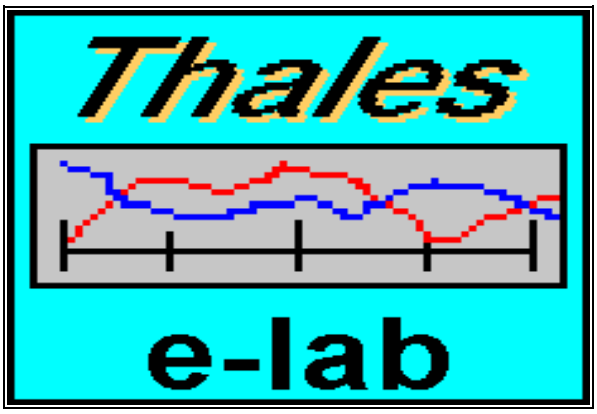


Thales VCL

by E-LAB Computers

Grombacher Str. 27
D-74906 Bad Rappenau
Tel. 07268/9124-0 Fax 07268/9124-24
e_lab_computers@compuserve.com

Thales Technische Komponenten für Borland Delphi®

Thales ist eine Visuelle-Komponenten-Bibliothek (VCL) für Borland Delphi®
Echte Delphi-VCL's, keine VBX'e, keine DLL's, keine Run-Time Lizenzen.
Ohne Source Code.

Thales gibt dem Software-Entwickler aus dem technisch-wissenschaftlichen und industriellen Bereich eine leistungsfähige und komplette Resource zur Erstellung von Programmen in der Messdaten-Erfassung und Auswertung als auch für Maschinensteuerung und Überwachung in die Hand.

Thales enthält weit über **50 Instrumente** und **Schalter** Komponenten.
Die Komponenten sind diverse Taster, Schiebeschalter, Drehschalter, Potis, Spin-Elemente für Text, Zahlen und Graphik, Digitale Displays, Analoge Instrumente in mehreren Ausführungen (Linear/logarithmisch, rund/panel), Serielle Schnittstellen Verwaltung, 8-Wege-Spin (grafische Maus), Maus-bedienbare Tastatur , INI-File Editor, numerische Ein/Ausgaben.

Thales enthält weitere **9 Graphik** Komponenten. Das sind: ein Curve Synthesizer, ein kompletter Logic-Analyzer bzw. Digitales Display, ein HPGL-Interpreter mit komfortablen Print-Funktionen, ein y/t Schreiber, ein x/y Schreiber, ein Oszilloscop, ein Daten-Logger (analog) und ein skalierbares Reissbrett.

Thales ist in der derzeitigen Form !nur! ein sogenanntes **Back-End** für den o.a. Anwendungsbereich, d.h. die Komponenten bilden das Interface zwischen dem Datenerfassungsteil und dem Verarbeitungsteil einerseits und dem Bediener andererseits. Die einzige "Signalverarbeitung" in den einzelnen Komponenten besteht aus der Eigenschaft (Property) "Damp". Dies ist eine Signal-Dämpfung bzw. Integration.

Thales Griechischer Philosoph und Mathematiker 625-547 v.Ch (Satz des Thales / Thales-Kreis)

Inhaltsverzeichnis

Inhaltsverzeichnis 2

Übersicht der Instrumente und Schalter 6

Übersicht der Schreiber und Displays 11

Allgemeine Prozeduren und Funktionen 13

Basis Klassen der Komponenten 14

Beschreibung der Komponenten 17

Bevel3D 17
Komponenten Deklaration 17

BackGround 19
Komponenten Deklaration 19

MultiBmp 20
Komponenten Deklaration 20

RotLabel 22
Komponenten Deklaration 22

LedButton 24
Komponenten Deklaration 24

SlideButton 26
Komponenten Deklaration 26

PictButton 28
Komponenten Deklaration 28

LedCheck 30
Komponenten Deklaration 30

TextSpin 32
Komponenten Deklaration 32

CombiSpin 34
Komponenten Deklaration 34

SerieSpin 36
Komponenten Deklaration 36

JoyStick 38
Komponenten Deklaration 38

UpDownT 40
Komponenten Deklaration 40

LeftRightT 41
Komponenten Deklaration 41

ArrowBtnT 42
Komponenten Deklaration 42

Slider 44
Komponenten Deklaration 44

SelectSlide 47

Komponenten Deklaration 47

Selector270 **49**
Komponenten Deklaration 49

PotiSlide **51**
Komponenten Deklaration 51

Poti270 **53**
Komponenten Deklaration 53

AlphaBar **55**
Komponenten Deklaration 55

FiFo **57**
Komponenten Deklaration 57

NumEdit **58**
Komponenten Deklaration 58

Spin7Seg **60**
Komponenten Deklaration 60

Disp7Seg **62**
Komponenten Deklaration 62

MagicEye **64**
Komponenten Deklaration 64

GaugeGP **65**
Komponenten Deklaration 65

MeterPanelH **67**
Komponenten Deklaration 67

MeterPanelV **70**
Komponenten Deklaration 70

Meter120 **74**
Komponenten Deklaration 74

Meter180 **78**
Komponenten Deklaration 78

Meter270 **82**
Komponenten Deklaration 82

Meter360 **86**
Komponenten Deklaration 86

Wecker **90**
Komponenten Deklaration 90

Kompass **93**
Komponenten Deklaration 93

CtrlTimer **95**
Komponenten Deklaration 95

Poti10T **98**
Komponenten Deklaration 98

BoxLine **101**
Komponenten Deklaration 101

BitWin 104
Komponenten Deklaration 104

Stroke 106
Komponenten Deklaration 106
Information 108

RollBar 109
Komponenten Deklaration 109

PickBox 111
Komponenten Deklaration 111

ComPortT 113
Komponenten Deklaration 113
Prozeduren, Funktionen und Properties 114

ComPort 121
Komponenten Deklaration 121
Beschreibung der Schnittstellen Komponenten ComPort 123
Eigenschaften/Properties: 123
Methoden/Prozeduren: 124

GamePort 128
Komponenten Deklaration 128

PlaySound 130
Komponenten Deklaration 130

IniEditor 131
Komponenten Deklaration 131

SmartTimer 132
Komponenten Deklaration 132

PrecTimer 133
Komponenten Deklaration 133
Hinweis zur Benutzung 133

FormSize 134
Komponenten Deklaration 134

DoubleInst 135
Komponenten Deklaration 135

SysInfo 136
Komponenten Deklaration 136

DataStrings 137
Komponenten Deklaration 137
Beschreibung 137

Object TDataList 138
Beschreibung 139

ExecExe 140
Komponenten Deklaration 140
Beschreibung 140

Iconize 142
Komponenten Deklaration 142
Beschreibung: 143

Beispiel Programm 143

Stack **144**
Komponenten Deklaration 144

Spy **145**
Komponenten Deklaration 145

VersInfo **146**
Komponenten Deklaration 146
Beschreibung 147

ThProcPriority **148**
Komponenten Deklaration 148

ScrnSaveCtrl **149**
Komponenten Deklaration 149

RecCtrl **150**
Komponenten Deklaration 150

Logic **152**
Komponenten Deklaration 152
Eigenschaften/Properties: 155
Methoden/Prozeduren: 158
Variable 160

Plotter **161**
Komponenten Deklaration 161

Oscillo **163**
Komponenten Deklaration 163
Eigenschaften/Properties: 167
Methoden/Prozeduren: 172
Variable 174

xyGraph **175**
Komponenten Deklaration 175
Eigenschaften/Properties 181
Methoden/Prozeduren 185
Variable 187

xyGraphN **188**
Komponenten Deklaration 188

ytGraph **195**
Komponenten Deklaration 195
Eigenschaften/Properties 201
Methoden/Prozeduren 206
Variable 208

CurveSynth **209**
Komponenten Deklaration 209

ChartBar **215**
Komponenten Deklaration 215

ChartBase **217**
Komponenten Deklaration 217

TextScroll **219**
Komponenten Deklaration 219

Übersicht der Instrumente und Schalter

Bevel3D Basis-Element, einsetzbar als Hintergrund, als Schalter mit Rast- oder Tastfunktion, oder als Textschild.

BackGround Hintergrund für Formulare und Panels etc. Einstellbarer Farbverlauf und Farben. (Gradient Fill).

MultiBmp Graphisches Element einstellbar als Leinwand für Multi-BitMaps mit div. Schalterfunktionen. Geeignet zum abspielen von Bildsequenzen und als Animated Button.

RotLabel In 1º-Schritten drehbarer Schriftzug mit beliebigen Fonts, Farben und Textgrössen. Als Schattenschrift, vertieft, normal oder erhöht.

LedButton runde oder rechteckige Led, aus, ein und blinkend, mit/ohne Text, als Schalter mit Rast- bzw. Tastfunktion.

SlideButton Schiebeschalter mit beliebiger Beschriftung und Farben.

PictButton Leuchtfeld auch als Schalter bzw. Taste verwendbar. Mit 1, 2 oder 3 Textzeilen oder altern. BitMap. Blinkend.

LedCheck Schalter, als Ersatz für CheckBox und RadioButton. Led Farbe, Grösse und Erscheinungsbild sowie Textposition in weiten Grenzen wählbar.

TextSpin SpinElement mit bel. vielen Texten. Automatische String - Numerik Konvertierung. Mit/ohne Anschlag.

CombiSpin mit bel. vielen Texten, BitMaps, Integers und Floats

SerieSpin Anreihbares Spinelement mit bel. vielen Texten. Automatische String - Numerik Konvertierung. Mit/ohne Anschlag.

Joystick Graphischer Joystick mit 8 Richtungen zur Bedienung mit der Maus. Mit Rast, Tast und Repeatfunktionen. Repeatrate einstellbar.

UpDown Zum Joystick passende Doppeltaste mit Rast, Tast und Repeatfunktion

LeftRight Zum Joystick passende Doppeltaste mit Rast, Tast und Repeatfunktion

ArrowBtn Pfeiltaste mit 4 Richtungen, mehreren Formen. Rast, Tast und Repeatfunktion

Slider Schieberegler oder Schalter mit/ohne Raststellungen, beliebige Anfangs und Endwerten. Horizontal und vertikal. Erscheinungsbild in weiten Grenzen änderbar: Farben, Ränder, Knopf etc.

SelectSlide Schiebeschalter mit bis zu 20 Raststellungen. Frei beschriftbar.

Selector270 Drehschalter mit bis zu 20 Raststellungen. Frei beschriftbar.

PotiSlide SchiebePoti mit frei wählbaren Anfangs- und Endwerten, sowie Skalenteilung. Frei beschriftbar. Umschaltbar linear/logarithmisch.

Poti270 Dreh-Poti mit frei wählbaren Anfangs- und Endwerten, sowie Skalenteilung. Frei beschriftbar.

AlphaBar Dient als einfacher Tastaturersatz für tastaturlose Systeme oder Auswahl Element für Sonderzeichen usw.

FiFo Text Ein/Ausgabefeld mit FiFo Funktion. Bei Ausgabe erfolgt ein Durchschieben des Textes von rechts nach links (Laufschrift). Bei der Eingabe erfolgt ebenfalls ein Schieben von rechts nach links. 255 Zeichen Buffer

NumEdit Numerisches Ein/Ausgabefeld. Einsetzbar als Digitales Instrument, Daten-Anzeige und Eingabefeld. Bei der numerischen Eingabe ist ein Min und ein Max-Wert einstellbar. Ebenfalls die Anzahl der Vor- und Nachkomma-Stellen als auch Integerformat. Text, Int oder Floatwerte vorgebbar und rücklesbar. Eingabeüberwachung.

Spin7Seg Daumenradschalter mit 7-Segment Anzeige. Anzeige umschaltbar Vorzeichen, Hex, dezimal, alphanumerisch.

Disp7Seg 1 bis 8-stelliges 7-Segment Display. Alphanum. Ansteuerung. 2 abschaltbare Leds mit Blinkfunktion.

MagicEye Spezial BarGraph mit einstellbaren Anfangs- und Endwerten. Geeignet z.B. für einfache Spektrum-Anzeige usw.

GaugeGp Universeller BarGraph. Steuerbar in Prozent und realen Werten. Bar- und Meter-Mode. Windows 95 Stil. Erscheinungsbild frei gestaltbar.

MeterPanelH Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Schleppzeiger und Over/Underange Event.

MeterPanelV Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Anzeige wählbar: simpleMeter, BarGraph oder LedBar. Zusätzliche, abschaltbare digitale Anzeige. Schleppzeiger und Over/Underange Event.

Meter120 Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Zusätzliche, abschaltbare digitale Anzeige. In 90grad-Schritten drehbar. Schleppzeiger und Over/Underange Event.

Meter180 Panel Meter mit 180grad Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Zusätzliche, abschaltbare digitale Anzeige. In 90grad-Schritten drehbar. Schleppzeiger und Over-Underange Event.

Meter270 Panel Meter mit 270grad Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Zusätzliche, abschaltbare digitale Anzeige. Schleppzeiger und Over/Underange Event.

Meter360 Panel Meter mit 360grad Skala. Zwei unabhängige Skalen und Zeiger. Separat einstellbare Dämpfung/Integration von 0..10. Frei beschriftbar. Vier abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Schleppzeiger. Linear/Logarithmisch separat wählbar. Schleppzeiger und Over/Underange Event.

AlarmClock Uhr mit Weckfunktion. Abschaltbare digitale Uhrzeit- und Datum Anzeige.

Compass mit digitaler Anzeige von Ist-Kurs und Abweichung

CtrlTimer Zeitschaltuhr zum steuern von maschinellen Abläufen, wie z.B. Brenn - Öfen, Füllvorgängen, Dosierungen usw. Fast beliebig konfigurierbar: hhmmss, hhmm, mmss, hh, mm, ss. Vier Schaltpunkte (Events), drei davon mit frei wählbarer Zeit, einer nach Ablauf auf 0. Alle Parameter vom Anwender einstellbar.

Poti10T Dreh-Poti mit frei wählbaren Anfangs- und Endwerten, sowie Skalenteilung. Frei beschriftbar. Äusserer Knopf mit 270grad Weg, innerer mit 10 Gängen bzw. 3600Grad. Digitale Anzeige des eingestellten Werts. Preset Knopf.

BoxLine Anzeige Einheit zur Darstellung von binären Informationen z.B. Ports, Bytes etc. Horizontale und vertikale Darstellung. Reihenfolge der Bits umkehrbar. Jedes Bit ist mit einem separaten Bitmap darstellbar.

BitWin Kleines ToolWindow ähnlich „Hint„ zum Anzeigen von Mouse Cursor abhängigen Informationen. Beispiel in CurveDemo und Thales_T/TextScroll

Stroke Vektor Zeichensatz zur Verwendung bei allen Canvas-Objekten. Beliebig rotierbar, skalierbar, spiegelbar, komfortable Align-Funktionen.

RollBar Kleine auf und abrollbare Panels. Menueleiste und Buttons sowie Gesamtgrösse und Farben in weiten Grenzen verstellbar.

PickBox Identisch mit der Delphi-ListBox, besitzt aber für jeden String eine zusätzliche, beschriftbare Schalt- bzw. Infofläche.

ComPortT Dialog zum Einstellen einer seriellen Schnittstelle. Übernimmt das komplette Handling von Status, Senden und Empfangen der seriellen Schnittstelle. XON-XOFF Protokoll, abschaltbar. Error handling

ComPort Dialog zum Einstellen bzw. Auswahl von 4 Schnittstellen. Übernimmt das komplette Handling von Status, Senden und Empfangen der 4 seriellen Schnittstellen. XON-XOFF Protokoll, abschaltbar.

GamePort Nicht-visuelle Komponente zur Auswertung von bis zu 2 GamePort-Schnittstellen. Übernimmt das komplette Handling.

PlaySnd Nicht-visuelle Komponente zur Ausgabe von Wave-Dateien über Sound-Blaster etc. Geeignet zur programmgesteuerten Sprachausgabe und Ansage von Meldungen und Warnungen in Wave-Files.

IniEdit Nicht-visuelle Komponente zum Lesen und Schreiben von Windows-INI-Files. Unterstützt den Programmierer beim Erstellen, Auslesen und Verändern des programm-eigenen INI-Files.

SmartTime Nicht-visuelle Timer-Komponente. Benutzt nur einen der wenigen Windows-Timer, stellt aber eine praktisch fast unbegrenzte Zahl von Timerkanälen zur Verfügung. OneShot, Blink und Dauerlauf.

PrecTime Nicht-visuelle Timer-Komponente. Benutzt einen speziellen praezisen Windows-Timer. Einstellbare Genauigkeit 1msec. Minimal Tick 1msec.

FormSize Nicht-visuelle Komponente. Einstellbare Grössenbegrenzung für Formulare. Minimal- und Maximalwerte unabhängig wählbar.

DoubleInst Nicht-visuelle Komponente. Verhindert den mehrfach Aufruf einer Application. Wird auf die MainForm plaziert.

SysInfo Nicht-visuelle Komponente. Liefert Daten über vorhandene Resourcen, Laufwerke, CPU, Interrupts, Speicher, Betriebssystem etc.

DataStrings Nicht-visuelle Komponente. Kleine Datenbank. Ablegen, suchen und finden von Daten über einen Namen. Schreiben und Lesen aus Datei.

ExecExe Nicht-visuelle Komponente. Zum komfortablen Aufruf einer Exe- oder Com Datei aus dem Programm heraus, mit Wartefunktion.

Iconize Nicht-visuelle Komponente. Startet das Programm als Icon in der Taskleiste oder ganz unsichtbar.

Stack Nicht-visuelle Komponente. Dient als Softwarestack für fast beliebige Datentypen, zur Unterstützung z.B. komplexen Rekursionen.

Spy Gag-Komponente. Die Augen folgen immer dem Mauskursor.

VerInfo Nicht-visuelle Komponente. Dient zum erfassen, editieren und einbauen der sog. VersionInfo (Copyright, Versionsnummer, Company etc.) in das Exe-file. Enthält weiterhin einen Dialog, der diese Daten zur Laufzeit anzeigen kann.

ProcPriority Komponente zum Einstellen der Priorität eines Programmes, auch zur Laufzeit.

SreenSaver Komponente zur Manipulation des Windows Screensaver. Der Screen Saver kann ganz abgeschaltet werden oder die Einschaltverzögerung in weiten Grenzen manipuliert werden.

Hint Property-Editor Implementiert mehrzeilige Hints. Eigener Dialog.
String Property-Editor Verbesserter Stringlisten Editor für TStrings und TStringList
BitMap Property-Editor Komfortabler BitMap/Picture Editor mit History etc.

Übersicht der Schreiber und Displays

RecCtrl Komponente zur Steuerung der Aufnahme und Wiedergabe der graph. Komponenten wie z.B. xyGraph oder ytGraph.

Logic Umfangreiche und leistungsfähige Komponente zum Darstellen und Untersuchen bzw. Analysieren von digitalen Daten. Einstellbar von einer einfachen graphischen binär-Anzeige bis zum Logic-Analyzer. Buffer für 16000 32bit-Worte. Patternsuche.

Plotter HPGL-Interpreter mit komfortablen Druckmöglichkeiten. Kann die wesentlichen HPGL-Plotterbefehle graphisch auf dem Bildschirm oder jedem Windows-tauglichen Drucker ausgeben. Einstellbare Blattformate, Stiftstärken und Farben. Keine Beschränkung in der Dateigrösse.

DrawWin Skalierbares Reissbrett mit bis zu 10 eigenständigen Koordinaten-Systemen im Float(double) Format. Sämtliche Graphikfunktionen integriert.

Oscillo Komponente mit allen Funktionen eines SpeicherOszilloskops. 4 Kanäle, Zoomfunktion, Speichern, Laden und Drucken der Daten. Sehr leistungsfähig und komfortabel. Einstellbar von einfacher Kurven-Ausgabe bis zum kompletten Oszillographen.

xyGraph Komponente zur graphischen Darstellung von bis zu 4 Messreihen mit jeweils bis zu 8000 xy-Koordinaten in einem frei wählbaren Koordinatensystem. Die x und y-Achsen sind unabhängig voneinander Linear oder Logarithmisch einstellbar. Alle Parameter über Properties zugänglich. Eingebaute Datei-Operationen (Laden, Speichern, BitMap). Mehrere Printer/Hardcopy Funktionen. Zoom und Messfunktionen.

xyGraphN Komponente zur graphischen Darstellung von bis zu 16 Messreihen mit jeweils bis zu 8000 xy-Koordinaten. Jeder Kanal hat sein eigenes frei wählbares Koordinatensystem. Die x und y-Achsen sind unabhängig voneinander Linear oder Logarithmisch einstellbar. Alle Parameter über Properties zugänglich. Eingebaute Datei-Operationen (Laden, Speichern, BitMap). Mehrere Printer/Hardcopy Funktionen. Zoom und Messfunktionen.

ytGraph Fortlaufende graphische Darstellung ankommender Daten/Messwerte. Bis zu einer gewissen Geschwindigkeit echtzeitfähig. Auch abspielen von gespeicherten Daten. Zoom, Print und Messfunktionen. 4 Kanäle. Y-Achse lin/log. Eingebaute Datei-Operationen (Laden, Speichern, BitMap).

CurvSynth Stellt Kurven aus bis 40 Punkten durch Interpolation dar. Lineare, Bezier und Spline Interpolation.

ChartBar Bargraph Element zum graphischen Darstellen von Daten wie bei Spread-Sheets. Umrandung, Farben und Transparenz einstellbar.

ChartBase Basiselement zum plazieren und steuern der ChartBars.

TextScroll Textanzeige/Editor. Umschaltbar von Ascii in Hexmodus. Umfangreiche Manipulations-Möglichkeiten des Erscheinungsbildes. Diverse Highlights.

Zukünftige Komponenten

ytGraphN 16-Kanal yt-Graph
xtGraph Wie ytGraph, jedoch wie Papierschreiber senkrechte Darstellung
Loggger 16-Kanal Datenlogger ähnlich ytGraph, jedoch optisch getrennte Kanäle

Allgemeine Prozeduren und Funktionen

Unit Thales.dcu

```
function ConvertTime(cTime : double; rest : boolean) : string;
function GetColPos(Col : TColor) : integer;
Procedure RotatePnt(A1 : double; pp, p1 : TPoint; Var p2 : TPoint);
Procedure RotatePntF(A1 : double; pp, p1 : rxyCoord; Var p2 : rxyCoord);
 {RotatePnt will rotate a point (p1) through angle A1
  about the pivot point (pp) and return the rotated point in p2.}

Function ArcSin(X : double) : double;
Function ArcCos(X : double) : double;
Function ATan(X, Y : double) : double;
function CosA(w : double) : double;
function SinA(w : double) : double;
procedure SwapInt(var i1, i2 : LongInt);
Function CheckVal(val : double) : integer;
Function CheckDecimal(val : double) : integer;
Function CheckLength(val1, val2 : double; canv : TCanvas) : integer;
Procedure FontAutoSize(canv : TCanvas; fRect : TRect; str : string);
Function NextNonLinear(Val : double) : double;
Function Exp10(Val : double) : double;
Function Log10(Val : double) : double;
Procedure LogValText(Val : double; posx, posy : integer; canv : TCanvas);
function FormatEinh(val : double; BasisStr : string; decimals, zeros : boolean)
                                                                  : string;
function FormatBasis(val1, val2 : double; var res : double; BasisStr : string)
                                                                  : string;
Function GetFloatFromStr(str : string) : double;
function GetLongFromStr(str : string) : LongInt;
function GetColFromStr(str : string) : TColor;
function GetStrFromStr(str : string) : string;
procedure DrawButtonUp(canv : TCanvas; r : TRect);
procedure DrawButtonDown(canv : TCanvas; r : TRect);
procedure DrawButtonFirst(canv : TCanvas; Var r : TRect);
procedure PfeilRight(Canv : TCanvas; r : TRect; strich : boolean);
procedure PfeilLeft(Canv : TCanvas; r : TRect; strich : boolean);
procedure PfeilDown(Canv : TCanvas; r : TRect; strich : boolean);
procedure PfeilUp(Canv : TCanvas; r : TRect; strich : boolean);
function DoWordWrap(canv : TCanvas; len : integer; var lStr : string) : integer;
function DelSpace(Var pValue : String) : string;
function BinaryWord(W : Word) : String;
function BinaryLong(L : LongInt) : String;
function BinaryByte(B : Byte) : String;

function GetShortPath(path : string) : string;
(* convert a win95 path/filename type to a DOS-compatible and executable path *)

function LimitPathStr(const Filename: string; MaxLen: Integer): string;
(* shorten a very long path to MaxLen chars. Not executable *)
```

Basis Klassen der Komponenten

Unit Thales.dcu

```
type
  TRahmen = class(TCustomControl)
  private
    { Private-Deklarationen }
    FFramed   : boolean;
    FDrawMode : TDrawMode;
    Procedure SetBevelIn (pValue : TPanelBevel);
    Procedure SetBevelOut(pValue : TPanelBevel);
    procedure SetBorderWidth(pValue : Integer);
    procedure SetBevelWidth(pValue : Integer);
    procedure SetBordRight(pValue : Boolean);
    procedure SetFramed(pValue : Boolean);
    procedure SetBordLeft(pValue : Boolean);
    procedure SetBordTop(pValue : Boolean);
    procedure SetBordBottom(pValue : Boolean);
    procedure DrawBevels(thick : integer; color1, color2 : TColor);
    procedure SetBorders(thick : integer);
    Procedure SetAutoSize(pValue : Boolean);
    Procedure SetColBgnd(pValue : TColor);
    Procedure SetColBack(pValue : TColor);
    procedure SetSchalter(pValue : TTastStyle);
    procedure SetColBtnBord(pValue : TColor);
    procedure SetState(pValue : boolean);
  protected
    { Protected-Deklarationen }
    FOnClick    : TNotifyEvent;
    FOnRepaint  : TNotifyEvent;
    FOnDown     : TNotifyEvent;
    FOnUp       : TNotifyEvent;
    FAutoSize   : Boolean;
    FColBtnBord : TColor;
    FColBgnd    : TColor;
    FColBack    : TColor;
    FTastStyle  : TTastStyle;
    BtnFrame    : boolean;
    procedure Inverse(inv : boolean);
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
                                                    X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
                                                              override;
    procedure Paint; override;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  public
    { Public-Deklarationen }
    FBevelIn    : TPanelBevel;
    FBevelOut   : TPanelBevel;
    FBevWidth   : Integer;
    FBordWidth  : Integer;
    bRect       : TRect;
    FBordRight  : Boolean;
    FBordLeft   : Boolean;
    FBordTop    : Boolean;
    FBordBottom : Boolean;
    isDown      : Boolean;
    property Canvas;
```

```
    property ColBack     : TColor       read FColBack    write SetColBack;
    Property FontAutom   : boolean      read FAutoSize   write SetAutoSize;
    Property Framed      : boolean      read FFramed     write SetFramed;
    Property ColBtnBord  : TColor       read FColBtnBord write SetColBtnBord;
    Property ButtonStyle : TTastStyle   read FTastStyle  write SetSchalter;
    property OnClick     : TNotifyEvent read FOnClick    write FOnClick;
    property OnDown      : TNotifyEvent read FOnDown     write FOnDown;
    property OnUp        : TNotifyEvent read FOnUp       write FOnUp;
    property Status      : boolean      read isDown      write SetState;
    property DrawMode    : TDrawMode    read FDrawMode   write FDrawMode;
  published
    { Published-Deklarationen }
    property Enabled;
    property Visible;
    property ParentShowHint;
    property ShowHint;
    property DragMode;
    property DragCursor;
    property OnDragOver;
    property OnDragDrop;
    property OnEndDrag;
    property OnRepaint : TNotifyEvent read FOnRepaint  write FOnRepaint;
    property BevelIn   : TPanelBevel read FBevelIn write SetBevelIn;
    property BevelOut  : TPanelBevel read FBevelOut write SetBevelOut;
    property BevWidth  : integer read FBevWidth   write SetBevelWidth;
    property BordWidth : integer read FBordWidth  write SetBorderWidth;
    property BordRight : Boolean read FBordRight  write SetBordRight;
    property BordLeft  : Boolean read FBordLeft   write SetBordLeft;
    property BordTop   : Boolean read FBordTop    write SetBordTop;
    property BordBottom: Boolean read FBordBottom write SetBordBottom;
    property ColBgnd   : TColor  read FColBgnd    write SetColBgnd;
  end;

type
  TThalesBase = class(TCustomControl)
  private
    { Private-Deklarationen }
    FFramed   : boolean;
    FDrawMode : TDrawMode;
    Procedure SetBevelIn (pValue : TPanelBevel);
    Procedure SetBevelOut(pValue : TPanelBevel);
    procedure SetBorderWidth(pValue : Integer);
    procedure SetBevelWidth(pValue : Integer);
    procedure SetBordRight(pValue : Boolean);
    procedure SetFramed(pValue : Boolean);
    procedure SetBordLeft(pValue : Boolean);
    procedure SetBordTop(pValue : Boolean);
    procedure SetBordBottom(pValue : Boolean);
    procedure DrawBevels(thick : integer; color1, color2 : TColor);
    procedure SetBorders(thick : integer);
    Procedure SetAutoSize(pValue : Boolean);
    Procedure SetColBgnd(pValue : TColor);
    Procedure SetColBack(pValue : TColor);
    procedure SetSchalter(pValue : TTastStyle);
    procedure SetColBtnBord(pValue : TColor);
    procedure SetState(pValue : boolean);
  protected
    { Protected-Deklarationen }
    FOnClick    : TNotifyEvent;
    FOnRepaint  : TNotifyEvent;
    FOnDown     : TNotifyEvent;
    FOnUp       : TNotifyEvent;
    FAutoSize   : Boolean;
```

```
    FColBtnBord : TColor;
    FColBgnd    : TColor;
    FColBack    : TColor;
    FTastStyle  : TTastStyle;
    BtnFrame    : boolean;
    painted     : Boolean;
    procedure Inverse(inv : boolean);
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
                                                    X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
                                                                  override;
    procedure Paint; override;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  public
    { Public-Deklarationen }
    FBevelIn    : TPanelBevel;
    FBevelOut   : TPanelBevel;
    FBevWidth   : Integer;
    FBordWidth  : Integer;
    bRect       : TRect;
    FBordRight  : Boolean;
    FBordLeft   : Boolean;
    FBordTop    : Boolean;
    FBordBottom : Boolean;
    BlitBmp     : TBitMap;
    OrgBmp      : TBitMap;
    isDown      : Boolean;
    property Canvas;
    property ColBack     : TColor       read FColBack    write SetColBack;
    Property FontAutom   : boolean      read FAutoSize   write SetAutoSize;
    Property Framed      : boolean      read FFramed     write SetFramed;
    Property ColBtnBord  : TColor       read FColBtnBord write SetColBtnBord;
    Property ButtonStyle : TTastStyle   read FTastStyle  write SetSchalter;
    property OnClick     : TNotifyEvent read FOnClick    write FOnClick;
    property OnDown      : TNotifyEvent read FOnDown     write FOnDown;
    property OnUp        : TNotifyEvent read FOnUp       write FOnUp;
    property Status      : boolean      read isDown      write SetState;
    property DrawMode    : TDrawMode    read FDrawMode   write FDrawMode;
  published
    { Published-Deklarationen }
    property Enabled;
    property Visible;
    property ParentShowHint;
    property ShowHint;
    property DragMode;
    property DragCursor;
    property OnDragOver;
    property OnDragDrop;
    property OnEndDrag;
    property OnRepaint : TNotifyEvent read FOnRepaint  write FOnRepaint;
    property BevelIn   : TPanelBevel read FBevelIn write SetBevelIn;
    property BevelOut  : TPanelBevel read FBevelOut write SetBevelOut;
    property BevWidth  : integer read FBevWidth   write SetBevelWidth;
    property BordWidth : integer read FBordWidth  write SetBorderWidth;
    property BordRight : Boolean read FBordRight  write SetBordRight;
    property BordLeft  : Boolean read FBordLeft   write SetBordLeft;
    property BordTop   : Boolean read FBordTop    write SetBordTop;
    property BordBottom: Boolean read FBordBottom write SetBordBottom;
    property ColBgnd   : TColor  read FColBgnd    write SetColBgnd;
  end;
```

Beschreibung der Komponenten

Bevel3D

Basis-Element, einsetzbar als Hintergrund, als Schalter mit Rast- oder Tastfunktion, oder als Textschild.

Komponenten Deklaration

```
type
  TBevel3D = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FCaption   : String;
    FCaption1  : String;
    FTextStyle : tTextStyle;
    FColCapt   : TColor;
    FColCapt1  : TColor;
    FColBack   : TColor;
    FCaptEna   : Boolean;
    FCaptBlink : Boolean;
    FWordWrap  : Boolean;
    FCaptBevel : TPanelBevel;
    FGroupIdx  : integer;
    FAllowUp   : boolean;
    textR      : TRect;
    painted    : boolean;

    procedure TimerExpired(Sender: TObject);
    procedure SetGroupIdx(pValue : integer);
    Procedure SetBtnDown(pValue : boolean);
    Procedure SetCaption(pValue : string);
    Procedure SetCaption1(pValue : string);
    Procedure SetColBack(pValue : TColor);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColCapt1(pValue : TColor);
    Procedure SetCaptEna(pValue : Boolean);
    Procedure SetCaptBlink(pValue : Boolean);
    Procedure SetWordWrap(pValue : Boolean);
    Procedure SetCaptBevel(pValue : TPanelBevel);
    Procedure SetTextStyle(pValue : TTextStyle);
    Procedure DrawText(TextOn : boolean);
    procedure UpdateExclusive;
    procedure CMButtonPressed(var Message: TMessage); message CM_BUTTONPRESSED;
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
                        Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
```

```
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Caption     : String   read FCaption   write SetCaption;
    property Caption1    : String   read FCaption1  write SetCaption1;
    property ColCaptBack : TColor   read FColBack   write SetColBack;
    property ColCaption  : TColor   read FColCapt   write SetColCapt;
    property ColCaption1 : TColor   read FColCapt1  write SetColCapt1;
    Property CaptEnable  : Boolean  read FCaptEna   write SetCaptEna;
    property CaptBevel : TPanelBevel read FCaptBevel write SetCaptBevel;
    property TextAdjust: TTextStyle read FTextStyle write SetTextStyle;
    property Down        : boolean  read isDown     write SetBtnDown;
    property WordWrap    : boolean  read FWordWrap  write SetWordWrap;
    Property AllowAllUp  : boolean  read FAllowUp   write FAllowUp;
    Property CaptBlink   : boolean  read FCaptBlink write SetCaptBlink;
    property GroupIndex  : integer  read FGroupIdx  write SetGroupIdx;
    Property ColBtnBord;
    property OnDown;
    property OnUp;
    property OnClick;
    Property ButtonStyle;
    Property FontAutom;
    Property Font;
    Property ParentColor;
    Property ParentFont;
    Property Framed;
  end;
```

BackGround

Hintergrund für Formulare und Panels etc. Einstellbarer Farbverlauf und Farben. (Gradient Fill).

