTimeStamp: T RecordPosition

current cursor position

see also: [T_RecordPosition](#)

Interface: [IData](#) read-only

2.1.42.31 MaxCalcDelay

property MaxCalcDelay: Integer

the maximum calculation delay (in samples) of all channel - or in other words: the currently highest calculation delay of all channels.

Note, that this value will/can only be set after all devices / maths / plugins have acquired the data, e.g. at the end of OnGetData cycle.

This means, that when the `OnGetData` function of a plugin is called for the first time, the value of `MaxCalcDelay` will be 0.

Note: if you need the maximum calculation delay of all input channels of your plugin you should just iterate over all those channels and find the maximum [IChannel.CalcDelay](#).

see also: [Calculation Delay](#), [IChannel.CalcDelay](#)

Interface: [IData](#) read-only

2.1.42.32 MeasureMode

property MeasureMode: WordBool

TRUE when you are in Measure Mode

see also: [AnalyseMode](#)

Interface: [IData](#) read-only

2.1.42.33 SRDivLCM

property SRDivLCM: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IData](#)  read-only

2.1.42.34 SampleRate

property SampleRate: Integer

`SampleRate` is the sampling rate not depending on the factor of the “reduced rate”; i.e.

- in channel setup, it will return the [Setup Sample Rate](#)
- in measure mode, it will return the [Dynamic Acquisition Rate](#)

Note: it will never return the [Reduced Sample Rate](#).

see also: [Sample Rates](#), [SampleRateEx](#)

Interface: [IData](#)  read-only

2.1.42.35 SampleRateEx

property SampleRateEx: Double

same as [SampleRate](#) but with Double precision: e.g. 10.5 Hz

see also: [Sample Rates](#), [SampleRate](#)

Interface: [IData](#)  read-only

2.1.42.36 Samples

property Samples: Integer

`Samples` is the number of data samples within one data block (for a synchronous channel without sample rate divider: i.e. [SRDiv](#)=1).

see also: [Sample Rates](#), [GetSamplesAcquired](#)

Interface: [IData](#)  read-only

2.1.42.37 SetExternalClock

```
procedure SetExternalClock(Value: Integer);
```

to set an external clock .

Interface: [IData](#)

Classifier	Name	Type	Description
	Value	Integer	rate of the external clock in Hz 0 means: no external clock

2.1.42.38 SetStartStoreTimeUTC

```
procedure SetStartStoreTimeUTC(Time: TDateTime);
```

will set the UTC time of the start of storing.

see also: [DateTime](#), [StartStoreTimeUTC](#)

Interface: [IData](#)

Classifier	Name	Type	Description
	Time	TDateTime	The start of the store time

2.1.42.39 ShownChannels

```
property ShownChannels: IChannelList
```

ShownChannels is **not** recommended to use!

The list of channels which can be shown (e.g. CAN-bus messages cannot be shown).

Use [AllChannels](#) or [UsedChannels](#) instead!

ShownChannels is a list of the channels being set to *Used* (see [IChannel.Used](#)), but this property is accurate only when the application is in measure mode and the setup screen has already been left.

see also: [AllChannels](#), [UsedChannels](#), [IChannel.Shown](#), [IChannelList](#)

Interface: [IData](#)  read-only

2.1.42.40 StartDataSync

```
procedure StartDataSync();
```

Used together with [EndDataSync](#) and [IChannelConnection](#) for synchronous manipulation of multiple channels.

Make sure that [EndDataSync](#) is always called after this call: e.g. use a `try-finally` block.

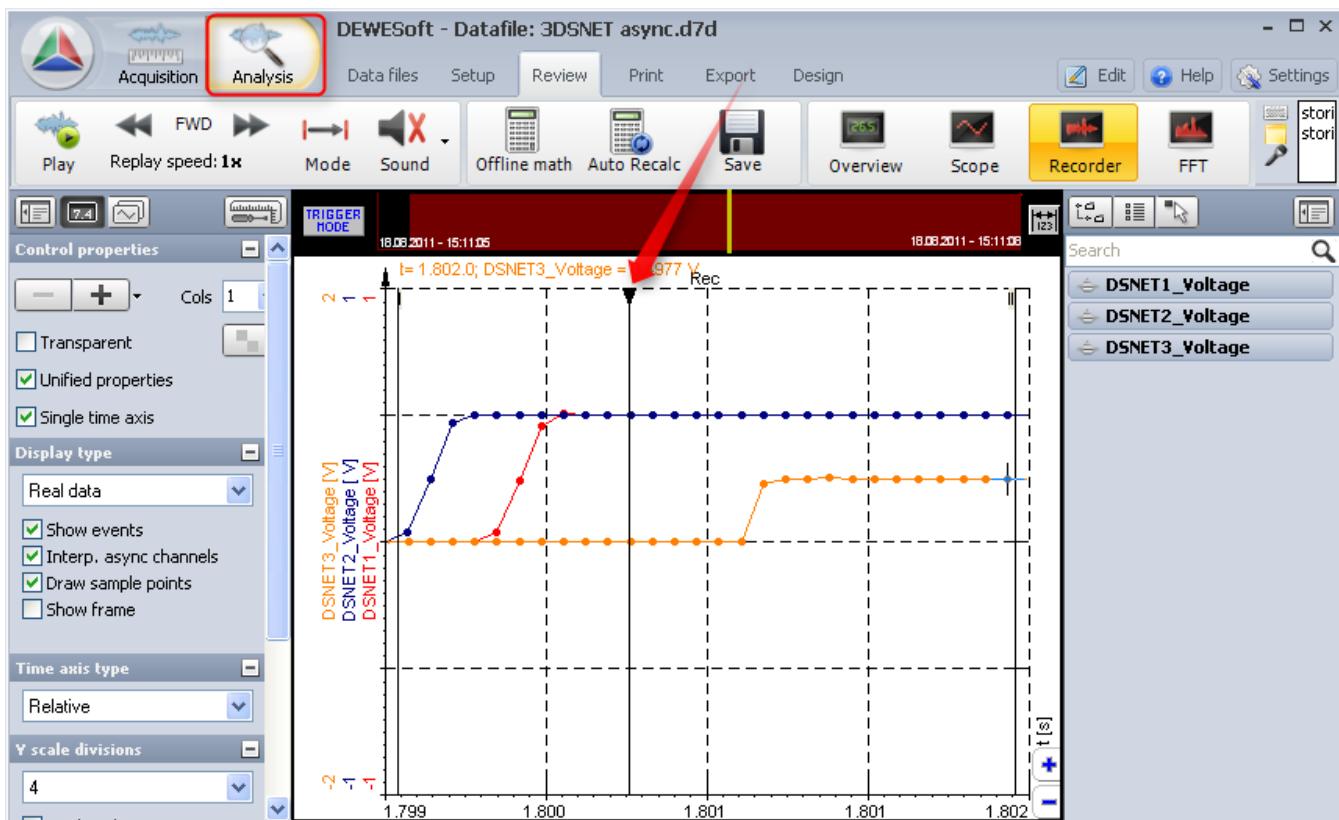
see also: [EndDataSync](#), [IChannelConnection](#)

Interface: [IData](#)

2.1.42.41 StartStamp

property StartStamp: T RecordPosition

StartStamp is used in analyze mode of DEWEsoft®. It corresponds to the first possible position of the black (or yellow) cursor as shown in the image below.



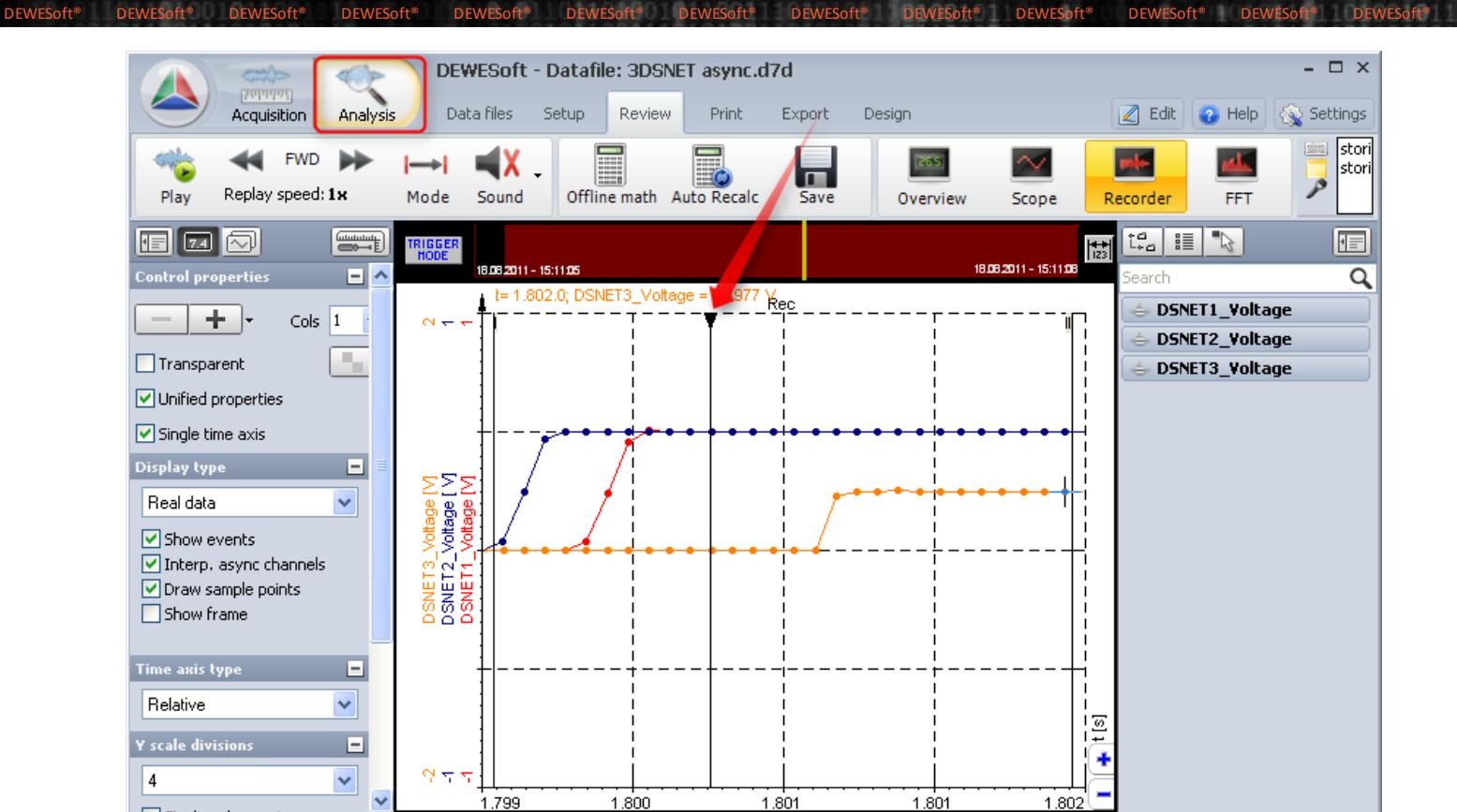
see also: [CurrentPos](#), [CurrentPosD](#), [StartStampD](#), [EndStamp](#), [EndStampD](#), [T_RecordPosition](#), [ILoadEngine](#), [ShrinkFile](#)

Interface: `IData` `read/write`

2.1.42.42 StartStampD

property StartStampD: Double

StartStampD is used in analyze mode of DEWEsoft®. It corresponds to the first possible time in seconds for the black (or yellow) cursor as shown in the image below.



see also: [CurrentPos](#), [CurrentPosD](#), [StartStamp](#), [EndStamp](#), [EndStampD](#), [T_RecordPosition](#)

Interface: [IData](#) read/write

2.1.42.43 StartStoreTime

property StartStoreTime: [TDateTime](#)

StartStoreTime is the absolute time of the start of storing.

see also: [DateTime](#)

Interface: [IData](#) read-only

2.1.42.44 StartStoreTimeUTC

property StartStoreTimeUTC: [TDateTime](#)

StartStoreTimeUTC is the UTC time of the start of storing.

This value should only be read in the [IPlugin2.OnGetData](#) or [IPlugin2.OnStopAcq](#) functions.

see also: [Start Events](#), [DateTime](#), [IMasterClock.GetCurrentTime](#)

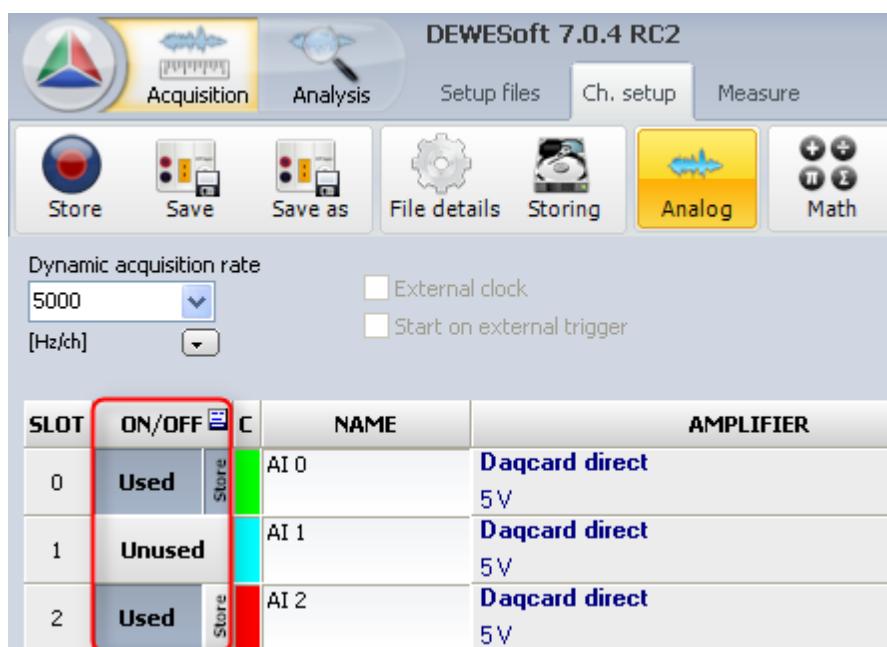
Interface: [IData](#) read-only

2.1.42.45 UsedChannels

property UsedChannels: [IChannelList](#)

A list of the channels which are set to used in the channel setup.

If the status of used has changed (from *Used* to *Unused* or vice versa), then you must call [BuildChannelList](#) to update this list.



see also: [How To Find Channels](#), [IChannel.Used](#), [AllChannels](#), [ShownChannels](#), [IChannelList](#)

Interface: [IData](#) read-only

2.1.43 IDataSection

A data section is the part between a start and a stop event. The data section is independent of the data blocks.

see also: [IDataSections](#), [ILoadEngine.DataSections](#)

2.1.43.1 DataCount

property DataCount: Integer

DataCount is the number of samples within a DataSection.

Interface: [IDataSection](#) read-only

2.1.43.2 ReadData

```
function ReadData(const Channel: IChannel; out Timestamps: OleVariant): OleVariant;
```

`ReadData` reads the whole set of data of a data section.

The data is returned as an `OleVariant` containing an array of data values of the type `Single`.

see also: [ReadData1](#)

Interface: [IDataSection](#)

Classifier	Name	Type	Description
const	Channel	IChannel	channel of which data should be retrieved
out	Timestamps	OleVariant	an OleVariant containing an array of timestamp data of the section. Timestamps are of the type Double
-	<i>RESULT</i>	OleVariant	the resulting data: as an OleVariant containing an array of data values of the type Single

2.1.43.3 ReadData1

```
procedure ReadData1(const Channel: IChannel  
; out Data: OleVariant; out Timestamps: OleVariant);
```

`ReadData1` is similar to `ReadData` but is implemented as a procedure instead of a function.

see also: [ReadData](#)

Interface: [IDataSection](#)

2.1.43.4 Time

property Time: TDateTime

Time is the absolute start time of a section.

Interface: [IDataSection](#)

2.1.43.5 TriPos

property TrigPos: Integer;

The sample position of the trigger event within a section, counted from the start of the section.

Interface: [IDataSection](#)

2.1.44 IDatasections

a list of data sections (see [IDataSection](#))

e.g. when you do a triggered acquisition then the data file will have one data section whenever the store-trigger condition becomes true.

see also: [ILoadEngine.DataSections](#)

2.1.44.1 Count

property Count: Integer

Count is the number of data sections (in the [Item](#) list) where measurement data is stored.

Interface: [IDatasections](#)  read-only

2.1.44.2 Item

property Item[Index: Integer]: IDatasection

Item[I] is the data section at index I. I is in the range of 0...[Count](#)-1.

see also: [IDataSection](#)

Interface: [IDatasections](#)  read-only

2.1.45 IDewePlugin

This interface is planned to be the only one required for all new add-ons. Currently only math ([Custom Mathematics](#)) and visual control ([Custom Visual Controls](#)) add-ons are supported.

2.1.45.1 OnMessage

```
function OnMessage(Msg: Integer; InParam: OleVariant; var OutParam: OleVariant): HResult;
```

the OnMessage function is used to pass messages and input parameters to DEWEsoft® and read back output parameters.

see also: [CustomMathMessages](#), [CustomVCMessages](#)

Interface: [IDewePlugin](#)

Classifier	Name	Type	Description
	Msg	Integer	the message ID
	InParam	OleVariant	depending on the message type (Msg parameter)
var	OutParam	OleVariant	depending on the message type (Msg parameter)
-	RESULT	HResult	negative values indicate an error, others ($>= 0$) indicate success

2.1.46 IDigitalTrigLevel

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.46.1 Coupling

property Coupling: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDigitalTrigLevel](#) read-only

2.1.46.2 ReTrigLevel

property ReTrigLevel: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDigitalTrigLevel](#)  read-only

2.1.46.3 TrigLevel

property TrigLevel: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDigitalTrigLevel](#) read-only

2.1.46.4 TrigType

```
property TrigType: Integer
```

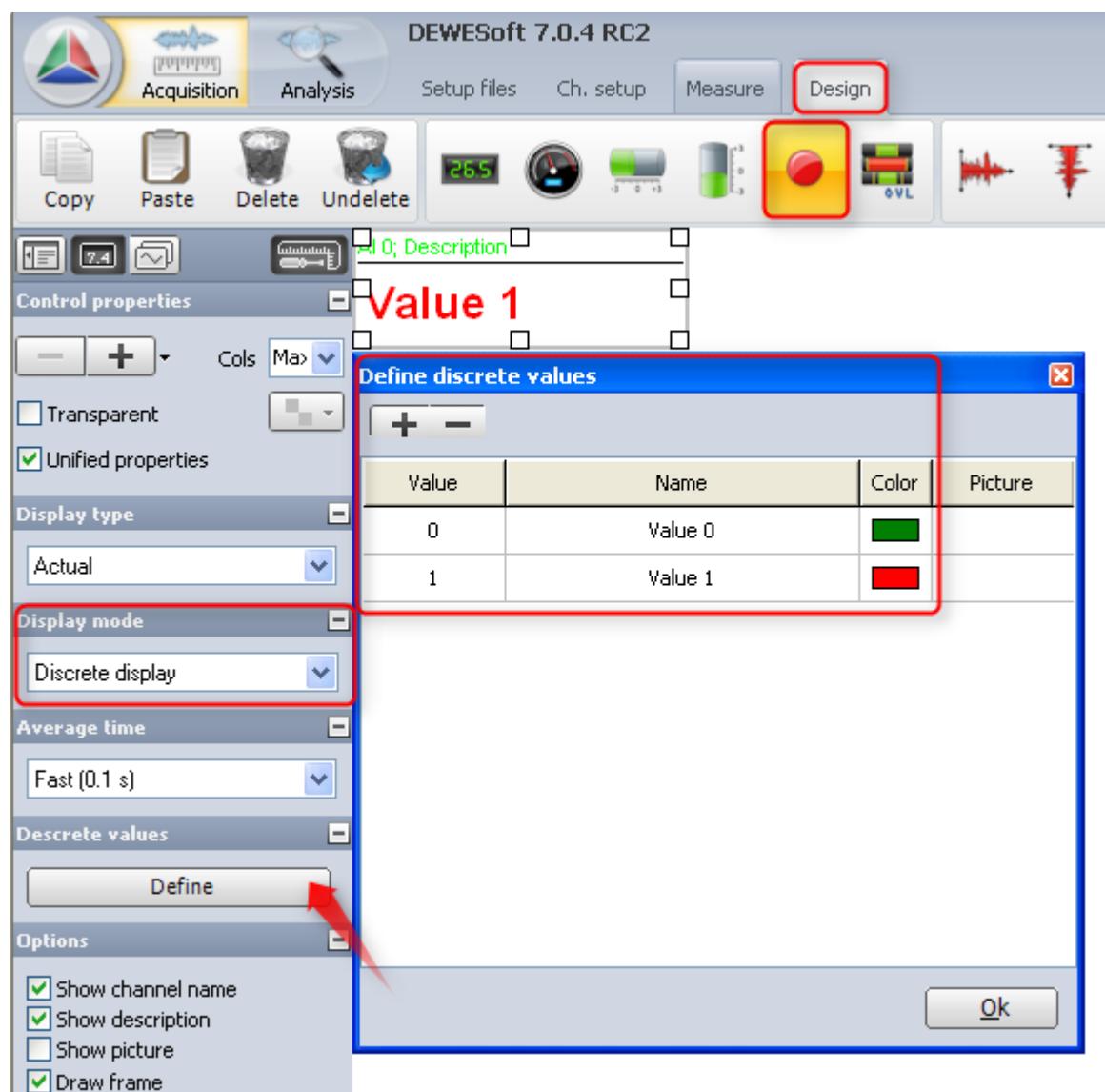
Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDigitalTrigLevel](#) read-only

2.1.47 IDiscreteItem

A single item in a [IDiscreteList](#).

In the image below you can see 2 discrete items.

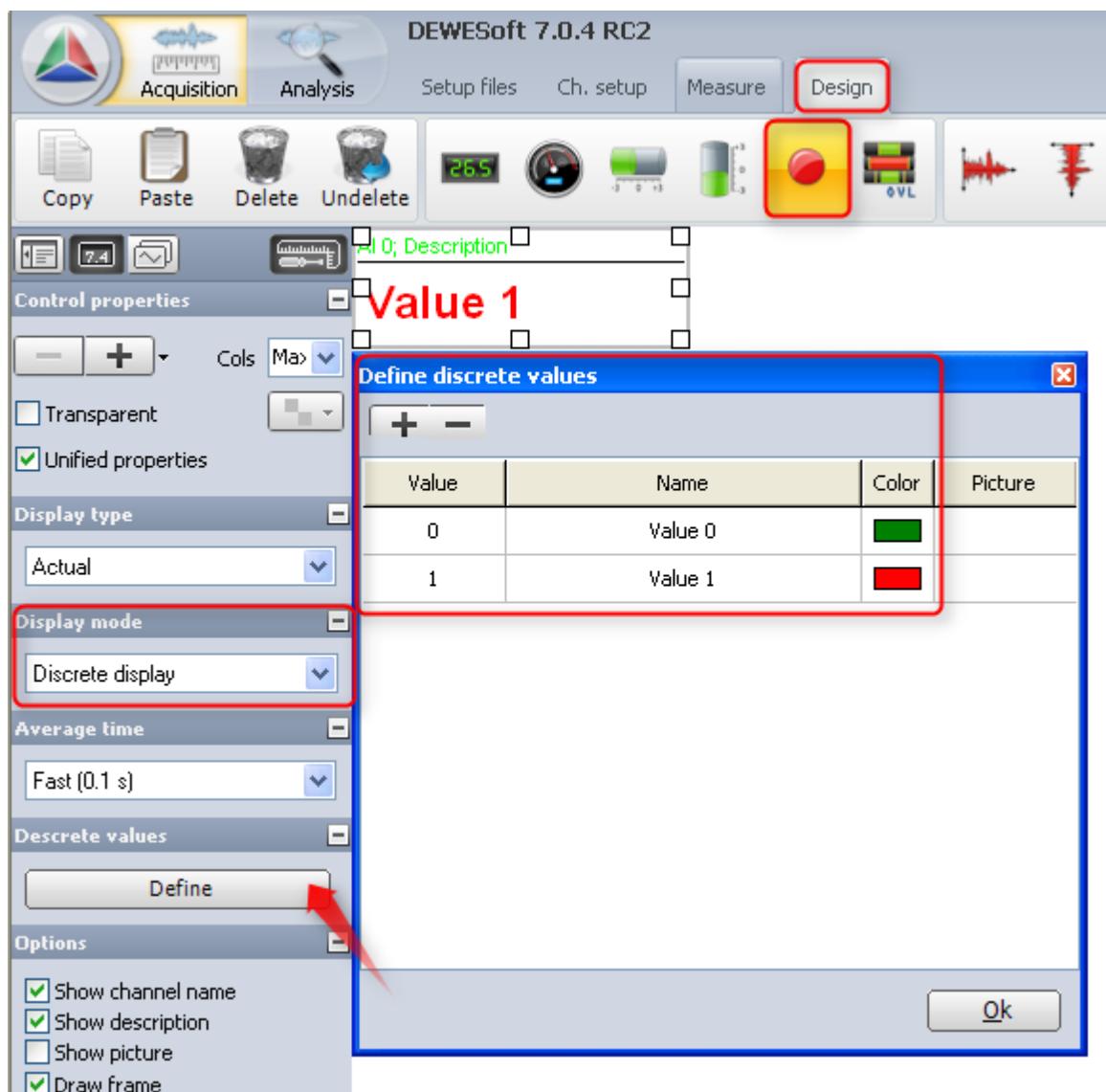


2.1.47.1 Caption

property Caption: WideString

the caption of a discrete item.

In the illustration below you can see 2 discrete items with the captions (aka. Name): **Value 0**, **Value 1**



see also: [IDiscreteList](#)

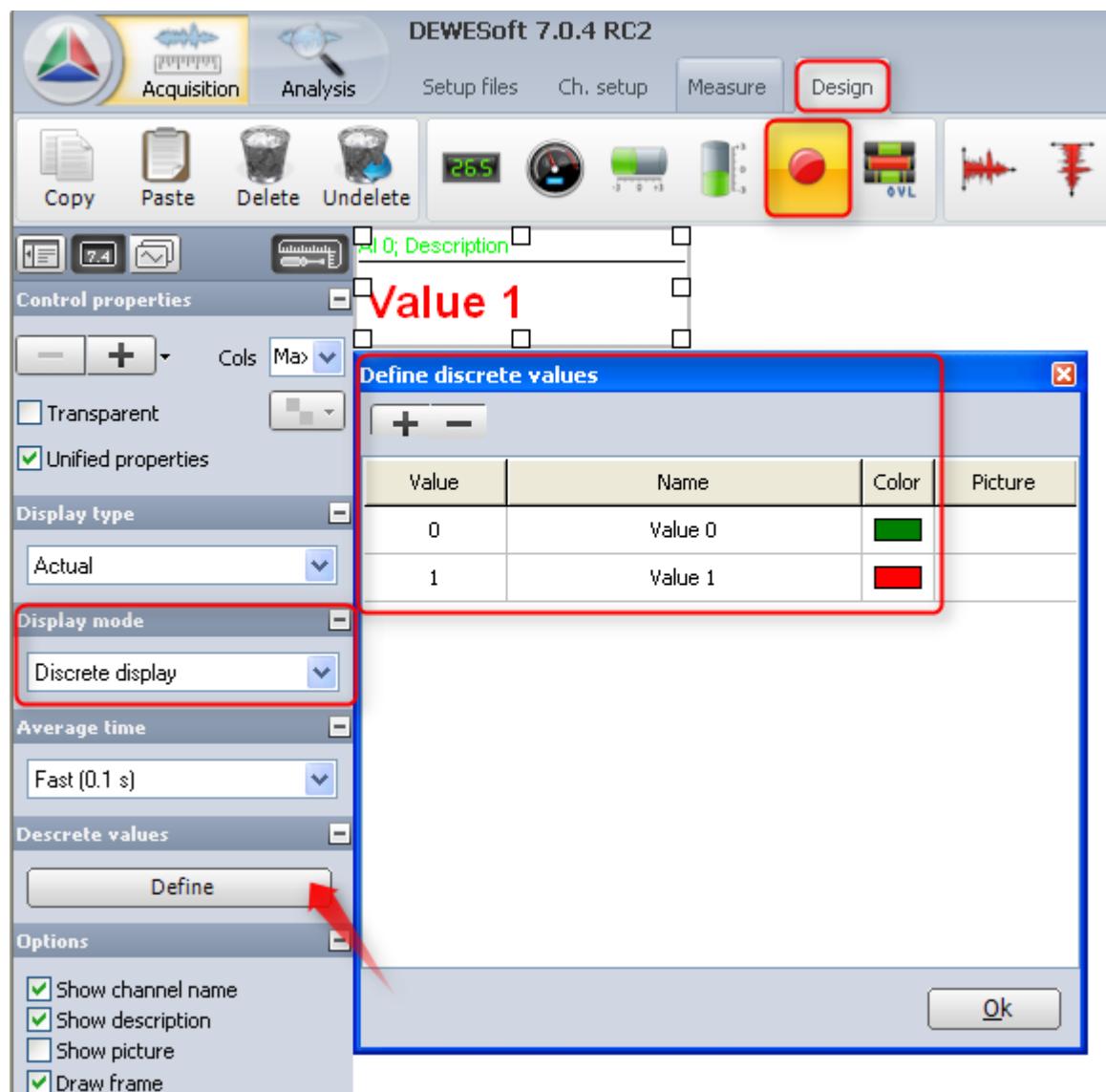
Interface: [IDiscreteItem](#) read/write

2.1.47.2 Color

property Color: Integer

the colour of a discrete item.

In the illustration below you can see 2 discrete items with the colours (column: *Color*): green, red



see also: [IDiscreteList](#)

Interface: [IDiscreteItem](#) read/write

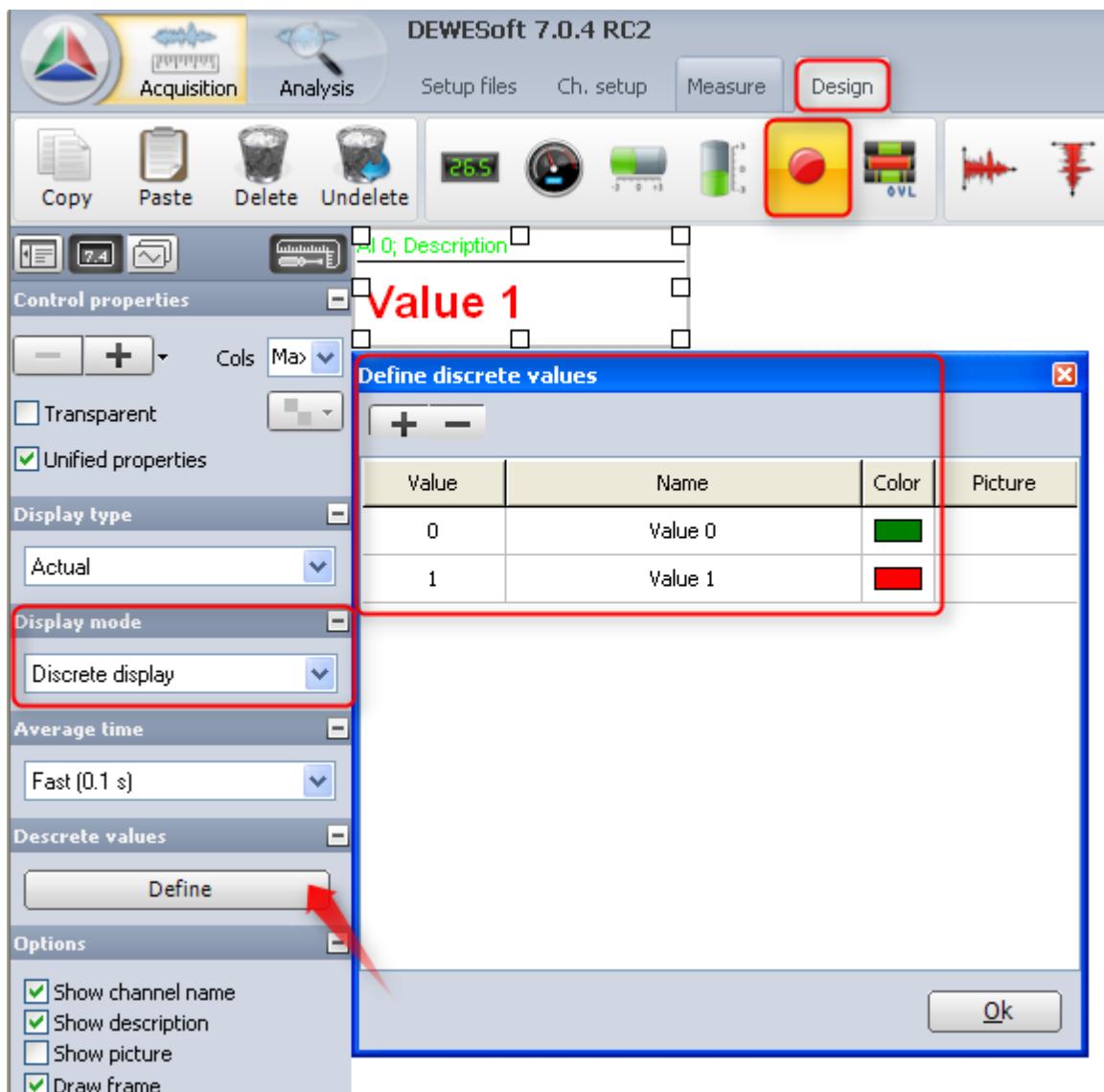
2.1.47.3 Value

property Value: Integer

the value of a discrete item.

When the assigned channel has this value, the caption will be displayed in the graphical display instead.

In the illustration below you can see 2 discrete items with the colours (column: Value): 0, 1



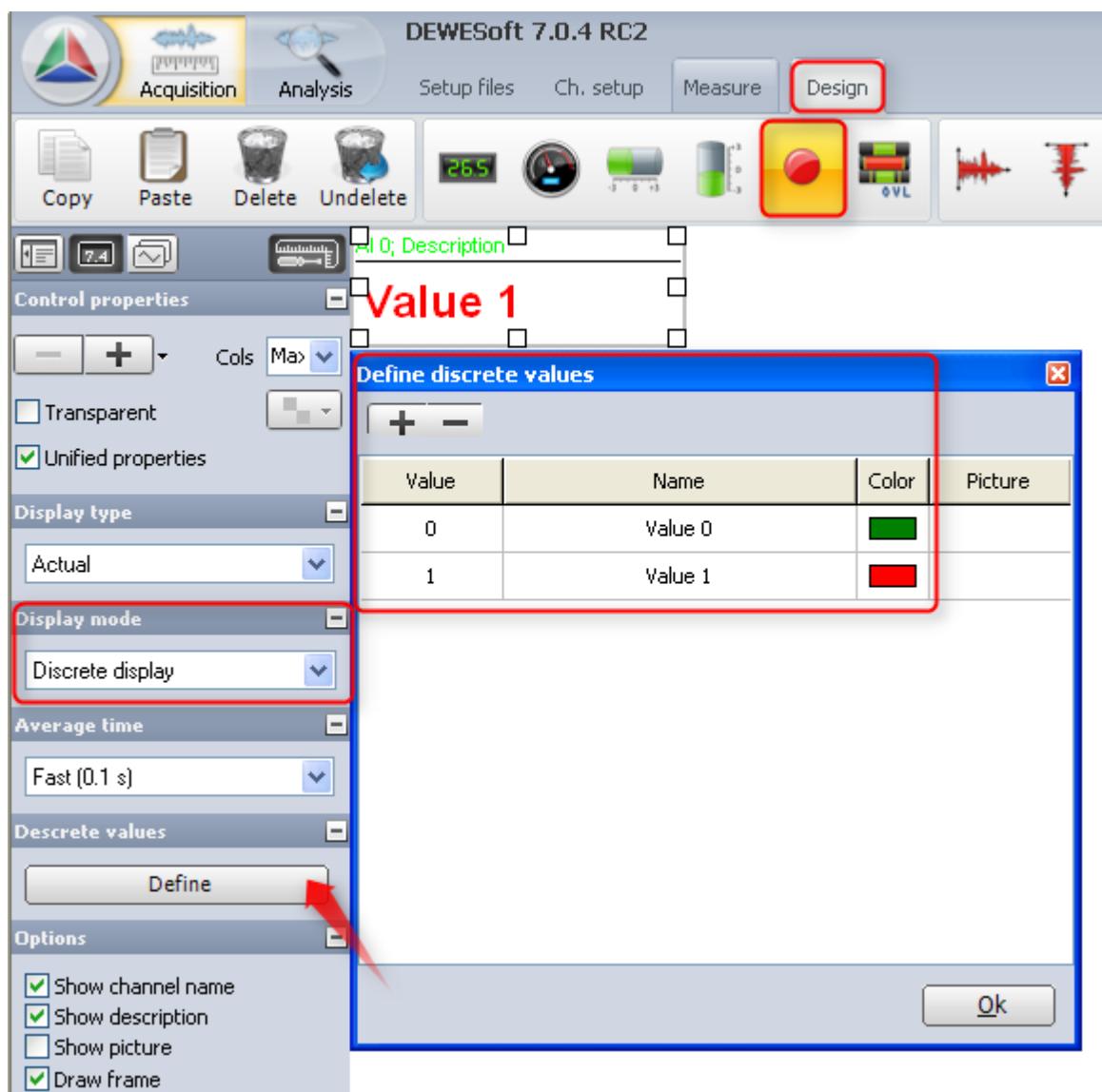
see also: [IDiscreteList](#)

Interface: [IDiscreteItem](#) read/write

2.1.48 IDiscreteList

A list of discrete items that you can use for display. e.g. instead of displaying the value 0 you may want to display the text OFF and for a value of 1 the text ON.

see also: [IChannel_DiscreteList](#), [IDiscreteItem](#)



2.1.48.1 Add

```
function Add(): IDiscreteItem;
```

will add a discrete item to the list

see also: [IDiscreteList](#), [IDiscreteItem](#)

Interface: [IDiscreteList](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier			Name			Type			Description		
-	<i>RESULT</i>		IDiscreteItem			the new item that will be added to the list					

2.1.48.2 Count

property Count: Integer

Count is the number of events in the [Item](#) list.

Interface: [IDiscreteList](#)  read-only

2.1.48.3 Find

function Find(Val: Integer): [IDiscreteItem](#);

will try to find a discrete item with this value (see also: [IDiscreteItem.Value](#)) or nil if no such item is found.

see also: [IDiscreteItem](#)

Interface: [IDiscreteList](#)

Classifier	Name	Type	Description
	Val	Integer	the value of the item you are searching for (see also: IDiscreteItem.Value)
-	<i>RESULT</i>	IDiscreteItem	the discrete item with the desired value Val (see also: IDiscreteItem.Value) or nil if no such item is found

2.1.48.4 Item

property Item[Index: Integer]: [IDiscreteItem](#)

Item[I] is the event at index I. I is in the range of 0...[Count](#)-1.

see also: [IDiscreteItem](#)

Interface: [IDiscreteList](#)  read-only

2.1.48.5 Remove

procedure Remove(Ind: Integer);

will remove the discrete item with the given index Ind from the list.