Komponenten Deklaration

```
TBackGround = class(TGraphicControl)
  private
    FDirection: TFillDirection;
    FBeginColor: TColor;
    FEndColor: TColor;
    FAutoSize: Boolean;
    FRealize: Boolean;
    FNumberOfColors: TNumberOfColors;
    FNumColors: TNumberOfColors;
    FPicture: TBitmap;
    FLogPal: PLogPalette;  {Pointer to a TLogPalette}
    FPalSize: LongInt;   {Used to calculate memory to allocate for TLogPalette}
    FPalette : HPalette;   {Used to store Palette created with LogPal}
    procedure SetFillDirection(Value: TFillDirection);
    procedure SetAutoSize(Value: Boolean);
    procedure SetRealize(Value: Boolean);
    procedure SetBeginColor(Value: TColor);
    procedure SetEndColor(Value: TColor);
    procedure SetNumberOfColors(Value: TNumberOfColors);
  protected
    procedure Paint; override;
    procedure MakePalette(BeginRed, BeginGreen, BeginBlue: Byte;
      RedDifference, GreenDifference, BlueDifference: Integer);
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    property Picture: TBitmap read FPicture;
    property PaletteHandle: HPalette read FPalette;
  published
    property AutoSize: Boolean read FAutoSize write SetAutoSize default False;
    property BeginColor: TColor read FBeginColor write SetBeginColor
        default clBlue;
    property EndColor: TColor read FEndColor write SetEndColor default clBlack;
    property FillDirection: TFillDirection read FDirection
        write SetFillDirection default fdTopToBottom;
    property NumberOfColors: TNumberOfColors read FNumColors
        write SetNumberOfColors default 16;
    property Realize: Boolean read FRealize write SetRealize default True;
    property Align;
    property DragCursor;
    property DragMode;
    property Enabled;
    property ParentShowHint;
    property ShowHint;
    property Visible;
  end;
```

MultiBmp

Graphisches Element einstellbar als Leinwand für Multi-BitMaps mit div. Schalterfunktionen. Geeignet zum abspielen von Bildsequenzen und als Animated Button.

Komponenten Deklaration

```
type
  tAnimSpeed  = (an10Ticks, an9Ticks, an8Ticks, an7Ticks, an6Ticks,
                 an5Ticks, an4Ticks, an3Ticks, an2Ticks, an1Ticks);

  TMultiBmp   = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FOnEndAnim : TNotifyEvent;
    FAnimSpeed : tAnimSpeed;
    FGlyph     : TBitMap;
    WorkBmp    : TBitMap;
    FBlinkOn   : Boolean;
    FGlyphNums : integer;
    FGlyphSel  : integer;
    FAutoSize  : boolean;
    FGlyphMode : boolean;
    FGroupIdx  : integer;
    FAllowUp   : boolean;
    FAnimate   : boolean;
    FColTransp : TColor;
    FGlyphDis  : integer;
    painted    : Boolean;
    sRect      : TRect;
    dRect      : TRect;
    saveGlyph  : integer;

    procedure TimerExpired(Sender: TObject);
    procedure SetAnimSpeed(pValue : tAnimSpeed);
    procedure SetGlyphNums(pValue : integer);
    procedure SetGlyphSel(pValue : integer);
    procedure SetGroupIdx(pValue : integer);
    Procedure SetBtnDown(pValue : boolean);
    Procedure SetGlyphMode(pValue : boolean);
    Procedure SetAutoSize(pValue : boolean);
    procedure DrawTaste(blink : boolean);
    Procedure SetBlinkOn(pValue : Boolean);
    Procedure SetAnimate(pValue : Boolean);
    Procedure SetColTransp(pValue : TColor);
    procedure SetGlyphDis(pValue : integer);
    function  GetGlyph : TBitmap;
    procedure SetGlyph(pValue: TBitmap);
    procedure DrawGlyph;
    procedure WMDrawGlyph(Var Message : TMessage); message WM_DRAWVALUE;
    procedure UpdateExclusive;
    procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
    procedure CMButtonPressed(var Message: TMessage); message CM_BUTTONPRESSED;
 protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
```

```
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
    procedure NextGlyph;
  published
    { Published-Deklarationen }
    property ColBack;
    Property ButtonStyle;
    property OnClick;
    property OnDown;
    property OnUp;
    Property ColBtnBord;
    Property ColTransp  : TColor     read FColTransp write SetColTransp;
    Property Glyph      : TBitMap    read GetGlyph   write SetGlyph;
    Property BlinkOn    : boolean    read FBlinkOn   write SetBlinkOn;
    property Down       : boolean    read isDown     write SetBtnDown;
    property AutoSize   : boolean    read FAutoSize  write SetAutoSize;
    property GlyphMode  : boolean    read FGlyphMode write SetGlyphMode;
    property Animate    : boolean    read FAnimate   write SetAnimate;
    property AnimSpeed  : tAnimSpeed read FAnimSpeed write SetAnimSpeed;
    property GlyphNums  : integer    read FGlyphNums write SetGlyphNums;
    property GlyphSel   : integer    read FGlyphSel  write SetGlyphSel;
    property GlyphDisSel: integer    read FGlyphDis  write SetGlyphDis;
    Property AllowAllUp : boolean    read FAllowUp   write FAllowUp;
    property GroupIndex : integer    read FGroupIdx  write SetGroupIdx;
    property OnEndAnim: TNotifyEvent read FOnEndAnim write FOnEndAnim;
  end;
```

RotLabel

In 1º-Schritten drehbarer Schriftzug mit beliebigen Fonts, Farben und Textgrössen. Als Schattenschrift, vertieft, normal oder erhöht.

Komponenten Deklaration

```
type
  TRotLabel = class(TCustomLabel)
  private
    fAngle      : longint;
    F3DEffect   : T3DEffect;
    FhOffSet,
    FvOffSet    : integer;
    FShadowCol  : TColor;
    procedure SetAngle(Value: longint);
    procedure setStyleEffect(Value : T3DEffect);
    procedure SetShadowColor(Value:TColor);
    procedure SetFhOffSet(value: integer);
    procedure SetFvOffSet(value: integer);
  protected
    { Protected declarations }
      fDegToRad, fCosAngle, fSinAngle: double;
      procedure Paint; override;
  public
    { Public declarations }
    fTranspar : boolean;
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure StyleChanged(Sender: TObject);
    procedure SetTransparent(value:boolean);
    procedure DirectAngle(Value: longint);
  published
     { Published declarations }
    property rlAngle      : longint   read fAngle     write SetAngle default 0;
    property rlStyle3D    : T3DEffect read F3DEffect  write setStyleEffect
                                                      default rlRaised;
    property rlShColor    : TColor    read FShadowCol write SetShadowColor
                                                      default clGray;
    property rlShOffSetH  : integer  read FhOffSet write SetFhOffSet default 5;
    property rlShOffSetV  : integer  read FvOffSet write SetFvOffSet default -5;
    property rlTransparent: boolean  read fTranspar   write SetTransparent
                                                      default true;

    property Align;
    property Caption;
    property Color;
    property DragCursor;
    property DragMode;
    property Enabled;
    property FocusControl;
    property Font;
    property ParentColor;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property ShowAccelChar;
    property ShowHint;
    property Visible;
```

```
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;
```

LedButton

Runde oder rechteckige Led, aus, ein und blinkend, mit/ohne Text, als Schalter mit Rast- bzw. Tastfunktion.

Komponenten Deklaration

```
type
  TLedButton   = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FLedOn     : Boolean;
    FLedBlink  : Boolean;
    FCaption   : String10;
    FCaption1  : String10;
    FCaptBevel : TPanelBevel;
    FLedShape  : TLedShape;
    FColLed    : TColor;
    FCaptType  : TCaptType;
    FColCapt   : TColor;
    FColCapt1  : TColor;
    FGroupIdx  : integer;
    FAllowUp   : boolean;
    painted    : boolean;
    PosLed     : TPoint;
    LedRect    : TRect;
    LedSize    : integer;
    TextFRect  : TRect;

    procedure SetGroupIdx(pValue : integer);
    Procedure SetCaptBevel(pValue : TPanelBevel);
    Procedure SetBtnDown(pValue : boolean);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColCapt1(pValue : TColor);
    procedure TimerExpired(Sender: TObject);
    Procedure SetLedButtonShape(pValue : TLedShape);
    Procedure SetCaptType(pValue : TCaptType);
    Procedure SetLedOn(pValue : Boolean);
    Procedure SetLedBlink(pValue : Boolean);
    Procedure SwitchLed(pValue : boolean);
    Procedure SetColLed(pValue : TColor);
    Procedure SetCaption(pValue : String10);
    Procedure SetCaption1(pValue : String10);
    Procedure DispString;
    procedure UpdateExclusive;
    procedure CMButtonPressed(var Message: TMessage); message CM_BUTTONPRESSED;
    procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
```

```
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    Property ButtonStyle;
    property OnClick;
    property OnDown;
    property OnUp;
    Property ColBtnBord;
    property ParentFont;
    Property AllowAllUp : boolean    read FAllowUp   write FAllowUp;
    property GroupIndex : integer    read FGroupIdx  write SetGroupIdx;
    Property ColLed      : TColor    read FColLed    write SetColLed;
    Property LedOn       : Boolean   read FLedOn     write SetLedOn;
    Property LedBlink    : boolean   read FLedBlink  write SetLedBlink;
    property LedShape    : TLedShape read FLedShape  write SetLedButtonShape;
    Property CaptType    : TCaptType read FCaptType  write SetCaptType;
    property Caption     : String10  read FCaption   write SetCaption;
    property Caption1    : String10  read FCaption1  write SetCaption1;
    property ColCaption  : TColor    read FColCapt   write SetColCapt;
    property ColCaption1 : TColor    read FColCapt1  write SetColCapt1;
    property CaptBevel   : TPanelBevel read FCaptBevel write SetCaptBevel;
    property Down        : boolean   read isDown     write SetBtnDown;
  end;
```

SlideButton

Schiebeschalter mit beliebiger Beschriftung und Farben.

Komponenten Deklaration

```
type
  swStatus     = (swPos2, swPos1);
  TSlideButton   = class(TRahmen)
  private
    { Private-Deklarationen }
    FKeyChange : TNotifyEvent;
    painted    : boolean;
    Slide      : TRect;
    BtnSize    : Integer;
    tSizePos1  : Integer;
    tSizePos2  : Integer;
    FTextPos1  : String10;
    FTextPos2  : String10;
    FBmpPos1   : TBitMap;
    FBmpPos2   : TBitMap;
    FColPos1   : TColor;
    FColPos2   : TColor;
    FColTextPos1 : TColor;
    FColTextPos2: TColor;
    FColButton : TColor;
    FPosition  : swStatus;
    FBmpEnable : boolean;

    Procedure SetBmpEnable(pValue : boolean);
    function  GetBmpPos1: TBitmap;
    procedure SetBmpPos1(pValue: TBitmap);
    function  GetBmpPos2 : TBitmap;
    procedure SetBmpPos2(pValue: TBitmap);
    Procedure SetTextPos1(pValue : String10);
    Procedure SetTextPos2(pValue : String10);
    Procedure SetColTextPos1(pValue : TColor);
    Procedure SetColTextPos2(pValue : TColor);
    Procedure SetColorOn(pValue : TColor);
    Procedure SetColorPos2(pValue : TColor);
    Procedure SetColButton(pValue : TColor);
    Procedure SetPosition(pValue : swStatus);
    Procedure DrawPosition;
    Function  CalcText(str : string) : integer;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    Property ParentFont;
```

```
    property KeyChange  : TNotifyEvent read FKeyChange write FKeyChange;
    Property ColTextPos1: TColor   read FColTextPos1 write SetColTextPos1;
    Property ColTextPos2: TColor   read FColTextPos2 write SetColTextPos2;
    Property ColorPos1  : TColor   read FColPos1     write SetColorOn;
    Property ColorPos2  : TColor   read FColPos2     write SetColorPos2;
    Property ColButton  : TColor   read FColButton   write SetColButton;
    property TextPos1   : String10 read FTextPos1    write SetTextPos1;
    property TextPos2   : String10 read FTextPos2    write SetTextPos2;
    Property BmpPos1    : TBitMap  read GetBmpPos1   write SetBmpPos1;
    Property BmpPos2    : TBitMap  read GetBmpPos2   write SetBmpPos2;
    property Position   : swStatus read FPosition    write SetPosition;
    property BmpEnable  : boolean  read FBmpEnable   write SetBmpEnable;
  end;
```

PictButton

Leuchtfeld auch als Schalter bzw. Taste verwendbar. Mit 1, 2 oder 3 Textzeilen oder altern. BitMap. Blinkend.

Komponenten Deklaration

```
type
  (* Taster *)
  TPictButton   = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FSymbol    : TBitMap;
    painted    : Boolean;
    FFont1     : TFont;
    FFont2     : TFont;
    selfChange : boolean;
    FText      : string20;
    FText1     : string20;
    FText2     : string20;
    FPictStyle : TPictStyle;
    FBlinkOn   : Boolean;
    FColBorder : TColor;
    FGroupIdx  : integer;
    FAllowUp   : boolean;
    TextR      : TRect;
    TextR1     : TRect;
    TextR2     : TRect;
    XStartP    : integer;
    YStartP    : integer;
    XAbsP      : integer;
    YAbsP      : integer;
    XStartL    : Integer;
    YStartL    : Integer;
    XAbsL      : Integer;
    YAbsL      : Integer;
    Textx,
    TextY,
    Text1x,
    Text1Y,
    Text2x,
    Text2Y     : Integer;

    procedure SetGroupIdx(pValue : integer);
    procedure TimerExpired(Sender: TObject);
    procedure SetColBorder(pValue : TColor);
    Procedure SetBtnDown(pValue : boolean);
    procedure DrawTaste(blink : boolean);
    Procedure SetBlinkOn(pValue : Boolean);
    Procedure SetShape(pValue : TShapeTypeT);
    Procedure SetFont1(pValue : TFont);
    Procedure SetFont2(pValue : TFont);
    Procedure SetText (pValue : String20);
    Procedure SetText1(pValue : String20);
    Procedure SetText2(pValue : String20);
    Procedure DrawText (txtDraw, txtDelete, txtSize : boolean);
    Procedure DrawText1(txtDraw, txtDelete, txtSize : boolean);
    Procedure DrawText2(txtDraw, txtDelete, txtSize : boolean);
    Procedure SetPictStyle(pValue : TPictStyle);
    function  GetSymbol: TBitmap;
```

```
    procedure SetSymbol(pValue: TBitmap);
    Procedure CalcTextStil;
    procedure ScalePnt(XL,YL : Integer; var XP,YP : integer);
    procedure UpdateExclusive;
    procedure CMButtonPressed(var Message: TMessage); message CM_BUTTONPRESSED;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape     : TShapeTypeT;

    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    procedure StyleChanged(Sender : TObject);
    property Align;
    Property FontAutom;
    Property ParentFont;
    property Font;
    property ColBack;
    Property ColBtnBord;
    Property ButtonStyle;
    property OnClick;
    property OnDown;
    property OnUp;
    Property AllowAllUp : boolean    read FAllowUp   write FAllowUp;
    property GroupIndex : integer    read FGroupIdx  write SetGroupIdx;
    Property Font1      : TFont      read FFont1     write SetFont1;
    Property Font2      : TFont      read FFont2     write SetFont2;
    Property Symbol     : TBitMap    read GetSymbol  write SetSymbol;
    Property Text       : String20   read FText      write SetText;
    Property Text1      : String20   read FText1     write SetText1;
    Property Text2      : String20   read FText2     write SetText2;
    property Shape      : TShapeTypeT read FShape     write SetShape;
    property PictStyle  : TPictStyle read FPictStyle write SetPictStyle;
    Property BlinkOn    : boolean    read FBlinkOn   write SetBlinkOn;
    property Down       : boolean    read isDown     write SetBtnDown;
    property ColBorder  : TColor     read FColBorder write SetColBorder;
  end;
```

LedCheck

Schalter, als Ersatz für CheckBox und RadioButton. Led Farbe, Grösse und Erscheinungsbild sowie Textposition in weiten Grenzen wählbar.

Komponenten Deklaration

```
type
  TLedCheck   = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FLedBlink  : Boolean;
    FLedShape  : TLedShape;
    FLedCol    : TColor;
    FLedSize   : integer;
    FLedBevel  : TPanelBevel;
    FCaptFrame : Boolean;
    FAutoFrame : Boolean;
    FColCapt   : TColor;
    FColCaptFrm: TColor;
    FColCapt1  : TColor;
    FCaption   : String;
    FCaption1  : String;
    FGroupIdx  : integer;
    FCaptPos   : TOrientType;
    FCheckMode : TCheckMode;
    painted    : Boolean;
    LedRect    : TRect;
    LedPosX    : integer;
    LedPosY    : integer;
    Ledsz      : integer;
    textR      : TRect;

    procedure TimerExpired(Sender: TObject);
    procedure SetGroupIdx(pValue : integer);
    procedure SetLedSize(pValue : integer);
    procedure SetLedBevel(pValue : TPanelBevel);
    Procedure SetChecked(pValue : boolean);
    Procedure SetLedBlink(pValue : boolean);
    procedure SetCheckMode(pValue : TCheckMode);
    Procedure SetCaptPos(pValue : TOrientType);
    procedure SetLedShape(pValue : TLedShape);
    procedure SetLedCol(pValue : TColor);
    Procedure SetCaption(pValue : String);
    Procedure SetCaption1(pValue : String);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetCaptFrame(pValue : boolean);
    Procedure SetAutoFrame(pValue : boolean);
    Procedure SetColCaptFrm(pValue : TColor);
    Procedure SetColCapt1(pValue : TColor);
    procedure UpdateExclusive;
    procedure DrawLed(lighted : boolean);
    procedure DrawText(lighted : boolean);
    procedure CMButtonPressed(var Message: TMessage); message CM_BUTTONPRESSED;
    procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
    procedure WMSize(var Message: TWMSize); message WM_SIZE;
 protected
    { Protected-Deklarationen }
```

```
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    procedure Click; override;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property OnClick;
    property OnDown;
    property OnUp;
    Property ColBtnBord;
    Property ParentColor;
    Property ParentFont;
    Property Framed;
    Property Font;
    Property CheckMode  : TCheckMode  read FCheckMode write SetCheckMode;
    property LedShape   : TLedShape   read FLedShape  write SetLedShape;
    property LedBevel   : TPanelBevel read FLedBevel  write SetLedBevel;
    Property LedCol     : TColor      read FLedCol    write SetLedCol;
    Property LedBlink   : boolean     read FLedBlink  write SetLedBlink;
    Property LedSize    : integer     read FLedSize   write SetLedSize;
    property Caption    : String      read FCaption   write SetCaption;
    property Caption1   : String      read FCaption1  write SetCaption1;
    property CaptionCol : TColor      read FColCapt   write SetColCapt;
    property CaptionCol1: TColor      read FColCapt1  write SetColCapt1;
    property CaptFrmCol : TColor      read FColCaptFrm write SetColCaptFrm;
    Property CaptFrame  : boolean     read FCaptFrame write SetCaptFrame;
    Property CaptAutoFrm: boolean     read FAutoFrame write SetAutoFrame;
    property Checked    : boolean     read isDown     write SetChecked;
    property CaptPos    : TOrientType read FCaptPos   write SetCaptPos;
    property GroupIndex : integer     read FGroupIdx  write SetGroupIdx;
  end;
```

TextSpin

SpinElement mit bel. vielen Texten. Automatische String - Numerik Konvertierung. Mit/ohne Anschlag.

Komponenten Deklaration

```
type
  TTextSpin = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging: TNotifyEvent;
    FOnChanged : TNotifyEvent;
    FPTexte    : TStringList;
    RepeatTimer: integer;
    FRepSpeed  : integer;
    ButtonUp,
    ButtonDown : TRect;
    isInvBUp,
    isInvBDown : Boolean;
    textR      : TRect;
    painted    : boolean;
    FPosition  : Integer;
    FLimiter   : boolean;
    FTextStyle : TTextStyle;
    TextPosY   : integer;
    hasChanged : boolean;

    procedure TimerExpired(Sender: TObject);
    Procedure SetColDisable(pValue : TColor);
    Procedure SetTextStyle(pValue : TTextStyle);
    procedure SetPTexte(pValue: TStringList);
    Procedure SetColBack(pValue : TColor);
    Procedure SetPosition(pValue : Integer);
    Procedure SetPosLong(pValue : Integer);
    function  GetPosLong : Integer;
    Procedure SetPosColor(pValue : TColor);
    function  GetPosColor : TColor;
    function  GetPosStr : string;
    procedure SetPosStr(pValue : string);
    function  GetPosFloat : double;
    procedure SetPosFloat(pValue : double);
    Procedure DrawText;
    procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColBack    : TColor;
    FColDisable : tColor;
    function    ExistLong(pValue : Integer) : boolean;
    function    ExistFloat(pValue : double) : boolean;
    procedure   WritePosStr(pos : integer; insert : boolean; str : string);
    function    NextPosition(next : boolean) : boolean;
```

```
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    procedure StyleChanged(Sender : TObject);
    property onChanging : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged  : TNotifyEvent read FOnChanged  write FOnChanged;
    property ColTxtBack : TColor       read FColBack    write SetColBack;
    property ColDisabled: TColor       read FColDisable write SetColDisable;
    property PosText    : TStringList  read FPTexte     write SetPTexte;
    property Position   : Integer      read FPosition   write SetPosition;
    property PosInteger : Integer      read GetPosLong  write SetPosLong;
    property PosColor   : TColor       read GetPosColor write SetPosColor;
    property PosString  : string       read GetPosStr   write SetPosStr;
    property PosFloat   : double       read GetPosFloat write SetPosFloat;
    property Limiter    : boolean      read FLimiter    write FLimiter;
    property TextAdjust : TTextStyle   read FTextStyle  write SetTextStyle;
    property RepSpeed   : integer      read FRepSpeed   write FRepSpeed;
    property Align;
    Property Font;
    Property FontAutom;
    Property ParentFont;
  end;
```

CombiSpin

mit bel. vielen Texten, BitMaps, Integers und Floats

Komponenten Deklaration

```
type
  TSpinType = (tsBMP, tsBMPText);
  PSpinRec  = ^TSpinRec;
  TSpinRec  = record
                bmp    : TBitMap;
                str20  : string20;
                floatT : double;
                longI  : Integer;
              end;
  TCombiSpin  = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging: TNotifyEvent;
    FOnChanged : TNotifyEvent;
    RepeatTimer: integer;
    FRepSpeed  : integer;
    CombiList  : TList;
    ButtonUp,
    ButtonDown : TRect;
    isInvBUp,
    isInvBDown : Boolean;
    textR      : TRect;
    bmpR       : TRect;
    FBMPRatio  : double;
    painted    : boolean;
    FPosition  : Integer;
    FMaxPos    : Integer;
    FLimiter   : boolean;
    FSpinType  : TSpinType;
    hasChanged : boolean;
    TextPosY   : integer;

    procedure TimerExpired(Sender: TObject);
    function  NewRecord : PSpinRec;
    Procedure SetColBack(pValue : TColor);
    Procedure SetMaxPos(pValue : Integer);
    Procedure SetPosition(pValue : Integer);
    function  GetPosLong : Integer;
    Procedure SetPosLong(pValue : Integer);
    function  GetPosStr : string20;
    procedure SetPosStr(pValue : string20);
    function  GetPosFloat : double;
    procedure SetPosFloat(pValue : double);
    function  GetPosBMP : TBitMap;
    procedure SetPosBMP(pValue : TBitMap);
    procedure SetSpinType(pValue : TSpinType);
    procedure SetBMPRatio(pValue : double);
    Procedure DrawCombi;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
```

```
public
  { Public-Deklarationen }
  FColBack : TColor;
  function  FindPosFloat(pValue : double) : Integer;
  function  FindPosStr(pValue : string20) : Integer;
  function  FindPosLong(pValue : Integer)  : Integer;
  function  NextPosition(next : boolean) : boolean;
  constructor Create(AOwner : TComponent); override;
  destructor  Destroy; override;
published
  { Published-Deklarationen }
  procedure StyleChanged(Sender : TObject);
  property onChanging : TNotifyEvent read FOnChanging write FOnChanging;
  property onChanged  : TNotifyEvent read FOnChanged  write FOnChanged;
  property RepSpeed   : integer      read FRepSpeed   write FRepSpeed;
  property ColorBack  : TColor       read FColBack    write SetColBack;
  property Position   : Integer      read FPosition   write SetPosition;
  property MaxPos     : Integer      read FMaxPos     write SetMaxPos;
  property PosLong    : Integer      read GetPosLong  write SetPosLong;
  property PosString  : string20     read GetPosStr   write SetPosStr;
  property PosBitMap  : TBitMap      read GetPosBMP   write SetPosBMP;
  property PosFloat   : double       read GetPosFloat write SetPosFloat;
  property Limiter    : boolean      read FLimiter    write FLimiter;
  property SpinType   : TSpinType    read FSpinType   write SetSpinType;
  property BMP_Ratio  : double       read FBMPRatio   write SetBMPRatio;
  property Align;
  Property Font;
  Property FontAutom;
  Property ParentFont;
end;
```

SerieSpin

Anreihbares Spinelement mit bel. vielen Texten. Automatische String - Numerik Konvertierung. Mit/ohne Anschlag.

Komponenten Deklaration

```
type
  TSerieSpin = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging: TNotifyEvent;
    FOnChanged : TNotifyEvent;
    FPTexte    : TStringList;
    RepeatTimer: integer;
    FRepSpeed  : integer;
    ButtonUp,
    ButtonDown : TRect;
    isInvBUp,
    isInvBDown : Boolean;
    textR      : TRect;
    painted    : boolean;
    FPosition  : Integer;
    FLimiter   : boolean;
    FTextStyle : TTextStyle;
    TextPosY   : integer;
    hasChanged : boolean;

    procedure TimerExpired(Sender: TObject);
    Procedure SetTextStyle(pValue : TTextStyle);
    procedure SetPTexte(pValue: TStringList);
    Procedure SetColBack(pValue : TColor);
    Procedure SetPosition(pValue : Integer);
    Procedure SetPosLong(pValue : Integer);
    function  GetPosLong : Integer;
    Procedure SetPosColor(pValue : TColor);
    function  GetPosColor : TColor;
    function  GetPosStr : string;
    procedure SetPosStr(pValue : string);
    function  GetPosFloat : double;
    procedure SetPosFloat(pValue : double);
    Procedure DrawText;
    procedure WMSize(var Message: TWMSize); message WM_SIZE;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColBack : TColor;
    function  ExistLong(pValue : Integer) : boolean;
    function  ExistFloat(pValue : double) : boolean;
    procedure WritePosStr(pos : integer; insert : boolean; str : string);
    function  NextPosition(next : boolean) : boolean;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
```

```
    { Published-Deklarationen }
    procedure StyleChanged(Sender : TObject);
    property onChanging : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged  : TNotifyEvent read FOnChanged  write FOnChanged;
    property ColTxtBack : TColor       read FColBack    write SetColBack;
    property PosText    : TStringList  read FPTexte     write SetPTexte;
    property Position   : Integer      read FPosition   write SetPosition;
    property PosInteger : Integer      read GetPosLong  write SetPosLong;
    property PosColor   : TColor       read GetPosColor write SetPosColor;
    property PosString  : string       read GetPosStr   write SetPosStr;
    property PosFloat   : double       read GetPosFloat write SetPosFloat;
    property Limiter    : boolean      read FLimiter    write FLimiter;
    property TextAdjust : TTextStyle   read FTextStyle  write SetTextStyle;
    property RepSpeed   : integer      read FRepSpeed   write FRepSpeed;
    property Align;
    Property Font;
    Property FontAutom;
    Property ParentFont;
  end;
```

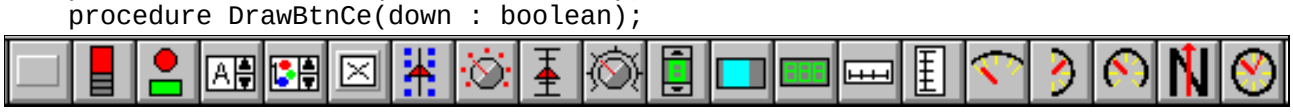
JoyStick

Graphischer Joystick mit 8 Richtungen zur Bedienung mit der Maus. Mit Rast, Tast und Repeatfunktionen. Repeatrate einstellbar.

Komponenten Deklaration

```
type
  TJoyStick = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FRepSpeed  : integer;
    painted    : boolean;
    FSymbolUp  : TBitMap;
    FSymbolDwn : TBitMap;
    fAllowAllUp  : boolean;
    fLockMode  : boolean;
    fAllowAllUpC : boolean;
    fColMain   : TColor;
    fColSek    : TColor;
    fCurrBtn   : Char;
    ButtonNo   : TRect;
    ButtonWe   : TRect;
    ButtonEa   : TRect;
    ButtonSo   : TRect;
    ButtonNE   : TRect;
    ButtonSE   : TRect;
    ButtonNW   : TRect;
    ButtonSW   : TRect;
    ButtonCe   : TRect;
    BtnNWDown  : boolean;
    BtnNoDown  : boolean;
    BtnNEDown  : boolean;
    BtnWeDown  : boolean;
    BtnCeDown  : boolean;
    BtnEaDown  : boolean;
    BtnSWDown  : boolean;
    BtnSoDown  : boolean;
    BtnSEDown  : boolean;
    fBtnStatus : word;
    pfw1, pfw2,
    pfw3       : integer;

    procedure TimerExpired(Sender: TObject);
    function  ButtonsUp : boolean;
    function  GetSymbolUp: TBitmap;
    procedure SetSymbolUp(pValue: TBitmap);
    function  GetSymbolDwn: TBitmap;
    procedure SetSymbolDwn(pValue: TBitmap);
    procedure SetColMain(pValue : TColor);
    procedure SetColSek(pValue : TColor);
    procedure SetBtnStatus(pValue : word);
  protected
    { Protected-Deklarationen }
    procedure DrawBtnNW(down : boolean);
    procedure DrawBtnNo(down : boolean);
    procedure DrawBtnNE(down : boolean);
    procedure DrawBtnWe(down : boolean);
    procedure DrawBtnCe(down : boolean);
```



```
    procedure DrawBtnEa(down : boolean);
    procedure DrawBtnSW(down : boolean);
    procedure DrawBtnSo(down : boolean);
    procedure DrawBtnSE(down : boolean);
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    isRepeating : boolean;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property AllowAllUp  : boolean read fAllowAllUp write fAllowAllUp;
    property LockMode    : boolean read fLockMode  write fLockMode;
    property RastCenter  : boolean read fAllowAllUpC write fAllowAllUpC;
    Property SymbolUp    : TBitMap read GetSymbolUp  write SetSymbolUp;
    Property SymbolDown  : TBitMap read GetSymbolDwn write SetSymbolDwn;
    property ColorMain   : TColor  read FColMain   write SetColMain;
    property ColorSek    : TColor  read FColSek    write SetColSek;
    property ButtonStatus: word    read FBtnStatus write SetBtnStatus;
    property Button      : Char    read FCurrBtn   write FCurrBtn;
    property RepSpeed    : integer read FRepSpeed  write FRepSpeed;
    property OnDown;
    property OnUp;
  end;
```

UpDownT

Zum Joystick passende Doppeltaste mit Rast, Tast und Repeatfunktion

Komponenten Deklaration

```
type
  TUpDownT = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FRepSpeed  : integer;
    painted    : boolean;
    fAllowAllUp : boolean;
    fLockMode  : boolean;
    FBtnStatus : word;
    fColUp     : TColor;
    fColDown   : TColor;
    fCurrBtn   : Char;
    ButtonUp   : TRect;
    ButtonDwn  : TRect;
    BtnUpDown  : boolean;
    BtnDwnDown : boolean;
    pfw1, pfw2,
    pfw3       : integer;

    procedure TimerExpired(Sender: TObject);
    function  ButtonsUp : boolean;
    procedure SetColUp(pValue : TColor);
    procedure SetColDown(pValue : TColor);
    procedure SetBtnStatus(pValue : word);
  protected
    { Protected-Deklarationen }
    procedure DrawBtnUp(down : boolean);
    procedure DrawBtnDwn(down : boolean);
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    isRepeating : boolean;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property LockMode    : boolean read fLockMode  write fLockMode;
    property AllowAllUp  : boolean read fAllowAllUp write fAllowAllUp;
    property ColorUp     : TColor  read FColUp     write SetColUp;
    property ColorDown   : TColor  read FColDown   write SetColDown;
    property ButtonStatus: word    read FBtnStatus write SetBtnStatus;
    property Button      : Char    read FCurrBtn   write FCurrBtn;
    property RepSpeed    : integer read FRepSpeed write FRepSpeed;
    property OnDown;
    property OnUp;
  end;
```

LeftRightT

Zum Joystick passende Doppeltaste mit Rast, Tast und Repeatfunktion

Komponenten Deklaration

```
type
  TLeftRightT = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    FRepSpeed  : integer;
    painted    : boolean;
    fAllowAllUp : boolean;
    fLockMode  : boolean;
    FBtnStatus : word;
    fColLeft   : TColor;
    fColRight  : TColor;
    fCurrBtn   : Char;
    ButtonLeft : TRect;
    ButtonRight: TRect;
    BtnLeftDwn : boolean;
    BtnRightDwn: boolean;
    pfw1, pfw2,
    pfw3       : integer;

    procedure TimerExpired(Sender: TObject);
    function  ButtonsUp : boolean;
    procedure SetColLeft(pValue : TColor);
    procedure SetColRight(pValue : TColor);
    procedure SetBtnStatus(pValue : word);
  protected
    { Protected-Deklarationen }
    procedure DrawBtnLeft(down : boolean);
    procedure DrawBtnRight(down : boolean);
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    isRepeating : boolean;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property LockMode    : boolean read fLockMode  write fLockMode;
    property AllowAllUp  : boolean read fAllowAllUp write fAllowAllUp;
    property ColorLeft   : TColor  read FColLeft   write SetColLeft;
    property ColorRight  : TColor  read FColRight  write SetColRight;
    property ButtonStatus: word    read FBtnStatus write SetBtnStatus;
    property Button      : Char    read FCurrBtn   write FCurrBtn;
    property RepSpeed    : integer read FRepSpeed write FRepSpeed;
    property OnDown;
    property OnUp;
  end;
```

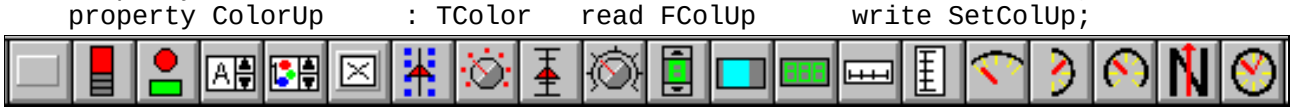
ArrowBtnT

Pfeiltaste mit 4 Richtungen, mehreren Formen. Rast, Tast und Repeatfunktion

Komponenten Deklaration

```
type
  TArrowBtn = class(TThalesBase)
  private
    { Private-Deklarationen }
    BlinkTimer  : integer;
    RepeatTimer : integer;
    FRepSpeed   : integer;
    fBtnMode    : TBtnMode;
    fDown       : boolean;
    wasDown     : boolean;
    FColUp      : TColor;
    FColDown    : TColor;
    FColBtn     : TColor;
    fOrient     : TOrientType;
    FGroupIdx   : integer;
    fShape      : TArrBtnShape;
    fShapeBlink : boolean;
    fShapeOn    : boolean;
    BlinkOn     : boolean;
    ButtonRect  : TRect;
    painted     : boolean;

    procedure TimerExpired(Sender: TObject);
    procedure BlinkExpired(Sender: TObject);
    procedure SetColUp(pValue : TColor);
    procedure SetColDown(pValue : TColor);
    procedure SetColBtn(pValue : TColor);
    procedure SetGroupIdx(pValue : integer);
    Procedure SetBtnDown(pValue : boolean);
    Procedure SetShape(pValue : TArrBtnShape);
    Procedure SetShapeBlink(pValue : boolean);
    Procedure SetShapeOn(pValue : boolean);
    procedure SetOrient(pValue : TOrientType);
    Procedure DrawBtn;
    procedure UpdateExclusive;
    procedure CMButtonPressed(var Message: TMessage); message CM_BUTTONPRESSED;
    procedure CMEnabledChanged(var Message: TMessage); message
CM_ENABLEDCHANGED;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    isRepeating : boolean;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    Property ButtonMode  : TBtnMode read fBtnMode    write fBtnMode;
    property ColorUp     : TColor   read FColUp      write SetColUp;
```



```
    property ColorDown   : TColor   read FColDown    write SetColDown;
    property ColorButton : TColor   read FColBtn     write SetColBtn;
    property Down        : boolean  read fDown       write SetBtnDown;
    property RepSpeed    : integer  read FRepSpeed   write FRepSpeed;
    property GroupIndex  : integer  read FGroupIdx   write SetGroupIdx;
    property Shape   : TArrBtnShape read fShape      write SetShape;
    property ShapeBlink  : boolean  read fShapeBlink write SetShapeBlink;
    property ShapeOn     : boolean  read fShapeOn    write SetShapeOn;
    property Orientation : TOrientType read fOrient  write SetOrient;
    property OnDown;
    property OnUp;
    property OnClick;
  end;
```

Slider

Schieberegler oder Schalter mit/ohne Raststellungen, beliebige Anfangs- und Endwerten. Horizontal und vertikal. Erscheinungsbild in weiten Grenzen änderbar: Farben, Ränder, Knopf etc.

Komponenten Deklaration

```
type
  TTrenchStyle = TPanelBevel;
  tBtnStyle    = TButtonStyle;
  TButtonDir   = (btCrossWise, btLengthWise, btSquare);

type
  TSlider = class(TThalesBase)
  private
    FSelector    : Boolean;
    FCaption     : string;
    FCaptEna     : Boolean;
    FCaptOffset  : Integer;
    FCaptColor   : tColor;
    FAlignment   : tAlignment;
    FValueMax,
    FValueMin,
    FValue       : double;
    FOrientation : TOrientation;
    FTrenchStyle : TTrenchStyle;
    FTrenchWidth : Integer;
    FTrenchCol   : TColor;
    FButtonCol   : TColor;
    FButtonDir   : TButtonDir;
    FBtnStyle    : TBtnStyle;
    FBtnSize     : Integer;
    FScalaShow   : Boolean;
    FScalaPart   : Integer;
    FScalaCol    : TColor;
    FPosition    : Integer;
    FDragDelay   : Integer;
    FOnChanging  : TNotifyEvent;
    FOnChanged   : TNotifyEvent;
    RepeatTimer  : integer;

    painted      : Boolean;
    ButtonBmp    : TBitMap;
    BtnOffs      : integer;
    XStartP      : integer;
    YStartP      : integer;
    XAbsP        : integer;
    YAbsP        : integer;
    XStartL      : double;
    YStartL      : double;
    XAbsL        : double;
    YAbsL        : double;
    BtnRect      : TRect;
    CaptRect     : TRect;
    TempDC       : HDC;
    OrgCursor    : HCursor;
    changed      : Boolean;
    ValueSoll    : double;
```

```
    procedure SetAlignment(pValue: tAlignment);
    procedure SetCaptColor(pValue : tColor);
    procedure SetCaptEna(pValue: boolean);
    procedure SetCaption(pValue: string);
    procedure SetCaptOffset(pValue: integer);
    procedure SetSelector(pValue: Boolean);
    procedure SetValueMax(pValue: double);
    procedure SetValueMin(pValue: double);
    procedure SetOrientation(pValue: TOrientation);
    procedure SetValue(pValue: double);
    procedure SetTrenchStyle(pValue: TTrenchStyle);
    procedure SetTrenchWidth(pValue: Integer);
    procedure SetDragDelay(pValue: Integer);
    procedure SetButtonDir(pValue: TButtonDir);
    procedure SetBtnStyle(pValue: TBtnStyle);
    procedure SetScalaShow(pValue: Boolean);
    procedure SetScalaPart(pValue: Integer);
    procedure SetPosition(pValue: Integer);
    procedure SetScalaCol(pValue: TColor);
    procedure SetButtonCol(pValue: TColor);
    procedure SetBtnSize(pValue: Integer);
    procedure SetTrenchCol(pValue: TColor);
    procedure TimerExpired(Sender: TObject);

    procedure WMSize(var Message: TWMSize); message WM_SIZE;
  protected
    LogRec    : TLOGFONT;
    OldFont   : HFont;
    NewFont   : HFONT;
    procedure Paint; override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    function  IsVert: Boolean;
    Procedure DrawValue(doDraw : boolean);
    Procedure DrawCaption;
    procedure DrawBtnBmp;
    procedure ScalePnt(XL,YL : double; var XP,YP : integer);
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property Enabled;
    property ParentShowHint;
    property ShowHint;
    property Visible;

    property CaptEna    : boolean   read FCaptEna     write SetCaptEna;
    property Caption    : string    read FCaption     write SetCaption;
    property CaptColor  : tColor    read fCaptColor   write SetCaptColor;
    property CaptOffset : integer   read FCaptOffset  write SetCaptOffset;
    property Alignment  : tAlignment read FAlignment  write SetAlignment;
    property Selector   : Boolean   read FSelector    write SetSelector;
    property ValueMax   : double    read FValueMax    write SetValueMax;
    property ValueMin   : double    read FValueMin    write SetValueMin;
    property Value      : double    read FValue       write SetValue;
    property TrenchWidth: Integer   read FTrenchWidth write SetTrenchWidth;
    property Position   : Integer   read FPosition    write SetPosition;
    property DragDelay  : Integer   read FDragDelay   write SetDragDelay;
    property ButtonDir : TButtonDir read FButtonDir   write SetButtonDir;
```

```
    property BtnStyle   : TBtnStyle read FBtnStyle    write SetBtnStyle;
    property ButtonSize : Integer   read FBtnSize     write SetBtnSize;
    property ScalaShow  : Boolean   read FScalaShow   write SetScalaShow;
    property ScalaPart  : Integer   read FScalaPart   write SetScalaPart;
    property ScalaColor : TColor    read FScalaCol    write SetScalaCol;
    property ButtonColor: TColor    read FButtonCol   write SetButtonCol;
    property TrenchColor: TColor    read FTrenchCol   write SetTrenchCol;
    property TrenchStyle: TTrenchStyle read FTrenchStyle write SetTrenchStyle;
    property Orientation: TOrientation read FOrientation write SetOrientation;
    property OnChanging : TNotifyEvent read FOnChanging  write FOnChanging;
    property OnChanged  : TNotifyEvent read FOnChanged   write FOnChanged;
  end;
```

SelectSlide

Schiebeschalter mit bis zu 20 Raststellungen. Frei beschriftbar.

Komponenten Deklaration

```
  (* Selector Linear *)
  TSelectorSlide  = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging : TNotifyEvent;
    FOnChanged  : TNotifyEvent;
    FOnPreset   : TNotifyEvent;
    FPresEnable : Boolean;
    FPTexte     : TStringList;
    FCaptEnable : Boolean;
    FColCaption : TColor;
    FColPreset  : TColor;
    FCaptBevel  : TPanelBevel;
    changed     : boolean;
    preset      : boolean;
    ButtonBmp   : TBitMap;
    SaveBmp     : TBitMap;
    painted     : Boolean;
    PresRect    : TRect;
    TextPos     : TRect;
    MouseX,
    MouseY      : Integer;
    XStartP     : integer;
    YStartP     : integer;
    XAbsP       : integer;
    YAbsP       : integer;
    Pointer     : TRect;
    PointerSize : Integer;

    Procedure SetColPreset(pValue : TColor);
    Procedure SetPosColor(pValue : TColor);
    function  GetPosColor : TColor;
    function  GetPosStr : string;
    procedure SetPosStr(pValue : string);
    Procedure SetPosLong(pValue : integer);
    function  GetPosLong : integer;
    function  GetPosFloat : double;
    procedure SetPosFloat(pValue : double);
    Procedure SetCaptBevel(pValue : TPanelBevel);
    Procedure SetColCaption(pValue : TColor);
    procedure SetPTexte(pValue: TStringList);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColButton(pValue : TColor);
    Procedure SetPosition(pValue : integer);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string10);
    Procedure SetCaptEnable(pValue : Boolean);
    Procedure SetPresEnable(pValue : Boolean);

    Procedure DrawPosition;
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
```

```
  public
    { Public-Deklarationen }
    FColScala  : TColor;
    FColButton : TColor;
    FCaption  : String10;
    FScalaPart : integer;
    FPosition  : Integer;

    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    property onChanging : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged  : TNotifyEvent read FOnChanged  write FOnChanged;
    property onPreset   : TNotifyEvent read FOnPreset   write FOnPreset;
    property PosText  : TStringList   read FPTexte     write SetPTexte;
    property ColScala   : TColor      read FColScala   write SetColScala;
    property ColButton  : TColor      read FColButton  write SetColButton;
    property Position   : integer     read FPosition   write SetPosition;
    property ScalaPart  : integer     read FScalaPart  write SetScalaPart;
    property Caption    : string10    read FCaption    write SetCaption;
    property CaptEnable : Boolean     read FCaptEnable write SetCaptEnable;
    property ColCaption : TColor      read FColCaption write SetColCaption;
    property ColPreset  : TColor      read FColPreset  write SetColPreset;
    property CaptBevel  : TPanelBevel read FCaptBevel  write SetCaptBevel;
    property PosColor   : TColor      read GetPosColor write SetPosColor;
    property Posinteger : integer     read GetPosLong  write SetPosLong;
    property PosFloat   : double      read GetPosFloat write SetPosFloat;
    property PosString  : string      read GetPosStr   write SetPosStr;
    property PresEnable : Boolean     read FPresEnable write SetPresEnable;
  end;
```

Selector270

Drehschalter mit bis zu 20 Raststellungen. Frei beschriftbar.

Komponenten Deklaration

```
  (* Selector 270 *)
  TSelector270 = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging : TNotifyEvent;
    FOnChanged  : TNotifyEvent;
    FOnPreset   : TNotifyEvent;
    FPresEnable : Boolean;
    FBtnSmall   : Boolean;
    FPosition   : Integer;
    FCaptEnable : Boolean;
    FColCaption : TColor;
    FColPreset  : TColor;
    FCaptBevel  : TPanelBevel;
    changed     : boolean;
    MittelP,
    PointerP,
    PointerP0   : TPoint;
    painted     : Boolean;
    PresRect    : TRect;
    TextPos     : TRect;
    MouseX,
    MouseY      : Integer;
    FPTexte     : TStringList;
    preset      : boolean;

    Procedure SetColPreset(pValue : TColor);
    Procedure SetPosColor(pValue : TColor);
    function  GetPosColor : TColor;
    function  GetPosStr : string;
    procedure SetPosStr(pValue : string);
    Procedure SetPosLong(pValue : integer);
    function  GetPosLong : integer;
    function  GetPosFloat : double;
    procedure SetPosFloat(pValue : double);
    Procedure SetCaptBevel(pValue : TPanelBevel);
    Procedure SetColCaption(pValue : TColor);
    Procedure SetCaptEnable(pValue : Boolean);
    Procedure SetPresEnable(pValue : Boolean);
    Procedure SetBtnSmall(pValue : Boolean);
    procedure SetPTexte(pValue: TStringList);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColButton(pValue : TColor);
    Procedure SetColLight(pValue : TColor);
    Procedure SetColShade(pValue : TColor);
    Procedure SetPosition(pValue : Integer);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string10);
    Procedure DrawPosition;
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
```

```
  public
    { Public-Deklarationen }
    FColPointer : TColor;
    FColScala  : TColor;
    FColButton : TColor;
    FColLight  : TColor;
    FColShade  : TColor;
    FCaption  : String10;
    FScalaPart : integer;

    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    property onChanging : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged  : TNotifyEvent read FOnChanged  write FOnChanged;
    property onPreset   : TNotifyEvent read FOnPreset   write FOnPreset;
    property PosText : TStringList  read FPTexte     write SetPTexte;
    property ColPointer  : TColor   read FColPointer write SetColPointer;
    property ColScala    : TColor   read FColScala   write SetColScala;
    property ColButton   : TColor   read FColButton  write SetColButton;
    property ColLight    : TColor   read FColLight   write SetColLight;
    property ColShade    : TColor   read FColShade   write SetColShade;
    property ColCaption  : TColor   read FColCaption write SetColCaption;
    property ColPreset   : TColor   read FColPreset  write SetColPreset;
    property ScalaPart   : integer  read FScalaPart  write SetScalaPart;
    property Position    : integer  read FPosition   write SetPosition;
    property Caption     : string10 read FCaption    write SetCaption;
    property ButtonSmall : Boolean  read FBtnSmall   write SetBtnSmall;
    property CaptEnable  : Boolean  read FCaptEnable write SetCaptEnable;
    property CaptBevel: TPanelBevel read FCaptBevel  write SetCaptBevel;
    property PresEnable  : Boolean  read FPresEnable write SetPresEnable;
    property PosColor    : TColor   read GetPosColor write SetPosColor;
    property Posinteger  : integer  read GetPosLong  write SetPosLong;
    property PosFloat    : double   read GetPosFloat write SetPosFloat;
    property PosString   : string   read GetPosStr   write SetPosStr;
  end;
```

PotiSlide

SchiebePoti mit frei wählbaren Anfangs- und Endwerten, sowie Skalenteilung. Frei beschriftbar. Umschaltbar linear/logarithmisch.

Komponenten Deklaration

```
(* Poti Linear *)
TPotiSlide  = class(TRahmen)
private
  { Private-Deklarationen }
  FOnChanging : TNotifyEvent;
  FOnChanged  : TNotifyEvent;
  FOnPreset   : TNotifyEvent;
  FPresEnable : Boolean;
  FCaptEnable : Boolean;
  FColCaption : TColor;
  FColPreset  : TColor;
  FCaptBevel  : TPanelBevel;
  FValue      : double;
  FDragDelay  : Integer;
  RepeatTimer : integer;
  ValueSoll   : double;

  changed     : boolean;
  ButtonBmp   : TBitMap;
  SaveBmp     : TBitMap;
  painted     : Boolean;
  LogStart    : double;
  LogEnd      : double;
  TextPos     : TRect;
  PresRect    : TRect;
  XStartP     : integer;
  YStartP     : integer;
  XAbsP       : integer;
  YAbsP       : integer;
  XStartL     : integer;
  YStartL     : integer;
  XAbsL       : integer;
  YAbsL       : integer;
  Pointer     : TRect;
  PointerSize : Integer;
  preset      : boolean;

  procedure TimerExpired(Sender: TObject);
  procedure SetDragDelay(pValue: Integer);
  Procedure SetColPreset(pValue : TColor);
  Procedure SetCaptBevel(pValue : TPanelBevel);
  Procedure SetColCaption(pValue : TColor);
  Procedure SetColScala(pValue : TColor);
  Procedure SetColButton(pValue : TColor);
  Procedure SetValue(pValue : double);
  Procedure SetValueStart(pValue : double);
  Procedure SetValueEnd(pValue : double);
  Procedure SetScalaPart(pValue : integer);
  Procedure SetCaption(pValue : string10);
  Procedure SetCaptEnable(pValue : Boolean);
  Procedure SetPresEnable(pValue : Boolean);
```

```
    Procedure SetNonLinear(pValue : Boolean);
    Procedure DrawValue;
    procedure ScalePnt(XL,YL : integer; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColScala  : TColor;
    FColButton : TColor;
    FCaption  : String10;
    FScalaPart : integer;
    FValueStart: double;
    FValueEnd  : double;
    FNonLinear : Boolean;

    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    property onChanging : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged  : TNotifyEvent read FOnChanged  write FOnChanged;
    property onPreset   : TNotifyEvent read FOnPreset   write FOnPreset;
    property ColScala    : TColor   read FColScala   write SetColScala;
    property ColButton   : TColor   read FColButton  write SetColButton;
    property ColCaption  : TColor   read FColCaption write SetColCaption;
    property ColPreset   : TColor   read FColPreset  write SetColPreset;
    property Value       : double   read FValue      write SetValue;
    property ValueStart  : double   read FValueStart write SetValueStart;
    property ValueEnd    : double   read FValueEnd   write SetValueEnd;
    property ScalaPart   : integer  read FScalaPart  write SetScalaPart;
    property Caption     : string10 read FCaption    write SetCaption;
    property CaptEnable  : Boolean  read FCaptEnable write SetCaptEnable;
    property CaptBevel : TPanelBevel read FCaptBevel write SetCaptBevel;
    property PresEnable  : Boolean  read FPresEnable write SetPresEnable;
    Property NonLinear   : Boolean  read FNonLinear  write SetNonLinear;
    property DragDelay   : Integer  read FDragDelay  write SetDragDelay;
  end;
```

Poti270

Dreh-Poti mit frei wählbaren Anfangs- und Endwerten, sowie Skalenteilung. Frei beschriftbar.

Komponenten Deklaration

```
type
  (* Poti 270 *)
  TPoti270     = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging : TNotifyEvent;
    FOnChanged  : TNotifyEvent;
    FOnPreset   : TNotifyEvent;
    FBtnSmall   : Boolean;
    FValue      : double;
    FCaptEnable : Boolean;
    FPresEnable : Boolean;
    FColCaption : TColor;
    FColPreset  : TColor;
    FCaptBevel  : TPanelBevel;
    FDragDelay  : Integer;
    RepeatTimer : integer;
    ValueSoll   : double;

    MittelP,
    PointerP,
    PointerP0   : TPoint;
    painted     : Boolean;
    TextPos     : TRect;
    PresRect    : TRect;
    MouseX,
    MouseY      : Integer;
    changed     : boolean;
    preset      : boolean;

    procedure TimerExpired(Sender: TObject);
    procedure SetDragDelay(pValue: Integer);
    Procedure SetCaptBevel(pValue : TPanelBevel);
    Procedure SetColPreset(pValue : TColor);
    Procedure SetColCaption(pValue : TColor);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetBtnSmall(pValue : Boolean);
    Procedure SetCaptEnable(pValue : Boolean);
    Procedure SetPresEnable(pValue : Boolean);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColButton(pValue : TColor);
    Procedure SetColLight(pValue : TColor);
    Procedure SetColShade(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string10);
    Procedure DrawValue;
  protected
    { Protected-Deklarationen }
```

```
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColPointer : TColor;
    FColScala  : TColor;
    FColButton : TColor;
    FColLight  : TColor;
    FColShade  : TColor;
    FCaption   : String10;
    FScalaPart : integer;
    FValueStart: double;
    FValueEnd  : double;

    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    property onChanging  : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged   : TNotifyEvent read FOnChanged  write FOnChanged;
    property onPreset    : TNotifyEvent read FOnPreset   write FOnPreset;
    property ColPointer  : TColor   read FColPointer write SetColPointer;
    property ColCaption  : TColor   read FColCaption write SetColCaption;
    property ColPreset   : TColor   read FColPreset  write SetColPreset;
    property ColScala    : TColor   read FColScala   write SetColScala;
    property ColButton   : TColor   read FColButton  write SetColButton;
    property ColLight    : TColor   read FColLight   write SetColLight;
    property ColShade    : TColor   read FColShade   write SetColShade;
    property Value       : double   read FValue      write SetValue;
    property ValueStart  : double   read FValueStart write SetValueStart;
    property ValueEnd    : double   read FValueEnd   write SetValueEnd;
    property ScalaPart   : integer  read FScalaPart  write SetScalaPart;
    property Caption     : string10 read FCaption    write SetCaption;
    property CaptEnable  : Boolean  read FCaptEnable write SetCaptEnable;
    property CaptBevel : TPanelBevel read FCaptBevel write SetCaptBevel;
    property PresEnable  : Boolean  read FPresEnable write SetPresEnable;
    property ButtonSmall : Boolean  read FBtnSmall   write SetBtnSmall;
    property DragDelay   : Integer  read FDragDelay  write SetDragDelay;
  end;
```

AlphaBar

Dient als einfacher Tastaturersatz für tastaturlose Systeme oder Auswahl Element für Sonderzeichen usw.

Komponenten Deklaration

```
type
  TAlphaBar = class(TCustomPanel)
  private
    { Private-Deklarationen }
    fActiveButton : char;
    fAlphaButtons : tStringList;
    fButtonFont : tFont;
    fBlankXSize, fBlankYSize, fButtonHeight, fButtonLeftMargin,
fButtonTopMargin,
      fButtonWidth : word;
    fButtonXSpacing, fButtonYSpacing : integer;
    fAllowAllUp, fRastButtons : boolean;
    firstButton : tSpeedButton;
    fOnValueChange : tNotifyEvent;
    procedure HandleButton(Sender :tObject);
    procedure SetActiveButton(value :char);
    procedure SetAllowAllUp(value :boolean);
    procedure SetAlphaButtons(value :tStringList);
    procedure SetBlankXSize(value :word);
    procedure SetBlankYSize(value :word);
    procedure SetButtonFont(value :tFont);
    procedure SetButtonHeight(value :word);
    procedure SetButtonLeftMargin(value :word);
    procedure SetButtonTopMargin(value :word);
    procedure SetButtonWidth(value :word);
    procedure SetRastStates;
    procedure SetRastButtons(value :boolean);
    procedure SetButtonXSpacing(value :integer);
    procedure SetButtonYSpacing(value :integer);
  protected
    { Protected-Deklarationen }
    procedure AddLetterButtons; virtual;
    procedure DestroyButtons;
    procedure Loaded; override;
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
    function GetButton(value :char) :tSpeedButton;
  published
    { Published-Deklarationen }
    property Align;
    property BevelInner;
    property BevelOuter;
    property BorderStyle;
    property BevelWidth;
    property Color;
    property Ctl3D;
    property Cursor;
    property Enabled;
    property Height default 22;
```

```
    property Locked;
    property ParentColor;
    property ParentCtl3D;
    property ParentShowHint;
    property PopupMenu;
    property ShowHint;
    property Width default 448;
    property Visible;
    property ActiveButton :char read fActiveButton write SetActiveButton default
#0;
    property AllowAllUp :boolean read fAllowAllUp write SetAllowAllUp default
false;
    property AlphaButtons :tStringList read fAlphaButtons write SetAlphaButtons;
    property BlankXSize :word read fBlankXSize write SetBlankXSize default 9;
    property BlankYSize :word read fBlankYSize write SetBlankYSize default 9;
    property ButtonFont :tFont read fButtonFont write SetButtonFont;
    property ButtonHeight :word read fButtonHeight write SetButtonHeight default
18;
    property ButtonLeftMargin :word read fButtonLeftMargin write
SetButtonLeftMargin default 2;
    property ButtonTopMargin :word read fButtonTopMargin write
SetButtonTopMargin default 2;
    property ButtonWidth :word read fButtonWidth write SetButtonWidth default
18;
    property ButtonXSpacing :integer read fButtonXSpacing write
SetButtonXSpacing default -1;
    property ButtonYSpacing :integer read fButtonYSpacing write
SetButtonYSpacing default -1;
    property RastButtons :boolean read fRastButtons write SetRastButtons;
    property OnValueChange :tNotifyEvent read fOnValueChange write
fOnValueChange;
  end;
```

FiFo

Text Ein/Ausgabefeld mit FiFo Funktion. Bei Ausgabe erfolgt ein Durchschieben des Textes von rechts nach links (Laufschrift). Bei der Eingabe erfolgt ebenfalls ein Schieben von rechts nach links. 255 Zeichen Buffer

Komponenten Deklaration

```
type
  TFiFo = class(TRahmen)
  private
    { Private-Deklarationen }
    fText      : String;
    FReadOnly  : boolean;
    FOnChange  : TNotifyEvent;
    textR      : TRect;
    painted    : boolean;
    isExit     : boolean;
    tx, ty, tw : integer;
    Procedure SetText(pValue : string);
    Procedure SetChar(pValue : Char);
    Function  GetChar : Char;
    Procedure SetColBack(pValue : TColor);
    Procedure SetTextBevel(pValue : TPanelBevel);
    Procedure DrawText;
  protected
    { Protected-Deklarationen }
    Procedure Click;  override;
    Procedure DoEnter;  override;
    Procedure DoExit;  override;
    procedure KeyPress(var Key: Char); override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColBack : TColor;
    FTextBevel : TPanelBevel;
    constructor Create(AOwner : TComponent); override;
  published
    { Published-Deklarationen }
    property Text        : String    read fText      write SetText;
    property TextChar    : Char      read GetChar    write SetChar;
    property ReadOnly    : boolean   read FReadOnly  write FReadOnly;
    property ColTxtBack  : TColor    read FColBack   write SetColBack;
    property TextBevel : TPanelBevel read FTextBevel write SetTextBevel;
    property OnChange  : TNotifyEvent read FOnChange  write FOnChange;
    Property Font;
    Property OnEnter;
    Property OnExit;
    Property TabStop;
    Property TabOrder;
  end;
```

NumEdit

Numerisches Ein/Ausgabefeld. Einsetzbar als Digitales Instrument, Daten-Anzeige und Eingabefeld. Bei der numerischen Eingabe ist ein Min und ein Max-Wert einstellbar. Ebenfalls die Anzahl der Vor- und Nachkommastellen als auch Integerformat. Text, Int oder Floatwerte vorgebbar und rücklesbar. Eingabeüberwachung.

Komponenten Deklaration

```
type
  TNumEdit = class(TRahmen)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    fText      : String;
    fUnit      : String;
    WorkStr    : String;
    FReadOnly  : boolean;
    FLedBlink  : Boolean;
    FLedOn     : Boolean;
    FLedEnable : Boolean;
    FColLed    : TColor;
    FInvert    : boolean;
    FOnChange  : TNotifyEvent;
    fDigits    : integer;
    fPrecision : integer;
    fValMin    : double;
    fValMax    : double;
    fValInt    : longint;
    fValFloat  : double;
    fErrorExit : boolean;
    fErrorShow : boolean;
    textR      : TRect;
    LedRect    : TRect;
    Position   : integer;
    maxPos     : integer;
    TextPosX   : integer;
    TextPosY   : integer;
    painted    : boolean;
    painting   : boolean;
    isExit     : boolean;
    isEnter    : boolean;

    procedure TimerExpired(Sender: TObject);
    Procedure SetValInt(pValue : longint);
    Procedure SetValFloat(pValue : double);
    Procedure SetDigits(pValue : integer);
    Procedure SetPrecision(pValue : integer);
    Procedure SetValMax(pValue : double);
    Procedure SetValMin(pValue : double);
    Procedure SetText(pValue : string);
    Procedure SetUnit(pValue : string);
    Procedure SetColBack(pValue : TColor);
    Procedure SetTextBevel(pValue : TPanelBevel);
    procedure MoveCursor(x : integer);
    Procedure DrawText(first : boolean);
    Procedure SetLedOn(pValue : Boolean);
    Procedure SetLedBlink(pValue : Boolean);
    Procedure SetLedEnable(pValue : Boolean);
    Procedure SwitchLed(pValue : boolean);
```

```
    Procedure SetColLed(pValue : TColor);
    Procedure SetInvert(pValue : boolean);
    Procedure DoChange;
  protected
    { Protected-Deklarationen }
    procedure KeyPress(var Key: Char); override;
    Procedure DoEnter;  override;
    Procedure DoExit;  override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColBack : TColor;
    FTextBevel : TPanelBevel;
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    Property LedCol      : TColor     read FColLed    write SetColLed;
    Property LedOn       : Boolean    read FLedOn     write SetLedOn;
    Property LedEnable   : Boolean    read FLedEnable write SetLedEnable;
    Property LedBlink    : boolean    read FLedBlink  write SetLedBlink;
    Property DispInvert  : boolean    read FInvert    write SetInvert;
    property ValueText   : String     read fText      write SetText;
    property UnitText    : String     read fUnit      write SetUnit;
    property ValueInt    : longint    read fValInt    write SetValInt;
    property ValueFloat  : double     read fValFloat  write SetValFloat;
    property Digits      : Integer    read fDigits    write SetDigits;
    property Precision   : Integer    read fPrecision write SetPrecision;
    property ValueMin    : double     read fValMin    write SetValMin;
    property ValueMax    : double     read fValMax    write SetValMax;
    property ReadOnly    : boolean    read FReadOnly  write FReadOnly;
    property ErrorExit   : boolean    read fErrorExit write fErrorExit;
    property ErrorShow   : boolean    read fErrorShow write fErrorShow;
    property ColTxtBack  : TColor     read FColBack   write SetColBack;
    property TextBevel : TPanelBevel  read FTextBevel write SetTextBevel;
    property OnChange  : TNotifyEvent read FOnChange  write FOnChange;
    property OnDblClick;
    Property Font;
    Property FontAutom;
    Property OnEnter;
    Property OnExit;
    Property TabStop;
    Property TabOrder;
  end;
```

Spin7Seg

Daumenradschalter mit 7-Segment Anzeige. Anzeige umschaltbar. Vorzeichen, Hex, dezimal, alphanumerisch

Komponenten Deklaration

```
  TSpin7Seg  = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnChanging: TNotifyEvent;
    FOnChanged : TNotifyEvent;
    FOnWrapCarry  : TNotifyEvent;
    FOnWrapBorrow : TNotifyEvent;
    RepeatTimer: integer;
    FRepSpeed  : integer;
    Spin7      : TDigit;
    FValueChr  : char;
    FValueInt  : Integer;
    FLimiter   : boolean;
    FItalic    : boolean;
    ButtonUp,
    ButtonDown : TRect;
    isInvBUp,
    isInvBDown : Boolean;
    hasChanged : Boolean;

    procedure TimerExpired(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetDispTyp(pValue : TDispType);
    Procedure SetColSegm(pValue : TColor);
    Procedure SetValueChr(pValue : char);
    Procedure SetValueInt(pValue : Integer);
    Procedure SetItalic(pValue : boolean);
    Procedure DispDigit;
  protected
    { Protected-Deklarationen }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FDispTyp   : TDispType;
    FShape     : TmShapeType;
    FColSegm   : TColor;

    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    function NextValue(upw : boolean; var Wrap : integer) : boolean;
    procedure StyleChanged(Sender : TObject);
    property Align;
    property ColBack;
    property onChanged : TNotifyEvent read FOnChanged  write FOnChanged;
    property onChanging: TNotifyEvent read FOnChanging write FOnChanging;
    property onWrapCarry : TNotifyEvent read FonWrapCarry  write FonWrapCarry;
```

```
    property onWrapBorrow: TNotifyEvent read FonWrapBorrow write FonWrapBorrow;
    property DispTyp  : TDispType    read FDispTyp  write SetDispTyp;
    property Shape    : TmShapeType  read FShape    write SetShape;
    property ColSegm  : TColor       read FColSegm  write SetColSegm;
    property ValueChar: char         read FValueChr write SetValueChr;
    property ValueInt : Integer      read FValueInt write SetValueInt;
    property RepSpeed : integer      read FRepSpeed write FRepSpeed;
    property Limiter  : boolean      read FLimiter  write FLimiter;
    property Italic   : boolean      read FItalic   write SetItalic;
  end;
```

Disp7Seg

1 bis 8-stelliges 7-Segment Display. Alphanum. Ansteuerung. 2 abschaltbare Leds mit Blinkfunktion.dezimal, alphanumerisch

Komponenten Deklaration

```
(* 7-Segment Display *)
type
  TDisp7Seg  = class(TRahmen)
  private
    { Private-Deklarationen }
    Disp7      : TDigDisplay;
    painted    : boolean;
    RepeatTimer: integer;
    PosLed1    : TPoint;
    PosLed2    : TPoint;
    LedSize    : integer;
    PosText    : TPoint;
    TextFRect  : TRect;
    FLedOn1    : Boolean;
    FLedOn2    : Boolean;
    FLedBlink1 : Boolean;
    FLedBlink2 : Boolean;
    FValueStr  : String;
    FValueChar : Char;
    FCharPos   : Integer;
    FValueInt  : LongInt;
    FValueFloat: double;
    FItalic    : Boolean;

    procedure TimerExpired(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetDispSize(pValue : Integer);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetTextEna(pValue : Boolean);
    Procedure SetTextEinh(pValue : String10);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure SwitchLed1(pValue : boolean);
    Procedure SwitchLed2(pValue : boolean);
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    Procedure SetColSegm(pValue : TColor);
    Procedure SetValueStr(pValue : String);
    Procedure SetValueInt(pValue : LongInt);
    Procedure SetValueFloat(pValue : double);
    Procedure SetValueChar(pValue : Char);
    Procedure SetCharPos(pValue : Integer);
    Procedure DrawEinheit;
    Procedure DispString;
    Procedure DispChar;
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
```

```
    { Public-Deklarationen }
    FShape     : TmShapeType;
    FDispSize  : Integer;
    FColSegm   : TColor;
    FColLed1   : TColor;
    FColLed2   : TColor;
    FLedEna    : Boolean;
    FTextEna   : Boolean;
    FTextEinh  : String10;

    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    property OnDblClick;
    property OnClick;
    property ColBack;
    Property DispSize   : Integer  read FDispSize   write SetDispSize;
    Property ColLed1    : TColor   read FColLed1    write SetColLed1;
    Property ColLed2    : TColor   read FColLed2    write SetColLed2;
    Property LedEnable  : Boolean  read FLedEna     write SetLedEna;
    Property LedOn1     : Boolean  read FLedOn1     write SetLedOn1;
    Property LedOn2     : Boolean  read FLedOn2     write SetLedOn2;
    Property LedBlink1  : boolean  read FLedBlink1  write SetLedBlink1;
    Property LedBlink2  : boolean  read FLedBlink2  write SetLedBlink2;
    Property TextEnable : Boolean  read FTextEna    write SetTextEna;
    Property TextEinh   : String10 read FTextEinh   write SetTextEinh;
    Property CharPos    : Integer  read FCharPos    write SetCharPos;
    Property DispItalic: Boolean  read FItalic    write SetDispItalic;
    property Shape  : TmShapeType  read FShape      write SetShape;
    property ColSegm    : TColor   read FColSegm    write SetColSegm;
    property ValueInt   : LongInt  read FValueInt   write SetValueInt;
    property ValueFloat : double   read FValueFloat write SetValueFloat;
    property ValueStr   : String   read FValueStr   write SetValueStr;
    property ValueChar  : char     read FValueChar  write SetValueChar;
  end;
```

MagicEye

Spezial BarGraph mit einstellbaren Anfangs- und Endwerten. Geeignet z.B. für einfache Spektrum-Anzeige usw.