Interface: [IDiscreteList](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
	Ind	Integer	the index of the item that you wan to remove Ind must be in the range of 0... Count-1									

2.1.49 IDisplayFrameTemplate

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.49.1 AddChannel

```
procedure AddChannel(GraphNode: OleVariant; Ind: Integer; const Ch: IChannel);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)

Classifier	Name	Type	Description
	GraphNode	OleVariant	
	Ind	Integer	
const	Ch	IChannel	

2.1.49.2 AddItemChannel

```
procedure AddItemChannel
(GraphNode: OleVariant; Ind: Integer; const Ch: IChannel; ItemInd: Integer);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)

Classifier	Name	Type	Description
	GraphNode	OleVariant	
	Ind	Integer	
const	Ch	IChannel	
	ItemInd	Integer	

2.1.49.3 CreateCustomGroupAndControl

```
function CreateCustomGroupAndControl
(const CustomVCGuid: WideString; const GroupName: WideString; const ControlName: WideString;
GroupInd: Integer; Left: Single; Width: Single; Top: Single; Height: Single): OleVariant;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)

Classifier	Name	Type	Description
const	CustomVCGuid	WideString	
const	GroupName	WideString	
const	ControlName	WideString	
	GroupInd	Integer	
	Left	Single	
	Width	Single	
	Top	Single	
	Height	Single	
-	RESULT	OleVariant	

2.1.49.4 CreateGroupAndControl

```
function CreateGroupAndControl
(const GroupName: WideString; const ControlName: WideString; GroupInd: Integer; Left: Single; Width: Single; Top: Single; Height: Single): OleVariant;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)

Classifier	Name	Type	Description
const	GroupName	WideString	
const	ControlName	WideString	
	GroupInd	Integer	
	Left	Single	
	Width	Single	
	Top	Single	
	Height	Single	
-	RESULT	OleVariant	

2.1.49.5 GroupName

```
property GroupName: WideString
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)  read/write

2.1.49.6 SetupDOMDoc

```
property SetupDOMDoc: OleVariant
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)  read-only

2.1.49.7 TemplateName

```
property TemplateName: WideString
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplate](#)  read/write

2.1.50 IDisplayFrameTemplates

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.50.1 Add

```
function Add(): IDisplayFrameTemplate;
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplates](#)

Classifier	Name	Type	Description
-	RESULT	IDisplayFrameTemplate	

2.1.50.2 Clear

```
procedure Clear();
```

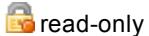
Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplates](#)

property Count: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayFrameTemplates](#)



2.1.50.4 Item

property Item[Index: Integer]: IDisplayFrameTemplate

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: IDisplayFrameTemplates



2.1.51 |DisplayTemplate

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2.1.51.1 PH

property DH: Integer;

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayTemplate](#)



2.1.51.2 DW

property DW: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayTemplate](#)



2.1.51.3 DisplayFrameTemplates

```
property DisplayFrameTemplates: IDisplayFrameTemplates
```

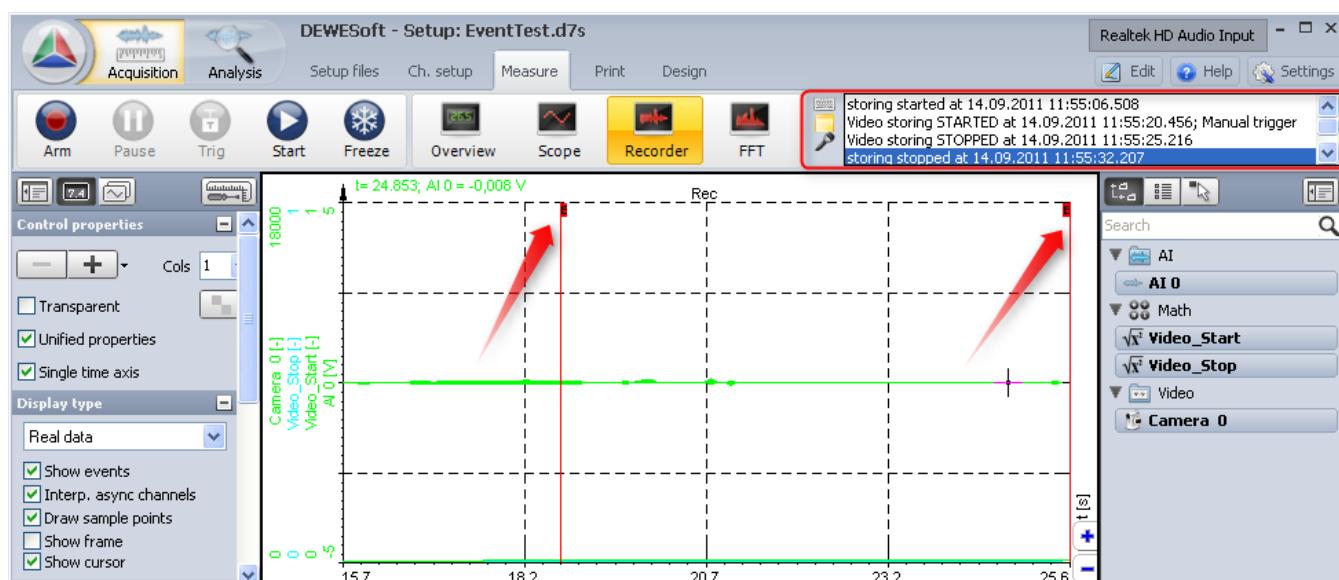
Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IDisplayTemplate](#)  read-only

2.1.52 IEvent

A single event that occurred during measurement: see [IEventList](#), [EventType](#) for more details.

In the illustration below you can see the event list (at the right top) and the events are also displayed in the recorder (the red arrows).



2.1.52.1 Data

```
property Data: OleVariant
```

Data contains special information about an event. E.g. this can be text or voice data.

e.g. the notice event will return the text

the alarm event will return a Boolean (Alarm on or off)

Interface: [IEvent](#)  read-only

2.1.52.2 PosDir

property PosDir: Integer

`PosDir` is the position of an event within a data block.

see also: [PosMid](#)

Interface: [IEvent](#)  read-only

2.1.52.3 PosMid

property PosMid: Integer

`PosMid` is the number of the data block where the event has happened.

see also: [PosDir](#)

Interface: [IEvent](#) read-only

2.1.52.4 TimeStamp

property TimeStamp: Double

`TimeStamp` is the time stamp of an event in seconds.

Interface: [IEvent](#)  read-only

2.1.52.5 TrigInfo

property TrigInfo: ITrigInfo

only relevant for events of type `etTrigger` - information about the source of the trigger.

Interface: [IEvent](#) read-only

21526 Type

property Type : Integer

The type of an event.

Note: the trailing underscore in the property name is required, because `type` is a keyword in Delphi and must not be used.

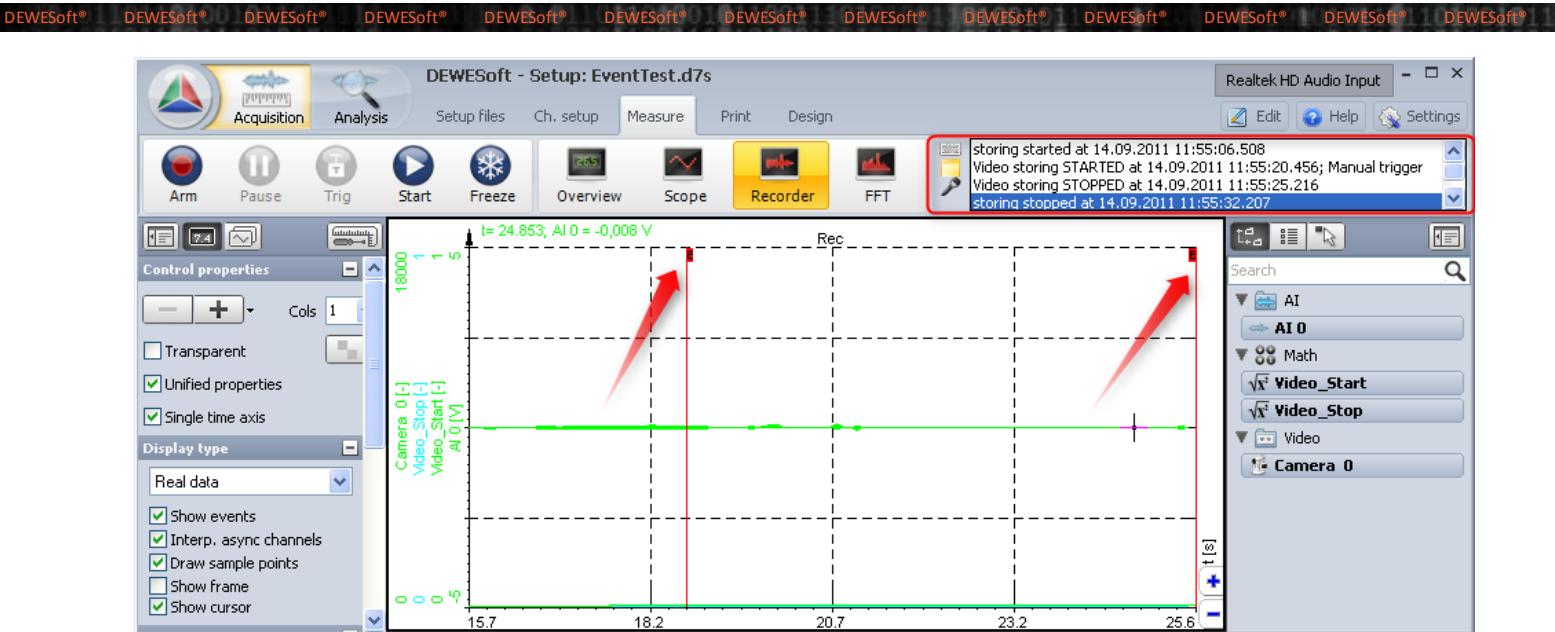
De c	He x	Name	Description
1	0x 01	etStar t	start of acquisition (see IApp_Start)
2	0x 02	etStop	stop of acquisition (see IApp_Stop)
3	0x 03	etTrigger	trigger indication (see IApp_Trigger)
11	0x 0B	etVSta rt	start video storing
12	0x 0C	etVSto p	stop video storing
20	0x 14	etKeyb oard	indicating keyboard event
21	0x 15	etNoti ce	indicating notice event
22	0x 16	etVoic e	indicating a voice event
23	0x 17	etPict ure	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
24	0x 18	etModu le	indicating a module event (e.g. bridge shunt on/off)
25	0x 19	etAlar m	indicating an alarm event

Interface: [IEvent](#)  read-only

2.1.53 IEventList

There are different types of events (see [IEvent.Type](#) for details) that can occur during measurement.

In the illustration below you can see the event list (at the right top) and the events are also displayed in the recorder (the red arrows).



see: [EventList](#)

2.1.53.1 Count

property Count: Integer

Count is the number of events in the [Item](#) list.

Interface: [IEventList](#)

read-only

2.1.53.2 Item

property Item[Index: Integer]: [IEVENT](#)

Item[I] is the event at index I. I is in the range of 0...[Count](#)-1.

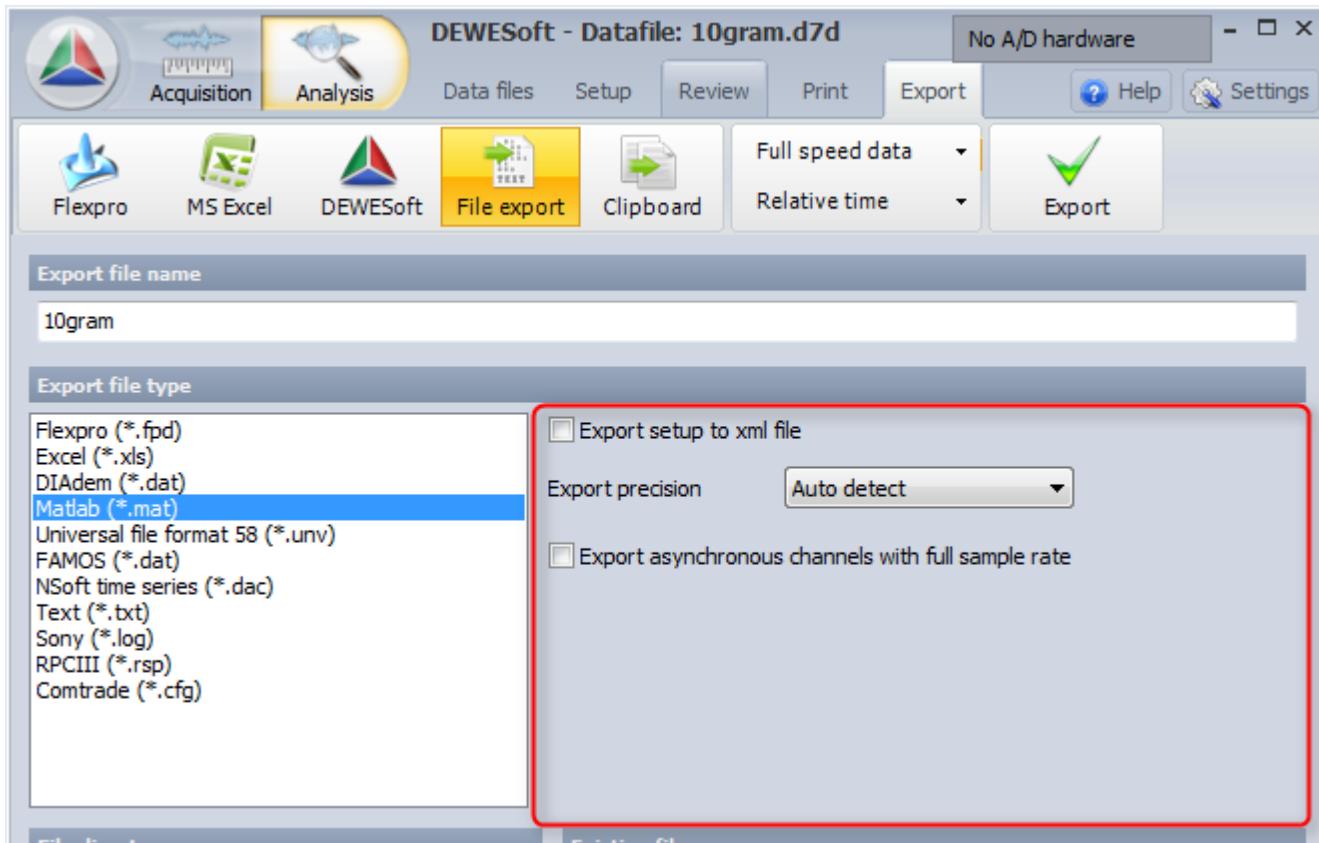
see also: [IEVENT](#)

Interface: [IEventList](#)

read-only

2.1.54 IExportFrame

each export function may use an `export frame` to display export-specific options to the user:

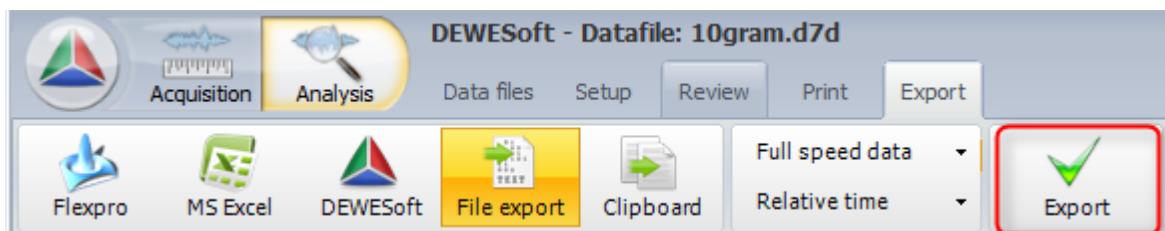


2.1.54.1 Apply

```
procedure Apply();
```

Apply will be called by DEWEsoft® when the Export button is clicked.

At this moment changes of the export configuration within the configuration frame can be applied.



Interface: [IExportFrame](#)

2.1.54.2 HideFrame

```
procedure HideFrame();
```

`HideFrame` is called when the custom export is left and its configuration frame should be hidden.

Interface: [IExportFrame](#)

2.1.54.3 SetExpApp

```
procedure SetExpApp (const App: IApp);
```

gives the export frame access to the application interface.

see also: [IApp](#)

Interface: [IExportFrame](#)

Classifier	Name	Type	Description
const	App	IApp	reference to the application interface

2.1.54.4 ShowFrame

```
procedure ShowFrame(Handle: Integer; out FrameHeight: Integer);
```

`ShowFrame` is called by DEWEsoft® when the setup frame of the custom export is shown.

Interface: [IExportFrame](#)

Classifier	Name	Type	Description
	Handle	Integer	the window handle which should be assigned to the <i>ParentWindow</i> property of the frame
out	FrameHeight	Integer	should be set to inform DEWEsoft® of the height of the configuration frame

2.1.55 IFileNameSettings

provides access to the settings of the naming options of the data files

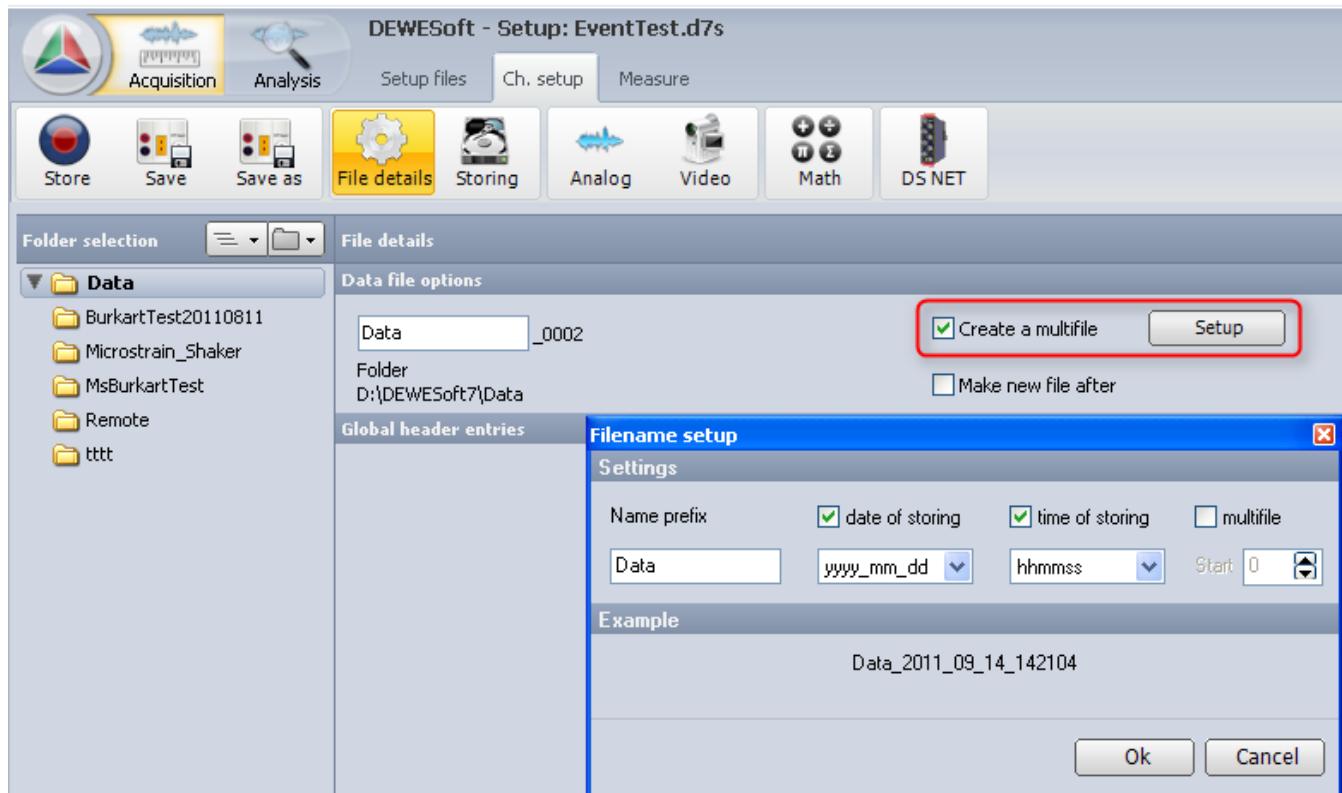
see also: [IApp.FileNameSettings](#)

2.1.55.1 AutoCreate

property AutoCreate: WordBool

`AutoCreate` determines whether the data file names should be created automatically (by using date/time or multifile index).

If this is TRUE, you can open the [Setup](#) dialogue (see illustration below) to enter more specifics about the generated filename.



see also: [BaseFileName](#), [UseDate](#), [UseTime](#), [UseMultiFile](#)

Interface: [IFileNameSettings](#)

read/write

2.1.55.2 AutoFlipAbsTime

property AutoFlipAbsTime: WordBool

`AutoFlipAbsTime` defines whether absolute time is used or not.

Only relevant if [AutoFlipFile](#) is TRUE, and if you have selected a temporal unit (h, m, s - see [AutoFlipUnit](#) for details)

If `AutoFlipAbsTime` is FALSE the [AutoFlipSize](#) is interpreted as the number of hours/minutes/seconds relative to the start of storing.

If `AutoFlipAbsTime` is TRUE the [AutoFlipSize](#) is interpreted as the number of hours/minutes/seconds of the clock; e.g.

when the `AutoFlipUnit` is set to hours (h) and the `AutoFlipSize` is set to 14, then a new file will be created every day at 14:00:



Interface: [IFileNameSettings](#)  read/write

2.1.55.3 AutoFlipFile

property AutoFlipFile: WordBool

`AutoFlipFile` defines whether a new file should be created after a certain criterion (time, file-size, number of tiggers): see [AutoFlipUnit](#) for details

To automatically flip the file can be useful for several reasons:

- to avoid big files (e.g. limit the size of files to 200MB) because you may want to transmit them over a slow connection (e.g. wireless internet connection)
 - to be able to analyse data files (if DEWEsoft® is still writing to the file, you cannot open it at the same time in analyse mode)

see also: [IApp.FileNameSettings](#), [AutoFlipSize](#), [AutoFlipUnit](#), [AutoFlipAbsTime](#)

Interface: [IFileNameSettings](#)

2.1.55.4 AutoFlipSize

property AutoFlipSize: Single

`AutoFlipSize` is the size of a certain unit (see [AutoFlipUnit](#)) after which a new file will be created (if `AutoFlipFile` is TRUE).

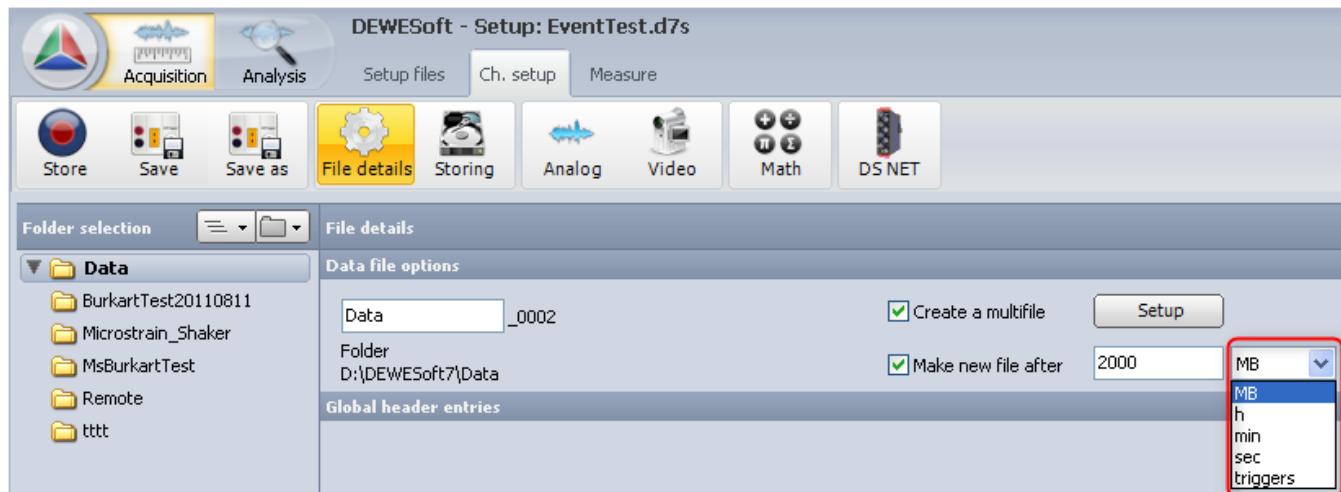
see also: [AutoFlipFile](#), [AutoFlipUnit](#), [AutoFlipAbsTime](#)

Interface: [IFileNameSettings](#)

2.1.55.5 AutoFlipUnit

property AutoFlipUnit: Integer

The unit related to the [AutoFlipSize](#) field (only relevant if [AutoFlipFile](#) is TRUE).



De c	He x	Name	Description
0	0x 00	MB	AutoFlipSize will be interpreted as a file size in MB (Megabyte)
1	0x 01	h	AutoFlipSize will be interpreted as a number of hours (relative or absolute: see AutoFlipAbsTime)
2	0x 02	min	AutoFlipSize will be interpreted as a number of minutes (relative or absolute: see AutoFlipAbsTime)
3	0x 03	sec	AutoFlipSize will be interpreted as a number of seconds (relative or absolute: see AutoFlipAbsTime)
4	0x 04	trigge rs	AutoFlipSize will be interpreted as the number of trigger occurrences

see also: [AutoFlipFile](#), [AutoFlipSize](#), [AutoFlipAbsTime](#)

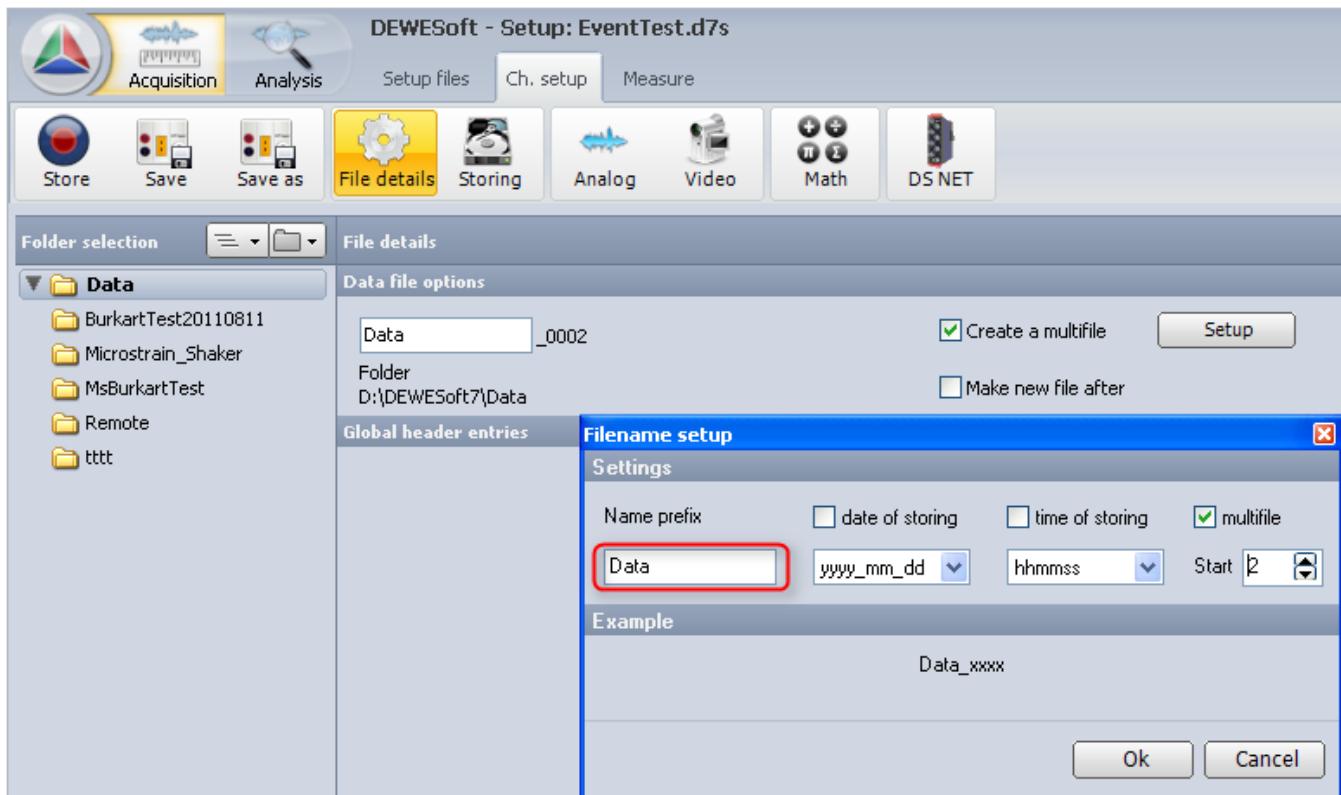
Interface: [IFileNameSettings](#)

read/write

2.1.55.6 BaseFileName

property BaseFileName: WideString

`BaseFileName` is the file name prefix for automatic naming of files (only useful if `AutoCreate` is `TRUE`):



see also: [AutoCreate](#), [UseDate](#), [UseTime](#), [UseMultiFile](#)

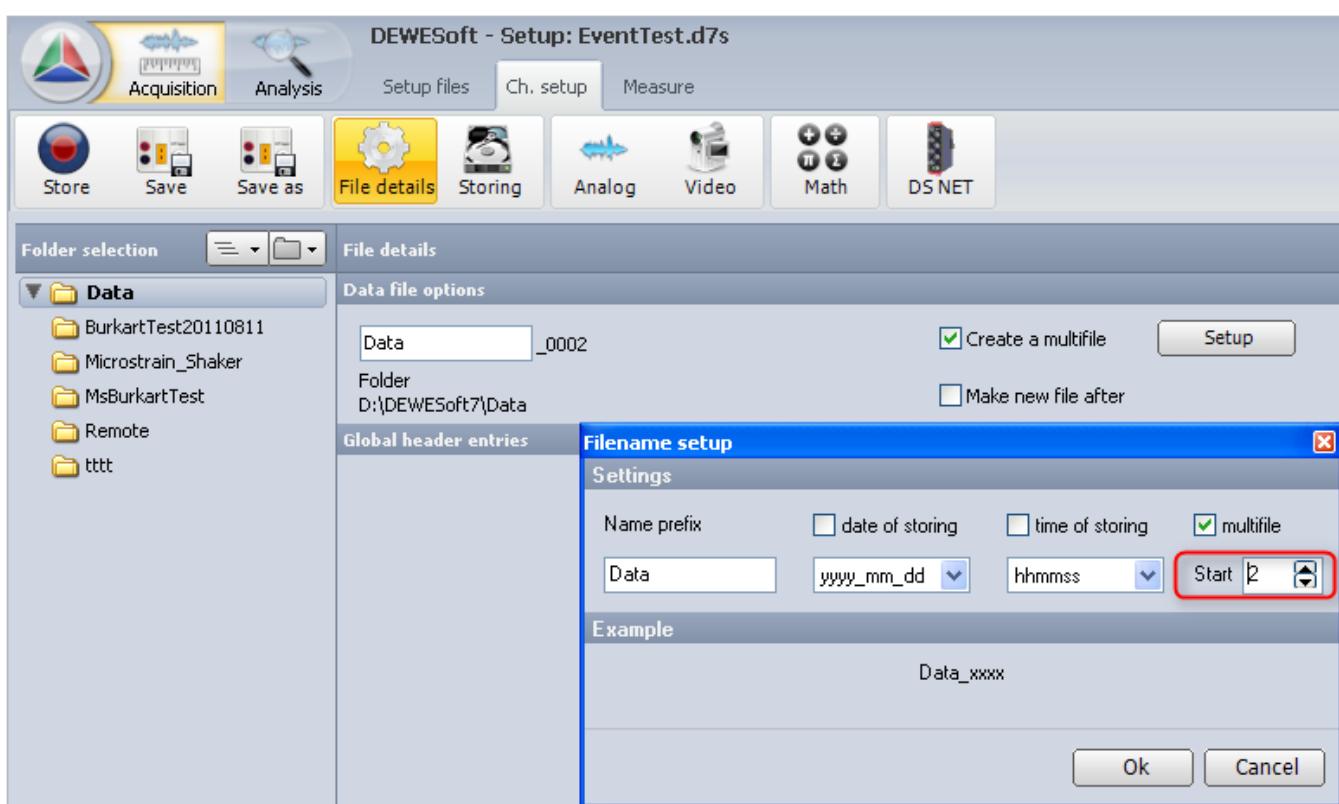
Interface: IFileNameSettings

 read/write

2.1.55.7 MultiFileStartIndex

property MultiFilestartIndex: Integer

`MultiFileStartIndex` is the number to start from for the naming of multi files (only useful if [AutoCreate](#) is TRUE and `UseMultiFile` is TRUE):



see also: [AutoCreate](#), [UseMultiFile](#)

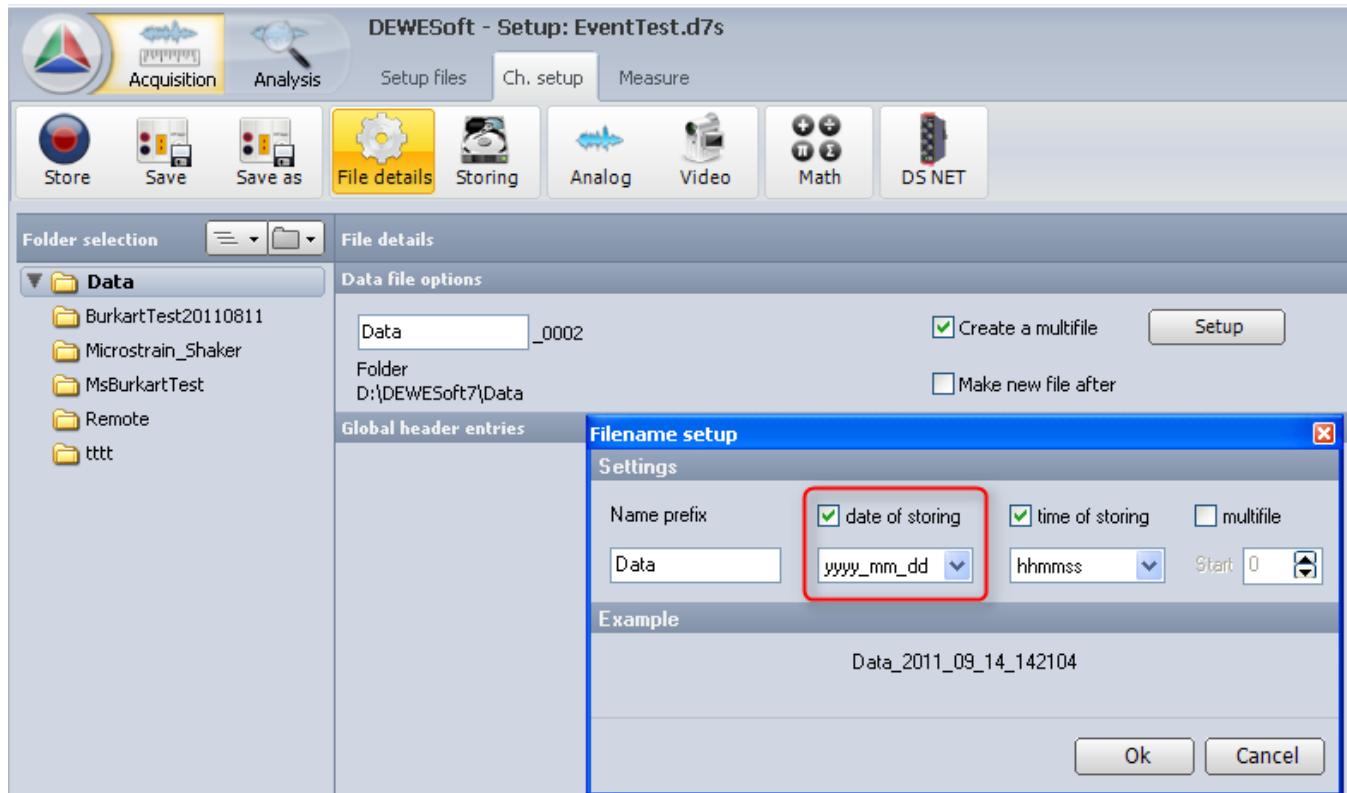
Interface: [IFileNameSettings](#)

read/write

2.1.55.8 UseDate

property UseDate: WordBool

`UseDate` determines whether the date should be included in the file name (only useful if `AutoCreate` is TRUE):



see also: [AutoCreate](#), [UseTime](#), [UseMultiFile](#)

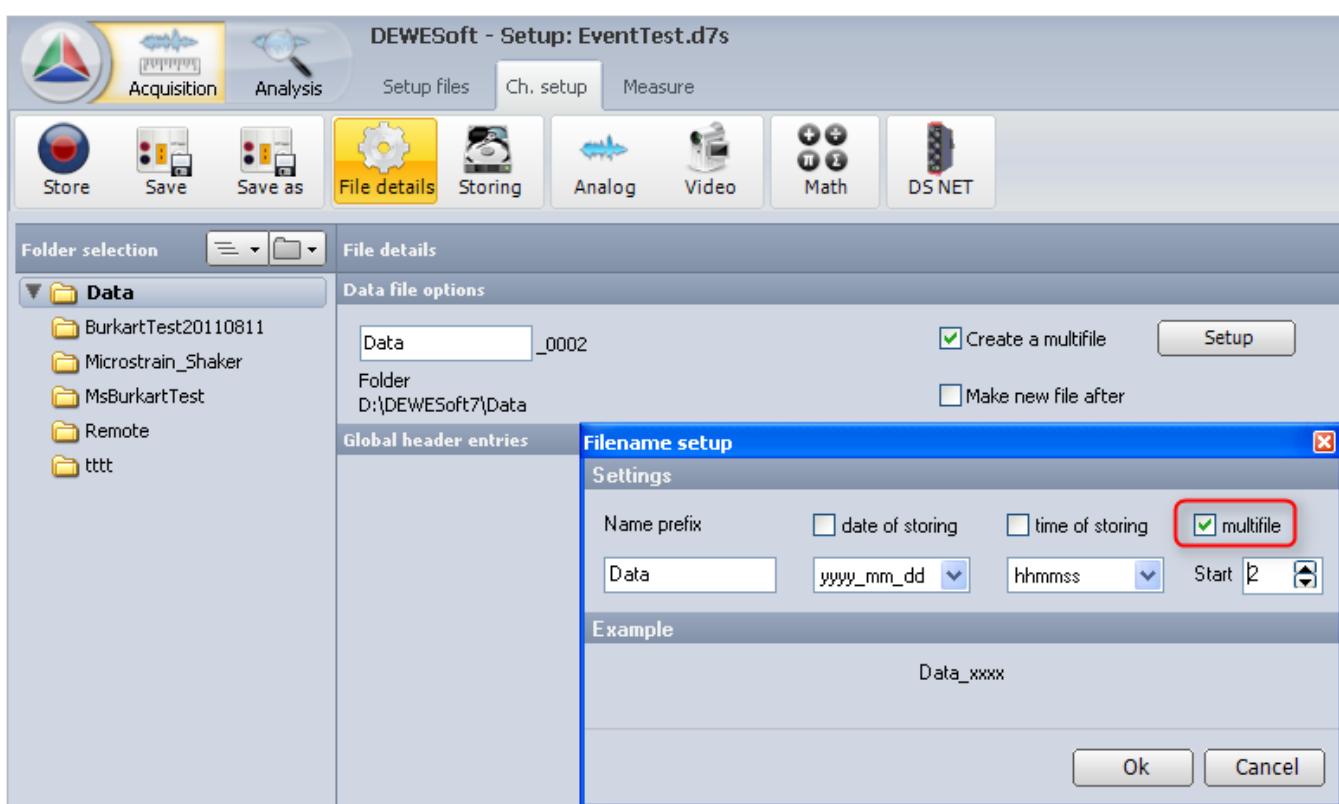
Interface: IFileNameSettings

 read/write

2.1.55.9 UseMultiFile

property UseMultiFile: WordBool

`UseMultiFile` determines whether to use the multi file index as part of the automatically created filename (only useful if [AutoCreate](#) is TRUE):



see also: [MultiFilestartIndex](#), [AutoCreate](#), [UseDate](#), [UseTime](#)

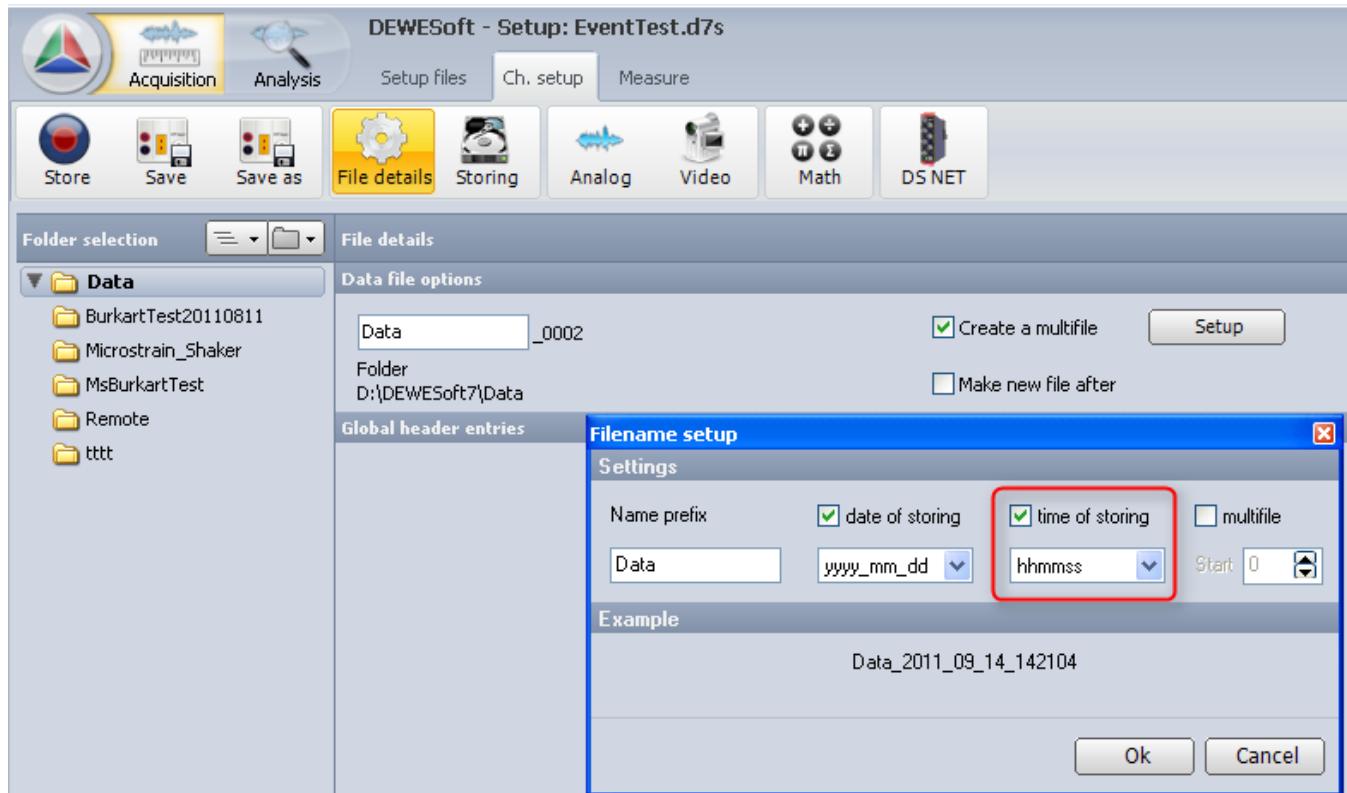
Interface: [IFileNameSettings](#)

read/write

2.1.55.10 UseTime

property UseTime: WordBool

`UseTime` determines whether the time should be included in the file name (only useful if `AutoCreate` is `TRUE`):



see also: [AutoCreate](#), [UseDate](#), [UseMultiFile](#)

Interface: [IFileNameSettings](#)

 read/write

2.1.56 IGHObject

A single data header object in the global data header (see [IGlobalHeader](#)).