Komponenten Deklaration

```
(* Madic Eye / Gauge horizontal/vertikal *)
type
  TMagicEye = class(TRahmen)
  private
    { Private-Deklarationen }
    painted    : boolean;
    spRect     : TRect;
    FStartVal  : double;
    FEndVal    : double;
    FStartPos  : double;
    FEndPos    : double;
    FOrient    : TOrientation;
    FColBack   : TColor;
    FColEye    : TColor;
    FColEyeInv : TColor;

    Procedure SetColBack(pValue : TColor);
    Procedure SetColEye(pValue : TColor);
    Procedure SetColEyeInv(pValue : TColor);
    Procedure SetStartPos(pValue : double);
    Procedure SetEndPos(pValue : double);
    Procedure SetStartVal(pValue : double);
    Procedure SetEndVal(pValue : double);
    Procedure SetOrient(pValue : TOrientation);
    procedure DrawValue;
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
  published
    { Published-Deklarationen }
    property ColorBack  : TColor  read FColBack   write SetColBack;
    property ColorEye   : TColor  read FColEye    write SetColEye;
    property ColorEyeInv: TColor  read FColEyeInv write SetColEyeInv;
    property StartValue : double  read FStartVal  write SetStartVal;
    property EndValue   : double  read FEndVal    write SetEndVal;
    property StartPos   : double  read FStartPos  write SetStartPos;
    property EndPos     : double  read FEndPos    write SetEndPos;
    property Direction  : TOrientation read FOrient    write SetOrient;
    property Align;
  end;
```

GaugeGP

Universeller BarGraph. Steuerbar in Prozent und realen Werten. Bar- und Meter-Mode. Windows 95 Stil. Erscheinungsbild frei gestaltbar

Komponenten Deklaration

```
Type
  TGaugeGp = class(TRahmen)
  private
    DampTimer  : integer;
    FValMin    : double;
    FValMax    : double;
    FValue     : double;
    FPercent   : double;
    FColBack   : TColor;
    FColBorder : TColor;
    FSegColor  : Tcolor;
    FSegWidth  : integer;
    FSegGap    : integer;
    FOrient    : TOrientation;
    FSegmBord  : boolean;
    FMeterMode : boolean;
    FPntrPrcnt : double;
    FDamp      : Integer;
    painted    : Boolean;
    isSizing   : Boolean;
    CurPos     : integer;

    Procedure SetDamp(pValue : integer);
    procedure DampExpired(Sender: TObject);
    procedure SetOrientation(pValue : TOrientation);
    procedure SetSegmBord(pValue : boolean);
    procedure SetMeterMode(pValue : boolean);
    procedure SetPercent(pValue : double);
    procedure SetPntrPrcnt(pValue : double);
    procedure SetColBack(pValue : Tcolor);
    procedure SetColBorder(pValue : Tcolor);
    procedure SetSegColor(pValue : Tcolor);
    procedure SetSegWidth(pValue : Integer);
    procedure SetSegGap(pValue : integer);
    procedure SetValue(pValue : double);
    procedure SetValMin(pValue : double);
    procedure SetValMax(pValue : double);
    procedure DrawValue;
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
    procedure WMSize(var Message: TWMSize); message WM_SIZE;
  protected
    procedure Paint; Override;
  public
    DampValue  : double;
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
    property OnClick;
    property OnDblClick;
    property OnMouseDown;
    property OnMouseMove;
```

```
    property OnMouseUp;
    property ValueMin     : double  read FValMin    write SetValMin;
    property ValueMax     : double  read FValMax    write SetValMax;
    property Value        : double  read FValue     write SetValue;
    property MeterMode    : boolean read FMeterMode write SetMeterMode;
    property SegmBorder   : boolean read FSegmBord  write SetSegmBord;
    property SegmGap      : integer read FSegGap    write SetSegGap;
    property SegmWidth    : integer read FSegWidth  write SetSegWidth;
    property SegmColor    : Tcolor  read FSegColor  write SetSegColor;
    property ColBack      : TColor  read FColBack   write SetColBack;
    property ColBorder    : TColor  read FColBorder write SetColBorder;
    property Percent      : double  read FPercent   write SetPercent;
    property PntrPercent  : double  read FPntrPrcnt write SetPntrPrcnt;
    property Damping      : integer read FDamp      write SetDamp;
    property Orientation : TOrientation read FOrient write SetOrientation;
  end;
```

MeterPanelH

Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Schleppzeiger und Over/Underange Event.

Komponenten Deklaration

```
(* Panel Meter rechteckig horizontal *)
type
  TMeterPanelH  = class(TThalesBase)
  private
    { Private-Deklarationen }
    FOnUpperLim: TNotifyEvent;
    FOnLowerLim: TNotifyEvent;
    RepeatTimer: integer;
    DampTimer  : integer;
    PosLed1    : TPoint;
    PosLed2    : TPoint;
    LedSize    : integer;
    XStartP    : integer;
    YStartP    : integer;
    XAbsP      : integer;
    YAbsP      : integer;
    XStartL    : LongInt;
    YStartL    : LongInt;
    XAbsL      : LongInt;
    YAbsL      : LongInt;
    PointerSize : integer;
    PeakPtrP,
    PeakPtrL,
    PeakPtrR,
    PointerP,
    PointerL,
    PointerR    : TPoint;
    painted    : Boolean;
    OverRange  : Boolean;
    UnderRange : Boolean;
    overOn     : Boolean;
    underOn    : Boolean;
    ovrField   : TRect;
    udrField   : TRect;
    FLedOn1    : Boolean;
    FLedOn2    : Boolean;
    FLedBlink1 : Boolean;
    FLedBlink2 : Boolean;
    FValue     : double;
    LogStart   : double;
    LogEnd     : double;
    FDamp      : Integer;
    FDampPeak  : Integer;
    FPeakMode  : TPeakPtrMode;
    FColPeakPtr: TColor;
    FScalaPsub : Integer;
    FUpperLim  : double;
    FLowerLim  : double;
    Led1Status : boolean;
```

```
    Led2Status : boolean;

    Procedure SetDamp(pValue : integer);
    Procedure SetDampPeak(pValue : integer);
    Procedure SetUpperLim(pValue : double);
    Procedure SetLowerLim(pValue : double);
    procedure TimerExpired(Sender: TObject);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetCaption(pValue : String32);
    Procedure SetFontCapt(pValue : TFont);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure DrawLed1;
    Procedure DrawLed2;
    Procedure SetNonLinear(pValue : Boolean);
    Procedure SetPeakMode(pValue : TPeakPtrMode);
    Procedure SetColPeakPtr(pValue : TColor);
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetScalaPsub(pValue : integer);
    procedure DrawRange(doInvert : boolean);
    procedure DrawValue(doDraw : boolean);
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
    procedure ScalePnt(XL,YL : longint; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape     : TmShapeType;
    FColScala  : TColor;
    FColShape  : TColor;
    FColLed1   : TColor;
    FColLed2   : TColor;
    FColPointer : TColor;
    FLedEna    : Boolean;
    FNonLinear : Boolean;
    FValueStart: double;
    FValueEnd  : double;
    FCaption  : String32;
    FFontCapt  : TFont;
    FScalaPart : Integer;
    DampValue  : double;
    PeakValue  : double;

    Procedure ResetPeakPtr;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
```

```
    property Font;
    property OnDblClick;
    property OnClick;
    property ColBack;
    Property ColLed1    : TColor   read FColLed1    write SetColLed1;
    Property ColLed2    : TColor   read FColLed2    write SetColLed2;
    Property LedEnable  : Boolean  read FLedEna     write SetLedEna;
    Property LedOn1     : Boolean  read FLedOn1     write SetLedOn1;
    Property LedOn2     : Boolean  read FLedOn2     write SetLedOn2;
    Property LedBlink1  : boolean  read FLedBlink1  write SetLedBlink1;
    Property LedBlink2  : boolean  read FLedBlink2  write SetLedBlink2;
    Property NonLinear  : Boolean  read FNonLinear  write SetNonLinear;
    Property Caption    : String32 read FCaption    write SetCaption;
    Property FontCapt   : TFont    read FFontCapt   write SetFontCapt;
    property Shape   : TmShapeType read FShape      write SetShape;
    property ColScala   : TColor   read FColScala   write SetColScala;
    property ColShape   : TColor   read FColShape   write SetColShape;
    property ColPointer : TColor   read FColPointer write SetColPointer;
    property UpperLimit : double   read FUpperLim   write SetUpperLim;
    property LowerLimit : double   read FLowerLim   write SetLowerLim;
    property Value      : double   read FValue      write SetValue;
    property ValueStart : double   read FValueStart write SetValueStart;
    property ValueEnd   : double   read FValueEnd   write SetValueEnd;
    property ScalaPart  : Integer  read FScalaPart  write SetScalaPart;
    property ScalaPsub  : Integer  read FScalaPsub  write SetScalaPsub;
    property Damping    : integer  read FDamp       write SetDamp;
    property ColPeakPtr: TColor   read FColPeakPtr write SetColPeakPtr;
    property DampPeak  : integer  read FDampPeak  write SetDampPeak;
    property PeakMode  : TPeakPtrMode read FPeakMode   write SetPeakMode;
    property OnUpperLim: TNotifyEvent read FOnUpperLim write FOnUpperLim;
    property OnLowerLim: TNotifyEvent read FOnLowerLim write FOnLowerLim;
  end;
```

MeterPanelV

Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung.

Anzeige wählbar: simpleMeter, BarGraph oder LedBar.
Zusätzliche, abschaltbare digitale Anzeige. Schleppzeiger und Over/Underange Event.

Komponenten Deklaration

```
(* Panel Meter rechteckig vertikal *)
type
  TMeterPanelV  = class(TThalesBase)
  private
    { Private-Deklarationen }
    FOnUpperLim: TNotifyEvent;
    FOnLowerLim: TNotifyEvent;
    RepeatTimer: integer;
    DampTimer  : integer;
    Display    : TDigDisplay;
    Disp,
    BarRect,
    einhRect,
    nameRect   : TRect;
    PosLed1    : TPoint;
    PosLed2    : TPoint;
    LedSize    : integer;
    XStartP    : integer;
    YStartP    : integer;
    XAbsP      : integer;
    YAbsP      : integer;
    XStartL    : LongInt;
    YStartL    : LongInt;
    XAbsL      : LongInt;
    YAbsL      : LongInt;
    VMin, VMid,
    PointerSize : integer;
    PeakPtrP,
    PeakPtrT,
    PeakPtrB,
    PointerP,
    PointerT,
    PointerB    : TPoint;
    painted    : Boolean;
    OverRange  : Boolean;
    UnderRange : Boolean;
    overOn     : Boolean;
    underOn    : Boolean;
    ovrField   : TRect;
    udrField   : TRect;
    FLedOn1    : Boolean;
    FLedOn2    : Boolean;
    FLedBlink1 : Boolean;
    FLedBlink2 : Boolean;
    LogStart   : double;
```

```
    LogEnd     : double;
    FValue     : double;
    FDamp      : Integer;
    FDampPeak  : Integer;
    FPeakMode  : TPeakPtrMode;
    FScalaPsub : Integer;
    FColPeakPtr: TColor;
    FItalic    : Boolean;
    FFixPnt    : Boolean;
    FUpperLim  : double;
    FLowerLim  : double;
    Led1Status : boolean;
    Led2Status : boolean;

    Procedure SetDamp(pValue : integer);
    Procedure SetDampPeak(pValue : integer);
    Procedure SetUpperLim(pValue : double);
    Procedure SetLowerLim(pValue : double);
    procedure TimerExpired(Sender: TObject);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetCaption(pValue : string);
    Procedure SetFontCapt(pValue : TFont);
    Procedure SetTextName(pValue : string);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure DrawLed1;
    Procedure DrawLed2;
    Procedure SetNonLinear(pValue : Boolean);
    Procedure SetMeterStyle(pValue : TMeterStyle);
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetColPeakPtr(pValue : TColor);
    Procedure SetPeakMode(pValue : TPeakPtrMode);
    Procedure SetDispEna(pValue : Boolean);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetDispFixPnt(pValue : Boolean);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetValMin(pValue : double);
    Procedure SetValMid(pValue : double);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColMin(pValue : TColor);
    Procedure SetColMid(pValue : TColor);
    Procedure SetColMax(pValue : TColor);
    Procedure SetColAnd(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetScalaPsub(pValue : integer);
    Procedure SetLedZahl(pValue : integer);
    procedure DrawRange(doInvert : boolean);
    procedure DrawValue(doDraw : boolean);
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
    procedure ScalePnt(XL,YL : longint; var XP,YP : integer);
  protected
```

```
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape     : TmShapeType;
    FColShape  : TColor;
    FColScala  : TColor;
    FColLed1   : TColor;
    FColLed2   : TColor;
    FColPointer : TColor;
    FColDBack  : TColor;
    FColDRand  : TColor;
    FColDisp   : TColor;
    FLedEna    : Boolean;
    FNonLinear : Boolean;
    FValueStart: double;
    FValueEnd  : double;
    FValMin    : double;
    FValMid    : double;
    FColMin    : TColor;
    FColMid    : TColor;
    FColMax    : TColor;
    FColAnd    : TColor;
    FCaption  : string;
    FTextName  : string;
    FFontCapt  : TFont;
    FScalaPart : Integer;
    FLedZahl   : Integer;
    FMeterStyle: TMeterStyle;
    FDispEna   : Boolean;
    DampValue  : double;
    PeakValue  : double;

    Procedure ResetPeakPtr;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    property OnDblClick;
    property OnClick;
    property ColBack;
    Property ColLed1    : TColor   read FColLed1    write SetColLed1;
    Property ColLed2    : TColor   read FColLed2    write SetColLed2;
    Property LedEnable  : Boolean  read FLedEna     write SetLedEna;
    Property LedOn1     : Boolean  read FLedOn1     write SetLedOn1;
    Property LedOn2     : Boolean  read FLedOn2     write SetLedOn2;
    Property LedBlink1  : boolean  read FLedBlink1  write SetLedBlink1;
    Property LedBlink2  : boolean  read FLedBlink2  write SetLedBlink2;
    Property NonLinear  : Boolean  read FNonLinear  write SetNonLinear;
    Property MeterStyle : TMeterStyle read FMeterStyle write SetMeterStyle;
    property ColDBack   : TColor   read FColDBack   write SetColDBack;
    property ColDRand   : TColor   read FColDRand   write SetColDRand;
    property ColDisplay : TColor   read FColDisp    write SetColDisp;
    Property DispEnable : Boolean  read FDispEna    write SetDispEna;
    Property DispFixPnt : Boolean  read FFixPnt     write SetDispFixPnt;
    Property DispItalic : Boolean  read FItalic     write SetDispItalic;
    Property Caption    : string   read FCaption    write SetCaption;
    Property TextName   : string   read FTextName   write SetTextName;
    Property FontCapt   : TFont    read FFontCapt   write SetFontCapt;
    property Shape  : TmShapeType  read FShape      write SetShape;
```

```
    property ColScala   : TColor   read FColScala   write SetColScala;
    property ColShape   : TColor   read FColShape   write SetColShape;
    property ColPointer : TColor   read FColPointer write SetColPointer;
    property UpperLimit : double   read FUpperLim   write SetUpperLim;
    property LowerLimit : double   read FLowerLim   write SetLowerLim;
    property Value      : double   read FValue      write SetValue;
    property ValueStart : double   read FValueStart write SetValueStart;
    property ValueEnd   : double   read FValueEnd   write SetValueEnd;
    property ScalaPart  : Integer  read FScalaPart  write SetScalaPart;
    property ScalaPsub  : Integer  read FScalaPsub  write SetScalaPsub;
    property LedZahl    : Integer  read FLedZahl    write SetLedZahl;
    property ColMin     : TColor   read FColMin     write SetColMin;
    property ColMid     : TColor   read FColMid     write SetColMid;
    property ColMax     : TColor   read FColMax     write SetColMax;
    property ColAnd     : TColor   read FColAnd     write SetColAnd;
    property ValueMin   : double   read FValMin     write SetValMin;
    property ValueMid   : double   read FValMid     write SetValMid;
    property Damping    : integer  read FDamp       write SetDamp;
    property ColPeakPtr: TColor   read FColPeakPtr write SetColPeakPtr;
    property DampPeak  : integer  read FDampPeak  write SetDampPeak;
    property PeakMode  : TPeakPtrMode read FPeakMode   write SetPeakMode;
    property OnUpperLim: TNotifyEvent read FOnUpperLim write FOnUpperLim;
    property OnLowerLim: TNotifyEvent read FOnLowerLim write FOnLowerLim;
  end;
```

Meter120

Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung.

Zusätzliche, abschaltbare digitale Anzeige.
In 90grad-Schritten drehbar.
Schleppzeiger und Over/Underange Event.

Komponenten Deklaration

```
typetype
  (* Tacho-Meter 120 *)
  TMeter120   = class(TThalesBase)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    DampTimer  : integer;
    Display    : TDigDisplay;
    XStartP    : integer;
    YStartP    : integer;
    XAbsP      : integer;
    YAbsP      : integer;
    XStartL    : integer;
    YStartL    : integer;
    XAbsL      : integer;
    YAbsL      : integer;
    ClipLeft   : integer;
    ClipRight  : integer;
    ClipTop    : integer;
    ClipBottom : integer;
    LedSize    : integer;
    PosLed1    : TPoint;
    PosLed2    : TPoint;
    Disp       : TRect;
    MittelP,
    PeakPtrP,
    PeakPtrL,
    PeakPtrR,
    PeakPtrP0,
    PeakPtrL0,
    PeakPtrR0,
    PointerP,
    PointerL,
    PointerR,
    PointerP0,
    PointerL0,
    PointerR0  : TPoint;
    OverRange  : Boolean;
    UnderRange : Boolean;
    overOn     : Boolean;
    underOn    : Boolean;
    ovrField   : TRect;
    udrField   : TRect;
    dir        : integer;
```

```
    StartW     : double;
    FLedOn1    : Boolean;
    FLedOn2    : Boolean;
    FLedBlink1 : Boolean;
    FLedBlink2 : Boolean;
    FValue     : double;
    LogStart   : double;
    LogEnd     : double;
    FDamp      : Integer;
    FDampPeak  : Integer;
    FScalaPsub : Integer;
    FOnUpperLim: TNotifyEvent;
    FOnLowerLim: TNotifyEvent;
    FPeakMode  : TPeakPtrMode;
    Led1Status : boolean;
    Led2Status : boolean;

    Procedure SetDamp(pValue : integer);
    Procedure SetDampPeak(pValue : integer);
    Procedure SetScalaPsub(pValue : integer);
    procedure TimerExpired(Sender: TObject);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetNonLinear(pValue : Boolean);
    Procedure SetOrient(pValue : TOrientType);
    Procedure SetColMin(pValue : TColor);
    Procedure SetColMid(pValue : TColor);
    Procedure SetColMax(pValue : TColor);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColPeakPtr(pValue : TColor);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetValMinA(pValue : double);
    Procedure SetValMinE(pValue : double);
    Procedure SetValMidA(pValue : double);
    Procedure SetValMidE(pValue : double);
    Procedure SetValMaxA(pValue : double);
    Procedure SetValMaxE(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string);
    Procedure SetDispEna(pValue : Boolean);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetDispFixPnt(pValue : Boolean);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure SetPeakMode(pValue : TPeakPtrMode);
    procedure DrawRange(doInvert : boolean);
    Procedure DrawLed1;
    Procedure DrawLed2;
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    procedure DrawValue(doDraw : boolean);
    procedure DrawOrgBmp;
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
```

```
    procedure ScalePnt(XL,YL : integer; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape     : TmShapeType;
    FOrient    : TOrientType;
    FColMin    : TColor;
    FColMid    : TColor;
    FColMax    : TColor;
    FColPointer: TColor;
    FColPeakPtr: TColor;
    FColDBack  : TColor;
    FColDRand  : TColor;
    FColDisp   : TColor;
    FColShape  : TColor;
    FColScala  : TColor;
    FColLed1   : TColor;
    FColLed2   : TColor;
    FCaption   : string;
    FScalaPart : integer;
    FValMinA   : double;
    FValMinE   : double;
    FValMidA   : double;
    FValMidE   : double;
    FValMaxA   : double;
    FValMaxE   : double;
    FValueStart: double;
    FValueEnd  : double;
    FLedEna    : Boolean;
    FDispEna   : Boolean;
    FItalic    : Boolean;
    FFixPnt    : Boolean;
    FNonLinear : Boolean;
    DampValue  : double;
    PeakValue  : double;

    Procedure ResetPeakPtr;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property OnDblClick;
    property OnClick;
    property Font;
    Property FontAutom;
    property ColBack;
    property Value     : double   read FValue     write SetValue;
    property Shape     : TmShapeType read FShape  write SetShape;
    property Orientation  : TOrientType read FOrient write SetOrient;
    property ColMin    : TColor   read FColMin    write SetColMin;
    property ColMid    : TColor   read FColMid    write SetColMid;
    property ColMax    : TColor   read FColMax    write SetColMax;
    property ColPointer: TColor   read FColPointer write SetColPointer;
    property ColDBack  : TColor   read FColDBack  write SetColDBack;
    property ColDRand  : TColor   read FColDRand  write SetColDRand;
    property ColDisplay: TColor   read FColDisp   write SetColDisp;
    property ColScala  : TColor   read FColScala  write SetColScala;
    property ColShape  : TColor   read FColShape  write SetColShape;
    property ValueStart: double   read FValueStart write SetValueStart;
```

```
    property ValueEnd  : double   read FValueEnd  write SetValueEnd;
    property ValueMinA : double   read FValMinA   write SetValMinA;
    property ValueMinE : double   read FValMinE   write SetValMinE;
    property ValueMidA : double   read FValMidA   write SetValMidA;
    property ValueMidE : double   read FValMidE   write SetValMidE;
    property ValueMaxA : double   read FValMaxA   write SetValMaxA;
    property ValueMaxE : double   read FValMaxE   write SetValMaxE;
    property ScalaPart : integer  read FScalaPart write SetScalaPart;
    property ScalaPsub : Integer  read FScalaPsub write SetScalaPsub;
    property Caption   : string   read FCaption   write SetCaption;
    Property ColLed1   : TColor   read FColLed1   write SetColLed1;
    Property ColLed2   : TColor   read FColLed2   write SetColLed2;
    Property LedEnable : Boolean  read FLedEna    write SetLedEna;
    Property LedOn1    : boolean  read FLedOn1    write SetLedOn1;
    Property LedOn2    : boolean  read FLedOn2    write SetLedOn2;
    Property LedBlink1 : boolean  read FLedBlink1 write SetLedBlink1;
    Property LedBlink2 : boolean  read FLedBlink2 write SetLedBlink2;
    Property DispEnable: Boolean  read FDispEna   write SetDispEna;
    Property DispItalic: Boolean  read FItalic    write SetDispItalic;
    Property DispFixPnt: Boolean  read FFixPnt    write SetDispFixPnt;
    Property NonLinear : Boolean  read FNonLinear write SetNonLinear;
    property Damping   : integer  read FDamp      write SetDamp;
    property ColPeakPtr: TColor   read FColPeakPtr write SetColPeakPtr;
    property DampPeak  : integer  read FDampPeak  write SetDampPeak;
    property PeakMode  : TPeakPtrMode read FPeakMode   write SetPeakMode;
    property OnUpperLim: TNotifyEvent read FOnUpperLim write FOnUpperLim;
    property OnLowerLim: TNotifyEvent read FOnLowerLim write FOnLowerLim;
  end;
```

Meter180

Panel Meter mit umschaltbarer linearer/logarithmischer Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar. Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung.

Zusätzliche, abschaltbare digitale Anzeige.
In 90grad-Schritten drehbar.
Schleppzeiger und Over/Underange Event.

Komponenten Deklaration

```
(* Tacho-Meter 180 *)
TMeter180   = class(TThalesBase)
private
  { Private-Deklarationen }
  RepeatTimer: integer;
  DampTimer  : integer;
  Display    : TDigDisplay;
  XStartP    : integer;
  YStartP    : integer;
  XAbsP      : integer;
  YAbsP      : integer;
  XStartL    : integer;
  YStartL    : integer;
  XAbsL      : integer;
  YAbsL      : integer;
  LedSize    : integer;
  PosLed1    : TPoint;
  PosLed2    : TPoint;
  Disp       : TRect;
  MittelP,
  PeakPtrP,
  PeakPtrL,
  PeakPtrR,
  PeakPtrP0,
  PeakPtrL0,
  PeakPtrR0,
  PointerP,
  PointerL,
  PointerR,
  PointerP0,
  PointerL0,
  PointerR0  : TPoint;
  OverRange  : Boolean;
  UnderRange : Boolean;
  overOn     : Boolean;
  underOn    : Boolean;
  ovrField   : TRect;
  udrField   : TRect;
  dir        : integer;
  StartW     : double;
  FLedOn1    : Boolean;
  FLedOn2    : Boolean;
  FLedBlink1 : Boolean;
```

```
    FLedBlink2 : Boolean;
    FValue     : double;
    LogStart   : double;
    LogEnd     : double;
    FDamp      : Integer;
    FDampPeak  : Integer;
    FScalaPsub : Integer;
    FOnUpperLim: TNotifyEvent;
    FOnLowerLim: TNotifyEvent;
    FPeakMode  : TPeakPtrMode;
    Led1Status : boolean;
    Led2Status : boolean;

    Procedure SetDamp(pValue : integer);
    Procedure SetDampPeak(pValue : integer);
    Procedure SetNonLinear(pValue : Boolean);
    Procedure SetScalaPsub(pValue : integer);
    procedure TimerExpired(Sender: TObject);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetOrient(pValue : TOrientType);
    Procedure SetColMin(pValue : TColor);
    Procedure SetColMid(pValue : TColor);
    Procedure SetColMax(pValue : TColor);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColPeakPtr(pValue : TColor);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetColCircle(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetValMinA(pValue : double);
    Procedure SetValMinE(pValue : double);
    Procedure SetValMidA(pValue : double);
    Procedure SetValMidE(pValue : double);
    Procedure SetValMaxA(pValue : double);
    Procedure SetValMaxE(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string);
    Procedure SetDispEna(pValue : Boolean);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetDispFixPnt(pValue : Boolean);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure SetPeakMode(pValue : TPeakPtrMode);
    procedure DrawRange(doInvert : boolean);
    Procedure DrawLed1;
    Procedure DrawLed2;
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    procedure DrawValue(doDraw : boolean);
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
    procedure ScalePnt(XL,YL : integer; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
```

```
  public
    { Public-Deklarationen }
    FShape     : TmShapeType;
    FOrient    : TOrientType;
    FColMin    : TColor;
    FColMid    : TColor;
    FColMax    : TColor;
    FColPeakPtr: TColor;
    FColPointer: TColor;
    FColDBack  : TColor;
    FColDRand  : TColor;
    FColDisp   : TColor;
    FColShape  : TColor;
    FColCircle : TColor;
    FColScala  : TColor;
    FColLed1   : TColor;
    FColLed2   : TColor;
    FCaption  : string;
    FScalaPart : integer;
    FValMinA   : double;
    FValMinE   : double;
    FValMidA   : double;
    FValMidE   : double;
    FValMaxA   : double;
    FValMaxE   : double;
    FValueStart: double;
    FValueEnd  : double;
    FNonLinear : Boolean;
    FLedEna    : Boolean;
    FDispEna   : Boolean;
    FItalic    : Boolean;
    FFixPnt    : Boolean;
    DampValue  : double;
    PeakValue  : double;

    Procedure ResetPeakPtr;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property OnDblClick;
    property OnClick;
    property Font;
    Property FontAutom;
    property Shape      : TmShapeType read FShape   write SetShape;
    property Orientation  : TOrientType read FOrient write SetOrient;
    property ColMin     : TColor   read FColMin     write SetColMin;
    property ColMid     : TColor   read FColMid     write SetColMid;
    property ColMax     : TColor   read FColMax     write SetColMax;
    property ColPointer : TColor   read FColPointer write SetColPointer;
    property ColPeakPtr : TColor   read FColPeakPtr write SetColPeakPtr;
    property ColDBack   : TColor   read FColDBack   write SetColDBack;
    property ColDRand   : TColor   read FColDRand   write SetColDRand;
    property ColDisplay : TColor   read FColDisp    write SetColDisp;
    property ColBack;
    property ColScala   : TColor   read FColScala   write SetColScala;
    property ColShape   : TColor   read FColShape   write SetColShape;
    property ColCircle  : TColor   read FColCircle  write SetColCircle;
    property Value      : double   read FValue      write SetValue;
    property ValueStart : double   read FValueStart write SetValueStart;
    property ValueEnd   : double   read FValueEnd   write SetValueEnd;
```

```
    property ValueMinA  : double   read FValMinA    write SetValMinA;
    property ValueMinE  : double   read FValMinE    write SetValMinE;
    property ValueMidA  : double   read FValMidA    write SetValMidA;
    property ValueMidE  : double   read FValMidE    write SetValMidE;
    property ValueMaxA  : double   read FValMaxA    write SetValMaxA;
    property ValueMaxE  : double   read FValMaxE    write SetValMaxE;
    property ScalaPart  : integer  read FScalaPart  write SetScalaPart;
    property ScalaPsub  : Integer  read FScalaPsub  write SetScalaPsub;
    Property NonLinear  : Boolean  read FNonLinear  write SetNonLinear;
    property Caption    : string   read FCaption    write SetCaption;
    Property ColLed1    : TColor   read FColLed1    write SetColLed1;
    Property ColLed2    : TColor   read FColLed2    write SetColLed2;
    Property LedEnable  : Boolean  read FLedEna     write SetLedEna;
    Property LedOn1     : boolean  read FLedOn1     write SetLedOn1;
    Property LedOn2     : boolean  read FLedOn2     write SetLedOn2;
    Property LedBlink1  : boolean  read FLedBlink1  write SetLedBlink1;
    Property LedBlink2  : boolean  read FLedBlink2  write SetLedBlink2;
    Property DispEnable : Boolean  read FDispEna    write SetDispEna;
    Property DispItalic : Boolean  read FItalic     write SetDispItalic;
    Property DispFixPnt : Boolean  read FFixPnt     write SetDispFixPnt;
    property Damping    : integer  read FDamp       write SetDamp;
    property DampPeak   : integer  read FDampPeak   write SetDampPeak;
    property PeakMode  : TPeakPtrMode read FPeakMode   write SetPeakMode;
    property OnUpperLim: TNotifyEvent read FOnUpperLim write FOnUpperLim;
    property OnLowerLim: TNotifyEvent read FOnLowerLim write FOnLowerLim;
  end;
```

Meter270

Panel Meter mit 270grad Skala. Einstellbare Dämpfung/Integration von 0..10. Over/Under-Range Anzeige durch blinken der entsprechenden Endwerte. Frei beschriftbar.

Abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Zusätzliche, abschaltbare digitale Anzeige. Schleppzeiger und Over/Underange Event.

Komponenten Deklaration

```
(* Tacho-Meter 270 *)
TMeter270    = class(TThalesBase)
private
  { Private-Deklarationen }
  RepeatTimer: integer;
  DampTimer  : integer;
  Display    : TDigDisplay;
  XStartP    : integer;
  YStartP    : integer;
  XAbsP      : integer;
  YAbsP      : integer;
  XStartL    : integer;
  YStartL    : integer;
  XAbsL      : integer;
  YAbsL      : integer;
  LedSize    : integer;
  PosLed1    : TPoint;
  PosLed2    : TPoint;
  Disp       : TRect;
  MittelP,
  PeakPtrP,
  PeakPtrL,
  PeakPtrR,
  PeakPtrP0,
  PeakPtrL0,
  PeakPtrR0,
  PointerP,
  PointerL,
  PointerR,
  PointerP0,
  PointerL0,
  PointerR0  : TPoint;
  OverRange  : Boolean;
  UnderRange : Boolean;
  overOn     : Boolean;
  underOn    : Boolean;
  ovrField   : TRect;
  udrField   : TRect;
  FLedOn1    : Boolean;
  FLedOn2    : Boolean;
  FLedBlink1 : Boolean;
  FLedBlink2 : Boolean;
  FValue     : double;
  LogStart   : double;
  LogEnd     : double;
  FDamp      : Integer;
  FDampPeak  : Integer;
```

```
    FScalaPsub : Integer;
    FOnUpperLim: TNotifyEvent;
    FOnLowerLim: TNotifyEvent;
    FPeakMode  : TPeakPtrMode;
    Led1Status : boolean;
    Led2Status : boolean;

    Procedure SetDamp(pValue : integer);
    Procedure SetDampPeak(pValue : integer);
    Procedure SetNonLinear(pValue : Boolean);
    Procedure SetScalaPsub(pValue : integer);
    procedure TimerExpired(Sender: TObject);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TShapeTypeT);
    Procedure SetColMin(pValue : TColor);
    Procedure SetColMid(pValue : TColor);
    Procedure SetColMax(pValue : TColor);
    Procedure SetColPeakPtr(pValue : TColor);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetColCircle(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetValMinA(pValue : double);
    Procedure SetValMinE(pValue : double);
    Procedure SetValMidA(pValue : double);
    Procedure SetValMidE(pValue : double);
    Procedure SetValMaxA(pValue : double);
    Procedure SetValMaxE(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string);
    Procedure SetDispEna(pValue : Boolean);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetDispFixPnt(pValue : Boolean);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    procedure DrawRange(doInvert : boolean);
    Procedure DrawLed1;
    Procedure DrawLed2;
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    Procedure SetPeakMode(pValue : TPeakPtrMode);
    procedure DrawValue(doDraw : boolean);
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
    procedure ScalePnt(XL,YL : integer; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape     : TShapeTypeT;
    FColMin    : TColor;
    FColMid    : TColor;
    FColMax    : TColor;
    FColPeakPtr: TColor;
```

```
    FColPointer: TColor;
    FColDBack  : TColor;
    FColDRand  : TColor;
    FColDisp   : TColor;
    FColScala  : TColor;
    FColShape  : TColor;
    FColCircle : TColor;
    FColLed1   : TColor;
    FColLed2   : TColor;
    FCaption  : string;
    FScalaPart : integer;
    FValMinA   : double;
    FValMinE   : double;
    FValMidA   : double;
    FValMidE   : double;
    FValMaxA   : double;
    FValMaxE   : double;
    FValueStart: double;
    FValueEnd  : double;
    FNonLinear : Boolean;
    FLedEna    : Boolean;
    FDispEna   : Boolean;
    FItalic    : Boolean;
    FFixPnt    : Boolean;
    DampValue  : double;
    PeakValue  : double;

    Procedure ResetPeakPtr;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property OnDblClick;
    property OnClick;
    property Font;
    Property FontAutom;
    property Shape   : TShapeTypeT read FShape      write SetShape;
    property ColMin    : TColor   read FColMin     write SetColMin;
    property ColMid    : TColor   read FColMid     write SetColMid;
    property ColMax    : TColor   read FColMax     write SetColMax;
    property ColPointer: TColor   read FColPointer write SetColPointer;
    property ColPeakPtr: TColor   read FColPeakPtr write SetColPeakPtr;
    property ColDBack  : TColor   read FColDBack   write SetColDBack;
    property ColDRand  : TColor   read FColDRand   write SetColDRand;
    property ColDisplay: TColor   read FColDisp    write SetColDisp;
    property ColBack;
    property ColScala  : TColor   read FColScala   write SetColScala;
    property ColShape  : TColor   read FColShape   write SetColShape;
    property ColCircle : TColor   read FColCircle  write SetColCircle;
    property Value     : double   read FValue      write SetValue;
    property ValueStart: double   read FValueStart write SetValueStart;
    property ValueEnd  : double   read FValueEnd   write SetValueEnd;
    property ValueMinA : double   read FValMinA    write SetValMinA;
    property ValueMinE : double   read FValMinE    write SetValMinE;
    property ValueMidA : double   read FValMidA    write SetValMidA;
    property ValueMidE : double   read FValMidE    write SetValMidE;
    property ValueMaxA : double   read FValMaxA    write SetValMaxA;
    property ValueMaxE : double   read FValMaxE    write SetValMaxE;
    property ScalaPart : integer  read FScalaPart  write SetScalaPart;
    property ScalaPsub : Integer  read FScalaPsub  write SetScalaPsub;
    Property NonLinear : Boolean  read FNonLinear  write SetNonLinear;
```

```
    property Caption   : string   read FCaption    write SetCaption;
    Property ColLed1   : TColor   read FColLed1    write SetColLed1;
    Property ColLed2   : TColor   read FColLed2    write SetColLed2;
    Property LedEnable : Boolean  read FLedEna     write SetLedEna;
    Property LedOn1    : Boolean  read FLedOn1     write SetLedOn1;
    Property LedOn2    : Boolean  read FLedOn2     write SetLedOn2;
    Property LedBlink1 : boolean  read FLedBlink1  write SetLedBlink1;
    Property LedBlink2 : boolean  read FLedBlink2  write SetLedBlink2;
    Property DispEnable: Boolean  read FDispEna    write SetDispEna;
    Property DispItalic: Boolean  read FItalic     write SetDispItalic;
    Property DispFixPnt: Boolean  read FFixPnt     write SetDispFixPnt;
    property Damping   : integer  read FDamp       write SetDamp;
    property DampPeak  : integer  read FDampPeak   write SetDampPeak;
    property PeakMode  : TPeakPtrMode read FPeakMode   write SetPeakMode;
    property OnUpperLim: TNotifyEvent read FOnUpperLim write FOnUpperLim;
    property OnLowerLim: TNotifyEvent read FOnLowerLim write FOnLowerLim;
  end;
```

Meter360

Panel Meter mit 360grad Skala. Zwei unabhängige Skalen und Zeiger.
Separat einstellbare Dämpfung/Integration von 0..10. Frei beschriftbar. Vier abschaltbare Leds. Alle Kolorierungen frei wählbar. Einstellbare Skalenteilung. Linear/Logarithmisch separat wählbar.

Schleppzeiger und Over/Underange Event.

Komponenten Deklaration

```
(* Tacho-Meter 360 *)
TMeter360    = class(TThalesBase)
private
  { Private-Deklarationen }
  RepeatTimer: integer;
  DampTimer  : integer;
  XStartP    : integer;
  YStartP    : integer;
  XAbsP      : integer;
  YAbsP      : integer;
  XStartL    : integer;
  YStartL    : integer;
  XAbsL      : integer;
  YAbsL      : integer;
  LedSize1   : integer;
  LedSize2   : integer;
  PosLed1    : TPoint;
  PosLed2    : TPoint;
  PosLed3    : TPoint;
  PosLed4    : TPoint;
  MittelP,
  PeakPtrP,
  PeakPtrL,
  PeakPtrR,
  PeakPtrP0,
  PeakPtrL0,
  PeakPtrR0,
  Pointer1P,
  Pointer1L,
  Pointer1R,
  Pointer1P0,
  Pointer1L0,
  Pointer1R0,
  Pointer2P,
  Pointer2L,
  Pointer2R,
  Pointer2P0,
  Pointer2L0,
  Pointer2R0  : TPoint;
  FLedOn1    : Boolean;
  FLedOn2    : Boolean;
  FLedOn3    : Boolean;
  FLedOn4    : Boolean;
  FLedBlink1 : Boolean;
  FLedBlink2 : Boolean;
  FLedBlink3 : Boolean;
```

```
    FLedBlink4 : Boolean;
    FValue1    : double;
    FValue2    : double;
    LogStart1  : double;
    LogEnd1    : double;
    LogStart2  : double;
    LogEnd2    : double;
    FDamp1     : Integer;
    FDamp2     : Integer;
    FDampPeak  : Integer;
    FScalaPsub1 : Integer;
    FScalaPsub2 : Integer;
    FPeakMode  : TPeakPtrMode;
    Led1Status : boolean;
    Led2Status : boolean;
    Led3Status : boolean;
    Led4Status : boolean;

    Procedure SetDamp1(pValue : integer);
    Procedure SetDamp2(pValue : integer);
    Procedure SetDampPeak(pValue : integer);
    Procedure SetNonLinear1(pValue : Boolean);
    Procedure SetNonLinear2(pValue : Boolean);
    Procedure SetScalaPsub1(pValue : integer);
    Procedure SetScalaPsub2(pValue : integer);
    procedure TimerExpired(Sender: TObject);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TShapeTypeT);
    Procedure SetColPeakPtr(pValue : TColor);
    Procedure SetColPointer1(pValue : TColor);
    Procedure SetColPointer2(pValue : TColor);
    Procedure SetColFont1(pValue : TColor);
    Procedure SetColFont2(pValue : TColor);
    Procedure SetColScala1(pValue : TColor);
    Procedure SetColScala2(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetColCircle(pValue : TColor);
    Procedure SetValue1(pValue : double);
    Procedure SetValue2(pValue : double);
    Procedure SetValueStart1(pValue : double);
    Procedure SetValueStart2(pValue : double);
    Procedure SetValueEnd1(pValue : double);
    Procedure SetValueEnd2(pValue : double);
    Procedure SetScalaPart1(pValue : integer);
    Procedure SetScalaPart2(pValue : integer);
    Procedure SetCaption1(pValue : string);
    Procedure SetCaption2(pValue : string);
    Procedure SetLedEna1(pValue : Boolean);
    Procedure SetLedEna2(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedOn3(pValue : Boolean);
    Procedure SetLedOn4(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure SetLedBlink3(pValue : Boolean);
    Procedure SetLedBlink4(pValue : Boolean);
    Procedure DrawLed1;
    Procedure DrawLed2;
    Procedure DrawLed3;
    Procedure DrawLed4;
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
```

```
    Procedure SetColLed3(pValue : TColor);
    Procedure SetColLed4(pValue : TColor);
    Procedure SetPeakMode(pValue : TPeakPtrMode);
    procedure DrawValue(doDraw : boolean);
    procedure WMDrawValue(Var Message : TMessage); message WM_DRAWVALUE;
    procedure ScalePnt(XL,YL : integer; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape       : TShapeTypeT;
    FColPeakPtr  : TColor;
    FColPointer1 : TColor;
    FColPointer2 : TColor;
    FColScala1   : TColor;
    FColScala2   : TColor;
    FColFont1    : TColor;
    FColFont2    : TColor;
    FColShape    : TColor;
    FColCircle   : TColor;
    FColLed1     : TColor;
    FColLed2     : TColor;
    FColLed3     : TColor;
    FColLed4     : TColor;
    FCaption1    : string;
    FCaption2    : string;
    FScalaPart1  : integer;
    FScalaPart2  : integer;
    FValueStart1 : double;
    FValueStart2 : double;
    FValueEnd1   : double;
    FValueEnd2   : double;
    FNonLinear1  : Boolean;
    FNonLinear2  : Boolean;
    FLedEna1     : Boolean;
    FLedEna2     : Boolean;
    DampValue1   : double;
    DampValue2   : double;
    PeakValue    : double;

    Procedure ResetPeakPtr;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property OnDblClick;
    property OnClick;
    property Font;
    Property FontAutom;
    property Shape     : TShapeTypeT read FShape       write SetShape;
    property ColPointer1 : TColor   read FColPointer1 write SetColPointer1;
    property ColPointer2 : TColor   read FColPointer2 write SetColPointer2;
    property ColPeakPtr  : TColor   read FColPeakPtr  write SetColPeakPtr;
    property ColBack;
    property ColFont1    : TColor   read FColFont1    write SetColFont1;
    property ColFont2    : TColor   read FColFont2    write SetColFont2;
    property ColScala1   : TColor   read FColScala1   write SetColScala1;
    property ColScala2   : TColor   read FColScala2   write SetColScala2;
    property ColShape    : TColor   read FColShape    write SetColShape;
    property ColCircle   : TColor   read FColCircle   write SetColCircle;
```

```
    property Value1      : double   read FValue1      write SetValue1;
    property Value2      : double   read FValue2      write SetValue2;
    property ValueStart1 : double   read FValueStart1 write SetValueStart1;
    property ValueEnd1   : double   read FValueEnd1   write SetValueEnd1;
    property ValueStart2 : double   read FValueStart2 write SetValueStart2;
    property ValueEnd2   : double   read FValueEnd2   write SetValueEnd2;
    property ScalaPart1  : integer  read FScalaPart1  write SetScalaPart1;
    property ScalaPsub1  : Integer  read FScalaPsub1  write SetScalaPsub1;
    property ScalaPart2  : integer  read FScalaPart2  write SetScalaPart2;
    property ScalaPsub2  : Integer  read FScalaPsub2  write SetScalaPsub2;
    Property NonLinear1  : Boolean  read FNonLinear1  write SetNonLinear1;
    Property NonLinear2  : Boolean  read FNonLinear2  write SetNonLinear2;
    property Caption1    : string   read FCaption1    write SetCaption1;
    property Caption2    : string   read FCaption2    write SetCaption2;
    Property ColLed1     : TColor   read FColLed1     write SetColLed1;
    Property ColLed2     : TColor   read FColLed2     write SetColLed2;
    Property ColLed3     : TColor   read FColLed3     write SetColLed3;
    Property ColLed4     : TColor   read FColLed4     write SetColLed4;
    Property LedEnable1  : Boolean  read FLedEna1     write SetLedEna1;
    Property LedEnable2  : Boolean  read FLedEna2     write SetLedEna2;
    Property LedOn1      : Boolean  read FLedOn1      write SetLedOn1;
    Property LedOn2      : Boolean  read FLedOn2      write SetLedOn2;
    Property LedOn3      : Boolean  read FLedOn3      write SetLedOn3;
    Property LedOn4      : Boolean  read FLedOn4      write SetLedOn4;
    Property LedBlink1   : boolean  read FLedBlink1   write SetLedBlink1;
    Property LedBlink2   : boolean  read FLedBlink2   write SetLedBlink2;
    Property LedBlink3   : boolean  read FLedBlink3   write SetLedBlink3;
    Property LedBlink4   : boolean  read FLedBlink4   write SetLedBlink4;
    property Damping1    : integer  read FDamp1       write SetDamp1;
    property Damping2    : integer  read FDamp2       write SetDamp2;
    property DampPeak    : integer  read FDampPeak    write SetDampPeak;
    property PeakMode : TPeakPtrMode read FPeakMode   write SetPeakMode;
  end;
```

Wecker

AlarmClock Uhr mit Weckfunktion. Abschaltbare digitale Uhrzeit- und Datum

Komponenten Deklaration

```
(* Wecker *)
TWecker      = class(TThalesBase)
private
  { Private-Deklarationen }
  RepeatTimer: integer;
  DisplayD   : TDigDisplay;
  DisplayT   : TDigDisplay;
  FShape     : TShapeTypeT;
  FColZeigH  : TColor;
  FColZeigM  : TColor;
  FColZeigS  : TColor;
  FColDBack  : TColor;
  FColDRand  : TColor;
  FColDisp   : TColor;
  FColSkala  : TColor;
  FColCircle : TColor;
  FColShape  : TColor;
  FColLed1   : TColor;
  FDispTEna  : Boolean;
  FDispDEna  : Boolean;
  FTimeHour  : word;
  FTimeMin   : word;
  FTimeSec   : word;
  fTimeDT    : tDateTime;
  fDateDT    : tDateTime;
  FItalic    : Boolean;
  fUserTime  : boolean;

  XStartP    : integer;
  YStartP    : integer;
  XAbsP      : integer;
  YAbsP      : integer;
  XStartL    : LongInt;
  YStartL    : LongInt;
  XAbsL      : LongInt;
  YAbsL      : LongInt;
  LedSize    : integer;
  PosLed1    : TPoint;
  Blink1     : Boolean;
  DispD,
  DispT      : TRect;

  hour,
  min,
  sec,
  msec   : word;

  PointerHPL,
  PointerHPR,
  PointerHBL,
  PointerHBR,
```

```
    PointerHPL0,
    PointerHPR0,
    PointerHBL0,
    PointerHBR0,

    PointerMPL,
    PointerMPR,
    PointerMBL,
    PointerMBR,
    PointerMPL0,
    PointerMPR0,
    PointerMBL0,
    PointerMBR0,

    PointerSP,
    PointerS0,
    MittelP    : TPoint;

    painted    : Boolean;
    FAlarmOn   : Boolean;
    FAlarmStd  : Word;
    FAlarmMin  : Word;
    FAlarmTon  : Word;
    FOnAlarm   : TNotifyEvent;
    AlarmExec  : Boolean;

    procedure TimerExpired(Sender: TObject);
    Procedure SetShape(pValue : TShapeTypeT);
    Procedure SetColShape(pValue : TColor);
    Procedure SetColCircle(pValue : TColor);
    Procedure SetColZeigH(pValue : TColor);
    Procedure SetColZeigM(pValue : TColor);
    Procedure SetColZeigS(pValue : TColor);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetUserTime(pValue : Boolean);
    Procedure SetTimeDT(pValue : tDateTime);
    Procedure SetDateDT(pValue : tDateTime);
    Procedure SetTimeHour(pValue : word);
    Procedure SetTimeMin(pValue : word);
    Procedure SetTimeSec(pValue : word);
    Procedure SetColSkala(pValue : TColor);
    Procedure SetDispTEna(pValue : Boolean);
    Procedure SetDispDEna(pValue : Boolean);
    Procedure SwitchLed1(pValue : boolean);
    Procedure SetColLed1(pValue : TColor);
    Procedure SetAlarmOn(pValue : boolean);
    Procedure DrawValue(doDraw : boolean);
    procedure ScalePnt(XL,YL : longint; var XP,YP : integer);
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property ondblClick;
```

```
    property onClick;
    property OnAlarm : TNotifyEvent read FOnAlarm write FOnAlarm;
    property Font;
    Property FontAutom;
    property ColBack;
    property Shape  : TShapeTypeT read FShape    write SetShape;
    property ColShape  : TColor  read FColShape  write SetColShape;
    property ColCircle : TColor  read FColCircle  write SetColCircle;
    property ColPointerH: TColor  read FColZeigH write SetColZeigH;
    property ColPointerM: TColor  read FColZeigM write SetColZeigM;
    property ColPointerS: TColor  read FColZeigS write SetColZeigS;
    property ColDBack  : TColor  read FColDBack write SetColDBack;
    property ColDRand  : TColor  read FColDRand write SetColDRand;
    property ColDisplay: TColor  read FColDisp  write SetColDisp;
    property ColSkala  : TColor  read FColSkala write SetColSkala;
    Property ColLed1   : TColor  read FColLed1  write SetColLed1;
    Property DispTEna  : Boolean read FDispTEna write SetDispTEna;
    Property DispDEna  : Boolean read FDispDEna write SetDispDEna;
    Property DispItalic: Boolean read FItalic   write SetDispItalic;
    Property AlarmOn   : Boolean read FAlarmOn  write SetAlarmOn;
    Property UserTime  : Boolean read FUserTime write SetUserTime;
    Property AlarmHour : Word    read FAlarmStd write FAlarmStd;
    Property AlarmMin  : Word    read FAlarmMin write FAlarmMin;
    Property AlarmTon  : Word    read FAlarmTon write FAlarmTon;
    Property TimeHour  : Word    read FTimeHour write SetTimeHour;
    Property TimeMin   : Word    read FTimeMin  write SetTimeMin;
    Property TimeSec   : Word    read FTimeSec  write SetTimeSec;
    Property TimeDT    : tDateTime read FTimeDT write SetTimeDT;
    Property DateDT    : tDateTime read FDateDT write SetDateDT;
  end;
```

Kompass

mit digitaler Anzeige von Ist-Kurs und Abweichung

Komponenten Deklaration

```
type
  (* Kompass *)
  TKompass    = class(TThalesBase)
  private
    { Private-Deklarationen }
    DampTimer  : integer;
    Display1   : TDigDisplay;
    Display2   : TDigDisplay;
    XStartP    : integer;
    YStartP    : integer;
    XAbsP      : integer;
    YAbsP      : integer;
    XStartL    : LongInt;
    YStartL    : LongInt;
    XAbsL      : LongInt;
    YAbsL      : LongInt;
    Disp1      : TRect;
    Disp2      : TRect;
    MittelP,
    Pointer1P,
    Pointer1L,
    Pointer1R,
    Pointer2P,
    Pointer2L,
    Pointer2R,
    Pointer1P0,
    Pointer1L0,
    Pointer1R0,
    Pointer2P0,
    Pointer2L0,
    Pointer2R0  : TPoint;
    FDamp      : Integer;
    painted    : Boolean;

    Procedure SetDamp(pValue : integer);
    procedure DampExpired(Sender: TObject);
    Procedure SetShape(pValue : TShapeTypeT);
    Procedure SetColKrRand(pValue : TColor);
    Procedure SetColKreis(pValue : TColor);
    Procedure SetColShape(pValue : TColor);
    Procedure SetColInnen(pValue : TColor);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColZRand(pValue : TColor);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColDisp1(pValue : TColor);
    Procedure SetColDisp2(pValue : TColor);
    Procedure SetKursIst(pValue : integer);
    Procedure SetKursSoll(pValue : integer);
    function  DispKurs(doDraw : boolean) : boolean;
    function  DispDigits(doDraw : boolean) : boolean;
    Procedure SetDispItalic(pValue : Boolean);
    procedure ScalePnt(XL,YL : longint; var XP,YP : integer);
  protected
```

```
    { Protected-Deklarationen }
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FShape     : TShapeTypeT;
    FColShape  : TColor;
    FColKrRand : TColor;
    FColKreis  : TColor;
    FColInnen  : TColor;
    FColPointer : TColor;
    FColZRand  : TColor;
    FColDBack  : TColor;
    FColDRand  : TColor;
    FColDisp1  : TColor;
    FColDisp2  : TColor;
    FKursIst   : integer;
    FKursSoll  : integer;
    FItalic    : Boolean;
    DampValue  : double;

    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property ondblClick;
    property onClick;
    property Font;
    property ColCircleBord : TColor  read FColKrRand write SetColKrRand;
    property ColCircle : TColor  read FColKreis  write SetColKreis;
    property ColInner  : TColor  read FColInnen  write SetColInnen;
    property ColShape  : TColor  read FColShape  write SetColShape;
    property ColPointer : TColor  read FColPointer write SetColPointer;
    property ColPointerBord  : TColor  read FColZRand  write SetColZRand;
    property ColDBack  : TColor  read FColDBack  write SetColDBack;
    property ColDBord  : TColor  read FColDRand  write SetColDRand;
    property ColDisp1  : TColor  read FColDisp1  write SetColDisp1;
    property ColDisp2  : TColor  read FColDisp2  write SetColDisp2;
    Property DispItalic: Boolean read FItalic    write SetDispItalic;
    property ColBack;
    property Shape     : TShapeTypeT read FShape  write SetShape;
    property CourseIs : integer read FKursIst   write SetKursIst;
    property CourseShould : integer read FKursSoll  write SetKursSoll;
    property Damping   : integer read FDamp      write SetDamp;
  end;
```

CtrlTimer

Zeitschaltuhr zum steuern von maschinellen Abläufen, wie z.B. Brennöfen, Füllvorgängen, Dosierungen usw. Fast beliebig konfigurierbar:
hhmmss, hhmm, mmss, hh, mm, ss.
Vier Schaltpunkte (Events), drei davon mit frei wählbarer Zeit, einer nach Ablauf auf 0.
Alle Parameter vom Anwender einstellbar.

Komponenten Deklaration

```
(* technical timer *)
type
  TCtrlTRes   = (hh, mm, ss, hhmm, mmss, hhmmss);
  tCtrlTMode  = (reset, run, stop, elapsed, trig1, trig2, trig3);

type
{ Forward declarations }
  TCtrlTimer = class;

  TCtlTime = class(TPersistent)
  private
    FControl   : TCtrlTimer;
    hours10    : integer;
    hours1     : integer;
    minutes10  : integer;
    minutes1   : integer;
    seconds10  : integer;
    seconds1   : integer;
    Procedure SetValHours(pValue : integer);
    Function  GetValHours : integer;
    Procedure SetValMinutes(pValue : integer);
    Function  GetValMinutes : integer;
    Procedure SetValSeconds(pValue : integer);
    Function  GetValSeconds : integer;
    constructor Create(AControl: TCtrlTimer);
  public
    ValChanged : boolean;
    isZero     : boolean;
    procedure ResetAll;
    procedure Decrement;
    Function  GetValue : integer;
    procedure Assign(Source: TPersistent); override;
  published
    property ValueHours   : integer read GetValHours   write SetValHours;
    property ValueMinutes : integer read GetValMinutes write SetValMinutes;
    property ValueSeconds : integer read GetValSeconds write SetValSeconds;
  end;

  TCtrlTimer  = class(TRahmen)
  private
    { Private-Deklarationen }
    Disp7      : TDigDisplay;
    Timer      : integer;
    BlinkTimer : integer;
    FStartTime : TCtlTime;
    FTrigger1  : TCtlTime;
    FTrigger2  : TCtlTime;
    FTrigger3  : TCtlTime;
    ElapsTime  : TCtlTime;
    ShowTime   : TCtlTime;
```

```
    OrgCursor  : HCursor;
    painted    : boolean;
    PosText    : TPoint;
    CtrlTBtns  : tCtrlTBtns;
    BtnRun,
    BtnStop,
    BtnRes,
    BtnTrig,
    TextFRect,
    TextR1,
    TextR2,
    TextR3,
    PanelRect,
    Btn1Up,
    Btn1Down,
    Btn2Up,
    Btn2Down,
    Btn3Up,
    Btn3Down,
    Btn4Up,
    Btn4Down,
    Btn5Up,
    Btn5Down,
    Btn6Up,
    Btn6Down    : TRect;
    FItalic     : Boolean;
    FShape      : TmShapeType;
    FCtrlRes    : TCtrlTRes;
    FMode       : tCtrlTMode;
    fColMode    : tColor;
    FColSegm    : TColor;
    FColElaps   : TColor;
    FCaption    : String10;
    FCaptHour   : String10;
    FCaptMin    : String10;
    FCaptSec    : String10;
    fColCapt    : tColor;
    fColTimes   : tColor;
    FOnStart    : TNotifyEvent;
    FOnElapsed  : TNotifyEvent;
    FOnTrigger1 : TNotifyEvent;
    FOnTrigger2 : TNotifyEvent;
    FOnTrigger3 : TNotifyEvent;
    FOnStop     : TNotifyEvent;
    FOnReset    : TNotifyEvent;

    procedure SetStart(pValue: TCtlTime);
    function  GetStart : TCtlTime;
    procedure SetTrig1(pValue: TCtlTime);
    function  GetTrig1 : TCtlTime;
    procedure SetTrig2(pValue: TCtlTime);
    function  GetTrig2 : TCtlTime;
    procedure SetTrig3(pValue: TCtlTime);
    function  GetTrig3 : TCtlTime;
    procedure TimerExpired(Sender: TObject);
    procedure Blinker(Sender: TObject);
    Procedure SetShape(pValue : TmShapeType);
    Procedure SetCtrlRes(pValue : TCtrlTRes);
    Procedure SetMode(pValue : tCtrlTMode);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetCaption(pValue : String10);
    Procedure SetCaptHour(pValue : String10);
    Procedure SetCaptMin(pValue : String10);
```

```
    Procedure SetCaptSec(pValue : String10);
    Procedure SetColSegm(pValue : TColor);
    Procedure SetColTimes(pValue : TColor);
    Procedure SetColElaps(pValue : TColor);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColMode(pValue : TColor);
    Procedure DrawCaption;
    procedure DrawButtons;
    procedure WMElapsed(Var Message : TMessage); message WM_DRAWVALUE;
  protected
    { Protected-Deklarationen }
    procedure Paint; override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
    Procedure DispTime;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    property ColBack;
    property StartTime : TCtlTime   read GetStart    write SetStart;
    property Trigger1  : TCtlTime   read GetTrig1    write SetTrig1;
    property Trigger2  : TCtlTime   read GetTrig2    write SetTrig2;
    property Trigger3  : TCtlTime   read GetTrig3    write SetTrig3;
    Property Resolution: TCtrlTRes  read FCtrlRes    write SetCtrlRes;
    Property Mode      : tCtrlTMode read FMode       write SetMode;
    Property Caption   : String10   read FCaption    write SetCaption;
    Property CaptHour  : String10   read FCaptHour   write SetCaptHour;
    Property CaptMin   : String10   read FCaptMin    write SetCaptMin;
    Property CaptSec   : String10   read FCaptSec    write SetCaptSec;
    Property DispItalic  : Boolean  read FItalic     write SetDispItalic;
    property Shape   : TmShapeType  read FShape      write SetShape;
    property ColSegm    : TColor    read FColSegm    write SetColSegm;
    property ColElapsed : TColor    read FColElaps   write SetColElaps;
    property ColTimes   : TColor    read FColTimes   write SetColTimes;
    property CaptCol    : TColor    read FColCapt    write SetColCapt;
    property ModeColor  : TColor    read FColMode    write SetColMode;
    property OnStart    : TNotifyEvent read FOnStart    write FOnStart;
    property OnElapsed  : TNotifyEvent read FOnElapsed  write FOnElapsed;
    property OnTrigger1 : TNotifyEvent read FOnTrigger1 write FOnTrigger1;
    property OnTrigger2 : TNotifyEvent read FOnTrigger2 write FOnTrigger2;
    property OnTrigger3 : TNotifyEvent read FOnTrigger3 write FOnTrigger3;
    property OnStop     : TNotifyEvent read FOnStop     write FOnStop;
    property OnReset    : TNotifyEvent read FOnReset    write FOnReset;
  end;
```

Poti10T

Dreh-Poti mit frei wählbaren Anfangs- und Endwerten, sowie Skalenteilung. Frei beschriftbar. Äusserer Knopf mit 270grad Weg, innerer mit 10 Gängen bzw. 3600Grad. Digitale Anzeige des eingestellten Werts. Preset Knopf.

Komponenten Deklaration

```
type
  (* Poti 10 turns *)
  TPoti10T     = class(TThalesBase)
  private
    { Private-Deklarationen }
    FOnChanging : TNotifyEvent;
    FOnChanged  : TNotifyEvent;
    FOnPreset   : TNotifyEvent;
    FDispEna    : Boolean;
    FValue      : double;
    FCaptEnable : Boolean;
    FPresEnable : Boolean;
    FColCaption : TColor;
    FColPreset  : TColor;
    FDragDelay  : Integer;
    FColPoint1  : TColor;
    FColPoint2  : TColor;
    FColScala   : TColor;
    FColButton  : TColor;
    FColLight   : TColor;
    FColShade   : TColor;
    FCaption    : String10;
    FScalaPart  : integer;
    FValueStart : double;
    FValueEnd   : double;
    FColDBack   : TColor;
    FColDRand   : TColor;
    FColDisp    : TColor;
    FItalic     : Boolean;
    FFixPnt     : Boolean;
    FLedOn1     : Boolean;
    FLedOn2     : Boolean;
    FLedBlink1  : Boolean;
    FLedBlink2  : Boolean;
    FColLed1    : TColor;
    FColLed2    : TColor;
    FLedEna     : Boolean;
    RepeatTimer : integer;
    Blinker     : integer;
    Display     : TDigDisplay;

    ValueSoll   : double;
    CircleVal   : double;
    CircleRad   : integer;
    MittelP,
    CircleC,
    CircleC0,
    PointerR,
    PointerR0,
    PointerL,
    PointerL0,
```

```
    PointerP,
    PointerP0   : TPoint;
    painted     : Boolean;
    TextPos     : TPoint;
    PresRect    : TRect;
    DispRect    : TRect;
    Rad1        : integer;
    innerBtn    : boolean;
    MouseX,
    MouseY      : Integer;
    changed     : boolean;
    preset      : boolean;
    LedSize     : integer;
    PosLed1     : TPoint;
    PosLed2     : TPoint;
    Led1Status  : boolean;
    Led2Status  : boolean;

    procedure TimerExpired(Sender: TObject);
    procedure Blink(Sender: TObject);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetDispItalic(pValue : Boolean);
    Procedure SetDispFixPnt(pValue : Boolean);
    procedure SetDragDelay(pValue: Integer);
    Procedure SetColDBack(pValue : TColor);
    Procedure SetColDRand(pValue : TColor);
    Procedure SetColPreset(pValue : TColor);
    Procedure SetColCaption(pValue : TColor);
    Procedure SetColPoint1(pValue : TColor);
    Procedure SetColPoint2(pValue : TColor);
    Procedure SetDispEna(pValue : Boolean);
    Procedure SetCaptEnable(pValue : Boolean);
    Procedure SetPresEnable(pValue : Boolean);
    Procedure SetColScala(pValue : TColor);
    Procedure SetColButton(pValue : TColor);
    Procedure SetColLight(pValue : TColor);
    Procedure SetColShade(pValue : TColor);
    Procedure SetValue(pValue : double);
    Procedure SetValueStart(pValue : double);
    Procedure SetValueEnd(pValue : double);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetCaption(pValue : string10);
    Procedure SetLedEna(pValue : Boolean);
    Procedure SetLedOn1(pValue : Boolean);
    Procedure SetLedOn2(pValue : Boolean);
    Procedure SetLedBlink1(pValue : Boolean);
    Procedure SetLedBlink2(pValue : Boolean);
    Procedure SetColLed1(pValue : TColor);
    Procedure SetColLed2(pValue : TColor);
    Procedure DrawLed1;
    Procedure DrawLed2;
    procedure DrawOrgBmp;
    Procedure DrawValue(doDraw : boolean);
  protected
    { Protected-Deklarationen }

    procedure Paint; override;
  public
    { Public-Deklarationen }

    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
```

```
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    procedure StyleChanged(Sender : TObject);
    property Font;
    Property FontAutom;
    property onChanging  : TNotifyEvent read FOnChanging write FOnChanging;
    property onChanged   : TNotifyEvent read FOnChanged  write FOnChanged;
    property onPreset    : TNotifyEvent read FOnPreset   write FOnPreset;
    property ColPointer1 : TColor   read FColPoint1  write SetColPoint1;
    property ColPointer2 : TColor   read FColPoint2  write SetColPoint2;
    property ColCaption  : TColor   read FColCaption write SetColCaption;
    property ColPreset   : TColor   read FColPreset  write SetColPreset;
    property ColScala    : TColor   read FColScala   write SetColScala;
    property ColButton   : TColor   read FColButton  write SetColButton;
    property ColLight    : TColor   read FColLight   write SetColLight;
    property ColShade    : TColor   read FColShade   write SetColShade;
    property Value       : double   read FValue      write SetValue;
    property ValueStart  : double   read FValueStart write SetValueStart;
    property ValueEnd    : double   read FValueEnd   write SetValueEnd;
    property ScalaPart   : integer  read FScalaPart  write SetScalaPart;
    property Caption     : string10 read FCaption    write SetCaption;
    property CaptEnable  : Boolean  read FCaptEnable write SetCaptEnable;
    property PresEnable  : Boolean  read FPresEnable write SetPresEnable;
    property DispEnable  : Boolean  read FDispEna    write SetDispEna;
    property ColDBack    : TColor   read FColDBack   write SetColDBack;
    property ColDRand    : TColor   read FColDRand   write SetColDRand;
    property ColDisplay  : TColor   read FColDisp    write SetColDisp;
    Property DispItalic : Boolean   read FItalic     write SetDispItalic;
    Property DispFixPnt : Boolean   read FFixPnt     write SetDispFixPnt;
    property DragDelay   : Integer  read FDragDelay  write SetDragDelay;
    Property ColLed1   : TColor   read FColLed1   write SetColLed1;
    Property ColLed2   : TColor   read FColLed2   write SetColLed2;
    Property LedEnable : Boolean  read FLedEna    write SetLedEna;
    Property LedOn1    : boolean  read FLedOn1    write SetLedOn1;
    Property LedOn2    : boolean  read FLedOn2    write SetLedOn2;
    Property LedBlink1 : boolean  read FLedBlink1 write SetLedBlink1;
    Property LedBlink2 : boolean  read FLedBlink2 write SetLedBlink2;
  end;
```

BoxLine

Anzeige Einheit zur Darstellung von binären Informationen z.B. Ports, Bytes etc. Horizontale und vertikale Darstellung. Reihenfolge der Bits umkehrbar. Jedes Bit ist mit einem separaten Bitmap darstellbar.