In the illustration below you can see 3 `IGHObject` objects: one of type `Info` and 2 input elements (of type `Input` and `Selection`)

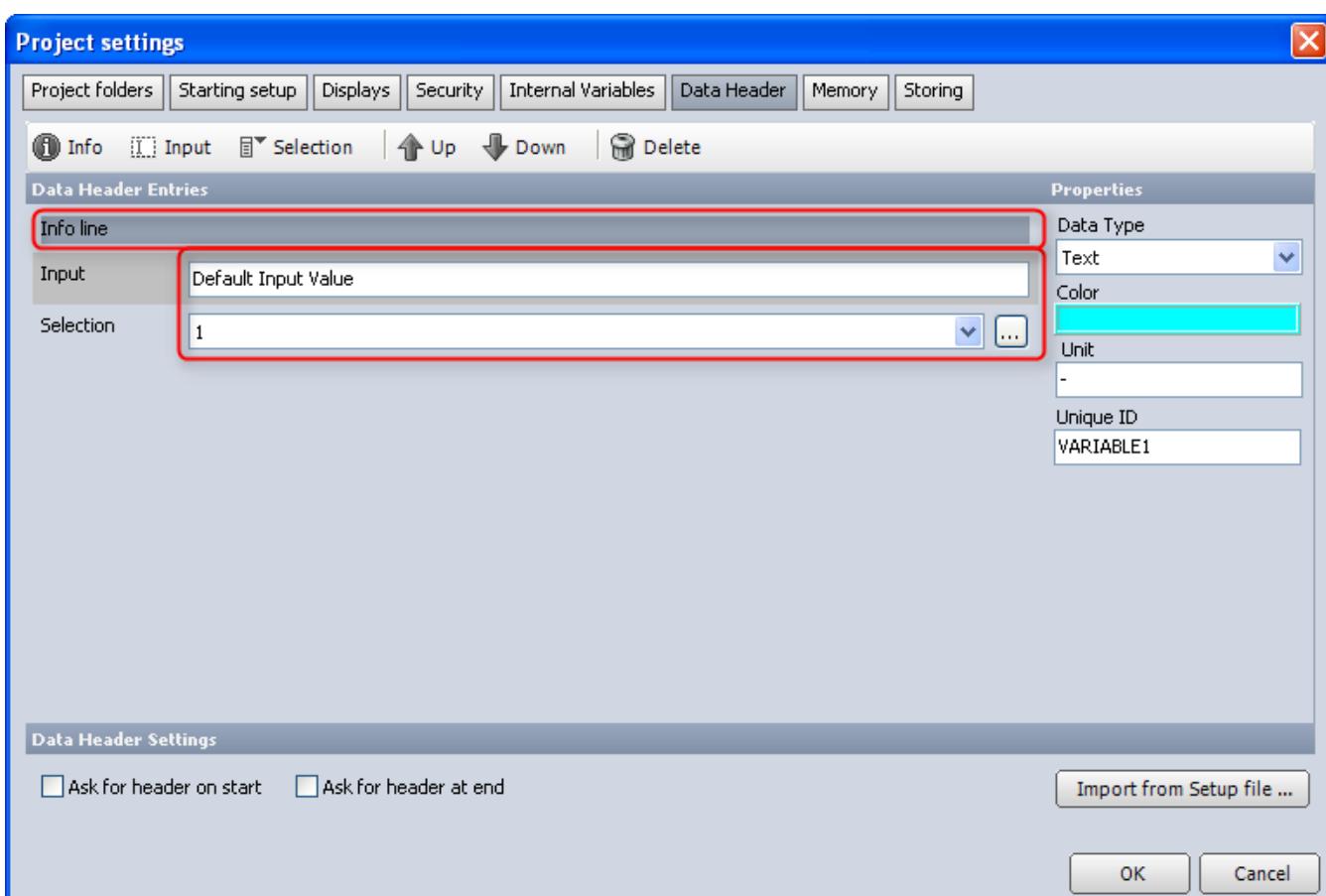


Illustration 170: 3 Global Header Definitions

The Illustration below show the input dialogue with the defined data header entries:

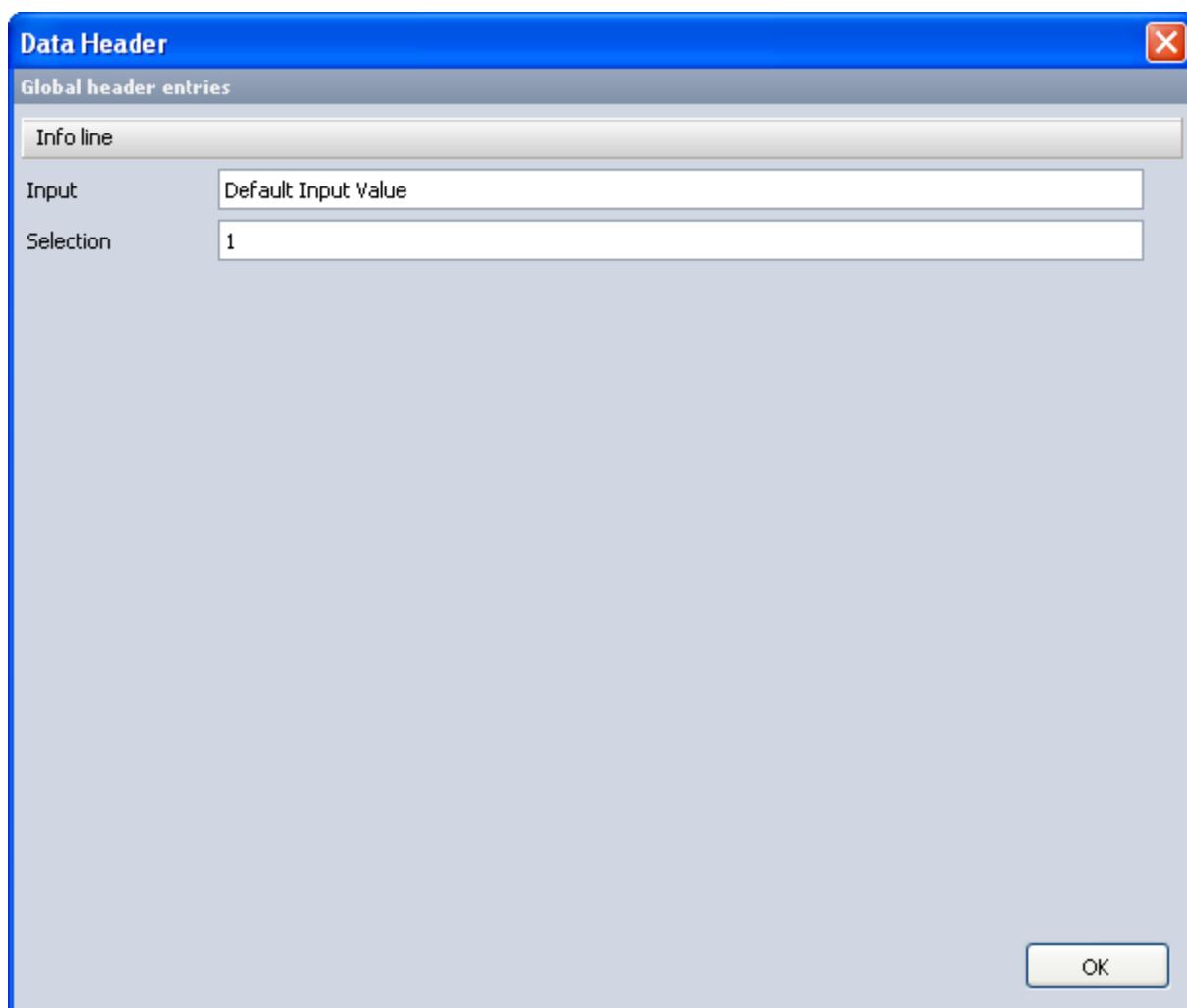


Illustration 171: Global Header Dialogue

see also: [IApp.GlobalHeader](#), [IGlobalHeader](#)

2.1.56.1 Caption

property Caption: WideString

The caption of the global data header object is an arbitrary text the will be shown in the data header input dialogue.
In the illustration below you can see the captions of 3 global data header objects.

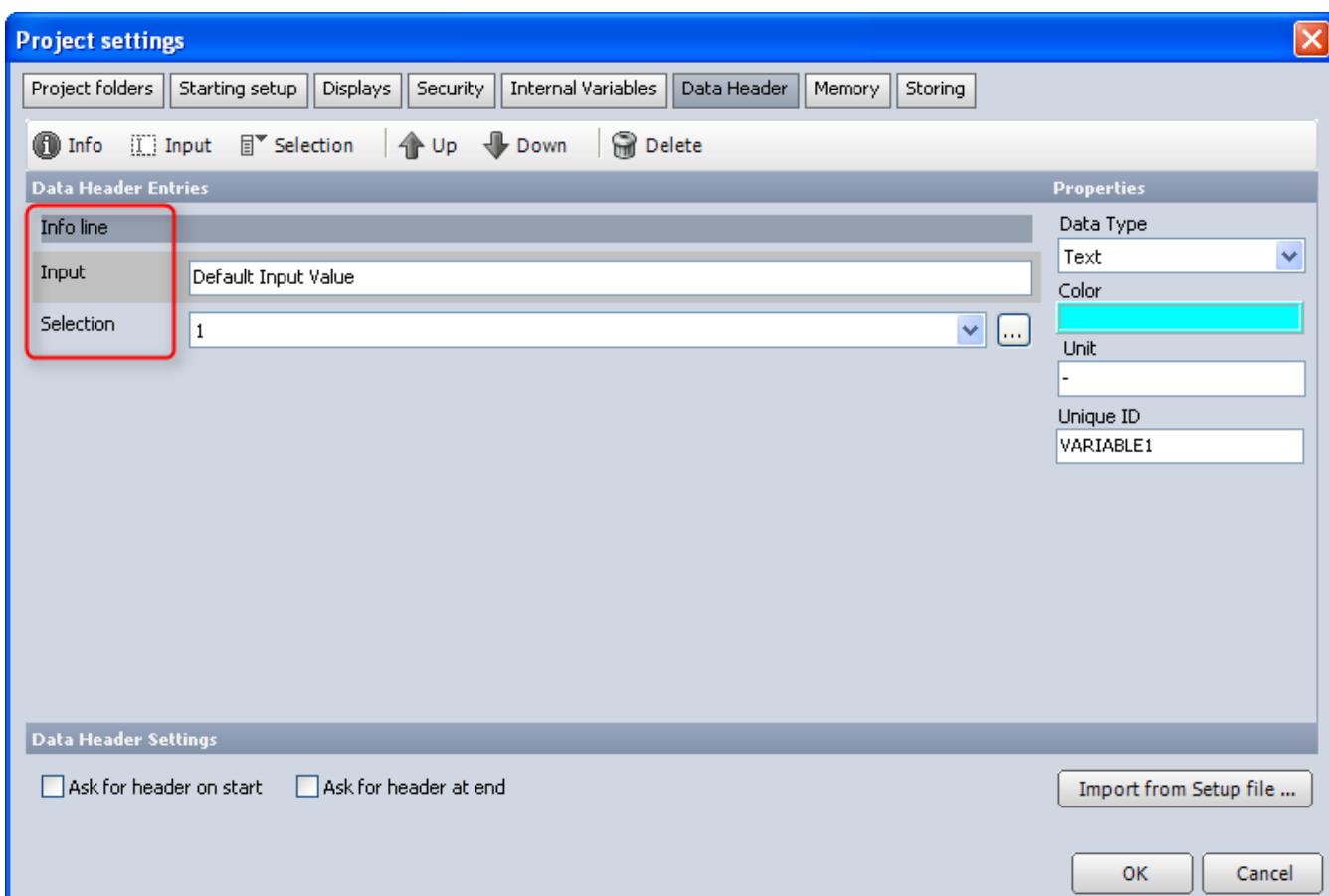


Illustration 172: Global header captions

Interface: [IGHObject](#) read/write

2.1.56.2 Data

property Data: WideString

The data of the global data header object are arbitrary text values that will be shown in the data header input dialogue.

The default values for the data can be specified in the *Data Header* section of the *Project settings*.

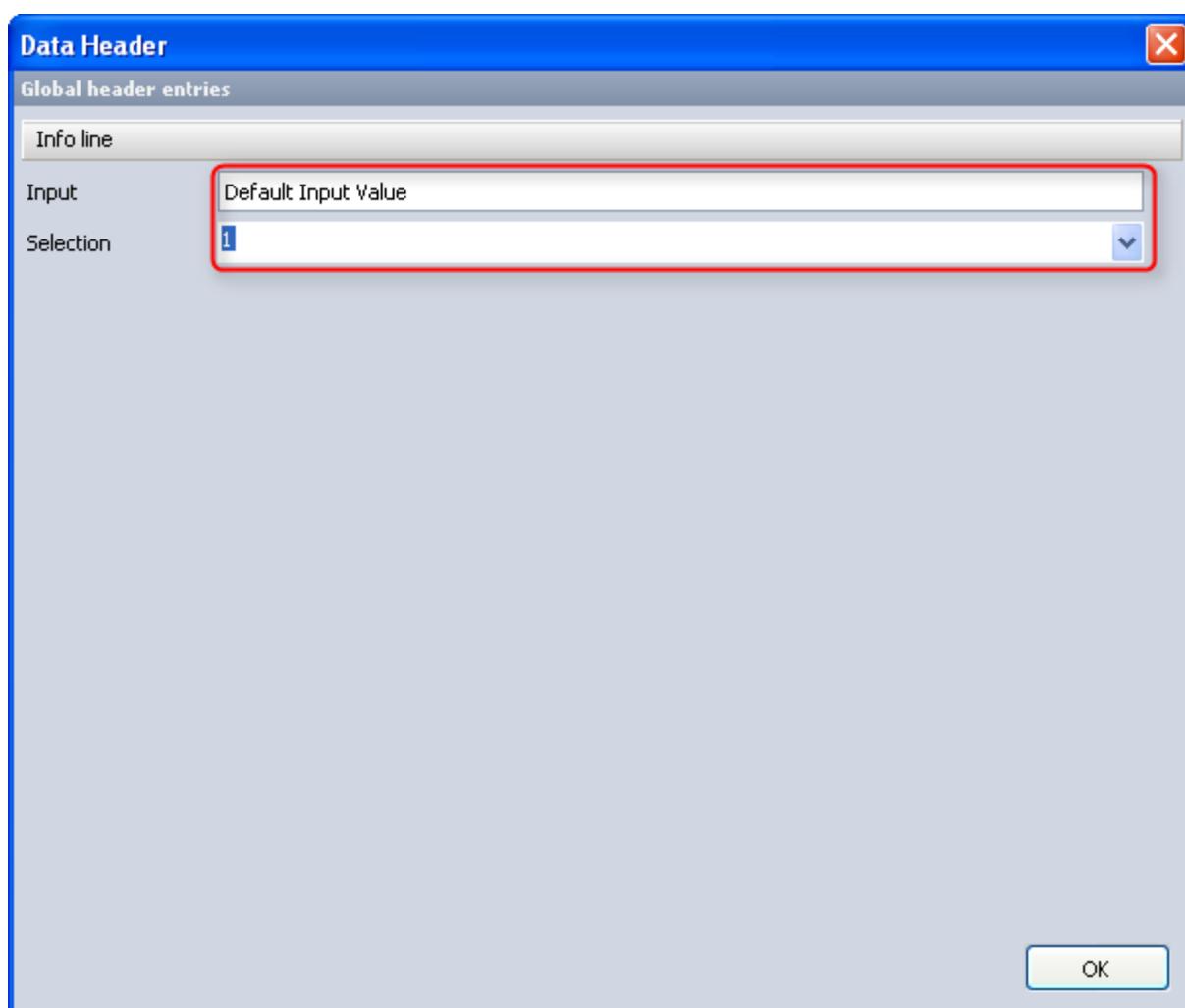
The user can change these values in the data header input dialogue (except for objects of type hcInfo: see [ObjType](#)).

In the illustration below you can see 3 [IGHObject](#) objects

1. "Info line" is of type hcInfo: the user cannot change this value and it is always the same as the [Caption](#)

2. "Input" is of type hcTextBox: the user can change the default data to an arbitrary text

3. "Selection" is of type hcComboBox: the user can select any elements from the drop-down list



Interface: [IGHObject](#) read/write

2.1.56.3 ObjType

property ObjType: Integer

see also: [GHObjectType](#)

The type of the [IGHObject](#):

Dec	Hex	Name	Description
0	0x00	hcInfo	This is just an informational text that the user can read, but not change in the data header input dialogue. see ❶ in the Illustration below.
1	0x01	hcTextBox	This is an edit field in the data header input dialogue. The user can change the default text to any arbitrary text. see ❷ in the Illustration below.
2	0x02	hcComboBox	This is a combo-box in the data header input dialogue. The user can select any of the options in the drop-down list. see ❸ in the Illustration below.

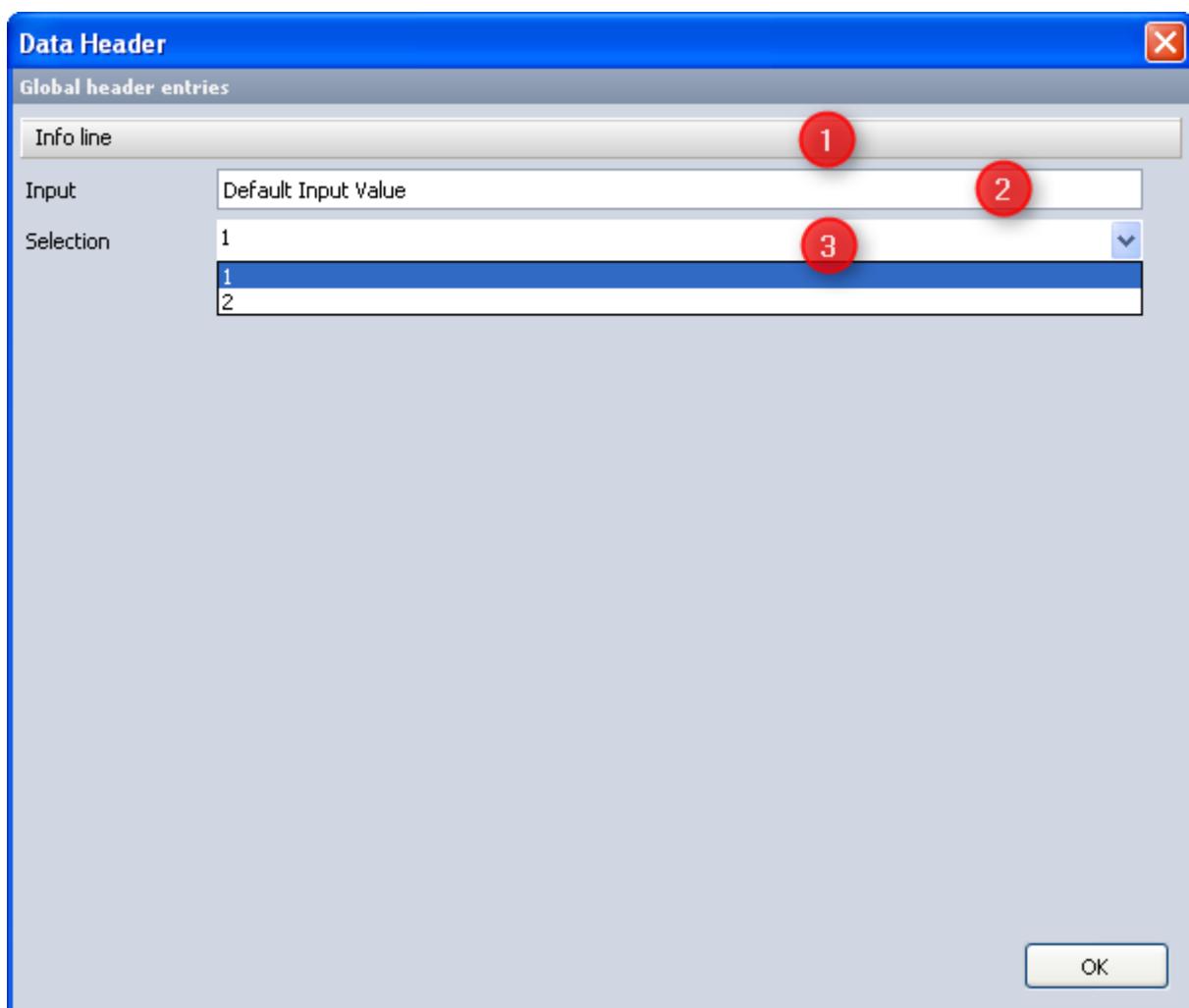


Illustration 173: Global header input dialog

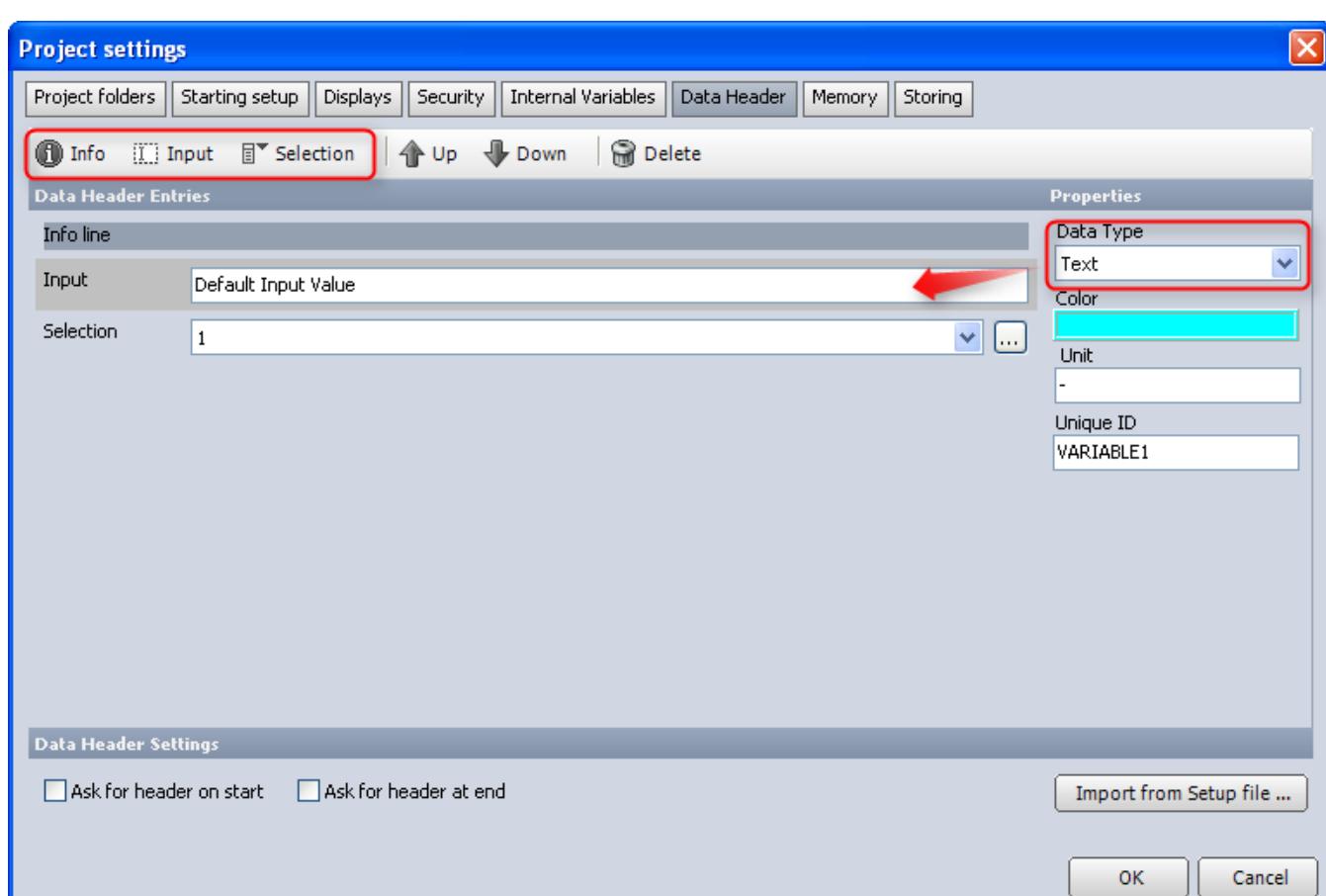


Illustration 174: Global header definition (Project Setup)

Interface: [IGHObject](#)



2.1.57 IGlobalHeader

This interface gives you access to the data header which can be defined in the Project setup. In the illustration below you see 3 Global Data Header Objects (see [IGHObject](#)):

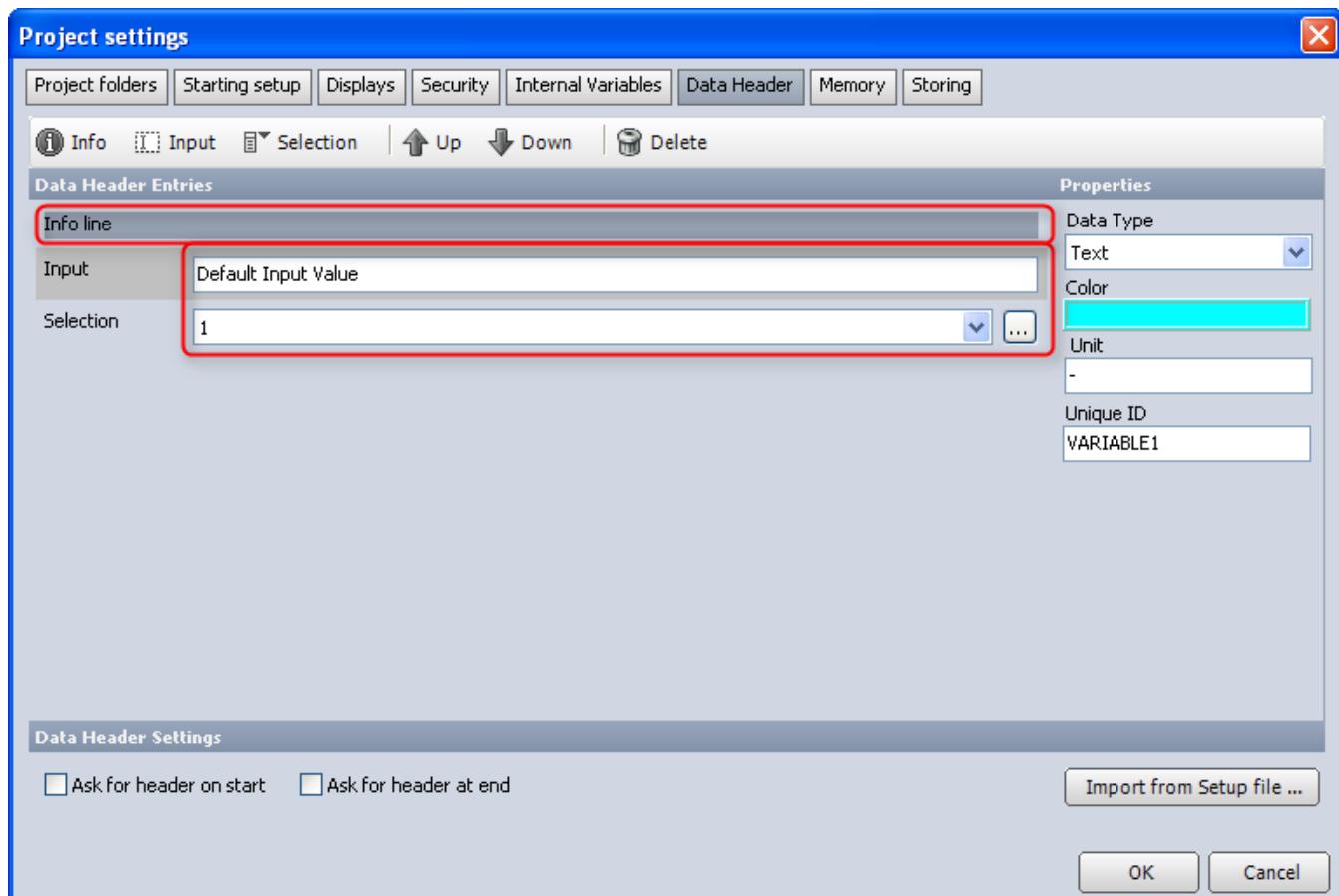


Illustration 175: Global Header (Project Setup)

The illustration below shows the input dialogue with the defined data header entries:

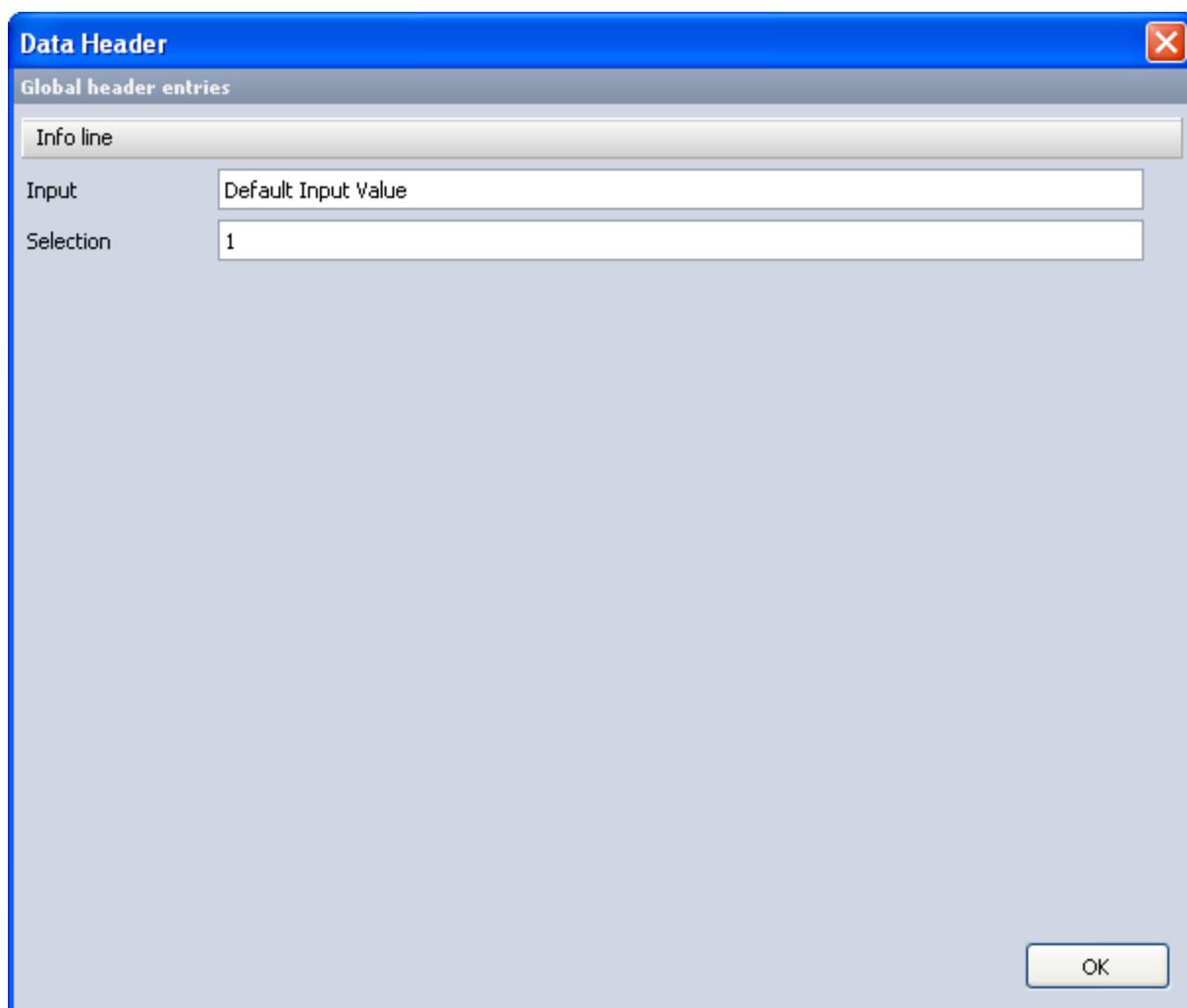


Illustration 176: Global Header Dialogue

see also: [IApp.GlobalHeader](#), [IApp.SetHeaderData](#), [IGHObject](#)

2.1.57.1 Count

property Count: Integer

The number of entries (of type [IGHObject](#)) in the [Item](#) list.

In the example below, the Count would be 3.

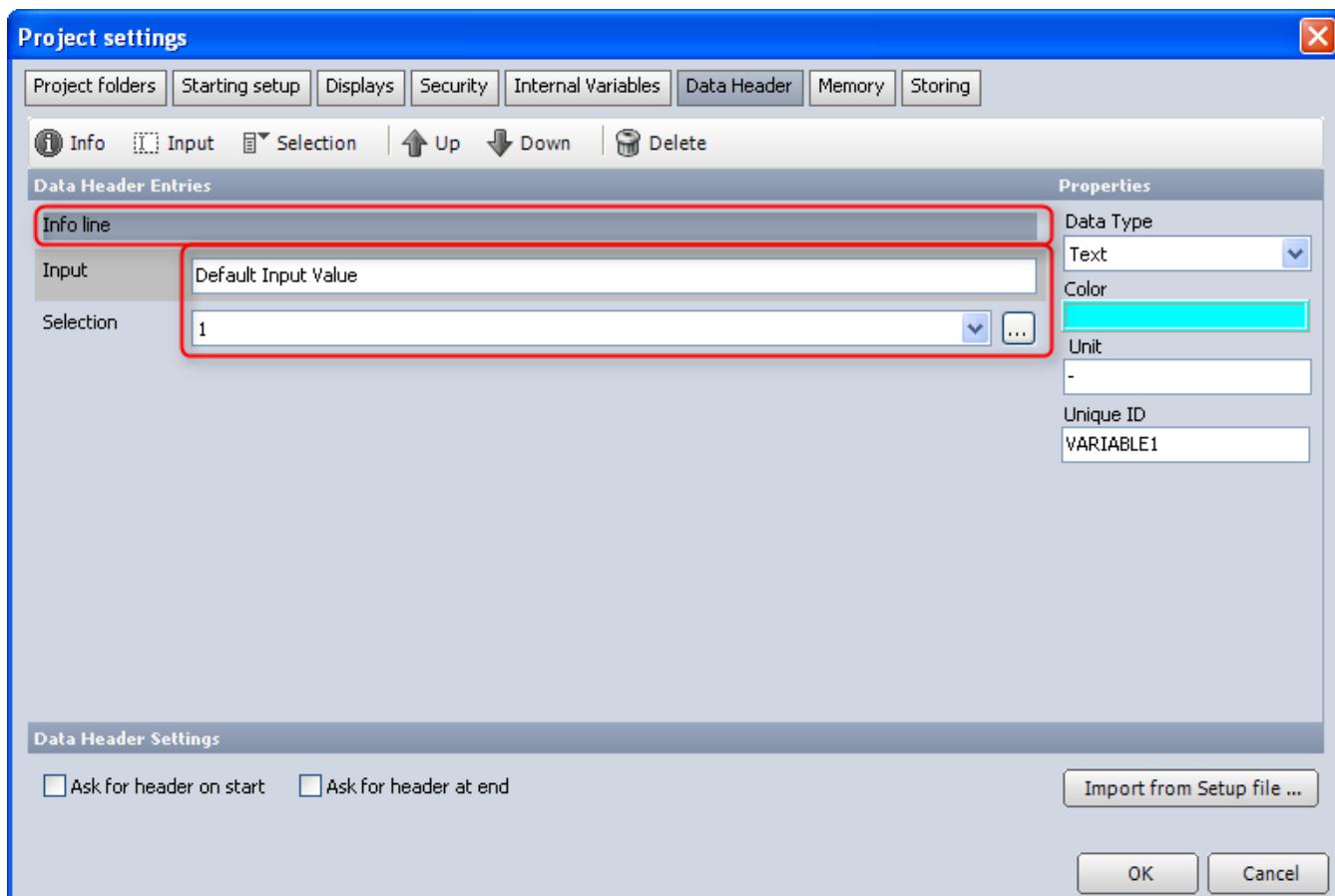


Illustration 177: Global Header (Project Setup)

Interface: [IGlobalHeader](#) read-only

2.1.57.2 Item

```
property Item[Ind: Integer]: IGHObject
```

A list of data header objects (of type [IGHObject](#)).

See also [Count](#).

Interface: [IGlobalHeader](#) read-only

2.1.58 IIImportChannel

currently this is only a marker interface so that you can see that this channel is used for a [Custom Import](#).

2.1.59 IImportGroup

special groups for channels that are used in [Custom Import](#) add-ons...

parent interface: `IChannelGroup`, `IDataGroups`

2.1.59.1 MountChannel

```
function MountChannel(): IChannel;
```

will mount a channel for the [importGroup](#)

see also: [Important](#)

Interface: ImportGroup			
Classifier	Name	Type	Description
-	<i>RESULT</i>	IChannel	the new mounted channel

2.1.60 IIndexChanger

can be used by plugins to change its index structure

see also `evChangeIndex` in [IPlugin4.OnEvent](#)

2.1.60.1 ChangePluginChIndex

```
procedure ChangePluginChIndex  
(const Guid: WideString; OldIndex: OleVariant; NewIndex: OleVariant);
```

allows the plugin to change the index of a channel

Interface: [IIndexChanger](#)

Classifier	Name	Type	Description
const	Guid	WideString	the GUID of the plugin
	OldIndex	OleVariant	the old index: array of integer the indizes are relative to the plugin (i.e. Index[2] in Channel Index)
	NewIndex	OleVariant	the new index: array of integer the indizes are relative to the plugin (i.e. Index[2] in Channel Index)

2.1.60.2 ChangePluginChIndex1

```
procedure ChangePluginChIndex1  
(const Guid: WideString; const Ch: IChannel; newIndex: OleVariant);
```

allows the plugin to change the index of a channel.

Interface: [IIndexChanger](#)

Classifier	Name	Type	Description
const	Guid	WideString	the GUID of the plugin
const	Ch	IChannel	the channel for which you want to set a new index
	NewIndex	OleVariant	the new index: array of integer the indices are relative to the plugin (i.e. Index[2] in Channel Index)

2.1.61 IInputGroup

[IInputGroup](#) is used to build a group for related input channels; e.g. a GPS input group may consist of the channels *Latitude*, *Longitude* and *Heading*.

In the illustration below, you can see that the GPS visual control only accepts an input group for GPS x and y coordinates.

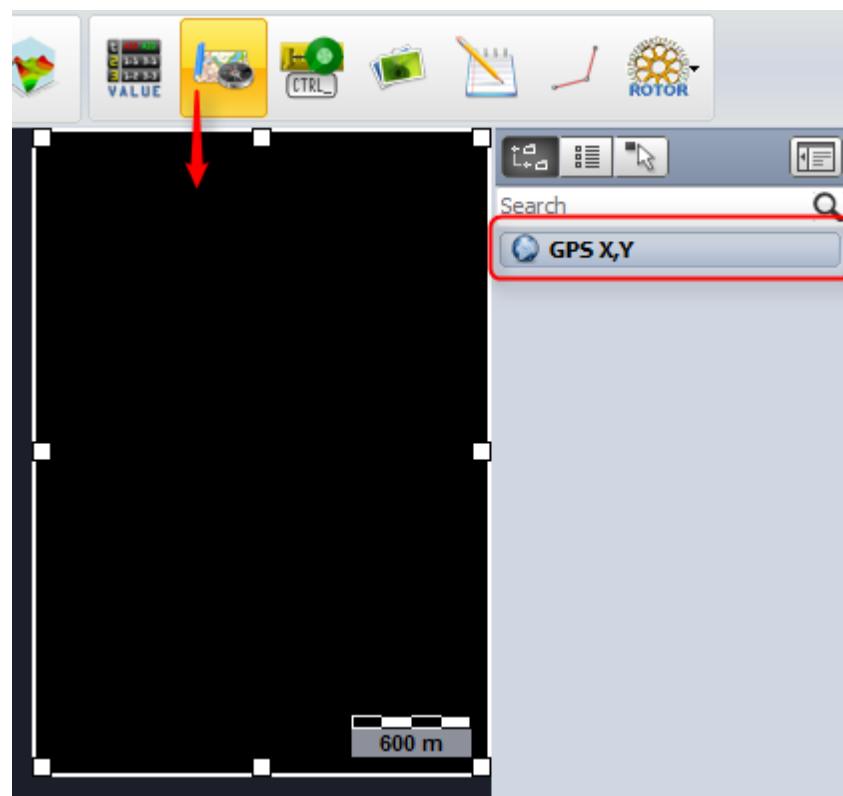


Illustration 178: GPS Input Group

see also: [IData.InputGroups](#), [IInputGroups](#)

2.1.61.1 Guid

property Guid: WideString

Unique [GUID](#) of the input group.

Interface: [IInputGroup](#) read/write

2.1.61.2 Index

property Index: OleVariant

can be used to nest the input group for display in the channel lists (like the channel index - see [Channel Index](#))

Interface: [IInputGroup](#)  read-only

2.1.61.3 Name

property Name: WideString

name of the input group

Interface: [IInputGroup](#) read/write

2.1.61.4 Properties

property Properties: [IProperties](#)

a plugin can pass arbitrary name-value pairs to visual controls

see also: [IProperties](#)

Interface: [IInputGroup](#)  read-only

2.1.62 IInputGroups

a list of `IInputGroup` objects

see also: [IDataInputGroups](#)

2.1.62.1 Count

property Count: Integer

Count is the number of items in the [Item](#) list.

Interface: [IInputGroups](#)  read-only

2.1.62.2 Item

property Item[Index: Integer]: [IInputGroup](#)

Item[I] is the input group at index I. I is in the range of 0...[Count](#)-1.

see also: [IInputGroup](#)

Interface: [IInputGroups](#)  read-only

2.1.63 ILoadEngine

a load engine is used to load DEWEsoft® data files and video files.

see: [IApp.LoadEngine](#), [IVideoLoadEngines](#)

2.1.63.1 CloseFile

procedure CloseFile();

CloseFile closes a measurement file.

Interface: [ILoadEngine](#)

2.1.63.2 DataSections

property DataSections: [IDataSections](#)

DataSections provides information about data sections which are the parts between a start and a stop event.

see also: [IDataSections](#)

Interface: [ILoadEngine](#)  read-only  use only in automation applications

2.1.63.3 FileOpened

property FileOpened: WordBool

`FileOpened` specifies whether a measurement file is currently opened or not.

Interface: [ILoadEngine](#)  

2.1.63.4 GetVideoCompressDone

function GetVideoCompressDone(): Integer;

you can have auto-compression of video files after the measurement

Interface: [ILoadEngine](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	Integer	percentage of video compression progress

2.1.63.5 IsVideoCompressDone

function IsVideoCompressDone(): WordBool;

Will be `TRUE` if the compression is completely done: see also [GetVideoCompressDone](#)

Interface: [ILoadEngine](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WordBool	<code>TRUE</code> if the compression is completely done

2.1.63.6 NextDBLoad

function NextDBLoad(): WordBool;

`NextDBLoad` is used in conjunction with [StartDBLoad](#). It proceeds to the next data block of the direct buffer. The function returns `True`, if a data block is available, `False` otherwise.

Interface: [ILoadEngine](#) 

Classifier	Name	Type	Description
-	<i>RESULT</i>	WordBool	<code>True</code> , if a data block is available, <code>False</code> otherwise.

2.1.63.7 NumBlocks

property NumBlocks: Integer

NumBlocks is the number of data blocks within a measurement file.

Interface: [ILoadEngine](#)  read-only  use only in automation applications

2.1.63.8 ReducedOnly

property ReducedOnly: WordBool

ReducedOnly specifies whether data has been stored using reduced rate only.

see also: [Sample Rates](#)

Interface: [ILoadEngine](#)  read-only  use only in automation applications

2.1.63.9 Reload

procedure Reload(Start: [T_RecordPosition](#); Stop: [T_RecordPosition](#)) ;

Reload reloads a specific section of data.

see also: [ReloadEx](#), [ReloadBlock](#), [T_RecordPosition](#)

Interface: [ILoadEngine](#)  use only in automation applications

Classifier	Name	Type	Description
	Start	T_RecordPosition	the beginning of the data section to reload
	Stop	T_RecordPosition	the end of the data section to reload

2.1.63.10 ReloadBlock

procedure ReloadBlock(Num: Integer) ;

ReloadBlock reloads the specified data block.

see also: [Reload](#)

Interface: [ILoadEngine](#)  use only in automation applications

Classifier	Name	Type	Description
	Num	Integer	the data block to reload.

2.1.63.11 ReloadEx

```
procedure ReloadEx(StartBlock: Integer; EndBlock: Integer; const Channel: IChannel  
; MinLevel: Integer);
```

`ReloadEx` reloads a specific section of data. Extended version of [Reload](#).

see also: [Reload](#), [ReloadBlock](#), [IChannel](#)

Interface: [ILoadEngine](#)

Classifier	Name	Type	Description
	StartBlock	Integer	the start block to reload (starts from 0)
	EndBlock	Integer	the end block to reload (see also NumBlocks)
const	Channel	IChannel	nil will load all channels, a certain channel instance (the buffers of this channel will be filled with data)
	MinLevel	Integer	0 will fill the direct buffer, 1 the first intermediate buffer, ..

2.1.63.12 ShrinkFile

```
procedure ShrinkFile(const FileName: WideString);
```

`ShrinkFile` can be used to export a part of the loaded file to DEWEsoft® file format. Prior to using this function the properties `IData.StartStamp` and `IData.EndStamp` must be set.

Interface: `ILoadEngine` use only in automation applications

Classifier	Name	Type	Description
const	FileName	WideString	the name of the file (including the path) new file

2.1.63.13 StartDBLoad

```
procedure StartDBLoad(Start: T_RecordPosition; Stop: T_RecordPosition  
; BlockSize: Integer);
```

Starts the loading of data blocks beginning at a certain position (`Start`) within the direct buffer, to a certain position within the direct buffer (`Stop`), of a certain `BlockSize`.

The function `NextDBLoad` has to be used to proceed to the next data block.

see also: [NextDBLoad](#), [T_RecordPosition](#), [The Buffer Structure](#)

Interface: `ILoadEngine`  use only in automation applications

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	Start	T_RecordPosition	the start position within the direct buffer								
	Stop	T_RecordPosition	the stop position within the direct buffer								
	BlockSize	Integer	the block size to read								

2.1.63.14 StartVideoCompress

```
procedure StartVideoCompress();
```

starts the video compression

see also: [StopVideoCompress](#), [IsVideoCompressDone](#), [GetVideoCompressDone](#)

Interface: [ILoadEngine](#)

2.1.63.15 StopVideoCompress

```
function StopVideoCompress(): WordBool;
```

stops the video compression. should be called after [IsVideoCompressDone](#) returned TRUE.

see also: [StartVideoCompress](#), [IsVideoCompressDone](#), [GetVideoCompressDone](#)

Interface: [ILoadEngine](#)

Classifier	Name	Type	Description
-	RESULT	WordBool	TRUE if the compression was successful

2.1.63.16 VideoLoadEngines

```
property VideoLoadEngines: IVideoLoadEngines
```

provides access to the video load engines: see [IVideoLoadEngines](#) for details

Interface: [ILoadEngine](#)  read-only

2.1.64 IMasterClock

The master clock is the time reference used for DEWEsoft® data. If an analogue card is active, then the analogue card will provide the master clock. In the absence of an analogue card, the PC clock will be used as master clock (but also a plug-in could provide the master clock in this case).

The can be used to get the current time in seconds (see [IMasterClock.GetCurrentTime](#)) during measurement. This information can be used by plug-ins to timestamp asynchronous data.

see also: [IApp.MasterClock](#), [IPlugin3.ProvidesClock](#), [IPlugin3.OnGetClock](#)

2.1.64.1 GetcurrentTime

```
function GetcurrentTime(): Double;
```

The current time stamp in seconds since the start of the measurement.

This value should only be read in the `IPlugin2.OnGetData` or `IPlugin2.OnStopAcq` functions.

ATTENTION!

Do not call this function in `IPlugin2.OnStartStoring` or `IPlugin2.OnStopStoring`! see [Start Events](#) for details.

see also: [Start Events](#), [IApp.MasterClock](#), [IPlugin3.ProvidesClock](#), [IPlugin3.OnGetClock](#), [IData.StartStoreTimeUTC](#)

Interface: [IMasterClock](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	Double	The current time stamp in seconds during measurement

2.1.65 IMath

a list of [IMathObjects](#)

see also: [IApp.Math](#)

2.1.65.1 AddObj

```
function AddObj(const Guid: WideString): IMathObject;
```

adds the math object

Classifier	Name	Type	Description
const	Guid	WideString	the GUID of the new math object
-	<i>RESULT</i>	IMathObject	the new math object that has been added to the list

2.1.65.2 Count

property Count: Integer

Count is the number of items in the [MathObject](#) list

Interface: [IMath](#)  read-only

2.1.65.3 FindObjByID

function FindObjByID(Index: Integer): [IMathObject](#);

returns the math object with the given Index. The index must be in the range of 0...[Count](#)-1.

see also: [IMathObject](#)

Interface: [IMath](#)

Classifier	Name	Type	Description
	Index	Integer	the index of the math object to find. The index must be in the range of 0... Count -1.
-	RESULT	IMathObject	returns the math object with the given Index. The index must be in the range of 0... Count -1.

2.1.65.4 MathObject

property MathObject[Index: Integer]: [IMathObject](#)

Item[I] is the math object at index I. I is in the range of 0...[Count](#)-1.

see also: [IMathObject](#)

Interface: [IMath](#)  read-only

2.1.65.5 RemoveObj

procedure RemoveObj(I: Integer);

Removes the math object at index I. I is in the range of 0...[Count](#)-1.

Interface: [IMath](#)

Classifier	Name	Type	Description
	I	Integer	Removes the math object at index I. I is in the range of 0... Count -1.

2.1.66 IMathChannel

special interface for mathematical channels

see Mathematics

2.1.66.1 DefaultMax

property DefaultMax: Double

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IMathChannel](#)

2.1.66.2 DefaultMin

property DefaultMin: Double

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IMathChannel](#)

2.1.66.3 DefaultRes

property DefaultRes: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IMathChannel](#)

2.1.67 |MathContext

context object that will be passed to custom math plugins

do not confuse with [IMathObjContext](#)

see also: [Mathematics](#) | [DewePlugin](#) | [CustomMathMessages](#) | [CustomMathFrameMessages](#)

2.1.67.1 InputChannels

property InputChannels: IChannelList

a list of all input channels.

see also: `mthAcceptInputChannel` in [CustomMathMessages](#), [IChannelList](#)

Interface: [IMathContext](#)  read-only

2.1.67.2 MountChannel

function MountChannel(Id: Integer; const Descriptor: WideString): IChannel;

can be called by the plugin to mount channels. It should be used when the `mthMountChannels` (see: [CustomMathMessages](#)) event is received .

see also: [MountChannelEx](#), [IChannel](#)

Interface: [IMathContext](#)

Classifier	Name	Type	Description
	Id	Integer	a unique ID of the channel
const	Descriptor	WideString	a text description of the channel
-	RESULT	IChannel	the channel that has been mounted. The plugin could keep a reference to this channel or use OutputChannels

2.1.67.3 MountChannelEx

**function MountChannelEx
(out Created: WordBool; Id: Integer; const Descriptor: WideString; const AUnit: WideString; const Name: WideString; Color: Integer): IChannel;**

can be called by the plugin to mount channels. It should be used when the `mthMountChannels` (see: [CustomMathMessages](#)) event is received .

see also: [MountChannel](#), [IChannel](#)

Interface: [IMathContext](#)

Classifier	Name	Type	Description
out	Created	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
	Id	Integer	a unique ID of the channel
const	Descriptor	WideString	a text description of the channel: see also IChannel.Description
const	AUnit	WideString	the unit of the channel: see also IChannel.Unit
const	Name	WideString	the name of the channel: see IChannel.Name
	Color	Integer	the color of the channel: see IChannel.MainDisplayColor
-	RESULT	IChannel	the channel that has been mounted. The plugin could keep a reference to this channel or use OutputChannels

2.1.67.4 MountInputGroup

```
function MountInputGroup(Id: Integer): IInputGroup;
```

can be used to mount an input group for the math plugin

see also: [IInputGroup](#)

Interface: [IMathContext](#)

Classifier	Name	Type	Description
	Id	Integer	unique ID of the input group
-	RESULT	IInputGroup	the mounted input group

2.1.67.5 OutputChannels

```
property OutputChannels: IChannelList
```

the output channels of the plugin (that have been mounted by the following functions: [MountChannel](#), [MountChannelEx](#))

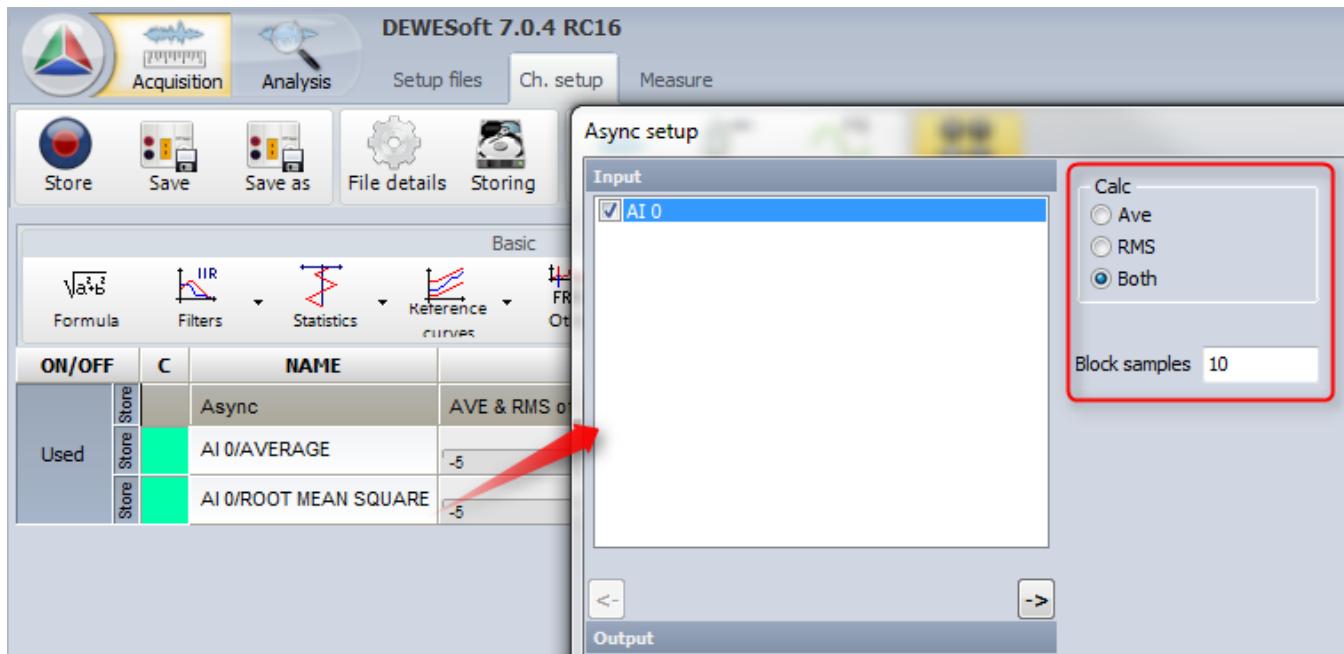
see also: [IChannelList](#)

Interface: [IMathContext](#)  read-only

2.1.68 IMathFrameContext

will be passed to the math plugin to pass configuration data from the plugin to the GUI and back.

e.g. usually these parameters are shown in a frame inside the channel setup (see red rectangle in the illustration below):



and the math plugin will then store these settings in the context of the math object.

see also [Mathematics](#), [MathObjContext](#)

2.1.68.1 Apply

```
procedure Apply(Props: OleVariant);
```

should be called by the plugin to set the properties of the math plugin. e.g. when the user changes a value in the channel setup of the math plugin frame, this function will apply the changes to the context.

see also: [mfrSetFrame](#) in [CustomMathFrameMessages](#)

Interface: [IMathFrameContext](#)

Classifier	Name	Type	Description
	Props	OleVariant	the plugin can store arbitrary data in this OleVariant: e.g. also an array of other OleVariants, etc..

2.1.69 IMathItem

`IMathItem` provides access to the input and output channels for math objects.

see Mathematics

child interfaces: [IMathModule](#), [IMathObjContext](#)

2.1.69.1 InputChannels

property InputChannels: IChannelList

a list of input channels for the mathematic function.

Interface: [IMathItem](#)

2.1.69.2 OutputChannels

property OutputChannels: IChannelList

a list of output channels for the mathematic function.

see also: [Mathematics](#), [IChannelList](#), [IMathModule](#), [IMathObject](#)

Interface: [IMathItem](#)

2.1.70 IMathModule

Each math object (see: [IMathObject](#)) can have one or more `IMathModules` which provide access to their input and output channels.

see Mathematics

parent interface: [IMathItem](#)

2.1.70.1 Id

property Id: Integer

unique Id of the [IMathModule](#)

see also: [IMathObject.FindModuleByID](#)

Interface: [IMathModule](#)  read-only

2.1.70.2 MathObject

property MathObject: [IMathObject](#)

reference to the [IMathObject](#) of the math module.

see also: [IMathObject](#)

Interface: [IMathModule](#)  read-only

2.1.71 IMathObjContext

the math object context allows the math object to store its properties.

parent interface: [IMathItem](#)

see [Mathematics](#)

see also: [IMathObject.MathObjContext](#)

2.1.72 IMathObject

Math objects can have one or more math modules (see [IMathModule](#)).

see [Mathematics](#)

2.1.72.1 Count

property Count: Integer

Count is the number of items (of type [IMathModule](#)) in the [MathModule](#) list

Interface: [IMathObject](#)  read-only

2.1.72.2 FindModuleByID

```
function FindModuleByID(I: Integer): IMathModule;
```

returns the math module at the given `Index` or `nil`. The index must be in the range of `0...Count-1`.

see also: [IMathModule](#)

Interface: [IMathObject](#)

Classifier	Name	Type	Description
	I	Integer	the index of the math module to find. The index must be in the range of 0... Count -1.
-	RESULT	IMathModule	the math module at the given Index or nil

2.1.72.3 Id

property Id: Integer

unique ID of the math object

see also: [IMath.FindObjByID](#)

3.1.72.4 MathGLID

property MathGUID: wideString;

the GUID of the math object

Interface: [IMathObject](#)  read-only

2.1.72.5 MathModule

```
property MathModule[Index: Integer]: IMathModule
```

Item[*I*] is the math module at index *I*. *I* is in the range of $0 \dots \text{Count} - 1$.

see also: [IMathModule](#)

Interface: [IMathObject](#)  read-only

2.1.72.6 MathObjContext

property MathObjContext: [IMathObjContext](#)

the math object context allows the math object to store it's properties.

see also: [IMathObjContext](#)

Interface: [IMathObject](#)  read-only

2.1.72.7 MathType

property MathType: WideString

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IMathObject](#)  read-only

2.1.72.8 Name

property Name: WideString

the name of the math object

Interface: [IMathObject](#)  read/write

2.1.73 IModule

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

2.1.73.1 ClearModule

procedure ClearModule();

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

ClearModule sets the address of a PAD-module back to 0.

Interface: [IModule](#)

2.1.73.2 DaqData

property DaqData: [IDaqData](#)

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

DaqData provides the data of a DAQ-module.

see also: [IDaqData](#)

Interface: [IModule](#)  read-only

2.1.73.3 DetectModule

procedure DetectModule(Address: Integer);

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

DetectModule detects a module at a certain address.

Interface: [IModule](#)

Classifier	Name	Type	Description
	Address	Integer	is the address at which the module has to be detected.

2.1.73.4 FREQAFindTriggerLevel

procedure FREQAFindTriggerLevel();

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

FREQAFindTriggerLevel activates the auto-trigger of the FREQA-module.

Interface: [IModule](#)

2.1.73.5 FillModule

function FillModule(Address: Integer; Timeout: Integer): WordBool;

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Tries to set a PAD-module having address 0 to the given address and waits for pressing the button on the module.

Interface: [IModule](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	Address	Integer	is the new address to set for the PAD-module								
	Timeout	Integer	the time to wait for the operator pressing the button on the module or the time it may take to detect the module at the given address. Its unit is ms.								
-	<i>RESULT</i>	WordBool	TRUE is success								

2.1.73.6 GetDataPad

```
procedure GetDataPad();
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

gets the Pad data

Interface: [IModule](#)

2.1.73.7 GetSerialNumber

```
function GetSerialNumber(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`GetSerialNumber` reads out the serial number of the module.

Interface: [IModule](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	reads out the serial number of the module

2.1.73.8 Index

```
property Index: Smallint
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`Index` is the index of the module which is in the range 0... [IModules.Count](#)-1

Interface: [IModule](#)



2.1.73.9 ModuleType

property ModuleType: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`:ModuleType` denotes the type of a module:

0...none

1...PAD

2...DAQ

Interface: [IModule](#)



2.1.73.10 PadData

property PadData: IPadData

DEPRECATED! This feature is deprecated and should not be used in DEWESESoft® version 7.0 and higher.

provides access to the Pad data.

see also: [IPadData](#)

Interface: [IModule](#)



2.1.73.11 SetDaq

```
function SetDag(): WordBool;
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`SetDag` applies the configuration to a PAD-module after previously having changed its properties.

Interface: [IModule](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	WordBool	TRUE is success

2.1.73.12 SetDaqAddress

```
procedure SetDaqAddress(Address: Integer; Timeout: Integer);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESESoft® version 7.0 and higher.

`SetDaqAddress` changes the address of the DAQ-module.

Interface: [IModule](#)

Classifier	Name	Type	Description
	Address	Integer	the new address to set for the module
	Timeout	Integer	specifies the timeout-limit. Its unit is [ms]

2.1.73.13 SetModule

```
procedure SetModule(SetType: Integer);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`SetModule` applies the properties of a module which were specified before.

Interface: [IModule](#)

Classifier	Name	Type	Description
	SetType	Integer	must always be 2. (Other values of <code>SetType</code> are for internal use only.)

2.1.73.14 SetPad

```
procedure SetPad(NewAddress: Integer);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`SetPad` sets the address and applies the configuration to a PAD-module.

Interface: [IModule](#)

Classifier	Name	Type	Description
	NewAddress	Integer	the address to set for the PAD-module. The values must be within the range 0...255.

2.1.74 IModules

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

see also: [IApp.Modules](#)

2.1.74.1 Count

```
property Count: Integer
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`Count` is the number of available modules in the [Item](#) list.

Interface: [IModules](#)  read-only

2.1.74.2 Item

property Item[Index: Integer]: IModule

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Item [I] is the module at index I. I is in the range of 0...Count-1.

see also: [IModule](#)

Interface: [IModules](#) read-only

2.1.75 | Nothing

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

2176 | OfflineCalc

interface for offline math calculations

see also: [IApp.OfflineCalc](#)

2.1.76.1 Calculate

```
procedure Calculate();
```

starts the offline calculation

Interface: [IOfflineCalc](#)

2.1.76.2 StoreCalculatedChannels

```
procedure StoreCalculatedChannels();
```

will be called when the calculated channels must be stored

Interface: [OfflineCalc](#)

2.1.77 IPadData

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

2.1.77.1 Address

property Address: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

Address is the PAD-module's address.

Interface: IPadData

2.1.77.2 ConfigCode

property ConfigCode: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

`ConfigCode` is the configuration code of a PAD-module.

Interface: [IPadData](#)

2.1.77.3 CopyToString

```
function CopyToString(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

`CopyToString` returns the data of the PAD-module as a string.

Interface: IPadData

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	the data of the PAD module as string

2.1.77.4 CopyUnitToString

```
function CopyUnitToString(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

`CopyUnitToString` returns the unit of the PAD-module as a string.

Interface: IPadData

Classifier	Name	Type	Description
-	<i>RESULT</i>	WideString	the unit of the PAD-module as a string.

2.1.77.5 Data

property Data[Index: Integer]: Single

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

Data is the measurement data of the PAD-module as an array of Single values.

Interface: [IPadData](#) read-only

2.1.77.6 ModuleAmpl

```
function ModuleAmpl(): Single;
```

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

`ModuleAmpl` returns the module's scale factor.

Interface: IPadData

Classifier	Name	Type	Description
-	<i>RESULT</i>	Single	the module's scale factor

2.1.77.7 ModuleOffset

```
function ModuleOffset(): Single;
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`ModuleOffset` returns the offset of the PAD-module.

Interface: IPadData

Classifier	Name	Type	Description
-	<i>RESULT</i>	Single	the offset of the PAD-module.

2.1.77.8 ModuleType

property ModuleType: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

the type of the PAD-module which can be one of the following:

- 0...Unknown
- 1...PAD-V8
- 2...PAD-VTH
- 3...PAD-TH8
- 4...PAD-RTD3
- 5...PAD-AO1
- 6...PAD-CNT
- 7...PAD-DI8
- 8...PAD-DO7
- 9...PAD-V8-P
- 10...PAD-TH8-P

Interface: [IPadData](#)  read-only

2.1.77.9 Name

property Name: WideString

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Name is the name of the PAD-Module.

Interface: [IPadData](#)  read-only

2.1.77.10 RangeCode

property RangeCode: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

RangeCode is the range code of the PAD-module.

Interface: [IPadData](#)  read/write

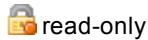
2.1.77.11 RangeIndex

```
property RangeIndex[Index: Integer]: Integer
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`RangeIndex` gives the range code of chosen range at `Index`.

Interface: [IPadData](#)



2.1.77.12 Ranges

```
property Ranges[Index: Integer]: WideString
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`Ranges` gives the description of a range at the index.

Interface: [IPadData](#)



2.1.77.13 RangesCount

```
property RangesCount: Integer
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`RangesCount` defines the number of ranges available for the module.

Interface: [IPadData](#)



2.1.77.14 ShortCopyToString

```
function ShortCopyToString(): WideString;
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

`ShortCopyToString` returns the data of the PAD-module in a short version (only the values) as a string.

Interface: [IPadData](#)

Classifier	Name	Type	Description
-	<code>RESULT</code>	<code>WideString</code>	the data of the PAD-module in a short version (only the values) as a string

2.1.77.15 SpeedCode

property SpeedCode: Smallint

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

The baudrate set on the module. The code is within the range 3...10 and having the following meaning:

- 3...1200 baud
- 4...2400 baud
- 5...4800 baud
- 6...9600 baud
- 7...19200 baud
- 8...38400 baud
- 9...57600 baud
- 10...115200 baud

Interface: [IPadData](#)  read-only

2.1.78 IPlugin

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

old interface for DEWESoft® plugins.

see also: [IPlugin2](#), [IPlugin3](#), [IPlugin4](#)

2.1.78.1 Configure

procedure Configure();

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Interface: [IPlugin](#)

2.1.78.2 Initiate

procedure Initiate();

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

Interface: [IPlugin](#)

2.1.78.3 OnGetData

```
procedure OnGetData();
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see [IPlugin2](#)

Interface: [IPlugin](#)

2.1.78.4 OnStartAcq

```
procedure OnStartAcq();
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see [IPlugin2](#)

Interface: [IPlugin](#)

2.1.78.5 OnStartStoring

```
procedure OnStartStoring();
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see [IPlugin2](#)

Interface: [IPlugin](#)

2.1.78.6 OnStopAcq

```
procedure OnStopAcq();
```

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see [IPlugin2](#)

Interface: [IPlugin](#)

2.1.78.7 OnStopStoring

```
procedure OnStopStoring();
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

see [IPlugin2](#)

Interface: [IPlugin](#)

2.1.78.8 OnTrigger

```
procedure OnTrigger(Time: Double);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

see [IPlugin2](#)

Interface: [IPlugin](#)

Classifier	Name	Type	Description
	Time	Double	-

2.1.78.9 SetApp

```
procedure SetApp(const App: IApp);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

see [IPlugin2](#)

see also: [IApp](#)

Interface: [IPlugin](#)

Classifier	Name	Type	Description
const	App	IApp	-

2.1.78.10 SetData

```
procedure SetData(const Data: IData);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

see also: [IData](#)

Interface: [IPlugin](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description									
const	Data	IData	-									

2.1.79 IPlugin2

interface for DEWEsoft® plugins

child interface: [IPlugin3](#)

see also: [IPlugin1](#), [IPlugin3](#), [IPlugin4](#)

2.1.79.1 ClearChannelsInstance

```
procedure ClearChannelsInstance();
```

`ClearChannelsInstance` is called by DEWEsoft® right before all channels ([IChannel](#)) are destroyed.

The plugin MUST set all references to DEWEsoft® channel instances to [nil](#).

see also: [How to: Mount Dewesoft Channels](#)

Interface: [IPlugin2](#)  use only in plug-ins

2.1.79.2 Configure

```
procedure Configure();
```

`Configure` is called by DEWEsoft® when a user presses the [Configure](#) button within the plug-in tab of the measurement setup screen.

This button will appear, when the plug-in has no frame which could be embedded in the setup screen. In this case the plug-in could open a dialog window to let the user change the plugin-settings.

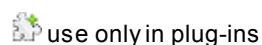
Interface: [IPlugin2](#)  use only in plug-ins

2.1.79.3 HideFrame

```
procedure HideFrame();
```

`HideFrame` is called by DEWEsoft® when the channel setup screen is hidden: e.g. when the user switches to measure or to analyze mode.

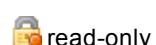
see also: [ShowFrame](#), [UpdateFrame](#), [IPlugin4.OnEvent](#) (evHideAnalysisFrame)

Interface: [IPlugin2](#)

2.1.79.4 Id

property Id: WideString

`Id` is the [GUID](#) (Globally Unique Identifier) of the plug-in library. This identifier is automatically generated by the development environment.

Interface: [IPlugin2](#)

2.1.79.5 Initiate

procedure Initiate(const DweApp: [IApp](#));

`Initiate` is called every time when DEWEsoft® is started or when the plug-in is set to Used/Unused in the hardware setup.

see also: [IApp](#)Interface: [IPlugin2](#)

Classifier	Name	Type	Description
const	DweApp	IApp	the DEWEsoft® application instance

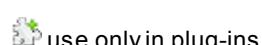
2.1.79.6 LoadSetup

procedure LoadSetup(Data: OleVariant);

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see [XML Setup](#)

`LoadSetup` is called by DEWEsoft® every time a channel setup is loaded. This corresponds to the menu item [File - Load Setup](#). It will also be called e.g. when you switch from analyze mode to measurement setup.

see also [SaveSetup](#)Interface: [IPlugin2](#)

Classifier	Name	Type	Description
	Data	OleVariant	Data can contain any information about special settings of the plug-in if it has been stored to the setup file (.dss) at SaveSetup.

2.1.79.7 NewSetup

```
procedure NewSetup();
```

NewSetup is called by DEWESoft® when a new setup is created.(corresponds to [File - New Setup](#)). NewSetup may also be called at startup of DEWESoft®.

Interface: [IPlugin2](#)



2.1.79.8 OnGetData

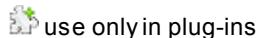
```
procedure OnGetData();
```

OnGetData is called by DEWESoft® periodically (about every 40ms - see [IApp.TimerInterval](#)). Inside of this function you can read data from other channels and can add data to the plug-in channels.

DEWESoft® will make sure, that the data you read is consistent (there is no need to call any synchronization functions, like [IData.StartDataSync](#), [IData.EndDataSync](#)).

see also: [How to: Write Data To Channels](#), [How to: Read Data From Channels](#)

Interface: [IPlugin2](#)



2.1.79.9 OnOleMsg

```
procedure OnOleMsg(Msg: Integer; Param: Integer);
```

OnOleMsg will be called by DEWESoft® after [IApp.MainWndMessage](#) has been called by a thread, other than the main thread, of a plug-in.

see also: [IApp.MainWndMessage](#)

Interface: [IPlugin2](#)



Classifier	Name	Type	Description
	Msg	Integer	Msg and Param correspond to the arguments of IApp.MainWndMessage which are named equal. These parameters can be used to specify the message which has to be sent to DEWESoft®.
	Param	Integer	

2.1.79.10 OnStartAcq

```
procedure OnStartAcq();
```

OnStartAcq is called by DEWESoft® right after the start of the acquisition; i.e. when it is switched to one of the

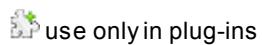
measurement screens (e.g. *Overview*, *Recorder*, etc.).

This will also be called when you start storing from channel setup directly without switching to the measurement screens first.

Note: this is the same as: [IPlugin3.OnAfterStartAcq](#)

see also: [IPlugin3.OnBeforeStartAcq](#), [OnStopAcq](#), [OnStartStoring](#)

Interface: [IPlugin2](#)



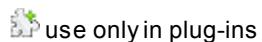
2.1.79.11 OnStartStoring

```
procedure OnStartStoring();
```

OnStartStoring is called by DEWESoft® when the storing of data is started or a trigger for storing is armed, corresponding to the *Store* button.

see also: [IPlugin2.OnStartAcq](#)

Interface: [IPlugin2](#)



2.1.79.12 OnStopAcq

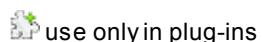
```
procedure OnStopAcq();
```

OnStopAcq is called by DEWESoft® when the acquisition is stopped. This can be by clicking the *Stop* button, switching to the setup screen or to the analyze mode or by closing DEWESoft®.

It is called after [IPlugin3.OnBeforeStopAcq](#) and before [IPlugin3.OnAfterStopAcq](#).

see also: [IPlugin3.OnBeforeStopAcq](#), [IPlugin3.OnAfterStopAcq](#)

Interface: [IPlugin2](#)



2.1.79.13 OnStopStoring

```
procedure OnStopStoring();
```

OnStopStoring is called by DEWESoft® when the storing of data is stopped or a trigger for storing data is unarmed. This can be by clicking the *Stop* button, switching to the setup screen or to the analyze mode or by closing DEWESoft®.

Interface: [IPlugin2](#)



2.1.79.14 OnTrigger

```
procedure OnTrigger(Time: Double);
```

`OnTrigger` is called by DEWEsoft® every time a trigger occurs.

Interface: [IPlugin2](#)

 use only in plug-ins

Classifier	Name	Type	Description
	Time	Double	the time when the trigger event occurred

2.1.79.15 SaveSetup

```
procedure SaveSetup(var Data: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESESoft® version 7.0 and higher.

see [XML Setup](#)

`SaveSetup` is called by DEWEsoft® when a channel setup is saved. This corresponds to [File - Save Setup](#).

Interface: [IPlugin2](#)  use only in plug-ins

Classifier	Name	Type	Description
var	Data	OleVariant	Data is of the type OleVariant and can be used to specific settings for the plug-in to a setupfile (.dss). These data will read back on LoadSetup .

2.1.79.16 ShowFrame

```
function ShowFrame(Parent: Integer): WordBool;
```

The plug-in must return `TRUE` if it has a channel setup frame to show, otherwise `FALSE`. (e.g. when using Visual Basic, it's not possible to have a setup frame).

see also: [HideFrame](#), [UpdateFrame](#), [IPlugin4.OnEvent\(evShowAnalysisFrame\)](#)

Interface: [IPlugin?](#)

 use only in plug-ins

Classifier	Name	Type	Description
	Parent	Integer	the ID of the parent for the setup frame
-	RESULT	WordBool	must return TRUE if it has a setup frame to show, otherwise FALSE .(e.g. when using Visual Basic, it's not possible to have a setup frame)

2.1.79.17 UpdateFrame

```
procedure UpdateFrame();
```

Will be called by DEWESoft® periodically (e.g. every 0.1 seconds) for the currently active channel setup frame.plugin.

see also: [HideFrame](#), [ShowFrame](#)

Interface: [IPlugin2](#)



use only in plug-ins

2.1.79.18 Used

```
property Used: WordBool
```

TRUE if the plugin is set to *Used* in the *Hardware setup*, otherwise FALSE.

Interface: [IPlugin2](#)



read/write



use only in plug-ins

2.1.80 IPlugin3

extended version of the [IPlugin2](#) interface for DEWESoft® plugins

parent interface: [IPlugin2](#)

see also: [IPlugin3](#), [IPlugin4](#)

2.1.80.1 GetDWTypeLibVersion

```
function GetDWTypeLibVersion(): Integer;
```

GetDWTypeLibVersion the version number of the DEWESoft® type library must be returned. It is defined in the DEWESoft® type library as DEWEsoftMinorVersion.

Interface: [IPlugin3](#)



use only in plug-ins

Classifier	Name	Type	Description
-	RESULT	Integer	the version number of the DEWESoft® type library

2.1.80.2 OnAfterCalcMath

```
procedure OnAfterCalcMath();
```

`OnAfterCalcMath` is called after the Math-channels have been calculated.

This procedure should be used if the results of the Math-channels are relevant for the plug-in. This procedure is similar to [IPlugin2.OnGetData](#), but is called after calculation of the Math-channels. (*DEWEsoft®* channels are processed in the following order: *Devices*, *Plug-ins*, *Math*)

Interface: [IPlugin3](#)

2.1.80.3 OnAfterStartAcq

```
procedure OnAfterStartAcq();
```

OnAfterStartAcq is called by DEWEsoft® right after the acquisition has been started -

this is the same, as [IPlugin2.OnStartAcq](#).

see also: [IPlugin2.OnStartAcq](#), [IPlugin3.OnBeforeStartAcq](#), [OnStopAcq](#), [OnStartStoring](#)

Interface: [IPlugin3](#)

2.1.80.4 OnAfterStopAcq

```
procedure OnAfterStopAcq();
```

`OnAfterStopAcq` is called by DEWEsoft® after the acquisition has been stopped.

It is the same as [IPlugin2.OnStopAcq](#).

see also: [IPlugin2.OnStopAcq](#), [IPlugin3.OnBeforeStopAcq](#)

Interface: [IPlugin3](#)  use only in plug-ins

2.1.80.5 OnAlarm

```
procedure OnAlarm(CondIndex: Integer; Status: WordBool);
```

`OnAlarm` is called by *DEWEsoft®* when an alarm condition changes its status.

Interface: [IPlugin3](#) use only in plug-ins

Classifier	Name	Type	Description
	CondIndex	Integer	gives the index of the related alarm condition
	Status	WordBool	TRUE if the alarm has become active, FALSE if the alarm has been reset

2.1.80.6 OnBeforeStartAcq

```
procedure OnBeforeStartAcq(var AllowStart: WordBool);
```

OnBeforeStartAcq is called by DEWEsoft® before the acquisition is started and before [IPlugin2.OnStartAcq](#). In this procedure some kind of initialization can be done by the plug-in.

see also: [IPlugin2.OnStartAcq](#), [IPlugin3.OnAfterStartAcq](#), [OnStopAcq](#), [OnStartStoring](#)

Interface: [IPlugin3](#) use only in plug-ins

Classifier	Name	Type	Description
var	AllowStart	WordBool	the plug-in can set this parameter to FALSE to prevent the start of the acquisition

2.1.80.7 OnBeforeStopAcq

```
procedure OnBeforeStopAcq(var AllowStop: WordBool);
```

OnBeforeStopAcq is called before the acquisition is stopped.

see also: [IPlugin2.OnStopAcq](#), [IPlugin3.OnAfterStopAcq](#)

Interface: [IPlugin3](#) use only in plug-ins

Classifier	Name	Type	Description
var	AllowStop	WordBool	defines whether the acquisition is allowed to stop or not. Its default value is TRUE, but the plug-in code may set this to FALSE.

2.1.80.8 OnBigListLoad

```
procedure OnBigListLoad(const TextSetup: WideString);
```

For internal use only! This feature must only be used by DEWEsoft® itself. Do not use in automation applications or add-ons.

Interface: [IPlugin3](#) use only in plug-ins

Classifier	Name	Type	Description
const	TextSetup	WideString	

2.1.80.9 OnExit

```
procedure OnExit();
```

`OnExit` is called right before *DEWESoft®* is closed.

Interface: IPlugin3



'use only in plug-ins

2.1.80.10 OnGetClock

```
procedure OnGetClock(var ClockLow: Integer; var ClockHigh: Integer);
```

~~ONCE~~CLOCK a clock (time information) can

[View Details](#)



[View Details](#) | [Edit](#) | [Delete](#)

Interface: IPlugin3	 use only in plug-ins		
Classifier	Name	Type	Description
var	ClockLow	Integer	low part of the 64-bit value
var	ClockHigh	Integer	high part of the 64-bit value

2.1.80.11 OnGetSetupData

```
procedure OnGetSetupData();
```

`OnGetSetupData` is similar to `IPlugin2.OnGetData`, but will be called periodically when the setup screen is active.

`OnGetSetupData` can be used for displaying acquired data during setup.

Interface: [IPlugin3](#)



Use only in plug-ins

2.1.80.12 OnHideHWFrm

```
procedure OnHideHWFrame();
```

will be called when the hardware frame is hidden

see also: [OnShowHWFrame](#)

Interface: IPlugin3

2.1.80.13 OnRepaintFrame

```
procedure OnRepaintFrame();
```

OnRepaintFrame is called frequently (about 10 times per second) and can be used to update some information displayed in the plug-in frame.

see also: [IPlugin4.OnEvent](#) (evRepaintAnalysisFrame)

Interface: [IPlugin3](#)  use only in plug-ins

2.1.80.14 OnResizeFrame