Komponenten Deklaration

```
type
  TBoxLBox = class(TPersistent)
  private
  protected
    { Protected-Deklarationen }
    Box         : TRect;
    mask        : byte;
    Clipped     : boolean;
    procedure   CreateGlyphs(gNum : integer);
    constructor Create(AControl: TBoxLine);
    destructor  Destroy; override;
  public
    { Public-Deklarationen }
    Procedure DrawBox;
    procedure DrawCaption;
    procedure Assign(Source: TPersistent); override;
  published
    { Published-Deklarationen }
    property Enabled     : boolean      read FEnable   write SetEnable;
    property Visible     : boolean      read FVisible  write SetVisible default
true;
    property Status      : boolean      read FStatus   write SetStatus;
    property CharOn      : char         read FChrOn    write SetChrOn;
    property CharOff     : char         read FChrOff   write SetChrOff;
    property CaptShort   : char         read FCaptShort write SetCaptShort;
    property CaptLong    : string       read FCaptLong write SetCaptLong;
    property NumGlyphs   : integer      read FNumGlyph write SetNumGlyph;
    property GlyphOn     : TBitMap      read GetBMPon  write SetBMPon;
    property GlyphOff    : TBitMap      read GetBMPoff write SetBMPoff;
  end;


  TBoxLine = class(TCustomControl)
  private
    { Private-Deklarationen }
    FOnClick0 : TNotifyEvent;
    FOnClick1 : TNotifyEvent;
    FOnClick2 : TNotifyEvent;
    FOnClick3 : TNotifyEvent;
    FOnClick4 : TNotifyEvent;
    FOnClick5 : TNotifyEvent;
    FOnClick6 : TNotifyEvent;
    FOnClick7 : TNotifyEvent;
    FOnClick  : TBoxLineEvent;
    FOnRepaint: TNotifyEvent;
    FToggle   : boolean;
    painted   : boolean;
    FOrient   : TOrientation;
    FOrderUp  : boolean;
    FCaptLeft : boolean;
    FEnaColor : TColor;
```

```
    FOnColor  : TColor;
    FDisColor : TColor;
    FColBgnd  : TColor;
    FFontEnaCol : TColor;
    FFontDisCol : TColor;
    FFontOnCol  : TColor;
    FBoxWidth : integer;
    FBoxSpace : integer;
    FEnaByte  : byte;
    FStatByte : byte;
    FBox0     : TBoxLBox;
    FBox1     : TBoxLBox;
    FBox2     : TBoxLBox;
    FBox3     : TBoxLBox;
    FBox4     : TBoxLBox;
    FBox5     : TBoxLBox;
    FBox6     : TBoxLBox;
    FBox7     : TBoxLBox;
    procedure SetBox0(pValue: TBoxLBox);
    function  GetBox0 : TBoxLBox;
    procedure SetBox1(pValue: TBoxLBox);
    function  GetBox1 : TBoxLBox;
    procedure SetBox2(pValue: TBoxLBox);
    function  GetBox2 : TBoxLBox;
    procedure SetBox3(pValue: TBoxLBox);
    function  GetBox3 : TBoxLBox;
    procedure SetBox4(pValue: TBoxLBox);
    function  GetBox4 : TBoxLBox;
    procedure SetBox5(pValue: TBoxLBox);
    function  GetBox5 : TBoxLBox;
    procedure SetBox6(pValue: TBoxLBox);
    function  GetBox6 : TBoxLBox;
    procedure SetBox7(pValue: TBoxLBox);
    function  GetBox7 : TBoxLBox;
    procedure SetCaptLeft(pValue : boolean);
    procedure SetOrderUp(pValue : boolean);
    procedure SetEnaColor(pValue : TColor);
    procedure SetDisColor(pValue : TColor);
    procedure SetOnColor(pValue : TColor);
    procedure SetFontEnaCol(pValue : TColor);
    procedure SetFontDisCol(pValue : TColor);
    procedure SetColBgnd(pValue : TColor);
    procedure SetFontOnCol(pValue : TColor);
    procedure SetBoxWidth(pValue : integer);
    procedure SetBoxSpace(pValue : integer);
    Procedure SetOrient(pValue : TOrientation);
    Procedure SetEnaByte(pValue : Byte);
    Procedure SetStatByte(pValue : Byte);
  protected
    { Protected-Deklarationen }
    LogRec    : TLOGFONT;
    OldFont   : HFont;
    NewFont   : HFONT;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property ColBgnd     : TColor       read FColBgnd  write SetColBgnd;
```

```
    property Orientation : TOrientation read FOrient   write SetOrient;
    property OrderUp     : boolean      read FOrderUp  write SetOrderUp;
    property CaptLeft    : boolean      read FCaptLeft write SetCaptLeft;
    property Toggle      : boolean      read FToggle   write FToggle;
    property BoxWidth    : integer      read FBoxWidth write SetBoxWidth;
    property BoxSpace    : integer      read FBoxSpace write SetBoxSpace;
    property ColorFontEna: TColor       read FFontEnaCol write SetFontEnaCol;
    property ColorFontDis: TColor       read FFontDisCol write SetFontDisCol;
    property ColorFontOn : TColor       read FFontOnCol  write SetFontOnCol;
    property ColorEnable : TColor       read FEnaColor write SetEnaColor;
    property ColorOn     : TColor       read FOnColor  write SetOnColor;
    property ColorDisable: TColor       read FDisColor write SetDisColor;
    property BoxEnaByte  : byte         read FEnaByte  write SetEnaByte;
    property BoxStatByte : byte         read FStatByte write SetStatByte;
    property Box0        : TBoxLBox     read GetBox0   write SetBox0;
    property Box1        : TBoxLBox     read GetBox1   write SetBox1;
    property Box2        : TBoxLBox     read GetBox2   write SetBox2;
    property Box3        : TBoxLBox     read GetBox3   write SetBox3;
    property Box4        : TBoxLBox     read GetBox4   write SetBox4;
    property Box5        : TBoxLBox     read GetBox5   write SetBox5;
    property Box6        : TBoxLBox     read GetBox6   write SetBox6;
    property Box7        : TBoxLBox     read GetBox7   write SetBox7;
    property OnBox0Click : TNotifyEvent read FOnClick0 write FOnClick0;
    property OnBox1Click : TNotifyEvent read FOnClick1 write FOnClick1;
    property OnBox2Click : TNotifyEvent read FOnClick2 write FOnClick2;
    property OnBox3Click : TNotifyEvent read FOnClick3 write FOnClick3;
    property OnBox4Click : TNotifyEvent read FOnClick4 write FOnClick4;
    property OnBox5Click : TNotifyEvent read FOnClick5 write FOnClick5;
    property OnBox6Click : TNotifyEvent read FOnClick6 write FOnClick6;
    property OnBox7Click : TNotifyEvent read FOnClick7 write FOnClick7;
    property OnClick     : TBoxLineEvent read FOnClick  write FOnClick;
    property OnRepaint   : TNotifyEvent read FOnRepaint write FOnRepaint;
  end;
```

BitWin

Kleines ToolWindow ähnlich „Hint„ zum Anzeigen von Mouse Cursor abhängigen Informationen. Beispiel in CurveDemo und Thales_T/TextScroll

Komponenten Deklaration

```
type
  TBitWinEvent =
    procedure(Sender : TObject; var sMousePos : TPoint;
                                var bwInfo : string;
                                var bwShow : boolean) of object;

  TbsOriginH   = (bwhLeft, bwhCenter, bwhRight);
  TbsOriginV   = (bwvTop, bwvCenter, bwvBottom);

type
  TCustomBitWin  = class(TCustomControl)
  private
    { Private-Deklarationen }
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    procedure FontChange(chFont : TFont);
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property Font;
    property Color;
    property OriginHor  : TbsOriginH   Read FOriginH  write SetOrgH;
    property OriginVert : TbsOriginV   Read FOriginV  write SetOrgV;
    Property Info       : string       Read FInfo     write SetInfo;
    property ShowDelay  : integer      Read FShowDel  write SetShowDel;
    Property Hide       : boolean      Read GetHide   write SetHide;
    Property OffSetX    : integer      read FOffsetX  write SetOffsX;
    Property OffSetY    : integer      read FOffsetY  write SetOffsY;
    property OnShow     : TBitWinEvent read FOnShow   write FOnShow;
    property OnHide     : TNotifyEvent read FOnHide   write FOnHide;
  end;


type
  TBitWin = class(TComponent)
  private
    { Private-Deklarationen }
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property Parent     : TWinControl  read GetParent write SetParent;
    property Font       : TFont        read FFont     write SetFont;
    property Color      : Tcolor       read FColor    write SetColor;
```

```
    Property Info       : string       Read FInfo     write SetInfo;
    property ShowDelay  : integer      Read FShowDel  write SetShowDel;
    Property Hide       : boolean      Read GetHide   write SetHide;
    Property OffSetX    : integer      read FOffsetX  write SetOffsX;
    Property OffSetY    : integer      read FOffsetY  write SetOffsY;
    property OriginHor  : TbsOriginH   Read FOriginH  write SetOrgH;
    property OriginVert : TbsOriginV   Read FOriginV  write SetOrgV;
    property OnShow     : TBitWinEvent read FOnShow   write SetOnShow;
    property OnHide     : TNotifyEvent read fOnHide   write SetOnHide;
  end;
```

Str Stroke

Vektor Zeichensatz zur Verwendung bei allen Canvas-Objekten. Beliebig rotierbar, skalierbar, spiegelbar, komfortable Align-Funktionen.

Komponenten Deklaration

```
CONST StrokeSize   = 23;
      StrokeMax    = 255;
      StrokeStrMax = 63;
TYPE
      TxyCoord    = record
                      svx : double;
                      svy : double;
                      pen : integer;
                      (* 0 -> penUp, 1 -> penDown, -1 -> ende *)
                    end;

      TStrokeChr  = array[0..StrokeSize] of TxyCoord;
      PStrokeChr  = ^TStrokeChr;

      TStrokeStr  = array[0..StrokeStrMax] of TStrokeChr;
      PStrokeStr  = ^TStrokeStr;

      TStrokeArr  = ARRAY[0..StrokeMax, 0..StrokeSize] OF BYTE;
      PStrokeArr  = ^TStrokeArr;

type
  TStrokeSet = class(TComponent)
  private
    { Private declarations }
    ABCextern : boolean;
  protected
    { Protected declarations }
  public
    { Public declarations }
    StrokeStr    : PStrokeStr;
    StrokeChr    : TStrokeChr;
    TextCursorX  : double;
    TextCursorY  : double;
    Function  SetAbc(extern : boolean) : boolean;
    Procedure GetStrokeChar(ch : Char; rot, sizeX, sizeY : double;
                            hor, vert : integer; mirr, ital : boolean);
    Procedure GetStrokeStr(TextStr : string; rot, sizeX, sizeY : double;
                           hor, vert : integer; mirr, ital : boolean);
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; virtual;
  end;

type
  TJustHor   = (stLeft, stCenter, stRight);
  TJustVert  = (stBottom, stMiddle, stTop);

type
  TStroke = class(TRahmen)
  private
    { Private declarations }
    StSet     : TStrokeSet;
```

```
    fSizeX    : word;
    fSizeY    : word;
    fRotate   : Integer;
    fMirror   : boolean;
    fItalic   : boolean;
    fJustHor  : TJustHor;
    fJustVert : TJustVert;
    fPosX     : Integer;
    fPosY     : Integer;
    fColor    : TColor;
    fBold     : word;
    fStrokeStr: string;
    Function  GetAbcTyp : boolean;
    Procedure SetAbcTyp(pValue : boolean);
    Procedure SetStrokeStr(pValue : string);
    Procedure SetRotate(pValue : integer);
    Procedure SetBold(pValue : word);
    Procedure SetSizeX(pValue : word);
    Procedure SetSizeY(pValue : word);
    Procedure SetMirror(pValue : boolean);
    Procedure SetItalic(pValue : boolean);
    Procedure SetJustHor(pValue : TJustHor);
    Procedure SetJustVert(pValue : TJustVert);
    Procedure SetPosX(pValue : Integer);
    Procedure SetPosY(pValue : Integer);
    Procedure SetColor(pValue : TColor);
  protected
    { Protected declarations }
    procedure Paint; override;
  public
    { Public declarations }
    constructor Create(AOwner: TComponent); override;
  published
    { Published-Deklarationen }
    property sString : string  read fStrokeStr  write SetStrokeStr;
    property FileABC   : boolean read GetAbcTyp   write SetAbcTyp;
    property SizeX     : word    read fSizeX      write SetSizeX;
    property SizeY     : word    read fSizeY      write SetSizeY;
    property Rotate    : integer read fRotate     write SetRotate;
    property Mirror    : boolean read fMirror     write SetMirror;
    property Italic    : boolean read fItalic     write SetItalic;
    property JustifyHor : TJustHor read fJustHor   write SetJustHor;
    property JustifyVert: TJustVert read fJustVert write SetJustVert;
    property PositionX : integer read fPosX       write SetPosX;
    property PositionY : integer read fPosY       write SetPosY;
    property Color     : TColor  read fColor      write SetColor;
    property Bold      : word    read fBold       write SetBold;
    property Status;
    property OnDown;
    property OnUp;
    property OnClick;
    Property ButtonStyle;
  end;


Function LoadStrokeSet(FName : string) : Boolean;
Function SaveStrokeSet(FName : string; intern : boolean) : Boolean;
```

Information

```
(* STROKE CHARACTER SET - Vektor Zeichensatz *)

   (* die oberen 4 Bits sind die "X"-Koordinate  (0..07, pen) *)
   (* die unteren 4 Bits sind die "Y"-Koordinate (0..11) *)
   (* Koordinaten Ursprung ist links unten *)
   (* Die erste Koordinate ist immer eine Cursor-Bewegung!! *)
   (* in Bit 3 der X-Koordinate ist Pen-Up/Pen-Down kodiert *)
   (* ist Bit 3 = 1, wird der Vektor zum naechsten Schnittpunkt *)
   (* nicht gezeichnet. *)

   (* BEISPIEL: *)
   (* $11, $61, $C6, $4A = *)
   (* $11 = fahre Cursor nach x=1 y=1 *)
   (* $61 = zeichne Linie von 1,1 nach x=6 y=1 *)
   (* $C6 = zeichne Linie von 6,1 nach x=4 y=6 *)
   (* $4A = fahre Cursor nach x=4 y=10 *)
```

RollBar

Kleine auf- und abrollbare Panels. Menueleiste und Buttons sowie Gesamtgrösse und Farben in weiten Grenzen verstellbar.

Komponenten Deklaration

```
type
  TRollBar = class(TCustomPanel)
  private
    { Private Variables }
    painted     : boolean;
    rolling     : boolean;
    TitleBar    : TRect;                { Title bar rectangle      }
    BtnSize     : integer;
    iRollSpeed  : integer;
    bDragging   : boolean;              { State variable           }
    HeightMin   : integer;

    CloseBtn    : TSpeedButton;
    RollDownBtn : TSpeedButton;
    RollUpBtn   : TSpeedButton;
    MenuBtn     : TSpeedButton;

    MouseP      : TPoint;

    FOnUpDown   : TNotifyEvent;
    FOnClose    : TNotifyEvent;
    FOnClickI   : TNotifyEvent;
    FCanClose   : boolean;
    FCanRoll    : boolean;
    FCanMove    : boolean;
    FColCaptBack: TColor;
    FColCaption : TColor;
    FCaption    : string;
    FDown       : boolean;
    FRollSpeed  : trRollSpeed;
    FCapHeight  : Integer;
    FHeightMax  : integer;              { Max height for rollup    }

    { Private Methods }
    procedure DrawTitleBar;
    procedure MoveDown;
    procedure MoveUp;
    procedure MouseDown(Sender: TObject; Button: TMouseButton;
                        Shift: TShiftState; X, Y: Integer);
    procedure MouseMove(Sender: TObject; Shift: TShiftState;
                          X, Y: Integer);
    procedure MouseUp(Sender: TObject; Button: TMouseButton;
                       Shift: TShiftState; X, Y: Integer);
    procedure RollDownBtnClick(Sender: TObject);
    procedure RollUpBtnClick(Sender: TObject);
    procedure CloseBtnClick(Sender: TObject);
    procedure MenuBtnClick(Sender: TObject);
    procedure SetCanClose(pValue : boolean);
    procedure SetColBack(pValue : TColor);
    procedure SetColCaption(pValue : TColor);
    procedure SetCaption(pValue : String);
    procedure SetDown(pValue : boolean);
```

```
    procedure SetCanRoll(pValue : boolean);
    procedure SetHeightMax(pValue : integer);
    procedure SetRollSpeed(pValue : trRollSpeed);
    procedure SetCapHeight(pValue : integer);
  protected
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property BevelInner;
    property BevelOuter;
    property BevelWidth;
    property BorderWidth;
    property BorderStyle;
    property Color;
    property ParentShowHint;
    property ShowHint;
    property Visible;
    property Enabled;
    property Ctl3D;
    property TabOrder;
    property TabStop;
    property HelpContext;
    property Align;
    property OnEnter;
    property OnExit;
    property OnClick;
    property OnDblClick;
    property PopupMenu;
    property DragMode;
    property DragCursor;
    property OnDragOver;
    property OnDragDrop;
    property OnEndDrag;
    property OnUpDown    : TNotifyEvent read FOnUpDown   write FOnUpDown;
    property OnClose     : TNotifyEvent read FOnClose    write FOnClose;
    property CanClose    : boolean     read FCanClose    write SetCanClose;
    property CanRoll     : boolean     read FCanRoll     write SetCanRoll;
    property CanMove     : boolean     read FCanMove     write FCanMove;
    property ColCaptBack : TColor      read FColCaptBack write SetColBack;
    property ColCaption  : TColor      read FColCaption  write SetColCaption;
    property Caption     : string      read FCaption     write SetCaption;
    property Down        : boolean     read FDown        write SetDown;
    property CapHeight   : integer     read FCapHeight   write SetCapHeight;
    property HeightMax   : integer     Read FHeightMax   write SetHeightMax;
    property RollSpeed   : trRollSpeed Read FRollSpeed   write SetRollSpeed;
  end;
```

PickBox

Identisch mit der Delphi-ListBox, besitzt aber für jeden String eine zusätzliche, beschriftbare Schalt- bzw. Infofläche.

Komponenten Deklaration

```
type
  TPickBox = class(TCustomListBox)
  private
  { }
    FBtnFont   : TFont;
    SaveFont   : TFont;
    FBtnWidth  : integer;
    FBtnCapts  : TStringList;
    procedure SetBtnCapts(pValue: TStringList);
    Procedure SetBtnFont(pValue : TFont);
    Procedure SetBtnWidth(pValue : integer);
  protected
  { }
  public
  { }
    procedure DrawItem(Index: Integer;
                       Rect: TRect; State: TOwnerDrawState); override;
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
  { }
    Property BtnFont  : TFont       read FBtnFont  write SetBtnFont;
    property BtnWidth : integer     read FBtnWidth write SetBtnWidth;
    property BtnCapts : TStringList read FBtnCapts write SetBtnCapts;
    property OnDrawItem;
    property Align;
    property BorderStyle;
    property Color;
    property Columns;
    property Ctl3D;
    property DragCursor;
    property DragMode;
    property Enabled;
    property ExtendedSelect;
    property Font;
    property IntegralHeight;
    property ItemHeight;
    property Items;
    property MultiSelect;
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property ShowHint;
    property Sorted;
    property TabOrder;
    property TabStop;
    property Visible;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
```

```
    property OnDragOver;
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMeasureItem;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;
```

ComPortT

Dialog zum Einstellen einer seriellen Schnittstelle. Übernimmt das komplette Handling von Status, Senden und Empfangen der seriellen Schnittstelle. XON-XOFF Protokoll, abschaltbar. Error handling

Komponenten Deklaration

```
  TComPortT = class(TComponent)
  private

  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    function OpenComPort : integer;
    procedure CloseComPort;
    function ComOk : boolean;
    function TransmBuf(const Buf; Count: Integer): integer;
    function ReceiveBuf(var Buf; Count: Integer): integer;
    function TransmStr(st :string): boolean;
    function ReceiveStr(var str : string; Count : word): integer;
    function TransmChr(ch : char): boolean;
    function ReceiveChr(var ch : char): boolean;
    function ReceiveStat: Integer;
    function TransmStat: Integer;
    procedure FlushComRx;
    procedure FlushComTx;
    {Comm escape functions}
    procedure ComSetDTR(State: Boolean);
    procedure ComSetRTS(State: Boolean);
    procedure ComSetBREAK(State: Boolean);
    procedure ComSetXON (State: Boolean);
    {Comm status flags}
    property ComGetCTS: Boolean index 1 read GetModemState;
    property ComGetDSR: Boolean index 2 read GetModemState;
    property ComGetRING: Boolean index 3 read GetModemState;
    property ComGetRLSD: Boolean index 4 read GetModemState;
    {DeviceHandle property}
    property ComGetHandle: THandle read FHandle;
    property ProviderSubtype: Integer read GetProviderSubtype;
  puplished
    property ShowErrors : boolean read fShowErr write fShowErr;
    property ComDialog : boolean read FVisible  write SetVisible;
    property ComName: string read FComName write SetComName;
    property ComAvail : TCommAvail read fCommAvail write fCommAvail;
    property Events: TCommEventType read FEvents write SetEvents;
    property BaudRate: TBaudRate read FBaudRate write SetBaudRate default
                                                              bd9600;
    property Parity: TParity read FParity write SetParity default parNone;
    property Stopbits: TStopbits read FStopbits write SetStopbits default stp1;
    property DataLength: TDataLength read FDataLength write SetDataLength
                                                              default dl8;
    property RxBufSize: Integer read FRxBufSize write SetRxBufSize default 4096;
    property TxBufSize: Integer read FTxBufSize write SetTxBufSize default 2048;
    property RxTimeOut: Integer read FRxTimeOut write SetRxTimeOut default 250;
    property Options: TCommOptions read FOptions write SetOptions;
    property FlowControl: TFlowControl read FFlowControl write SetFlowControl
                                                          default fcDefault;
    property ErrorChar : Char read dcbErrorChar write SetdcbErrorChar;
```

```
    property EventChars: TCommEventChars read FEventChars;
    property OnBreak: TNotifyEvent read FOnBreak write FOnBreak;
    property OnCTS: TNotifyEvent read FOnCTS write FOnCTS;
    property OnDSR: TNotifyEvent read FOnDSR write FOnDSR;
    property OnRing: TNotifyEvent read FOnRing write FOnRing;
    property OnRLSD: TNotifyEvent read FOnRLSD write FOnRLSD;
    property OnError: TCommErrorEvent read FOnError write FOnError;
    property OnRxChar: TCommRxCharEvent read FOnRxChar write FOnRxChar;
    property OnRxFlag: TNotifyEvent read FOnRxFlag write FOnRxFlag;
    property OnTxEmpty: TNotifyEvent read FOnTxEmpty write FOnTxEmpty;
  end;
```

Prozeduren, Funktionen und Properties

ComName

```
property ComName : string;
```

Name der Schnittstelle, Com1, Com2 etc. Kann nur beschrieben werden, wenn das Port geschlossen ist.

Events

```
TCommEventState = (evBreak, evCTS, evDSR, evError, evRing, evRlsd, evRxChar,
                   evRxFlag, evTxEmpty);
TCommEventType = set of TCommEventState;

property MonitorEvents: TCommEventType;
```

Freigabe oder Sperrung verschiedener Schnittstellen Ereignisse

```
evBREAK    - Ein break wurde am Eingang festgestellt.
evCTS      - Das CTS (clear-to-send) Signal hat seinen Status geändert
evDSR      - Das DSR (data-set-ready) Signal hat seinen Status geändert
evERROR    - Ein line-status Fehler ist aufgetreten. Mögliche Fehler:
             CE_FRAME, CE_OVERRUN, und CE_RXPARITY.
evRING     - Ein Ring Signal wurde empfangen
evRLSD     - Das RLSD (receive-line-signal-detect) Signal hat sich geändert
evRXCHAR   - Ein Zeichen wurde empfangen und ist im RxBuffer
evRXFLAG   - Das Event Zeichen wurde empfangen und ist im RxBuffer
             Das Event Zeichen wird durch das property EventChars spezifiziert
             und durch Options Freigegeben bzw. gesperrt.
evTXEMPTY - Das letzte Zeichen des TxBuffers wurde abgeschickt
```

BaudRate

```
TBaudRate = (bd110, bd300, bd600, bd1200, bd2400, bd4800, bd9600, bd14400,
             bd19200, bd38400, bd56000, bd57600, bd115200, bd128000, bd256000);

property BaudRate: TBaudRate;
```

Parity

```
TParity = (parNone, parOdd, parEven, parMark, parSpace);

property Parity: TParity;
```

StopBits

```
TStopbits = (stp1, stp1p5, stp2);
```

property Stopbits: TStopbits;

DataSize

TDataSize = (ds4, ds5, ds6, ds7, ds8);

property Databits: TDatabits;

RxBufSize

property RxBufSize: Integer;

Spezifiziert die erforderliche interne Empfangsbuffer Länge in Bytes.

TxBufSize

property TxBufSize: Integer;

Spezifiziert die erforderliche interne Sendebuffer Länge in Bytes.

RxTimeOut

property RxTimeout: Integer;

Spezifiziert die maximale Zeit in Millisekunden, die verstreichen darf wenn eine Prozedur auf mehrere Zeichen wartet (ReceiveBuf, ReceiveStr.
Während eine Wartevorgangs beginnt das TimeOut nachdem das erste Zeichen empfangen wurde. Wenn die Zeit bis zum nächsten Zeichen das TimeOut überschreitet, wird die Leseoperation abgebrochen und die Funktion kehrt mit den bisjetzt empfangenen Zeichen zurück.
Ist RxTimeOut = 0 kehrt die Lese-Funktion sofort zurück.

Options

TCommOption = (oParityCheck, oDsrSensitivity, oIgnoreXOff, oErrorChar,
oNullStrip);
TCommOptions = set of TCommOption;

property Options: TCommOptions;

oParityCheck:
Gibt Paritycheck frei. Parity Fehler werden durch den OnError Event angezeigt.

oDsrSensitivity:
Wenn freigegeben, reagiert das Port auf das DSR-Signal. Dabei werden nur Zeichen akzeptiert, wenn die DSR-Leitung high ist.

oIgnoreXoff:
Spezifiziert ob die Übertragung stoppt, wenn der RxBuffer voll ist und ein XOff byte geschickt werden soll.
Ist dieser Schalter true, wird mit der Übertragung erst weitergefahren, nachdem wieder ausreichend Platz im RxBuffer ist und ein XOff gesendet wurde.
Ist dieser Schalter false, wird mit der Übertragung erst weitergefahren, nachdem wieder ausreichend Platz im RxBuffer ist und ein XOn gesendet wurde.

oErrorChar:
Bestimmt ob Zeichen, die mit einem Parity Fehler empfangen wurden, durch das Zeichen ersetzt werden soll, das durch das property ErrorChar definiert ist.
Wenn ErrorChar = TRUE und ein Parity Fehler ist aufgetreten, wird das empfangene Zeichen durch ErrorChar ersetzt.

oNullStrip:
Wenn true, werden Null-Zeichen (#0) ignoriert beim Empfang.

EventChars

XonChar
property XonChar : char;
Spezifiziert das XON-Zeichen sowohl für Transmit als auch für Receive.

XoffChar
property XoffChar : char;
Spezifiziert das XOFF-Zeichen sowohl für Transmit als auch für Receive.

FlowControl

TFlowControl = (fcNone, fcCTS, fcDTR, fcSoftware, fcDefault);

property FlowControl: TFlowControl;

fcNone	No flowcontrol
fsCTS	Rts/Cts flowcontrol
fsDTR	Specifies the DTR (data-terminal-ready) flow control.
fsSoftware	Xon/Xoff flowcontrol
fcDefault	Use system settings

COMavail

function COMavail(port : string) : boolean;

Gibt den Device Status des aktuellen Ports zurück, true=Open, false=Closed

COMok

function COMok : Boolean;

Gibt den Device Status des aktuellen Ports zurück, true=Open, false=Closed

OpenComPort

function OpenComPort;

Eröffnet das Device **ComName** und konfiguriert es. OpenComPort **muss** aufgerufen werden, bevor auf das Port zugegriffen werden kann.
Wenn ein nicht vorhandenes oder belegtes Port geöffnet werden soll, wird ein Fehlercode zurückgegeben.

2	File not found.
3	Invalid file name.
4	Too many open files.
5	Access denied.
100	EOF.
101	Disk full.
106	Invalid input.

CloseComPort

procedure CloseComPort;

Schliesst das geöffnete Device/Port.

ReceiveStat

function ReceiveStat : Integer;

Anzahl der Bytes im ReceiveBuffer.

TransmStat
function TransmStat : Integer;

Anzahl der Bytes im TransmitBuffer.

TransmBuf
function TransmBuf(const Buf; Count: Integer): integer;

Buf ist eine beliebige Variable. TransmBuf schreibt Count Bytes in das ComPort, beginnend mit dem ersten Zeichen in Buf. Die tatsächliche Anzahl von Bytes, die gesendet wurden (<= Count) wird als Ergebniss zurückgegeben. Im Fehlerfall wird eine -1 zurückgegeben.

ReceiveBuf
function ReceiveBuf(var Buf; Count: Integer): integer;

Buf ist eine beliebige Variable. ReceiveBuf liest Count Bytes (oder weniger) von ComPort in Buf. Die tatsächliche Anzahl von Bytes, die gelesen wurden (<= Count) wird als Ergebniss zurückgegeben. Im Fehlerfall wird eine -1 zurückgegeben.
Der Parameter count darf nicht frei gewählt werden. Er muss von RxCharEvent oder ReceiveStat übernommen werden.

TransmStr
function TransmStr(st : string) : boolean;

TransmStr schreibt den String st in den Sendebuffer. Im Fehlerfall wird false zurückgegeben.

ReceiveStr
function ReceiveStr(var str : string; Count : word): integer;

ReceiveStr liest Count Bytes (oder weniger) von ComPort in str. Die tatsächliche Anzahl von Bytes, die gelesen wurden (<= Count) wird als Ergebniss zurück-gegeben. Im Fehlerfall wird eine -1 zurückgegeben.
Der Parameter count darf nicht frei gewählt werden. Er muss von RxCharEvent oder ReceiveStat übernommen werden.

TransmChr
function TransmChr(ch : char): boolean;

TransmChr schreibt das Zeichen ch in den Sendebuffer. Im Fehlerfall wird false zurückgegeben.

ReceiveChr
function ReceiveChr(var ch : char): boolean;

ReceiveChr liest ein Zeichen aus dem Receivebuffer. Ist die Operation fehlgeschlagen, wird false zurückgegeben.

ACHTUNG:
Für alle Lese-Operationen gilt: Wird die Funktion von einem Event aus aufgerufen so muss im Event selbst nachgeprüft werden, ob der Empfangsbuffer leer ist, denn der nächste Event tritt erst wieder bei einem neuen Zeichen auf!!! Kommen keine

neuen Zeichen mehr herein, so tritt auch kein Event mehr auf. Es ist auch möglich den ReceiveStat zyklisch zu pollen.

FlushComRx

procedure FlushComRx;

Entferne alle Zeichen aus dem Receivebuffer

FlushComTx

procedure FlushComTx;

Entferne alle Zeichen aus dem TransmitBuffer

ComSetDTR

procedure ComSetDTR(State: Boolean);

False - Setzt die DTR (data-terminal-ready) Leitung zurück.
True - Setzt die DTR (data-terminal-ready) Leitung.

ComSetRTS

procedure ComSetRTS(State: Boolean);

False - Setzt die RTS (request-to-send) Leitung zurück.
True - Setzt die RTS (request-to-send) Leitung.

ComSetBreak

procedure ComSetBREAK(State: Boolean);

False - Nimmt die Übertragung wieder auf und setzt die TxData Leitung in
den normal Status.
True - Sperrt die Übertragung und setzt die TxData Leitung in den Break Zustand

ComSetXON

procedure ComSetXON(State: Boolean);

False - Simulation eines XOFF Empfangs
True - Simulation eines XON Empfangs

ShowErrors

property ShowErrors : boolean;

Einige Operationen erzeugen Fehlermeldungen, die dem User angezeigt werden. Diese Meldungen können mit ShowErrors abgeschaltet werden.

ComDialog

property ComDialog : boolean;

Hiermit kann der Standard Windows Dialog für serielle Schnittstellen angezeigt werden. Die dargestellten Parameter werden von der Komponente übergeben. Werden vom User Änderungen im Dialog vorgenommen, so werden diese von der Komponente wiederum übernommen.

ComGetCTS

property ComGetCTS: Boolean;

True - Die CTS (clear-to-send) Leitung ist aktiv

ComGetDSR
property ComGetDSR: Boolean;

True - Die DSR (data-set-ready) Leitung ist aktiv

ComGetRING
property ComGetRING: Boolean;

True - Die RING Leitung ist aktiv

ComGetRING
property ComGetRLSD: Boolean;

True - Das RLSD (receive-line-signal-detect) Signal ist aktiv

ComGetHandle
property ComGetHandle: THandle;

Reference auf das interne Devicehandle (Readonly property)

ProviderSubtype
property ProviderSubtype: Integer;

Device Id. Mit Hilfe der Funktion **GetProviderSubTypeName** erhält man den Namen des des tatsächlich verwendeten Devices.

Mögliche provider subtypes:

PST_FAX	FAX device
PST_LAT	LAT protocol
PST_MODEM	Modem device
PST_NETWORK_BRIDGE	Unspecified network bridge
PST_PARALLELPORT	Parallel port
PST_RS232	RS-232 serial port
PST_RS422	RS-422 port
PST_RS423	RS-423 port
PST_RS449	RS-449 port
PST_SCANNER	Scanner device
PST_TCPIP_TELNET	TCP/IP Telnet® protocol
PST_UNSPECIFIED	Unspecified
PST_X25	X.25 standards

OnBreak
property OnBreak: TNotifyEvent;

Ein Break Signal wurde empfangen.

OnCTS
property OnCTS: TNotifyEvent;

Die CTS (clear-to-send) Leitung hat sich geändert.

OnDSR
property OnDSR: TNotifyEvent;

Die DSR (data-set-ready) Leitung hat sich geändert.

OnRing

property OnRing: TNotifyEvent;

Ein Ring Signal wurde festgestellt.

OnRLSD

property OnRLSD: TNotifyEvent;

Die RLSD (receive-line-signal-detect) Leitung hat sich verändert.

OnError

TCommErrorEvent = procedure(Sender: TObject; Errors: Integer) of object;

property OnError: TCommErrorEvent;

Der Übertragungsstatus hat sich geändert.

CE_BREAK	Die Hardware hat ein Break festgestellt.
CE_DNS	Das Parallelport (Printer) meldet : not selected.
CE_FRAME	Ein Framing error wurde festgestellt.
CE_IOE	Ein I/O error ist aufgetreten.
CE_MODE	Die angeforderte Betriebsart ist nicht möglich
CE_OOP	Das Printerport meldet : out of paper.
CE_OVERRUN	Ein Buffer overrun ist aufgetreten
CE_PTO	Ein Printer TimeOut ist aufgetreten.
CE_RXOVER	Ein Receive buffer overflow ist aufgetreten.
CE_RXPARITY	Ein Parity Error ist aufgetreten.
CE_TXFULL	Der Transmit Buffer ist voll.

OnRxChar

TCommRxCharEvent = procedure(Sender: TObject; Count: Integer) of object;

property OnRxChar: TCommRxCharEvent;

Ein Zeichen wurde empfangen und steht im RxBuffer. Count ist die totale Anzahl der im Buffer wartenden Bytes.