```
procedure OnResizeFrame(Width: Integer; Height: Integer);
```

OnResizeFrame will be called by DEWESoft® whenever the plugin needs to resize it's setup frame.

see also: [IPlugin4.OnEvent](#) (evResizeAnalysisFrame)

Interface: [IPlugin3](#)

Classifier	Name	Type	Description
	Width	Integer	width of the setup screen
	Height	Integer	height of the setup screen

2.1.80.15 OnShowHWFrame

```
function OnShowHWFrame(Parent: Integer): WordBool;
```

will be called when the hardware frame is shown.

Plugins should create their form in this function and assign the Parent to the new form.

see also: [OnHideHWFrame](#)

Interface: [IPlugin3](#)

Classifier	Name	Type	Description
	Parent	Integer	handle of the parent window
-	RESULT	WordBool	return TRUE if everything is okay

2.1.80.16 OnStartSetup

```
procedure OnStartSetup();
```

`OnStartSetup` is called by *DEWESoft®* on entering the setup screen.

see also: [OnStopSetup](#)

Interface: [IPlugin3](#) use only in plug-ins

2.1.80.17 OnStopSetup

```
procedure OnStopSetup();
```

`OnStopSetup` is called by *DEWESoft®* on leaving the setup screen.

see also: [OnStartSetup](#)

Interface: [IPlugin3](#)

2.1.80.18 OnTriggerStop

```
procedure OnTriggerStop(Time: Double; TrigDuration: Double);
```

`OnStopTrigger` is called by *DEWESoft®* when the stop-trigger event happens.

Interface: [IPlugin3](#)

Classifier	Name	Type	Description
	Time	Double	the timestamp when the stop-trigger condition occurred
	TrigDuration	Double	the duration of the trigger event

2.1.80.19 ProvidesClock

```
procedure ProvidesClock(var Value: WordBool);
```

`ProvidesClock` sets whether the plug-in provides a clock to *DEWESoft®* or not.

see also: [OnGetClock](#)

see also: [IApp.MasterClock](#)

Interface: [IPlugin3](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
var	Value	WordBool	must be set to <code>True</code> if the plug-in provides a clock. The default value is <code>False</code> .								

2.1.80.20 SetCANPort

```
procedure SetCANPort(Port: Integer);
```

Setting a CAN-port for using it for the plug-in.

Interface: [IPlugin3](#)

 use only in plug-ins

Classifier	Name	Type	Description
	Port	Integer	CAN port number

2.1.81 IPlugin4

interface for DEWEsoft® plugins

see also: [IPlugin](#), [IPlugin3](#), [IPlugin4](#)

2.1.81.1 OnEvent

```
procedure OnEvent(EventID: EventIDs;
; InParam: OleVariant; var OutParam: OleVariant);
```

a generic event function that can have input and output parameters.

Interface: [IPlugin4](#)

Classifier	Name	Type	Description
	EventID	EventIDs	the ID of the event: see table below for details
	InParam	OleVariant	the in parameter/s of the event: see table below for details
var	OutParam	OleVariant	the out parameter/s of the event: see table below for details

De c	He x	Name	InParam	OutParam	Description
10 0	0x 64	evOnInitiateAc q	Null	Null	called before the acquisition is started
10 1	0x 65	evOnAlarmEx	is an array: [Index] name: type [0] CondIndex: Integer [1] Status: Boolean [2] AlarmTime: Double	Null	extended version of IPlugin3.OnAlarm , which also includes the time of the alarm

De c	He x	Name	InParam	OutParam	Description
10 2	0x 66	evOnGetSampleR ates	<u>Null</u>	array of Double	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
10 3	0x 67	evOnSetSampleR ate	Double	Double	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
10 4	0x 68	evOnInitiateSe tup	Double (current sample rate)	Double (new sample rate)	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
10 5	0x 69	evOnInitiateHa rdware	<u>Null</u>	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
10 6	0x 6A	evHasAbsoluteC lock	<u>Null</u>	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
10 7	0x 6B	evOnCheckSampl eRate	Double	Double	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
10 8	0x 6C	evOnUpdateXML	is an array: [Index] name: type [0] DomDocument: IDomDocument [1] DomNode: IDomNode [2] Write: WordBool [3] DataFile: Boolean [4] ChannelHelper: IPluginChannelXMLHelp er [5] SetupMessages: ISetupMessages	XMLWritten: WordBool TRUE if the function wrote the XML setup FALSE if not	will be called to read/write the channel setup in XML format see also: IChannel.UpdateXML note: parameter 5 (SetupMessages) is optional (so: it may be missing and then the array has only 4 elements!)

De c	He x	Name	InParam	OutParam	Description
10 9	0x 6D	evIsMasterCloc k	Null	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
11 0	0x 6E	evShowAnalysis Frame	Parent: Integer handle of the parent window	must return TRUE if it has a setup frame to show, otherwise FALSE (e.g. when using Visual Basic, it's not possible to have a setup frame)	is called by DEWEsoft® when the plugin should show it's channel setup frame: see also: IPlugin2.ShowFrame
11 1	0x 6F	evUpdateAnalys isFrame	Null	Null	see evRepaintAnalysisFram e
11 2	0x 70	evRepaintAnaly sisFrame	Null	Null	OnRepaintFrame is called frequently (about 10 times per second) and can be used to update some information displayed in the plug-in frame. see also: IPlugin3. OnRepaintFrame
11 3	0x 71	evHideAnalysis Frame	Null	Null	is called by DEWEsoft® when the plugin should hide it's channel setup frame: see also: IPlugin2.HideFrame
11 4	0x 72	evResizeAnaly sisFrame	is an array: [Index] name: type [0] Width: Integer [1] Height: Integer	Null	OnResizeFrame will be called by DEWEsoft® whenever the plugin needs to resize it's setup frame. see also: IPlugin3. OnResizeFrame
11 5	0x 73	evStartAnalyti cs	Null	Null	called when the analysis is started see also: evPrepareAnalysis , evCalculateAnalysis , evStopAnalysis
11 6	0x 74	evStopAnalysis	Null	Null	called when the analysis is stopped see also: evStartAnalysis , evPrepareAnalysis , evCalculateAnalysis
11 7	0x 75	evCalculateAna lysis	Null	Null	called when the analysis should be calculated
11 8	0x 76	evPrepareAnaly sis	InputList: IChannelListEx	Null	called before the analysis is done see also: evStartAnalysis , evCalculateAnalysis , evStopAnalysis

De c	He x	Name	InParam	OutParam	Description
11 9	0x 77	evAfterLoadFil e	<u>Null</u>	<u>Null</u>	called after a file has been loaded for analyse mode
12 0	0x 78	evPreInitiate	<u>Null</u>	<u>Null</u>	is called before the initialization is done
12 1	0x 79	evOnUpdateHWXM L	is an array: [Index] name: type [0] DomDocument: IDomDocument [1] DomNode: IDomNode [2] Write: WordBool	XMLWritten: WordBool TRUE if the function wrote the XML setup FALSE if not	will be called to read/write the hardware setup in XML format see also: IChannel.UpdateXML
12 2	0x 7A	evGetIndexLeve lName	is an array: [Index] name: type [0] IndexLevel: Integer [1] PrevIndices: array of LongWord	String	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
12 3	0x 7B	evGetInputGrou ps	is an array: [Index] name: type [0] InputGroups: IInputGroups [1] AllCh: array of Boolean	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
12 4	0x 7C	evGetReplayMod e	<u>Null</u>	WordBool	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
12 5	0x 7D	evSetRegistrat ionHelper	IRegistrationHelper	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
12 6	0x 7E	evSetDWVersion	String	<u>Null</u>	will be called by DEWESoft® to set the DEWESoft® version
12 7	0x 7F	evNewDataFileC reated	<u>Null</u>		<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
12 8	0x 80	evGetStartErro rParams	<u>Null</u>	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

De c	He x	Name	InParam	OutParam	Description
12 9	0x 81	evEstablishConnections	Null	Null	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
13 0	0x 82	evEnterHardwareSetup	Null	Null	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
13 1	0x 83	evChannelRemoved	IChannel	Null	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
13 2	0x 84	evGetDisabledChannels	IChannelList	Null	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
13 3	0x 85	evOnEnterPluginSetupScreen	Null	Null	called when the user enters the channel setup screen of the plugin see also: evOnLeavePluginSetupScreen
13 4	0x 86	evOnLeavePluginSetupScreen	Null	Null	called when the user leaves the channel setup screen of the plugin see also: evOnEnterPluginSetupScreen
13 5	0x 87	evGetFreeCode	Integer	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
13 6	0x 88	evGetMaxCalcDelay	Null	Single	get the maximum calculation delay of the plugin in seconds: see Calculation Delay
13 7	0x 89	evOnPrepareAcq	Null	Null	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

De c	He x	Name	InParam	OutParam	Description
13 8	0x 8A	evFileStoringF inished	String	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
13 9	0x 8B	evEnableZero	<u>Null</u>	Boolean	return <i>TRUE</i> if your plugin supports zeroing in measuremode see also: evOnSetZero below, ZeroAllAutoChannels
14 0	0x 8C	evOnSetZero	<u>Null</u>	<u>Null</u>	this event will be called when the user presses the Zero button in measure mode see also: evEnableZero above, ZeroAllAutoChannels
14 1	0x 8D	evOnClearData	<u>Null</u>	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
14 2	0x 8E	evHasDisplayTe mplate	<u>Null</u>	Boolean	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
14 3	0x 8F	evUpdateDispla yTemplate	IDisplayTemplate	EmptyParam	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
14 4	0x 90	evOnChannelVal ueChanged	is an array: [Index] name: type [0] Channel: IChannel [1] Value: Double	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
14 5	0x 91	evOnChannelEna bled	IChannel	<u>Null</u>	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
14 6	0x 92	evChangeIndex	IIndexChanger	<u>Null</u>	can be used by the plugin to change the index structure
14 7	0x 93	evGetDisplayNa me	<u>Null</u>	String	<i>Reserved for internal use!</i> Ask

De c	He x	Name	InParam	OutParam	Description
					our support if you need to know details: support@dewesoft.com

2.1.82 IPluginChannel

a special data channel to be used in plugins

see also: [IChannel](#), [IPluginGroup.MountChannel](#), [IPluginGroup.MountChannelEx](#)

2.1.82.1 AlwaysReserveMemoryInSetup

property AlwaysReserveMemoryInSetup: WordBool

Set this to TRUE if you want to write data to the plugin channel while the user is in channel setup mode.

If it is set to FALSE, you can only add data in measure mode.

Interface: [IPluginChannel](#)  read/write

2.1.82.2 AsyncBufSize

property AsyncBufSize: Integer

to set the buffer size manually (usually DEWEsoft® will choose a reasonable buffer size depending on the expected async rate - [IChannel.ExpectedAsyncRate](#)).

call [ReserveMemory](#) after you have set it.

Interface: [IPluginChannel](#)  read/write

2.1.82.3 DefaultMax

property DefaultMax: Double

the default maximum value to use in visual controls.

Interface: [IPluginChannel](#)  read/write

2.1.82.4 DefaultMin

property DefaultMin: Double

the default minimum value to use in visual controls.

Interface: [IPluginChannel](#) read/write

2.1.82.5 DefaultRes

property DefaultRes: Integer

the default resolution of this channel (i.e. number of decimal places to display)

Interface: [IPluginChannel](#) read/write

2.1.82.6 FreeMemory

```
procedure FreeMemory();
```

to free the memory that has been reserved for the channel.

You may need to call this when you change the data type of

see also: [ReserveMemory](#), [AlwaysReserveMemoryInSetup](#)

Interface: [IPluginChannel](#)

2.1.82.7 LongName

property LongName: WideString

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

Interface: [IPluginChannel](#)

2.1.82.8 MarkAsOffline

```
procedure MarkAsOffline();
```

call this function to indicate that this channel is offline: i.e. it will not be filled with data during measurement, but only later in **Analyse mode**.

Interface: [IPluginChannel](#)

2.1.82.9 PluginGUID

property PluginGUID: WideString

the [GUID](#) of the plugin, that this channel belongs to.

Interface: [IPluginChannel](#)  read-only

2.1.82.10 ReserveMemory

procedure ReserveMemory(DBSize: Integer);

in DEWEsoft® 7 you can always pass -1 for the DBSize parameter. DEWEsoft® will handle the buffersizes.

see also: [AsyncBufSize](#)Interface: [IPluginChannel](#)

Classifier	Name	Type	Description
	DBSize	Integer	in DEWEsoft® 7 you can always pass -1 for the DBSize parameter. DEWEsoft® will handle the buffersizes.

2.1.82.11 SetChNo

procedure SetChNo(const Value: WideString);

This procedure can be used to override the default channel number that will be shown in the channel setup of the *Analysis* mode - column *Ch. no*:

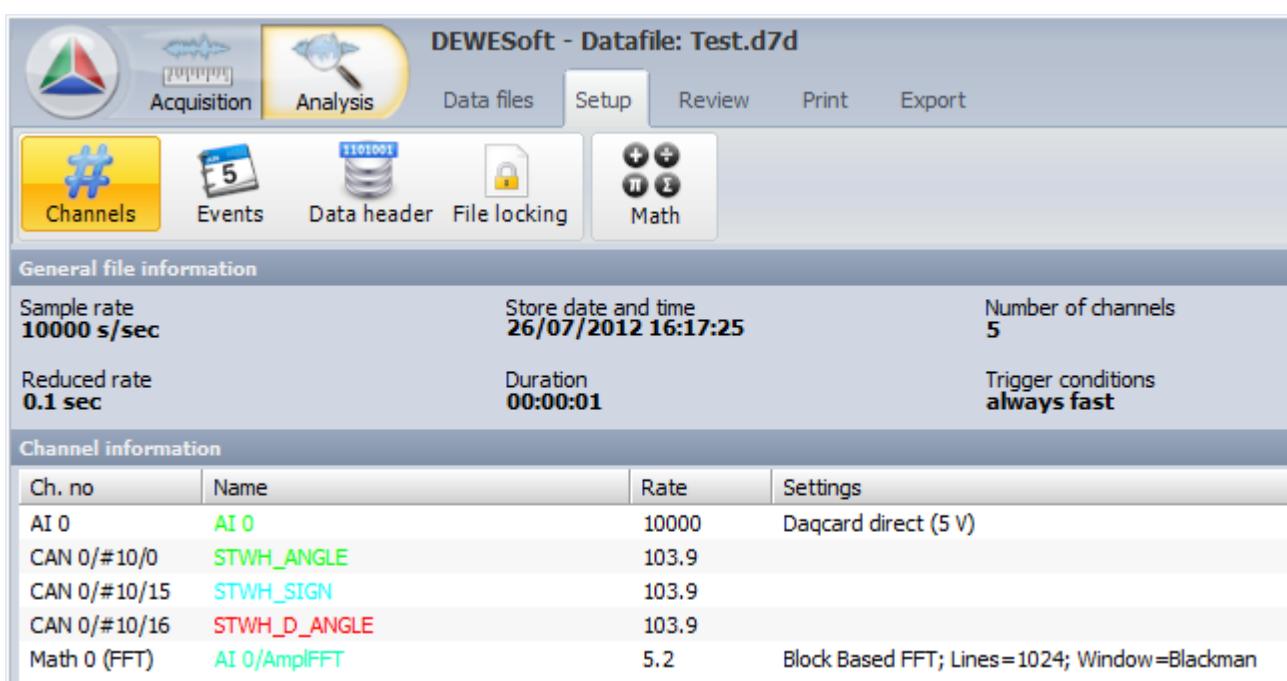


Illustration 179: Analysis mode: channel setup

Interface: [IPluginChannel](#)

Classifier	Name	Type	Description
const	Value	WideString	an arbitrary string to describe the channel number

2.1.82.12 SetIndex

```
procedure SetIndex(Level: Integer; Ind: OleVariant);
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

the channel index should already be set by [IPluginGroup.MountChannelEx](#)

Interface: [IPluginChannel](#)

Classifier	Name	Type	Description
	Level	Integer	
	Ind	OleVariant	

2.1.82.13 SetSettings

```
procedure SetSettings(const Value: WideString);
```

This procedure can be used to set custom settings that will be shown in the channel setup of the *Analysis* mode - column **Settings:**

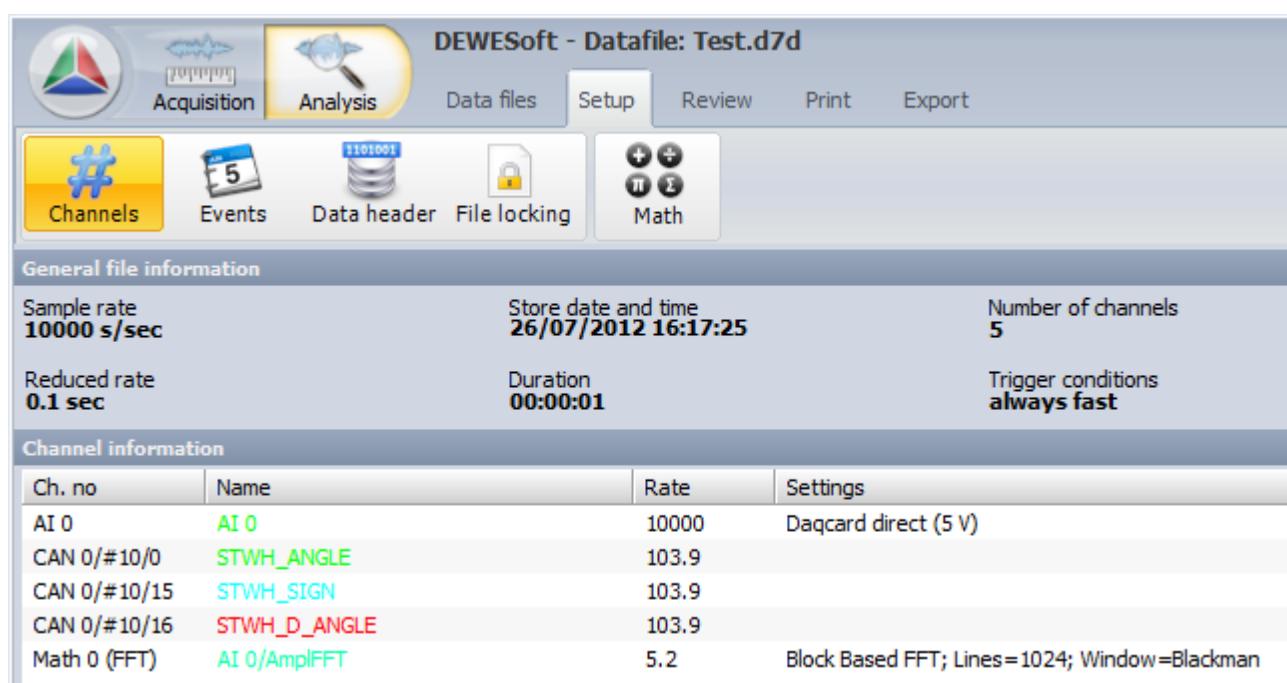


Illustration 180: Analysis mode: channel setup

Interface: [IPluginChannel](#)

Classifier	Name	Type	Description
const	Value	WideString	an arbitrary string, describing the settings of this channel

2.1.83 IPluginChannelXMLHelper

This is a project specific feature.

helper class for reading/writing plugin channels

see also: [evOnUpdateXML](#) in [IPlugin4.OnEvent](#)

2.1.83.1 ExtractNextChannel

```
function ExtractNextChannel  
(out IndexLevel: Integer; out Ind: OleVariant): WordBool;
```

This is a project specific feature.

call [StartExtractChannels](#) first to initialize the iteration over all channels and then call

in a loop

until it returns FALSE.

Interface: [IPluginChannelXMLHelper](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	
Classifier	Name	Type	Description										
out	IndexLevel	Integer	the index level of the channel. see also Channel Index										
out	Ind	OleVariant	the index array of the channel. see also Channel Index										
-	RESULT	WordBool	FALSE for the last channel - TRUE otherwise.										

2.1.83.2 FindNode

```
function FindNode
(IndexLevel: Integer; Ind: OleVariant; ChannelNode: WordBool; AllowCreate: WordBool
): OleVariant;
```

This is a project specific feature.

will find a channel node

Interface: [IPluginChannelXMLHelper](#)

Classifier	Name	Type	Description
	IndexLevel	Integer	
	Ind	OleVariant	
	ChannelNode	WordBool	
	AllowCreate	WordBool	
-	RESULT	OleVariant	

2.1.83.3 MountAllChannels

```
procedure MountAllChannels();
```

This is a project specific feature.

will mount all channels of the xml setup

Interface: [IPluginChannelXMLHelper](#)

2.1.83.4 StartExtractChannels

```
procedure StartExtractChannels();
```

This is a project specific feature.

call this function to initialize the iteration over all channels and then call [ExtractNextChannel](#) in a loop until it returns FALSE.

see also: [ExtractNextChannel](#)

Interface: [IPluginChannelXMLHelper](#)

2.1.84 IPluginGroup

special group for plugins: see [IData_Groups](#) [8]

parent interface: [IChannelGroup](#)

see also: [IChannelGroups](#)

2.1.84.1 AddIndexName

```
procedure AddIndexName(Level: Integer; Ind: OleVariant; const Name: WideString);
```

to add a name for an index see [Channel Index Example](#)

see also: [AddIndexNameEx](#)

Interface: [IPluginGroup](#)

Classifier	Name	Type	Description
	Level	Integer	the number of items in the Ind array to process
	Ind	OleVariant	array of integers
const	Name	WideString	the name will be shown in the measurement channel list

2.1.84.2 AddIndexNameEx

```
procedure AddIndexNameEx
(const PluginGUID: WideString; Level: Integer; Ind: OleVariant; const Name: WideString);
```

to add a name for an index see [Channel Index Example](#)

see also: [AddIndexName](#)

Interface: [IPluginGroup](#)

Classifier	Name	Type	Description
const	PluginGUID	WideString	the GUID of the plugin
	Level	Integer	the number of items in the Ind array to process
	Ind	OleVariant	array of integers
const	Name	WideString	the name will be shown in the measurement channel list

2.1.84.3 ClearAllChannels

```
procedure ClearAllChannels();
```

This is a project specific feature.

To clear all mounted channels of this plugin group.

Interface: [IPluginGroup](#) use only in plug-ins

2.1.84.4 FindChannel

```
function FindChannel  
(const PluginGUID: WideString; IndexLevel: Integer; Ind: OleVariant; AllowMount: Wo  
rldBool): IChannel;
```

to find a plugin channel by index and optionally mount it if it does not exist yet.

see also: [IChannel](#)

Interface: [IPluginGroup](#)

Classifier	Name	Type	Description
const	PluginGUID	WideString	the GUID of the plugin that the channel belongs to
	IndexLevel	Integer	the number of items in the Ind array to process
	Ind	OleVariant	the index array (array of integer) - see Channel Index
	AllowMount	WordBool	TRUE: if the channel does not exist yet, it will be mounted
-	RESULT	IChannel	the channel that has been found (or mounted)

2.1.84.5 FindInputGroup

```
function FindInputGroup  
(const PluginGUID: WideString; IndexLevel: Integer; Ind: OleVariant): IInputGroup;
```

Interface: [IPluginGroup](#)

Classifier	Name	Type	Description
const	PluginGUID	WideString	the GUID of the plugin
	IndexLevel	Integer	the index level (relative to plugin)
	Ind	OleVariant	the Index array
-	RESULT	IIInputGroup	the found index group or <code>nil</code>

2.1.84.6 MountChannel

```
function MountChannel(DataType: Integer; Async: WordBool; DBSize: Integer):  
IChannel;
```

DEPRECATED! This feature is deprecated and should not be used in DEWESoft® version 7.0 and higher.

see: [IChannel](#), [MountChannelEx](#)

Interface: [IPluginGroup](#)  use only in plug-ins

Classifier	Name	Type	Description
	DataType	Integer	see Data Types
	Async	WordBool	TRUE if it's an <i>asynchronous</i> channel, FALSE for <i>synchronous</i> channels: see Synchronism
	DBSize	Integer	can be set to -1 in DEWESoft® 7, so that DEWESoft® will choose a suitable buffer size
-	RESULT	IChannel	the mounted channel

2.1.84.7 MountChannelEx

```
function MountChannelEx  
(const PluginGUID: WideString; IndexLevel: Integer; Ind: OleVariant): IChannel;
```

Extended version of [MountChannel](#), where you can also specify the exact channel index (see [Channel Index](#)).

Note: you only must provide the part of the channel index relative to your plugin.

Example how to mount a channel in the [MountChannel](#) function:

```
Ind := VarArrayCreate([0, 0], varInteger);  
// the value of the channel index must be unique - for the 1st channel we choose the value 0  
Ind[0] := 33;  
// now we mount (create) the 1st channel  
FChOutput1 := IPluginGroup.MountChannelEx(  
    GUIDToString(CLASS_TPluginExample),  
    1, Ind);
```

Note, that we only provide one index level and choose the channel index 33 for the channel (usually you would start with index 0, not 33 - but 33 makes it easier to see which part of the channel index it is, in this example).

The complete channel index of this mounted channel is this:

```
[0, 100000, 78787878, 33]
```

The first 3 values are defined by DEWESoft® (see [Channel Index](#) for more details):

- 0... means that the channel is mounted on the local machine (i.e. it is not a channel on a remote machine, which is transferred via the NET-Option)
- 100000...is a fixed value for all plugins

- 78787878...is a value that is calculated out of the plugin [GUID](#)

and only the last value (33), is what you have specified in your [MountChannel](#) function.

see also: [How to: Mount Dewesoft Channels](#), [IChannel](#)

Interface: [IPluginGroup](#)  use only in plug-ins

Classifier	Name	Type	Description
const	PluginGUID	WideString	GUID of the plugin
	IndexLevel	Integer	the number of items in the <code>Ind</code> array to process
	Ind	OleVariant	the index array (array of integer) - see Channel Index
-	RESULT	IChannel	the channel that has been mounted

2.1.84.8 MountInputGroup

```
function MountInputGroup(const PluginGUID: WideString; Index: OleVariant):  
  IInputGroup;
```

to mount an input group for the plugin

see also: [IInputGroup](#)

Interface: [IPluginGroup](#)

Classifier	Name	Type	Description
const	PluginGUID	WideString	the GUID of the plugin
	Index	OleVariant	index of teh input group: see Channel Index Example
-	<i>RESULT</i>	IInputGroup	the input group that has been mounted

2.1.84.9 UnmountChannel

```
procedure UnmountChannel(var Channel: IChannel);
```

`UnmountChannel` allows removing one channel separately.

see also: [IChannel](#)

Interface: [IPluginGroup](#)

Classifier	Name	Type	Description
var	Channel	IChannel	the channel that should be unmounted

2.1.85 IPluginLicense

internal use for plugins made by *DEWESoft®*.

child interfaces: [IPluginLicense2](#)

2.1.85.1 GetHardwareCode

```
function GetHardwareCode(): WideString;
```

internal use for plugins made by *DEWESoft®*.

Interface: [IPluginLicense](#)

Classifier	Name	Type	Description
-	RESULT	WideString	

2.1.85.2 GetRegTypeWanted

```
function GetRegTypeWanted(): Integer;
```

internal use for plugins made by *DEWESoft®*.

Interface: [IPluginLicense](#)

Classifier	Name	Type	Description
-	RESULT	Integer	

2.1.85.3 GetTrustedCode

```
function GetTrustedCode(Param: Integer): Integer;
```

internal use for plugins made by *DEWESoft®*.

Interface: [IPluginLicense](#)

Classifier	Name	Type	Description
	Param	Integer	
-	RESULT	Integer	

2.1.85.4 SetLicenseCode

```
function SetLicenseCode(const LicenseCode: WideString): WordBool;
```

internal use for plugins made by DEWESoft®.

Interface: [IPluginLicense](#)

Classifier	Name	Type	Description
const	LicenseCode	WideString	
-	RESULT	WordBool	

2.1.86 IPluginLicense2

internal use for plugins made by DEWESoft®.

parent interface: `IPluginLicense`

2.1.86.1 GetLicenseCode

```
function GetLicenseCode(Builtin: WordBool): WideString;
```

internal use for plugins made by DEWESoft®.

Interface: [IPluginLicense2](#)

Classifier	Name	Type	Description
	Builtin	WordBool	
-	RESULT	WideString	

2.1.87 IPowerModule

This is a project specific feature.

2.1.87.1 FFTBlockSize

property FFTBlockSize: Integer

This is a project specific feature.

Interface: [IPowerModule](#) read-only

2.1.87.2 FFTSampleRate

```
property FFTSampleRate: Single
```

This is a project specific feature.

Interface: [IPowerModule](#)  read-only

2.1.87.3 GetFFT

```
procedure GetFFT(ValueType: Integer; Phase: Integer; out Data: OleVariant);
```

This is a project specific feature.

Interface: [IPowerModule](#)

Classifier	Name	Type	Description
	ValueType	Integer	
	Phase	Integer	
out	Data	OleVariant	

2.1.87.4 GetVectorScopeData

```
function GetVectorScopeData(): OleVariant;
```

This is a project specific feature.

Interface: [IPowerModule](#)

Classifier	Name	Type	Description
-	RESULT	OleVariant	

2.1.87.5 LoadFromXML

```
procedure LoadFromXML(AType: XMLType; XML: OleVariant);
```

This is a project specific feature.

Interface: [IPowerModule](#)

Classifier	Name	Type	Description
	AType	XMLType	
	XML	OleVariant	

2.1.87.6 LoadFromXML1

```
procedure LoadFromXML1(AType: Integer; XML: OleVariant);
```

This is a project specific feature.

Interface: [IPowerModule](#)

Classifier	Name	Type	Description
	AType	Integer	
	XML	OleVariant	

2.1.87.7 ModuleIndex

property ModuleIndex: Integer

This is a project specific feature.

Interface: [IPowerModule](#) read-only

2.1.87.8 SaveToXML

```
procedure SaveToXML(AType: XMLType; var XML: OleVariant);
```

This is a project specific feature.

see also: [XMLType](#)

Interface: [IPowerModule](#)

Classifier	Name	Type	Description
	AType	XMLType	
var	XML	OleVariant	

2.1.87.9 SaveToXML1

```
procedure SaveToXML1(AType: Integer; out XML: OleVariant);
```

This is a project specific feature.

Interface: [IPowerModule](#)

Classifier	Name	Type	Description
	AType	Integer	
out	XML	OleVariant	

2.1.88 IPowerModules

This is a project specific feature.

see also: [IApp.PowerModules](#)

2.1.88.1 Add

```
function Add(): IPowerModule;
```

This is a project specific feature.

see also: [IPowerModule](#)

Interface: [IPowerModules](#)

Classifier	Name	Type	Description
-	RESULT	IPowerModule	

2.1.88.2 Count

```
property Count: Integer
```

This is a project specific feature.

Count is the number of items in the [Item](#) list.

Interface: [IPowerModules](#)  read-only

2.1.88.3 Item

```
property Item[Index: Integer]: IPowerModule
```

This is a project specific feature.

Item[I] is the power module at index I. I is in the range of 0...[Count](#)-1.

see also: [IPowerModule](#)

Interface: [IPowerModules](#)  read-only

2.1.88.4 Remove

```
procedure Remove(Ind: Integer);
```

This is a project specific feature.

Interface: IPowerModules			
Classifier	Name	Type	Description
	Ind	Integer	

2.1.89 | ProjectManager

gives you access to the *DEWESoft®* projectmanager.

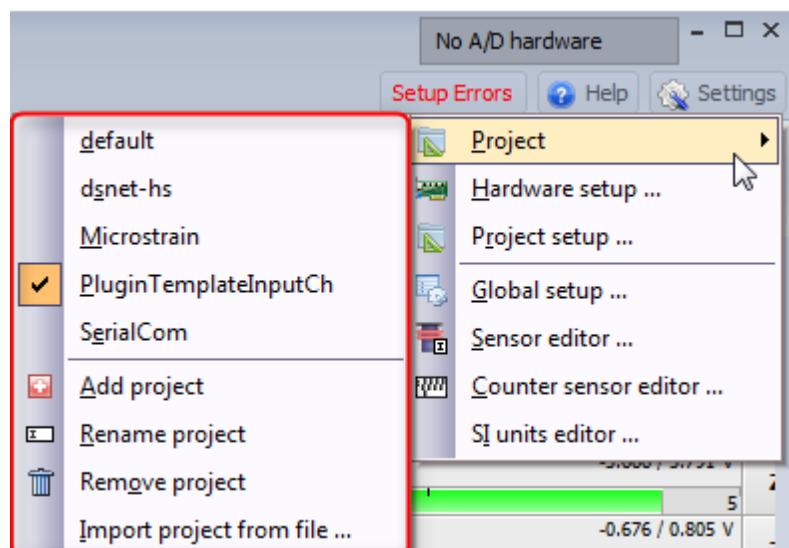


Illustration 181: Project Manager

2.1.89.1 ChangeProject

```
function ChangeProject(const ProjectName: WideString): WordBool;
```

will change to the given project

Interface: IProjectManager

Classifier	Name	Type	Description
const	ProjectName	WideString	the project name including the file extension: e.g. default.d7p
-	RESULT	WordBool	TRUE, if the project has been changed

2.1.89.2 GetCurrentProject

```
function GetCurrentProject(): WideString;
```

returns the name of the project that is currently active

Interface: [IProjectManager](#)

Classifier	Name	Type	Description
-	RESULT	WideString	the name of the project that is currently active

2.1.89.3 GetProjects

```
function GetProjects(): OleVariant;
```

returns an array of strings with the filenames of all project files

Interface: [IProjectManager](#)

Classifier	Name	Type	Description
-	RESULT	OleVariant	an array of strings with the filenames of all project files

2.1.90 IProperties

a list of properties - the elements may be of any data type.

i.e. used in plugins to pass arbitrary name-value pairs to visual controls

see also: [IInputGroup.Properties](#)

2.1.90.1 Add

```
procedure Add(const Name: WideString; Value: OleVariant);
```

add a new property with the given Name and Value.

Interface: [IProperties](#)

Classifier	Name	Type	Description
const	Name	WideString	Name of the property
	Value	OleVariant	Value of the property

2.1.90.2 Count

property Count: Integer

`Count` is the number of items in the `Item` list.

Interface: [IProperties](#) read-only

2.1.90.3 Item

property Item[Index: OleVariant]: OleVariant

Item[I] is the item at index I. I is in the range of 0...Count-1.

Interface: [IProperties](#)  read/write

2.1.91 |RegistrationHelper

internal use for plugins made by DEWEsoft®

2.1.91.1 CheckRegistration

```
procedure CheckRegistration();
```

internal use for plugins made by DEWESoft®

Interface: `IRegistrationHelper`

2192 | Screen

A screen can be e.g. an Overview, a Scope, a Recorder, etc.

see also: IScreens, GUI Navigation

2.1.92.1 GetCursor

```
function GetCursor(ACursor: Integer; AbsTime: WordBool): Double;
```

can be used to get the current time of the cursors.

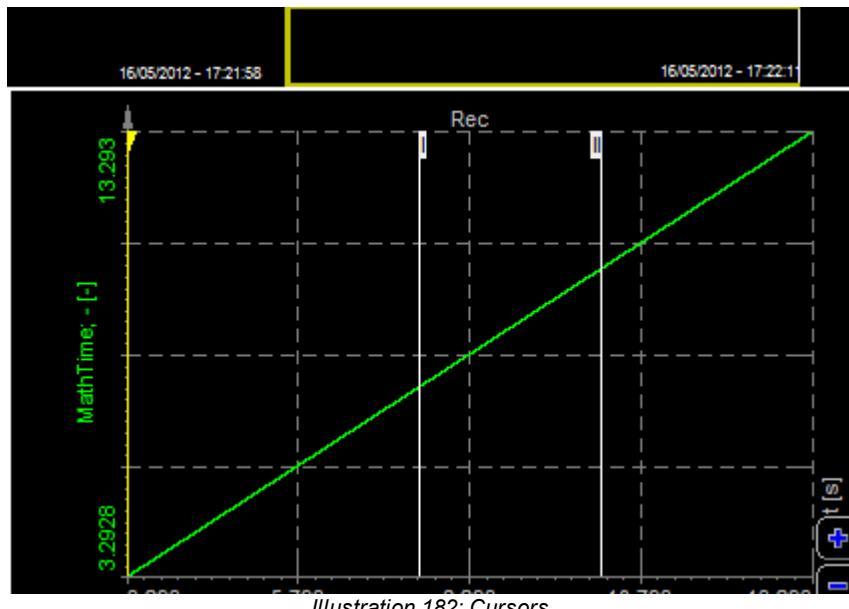


Illustration 182: Cursors

see also: [SetCursor](#)

Interface: [IScreen](#)

Classifier	Name	Type	Description
	ACursor	Integer	The ID of the cursor; i.e. in the image above the ID's are shown as Roman numbers at the top of the cursors: I, II)
	AbsTime	WordBool	If you want to get the result as a relative (<code>FALSE</code>) or absolute (<code>TRUE</code>) time.
-	<i>RESULT</i>	Double	the relative or absolute time where the cursor is currently positioned

2.1.92.2 Id

```
property Id: Integer
```

currently not implemented - always returns 0.

Interface: [IScreen](#) read-only

2.1.92.3 IsCurrent

```
property IsCurrent: WordBool
```

IsCurrent is True if the screen is currently shown.

Interface: [IScreen](#) read-only

2.1.92.4 Name

```
property Name: WideString
```

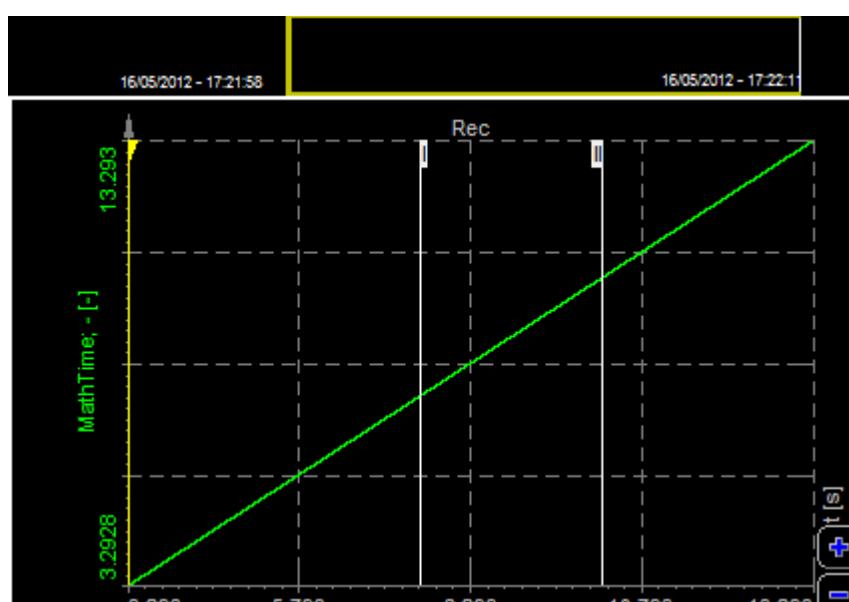
The name of the screen - can be changed by the user.

Interface: [IScreen](#) read-only

2.1.92.5 SetCursor

```
procedure SetCursor(ACursor: Integer; Time: Double; AbsTime: WordBool);
```

can be used to set the current time of the cursors; i.e. move the cursor to a specific time.



Interface: [IScreen](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier			Name			Type			Description		
			ACursor			Integer			The ID of the cursor; i.e. in the image above the ID's are shown as Roman numbers at the top of the cursors: I, II)		
			Time			Double			the relative or absolute time where the cursor is currently positioned		
			AbsTime			WordBool			If you want to get the result as a relative (FALSE) or absolute (TRUE) time.		

2.1.92.6 Show

```
procedure Show();
```

To show one specific screen. It will then be the current screen (see [IsCurrent](#), [IScreens.Current](#)) .

Interface: [IScreen](#)

2.1.92.7 ZoomIn

```
procedure ZoomIn();
```

To zoom into the data; same as pressing the + (plus) button for recorder visual controls.

see also: [ZoomOut](#)

Interface: [IScreen](#)

2.1.92.8 ZoomOut

```
procedure ZoomOut();
```

To zoom out of the data; same as pressing the - (minus) button for recorder visual controls.

see also: [ZoomIn](#)

Interface: [IScreen](#)

2.1.93 IScreens

A screen can be e.g. an *Overview*, a *Scope*, a *Recorder*, etc.

see also: [IApp.Screens](#), [IScreen](#)

2.1.93.1 Count

property Count: Integer

`Count` is the number of screens in the `Item` list.

Interface: [IScreens](#)  read-only

2.1.93.2 Current

property Current: IScreen

this is the current screen that is shown.

see also: [IScreen](#)

Interface: IScreens

2.1.93.3 Item

property Item[Index: Integer]: IScreen

Item[I] is the screen at index I. I is in the range of 0...Count-1.

see also: [IScreen](#)

Interface: [IScreens](#)

2.1.94 ISetupMessages

provides access to the channel setup messages

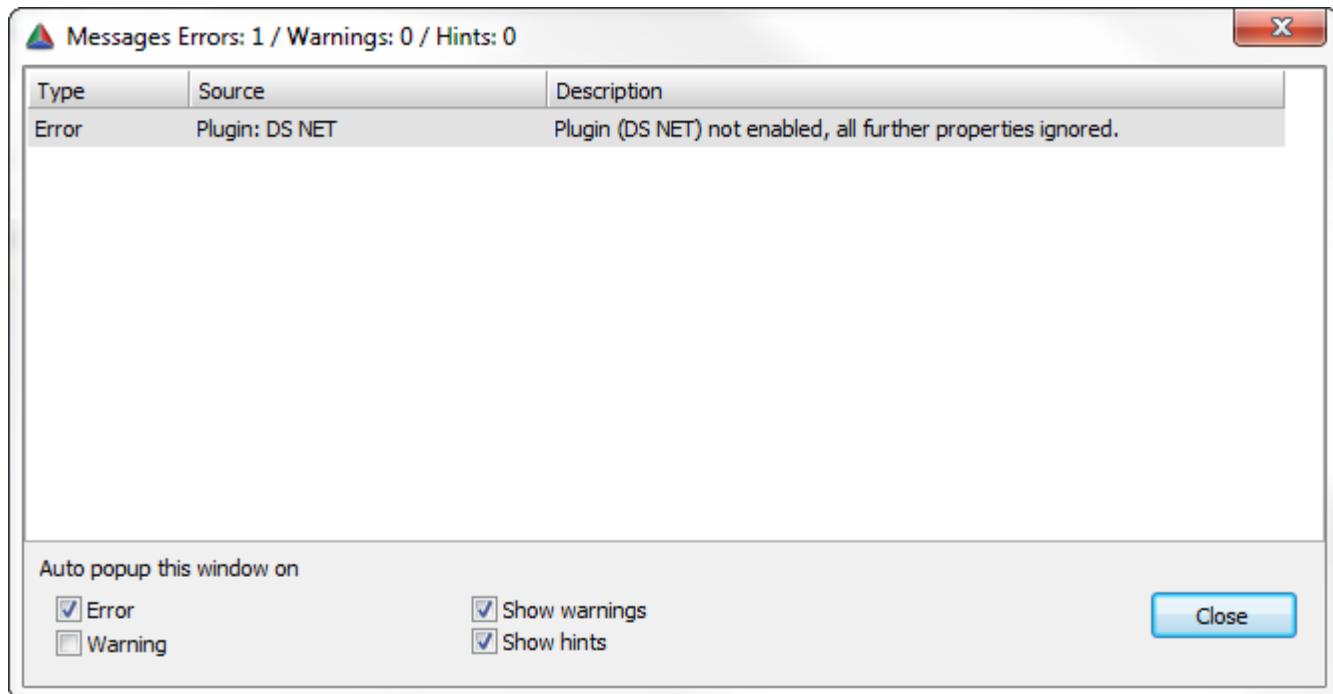


Illustration 184: Setup Messages

2.1.94.1 Add

```
procedure Add(MsgType: SetupMessageType  
; const Header: WideString; const Text: WideString);
```

can be used to add a setup message which will be shown to the user

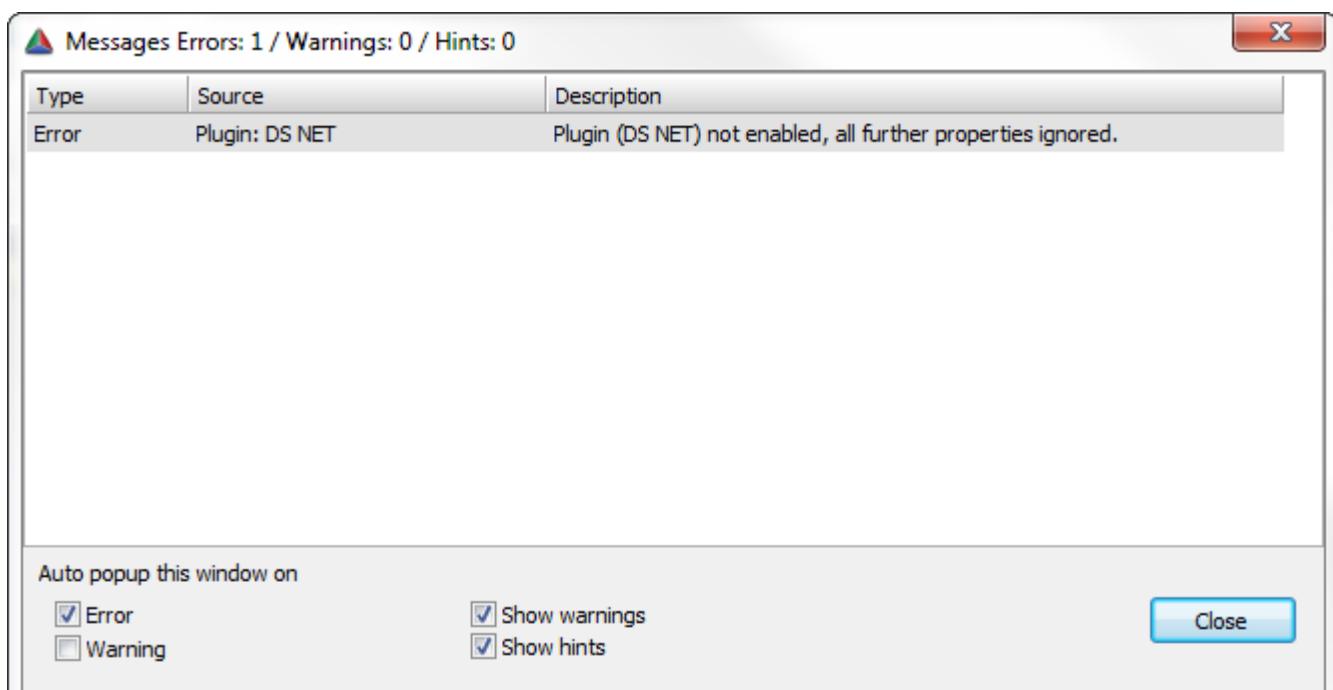


Illustration 185: Setup Messages

Interface: ISetupMessages

Classifier	Name	Type	Description
	MsgType	SetupMessageType	the message type
const	Header	WideString	the header of the message
const	Text	WideString	the message

2.1.95 IStoreEngine

provides access to the store engine which is responsible for storing the data file during measurement.

see also: [IApp.StoreEngine](#)

2.1.95.1 AddNewEvent

```
procedure AddNewEvent(Type : EventType; Data: OleVariant);
```

`AddNewEvent`, adds an event during storing of data.

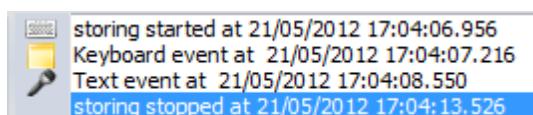


Illustration 186: Events

see also: [EventType](#)

Interface: [IStoreEngine](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description								
	Type_	EventType	defines which event to add: see EventType								
	Data	OleVariant	dependant on the Type : see EventType								

2.1.95.2 AllowIBSkipping

```
property AllowIBSkipping: WordBool
```

For internal use only! This feature must only be used by DEWEsoft® itself. Do not use in automation applications or add-ons.

Interface: [IStoreEngine](#)  read/write

2.1.95.3 FileName

```
property FileName: WideString
```

the name of the data file

Interface: [IStoreEngine](#)  read-only

2.1.95.4 FileSize

```
property FileSize: Largeuint
```

FileSize is the current size of the measurement file in bytes. When you start storing this number will be 0 and then increase during storing.

Interface: [IStoreEngine](#)  read-only

2.1.95.5 IsTriggering

```
property IsTriggering: WordBool
```

IsTriggering indicates whether the application is currently triggering. This corresponds to the state between a start- and a stop-trigger.

E.g., if the storing option is set to *fast on trigger*, IsTriggering will be *True* while data is stored, otherwise *False*.

Interface: [IStoreEngine](#)  read-only

2.1.95.6 Paused

property Paused: WordBool

Will be TRUE when the storing is currently paused (e.g. the user has clicked the pause button, or the triggering is not active).

Interface: [IStoreEngine](#)  read-only

2.1.95.7 StartStoreTimeChanged

```
procedure StartStoreTimeChanged();
```

For internal use only! This feature must only be used by DEWEsoft® itself. Do not use in automation applications or add-ons.

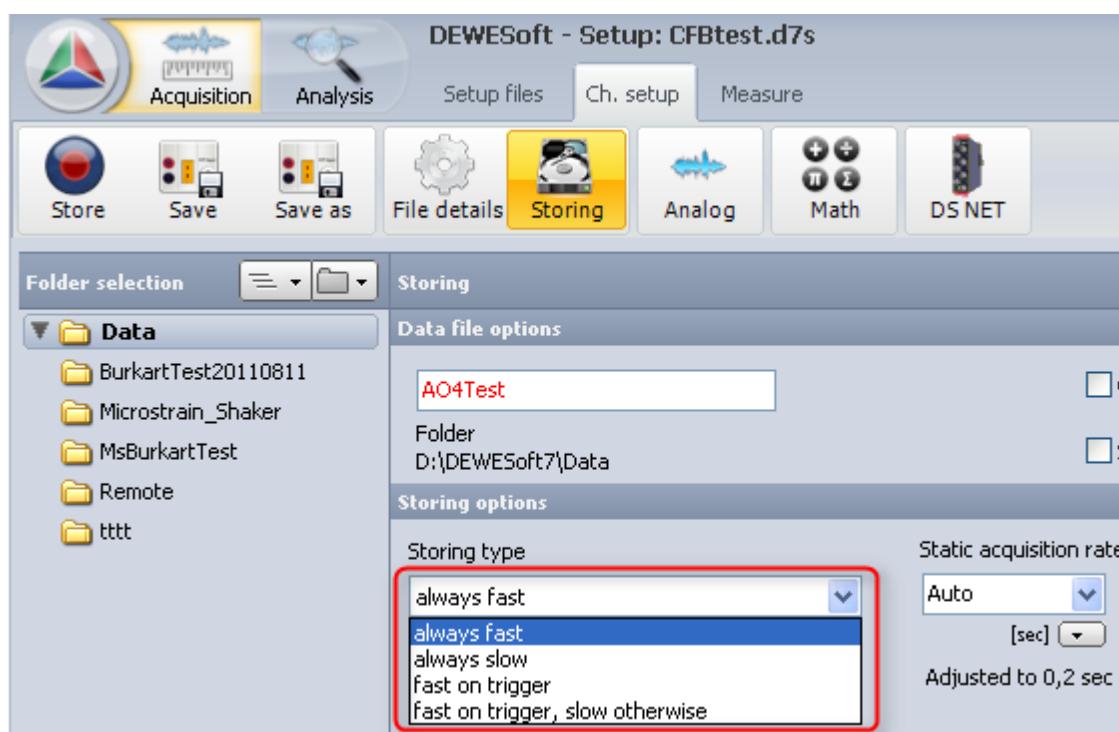
Interface: [IStoreEngine](#)

2.1.95.8 StoreMode

property StoreMode: Integer

`StoreMode` corresponds to the store options that can be set in the measurement setup (see illustration below):

Value	Mode	Description
0	always fast	will always store data with the fast dynamic acquisition rate (see IApp.MeasureSampleRate , IApp.MeasureSampleRateEx)
1	always slow	will always store reduced data with the slow reduced sample rate (see IApp.ReducedRate)
2	fast on trigger	will only acquire data with the fast dynamic acquisition rate (see IApp.MeasureSampleRate , IApp.MeasureSampleRateEx) when a specified trigger is active.
3	fast on trigger, slow otherwise	When the trigger is not active, no data will be stored. will acquire data with the fast dynamic acquisition rate (see IApp.MeasureSampleRate , IApp.MeasureSampleRateEx) when a specified trigger is active. When the trigger is not active, reduced data will be stored with the with the slow reduced sample rate (see IApp.ReducedRate)



see also: [Sample Rates](#)

Interface: [IStoreEngine](#) read-only

2.1.95.9 Storing

property Storing: WordBool

Whether storing is currently active or not.

Interface: [IStoreEngine](#) read-only

2.1.95.10 TrackingOffset

property TrackingOffset: Double

For internal use only! This feature must only be used by DEWESoft® itself. Do not use in automation applications or add-ons.

Interface: [IStoreEngine](#) read-only

2.1.96 ISyncSource

used for array-channels.

see also: [IArrayInfo.SyncSource](#)

2.1.96.1 IsSyncSource

property IsSyncSource: WordBool

`TRUE` if the sample-rate is fixed, `FALSE` otherwise.

Interface: [ISyncSource](#) read/write

2.1.96.2 SampleRate

property SampleRate: Double

only relevant if `IsSyncSource` is TRUE.

3.1.97 | Timing

Reserved for internal use! Ask our support if you need to know details: support@devsoft.com

see also: [App Timing](#)

21971 Tracking

property Tracking: WordBool

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: **ITiming** read-only

2198 | Tri

provides access to the trigger conditions

e.g. IAlarmCond_StopTrigger

2.1.98.1 GetTrigIndexEx

```
procedure GetTrigIndexEx(var CondIndex: Integer; var ChIndex: Integer; var Ch: IChannel);
```

returns the index of the trigger condition, the channel index and the channel.

see also: [IChannel](#)

Interface: [ITrig](#)

Classifier	Name	Type	Description
var	CondIndex	Integer	the index of the trigger condition
var	ChIndex	Integer	the channel index
var	Ch	IChannel	the channel

2.1.98.2 NotOrList

```
property NotOrList: ITriggerCondList
```

NotOrList provides a trigger condition list of the type [ITriggerCondList](#).

The trigger conditions of this list are equal to the *Don't store* conditions as found in DEWESoft's measurement setup:

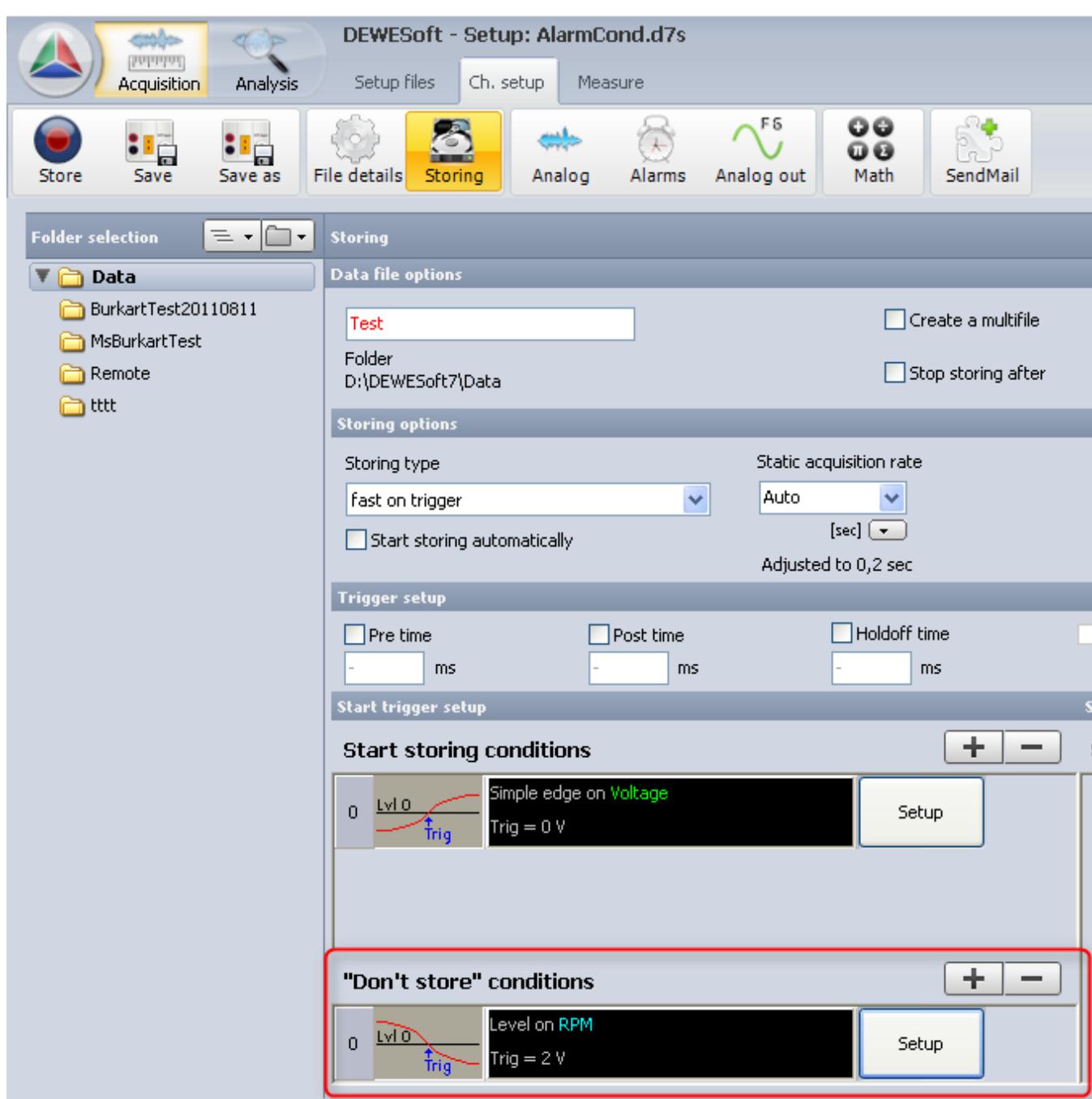


Illustration 187: Don't store list

compare to: [OrList](#)

see also: [ITriggerCondList](#)

Interface: [ITrig](#) read-only

2.1.98.3 OrList

property OrList: [ITriggerCondList](#)

`OrList` provides a trigger condition list of the type `ITriggerCondList`. The trigger conditions of this list are OR-combined.

The trigger conditions of this list are equal to the *storing* conditions as found in DEWEsoft's measurement setup:

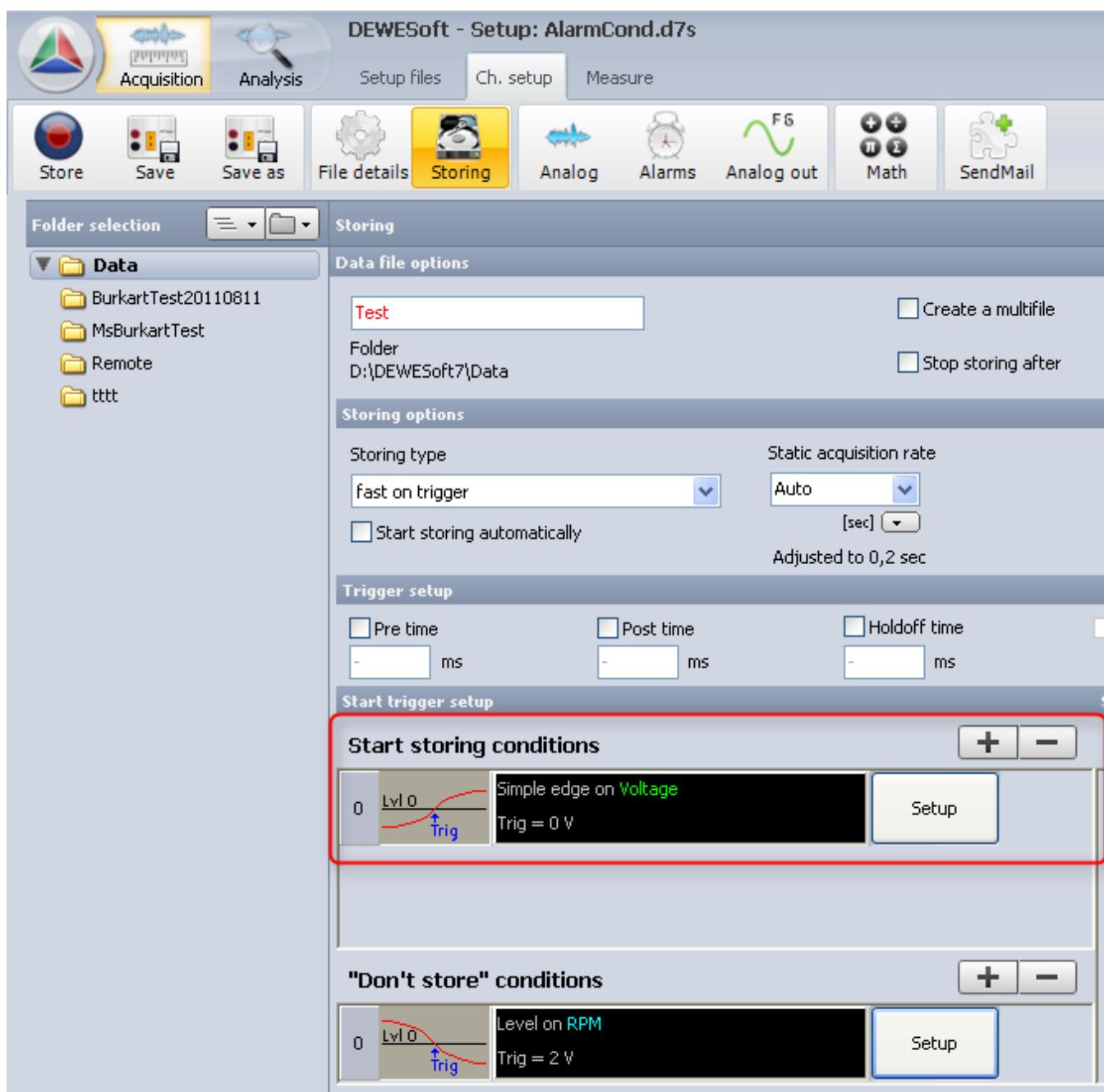


Illustration 188: Start storing conditions list

compare to: [NotOrList](#)

see also: [ITriggerCondList](#)

Interface: [ITrig](#) read-only

2.1.98.4 TrigIndex

property TrigIndex: Integer

The index of the condition in the [OrList](#) which caused the trigger.

Interface: [ITrig](#) read-only

2.1.99 ITrigInfo

used for trigger events; see also [IEvent.TrigInfo](#)

2.1.99.1 Channel

property Channel: IChannel

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)

2.1.99.2 DeltaTime

property DeltaTime: Double

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)

2.1.99.3 Direction

property Direction: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

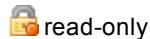
Interface: [ITrigInfo](#)

2.1.99.4 Direction1

property Direction1: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)



2.1.99.5 Level1

property Level1: Double

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)



2.1.99.6 Level2

property Level2: Double

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)



2.1.99.7 Manual

property Manual: WordBool

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)



2.1.99.8 Mode

property Mode: Integer

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITrigInfo](#)



2.1.99.9 TrigValue

property TrigValue: Integer

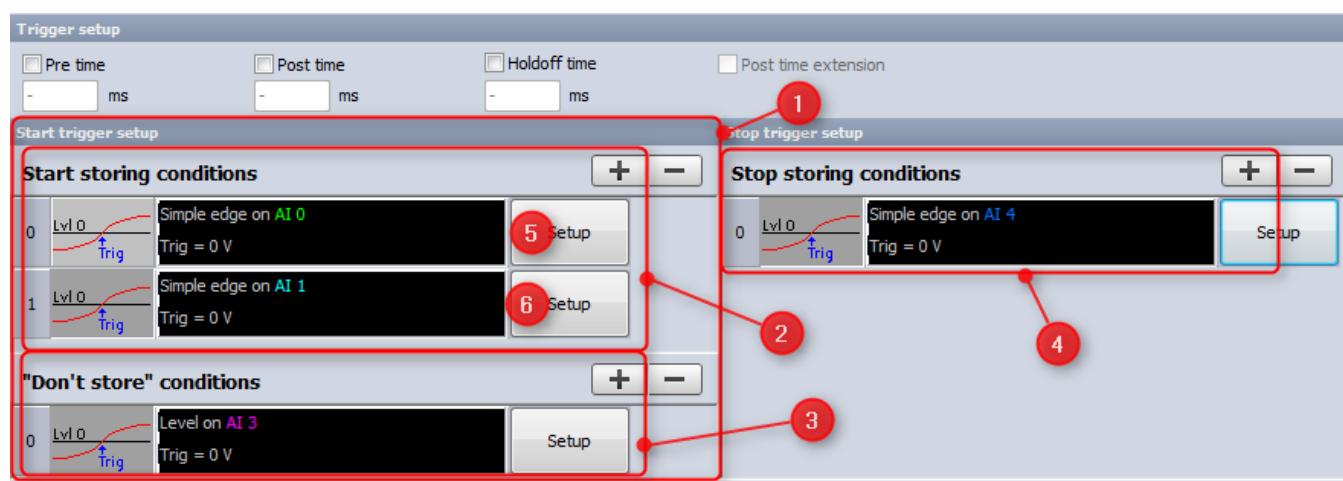
Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: ITrigInfo



2.1.100 ITrigger

Provides access to the store trigger conditions (see also [App.Trigger](#)).



- (1) [StartTrig](#) (of type [ITrig](#)) consists of
 - (2) *Start storing conditions:* [ITrig.OrList](#) (of type [ITriggerCondList](#)) which in turn consists of [ITriggerCondition](#)'s: (5) and (6)
 - (3) *Don't store conditions:* [ITrig.NotOrList](#) (of type [ITriggerCondList](#)) which in turn consists of [ITriggerCondition](#)'s
 - (4) [StopTrig](#) (of type [ITrig](#)) is a list of *Stop storing conditions* (of type [ITriggerCondList](#)) also consists of [ITriggerCondition](#)'s

2.1.100.1 HoldoffTime

property HoldoffTime: Single

The hold-off time in ms.

Gives you the possibility to suppress trigger events for a certain time after the last event had happened. This feature is not selected as a standard and will normally be used when you have plenty of events or very long storage times.

see also: [HoldoffTimeUsed](#)

Interface: [ITrigger](#) read/write

2.1.100.2 HoldoffTimeUsed

property HoldoffTimeUsed: WordBool

HoldoffTimeUsed defines whether the hold-off time ([HoldoffTime](#)) is used or not.

Interface: [ITrigger](#) read/write

2.1.100.3 PostTime

property PostTime: Single

Post trigger time, defined in milliseconds. This value defines the storage duration after the trigger event has been finished ? DEWEsoft will continue to store until we stop it manually or stop condition occurs.

see also: [PostTimeUsed](#)

Interface: [ITrigger](#) read/write

2.1.100.4 PostTimeExtensionUsed

property PostTimeExtensionUsed: WordBool

Whether the post time extension is used or not.

The post time extension is checked automatically as long as the Post time is not selected. The acquisition duration will be prolonged when further trigger events appear while the first one is still recorded.

Interface: [ITrigger](#) read/write

2.1.100.5 PostTimeUsed

property PostTimeUsed: WordBool

Whether the [PostTime](#) is used or not.

Interface: [ITrigger](#) read/write

2.1.100.6 PreTime

property PreTime: Single

The Pre trigger time, defined in milliseconds. This value defines the storage duration before the trigger event occurs. *DEWESoft®* will keep the data in the buffer until the trigger event occurs and then store also this data to the file.

As a standard, this feature is not selected and the storage starts with the trigger event itself.

see also: [PreTimeUsed](#)

Interface: [ITrigger](#) read/write

2.1.100.7 PreTimeUsed

property PreTimeUsed: WordBool

Whether the PreTime is used or not.

2.1.100.8 StartTrig

property StartTrig: ITrig

The start trigger property consists of *Start storing conditions*: [ITrig.OrList](#) (of type [ITriggerCondList](#)) and *Don't store conditions*: [ITrig.NotOrList](#) (of type [ITriggerCondList](#)) .

Interface: `ITrigger` read-only

2.1.100.9 StopTria

property StopTrig: ITrig

The stop trigger property consists of *Stop storing conditions*.

Interface: ITrigger read-only

2.1.101 ITriggerCondList

a list of [ITriggerCondition](#)'s - see also [ITrigger](#)

see also: [ITrigger.StartTrig](#), [ITrigger.StopTrig](#)

2.1.101.1 Add

```
function Add(): ITriggerCondition;
```

Adds a trigger condition of the type [ITriggerCondition](#) to the list of trigger conditions.

see also: [ITriggerCondition](#)

Interface: [ITriggerCondList](#)

Classifier	Name	Type	Description
-	RESULT	ITriggerCondition	the new trigger condition object.

2.1.101.2 Count

```
property Count: Integer
```

Count is the number of trigger conditions within the [Item](#) list.

Interface: [ITriggerCondList](#)



2.1.101.3 Item

```
property Item[Index: Integer]: ITriggerCondition
```

Item[I] is the input group at index I. I is in the range of 0...[Count](#)-1.

see also: [ITriggerCondition](#)

Interface: [ITriggerCondList](#)



2.1.101.4 Remove

```
procedure Remove(Ind: Integer);
```

Removes the trigger condition specified by the index Ind. Ind has to be within the range 0...[Count](#).-1.

Interface: [ITriggerCondList](#)

DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®	DEWEsoft®
Classifier	Name	Type	Description										
	Ind	Integer	The index of the trigger condition to remove. Ind has to be within the range 0 .. Count . -1.										

2.1.102 ITriggerCondition

a single trigger condition

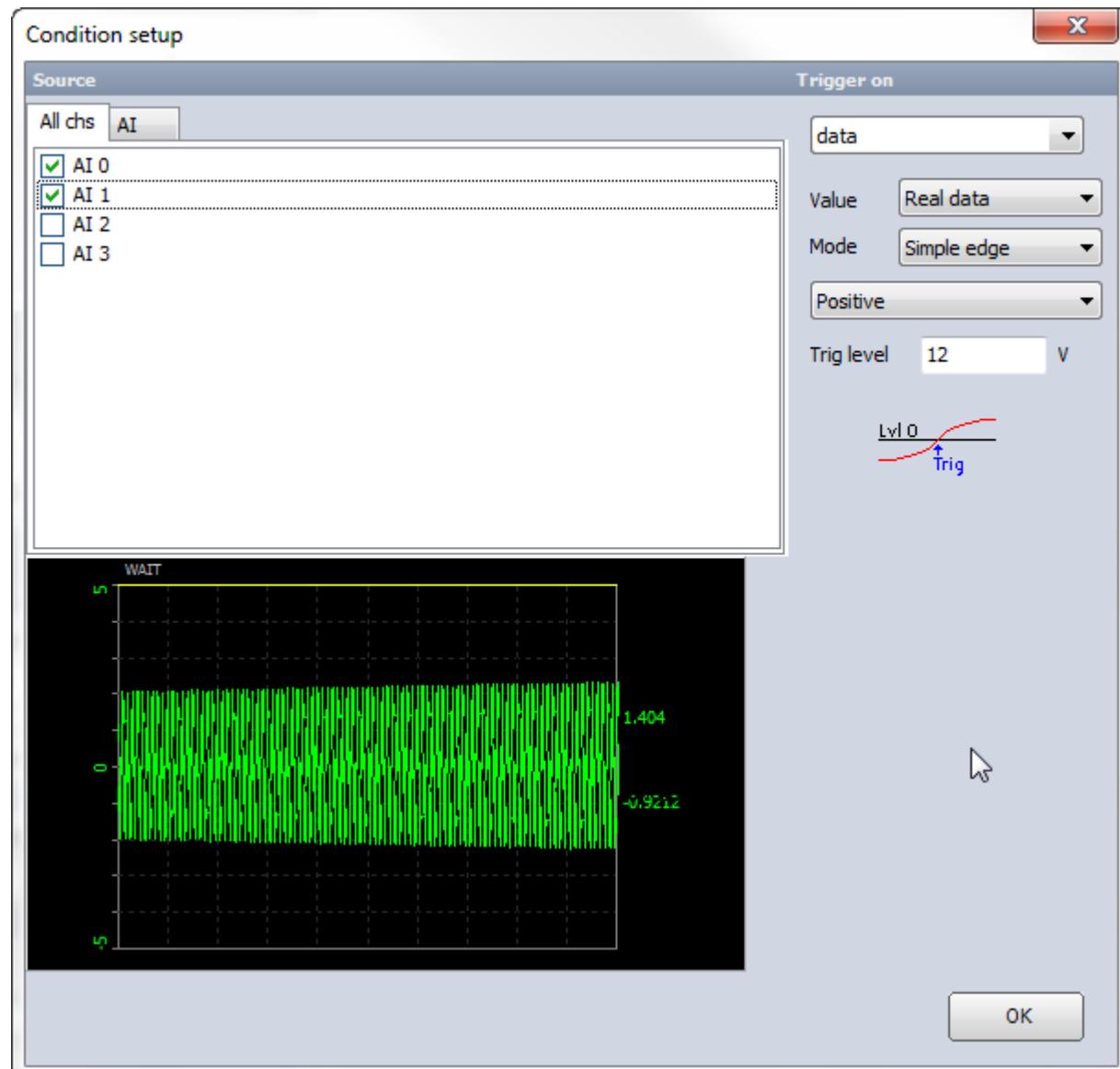


Illustration 190: Trigger condition

see also: [ITrigger](#), [ITriggerCondList](#), [IApp.Trigger](#)

2.1.102.1 AddChannel

```
procedure AddChannel(const Ch: IChannel);
```

will add a channel to the trigger condition.

see also: [IChannel](#), [Channels](#)

Interface: [ITriggerCondition](#)

Classifier	Name	Type	Description
const	Ch	IChannel	the channel that will be added to the trigger condition

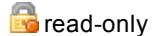
2.1.102.2 Channels

```
property Channels: IChannelList
```

a list of all channels for this trigger condition

see also: [IChannelList](#), [AddChannel](#)

Interface: [ITriggerCondition](#)



2.1.102.3 ClearChannels

```
procedure ClearChannels();
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [ITriggerCondition](#)

2.1.102.4 DeleteChannel

```
procedure DeleteChannel(Index: Integer);
```

removes a channel from this trigger condition: 0... [Channels.Count](#)-1

Interface: [ITriggerCondition](#)

Classifier	Name	Type	Description
	Index	Integer	the index of the channel to remove: 0... Channels.Count -1

2.1.102.5 DeltaTime

property DeltaTime: Double

only relevant if the [TrigType](#) is *on data* and the meaning of [DeltaTime](#) is dependant on the [Mode](#):

- for *Pulse Width* it is the *Time*
 - for *Window and pulselwidth* it is the *Time*
 - for *Slope* it is the *Delta time*

Interface: [ITriggerCondition](#)



read/write

2.1.102.6 Direction

property Direction: Integer

Direction defines the direction of a trigger condition - only relevant if the [TrigType](#) is *on data* and the meaning of Direction is dependant on the [Mode](#):

Simple edge, Filtered edge

Value	Description
0	positive
1	negative

Window

Value	Description
0	enter range
1	leave range

Pulse Width

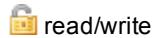
Value	Description
0	positive pulse
1	negative pulse

Window and Pulse Width

Value	Description
0	In range
1	Out of range

Slope, Delta Amplitude

Value	Description
0	positive
1	negative
2	any slope

Interface: [ITriggerCondition](#)

2.1.102.7 Direction1

property Direction1: Integer

is the second trigger level - only relevant if the [TrigType](#) is *on data* and the meaning of is dependant on the [Mode](#):

Pulse Width, Window and Pulse Width

Value	Description
0	longer than (see also DeltaTime)
1	shorter than (see also DeltaTime)

Slope

Value	Description
0	smoother than (see also DeltaTime)
1	steeper than (see also DeltaTime)

Interface: [ITriggerCondition](#)

2.1.102.8 Level1

property Level1: Single

`Level1` is the first trigger level - only relevant if the `TrigType` is *on data*.

The meaning of Level 1 is dependant on the [Mode](#):

- for *Simple Edge* and *Filtered Edge* it's the (one and only) *Trigger level*
 - for *Window* it is the *Upper level*
 - for *Pulse Width* it is the *Trigger level*
 - for *Window and pulselwidth* it is the *Upper level*
 - for *Slope* and *Delta amplitude* it is the *Delta level*

Interface: [ITriggerCondition](#)



read/write

2.1.102.9 Level2

property Level12: Single

Level2 is the second trigger level - only relevant if the [TrigType](#) is *on data* and the meaning of **Level2** is dependant on the [Mode](#):

- for *Filtered Edge* it's the *Rearm level*
 - for *Window* it is the *Lower level*
 - for *Window and pulselwidth* it is the *Lower level*
 - for *Slope* and *Delta amplitude* it is the *Delta level*

Interface: [ITriggerCondition](#)



read/write

2.1.102.10 Mode

property Mode: Integer

Mode defines the trigger mode of the trigger condition - only relevant if the [TrigType](#) is *on data*.

Value	Description
0	Simple edge
1	Filtered edge
2	Window
3	Pulse Width
4	Window and Pulse Width
5	Slope
6	Delta Amplitude

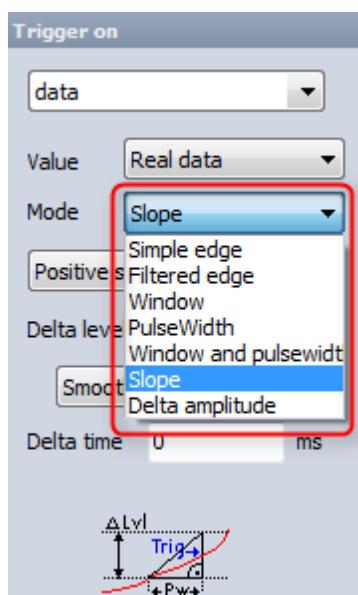


Illustration 191: Trigger Mode - Real Data

Note, that the available modes are dependant on the selected [TrigValue](#); i.e. for [TrigValue Average](#) only the following modes are available: *Simple edge*, *Filtered edge*, *Window*, *Slope*.

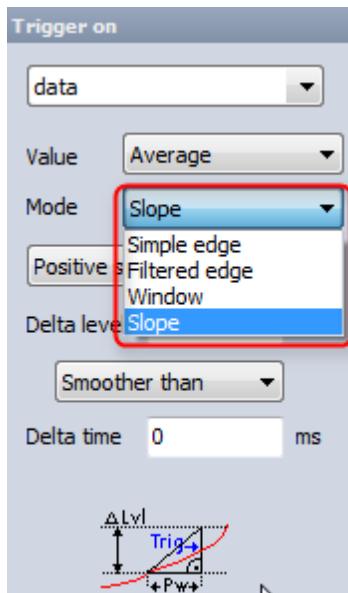


Illustration 192: Trigger mode

see also: [TrigType](#), [TrigValue](#)

Interface: [ITriggerCondition](#)



2.1.102.11 TimeCond

property TimeCond: Integer

The time condition for an *on time* trigger - only relevant if the [TrigType](#) is *on time*.

Value	Description
0	equal to
1	every

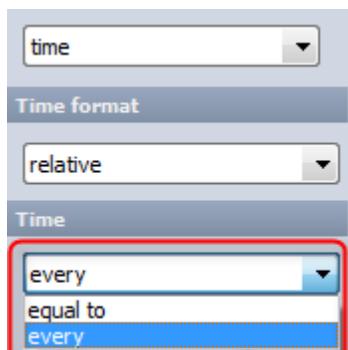


Illustration 193: Time condition

Interface: [ITriggerCondition](#)



2.1.102.12 TimeFormat

property TimeFormat: Integer

the time format for an *on time* trigger - only relevant if the [TrigType](#) is *on time*.

Value	Description
0	relative
1	absolute (time only)

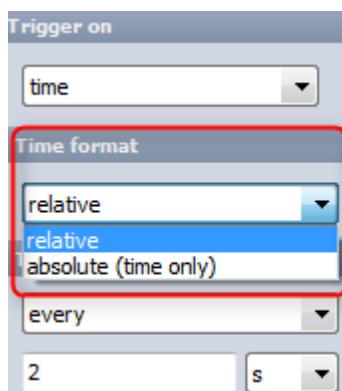


Illustration 194: Time format

Interface: [ITriggerCondition](#)

read/write

2.1.102.13 TimeUnit

property TimeUnit: Integer

The time unit for the [TimeValue](#) - only relevant if the [TrigType](#) is *on time*.

Value	Description
0	second
1	minute
2	hour

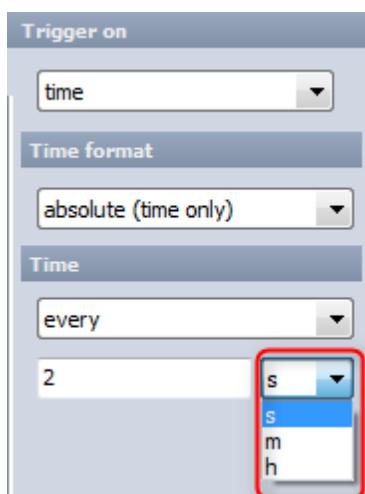


Illustration 195: Time unit

Interface: [ITriggerCondition](#)



2.1.102.14 TimeValue