OnRxFlag

property OnRxFlag: TNotifyEvent;

Das Event Zeichen (z.B. XON/XOFF) wurde empfangen und steht im Empfangsbuffer.

OnRxFlag

property OnTxEmpty: TNotifyEvent;

Das letzte Zeichen im Sendebuffer wurde abgeschickt.

ComPort

Dialog zum Einstellen bzw. Auswahl von **4 Schnittstellen**. Übernimmt das komplette Handling von Status, Senden und Empfangen der 4 seriellen Schnittstellen. XON-XOFF Protokoll, abschaltbar.

Nicht für Neuentwicklungen! Verwenden Sie besser **TComPortT**

Komponenten Deklaration

```
type
  TComPort = class(TComponent)
  private
    RepeatTimer: integer;
    FOnCom1Rx  : TComRxEvent;
    FOnCom2Rx  : TComRxEvent;
    FOnCom3Rx  : TComRxEvent;
    FOnCom4Rx  : TComRxEvent;
    FEvent1Ena : Boolean;
    FEvent2Ena : Boolean;
    FEvent3Ena : Boolean;
    FEvent4Ena : Boolean;
    FBuffLen   : LongInt;
    FCOM1Avail : boolean;
    FCOM2Avail : boolean;
    FCOM3Avail : boolean;
    FCOM4Avail : boolean;
    FCOM1InUse : boolean;
    FCOM2InUse : boolean;
    FCOM3InUse : boolean;
    FCOM4InUse : boolean;
    FVisible1  : boolean;
    FVisible2  : boolean;
    FVisible3  : boolean;
    FVisible4  : boolean;
    ComPortArr : ComPortArrTy;
    DCB1       : TDCB;
    DCB2       : TDCB;
    DCB3       : TDCB;
    DCB4       : TDCB;

    procedure TimerExpired(Sender: TObject);
    procedure SetBuffLen(pValue : LongInt);
    Function  GetPort(ComPort : string) : Integer;
    Procedure SetBaud1(pValue : LongInt);
    function  GetBaud1 : LongInt;
    Procedure SetBaud2(pValue : LongInt);
    function  GetBaud2 : LongInt;
    Procedure SetBaud3(pValue : LongInt);
    function  GetBaud3 : LongInt;
    Procedure SetBaud4(pValue : LongInt);
    function  GetBaud4 : LongInt;
    function  GetXonoff1 : boolean;
    Procedure SetXonoff1(pValue : boolean);
    function  GetXonoff2 : boolean;
    Procedure SetXonoff2(pValue : boolean);
    function  GetXonoff3 : boolean;
    Procedure SetXonoff3(pValue : boolean);
```

```
    function  GetXonoff4 : boolean;
    Procedure SetXonoff4(pValue : boolean);
    Procedure SetVisible1(pValue : boolean);
    Procedure SetVisible2(pValue : boolean);
    Procedure SetVisible3(pValue : boolean);
    Procedure SetVisible4(pValue : boolean);
    Procedure SetCOM1Avail(pValue : boolean);  (* Dummy *)
    Procedure SetCOM2Avail(pValue : boolean);  (* Dummy *)
    Procedure SetCOM3Avail(pValue : boolean);  (* Dummy *)
    Procedure SetCOM4Avail(pValue : boolean);  (* Dummy *)
    Procedure SetCOM1InUse(pValue : boolean);  (* Dummy *)
    Procedure SetCOM2InUse(pValue : boolean);  (* Dummy *)
    Procedure SetCOM3InUse(pValue : boolean);  (* Dummy *)
    Procedure SetCOM4InUse(pValue : boolean);  (* Dummy *)
  public
    { Public-Deklarationen }
    Procedure CheckPorts;
    Function  ComPortOk(ComPort : string) : boolean;
    Function  ComStatus(ComPort : string; var ComStat : TComStat) : Integer;
    Function  OpenComPort(ComPort : string) : boolean;
    Function  TransmChr(ComPort : string; ch : Char): Boolean;
    Function  ReceiveCh(ComPort : string; var ch : Char) : Boolean;
    Function  ReceiveStat(ComPort : string) : integer;
    Function  TransmStr(ComPort : string; str : string): integer;
    Function  ReceiveStr(ComPort : string; len : word; var str : string) :
integer;
    Function  ComSetDTR(ComPort : string; OnOff : Boolean): Boolean;
    Function  ComSetRTS(ComPort : string; OnOff : Boolean): Boolean;
    Function  ComGetCTS(ComPort : string): Boolean;
    Function  ComGetDSR(ComPort : string): Boolean;
    Procedure CloseComPort(ComPort : string);
    Procedure FlushComRx(ComPort : string);
    Procedure FlushComTx(ComPort : string);
    Procedure ResetAllPorts;
    Procedure GetDCB(ComPort : string; var DCB : TDCB);
    Procedure SetDCB(ComPort : string; DCB : TDCB);
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    property Com1Dialog : boolean read FVisible1   write SetVisible1;
    property Com2Dialog : boolean read FVisible2   write SetVisible2;
    property Com3Dialog : boolean read FVisible3   write SetVisible3;
    property Com4Dialog : boolean read FVisible4   write SetVisible4;
    property BaudRate1  : LongInt read GetBaud1    write SetBaud1;
    property BaudRate2  : LongInt read GetBaud2    write SetBaud2;
    property BaudRate3  : LongInt read GetBaud3    write SetBaud3;
    property BaudRate4  : LongInt read GetBaud4    write SetBaud4;
    property COM1InUse  : boolean read FCOM1InUse  write SetCOM1InUse;
    property COM2InUse  : boolean read FCOM2InUse  write SetCOM2InUse;
    property COM3InUse  : boolean read FCOM3InUse  write SetCOM3InUse;
    property COM4InUse  : boolean read FCOM4InUse  write SetCOM4InUse;
    property COM1Avail  : boolean read FCOM1Avail  write SetCOM1Avail;
    property COM2Avail  : boolean read FCOM2Avail  write SetCOM2Avail;
    property COM3Avail  : boolean read FCOM3Avail  write SetCOM3Avail;
    property COM4Avail  : boolean read FCOM4Avail  write SetCOM4Avail;
    property XOnOff1    : boolean read GetXOnOff1  write SetXOnOff1;
    property XOnOff2    : boolean read GetXOnOff2  write SetXOnOff2;
    property XOnOff3    : boolean read GetXOnOff3  write SetXOnOff3;
    property XOnOff4    : boolean read GetXOnOff4  write SetXOnOff4;
    property BufferLen  : Longint read FBuffLen    write SetBuffLen;
    property Event1Ena  : boolean read FEvent1Ena write FEvent1Ena;
    property Event2Ena  : boolean read FEvent2Ena write FEvent2Ena;
```

```
    property Event3Ena  : boolean read FEvent3Ena write FEvent3Ena;
    property Event4Ena  : boolean read FEvent4Ena write FEvent4Ena;
    property OnCom1Rx: TComRxEvent read FOnCom1Rx write FOnCom1Rx;
    property OnCom2Rx: TComRxEvent read FOnCom2Rx write FOnCom2Rx;
    property OnCom3Rx: TComRxEvent read FOnCom3Rx write FOnCom3Rx;
    property OnCom4Rx: TComRxEvent read FOnCom4Rx write FOnCom4Rx;
  end;
```

Beschreibung der Schnittstellen Komponenten ComPort

ComPort dient zum Einstellen der vorhandenen System-Schnittstellen und zum Senden bzw. Empfangen von Zeichen ueber die ausgewaehlte Schnittstelle.

ComPort besteht aus einem nichtsichtbaren Teil, der immer aktiv und damit immer ansprechbar ist. Dieser Teil ist das Interface zwischen Windows und dem Anwendungsprogramm.

Der sichtbare Teil, der im Normalfall abgeschaltet ist (property 'COMxDialog'), stellt dem Anwender einen Einstell-Dialog zur Verfuegung.

Ein nicht vorhandenes, oder von Windows benutztes oder durch einen anderen Treiber belegtes Port wird im Property 'COMxAvail' als false gekennzeichnet.

Ein durch ComPort selbst belegtes Port wird durch das property 'COMxInUse' = true gekennzeichnet.
Die Standard Einstellung der seriellen Schnittstellen sind: 8 Databits, 2 Stopbits, keine Parität und keine Hardware Handshake.

Optional kann ein Software Handshake „Xon'Xoff“ eingestellt werden. Die Handshake Leitungen „DTR“ und „RTS“ koennen per property gesetzt bzw. rueckgesetzt werden. Die Handshake Leitungen „CTS“ und „DSR“ koennen jederzeit abgefragt werden. Somit ist eine zusaetzliche Steuerungs-moeglichkeit fuer das angeschlossene Geraet vorhanden.

Eigenschaften/Properties:

```
    property Com1Dialog  : boolean;
    property Com2Dialog  : boolean;
    property Com3Dialog  : boolean;
    property Com4Dialog  : boolean;
      Oeffnet einen Schnittstellen Dialog zum interaktiven Einstellen der
      Schnittstellen Parameter.

    property BaudRate1  : Integer;
    property BaudRate2  : Integer;
    property BaudRate3  : Integer;
    property BaudRate4  : Integer;
      Baudrate für die Schnittstellen COM1..COM4. Werte als Integer, z.B. 9600
      oder 19200.

    property COM1Avail  : boolean;
    property COM2Avail  : boolean;
    property COM3Avail  : boolean;
    property COM4Avail  : boolean;
      Read Only. Zeigt an, ob die jeweilige serielle Schnittstelle noch frei ist
      bzw. von Windows  nicht belegt ist. Durch die Komponente selbst belegte
```

Ports werden in „ComxAvail“ nicht beruecksichtigt. Diese werden weiterhin als „available“ angezeigt.

property ***COM1InUse*** : boolean;
property ***COM2InUse*** : boolean;
property ***COM3InUse*** : boolean;
property ***COM4InUse*** : boolean;
Read Only. Zeigt an, ob die jeweilige serielle Schnittstelle von der Komponente selbst ueber „OpenCom“ belegt ist.

property ***XOnOff1*** : boolean;
property ***XOnOff2*** : boolean;
property ***XOnOff3*** : boolean;
property ***XOnOff4*** : boolean;
Stellt das XON-XOFF Protokoll ein. Bei true wird das Senden von Zeichen durch die zugehörige Schnittstelle unterbrochen, wenn der Empfänger das ASCII-Zeichen „XOFF“ geschickt hat. Das Senden wird wieder aufgenommen, wenn der Empfänger das Zeichen „XON“ gesendet hat. Dieses Protokoll ist nur bei reiner ASCII-Übertragung sinnvoll.

property ***BufferLen*** : Integer;
Stellt die Länge der Empfangs und Sendebuffer ein. Default : 1000 Zeichen.

property ***Event1Ena*** : boolean;
property ***Event2Ena*** : boolean;
property ***Event3Ena*** : boolean;
property ***Event4Ena*** : boolean ;
Sperrt den Eventx (OnComxRx) temporaer.

property ***OnCom1Rx***: TComRxEvent;
property ***OnCom2Rx***: TComRxEvent;
property ***OnCom3Rx***: TComRxEvent;
property ***OnCom4Rx***: TComRxEvent;
Dieses Ereignis tritt auf, wenn die jeweilige Schnittstelle ein Zeichen im Empfangsbuffer hat. In der Behandlungs Routine des Events wird normalerweise die Function „ReceiveCh“ oder „ReceiveStr“ aufgerufen. Gibt ReceiveCh ein true zurück, enthält „ch“ das empfangene Zeichen. Bei false **muss** unbedingt noch der Schnittstellenstatus mittels „ComStatus“ abgefragt werden, um ein evtl. vorhandener Fehler zurückzusetzen. ComStatus kann immer abgefragt werden. Der Event uebergibt als zusaetzlichen Parameter in "RxCount" die Anzahl der vorhandenen Zeichen.

Methoden/Prozeduren:

Function ***OpenComPort***(ComPort : string) : boolean;
Ein mittels „ComPortOk“ oder „ComxAvail“ als frei gemeldetes Port muss, um damit arbeiten zu können, mit „OpenComPort“ eröffnet werden. Zuvor **muss** allerdings zumindest das XonXoff protokoll und die Baudrate korrekt eingestellt werden. DTR und RTS können auch während des Arbeitens mit dem Port verändert werden. OpenComPort belegt das entsprechende Port und ComxInUse meldet es ab jetzt als belegt. Ganz spezielle Funktionen oder Verhaltensweise der Schnittstelle können vor dem öffnen im zugehörigen DCB- Parameterblock eingestellt werden (GetDCB und SetDCB). Eine interaktive Einstellung ist auch mit dem property ComxDialog moeglich. Der Aufbau des DCB ist in der Delphi-Hilfe unter DCB oder TDCB. Ein mit OpenComPort geöffnetes Port muss immer, spätestens bei Programm Ende mit CloseComPort wieder geschlossen werden.

Function ***TransmCh***r(ComPort : string; ch : Char): Boolean;
Schickt ein Zeichen über die Schnittstelle „ComPort“ an den angeschlossenen Empfänger. Die Schnittstelle muss, bevor das erste Mal ein Zeichen geschickt werden kann, als frei gemeldet worden und mittels OpenComPort geöffnet worden sein. Um sicher zu sein, dass das Zeichen auch tatsächlich übertragen wurde, ist es ratsam, wenn die Funktion false zurückliefert, mit „ComStatus“ die Ursache dafür herauszufinden.

Function ***ReceiveCh***(ComPort : string; var ch : Char) : Boolean;
Wenn die Funktion true ist, befindet sich in „ch“ der empfangene Character. Diese Funktion muss, falls Events (OnComxRx) programmiert wurden, innerhalb der Event-Prozedur aufgerufen werden.

Function ***ReceiveStat***(ComPort : string) : integer;
Der Result gibt die Anzahl der im Empfangsbuffer stehenden Zeichen an.

Function ***TransmStr***(ComPort : string; str : string): integer;
Mit dieser Funktion kann ein ganzer String übertragen werden.

Function ***ReceiveStr***(ComPort : string; len : word; var str : string) : integer;
Die Funktion gibt im result integer die Anzahl der empfangenen Zeichen zurueck = Length(str).

Function ***ComStatus***(ComPort : string; var ComStat : TComStat) : Integer;
Die Funktion liefert einen Integer Wert zurück der folgendermassen interpretiert werden muss (siehe auch „Instrum.dpr“ Unit „FComPort.pas“) : Jedes einzelne Bit hat eine Fehlerbedeutung und muss mittels der in Delphi vordefinierten Masken ausmaskiert werden. Ist Integer = 0, so ist kein Fehler aufgetreten, ist Integer = -1, dann ist das Port nicht eroeffnet.. Ansonsten gilt

CE_BREAK	Die Hardware stellte eine Abbruchbedingung fest.
CE_CTSTO	Clear-to-send-Unterbrechung (time out). CTS war für die durch das Element fCtsHold der TCOMSTAT -Struktur festgelegte Dauer gelöscht, während gleichzeitig versucht wurde, ein Zeichen zu übertragen.
CE_DNS	Die parallele Schnittstelle war nicht gewählt/selektiert.
CE_DSRTO	Data-set-ready-Unterbrechung. DSR war für die durch das Element fDsrHold der TCOMSTAT-Struktur festgelegte Dauer gelöscht, während gleichzeitig versucht wurde, ein Zeichen zu übertragen.
CE_FRAME	Die Hardware stellt einen Framing-Error fest.
CE_IOE	Ein I/O-Error trat während des Versuchs der Kommunikation mit einer parallelen Schnittstelle auf.
CE_MODE	Der gewünschte Modus wird nicht unterstützt, oder der Parameter Cid ist fehlerhaft. Ist der Parameterm gesetzt, ist CE_MODE der einzig gültige Fehler.
CE_OOP	Gerät an der parallelen Schnittstelle meldet Papierende.
CE_OVERRUN	Ein Zeichen wird von der Hardware nicht rechtzeitig gelesen. Das Zeichen ist verloren.
CE_PTO	Bei der Kommunikation mit der parallelen Schnittstelle trat ein Timeout auf.
CE_RLSDTO	Receive-line-signal-detect-Unterbrechung. RLSD war für die durch das Element fRlsdHold der TCOMSTAT-Struktur festgelegte Dauer herabgesetzt, während gleichzeitig versucht wurde, ein Zeichen zu übertragen.
CE_RXOVER	Überlauf der Empfangswarteschlange. Entweder reicht der

	Platz in der Empfangswarteschlange nicht aus, oder die Übertragung wurde nach Erhalt des End-of-file-Zeichens fortgesetzt.
CE_RXPARITY	Die Hardware stellte einen Parity-Error fest.
CE_TXFULL	Die Sendewarteschlange war voll. Gleichzeitig wurde versucht, ein Zeichen zur Warteschlange hinzuzufügen. Auch wenn offensichtlich kein Fehler über das Integer Resultat zurückgemeldet wurde gibt der Parameterblock „ComStat“ über den aktuellen Zustand der Schnittstelle Auskunft (siehe Delphi Hilfe unter ComStat oder TComStat) :

Der wesentliche Bestandteil des ComStat Records ist das Byte „flags“.
Auch hier muss mit den vordefinierten Maske der Fehler ausmaskiert und interpretiert werden

CSTF_CTSHOLD	Bestimmt, ob die Übertragung darauf wartet, daß das CTS-Signal (clear-to-send) gesendet wird.
CSTF_DSRHOLD	Bestimmt, ob die Übertragung darauf wartet, daß das DSR-Signal (data-set-ready) gesendet wird.
CSTF_RLSDHOLD	Bestimmt, ob die Übertragung darauf wartet, daß das RLSD-Signal (receive-line-signal-detect) gesendet wird.
CSTF_XOFFHOLD	Bestimmt, ob die Übertragung wartet, wenn das Zeichen XOFF empfangen wurde.
CSTF_XOFFSENT	Bestimmt, ob die Übertragung wartet, wenn das Zeichen XOFF übertragen wurde. Die Übertragung muß nach dem Senden eines XOFF-Zeichens für Systeme gestoppt werden, die das folgende Zeichen unabhängig von seinem tatsächlichen Wert als XON interpretieren.
CSTF_EOF	Bestimmt, ob das Zeichen für Dateiende (EOF) empfangen worden ist.
CSTF_TXIM	Bestimmt, ob ein Zeichen darauf wartet, übertragen zu werden.

Procedure ***FlushComRx***(ComPort : string);
Leert den Empfangs Buffer komplett.

Procedure ***FlushComTx***(ComPort : string);
Leert den Sende Buffer komplett.

Procedure ***CloseComPort***(ComPort : string);
Die Schnittstelle wird komplett freigegeben und erscheint in „ComPortOk“ bzw. „ComxInUse“ wieder als frei. Die Prozedur muss immer angewendet werden, wenn das entsprechende Port mittels „OpenComPort“ eröffnet wurde.

Procedure ***CheckPorts***;
Nach Freigabe von Ports oder sonstigen gravierenden Änderungen kann diese Prozedure aufgerufen werden, um die „ComAvails“ und „ComInUse“ auf den neuesten Stand zu bringen.

Function ***ComPortOk***(ComPort : string) : boolean;
ComPortOk entspricht den Eigenschaften „ComxAvail“.

Function ***ComSetDTR***(ComPort : string; OnOff : Boolean): Boolean;
Function ***ComSetRTS***(ComPort : string; OnOff : Boolean): Boolean;
Die Hardware Handshake Leitungen des angegebenen Ports werden gesetzt resp. zurückgesetzt. Kann als Signal für spezielle angeschlossene Hardware benutzt werden.

Function ***ComGetDSR***(ComPort : string): Boolean;
Function ***ComGetCTS***(ComPort : string): Boolean;

Die Hardware Handshake Leitungen des angegebenen Ports abgefragt.Kann als Signal für spezielle angeschlossene Hardware benutzt werden.

Procedure ***ResetAllPorts***;
Schliesst alle zuvor mit „OpenComPort“ geöffnete Ports. Siehe CloseComPort.

Procedure ***GetDCB***(ComPort : string; var DCB : TDCB);
Procedure ***SetDCB***(ComPort : string; DCB : TDCB);
Die beiden Prozeduren dienen für spezielle Einstellungen des gewählten Ports. Änderungen haben nur Gültigkeit, wenn sie vor OpenComPort vorgenommen wurden. Eine exakte Aufschlüsselung der einzelnen Parameter und ihrer Funktion erhält man mit der Delphi Hilfe unter dem Suchbegriff „TDCB“.

GamePort

Nicht-visuelle Komponente zur Auswertung von bis zu 2 GamePort-Schnittstellen.
Übernimmt das komplette Handling

Komponenten Deklaration

```
type
  TGamePort = class(TComponent)
  private
    { Private-Deklarationen }
    FWndHandle: THandle;       //Fensterhandle
    FNumDevs: Word;     {Anzahl d. unterstützten Geräte}
    FxPos1, FyPos1, FzPos1, FxPos2, FyPos2, FzPos2: Integer; //akt. Position
    Fdx1, Fdy1, Fdz1, Fdx2, Fdy2, Fdz2: Integer;             //diff. Bewegung
    FEnabled: Boolean;                    //Ein-Ausschalten der JoySticks
    FUseDevice: TUseDevice;            // Verwendung Joy1, Joy2
    FPortOk : boolean;
    FStick1Ok : boolean;
    FStick2Ok : boolean;
    FOnJoy1Move, FOnJoy2Move: TJoyMoveEvent;          //Bewegungsereignisse
    FOnJoy1ButtonDown, FOnJoy2ButtonDown,
    FOnJoy1ButtonUp, FOnJoy2ButtonUp: TJoyButtonEvent;  //Tastenereignisse
    procedure SetEnabled(Value: Boolean);
    procedure SetUseDevice(Value: TUseDevice);
    function  GetPortOk : boolean;
    procedure SetPortOk(pValue : boolean);
    procedure SetStick1Ok(pValue : boolean);
    procedure SetStick2Ok(pValue : boolean);
    function  TranslateButton(Buttons: LongInt): TJoyButton;
    function  TranslateButtonChanged(Buttons: LongInt): TJoyButtonChanged;
    procedure UpdateCoords(JoyID, x, y, z: Integer);
  protected
    { Protected-Deklarationen }
    procedure WndProc(var Msg: TMessage);
    procedure DoJoyMove(JoyID, x, y, z: Integer; Buttons: LongInt);
    procedure DoJoyButtonDown(JoyID, x, y, z: Integer; Buttons: LongInt);
    procedure DoJoyButtonUp(JoyID, x, y, z: Integer; Buttons: LongInt);

  public
    { Public-Deklarationen }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    //öffentl. Funktionen
    procedure Activate;
    procedure DeActivate;
    //Eigenschaften
    property xPos1: Integer read FxPos1;
    property yPos1: Integer read FyPos1;
    property zPos1: Integer read FzPos1;
    property xPos2: Integer read FxPos2;
    property yPos2: Integer read FyPos2;
    property zPos2: Integer read FzPos2;
    property dx1: Integer read Fdx1;
    property dy1: Integer read Fdy1;
    property dz1: Integer read Fdz1;
    property dx2: Integer read Fdx2;
    property dy2: Integer read Fdy2;
    property dz2: Integer read Fdz2;
```

```
  published
    { Published-Deklarationen }
    //Eigenschaften
    property PortOk    : boolean   read GetPortOk   write SetPortOk;
    property StickOk1  : boolean   read FStick1Ok   write SetStick1Ok;
    property StickOk2  : boolean   read FStick2Ok   write SetStick2Ok;
    property Enabled   : Boolean   read FEnabled    write SetEnabled default
True;
    property UseDevice : TUseDevice read FUseDevice write SetUseDevice;
    //Ereignisse
    property OnJoy1Move: TJoymoveEvent read FOnJoy1Move write FOnJoy1Move;
    property OnJoy2Move: TJoyMoveEvent read FOnJoy2Move write FOnJoy2Move;
    property OnJoy1ButtonDown: TJoyButtonEvent read FOnJoy1ButtonDown
      write FOnJoy1ButtonDown;
    property OnJoy2ButtonDown: TJoyButtonEvent read FOnJoy2ButtonDown
      write FOnJoy2ButtonDown;
    property OnJoy1ButtonUp: TJoyButtonEvent read FOnJoy1ButtonUp
      write FOnJoy1ButtonUp;
    property OnJoy2ButtonUp: TJoyButtonEvent read FOnJoy2ButtonUp
      write FOnJoy2ButtonUp;
  end;
```

PlaySound

Nicht-visuelle Komponente zur Ausgabe von Wave-Dateien über Sound-Blaster etc. Geeignet zur programmgesteuerten Sprachausgabe und Ansage von Meldungen und Warnungen in Wave-Files

Komponenten Deklaration

```
type
  tPlayState = (psStopped, psStartWait, psStartExit, psContinous);

type
  TPlaySound = class(TComponent)
  private
    { Private-Deklarationen }
    RepeatTimer: integer;
    fPlayState : tPlayState;
    fWaveFile  : string;
    fRepDel    : double;
    wavName    : array [0..64] of char;
    procedure RepeatSound(Sender: TObject);
    procedure SetPlayState(pValue : tPlayState);
    procedure SetRepDel(pValue : double);
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    Property WaveFile  : string     read FWaveFile write fWaveFile;
    Property PlayState : tPlayState read fPlayState write SetPlayState;
    property RepDelay  : double     read fRepDel   write SetRepDel;
  end;
```

IniEditor

Nicht-visuelle Komponente zum Lesen und Schreiben von Windows-INI-Files. Unterstützt den Programmierer beim Erstellen, Auslesen und Verändern des programm-eigenen INI-Files.

Komponenten Deklaration

```
type
  TFileLocation = (ApplicationPath, WindowsPath, PathInFileName);

  TIniEditor = class(TComponent)
  private
    FFileName: string;
    FFilePath: string;
    FFileLocation: TFileLocation;
    procedure SetFileName(strFileName: string);
    procedure SetFileLocation(Location: TFileLocation);
    procedure AppendFilePath(var FullPath: string);
  public
    constructor Create(AOwner: TComponent); override;
  published
    property FileName: string read FFileName write SetFileName;
    property FileLocation: TFileLocation read FFileLocation write
SetFileLocation;
    procedure WriteBool(Section, Entry: string; Value: boolean);
    procedure WriteChar(Section, Entry: string; Value: char);
    procedure WriteInt(Section, Entry: string; Value: LongInt);
    procedure WriteReal(Section, Entry: string; Value: Extended);
    procedure WriteStr(Section, Entry, Value: string);
    function ReadBool(Section, Entry: string; DefaultValue: boolean): boolean;
    function ReadChar(Section, Entry: string; DefaultValue: char): char;
    function ReadInt(Section, Entry: string; DefaultValue: LongInt): LongInt;
    function ReadReal(Section, Entry: string; DefaultValue: Extended): Extended;
    function ReadStr(Section, Entry, DefaultValue: string): string;
    procedure InitEntry(Section, Entry : string);
    procedure InitSection(Section : string);
    procedure EraseSection(Section: string);
    procedure EraseEntry(Section, Entry: string);
  end;
```

SmartTimer

Nicht-visuelle Timer-Komponente. Benutzt nur einen der wenigen Windows-Timer, stellt aber eine praktisch fast unbegrenzte Zahl von Timerkanälen zur Verfügung. OneShot, Blink und Dauerlauf.

Komponenten Deklaration

```
type
  TTimerEvent = procedure (Sender: TObject; BlinkHigh : boolean) of object;
  TTimerMode  = (tmStopped, tmTicking, tmBlinking, tmIterate);

type
  TSmartTimer = class(TComponent)
  private
    { Private-Deklarationen }
    TimerHandle : integer;
    fTime       : word;
    fCycles     : LongInt;
    FTimerEvent : TTimerEvent;
    fTimerMode  : TTimerMode;
    procedure ClearTimer;
    procedure SetTime(pValue : word);
    procedure SetTimerMode(pValue : TTimerMode);
    procedure TimerInt(Sender: TObject);
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    Property TimerEvent : TTimerEvent  read FTimerEvent write FTimerEvent;
    property Time       : word         read fTime       write SetTime;
    property TimerMode  : TTimerMode   read fTimerMode  write SetTimerMode;
    property Cycles     : LongInt      read fCycles     write fCycles;
  end;
```

PrecTimer

Nicht-visuelle Timer-Komponente. Benutzt einen speziellen praezisen Windows-Timer. Einstellbare Genauigkeit 1msec. Minimal Tick 1msec. Blink und Dauerlauf.

Komponenten Deklaration

```
type
  TPrecTimer = class(TComponent)
  private
    { Private-Deklarationen }
    FWnd: hWnd;                       (* FensterHandle *)
    FTimeCaps: TTimeCaps;             (* Dev-Caps MMTimer *)
    FEnabled: Boolean;
    FInterval: Cardinal;
    FResolution: Cardinal;
    FPeriodic: Boolean;
    FID: Integer;                     (* ID des MMTimers *)
    FOnTimer: TPrecTimerEvent;
    FResult: Integer;
  protected
    { Protected-Deklarationen }
    procedure UpdateTimer;
    procedure SetEnabled(Value: Boolean);
    procedure SetInterval(Value: Cardinal);
    procedure SetResolution(Value: Cardinal);
    procedure SetPeriodic(Value: Boolean);
    procedure WndProc(var Msg: TMessage);
    procedure DoOnTimer(Time: Cardinal); dynamic;
  public
    { Public-Deklarationen }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    property Result: Integer read FResult;
published
    { Published-Deklarationen }
    property Interval: Cardinal read FInterval write SetInterval;
    property Resolution: Cardinal read FResolution write SetResolution;
    property Periodic: Boolean read FPeriodic write SetPeriodic;
    property Enabled: Boolean read FEnabled write SetEnabled;
    property OnTimer: TPrecTimerEvent read FOnTimer write FOnTimer;
  end;
```

Hinweis zur Benutzung

Diese Komponente lässt sich von 1msec bis high(Integer) einstellen. Die Auflösung (Precision) gibt dabei die Genaugkeit und indirekt den notwendigen internen Rechenaufwand an. Diese Eigenschaft sollte deshalb nicht auf unnötig kleine Werte eingestellt werden. Im Event dieses Timers sollten nur Aktionen veranstaltet werden, die auch sicher zu Ende kommen, bevor der nächste Event zuschlägt, sonst kann man WIN95-98-NT langfristig problemlos zum Absturz bringen.

FormSize

Nicht-visuelle Komponente. Einstellbare Grössenbegrenzung für Formulare. Minimal- und Maximalwerte unabhängig wählbar.

Komponenten Deklaration

```
type
  TFormSize = class(TComponent)
  private
    { Private-Deklarationen }
    FOnResize   : TNotifyEvent;
    FWidthMin   : Integer;
    FWidthMax   : Integer;
    FHeightMin  : Integer;
    FHeightMax  : Integer;
    FTopMin     : Integer;
    FTopMax     : Integer;
    FLeftMin    : Integer;
    FLeftMax    : Integer;

    Form            : TForm;

    procedure SetTopMin(pValue : Integer);
    procedure SetTopMax(pValue : Integer);
    procedure SetLeftMin(pValue : Integer);
    procedure SetLeftMax(pValue : Integer);
    procedure SetWidthMin(pValue : Integer);
    procedure SetWidthMax(pValue : Integer);
    procedure SetHeightMin(pValue : Integer);
    procedure SetHeightMax(pValue : Integer);
    procedure SetSize;
  protected
    { Protected-Deklarationen }
    procedure FormResize(Sender: TObject);
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property WidthMin    : Integer     read FWidthMin    write SetWidthMin;
    property WidthMax    : Integer     read FWidthMax    write SetWidthMax;
    property HeightMin   : Integer     read FHeightMin   write SetHeightMin;
    property HeightMax   : Integer     read FHeightMax   write SetHeightMax;
    property OnResize    : TNotifyEvent read FOnResize   write FOnResize;
  end;
```

DoubleInst

Nicht-visuelle Komponente. Verhindert den mehrfach Aufruf einer Application. Wird auf die MainForm plaziert.

Komponenten Deklaration

```
type
  { The OneInstance Class declaration }
  TDoubleInst = class(TComponent)
  private
    { Private declarations - Attributes }
    FSwitchToPrev: boolean;         { Bring previous instance to front? }
    FInfoText: string;          { Custom Message if previous detected }
    FShowInfo: boolean;           { Display Message if previous detected? }
    FOnDuplicate: TDuplicate;         { User Method Pointer to be called }
    PrevInstWnd : HWND;
    { Private declarations - Methods }
    procedure CheckForPrevious;         { Actually checks for previous instance
}
  public
    { Public declarations - Methods }
    procedure Loaded; override;
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
    { Published declarations }
    property SwitchToPrev: boolean read FSwitchToPrev write FSwitchToPrev;
    property InfoText: string read FInfoText write FInfoText;
    property ShowInfo: boolean read FShowInfo write FShowInfo;
    property OnDuplicate: TDuplicate read FOnDuplicate write FOnDuplicate;
  end;
```

SysInfo

Nicht-visuelle Komponente. Liefert Daten über vorhandene Resourcen, Laufwerke, CPU, Interrupts, Speicher, Betriebssystem etc.

Komponenten Deklaration

```
var CompanyName : string;
    UserName    : string;

type
  TSysInfo = class(TComponent)
  private
    FWinVers    : string;
    FCPUtyp     : string;
    FSysMem     : string;
    FSysMemUsed : string;
    FSysMemFree : string;
    FDiskSpace  : string;
    FDiskSize   : string;
    FDriveList  : TStringList;
    FDriveCount : integer;

    procedure SetDummy(pValue : string);
    procedure SetEvaluate(pValue : boolean);
    function  GetEvaluate : boolean;
    function  GetUserName : string;
    function  GetCompName : string;
    procedure SetDiskSize(pValue : boolean);
    function  GetDiskSize : boolean;
    procedure SetDriveList(pValue : TStringList);
    procedure SetTaskCount(pValue : Integer);
    procedure SetDriveCount(pValue : Integer);
  public
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
    property NameUser   : string  read GetUserName write SetDummy;
    property NameComp   : string  read GetCompName write SetDummy;
    property VersWin    : string  read FWinVers    write SetDummy;
    property CPUtyp     : string  read FCPUtyp     write SetDummy;
    property SysMem     : string  read FSysMem     write SetDummy;
    property SysMemUsed : string  read FSysMemUsed write SetDummy;
    property SysMemFree : string  read FSysMemFree write SetDummy;
    property DiskSize   : string  read FDiskSize   write SetDummy;
    property DiskSpace  : string  read FDiskSpace  write SetDummy;
    property Evaluate   : boolean read GetEvaluate write SetEvaluate;
    property CheckDrive : boolean read GetDiskSize write SetDiskSize;
    property DriveList: TStringList read FDriveList write SetDriveList;
    property DriveCount : Integer read FDriveCount write SetDriveCount;
  end;
```

DataStrings

Nicht-visuelle Komponente. Kleine Datenbank. Ablegen, suchen und finden von Daten über einen Namen. Schreiben und Lesen aus Datei.