```
property TimeValue: Double
```

The time for an *on time* trigger in the [TimeUnit](#) - only relevant if the [TrigType](#) is *on time*.

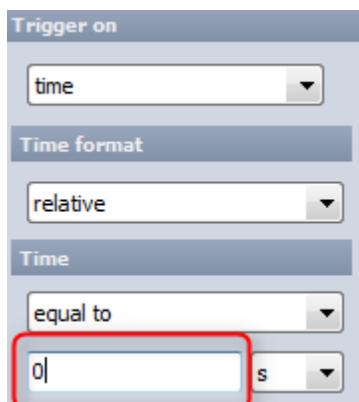


Illustration 196: Time Value

Interface: [ITriggerCondition](#)



2.1.102.15 TrigType

property TrigType: Integer

TrigType defines the type of a trigger which can be one of the following:

Value	Description
0	on data
1	on time see also: TimeCond , TimeFormat , TimeValue , TimeUnit
2	on FFT

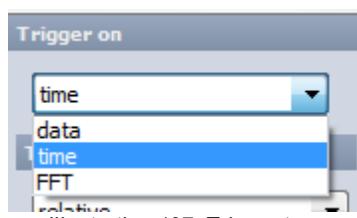


Illustration 197: Trigger type

Interface: [ITriggerCondition](#)

read/write

2.1.102.16 TrigValue

property TrigValue: Integer

TrigValue determines which kind of value is used for the trigger condition - only relevant if the [TrigType](#) is *on data*.

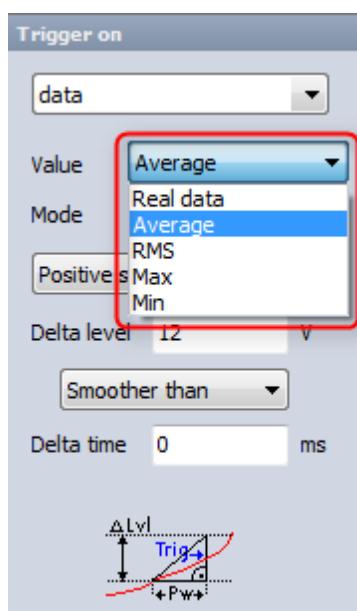


Illustration 198: Trigger value

Interface: [ITriggerCondition](#)



write

2.1.103 |UserInterface

this interface provides functions to the DEWESoft® user interface; i.e. to open dialogue windows via DCOM

2.1.103.1 ChangeSetupScreen

```
procedure ChangeSetupScreen(const ScreenName: WideString);
```

will change to the given setup screen by it's name:



Illustration 199: Setup screens

Interface: [IUserInterface](#)

Classifier	Name	Type	Description
const	ScreenName	WideString	the name of the setup screen: e.g. ' <i>Analog</i> ', ' <i>Math</i> ', .. note: the name must be used as it is displayed: e.g. when the user has selected another language then you must use those names: e.g. for German, <i>Counter</i> would be called ' <i>Zähler</i> '

2.1.103.2 ShowTrigCondSetup

```
procedure ShowTrigCondSetup(const Cond: ITriggerCondition);
```

Opens the condition setup window for the given trigger condition:

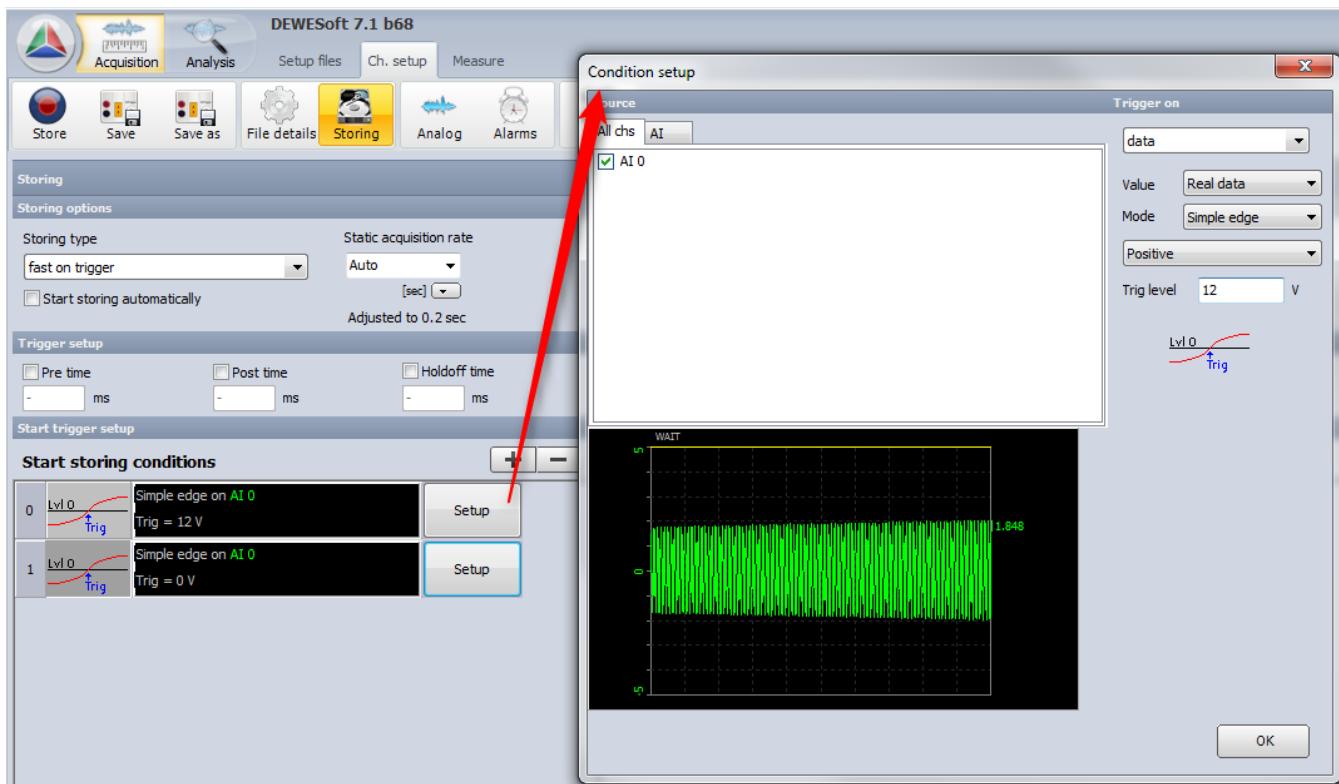


Illustration 200: Trigger Condition Setup

see also: [ITriggerCondition](#), [IApp.Trigger](#)

Interface: [IUserInterface](#)

Classifier	Name	Type	Description
const	Cond	ITriggerCondition	The trigger condition for which you want to open the setup window.

2.1.104 IVCCContext

context object for visual controls

2.1.104.1 BroadcastPosChanged

```
procedure BroadcastPosChanged();
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IVCContext](#)

2.1.104.2 BroadcastScaleChanged

```
procedure BroadcastScaleChanged(Min: T\_RecordPosition; Max: T\_RecordPosition;
; ZoomLevel: Integer; DeltaT: Double);
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

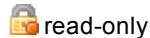
Interface: [IVCContext](#)

Classifier	Name	Type	Description
	Min	T_RecordPosition	
	Max	T_RecordPosition	
	ZoomLevel	Integer	
	DeltaT	Double	

2.1.104.3 DChannels

```
property DChannels: IChannelList
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [IChannelList](#)Interface: [IVCContext](#)

2.1.104.4 DIInputGroups

```
property DIInputGroups: IInputGroups
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [IInputGroups](#)Interface: [IVCContext](#)

2.1.104.5 DataRegionChanged

```
procedure DataRegionChanged();
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IVCContext](#)

2.1.104.6 Repaint

```
procedure Repaint();
```

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Interface: [IVCContext](#)

2.1.105 IVideo

this interface provides access to the video cameras

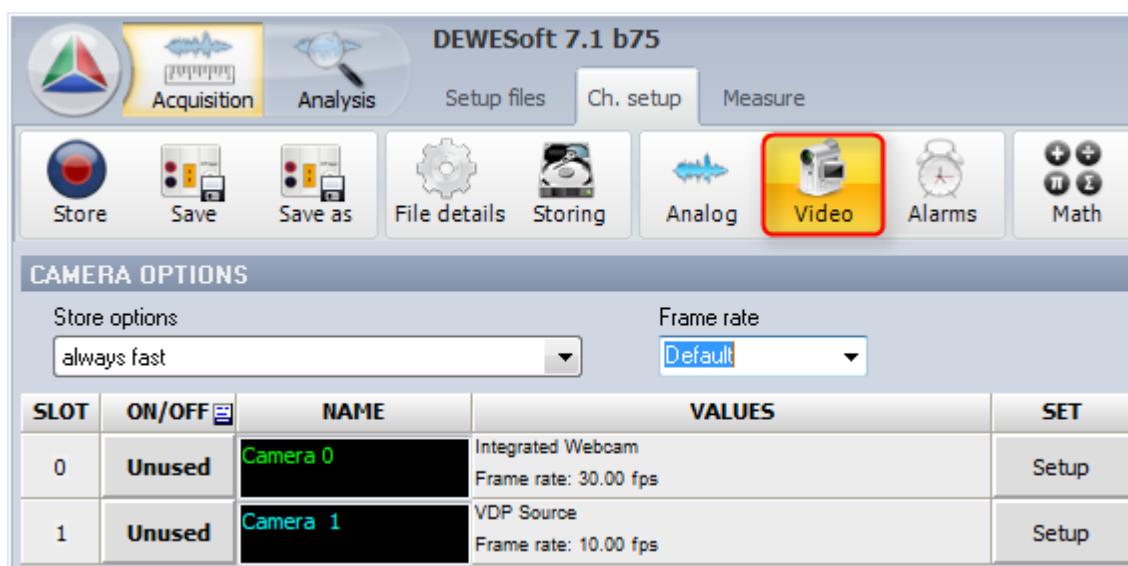


Illustration 201: IVideo

2.1.105.1 CameraCount

```
property CameraCount: Integer
```

the number of video cameras

see also: [Cameras](#)

Interface: [IVideo](#) read-only

2.1.105.2 Cameras

property Cameras[Index: Integer]: ICamera

returns the video camera at the given Index. Index is in the range of 0...CameraCount-1.

see also: [CameraCount](#)

Interface: [IVideo](#)  read-only

2.1.106 IVideoFrame

see also: [ICamera.FrameList](#)

2.1.106.1 BufSize

property BufSize: Integer

the size of the buffer in bytes

Interface: [IVideoFrame](#)

2.1.106.2 GetData

```
function GetData(): OleVariant;
```

the video data

Interface: [IVideoFrame](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	OleVariant	array of bytes

2.1.106.3 GetTS

```
function GetTS(): Double;
```

the timestamp of this frame

Interface: [IVideoFrame](#)

Classifier	Name	Type	Description
-	<i>RESULT</i>	Double	timestamp

2.1.107 IVideoloadEngine

this interface provides access to a single video load engine

see also: [IVideoloadEngines](#), [ILoadEngine](#), [VideoLoadEngines](#)

2.1.107.1 GetFramesInfo

```
procedure GetFramesInfo(out Frames: OleVariant);
```

Gives information about the timestamps of the video-frames of one video file.

Interface: [IVideoloadEngine](#)

Classifier	Name	Type	Description
out	Frames	OleVariant	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.1.108 IVideoloadEngines

provides access to the video load engines: [IVideoloadEngine](#)

see also: [ILoadEngine](#), [VideoLoadEngines](#)

2.1.108.1 Count

```
property Count: Integer
```

Count is the number of video load engines (of type [IVideoloadEngine](#)) in the [Item](#) list, which is equal to the number of videos captured.

Interface: [IVideoloadEngines](#)  read-only

2.1.108.2 Item

```
property Item[Index: Integer]: IVideoloadEngine
```

Item[I] is the video load engine at index I. I is in the range of 0...[Count](#)-1.

see also: [IVideoloadEngine](#)

Interface: [IVideoloadEngines](#)  read-only

2.2 Enumerations

A list of all enumerations that are used in the DEWEsoft® DCOM interface specification.

2.2.1 AOOperationMode

enumeration AOOperationMode

Describes the operation mode of the analogue output group: see [IAOGroup.OperationMode](#)

Dec	Hex	Name	Description
0	0x00	aomFixed	signals with a constant frequency: see IAOGroup.Freq
1	0x01	aomSweep	sweep frequency from a start frequency (see IAOGroup.StartFreq) to an end frequency (see IAOGroup.StopFreq)
2	0x02	aomStepSweep	sweep frequency with certain fixed frequencies
3	0x03	aomBurst	noise output
4	0x04	aomChirp	similar to <code>aomSweep</code> , but the signal is shorter in time and repeated after a defined time

2.2.2 AOSweepMode

enumeration AOSweepMode

see also [IAOGroup.SweepMode](#)

Dec	Hex	Name	Description
0	0x00	aosSingle	The signal will be output only once
1	0x01	aosLoop	The output signal will be repeated in a loop

2.2.3 AOWaveForm

enumeration AOWaveForm

Describes the wave for of the analogue output signal: see also [IAOChannel.WaveForm](#)

Dec	Hex	Name	Description
0	0x00	aowSine	a sine wave
1	0x01	aowTrian	a triangular signal
2	0x02	aowRect	a rectangular signal
3	0x03	aowSaw	a sawtooth signal
4	0x04	aowWhiteNoise	white noise signal: see also IAOChannel.FilterType
5	0x05	aowArbitrary	an arbitrary signal that you can define

2.2.4 ConnTypes

enumeration ConnTypes

see [IChannelConnection.AType](#)

2.2.5 ControlChFlags

enumeration ControlChFlags

flags for a control channel

see also: [Control Channels](#), [IChannel.ControlChannelFlags](#)

Dec	Hex	Name	Description
0	0x00	ctrlCh_NumType_Bool	boolean
1	0x01	ctrlCh_NumType_Int	integer
2	0x02	ctrlCh_NumType_Float	floating point
7	0x07	ctrlCh_NumType_Mask	
8	0x08	ctrlCh_AllowAlarmDO	if the control channel can be used as digital output for an alarm condition

enumeration CustomCANMessages

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	Description
0	0x00	canGetDeviceType	
1	0x01	canStartAcq	
2	0x02	canStopAcq	
3	0x03	canPrepareAcq	
4	0x04	canGetData	
5	0x05	canGetCardCount	
6	0x06	canGetCardCodes	
7	0x07	canportStartAcq	
8	0x08	canportStopAcq	
9	0x09	canportPrepareAcq	
10	0x0A	canportGetData	
11	0x0B	canportClearBuffers	
12	0x0C	canportGetBaudRates	
13	0x0D	canInit	
14	0x0E	canShowFrame	
15	0x0F	canHideFrame	
16	0x10	canportEnableOutput	
17	0x11	canportGetOutputEnabled	
18	0x12	canportWriteFrame	
19	0x13	canSupportsOutput	
20	0x14	canportCaptureSet	
21	0x15	canportCaptureStartRead	
22	0x16	canportCaptureEndRead	
23	0x17	canportCaptureGetMessageCount	
24	0x18	canportCaptureReadMessage	

2.2.7 CustomDAQMessages

enumeration CustomDAQMessages

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	Description
0	0x00	daqGetDeviceType	
1	0x01	daqUpdateXMLAISetup	
2	0x02	daqAmplMeasurementsCount	
3	0x03	daqAmplGetMeasurement	
4	0x04	daqAmplSetMeasurementCode	
50000	0xC350	daqSetNamePrefix	

2.2.8 CustomExpEventIDs

enumeration CustomExpEventIDs

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [ICustomExport3.OnEvent](#)

Dec	Hex	Name	InParam	OutParam	Description
0	0x00	evStartDataFold erEx			

2.2.9 CustomImpMessages

enumeration CustomImpMessages

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	InParam	OutParam	Description
0	0x00	impVersionCheck			
1	0x01	impInit			
2	0x02	impDeinit			
3	0x03	impOpenFile			
4	0x04	impCloseFile			
5	0x05	impShowFrame			
6	0x06	impHideFrame			
7	0x07	impPrepareImport			
8	0x08	impClearImport			
9	0x09	impStartImport			
10	0x0A	impEndImport			
11	0x0B	impImport			
12	0x0C	impGetBlockSize			
13	0x0D	impGetSampleRate			
14	0x0E	impGetBlockCount			
15	0x0F	impGetAbsStartTime			
16	0x10	impGetImportType			
17	0x11	impResizeFrame			

2.2.10 CustomMathFrameMessages

enumeration CustomMathFrameMessages

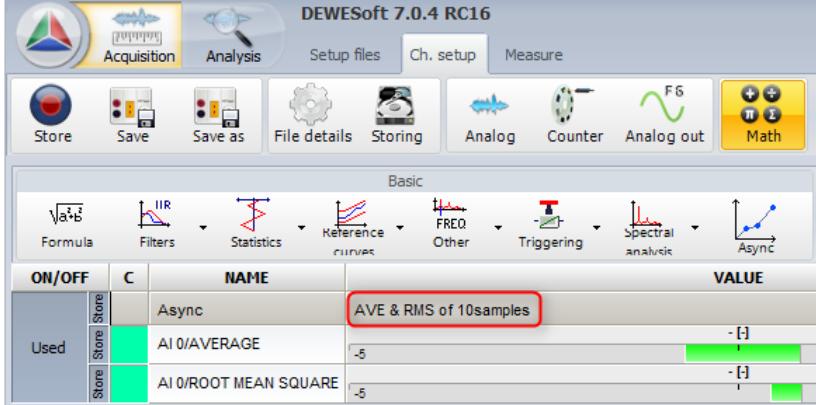
Messages that will be passed to the frame of math plugins: see also [IDewePlugin.OnMessage](#)

De c	He x	Name	InParam	OutParam	Description
0	0x00	mfrVersionCheck	the current version of the math interface type: Integer e.g. 1	the expected version of the math interface type: Integer e.g. 1	used for checking if the expected and actual version of the math interface are the same e.g. if the plugin returns 2 and the DEWESoft® instance has version 1, the math plugin is incompatible
1	0x01	mfrShowFrame	handle of the parent window of type LongWord (see also Data Types)	-	on this message, the plugin should create the frame and make it visible
2	0x02	mfrHideFrame	-	-	on this message the plugin should hide the frame that has been shown (on the mfrShowFrame message) and discard the parent window handle (that has been set in the mfrShowFrame message)
3	0x03	mfrResizeFrame	is an array: [Index] name: type [0] Width: Integer [0] Height: Integer	-	this message will be sent when the parent window is resized
4	0x04	mfrSetFrame	the same OleVariant that is set by IMathFrameContext.Apply	-	on this message, the plugin frame should set its edit fields, etc. according to the configuration properties
5	0x05	mfrInitFrame	handle to the DEWESoft® application interface: IApp	-	called to initialize the math plugin frame
6	0x06	mfrSetContext	a reference to the math frame context (see: IMathFrameContext)	-	called to pass a reference to the math frame context (see: IMathFrameContext) to the plugin

enumeration CustomMathMessages

Messages that will be passed to math plugins: see also [IDewePlugin.OnMessage](#), [IMathContext](#)

De c	He x	Name	InParam	OutParam	Description
0	0x00	mthVersionCheck	the current version of the math interface type: Integer e.g. 1	the expected version of the math interface type: Integer e.g. 1	used for checking if the expected and actual version of the math interface are the same e.g. if the plugin returns 2 and the DEWEsoft® instance has version 1, the math plugin is incompatible
1	0x01	mthInit	is an array: [Index] name: type [0] App: IApp [1] Ctx: IMathContext	-	called to initialize the math plugin
2	0x02	mthInitiate	-	-	called to initiate the math plugin (called after mthInit)
3	0x03	mthCalculate	-	-	when this message is received the math plugin should read data from the input channels (see IMathContext.InputChannels), do some calculation and write the results to the output channels (see IMathContext.OutputChannels)
4	0x04	mthReset	-	-	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
5	0x05	mthMountChannels	-	-	when this message is received, the math plugin can mount it's channels: IMathContext.MountChannel , IMathContext.MountChannelEx
6	0x06	mthGetProps	-	an OleVariant object (name: Props)	the math plugin can write arbitrary data into the OutParam (e.g. to store some properties that the plugins needs) see also mthSetProps
7	0x07	mthSetProps	an ole variant object (name: Props)	-	will be called to pass properties to the math plugin see also mthGetProps
8	0x08	mthGetDescriptor	-	an arbitrary text to	the math plugin should return an arbitrary text to describe itself (data type: String)

De c	He x	Name	InParam	OutParam	Description																
		ion		describe the math plugin (data type: String)	 <table border="1"> <thead> <tr> <th>ON/OFF</th> <th>C</th> <th>NAME</th> <th>VALUE</th> </tr> </thead> <tbody> <tr> <td>Used</td> <td>Store</td> <td>Async</td> <td>AVE & RMS of 10samples</td> </tr> <tr> <td>Used</td> <td>Store</td> <td>AI 0/AVERAGE</td> <td>-5</td> </tr> <tr> <td>Used</td> <td>Store</td> <td>AI 0/ROOT MEAN SQUARE</td> <td>-5</td> </tr> </tbody> </table>	ON/OFF	C	NAME	VALUE	Used	Store	Async	AVE & RMS of 10samples	Used	Store	AI 0/AVERAGE	-5	Used	Store	AI 0/ROOT MEAN SQUARE	-5
ON/OFF	C	NAME	VALUE																		
Used	Store	Async	AVE & RMS of 10samples																		
Used	Store	AI 0/AVERAGE	-5																		
Used	Store	AI 0/ROOT MEAN SQUARE	-5																		
9	0x09	mthSampleRateChanged	-	-	will be called to inform the math plugin that the sample rate has been changed																
10	0x0A	mthAcceptInputChannel	is an array: [Index] name: type [0] InputIndex: Integer [1] Ch: <u>IChanne</u> <u>l</u>	if the plugin can handle this input channel or not. TRUE...the plugin accepts the input channel and it will be included in <u>IMathContext</u> . <u>InputChann</u> els. FALSE... the plugin does not accept the input channel	allows the plugin to select which input channels it supports (e.g. plugins may support only synchronous channels)																
11	0x0B	mthGetOutputChDescriptio	is an array: [Index] name: type [0] Ch: <u>IChanne</u> <u>l</u>	String	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com																
12	0x0C	mthGetInputChannels	-	array of String	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com																
13	0x0D	mthUpdateXML	is an array: [Index]	-	called when the channel setup is written/read note: currently the function is only called when the setup is read																

Dec	Hex	Name	InParam	OutParam	Description
			name: type [0] DOMDoc: TXMLSet up [1] DOMNode : IDOMNod e [2] Write: Boolean if the setup is being written (TRUE) or read (FALSE)		
14	0x0E	mthPreInitiate	-	-	called before mthInitiate
15	0x0F	mthGetInputGroups	is an array: [Index] name: type [0] InputGr oups: <u>IInputG roups</u> [1] AllCh: Boolean	-	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
16	0x10	mthClearCalc	-	-	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
17	0x11	mthStopCalc	-	-	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
18	0x12	mthDeinit	-	-	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
19	0x13	mthGetDefaultChannelName	Channel : <u>IChanne l</u>	DefaultC hannelNa me: String	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
20	0x14	mthGetDefaultChannelCol	Channel : <u>IChanne l</u>	DefaultCh annelCol or:	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

De c	He x	Name	InParam	OutParam	Description
		olor	1	TColor	
21	0x15	mthGetR ecalcOn PosChan ged	-	Boolean	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
22	0x16	mthGetR egistra tion	-	Integer	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
23	0x17	mthNeed sMultip ass	-	MathMult ipassTyp e	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
24	0x18	mthPass Started	-	MathMult ipassTyp e	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
25	0x19	mthPass Comple ted	-	MathMult ipassTyp e	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.2.12 CustomVCMessages

enumeration CustomVCMessages

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	Description
0	0x00	vcVersionCheck	
1	0x01	vcDrawData	
2	0x02	vcInit	
3	0x03	vcAcceptChannel	
4	0x04	vcSave	
5	0x05	vcLoad	
6	0x06	vcGetDataRegion	
7	0x07	vcMouseWheel	
8	0x08	vcMouseWheelUp	
9	0x09	vcMouseWheelDown	
10	0x0A	vcMouseUp	
11	0x0B	vcMouseDown	
12	0x0C	vcMouseMove	
13	0x0D	vcMouseClicked	
14	0x0E	vcMouseDblClick	
15	0x0F	vcKeyDown	
16	0x10	vcKeyUp	
17	0x11	vcKeyChar	
18	0x12	vcStartAcq	
19	0x13	vcShowFrame	
20	0x14	vcHideFrame	
21	0x15	vcInitFrame	
22	0x16	vcResizeFrame	
23	0x17	vcResize	
24	0x18	vcAcceptInputGroup	
25	0x19	vcMouseEnter	
26	0x1A	vcMouseLeave	
27	0x1B	vcInitScreen	
28	0x1C	vcAddChannel	
29	0x1D	vcAddInputGroup	

2.2.13 EventIDs

enumeration EventIDs

see [IPlugin4.OnEvent](#) for details

2.2.14 EventReason

enumeration EventReason

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	Description
0	0x00	erGetData	
1	0x01	erStartStoring	
2	0x02	erStopStoring	
3	0x03	erTrigger	
4	0x04	erException	
5	0x05	erTriggerStop	
6	0x06	erAlarm	
7	0x07	erExit	
8	0x08	erDataLost	
9	0x09	erMessageBox	
10	0x0A	erProgress	

2.2.15 EventType

enumeration EventType

defines a DEWEsoft® event type.

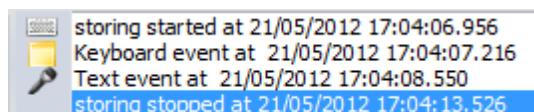


Illustration 202: Events

see also: [IStoreEngine.AddNewEvent](#), [ICustomExport2.WriteEvent](#), [IEvent](#)

Dec	Hex	Name	Description
1	0x01	etStart	when storing is started
2	0x02	etStop	when storing is stopped
3	0x03	etTrigger	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
11	0x0B	etVStart	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
12	0x0C	etVStop	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
20	0x14	etKeyboard	keyboard event (user clicks <i>SPACE</i> key)
21	0x15	etText	text event: user adds a text notice
22	0x16	etVoice	user recorded a sound message
23	0x17	etPicture	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com
24	0x18	etModule	<i>Reserved for internal use!</i> Ask our support if you need to know details: support@dewesoft.com

2.2.16 ExportTypes

enumeration ExportTypes

specifies the export type for custom data export add-ons.

see also: [ICustomExport.Get_ExportType](#)

Dec	Hex	Name	Description
0	0x00	etValueBased	the first value of each channel is exported before the second value of each channel and so on
1	0x01	etChannelBased	all values of the first channel are exported, then all values of the second channel and so on

2.2.17 GHObjectType

enumeration GHObjectType

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [IApp.GlobalHeader](#), [IGHObject.ObjType](#)

Dec	Hex	Name	Description
1	0x01	gtInput	
2	0x02	gtSelection	
3	0x03	gtMemo	
4	0x04	gtDir	
5	0x05	gtFileName	
6	0x06	gtDivision	
7	0x07	gtSound	

2.2.18 ImportStatus

enumeration ImportStatus

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	Description
0	0x00	impsFinished	
1	0x01	impsData	
2	0x02	impsGap	

2.2.19 ImportTypes

enumeration ImportTypes

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

Dec	Hex	Name	Description
0	0x00	itValueBased	
1	0x01	itChannelBased	

2.2.20 MathMultipassType

enumeration MathMultipassType

see also: mthNeedsMultipass, mthPassStarted, mthPassCompleted in [CustomMathMessages](#)

Dec	Hex	Name	Description
0	0x00	mptNone	
1	0x01	mptForward	
2	0x02	mptBackward	

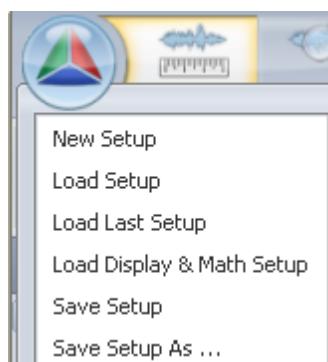
2.2.21 **MenuItems**

enumeration MenuItems

see: [IApp.MenuClick](#)

see also: [LoadSetup](#), [SaveSetup](#)

Dec	Hex	Name	Description
0	0x00	ItemSaveSetup	refers to the Save Setup menu item: see image below
1	0x01	ItemSaveSetupAs	refers to the Save Setup As... menu item: see image below
2	0x02	ItemLoadSetup	refers to the Load Setup menu item: see image below



2.2.22 ModulesFunction

enumeration ModulesFunction

DEPRECATED! This feature is deprecated and should not be used in DEWEsoft® version 7.0 and higher.

see [ExecuteModulesFunction](#) for details

2.2.23 SetupMessageType

enumeration SetupMessageType

see also: [ISetupMessages.Add](#)

Dec	Hex	Name	Description
0	0x00	smsgHint	a hint message
1	0x01	smsgWarning	a warning message
2	0x02	smsgError	an error message

2.2.24 SpecDirectory

enumeration SpecDirectory

enumeration of DEWEsoft® directories.

The exact path to the directories may be different dependant on the operating system and language, your installation type/settings, and your DEWEsoft® settings (e.g. current project settings).

see also [IApp.GetSpecDir](#)

Dec	Hex	Name	Description
0	0x00	sdSystemDir	returns the DEWEsoft® system directory: e.g. D:\\DEWEsoft7\\System\\V7_0
1	0x01	sdLogDir	returns the DEWEsoft® logging directory: e.g. D:\\DEWEsoft7\\System\\V7_0\\Logs see also: IApp.WriteErrorLog
2	0x02	sdScriptsDir	returns the DEWEsoft® scripts directory: e.g. D:\\DEWEsoft7\\System\\V7_0\\Scripts
3	0x03	sdTempDir	returns the DEWEsoft® temporary directory: e.g. D:\\DEWEsoft7\\System\\V7_0\\Temp
4	0x04	sdDataDir	returns the DEWEsoft® data directory: e.g. D:\\DEWEsoft7\\Data\\ see also: IApp.MainDataDir , IApp.UsedDatafile
5	0x05	sdSetupDataDir	returns the DEWEsoft® setup directory: e.g. D:\\DEWEsoft7\\Setups\\ see also: IApp.UsedSetupfile
6	0x06	sdExportDataDir	returns the DEWEsoft® export directory: e.g. D:\\DEWEsoft7\\Exports\\
7	0x07	sdIniDir	returns the DEWEsoft® ini-file directory: e.g. D:\\DEWEsoft7\\System\\V7_0 this is always the same as the sdSystemDir see also: IApp.IniFileDir
8	0x08	sdAddonsDir	returns the DEWEsoft® addons directory: e.g. D:\\DEWEsoft7\\Bin\\V7_0\\Addons\\ this is always relative to sdAppDir
9	0x09	sdAppDir	returns the DEWEsoft® binary directory: e.g. D:\\DEWEsoft7\\Bin\\V7_0\\ this is the directory from which Dewesoft.exe has been started from

2.2.25 TAixsType

enumeration TAixsType

describes the type of an axis

see also: [IAxisDef.AxisType](#)

Dec	Hex	Name	Description
0	0x00	atString	defined with a table of strings StringValues
1	0x01	atFloat	defined with a table of values FloatValues
2	0x02	atFloatLinearFunc	defined by StartValue , StepValue

2.2.26 TSRDivType

enumeration TSRDivType

the type of the sample rate divider describes how the skipped samples should be handled.



see also: [IChannel.SRDivType](#), [IChannel.SetSRDivType](#), [IChannel.SRDiv](#), [IChannel.SetSRDiv](#)

Dec	Hex	Name	Description
0	0x00	sdSkip	the samples are skipped
1	0x01	sdAverage	an average of the samples is used
2	0x02	sdFilter4th	a filter of 4th order is applied to the samples
3	0x03	sdFilter6th	a filter of 6th order is applied to the samples
4	0x04	sdFilter8th	a filter of 8th order is applied to the samples

2.2.27 XMLType

enumeration XMLType

Reserved for internal use! Ask our support if you need to know details: support@dewesoft.com

see also: [IPowerModule.LoadFromXML](#), [IPowerModule.LoadFromXML1](#), [IPowerModule.SaveToXML](#), [IPowerModule.SaveToXML1](#)

Dec	Hex	Name	Description
0	0x00	xmlFile	
1	0x01	xmlString	
2	0x02	xmlMSXML	

2.3 Types

A list of all types that are used in the DEWEsoft® DCOM interface specification.

2.3.1 DaqDeviceInfo

record DaqDeviceInfo

see: [IDaq.GetDeviceInfo](#)

Name	Type	Description
AICount	WideString	the number of analog input channels: e.g. the DEWE-43 has 8
AIUsedCount	WideString	the number of used analog input channels
CNTCount	WideString	number of counters
CNTUsedCount	WideString	number of used counters
CalDate	WideString	calibration data (e.g. 26/01/2012)
DICount	WideString	number of digital inputs
DIUsedCount	WideString	number of used digital inputs
DriverVersion	WideString	the version of the driver
Firmware	WideString	the firmware version: e.g. 5.5.4.16
FirmwareSub	WideString	sub firmware version
Name	WideString	name of the device: e.g. DEWE-43
OptionCount	WideString	number of options
OptionTypes	WideString	option types
SerialNumber	WideString	the serial number of the device: e.g. D0201A04
Used	WideString	if the device is used or not: e.g. YES

2.3.2 ITestRecord

record ITestRecord

For internal use only! This feature must only be used by *DEWESoft®* itself. Do not use in automation applications or add-ons.

Name	Type	Description
Field1	Integer	
Field2	Integer	

2.3.3 T CANFrame

record T CANframe

This record contains the bytes of the CAN message

see also: [ICANPort.SendFrame](#)

Name	Type	Description
Byte0	Byte	byte 0 of the CAN message
Byte1	Byte	byte 1 of the CAN message
Byte2	Byte	byte 2 of the CAN message
Byte3	Byte	byte 3 of the CAN message
Byte4	Byte	byte 4 of the CAN message
Byte5	Byte	byte 5 of the CAN message
Byte6	Byte	byte 6 of the CAN message
Byte7	Byte	byte 7 of the CAN message

2.3.4 T ChIndex

record T ChIndex

DEPRECATED! This feature is deprecated and should not be used in DEWEISoft® version 7.0 and higher.

USE `IChannel` `Get_IndexEx` INSTEAD - see also `Channel_Index`

`T_CbIndex` is a record of indexes containing information about a channel

see also: [IData.FindChannelByIndex](#), [IData.FindChannelByIndex1](#), [IData.GetIndexName](#), [IChannel.Index](#)

Name	Type	Description
IndexLevel	Smallint	defines the number of used index levels for specifying a channel - this depends on the type of channel (for a detailed description see T_ChIndex), e.g. if IndexLevel=1 then only Index1 will be used (Index2, etc. will be ignored)
Index1	Integer	the 1st index value
Index2	Integer	the 2nd index value
Index3	Integer	the 3rd index value
Index4	Integer	the 4th index value
Index5	Integer	the 5th index value

2.3.5 T_RecordPosition

```
record T_RecordPosition
```

specifies a certain record position by counting data blocks (`Dir`) and the remaining samples (`Mid`) that do not form a complete data block yet.

e.g. [IApp.InitScopeTrig](#), [IData.FirstTimeStamp](#), [IData.CurrentPos](#), [IData.EndStamp](#)

Name	Type	Description
Mid	Integer	the number of complete data blocks
Dir	Integer	the number of remaining samples

2.3.6 T_ReducedRec

```
record T_ReducedRec
```

`T_ReducedRec` is a record of reduced data values of the following kinds: Minimum, Maximum, Average and RMS.

see also: [The Buffer Structure](#)

Name	Type	Description
Ave	Single	Average
Max	Single	Maximum
Min	Single	Minimum
Rms	Single	RMS (Root Mean Square)

2.4 Constants

These constants are not in the type library, so you really have to use the values.

2.4.1 IOCodes

IOCodes for the Amplifier interface: see [IAmplInterfaces](#)

Error codes for Automation

Dec	Name	Description
-1	amplIOErrWrongIOCode	IO code not defined
-2	amplIOErrWrongVarType	variant <code>ParamIn</code> has wrong type
-3	amplIOErrIndexOutOfBounds	Index out of bounds

2.4.1.1 IOCodes for IAmplInterface

see [IAmPInterface](#) for a detailed example

see also: [IAmPIInterface.IOControl](#)

De c	Name	Out Param Type	Description
0	intDaqMaxChannels	Integer	maximum number of DAQ channels for the amplifier interface
1	intAdditionalModules	Integer	number of additional modules
2	intInterfaceType	String	interface type
3	intSendCommand	array of parameters	used to send a command to the amplifier:
4	intInterfaceDescription	String	interface description of the amplifier
5	intMainInterfaceOffline	Boolean	if the main interface is offline
6	intSubInterfaceOffline	Boolean	if a sub interface is offline

2.4.1.2 IO Codes for Chain properties

see also: [IAmpIChain.IOControl](#)

Dec	Name	Input Parameter Type	Out Parameter Type	Description
0	chainGetFoundIdx	-	Integer	<p>the highest amplifier index that has been found: e.g. when you have a DEWE-43 and PAD activated and MSI activated, there are 3 Amplifiers in the chain if no MSI adapter is connected, the highest index will be 0 if an MSI adapter is connected, the highest index will be 2</p>
1	chainGetAmplTypeAt	Integer	String	<p>will return a String to identify the amplifier type at the given index: e.g. when you have a DEWE-43 and PAD activated and MSI activated, there are 3 Amplifiers in the chain 0 will return DEWEUSB_AMPLIFIER 1 will return PAD_AMPLIFIER 2 will return MSI_AMPLIFIER</p>

2.4.1.3 IO Codes for Amplifier properties

see also: [Amplifier.IOCtrl](#)

De c	Name	Input Param eter Type	Out Para meter Type	Description
0	amplGetModuleName	-	String	name of the module: e.g. 43-V for a DEWE-43 amplifier
1	amplGetModuleUnit	-	String	unit of the module: e.g. mV/V for bridge measurement
2	amplGetShortInfoString	-	String	short information string: e.g. 1000 mV/V; Exc 5V
3	amplGetModuleType	-	Integer	
4	amplGetModuleSN	-	String	serial number of the amplifier: e.g. D0201A04
5	amplGetModuleRev	-	String	module revision number: e.g. 1.2.0.0
6	amplGetModuleFirmware	-	String	module firmware version: e.g. 1.7
7	amplGetModuleBaseType	-	Integer	module base type
8	amplGetModuleBaseSN	-	String	module base serial number
9	amplGetModuleBaseRev	-	String	module base revision
10	amplGetModuleBaseFirmware	-	String	module base firmware
11	amplCalDate	-	String	calibration date: e.g. 26/01/2012
12	amplAdjustDate	-	String	
13	amplFilterName	-	String	
14	amplFilterSN	-	String	
15	amplFilterRev	-	String	
16	amplGetOptionalGain	-	Double	
17	amplModuleBaseCorrGain	-	Double	
18	amplModuleBaseCorrOffset	-	Double	
19	amplModuleSubCorrGain	-	Double	

De c	Name	Input Parameter Type	Out Parameter Type	Description
20	amplModuleSubCor rOffset	-	Double	
21	amplSetOptionalGain	Double	-	
22	amplOptionDigOut	-	Byte	
Measurement				
10 0	amplGetMeasureme ntCount	-	Integer	number of available measurements: e.g. 2 for a typical DEWE-43 channel (which supports Voltage and Bridge measurement)
10 1	amplGetMeasureme ntIndex	-	Integer	Index of the measurement that is currently selected see also: amplSetMeasurementIndex
10 2	amplGetMeasureme ntStringAt	Integer	String	a string describing the measurement at the given index: e.g. Voltage, Bridge -1 returns the string for currently selected measurement see also: amplSetMeasurementString
10 3	amplGetMeasureme ntValueAt	Integer	String	a string describing the measurement at the given index: e.g. VOLTAGE, BRIDGE -1 returns the string for currently selected measurement see also: amplSetMeasurementValue
10 4	amplSetMeasureme ntIndex	Integer	-	change the measurement to the measurement at the Index of the given input parameter see also: amplGetMeasurementIndex
10 5	amplSetMeasureme ntString	String	-	change the measurement to the measurement identified by the given input parameter string see also: amplGetMeasurementStringAt
10 6	amplSetMeasureme ntValue	String	-	change the measurement to the measurement identified by the given input parameter string see also: amplGetMeasurementValueAt
Range				
11 0	amplGetRangesCou nt	-	Integer	the number of available measurement ranges for the amplifier - also depending on the currently selected measurement: e.g. DEWE-43 voltage measurement supports 4 ranges: 10V, 1V, 0.1V, 0.01V
11 1	amplGetRangeInde x	-	Integer	Index of the range that is currently selected see also: amplSetRangeIndex
11 2	amplGetRangeStri ngAt	Integer	String	a string describing the range at the given index: e.g. 0.1 V -1 returns the string for currently selected range string see also: amplSetRangeString
11 3	amplSetRangeInde x	Integer	-	change the range to the range at the Index of the given input parameter see also: amplGetRangeIndex
11 4	amplSetRangeStri ng	String	-	change the range to the range identified by the given input parameter string see also: amplGetRangeStringAt
11 5	amplGetRangeGain At	Double	-	not corrected in case the amplifier has CorrFactors
11 6	amplGetRangeOffs etAt	Double	-	not corrected in case the amplifier has CorrFactors
Low Pass Filters				
12 0	amplGetFiltersCo unt	-	Integer	the number of available low pass filters for the amplifier

De c	Name	Input Param eter Type	Out Para meter Type	Description
12 1	amplGetFilterIndex	-	Inte ger	Index of the low pass filter that is currently selected <i>see also:</i> : amplSetFilterIndex
12 2	amplGetFilterStringAt	Inte ger	Stri ng	a string describing the low pass filter at the given index: e.g. 100 kHz -1 returns the string for currently selected low pass filter string <i>see also:</i> : amplSetFilterString
12 3	amplGetFilterValueAt	Inte ger	Doub le	value of type Double for the low pass filter at the given index: e.g. 100000 -1 returns the value of type Double for currently selected low pass filter <i>see also:</i> : amplSetFilterValue
12 4	amplSetFilterIndex	Inte ger	-	change the low pass filter to the Index of the given input parameter <i>see also:</i> : amplGetFilterIndex
12 5	amplSetFilterString	Stri ng	-	change the low pass filter to the given input parameter string <i>see also:</i> : amplGetFilterStringAt
12 6	amplSetFilterValue	Doub le	-	change the low pass filter to the given input parameter <i>see also:</i> : amplGetFilterValueAt
High Pass Filters				
13 0	amplGetHPFiltersCount	-	Inte ger	the number of available high pass filters for the amplifier e.g. the MSI-BR-ACC adapter 3 high pass filters: 1.4Hz, 3Hz, 10Hz
13 1	amplGetHPFilterIndex	-	Inte ger	Index of the high pass filter that is currently selected <i>see also:</i> : amplSetHPFilterIndex
13 2	amplGetHPFiltersStringAt	Doub le	Stri ng	a string describing the high pass filter at the given index: e.g. 1.4 Hz -1 returns the string for currently selected high pass filter string <i>see also:</i> : amplSetHPFilterString
13 3	amplGetHPFilterValueAt	Inte ger	Doub le	value of type Double for the high pass filter at the given index: e.g. 1.4 -1 returns the value of type Double for currently selected high pass filter <i>see also:</i> : amplSetHPFilterValue
13 4	amplSetHPFilterIndex	Inte ger	-	change the high pass filter to the Index of the given input parameter <i>see also:</i> : amplGetHPFilterIndex
13 5	amplSetHPFiltersString	Stri ng	-	change the high pass filter to the given input parameter string <i>see also:</i> : amplGetHPFilterStringAt
13 6	amplSetHPFilterValue	Doub le	-	change the high pass filter to the given input parameter <i>see also:</i> : amplGetHPFilterValueAt
Input Types				
14 0	amplGetInputTypeCount	-	Inte ger	the number of available input types for the amplifier
14 1	amplGetInputTypeIndex	-	Inte ger	Index of the input type that is currently selected <i>see also:</i> : amplSetInputTypeIndex
14 2	amplGetInputTypeStringAt	Inte ger	Stri ng	a string describing the input type at the given index -1 returns the string for currently selected input type string <i>see also:</i> : amplSetInputTypeString
14 3	amplGetInputTypeValueAt	Inte ger	Stri ng	a string describing the input type at the given index -1 returns the string for currently selected input type string <i>see also:</i> : amplSetInputTypeValue
14 4	amplSetInputTypeIndex	Inte ger	-	change the input type to the Index of the given input parameter <i>see also:</i> : amplGetInputTypeIndex
14 5	amplSetInputTypeString	Stri ng	-	change the input type to the given input parameter string <i>see also:</i> : amplGetInputTypeStringAt

De c	Name	Input Parameter Type	Out Parameter Type	Description
14 6	amplSetInputType Value	String	-	change the input type to the given input parameter string see also: <code>amplGetInputTypeValueAt</code>
Shunts				
15 0	amplGetExtShunts Count	-	Integer	
15 1	amplGetExtShuntIndex	-	Integer	
15 2	amplGetExtShuntsStringAt	Integer	String	e.g. Shunt 1
15 3	amplGetExtShuntValueResAt	Integer	Double	e.g. 50
15 4	amplGetExtShuntValuePowAt	Integer	Double	e.g. 0.25
15 5	amplGetExtShuntValueImaxAt	Integer	Double	e.g. 5
15 6	amplSetExtShuntIndex	Integer	-	
15 7	amplSetExtShuntsString	String	-	
15 8	amplSetExtShuntCustRes	Double	-	
15 9	amplSetExtShuntCustPow	Double	-	
PAD Range				
16 0	amplGetPADRangeCode	-	Integer	the range code refers to the Module range of the PAD module
16 1	amplGetPADRangeIndex	-	Integer	the index of the range of the PAD module
16 2	amplSetPADRangeCode	Integer	-	set the the range code of the PAD module
Excitation				
17 0	amplGetExcCount	-	Integer	the number of available excitation voltages for the amplifier
17 1	amplGetExcIndex	-	Integer	the index of the currently selected excitation voltage
17 2	amplGetExcStringAt	Integer	String	a string describing the excitation voltage at the given index: e.g. 2.5 V -1 returns the string for currently selected excitation voltage see also: <code>amplSetExcString</code>
17 3	amplGetExcValueAt	Integer	Double	value of type Double for the excitation voltage at the given index: e.g. 2.5 -1 returns the value of type Double for currently selected excitation voltage see also: <code>amplSetExcValue</code>
17 4	amplSetExcIndex	Integer	-	change the excitation voltage to the Index of the given input parameter see also: <code>amplGetExcIndex</code>
17 5	amplSetExcString	String	-	change the excitation voltage to the given input parameter string see also: <code>amplGetExcStringAt</code>

De c	Name	Input Param eter Type	Out Para meter Type	Description
17 6	amplSetExcValue	Double	-	change the excitation voltage to the given input parameter <i>see also:</i> amplGetExcValueAt
17 7	amplGetExcInfo	String	-	informational string about the excitation voltage: e.g. Excitation voltage depends on connection
Excitation Unit				
18 0	amplGetExcUnitCo unt	-	Inte ger	the number of available excitation voltage units for the amplifier
18 1	amplGetExcUnitIn dex	-	Inte ger	the index of the currently selected excitation voltage unit <i>see also:</i> amplSetExcUnitIndex
18 2	amplGetExcUnitAt	Inte ger	Stri ng	a string describing the excitation voltage unit at the given index: e.g. V -1 returns the string for currently selected excitation voltage unit <i>see also:</i> amplSetExcUnit
18 3	amplSetExcUnitIn dex	Inte ger	-	change the excitation voltage unit to the <i>Index</i> of the given input parameter <i>see also:</i> amplGetExcUnitIndex
18 4	amplSetExcUnit	Stri ng	-	change the excitation voltage unit to the given input parameter <i>see also:</i> amplGetExcUnitAt

2.4.1.4 IO Codes for Amplifier commands

see also: [IAmplifier.IOControl](#)

Dec	Name	Description
100 00	amplUpdateAmpl	must be called when the amplifier properties have been changed: this includes amplPrepareScaling
100 01	amplPrepareScaling	must be called when you have changed any scaling properties

3 Document History

Version number	Comments
1.0.0	Initial revision
1.0.1	added Custom Export Call Diagram corrected MaxCalcDelay
1.0.2	added: GetDewesoftVersion , Start Events corrected: GetInterfaceVersion improved: Calculation Delay , CalcDelay , MaxCalcDelay , DCOM Server Registration
1.1.0	updated to new logo and new trademark: DEWESoft® added Example for buffer index corrected Pos parameter: GetValueAtAbsPos , GetIBValues , IBValues , GetUnscaledDataEx , GetUnscaledDataEx1 , GetScaledDataEx , GetScaledDataEx1 , GetTSDataEx , GetTSDataEx1 deprecated: GetRBValues

Index

_Unit 244

- A -

AbsMax 259
 AbsMin 259
 Acquiring 179
 ActiveChannels 372
 ActiveCount 170
 ActiveItem 170
 ActiveScreen 180
 ActualRunMode 180
 Add 170, 398, 402, 488, 490, 496, 510
 AddAsyncByteSample 260
 AddAsyncData 260
 AddAsyncDoubleSample 260
 AddAsyncInt64Sample 261
 AddAsyncIntegerSample 261
 AddAsyncShortintSample 261
 AddAsyncSingleSample 262
 AddAsyncSmallintSample 262
 AddAsyncString 262
 AddByteSample 263
 AddCh 320
 AddChannel 400, 512
 AddData 247, 263
 AddDoubleSample 263
 AddIn64Sample 264
 AddIndexName 480
 AddIndexNameEx 480
 AddIntegerSample 264
 AddIntegerSampleWithCalc 264

AddItemChannel 400
 AddNewEvent 497
 AddObj 435
 Address 364, 450
 AddShortintSample 264
 AddShortintSampleWithCalc 265
 AddSingleSample 265
 AddSingleSamples 265
 AddSmallintSample 266
 AddSmallintSampleWithCalc 266
 AddWordSample 266
 AdvCntMode 321
 AICount 544
 AISetupScreen 177
 AIUsedCount 544
 Alarms 181
 AIChannels 372
 AllowIBSkipping 498
 AlwaysEnableTrigger 181
 AlwaysReserveMemoryInSetup 474
 Ampl 144
 AmplChangeFactor 150
 AmplInterfaces 182
 AnalyseMode 373
 Analyze 182
 AOChannels 150
 AOGetManualAvail 178
 AOGroup 178
 aomBurst 527
 aomChirp 527
 aomFixed 527
 aomStepSweep 527
 aomSweep 527
 AOOperationMode 527
 AOSetManual 179
 aosLoop 527
 aosSingle 527

AOSweepMode	527	BaseMode	321
aowArbitrary	528	BaudRate	254
AOWaveForm	528	BitCount	268
aowRect	528	BlockSize	313
aowSaw	528	BroadcastPosChanged	522
aowSine	528	BroadcastScaleChanged	523
aowTrian	528	BufSize	525
aowWhiteNoise	528	BuildChannelList	373
Apply	408, 440	Byte0	545
ApplyChannels	373	Byte1	545
ApplyDBBuf	356	Byte2	545
ArrayChannel	267	Byte3	545
ArrayInfo	267	Byte4	545
ArraySize	267	Byte5	545
Async	267	Byte6	545
AsyncBufSize	474	Byte7	545
atFloat	542	Bytes	268
atFloatLinearFunc	542		
atString	542	- C -	
AType	311	CalcDelay	268
AutoCreate	410	CalcScopeTrig	183
AutoFlipAbsTime	410	CalcSRDiv	269
AutoFlipFile	411	Calculate	449
AutoFlipSize	411	CalculateFromPos	239
AutoFlipUnit	412	CalDate	544
AutoZero	361	CameraCount	524
Avail	164	Cameras	525
Ave	546	CAN	183
AveCount	239	CanAutoCalculate	321, 356
AveragedCPB	182	canGetCardCodes	529
AveragedFFT	182	canGetCardCount	529
AverageType	239	canGetData	529
AxisDef	237	canGetDeviceType	529
AxisType	243	canHideFrame	529
		canInit	529
- B -		canportCaptureEndRead	529
BaseFileName	413	canportCaptureGetMessageCount	529

canportCaptureReadMessage 529
 canportCaptureSet 529
 canportCaptureStartRead 529
 canportClearBuffers 529
 canportEnableOutput 529
 canportGetBaudRates 529
 canportGetData 529
 canportGetOutputEnabled 529
 canportPrepareAcq 529
 canportStartAcq 529
 canportStopAcq 529
 canportWriteFrame 529
 canPrepareAcq 529
 canShowFrame 529
 canStartAcq 529
 canStopAcq 529
 canSupportsOutput 529
 Caption 395, 419
 Capture 248
 Captured 254
 CardBitResolution 361
 CardChannel0 321
 CardChannel1 321
 CardCount 356
 CardGain 361
 CardOffset 361
 ChainList 173
 ChangeComPort 183
 ChangeDaqType 184
 ChangePluginChIndex 427
 ChangePluginChIndex1 428
 ChangeProject 489
 ChangeSetupScreen 521
 ChangeThreshold 269
 Channel 313, 505
 Channel Setup 16
 Channels 512
 CheckRegistration 491
 CheckSampleRate 326
 ChNo 269
 Clear 320, 402
 ClearAllChannels 481
 ClearChannels 512
 ClearChannelsInstance 457
 ClearModule 444
 CloseFile 430
 CntAux 322
 CntAuxInv 322
 CNTCount 544
 CntDoManualReset 322
 CntEncoderMode 322
 CntEncoderZero 323
 CntEventWithZero 323
 CntFilter 323
 CntGate 323
 CntGateInv 323
 CntMode 324
 CntnewValueUpdateMode 324
 CntPair 324
 CntResetOnStartMeasure 324
 CntSignalZero 325
 CntSource 325
 CntSourceInv 325
 CntUpDownMode 325
 CNTUsedCount 544
 Color 396
 ColorArr 237
 ConfigCode 450
 ConfigMode 185
 Configure 454, 457
 ConnTypes 528
 ControlChannelFlags 270
 ControlChannelState 270
 ControlChFlags 528

ControlsClock 152
CopyToString 365, 450
CopyUnitToString 365, 450
Count 170, 171, 172, 245, 256, 319, 355, 372, 392, 399, 403, 407, 425, 430, 436, 442, 448, 488, 491, 495, 510, 526
Coupling 393
CreateConnection 271
CreateCustomGroupAndControl 400
CreateGroupAndControl 401
ctLast 528
ctNew 528
ctOverlap 528
ctrlCh_AllowAlarmDO 528
ctrlCh_NumType_Bool 528
ctrlCh_NumType_Float 528
ctrlCh_NumType_Int 528
ctrlCh_NumType_Mask 528
ctTrigger 528
Current 495
CurrentPos 374
CurrentPosD 374
CurrentSource 365
CursorChannel 243
CustomCANMessages 529
CustomDAQMessages 530
CustomExpEventIDs 530
CustomImpMessages 531
CustomMathFrameMessages 532
CustomMathMessages 533
CustomName 164
CustomSensorOffset 362
CustomSensorScale 362
CustomVCMessages 537

daqAmplGetMeasurement 530
daqAmplMeasurementsCount 530
daqAmplSetMeasurementCode 530
DaqData 445
DaqDeviceInfo 544
daqGetDeviceType 530
DaqGroup 185
DaqNNames 365
DaqNNamesCount 366
daqSetNamePrefix 530
DaqType 357
daqUpdateXMLAISetup 530
Data 185, 404, 420, 451
DataCount 390
DataLost 185, 357
DataRegionChanged 523
DataSections 430
DataType 274
DBBufSize 271
DBDataSize 271
DBPos 271
DBTimeStamp 272
DBValues 272
DBValuesDouble 272
DCChangeFactor 152
DChannels 523
DefaultMax 437, 474
DefaultMin 437, 475
DefaultRes 437, 475
DeleteChannel 512
DeltaFreq 154
DeltaTime 505, 513
Description 274
DetectModule 445
DH 403
DIChannels 325
DICount 544

- D -

Daq 185

DIFilter	354	erExit	538
DIInvert	354	erGetData	538
DimCount	237	erMessageBox	538
DimSizes	237	erProgress	538
DInputGroups	523	erStartStoring	538
Dir	546	erStopStoring	538
Direction	505, 513	erTrigger	538
Direction1	506, 514	erTriggerStop	538
DisableKeyboardShortcuts	186	etChannelBased	539
DisableStoring	186	etKeyboard	538
DiscreteList	275	etModule	538
DisplayFrameTemplates	404	etPicture	538
DIUsedCount	544	etStart	538
DriverVersion	544	etStop	538
DStartDataAvail	273	etText	538
DStopDataAvail	273	etTrigger	538
DStopDataAvailDir	273	etValueBased	539
DW	403	etVoice	538
Dynamic Acquisition Rate	15	etVStart	538
		etVStop	538
- E -		evAfterLoadFile	538
Enabled	186	evCalculateAnalysis	538
EnableOutput	249	evChannelRemoved	538
EndAlarm	165	evEnterHardwareSetup	538
EndChannel	335	EventIDs	538
EndDataFolder	336	EventList	186
EndDataSync	375	EventReason	538
EndExport	336	EventType	538
EndHeader	336	evEstablishConnections	538
EndInfo	337	evGetDisabledChannels	538
EndRead	249	evGetIndexLevelName	538
EndStamp	375	evGetInputGroups	538
EndStampD	376	evGetReplayMode	538
EndValue	337	evGetStartErrorParams	538
erAlarm	538	evHasAbsoluteClock	538
erDataLost	538	evHideAnalysisFrame	538
erException	538	evlsMasterClock	538

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
evNewDataFileCreated	538												
evOnAlarmEx	538												
evOnCheckSampleRate	538												
evOnGetSampleRates	538												
evOnInitiateAcq	538												
evOnInitiateHardware	538												
evOnInitiateSetup	538												
evOnSetSampleRate	538												
evOnUpdateHWXML	538												
evOnUpdateXML	538												
evPreInitiate	538												
evPrepareAnalysis	538												
evRepaintAnalysisFrame	538												
evResizeAnalysisFrame	538												
evSetDWVersion	538												
evSetRegistrationHelper	538												
evShowAnalysisFrame	538												
evStartAnalysis	538												
evStartDataFolderEx	530												
evStopAnalysis	538												
evUpdateAnalysisFrame	538												
ExecuteModulesFunction	187												
ExpectedAsyncRate	276												
ExportData	189												
ExportDataEx	189												
Exported	278												
ExportOrder	276												
ExportRate	317												
ExportTypes	539												
ExternalClock	377												
ExternalTrigger	377												
ExtractNextChannel	478												
- F -													
FastCalc	278												
FastCalcInt32	279												
FFTBlockSize	485												
		FFTSampleRate	486										
		Field1	544										
		Field2	544										
		FileName	498										
		FileNameSettings	191										
		FileOpened	431										
		FileSize	498										
		FillModule	445										
		FilterCode	367										
		FilterFreq1	144										
		FilterFreq2	145										
		FilterOrder	145										
		FilterProtoType	146										
		Filters	367										
		FiltersCount	367										
		FilterType	146										
		Find	399										
		FindChannel	378, 481										
		FindChannelByIndex	378										
		FindChannelByIndex1	378										
		FindChannelByIndexEx	379										
		FindInputGroup	481										
		FindModuleByID	443										
		FindNode	479										
		FindObjByID	436										
		Firmware	544										
		FirmwareSub	544										
		FirstIBLevel	279										
		FirstScanDonePercent	191										
		FirstTimeStamp	379										
		FirstX	279										
		FixedExternalClock	191										
		FloatValues	243										
		FrameBufSize	257										
		FrameContentSize	257										
		FrameList	258										
		FramePos	258										

- F -

FastCalc	278	FrameContentSize	257
FastCalcInt32	279	FrameList	258
FFTBlockSize	485	FramePos	258

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	
FrameSizeInBytes	258												GetCPBData	239
FreeMemory	475												GetCPBXData	240
FreezeMode	380												GetCurrentProject	490
Freq	154												GetCurrentTime	329, 435
FREQAFindTriggerLevel	445												GetCursor	492
FREQAInputCoupling	366												GetData	240, 329, 525
FREQAOOutputFilter	366												GetDataBlocks	313
FREQATriggerLevel	366												GetDataBlocks1	313
FreqChangeFactor	155												GetDataPad	446
													GetDataValues	314
													GetDataValues1	314
- G -														
Get_AbsoluteTime	337												GetDBAddress	280
Get_CardFound	332												GetDeviceCode	330, 358
Get_DataCount	338												GetDeviceInfo	359
Get_ExportType	338												GetDeviceType	330
Get_Extension	338												GetDewesoftVersion	192
Get_FileName	339												GetDITrgLevel	358
Get_NumChannels	333												GetDWTypeLibVersion	329, 347, 462
Get_NumCNTChannels	332												GetFFT	486
Get_SupportsAsync	339												GetFFTData	241
Get_SupportsDouble	347												GetFramesInfo	526
Get_SupportsSRDiv	339												GetHardwareCode	484
Get_TimeIncrease	340												GetLBValues	281
GetBaudRate	249												GetIndex1	281
GetBaudRateList	250												GetIndexName	317, 381
GetBitmapInfoHeader	258												GetIndexName1	381
GetBitResolution	327												GetIndexNameShort	318, 381
GetBoardOpt	362												GetIndexNameShort1	382
GetBufferSize	327												GetInterfaceVersion	191
GetCardName	328												GetLicenseCode	485
GetChannelGain	328												GetMinMax	330
GetChannels	240												GetMsg	254
GetChannelSetup	279												GetOfflineStatus	282
GetClock	246												GetOptionName	331
GetClockOffset	246												GetOptionsCount	331
GetCNTBitResolution	328												GetProjects	490
GetCNTTrgLevel	357												GetRBValues	282

GetRegTypeWanted	484	gtInput	539
GetSampleRates	332	gtMemo	539
GetSamplesAcquired	382	gtSelection	539
GetScaledData	283	gtSound	539
GetScaledDataEx	283	Guid	429
GetScaledDataEx1	284		
GetSensor	362		- H -
GetSensorType	363	HardwareSetup	194
GetSerialNumber	446	HasFRF	195
GetSpecDir	193	Height	195
GetTrigIndexEx	502	HideCaptionBar	195
GetTrustedCode	484	HideFrame	409, 457
GetTS	525	HideSetupFrame	333
GetTSAAddress	284	HighpassType	367
GetTSBlocks	314	HoldoffTime	507
GetTSBlocks1	315	HoldoffTimeUsed	508
GetTSDData	285		- I -
GetTSDDataEx	285	IAISetupScreen	140, 141, 142
GetTSDDataEx1	286	IAlarmCond	164, 165, 166, 167, 168
GetTSValues	315	IAlarms	169, 170, 171
GetTSValues1	316	IAmplChain	171, 172
GetUnscaledData	286	IAmplChainList	172, 173
GetUnscaledDataEx	286	IAmplifier	177
GetUnscaledDataEx1	287	IAmplInterface	173, 174
GetValueAtAbsPos	287	IAmplInterfaces	174, 176
GetValueAtAbsPosDouble	288	IAOChannel	143, 144, 145, 146, 147, 148, 149
GetVectorScopeData	486	IAOGroup	149, 150, 152, 154, 155, 156, 157, 159, 160, 161, 162
GetVideoCompressDone	431	IApp	177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 189, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 219, 220, 221, 222, 223, 224, 225, 226, 228, 229, 230, 231, 232, 233, 234, 235, 236
GHOObjectType	539	IArryInfo	237, 238
GlobalHeader	193	IAveragedFFT	238, 239, 240, 241, 242
GoToInstruments	194	IAxisDef	242, 243, 244
Group	288		
GroupName	401		
Groups	383		
gtDir	539		
gtDivision	539		
gtFileName	539		

IBAbsMidRate 384
IBAbsRate 384
IBBufSize 289
IBDataSize 289
IBDataSizeEx 289
IBLevels 384
IBPos 289
IBPosEx 290
IBRate 384
IBValues 290
IBValuesEx 290
ICamera 257, 258, 259
ICAN 245, 246
ICANContext 246, 247
ICANMsg 247
ICANPort 248, 249, 250, 251, 252, 253
ICANPortContext 254, 255, 256
IChannel 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310
IChannelConnection 311, 313, 314, 315, 316, 317
IChannelGroup 317, 318
IChannelGroup2 318
IChannelGroups 318, 319
IChannelList 319
IChannelListEx 320
ICntChannel 320, 321, 322, 323, 324, 325, 326
ICNTGroup 256
ICPInput 368
ICustomDAQ 326, 327, 328, 329, 330, 331, 332, 333, 334
ICustomDAQ2 335
ICustomExport 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346
ICustomExport2 346, 347, 348, 349, 350, 351, 352
ICustomExport3 352
Id 442, 443, 458, 492
IDaq 356, 357, 358, 359, 360
IDaqChannel 361, 362, 363, 364
IDaqData 364, 365, 366, 367, 368, 369, 370, 371
IDaqGroup 371, 372
IData 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390
IDataSection 390, 391
IDataSections 392
IDewePlugin 392
IDIChannel 353, 354
IDigitalTrigLevel 393, 394
IDIGroup 355, 356
IDIPort 356
IDiscreteItem 394, 395, 396, 397
IDiscreteList 398, 399
IDisplayFrameTemplate 400, 401, 402
IDisplayFrameTemplates 402, 403
IDisplayTemplate 403, 404
IEvent 404, 405
IEventList 406, 407
IExportFrame 408, 409
IFileNameSettings 409, 410, 411, 412, 413, 415, 417
IGHObject 417, 419, 420, 421
IGlobalHeader 424, 425, 426
ImportChannel 426
ImportGroup 427
IndexChanger 427, 428
InputGroup 428, 429
InputGroup 429, 430
ILoadEngine 430, 431, 432, 433, 434
IMasterClock 434, 435
IMath 435, 436
IMathChannel 437

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
IMathContext	437, 438, 439											
IMathFrameContext	440											
IMathItem	441											
IMathModule	441, 442											
IMathObjContext	442											
IMathObject	442, 443, 444											
IModule	444, 445, 446, 447, 448											
IModules	448, 449											
impClearImport	531											
impCloseFile	531											
impDeinit	531											
impEndImport	531											
impGetAbsStartTime	531											
impGetBlockCount	531											
impGetBlockSize	531											
impGetImportType	531											
impGetSampleRate	531											
impHideFrame	531											
implImport	531											
implInit	531											
impOpenFile	531											
ImportStatus	540											
ImportTypes	540											
impPrepareImport	531											
impResizeFrame	531											
impsData	540											
impsFinished	540											
impsGap	540											
impShowFrame	531											
impStartImport	531											
impVersionCheck	531											
IncDBSamples	291											
Index	165, 291, 429, 446											
Index1	545											
Index2	545											
Index3	545											
Index4	545											
		Index5	545									
		IndexEx	291									
		IndexLevel	545									
		IniFileDir	195									
		Init	196, 238									
		Initiate	454, 458									
		InitScopeTrig	196									
		INothing	449									
		InputChannels	438, 441									
		InputGroups	384									
		IOControl	172, 173, 177, 359									
		IOOfflineCalc	449									
		IPadData	450, 451, 452, 453, 454									
		IPlugin	454, 455, 456									
		IPlugin2	457, 458, 459, 460, 461, 462									
		IPlugin3	462, 463, 464, 465, 466, 467, 468									
		IPlugin4	468									
		IPluginChannel	474, 475, 476, 477									
		IPluginChannelXMLHelper	478, 479									
		IPluginGroup	480, 481, 482, 483									
		IPluginLicense	484, 485									
		IPluginLicense2	485									
		IPowerModule	485, 486, 487									
		IPowerModules	488, 489									
		IProjectManager	489, 490									
		IProperties	490, 491									
		IRegistrationHelper	491									
		IsAcqRunning	196									
		IsControlChannel	292									
		IScreen	491, 492, 493, 494									
		IScreens	494, 495									
		IsCurrent	493									
		ISetupMessages	496									
		IsSetupMode	197									
		IsSingleValue	292									
		IsSyncSource	501									
		IStoreEngine	497, 498, 499, 500									

IsTriggering	498	LoadFromXML	486
IsVideoCompressDone	431	LoadFromXML1	487
ISyncSource	501	LoadModuleSetup	200
ITChannelBased	540	LoadProject	200
Item	171, 172, 173, 246, 256, 319, 356, 372, 392, 399, 403, 407, 426, 430, 449, 488, 491, 495, 510, 526	LoadSequence	200
ItemChannels	238	LoadSetup	201, 458
ItemLoadSetup	541	LoadSetupFromXML	201
ItemSaveSetup	541	LogicalIndex	292
ItemSaveSetupAs	541	LogicalName	292
ITestRecord	544	LogSweep	156
ITiming	501	LongName	475
ITrig	501, 502, 503, 505	- M -	
ITrigger	507, 508, 509	MainDataDir	202
ITriggerCondition	511, 512, 513, 514, 515, 516, 517, 518, 519, 520	MainDisplayColor	293
ITriggerCondList	510	MainInterface	176
ITrigInfo	505, 506, 507	MainWindowHandle	202
itValueBased	540	MainWndMessage	202
IUserInterface	521, 522	Manual	506
IVCCContext	522, 523, 524	ManualStart	202
IVideo	524, 525	ManualStop	203
IVideoFrame	525	MarkAsOffline	475
IVideoLoadEngine	526	MasterClock	204
IVideoLoadEngines	526	Math	205

-

LastKey	197	MathObjectContext	444
Left	197	MathObject	436, 442
Level1	506, 515	MathType	444
Level2	506, 515	Max	546
Lines	241	MaxCalcDelay	385
ListenOnly	255	Measure	205
LoadDBC	197	Measurement	294
LoadDisplaySetup	198	MeasureMode	385
LoadEngine	199	MeasureSampleRate	205
LoadFile	199	MeasureSampleRateEx	205

DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®	DEWESoft®
MenuClick 206													
MenuItems 541													
MessageCount 251													
mfAmplZeroAll 541													
mfAmplZeroByGroupID 541													
mfBridgeZeroAll 541													
mfBridgeZeroByGroupID 541													
mfChargeBZeroAll 541													
mfDAQP_MDAQ_SetPowerOnDefaultAll 541													
mfDAQP_ResetPowerOnDefaultAll 541													
mfFreqAFindTriggerAll 541													
mfMDAQ_CalFileClearAll 541													
mfRescanDAQPAAt 541													
mfRescanDAQPMDAQAll 541													
mfRescanDLRAAll 541													
mfRescanMDAQAt 541													
mfRescanMSIAAll 541													
mfRescanMSIAAt 541													
mfRescanPADAll 541													
mfRescanPADAt 541													
mfResetFirstScan 541													
mfrHideFrame 532													
mfrInitFrame 532													
mfrResizeFrame 532													
mfrSetContext 532													
mfrSetFrame 532													
mfrShowFrame 532													
mfrVersionCheck 532													
mfSetBestRangeAll 541													
mfSetBestRangeByGroupID 541													
mfSetFilter40PerOfSRAAll 541													
mfSetFilter40PerOfSRByGroupID 541													
mfSetFilterCustomAll 541													
mfSetFilterCustomByGroupID 541													
mfSetHighestFilterAll 541													
mfSetHighestFilterByGroupID 541													
mfSetHighestRangeAll 541													
mfSetHighestRangeByGroupID 541													
mfShortOnOffAll 541													
mfShuntCalCheckAll 541													
mfShuntCalCheckByGroupID 541													
mfShuntOnOffAll 541													
Mid 546													
Min 546													
Mode 506, 516													
ModuleAmpl 368, 451													
ModuleError 368													
ModuleGain 363													
ModuleIndex 487													
ModuleOffset 363, 368, 451													
Modules 206													
ModulesFunction 541													
ModuleType 363, 369, 447, 452													
MountAllChannels 479													
MountChannel 427, 438, 482													
MountChannelEx 438, 482													
MountInputGroup 439, 483													
mptBackward 540													
mptForward 540													
mptNone 540													
MRealTimeStamp 385													
mthAcceptInputChannel 533													
mthCalculate 533													
mthClearCalc 533													
mthDeinit 533													
mthGetDefaultChannelColor 533													
mthGetDefaultChannelName 533													
mthGetDescription 533													
mthGetInputChannels 533													
mthGetInputGroups 533													
mthGetOutChDescription 533													
mthGetProps 533													
mthGetRecalcOnPosChanged 533													
mthGetRegistration 533													

DEWESoft®
 mthInit 533
 mthInitiate 533
 mthMountChannels 533
 mthNeedsMultipass 533
 mthPassCompleted 533
 mthPassStarted 533
 mthPreInitiate 533
 mthReset 533
 mthSampleRateChanged 533
 mthSetProps 533
 mthStopCalc 533
 mthUpdateXML 533
 mthVersionCheck 533
 MType 292
 MultiFileStartIndex 413
 OnBeforeStopAcq 464
 OnBigListLoad 464
 OnEvent 352, 468
 OnExit 465
 OnGetClock 465
 OnGetData 455, 459
 OnGetSetupData 465
 OnHideHWFrame 465
 OnMessage 335, 392
 OnOleMsg 459
 OnRepaintFrame 466
 OnResizeFrame 466
 OnShowHWFrame 466
 OnStartAcq 455, 459
 OnStartSetup 467
 OnStartStoring 455, 460
 OnStopAcq 455, 460
 OnStopSetup 467
 OnStopStoring 456, 460
 OnTrigger 456, 461
 OnTriggerStop 467
 OperationMode 157
 OptionCount 544
 OptionTypes 544
 OrList 503
 OutputChannels 439, 441
 Overflow 369
 Overlap 241, 316

- N -

Name 165, 243, 259, 294, 318, 369, 429, 444, 452, 493, 544
 NameArr 238
 NETMode 207
 NewSetup 208, 459
 NextDBLoad 431
 NotifyTrackingChanged 208
 NotOrList 502
 NumBlocks 316, 432
 NumValues 316
 OnBeforeStopAcq 464
 OnBigListLoad 464
 OnEvent 352, 468
 OnExit 465
 OnGetClock 465
 OnGetData 455, 459
 OnGetSetupData 465
 OnHideHWFrame 465
 OnMessage 335, 392
 OnOleMsg 459
 OnRepaintFrame 466
 OnResizeFrame 466
 OnShowHWFrame 466
 OnStartAcq 455, 459
 OnStartSetup 467
 OnStartStoring 455, 460
 OnStopAcq 455, 460
 OnStopSetup 467
 OnStopStoring 456, 460
 OnTrigger 456, 461
 OnTriggerStop 467
 OperationMode 157
 OptionCount 544
 OptionTypes 544
 OrList 503
 OutputChannels 439, 441
 Overflow 369
 Overlap 241, 316

- O -

ObjType 421
 OfflineCalc 208
 Offset 147, 295
 OnAfterCalcMath 463
 OnAfterStartAcq 463
 OnAfterStopAcq 463
 OnAlarm 463
 OnBeforeStartAcq 464

- P -
 PadData 447
 Parent 208
 Paused 499
 PauseStoring 209
 Phase 148
 PhaseChangeFactor 157
 PluginGUID 476

PortCount	247	RemoteControlled	211
Ports	247	Remove	171, 399, 489, 510
PosDir	405	RemoveObj	436
PosMid	405	Repaint	524
PostTime	508	ReserveMemory	476
PostTimeExtensionUsed	508	Reset	317
PostTimeUsed	508	ResumeStoring	211
PowerModules	209	ReTrigLevel	393
Precision	243	Rms	546
PreTime	509	- S -	
PreTimeUsed	509	Sample Rate	15, 16
PrintScreen	209	SampleRate	159, 386, 501
ProjectManager	210	SampleRateEx	386
Properties	429	Samples	386
ProvidesClock	467	SaveSetup	212, 461
- R -			
Range	148	SaveToXML	487
RangeCode	370, 452	SaveToXML1	487
RangeIndex	453	Scale	298
Ranges	370, 453	Scale_	299
RangesCount	370, 453	ScaleValue	299
RBBufSize	295	ScaleValueDouble	299
RBDATASize	295	Screens	213
RBPos	295	sdAddonsDir	542
RBValues	296	sdAppDir	542
ReadData	391	sdAverage	543
ReadData1	391	sdDataDir	542
ReadMessage	251	sdExportDataDir	542
Reduced Sample Rate	17	sdFilter4th	543
ReducedOnly	432	sdFilter6th	543
ReducedRate	210	sdFilter8th	543
RegType	210	sdIniDir	542
Reload	432	sdLogDir	542
ReloadBlock	432	sdScriptsDir	542
ReloadEx	433	sdSetupDataDir	542
Remote	370	sdSkip	543

sdSystemDir	542	SetDoubleFloat	349
sdTempDir	542	SetErrMsgCount	255
SecondX	299	SetExpApp	409
SelectorIndex	300	SetExternalClock	387
SelectorIndexLevel	300	SetFreezeMode	301
SelectorIndexStartLevel	300	SetFullScreen	214
SendCommand	214	SetHeaderData	215
SendFrame	252	SetIndex	477
SendKey	214	SetInstrument	215
SerialNumber	544	SetIsSingleValue	302
Set_AbsoluteTime	340	SetLicenseCode	485
Set_DataCount	341	SetMainDataDir	216
Set_FileName	341	SetMainToolBar	216
Set_TimeIncrease	341	SetModule	448
SetAbsMax	347	SetPad	448
SetAbsMin	348	SetRangeMax	349
SetApp	333, 348, 456	SetRangeMin	350
SetAsStringChannel	300	SetRemoteMode	217
SetAsync	301	SetScopeParams	217
SetBaudRate	253	SetScopeUsed	219
SetBoardOpt	363	SetScreenIndex	219
SetCANPort	468	SetSensor	364
SetCardGain	364	SetSettings	477
SetCh	320	SetSRDiv	302
SetChannel	348	SetSRDivType	303
SetChannelColor	349	SetStartTimeUTC	387
SetChannelSetup	301	SetStoreMode	220
SetChNo	476	Settings	303
SetCNTTrgLevel	359	SetTotalMsgCount	255
SetColumnVisible	141	SetTrigOffset	350
SetCursor	493	SetupDOMDoc	402
SetDaq	447	SetupMessageType	541
SetDaqAddress	447	SetupSampleRate	221
SetData	456	SetupScreen	221
SetDataType	301	ShortCopyToString	371, 453
SetDeviceCalDate	360	Show	494
SetDITrgLevel	360	ShowCaptionBar	222

ShowChannelSetup	142, 303	StartModuleScan	228
ShowFrame	409, 461	StartRead	253
ShowInfoChannels	159	StartStamp	388
ShowInstrumentsInFullScreen	222	StartStampD	388
Shown	304	StartStoreTime	389
ShownChannels	387	StartStoreTimeChanged	499
ShowPropertyFrame	223	StartStoreTimeUTC	389
ShowSensorEditor	224	StartStoring	228
ShowSetupFrame	334	StartTime	161
ShowSROptions	224	StartTimeField	344
ShowStoreOptions	225	StartTrig	509
ShowStyle	226	StartValue	244, 344
ShowTrigCondSetup	522	StartVideoCompress	434
ShrinkFile	433	Status	165
SingleValue	304	StayOnTop	229
Size	244	StepValue	244
smsgError	541	Stop	229
smsgHint	541	StopAcq	334
smsgWarning	541	StopEvents	351
SpecDirectory	542	StopFreq	161
SpeedCode	454	StopModuleScan	229
SRDiv	297	StopOption	166
SRDivLCM	386	StopTime	162, 166
SRDivType	298	StopTrig	509
Start	228, 317	StopTrigger	167
StartAbsValue	341	StopVideoCompress	434
StartAcq	334	StoreCalculatedChannels	449
StartChannel	342	Stored	304
StartDataField	342	StoreEngine	230
StartDataFolder	343	StoreMode	499
StartDataSync	387	Storing	500
StartDBLoad	433	StringValues	244
StartEvents	350	SubInterface	174
StartExport	343	SupportsOutput	246
StartExtractChannels	479	SuppressMessages	230
StartFreq	160	SweepMode	162
StartInfo	343	SyncSource	238

TypicalMinValue 306

- T -

T_CANFrame 545
T_ChIndex 545
T_RecordPosition 546
T_ReducedRec 546
Tag 305
TAxisType 542
TemplateName 402
Termination 255
Text 305
ThermLinearize 371
Time 391
TimeCond 517
TimeFormat 518
TimerInterval 231
TimeStamp 405
TimeUnit 518
TimeValue 519
Timing 232
Top 232
TotalErrMsgCount 253
TotalMsgCount 253
Tracking 501
TrackingOffset 500
Trigger 168, 233
TrigIndex 505
TrigInfo 405
TrigLevel 393
TrigLevels 326, 354
TrigLevelsCombined 326
TrigPos 391
TrigType 394, 520
TrigValue 507, 520
TSRDivType 543
Type_ 405
TypicalMaxValue 306

- U -

Unit_ 306
UnmountChannel 483
UpdateFrame 462
UpdateHardwareSetup 233
UpdateSetupScreen 233
UpdateXML 307
Used 256, 259, 308, 462, 544
UseDate 415
UsedChannels 390
UsedDatafile 233
UsedSetupfile 234
UseMultiFile 415
UserInterface 234
UserScaleMax 309
UserScaleMin 310
UseTime 417

- V -

Value 397
vcAcceptChannel 537
vcAcceptInputGroup 537
vcAddChannel 537
vcAddInputGroup 537
vcDrawData 537
vcGetDataRegion 537
vcHideFrame 537
vcInit 537
vcInitFrame 537
vcInitScreen 537
vcKeyChar 537
vcKeyDown 537
vcKeyUp 537
vcLoad 537

DEWESoft®
vcMouseClicked 537
vcMouseDoubleClick 537
vcMouseDown 537
vcMouseEnter 537
vcMouseLeave 537
vcMouseMove 537
vcMouseUp 537
vcMouseWheel 537
vcMouseWheelDown 537
vcMouseWheelUp 537
vcResize 537
vcResizeFrame 537
vcSave 537
vcShowFrame 537
vcStartAcq 537
vcVersionCheck 537
Version 234
Video 234
VideoLoadEngines 434
Visible 235
VRange 371

- W -

WaveForm 149
Width 235
Window 242
WriteAsyncDoubleValue 351
WriteAsyncValue 344
WriteDoubleValue 351
WriteErrorLog 235
WriteErrorMessage 235
WriteEvent 352
WriteInfoDate 345
WriteInfoInteger 345
WriteInfoSingle 345
WriteInfoString 346
WriteValue 346

- X -

xmlFile 543
xmIMSXML 543
xmlString 543
XMLType 543

- Z -

ZeroAllAutoChannels 236
ZoomIn 494
ZoomOut 494