Komponenten Deklaration

```
type
  TDataStrings = class(TComponent)
  private
    { Private-Deklarationen }
    dList     : TStringList;
    function  GetCount : integer;
    procedure SetCount(pValue : integer);
    function  GetDupl  : TDuplicates;
    procedure SetDupl(pValue : TDuplicates);
    function  GetSorted : boolean;
    procedure SetSorted(pValue : boolean);
    function  GetOnChange : TNotifyEvent;
    procedure SetOnChange(pValue : TNotifyEvent);
    function  GetOnChanging : TNotifyEvent;
    procedure SetOnChanging(pValue : TNotifyEvent);
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    procedure Clear;
    function  Add(Name : string) : integer;
    procedure Delete(index : integer);
    function  IndexOfSep(const Name: string): Integer;
    function  GetSepStr(Index, sep : integer) : string;
    function  AppendSepStr(Index : integer; SubStr : string) : boolean;
    function  ChangeSepStr(Index, sep : integer; SubStr : string) : boolean;
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
    property Data       : TStringList  read dList         write DList;
    { Published-Deklarationen from TStringList }
    property Count      : integer      read GetCount      write SetCount;
    property Duplicates : TDuplicates  read GetDupl       write SetDupl;
    property Sorted     : boolean      read getSorted     write SetSorted;
    property OnChange   : TNotifyEvent read GetOnChange   write SetOnChange;
    property OnChanging : TNotifyEvent read GetOnChanging write SetOnChanging;
  end;
```

Beschreibung

Die Komponente TDataStrings ist eine Shell um das Object TStringList.
Eine ähnliche, nicht plazierbare Komponente, ist "TDataList" in dieser Bibliothek.

Der Zweck dieser Komponente ist es, dem Programmierer eine einfache Möglichkeit zu bieten, Daten unter einem Namen abzulegen, zu finden, in eine Datei zu schreiben und wieder zu lesen.

Dazu wurde TDataStrings um die Methoden IndexOfSep, GetSepStr, AppendSepStr und ChangeSepStr erweitert. Alle Metoden und Properties von TStringList bleiben

erhalten und können weiter genutzt werden.

Mit MyData.Data.Add('ddd'); werden neue Einträge angelegt.

Mit MyData.AppendSepStr(0, 'xxx'); wird dem Eintrag an der Indexstelle "0" ein Datum in Form eines Strings angefügt. Dieses Anfügen von SubStrings kann solange wiederholt werden, bis die max. Länge eines Strings erreicht ist.

Mit MyData.ChangeSepStr(0, 1, 'xxx'); wird der SubString "1" an der Indexstelle "0" durch den SubString "xxx" ausgetauscht.

Mit MyData.IndexOfSep('ddd'); wird der Index des Eintrags 'ddd' zurückgegeben.

Mit MyData.GetSepStr(0, 1); erhält man den SubString "1" an der Indexstelle "0". Der Substring kann jetzt z.B. mit StrToInt zu einem Integer umgeformt werden.

Mit MyData.Data.SaveToFile('fff.eee'); wird die Liste in ein File geschrieben.

Mit MyData.Data.LoadFromFile('fff.eee'); wird die Liste aus einem File geladen.

Object TDataList

```
type
  TDataList = class(TStringList)
  private
    { Private-Deklarationen }
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    function  IndexOfSep(const Name: string): Integer;
    function  GetSepStr(Index, sep : integer) : string;
    function  AppendSepStr(Index : integer; SubStr : string) : boolean;
    function  ChangeSepStr(Index, sep : integer; SubStr : string) : boolean;
  published
    { Published-Deklarationen from TStrings }
    property CommaText;
    property Count;
    property Names;
    property Objects;
    property Values;
    property Strings;
    property Text;
    { Published-Deklarationen from TStringList }
    property Duplicates;
    property Sorted;
    property OnChange;
    property OnChanging;
  end;
```

Beschreibung

Wie zu ersehen ist, ist die Komponente TDataList eine Ableitung von TStringList. Da TStringlist keine plazierbare Komponente ist, sonder nur ein Object, muss TDataList in der übergeordneten Komponente mittels

```
MyData      : TDataList;

MyData:= TDataList.create;

MyData.Free;
```

deklariert, erzeugt und freigeben werden. Eine ähnliche, plazierbare Komponente ist "TDataStrings" in dieser Bibliothek.

Der Zweck dieser Komponente ist es, dem Programmierer eine einfache Möglichkeit zu bieten, Daten unter einem Namen abzulegen, zu finden, in eine Datei zu schreiben und wieder zu lesen.

Dazu wurde TStringlist um die Methoden IndexOfSep, GetSepStr, AppendSepStr und ChangeSepStr erweitert. Alle Metoden und Properties von TStringList bleiben erhalten und können weiter genutzt werden.

Mit MyData.Add('ddd'); werden neue Einträge angelegt.

Mit MyData.AppendSepStr(0, 'xxx'); wird dem Eintrag an der Indexstelle "0" ein Datum in Form eines Strings angefügt. Dieses Anfügen von SubStrings kann solange wiederholt werden, bis die max. Länge eines Strings erreicht ist.

Mit MyData.ChangeSepStr(0, 1, 'xxx'); wird der SubString "1" an der Indexstelle "0" durch den SubString "xxx" ausgetauscht.

Mit MyData.IndexOfSep('ddd'); wird der Index des Eintrags 'ddd' zurückgegeben.

Mit MyData.GetSepStr(0, 1); erhält man den SubString "1" an der Indexstelle "0". Der Substring kann jetzt z.B. mit StrToInt zu einem Integer umgeformt werden.

Mit MyData.SaveToFile('fff.eee'); wird die Liste in ein File geschrieben.

Mit MyData.LoadFromFile('fff.eee'); wird die Liste aus einem File geladen.

ExecExe

Nicht-visuelle Komponente. Zum komfortablen Aufruf einer Exe- oder Com Datei aus dem Programm heraus, mit Wartefunktion.

Komponenten Deklaration

```
type
  TExecExe = class(TComponent)
  private
    { Private-Deklarationen }
    FWaitExe  : boolean;
    FExeName  : string;
    FWorkDir  : string;
    FPriority : tPriority;
  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    function StartExe : boolean;
  published
    property WaitExe    : boolean      read FWaitExe      write FWaitExe;
    property Priority   : tPriority    read FPriority     write FPriority;
    property ExeName    : string       read FExeName      write FExeName;
    property WorkDir    : string       read FWorkDir      write FWorkDir;
  end;
```

Beschreibung

Die Komponente TExecExe dient zum Ausführen eines weiteren Programms (Exe-File) unter der Kontrolle des aufrufenden Programms. Auch DOS-Programme sind möglich.

Die Eigenschaft (Property) "ExeName" muss dazu den kompletten Pfad des aufzurufenden Programms enthalten incl. dem Programmnamen und der Extension "exe".

Zusätzlich können noch Ausführungsparameter übergeben werden. Ein Beispiel:

'C:\TestProg.exe -x abc'

Die Eigenschaft (Property) "WaitExe" gibt an ob die Komponente warten soll, bis das aufgerufene Programm wieder geschlossen wurde.

Die Eigenschaft "Priority" legt die Priorität/Wichtigkeit des Programmes fest. Es sollte immer der Wert "Normal" gewählt werden.

Es gibt zwei Ausnahmefälle:
1. wenn das gestartete Programm im Hintergrund laufen soll, die Ausführungszeit keine Rolle spielt und das Hauptprogramm fast die ganze Rechenzeit benötigt (kein WaitExe) kann mit der Priority "Low" gearbeitet werden.

2. Wenn mit "WaitExe" gearbeitet wird, und das gestartete Programm alle Rechenzeit

benötigt um möglichst schnell beendet zu werden, kann die Priorität auf "High" gesetzt werden. Hierbei wird dem Programm >90% Rechenzeit zugebilligt, was für alle anderen Prozesse praktisch den Stillstand bedeutet.

Die Eigenschaft "WorkDir" stellt das Arbeitsverzeichnis für das aufzurufende Programm ein. Sollte das Programm Dateizugriffe ohne Pfad-Angaben machen, erfolgen diese immer in dieses Verzeichnis. Ist "WorkDir" leer bzw. nicht definiert, erhält das aufzurufende Programm die gleiche Arbeits-Directory wie das aufrufende Programm.

Die Funktion "StartExec" startet das Programm und kehrt, abhängig von der Eigenschaft "WaitExe" sofort oder erst nach der Ausführung, mit dem Ergebnis der Operation zurück. Das Ergebnis ist bei erfolgreicher Operation (Programm konnte geladen werden) "true".

Iconize

Nicht-visuelle Komponente. Startet das Programm als Icon in der Taskleiste oder ganz unsichtbar.

Komponenten Deklaration

```
type
(* To prevent your main form displaying at startup, select
   View / Project source from the menu and insert the lines

     ShowWindow(Application.Handle, SW_HIDE);
     Application.ShowMainForm := FALSE;

   before the Application.Run line.
*)
  TIconize = class(TComponent)
  private
    fWindowHandle : HWND;
    fIcon : TIcon;
    fEnabled : boolean;
    fIconThere : boolean;
    fHint : string;
    fPopupMenu : TPopupMenu;
    fAutoMenu : boolean;

    fOnLeftBtnClick : TNotifyEvent;
    fOnLeftBtnDblClick : TNotifyEvent;
    fOnRightBtnClick : TNotifyEvent;
    fOnMouseMove : TNotifyEvent;

    procedure WProc(var Msg: TMessage);
    procedure UpdateIcon (Flags : integer);

    procedure SetIcon (value : TIcon);
    procedure SetEnabled (value : boolean);
    procedure SetHint (value : string);
  protected
    { Protected declarations }
  public
    constructor Create (AOwner : TComponent); override;
    destructor Destroy; override;
  published
    property Icon : TIcon read fIcon write SetIcon;
    property Enabled : boolean read fEnabled write SetEnabled;
    property Hint : string read fHint write SetHint;
    property PopupMenu : TPopupMenu read fPopupMenu write fPopupMenu;
    property AutoMenu : boolean read fAutoMenu write fAutoMenu;

    property OnLeftBtnClick : TNotifyEvent read fOnLeftBtnClick write
fOnLeftBtnClick;
    property OnLeftBtnDblClick : TNotifyEvent read fOnLeftBtnDblClick write
fOnLeftBtnDblClick;
    property OnRightBtnClick : TNotifyEvent read fOnRightBtnClick write
fOnRightBtnClick;
    property OnMouseMove : TNotifyEvent read fOnMouseMove write fOnMouseMove;

  end;
```

Beschreibung:

Mit der Komponente TIconize kann ein Programm so gestartet werden, dass es entweder ganz unsichtbar bleibt oder nur als Icon in der Taskleiste sichtbar wird.

Dazu müssen immer im DPR-File die zwei folgenden Zeilen

```
ShowWindow(Application.Handle, SW_HIDE);
Application.ShowMainForm := FALSE;
```

vor der Zeile **`Application.Run`** eingefügt werden. Zusätzlich muss hier in der Uses-Anweisung noch "Windows" importiert werden.

Ist das property "Enabled" in der Komponente = true, erscheint in der Taskleiste das Icon der Komponente. Dazu muss aber unbedingt ein Icon im Object-Inspector definiert bzw. importiert werden.

Im Haupformular muss auch das property "visible" auf false gesetzt werden.

Das Programm bzw. dessen Main-Window kann jederzeit innerhalb des Programms mit

```
ShowWindow(Application.Handle, SW_Show);
Application.ShowMainForm:= true;
Application.MainForm.Visible:= true;
```

sichtbar gemacht werden. Z.B. mit event "OnLeftBtnClick".
Die Events beziehen sich alle auf das sichtbare Icon.

Wenn im property "Hint" ein Text eingefügt wurde, so wird ein Hint angezeigt, wenn die Maus auf dem Icon in der Taskleiste steht.

Ist das property "AutoMenu" = true so wird bei einem DoppelClick mit der Linken Maustaste auf das Icon derjenige Task ausgeführt, der hinter dem ersten Eintrag des PopUpMenues steht.

Voraussetzung dafür ist jedoch dass ein PopUpMenue definiert wurde und ein Click auf den ersten Eintrag auch eine Aktion zur Folge hat.

Beispiel Programm

Wecker bzw. AlarmClock in Demos\Clock

Stack

Nicht-visuelle Komponente. Dient als Softwarestack für fast beliebige Datentypen, zur Unterstützung z.B. komplexen Rekursionen.

Komponenten Deklaration

```
type
  TStack = class(TComponent)
  private
    { Private declarations }
    StackList  : TStringList;
  protected
    { Protected declarations }
  public
    { Public declarations }
    constructor Create (AOwner : TComponent); override;
    destructor Destroy; override;
    function StackEmpty : boolean;
    function StackItems : integer;
    procedure StackClear;

    function  pushString(item : string) : boolean;
    function  pushBool(item : boolean) : boolean;
    function  pushByte(item : byte) : boolean;
    function  pushInt(item : integer) : boolean;
    function  pushWord(item : word) : boolean;
    function  pushCard(item : cardinal) : boolean;
    function  pushFloat(item : double) : boolean;

    function  popString(var item : string) : boolean;
    function  popBool(var item : boolean) : boolean;
    function  popByte(var item : byte) : boolean;
    function  popInt(var item : integer) : boolean;
    function  popWord(var item : word) : boolean;
    function  popCard(var item : cardinal) : boolean;
    function  popFloat(var item : double) : boolean;
  published
    { Published declarations }
  end;
```

Spy

Gag-Komponente. Die Augen folgen immer dem Mauskursor.

Komponenten Deklaration

```
type
  TSpy = class(TGraphicControl)
  private
    FTimer      : TTimer;
    FBordColor,
    FEyeColor,
    FPupilColor : TColor;
    CurMouse,
    OldMouse    : TPoint;
    Pupil1,
    Pupil2      : TPoint;
    PupilWidth  : Integer;
    PupilDist   : Integer;
    function  GetActive: Boolean;
    procedure SetActive(Value: Boolean);
    procedure SetEyeColor(Value: TColor);
    procedure SetBordColor(Value: TColor);
    procedure SetPupilColor(Value: TColor);
  protected
    procedure Paint; override;
    procedure Invalidate; override;
    procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer); override;
    procedure Timer(Sender: TObject); virtual;
    function  EyeWidth: integer; virtual;
    function  EyeRight: integer; virtual;
    function  EyeCenter(Nr: integer): Integer; virtual;
    function  CalcPupil(Nr: integer): TPoint; virtual;
    procedure DrawPupils(P1,P2: TPoint; Black: Boolean); virtual;
    procedure UpdatePupils; virtual;
  public
    constructor Create(AOwner: TComponent); override;
  published
    property  Active :Boolean    read GetActive   write SetActive;
    property  EyeColor : TColor  read FEyeColor   write SetEyeColor;
    property  BordColor : TColor read FBordColor  write SetBordColor;
    property  PupilColor: TColor read FPupilColor write SetPupilColor;
    property Align;
    property DragCursor;
    property DragMode;
    property ParentShowHint;
    property PopupMenu;
    property ShowHint;
    property Visible;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnStartDrag;
```

VersInfo

Nicht-visuelle Komponente. Dient zum erfassen, editieren und einbauen der sog. VersionInfo (Copyright, Versionsnummer, Company etc.) in das Exe-file.
Enthält weiterhin einen Dialog, der diese Daten zur Laufzeit anzeigen kann.

Komponenten Deklaration

```
type
  TVerNum = array[0..3] of integer;
  TVos = (Windows16, Windows32);
  TVft = (App, Dll);
  TVersInfo = class(TComponent)
  private
    { Private declarations }
    fWrite           : boolean;
    FTitle           : String;
    FProduct2        : String;
    FTime            : integer;
    fResFilename,
    fFileVerNum,
    fProdVerNum      : string;
    fFVerNum,
    fPVerNum         : TVerNum;
    fVFileOS         : TVos;
    fVFileType       : TVft;
    fComments,
    fCompanyName,
    FUserName,
    fFileDescription,
    fFileVersion,
    fInternalName,
    fCopyright,
    fOrgFilename,
    fProductName,
    fProductVers     : string;
    procedure StrToVerNum(const Source: string; var Dest: TVerNum);
    procedure SetWrite(Value: boolean);
    procedure SetFileVerNum(const Value: string);
    procedure SetProdVerNum(const Value: string);
    function WriteResFile: boolean;
  protected
    { Protected declarations }
  public
    { Public declarations }
    constructor Create(AOwner: TComponent); override;
    function Execute: Boolean;
  published
    { Published declarations }
    property ResWrite   : boolean   read fWrite       write SetWrite stored
false;
    property ResFilename: string    read fResFilename write fResFilename;
    property VerNumFile : string    read fFileVerNum  write SetFileVerNum;
    property VerNumProd : string    read fProdVerNum  write SetProdVerNum;
    property VFileOS    : TVos      read fVFileOS     write fVFileOS;
    property VFileType  : TVft      read fVFileType   write fVFileType;
    property Comments   : string    read fComments    write fComments;
    property NameCompany: string    read fCompanyName write fCompanyName;
    property NameUser   : string    read fUserName    write fUserName;
```

```
    property FileDescription: string read fFileDescription write
fFileDescription;
    property FileVersion: string    read fFileVersion write fFileVersion;
    property NameInternal: string   read fInternalName write fInternalName;
    property Copyright  : string    read fCopyright   write fCopyright;
    property OrgFilename: string    read fOrgFilename write fOrgFilename;
    property NameProduct: string    read fProductName write fProductName;
    property NameProd2  : string    read fProduct2    write fProduct2;
    property ProductVers: string    read fProductVers write fProductVers;
    property Title       : String   read FTitle       write FTitle;
    property TimeControl : integer  read FTime        write FTime;
  end;
```

Beschreibung

Die Komponente TVersInfo erfüllt zwei Funktionen:

1. Versions Informationen in dem EXE-File zur Auswertung unter WIN
2. Ein About Dialog für den Anwender.

1.1 Versions Info

Ein gutes Programm für Windows hat in der Regel ein sog. Versions Info Teil im EXE-File, der der WIN und diverse Tools ausgelesen werden kann. Diese Info besteht zum Teil aus Daten, die extern über die Tools nicht ohne weiteres abfragbar sind und solche, die immer abfragbar sind.

Delphi bietet leider keine Möglichkeit dieses Info einfach zu erzeugen, da dies über eine Resourcen Datei mit dem ResourceWorkshop erledigt werden muss.

Die Daten bestehen aus Hersteller Informationen, Programm Informationen und Informationen über den Anwender, wenn bekannt.

Alle diese Daten werden im Object Inspector eingetragen. Die Komponente generiert eine Datei mit dem Namen "VERSION.RES". Diese Datei wird automatisch bei der Compilierung mit eingebunden.

Die Property Namen entsprechen denen von Windows für File-Infos verwendeten. Die genau Beschreibung der einzelnen Properties bzw. deren Verwendung können Sie in der Windows-API Hilfe unter "Version Info" bzw. "File Info" finden.

Durch einen DoppelClick auf das property "ResWrite" wird die Resourcendatei geschrieben. Um eine Änderung gültig zu machen, muss hier geclickt werden.

2.1 About Dialog

Der zweite Teil der Komponente ist der About Dialog
Dieser wird zur Laufzeit über seine Methode "Execute" aufgerufen. Der Dialog zeigt ein Teil der properties der Komponente an.
Sein Titel und die "Laufzeit" sind einstellbar. Wird "TimeControl" > 100 gesetzt, schaltet sich der Dialog automatisch nach "TimeControl" x msec aus.

ThProcPriority

Komponente zum Einstellen der Priorität eines Programmes, auch zur Laufzeit.

Komponenten Deklaration

```
type
  TThProcPriority = class(TComponent)
  private
    { Private-Deklarationen }
    fPriority : word;
    b         : longbool;
//  NORMAL_PRIORITY_CLASS;         (*  32 *)
//  IDLE_PRIORITY_CLASS            (*  64 *)
//  HIGH_PRIORITY_CLASS            (* 128 *)
//  REALTIME_PRIORITY_CLASS        (* 256 *)

    Procedure SetPriority(pValue : word);
    function GetPriority : word;

  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
  published
    { Published-Deklarationen }
    Property ProcessPriority : word  read GetPriority  write SetPriority;
  end;
```

ScrnSaveCtrl

Komponente zur Manipulation des Windows Screensaver. Der Screen Saver kann ganz abgeschaltet werden oder die Einschaltverzögerung in weiten Grenzen manipuliert werden.

Komponenten Deklaration

```
type
  TScrnSaveCtrl = class(TComponent)
  private
    { Private-Deklarationen }
    FEnabled   : boolean;
    FTimeOut   : integer;
    savEnabled : integer;
    savTimeOut : integer;

    Procedure SetEnabled(pValue : boolean);
    Procedure SetTimeOut(pValue : integer);
    Procedure SetOrgValues;

  protected
    { Protected-Deklarationen }
  public
    { Public-Deklarationen }
    constructor Create(aOwner :tComponent); override;
    destructor Destroy; override;
  published
    { Published-Deklarationen }
    Property SaverEnable : boolean    Read FEnabled  write SetEnabled;
    Property SaverTimeOut: integer    read FTimeOut  write SetTimeOut;
  end;
```

RecCtrl

Komponente zur Steuerung der Aufnahme und Wiedergabe der graph. Komponenten wie z.B. xyGraph oder ytGraph.

Komponenten Deklaration

```
type
  TrcBtnType = (rcPlay, rcPause, rcStop, rcPgPrev, rcPgNext,
                rcStepBack, rcStepForw, rcRecord, rcEject);
  TrcBtnSet = set of TrcBtnType;

  TrcGlyph  = (mgEnabled, mgDisabled, mgColored);

  Bitmaps   = array[TrcGlyph] of TBitmap;

  ErcNotify = procedure (Sender: TObject; Button: TrcBtnType) of object;

  TRecControl = class(TCustomControl)
  private
    BtnArray         : array[TrcBtnType] of Bitmaps;
    FVisibleButtons : TrcBtnSet;
    FEnabledButtons : TrcBtnSet;
    FColoredButtons : TrcBtnSet;
    FDownButtons    : TrcBtnSet;
    LockedButtons   : TrcBtnSet;
    InternEnabled   : TrcBtnSet;
    CurrentButton   : TrcBtnType;
    CurrentRect     : TRect;
    ButtonWidth     : Integer;
    MinBtnSize      : TPoint;
    FOnBtnDown      : ErcNotify;
    FOnBtnUp        : ErcNotify;

    procedure LoadBitmaps;
    procedure DestroyBitmaps;
    procedure SetEnabledButtons(pValue: TrcBtnSet);
    procedure SetColored(pValue: TrcBtnSet);
    procedure SetVisible(pValue: TrcBtnSet);
    procedure SetDown(pValue: TrcBtnSet);
    procedure DoMouseDown(XPos, YPos: Integer);
    procedure WMLButtonDown(var Message: TWMLButtonDown);
      message WM_LButtonDown;
    procedure WMLButtonDblClk(var Message: TWMLButtonDblClk);
      message WM_LButtonDblClk;
    procedure WMLButtonUp(var Message: TWMLButtonUp);
      message WM_LButtonUp;
    procedure WMSize(var Message: TWMSize);
      message WM_SIZE;
    function VisibleButtonCount: Integer;
    procedure Adjust;
    procedure DrawButton(Btn: TrcBtnType);
  protected
    procedure Paint; override;
    PROCEDURE ClipMouse(clip : boolean);
  public
    procedure EnableAll;
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
```

```
  published
    property BtnColored: TrcBtnSet read FColoredButtons write SetColored;
    property BtnEnabled: TrcBtnSet read FEnabledButtons write SetEnabledButtons;
    property BtnVisible: TrcBtnSet read FVisibleButtons write SetVisible;
    property BtnDown: TrcBtnSet read FDownButtons write SetDown;
    property Enabled;
    property Visible;
    property ParentShowHint;
    property ShowHint;
    property OnBtnDown  : ErcNotify read FOnBtnDown  write FOnBtnDown;
    property OnBtnUp    : ErcNotify read FOnBtnUp    write FOnBtnUp;
    property OnEnter;
    property OnExit;
  end;
```

Logic

Umfangreiche und leistungsfähige Komponente zum Darstellen und Untersuchen bzw. Analysieren von digitalen Daten. Einstellbar von einer einfachen graphischen binär-Anzeige bis zum Logic-Analyzer. Buffer für 16000 32bit-Worte. Patternsuche.

Komponenten Deklaration

```
type
  TBuffer32  = array[0..64000] of LongInt;
  TBuffer16  = array[0..128000] of Word;
  TBuffer8   = array[0..256000] of Byte;
  TModus     = (Diagram, TextDisp);
  TLogSize   = (Size8, Size16, Size32);
  TLogic     = class(TDisplay)
  private
    { Private-Deklarationen }
    RepeatTimer : integer;
    PointerPx    : LongInt;
    FPointerEna  : Boolean;
    FPointerTime : double;
    FPointerPos  : LongInt;
    FColMarker  : TColor;
    FColPointer  : TColor;
    MarkerPx    : LongInt;
    FMarkerEna  : Boolean;
    FMarkerTime : double;
    FMarkerPos  : LongInt;
    FOnZoomSet  : TNotifyEvent;
    FOnRefresh  : TNotifyEvent;
    FMatchVal   : string;
    FMatchAdr   : longint;
    FChanName   : TStringList;
    FChanFirst  : LongInt;
    FChanLast   : LongInt;
    FLogSize    : TLogSize;
    FColCurve   : TColor;
    FTimeBase   : double;
    FModus      : TModus;
    FColTextDisp: TColor;
    FColText    : TColor;
    FColDisplay : TColor;
    FColTrigger : TColor;
    FColGrid  : TColor;
    FStartPos   : LongInt;
    FEndPos     : LongInt;
    FStartTime  : double;
    FEndTime    : double;
    FTriggerPos : LongInt;
    Match       : longint;
    MatchAnd    : longint;
    painted     : Boolean;
    MarkerPosRect,
    MarkerTimeRect,
    PointerPosRect,
    PointerTimeRect,
    DiffPosRect,
    DiffTimeRect,
    BtnVertAll,
```

```
    BtnHorAll,
    BtnStepUp,
    BtnStepDown,
    BtnStepLeft,
    BtnStepRight,
    BtnPageLeft,
    BtnPageRight,
    HexRect,
    SizeRect    : TRect;
    SaveCursor  : TCursor;
    FZoomEna    : Boolean;
    Buffer32    : ^TBuffer32;
    Buffer16    : ^TBuffer16;
    Buffer8     : ^TBuffer8;
    FBufferSize : LongInt;
    FBufferCount: LongInt;
    ZoomPos1    : TPoint;
    ZoomPos2    : TPoint;
    VertAll,
    StepUp,
    StepDown,
    HorAll,
    StepLeft,
    StepRight,
    PageLeft,
    PageRight   : boolean;
    ModeChanged : boolean;
    CopyBmp     : TBitMap;
    xKorr       : double;
    Zeilen      : LongInt;
    ZeilenH     : LongInt;
    ChanMax     : LongInt;
    XStartP,
    YStartP     : integer;

    Procedure DrawSize;
    procedure DrawPointer;
    procedure DrawMarker;
    procedure ShowPos;
    function  FPrintDisp : TBitmap;
    function  FPrintAll : TBitmap;
    procedure FClearBmp(b : TBitmap);
    procedure DrawZeile(z : longint);
    procedure RepeatButton(Sender: TObject);
    procedure SetPointerEna(pValue: Boolean);
    procedure SetPointerTime(pValue: double);
    procedure SetPointerPos(pValue: LongInt);
    procedure SetMarkerEna(pValue: Boolean);
    procedure SetMarkerTime(pValue: double);
    procedure SetMarkerPos(pValue: LongInt);
    Procedure SetTriggerPos(pValue : LongInt);
    Procedure SetStartTime(pValue : double);
    Procedure SetStartPos(pValue : LongInt);
    Procedure SetEndTime(pValue : double);
    Procedure SetEndPos(pValue : LongInt);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColTextDisp(pValue : TColor);
    Procedure SetColText(pValue : TColor);
    Procedure SetColMarker(pValue : TColor);
    Procedure SetColDisplay(pValue : TColor);
    Procedure SetColGrid(pValue : TColor);
    Procedure SetColTrigger(pValue : TColor);
    procedure SetChName(pValue: TStringList);
```

```
    procedure SetChanFirst(pValue: LongInt);
    procedure SetChanLast(pValue: LongInt);
    procedure SetLogSize(pValue: TLogSize);
    procedure SetColCurve(pValue: TColor);
    procedure SetBufferSize(pValue : LongInt);
    procedure SetBufferCount(pValue : LongInt);
    procedure SetTimeBase(pValue : double);
    procedure SetModus(pValue : TModus);
    procedure SetMatchVal(pValue : string);
    procedure SetMatchAdr(pValue : LongInt);
    function  GetNextMatch : longint;
  protected
    { Protected-Deklarationen }
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    ZoomStartX  : LongInt;
    ZoomStartY  : LongInt;
    ZoomEndX    : LongInt;
    ZoomEndY    : LongInt;

    function  LoadFromFile(FileName : string) : boolean;
    function  SaveToFile(FileName : string) : boolean;
    procedure SetBuffer(adrs : LongInt; val : LongInt);
    function  GetBuffer(adrs : LongInt): LongInt;
    procedure ZoomRefresh;
    procedure DispRefresh;
    function  GetDispRect : Trect;
    Procedure DoStepLeft;
    Procedure DoStepRight;
    Procedure DoPageRight;
    Procedure DoPageLeft;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property canvas;
    property MatchValue : String   read FMatchVal    write SetMatchVal;
    property MatchAdr   : LongInt  read FMatchAdr    write SetMatchAdr;
    property NextMatch  : LongInt  read GetNextMatch;
    property ColTrigger : TColor   read FColTrigger  write SetColTrigger;
    property ColTextDisp: TColor   read FColTextDisp write SetColTextDisp;
    property ColText    : TColor   read FColText     write SetColText;
    property ColDisplay : TColor   read FColDisplay  write SetColDisplay;
    property ColGrid    : TColor   read FColGrid     write SetColGrid;
    property ColPointer : TColor   read FColPointer  write SetColPointer;
    property ColMarker  : TColor   read FColMarker   write SetColMarker;
    property TriggerPos : LongInt  read FTriggerPos  write SetTriggerPos;
    property DispStartPos: LongInt read FStartPos    write SetStartPos;
    property DispEndPos  : LongInt read FEndPos      write SetEndPos;
    property DispStartTime: double read FStartTime   write SetStartTime;
    property DispEndTime : double  read FEndTime     write SetEndTime;
    Property PointerEnable:Boolean  read FPointerEna   write SetPointerEna;
    Property PointerTime : double   read FPointerTime  write SetPointerTime;
    Property PointerPos  : LongInt  read FPointerPos   write SetPointerPos;
    Property MarkerEnable:Boolean  read FMarkerEna   write SetMarkerEna;
    Property MarkerTime : double   read FMarkerTime  write SetMarkerTime;
    Property MarkerPos  : LongInt  read FMarkerPos   write SetMarkerPos;
```

```
    property Modus     : TModus    read FModus       write SetModus;
    property TimeBase  : double    read FTimeBase    write SetTimeBase;
    property BufferSize: LongInt   read FBufferSize  write SetBufferSize;
    property BufferCount: LongInt  read FBufferCount write SetBufferCount;
    property ColCurve  : TColor    read FColCurve    write SetColCurve;
    property ChanNamen : TStringList  read FChanName  write SetChName;
    property ChanFirst : LongInt      read FChanFirst write SetChanFirst;
    property ChanLast  : LongInt      read FChanLast  write SetChanLast;
    property LogSize   : TLogSize     read FLogSize   write SetLogSize;
    property ZoomEnable: boolean      read FZoomEna   write FZoomEna;
    property onZoomSet : TNotifyEvent read FOnZoomSet write FOnZoomSet;
    property onRefresh : TNotifyEvent read FOnRefresh write FOnRefresh;
    Property PrintDisp : TBitMap      read FPrintDisp  write FClearBmp;
    Property PrintAll  : TBitMap      read FPrintAll   write FClearBmp;
end;
```

Eigenschaften/Properties:

property ***Align***;

Delphi kompatibel: automatische Anpassung etc.

property ***canvas*** : TCanvas;

Zeichenfläche der Komponente. Der Entwickler kann mit den vorhandenen Properties von „canvas“ eigene Zeichenoperationen durchführen und Texte plazieren. Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche. Siehe auch ScalePnt.

Beispiel:

```
var r : TRect;
   begin
       r:= Logic.GetDispRect;
       with r do begin
         Logic.Canvas.MoveTo(left, top);
         Logic.Canvas.LineTo(right, bottom);
         Logic.Canvas.TextOut(left, top + (bottom-top) div 2, 'Text');
     end;
   end;
```

property ***MatchValue*** : String;

Der string kann bis zu 32 Zeichen lang sein. Es sind nur die Zeichen ‘x’, ‘0’ und ‘1’ zulässig. Das ‘x’ steht für don’t care, die ‘0’ für boolean Null und die ‘1’ für boolean 1. Das letzte Zeichen entspricht der Bitstelle „0“ eines 32 Bit Wortes, das vorletzte der Bitstelle „1“ etc. etc. Ist der String kürzer als 32 Zeichen, wird er intern, von vorne beginnend, auf 32 Zeichen mit ‘x’ = don’t care aufgefüllt. Bei der Operation „NextMatch“ wird der Buffer, beginnend mit „MatchAdr“, solange durchsucht, bis entweder das nächste passende 32 Bit Wort gefunden ist, oder das Bufferende erreicht ist. Ist ein Match gefunden, so wird der grafische „Pointer“ auf die Adresse eingestellt, ebenso das Property MatchAdr.

property ***MatchAdr*** : LongInt;

MatchAdr ist die Buffer Adresse, bei der mit „NextMatch“ mit der Suche begonnen wird.

property ***NextMatch*** : LongInt; Read-Only

NextMatch durchsucht den Buffer nach einem „Match“ auf den String „MatchValue“. Im Erfolgsfall wird wird die neue MatchAdr zurückgegeben, ansonsten -1.
Siehe auch „MatchValue“.

property ***PrintAll*** : TBitMap;

PrintAll liefert die Komponente als BitMap. Hiermit lassen sich diverse Datei- oder Print Operationen durchführen. Das folgende Beispiel speichert die Komponente als BitMap File auf die Disk. Weitere Beispiele sind im Projekt „Logicana.dpr“ zu finden.

Beispiel:
```
Logic.PrintAll.SaveToFile('FileName.bmp');
Logic.PrintAll:= nil;                    (* gibt Resource frei *)
```

property ***PrintDisp*** : TBitMap;
Identisch mit PrintAll, ausser dass das BitMap nur das Display enthält.

property ***NameHead*** : String32;
NameHead ist die Überschrift der Komponente und erscheint im obersten Textfenster.

property ***ColHead*** : Tcolor;
Hintergrund Farbe der Überschrift

property ***ColTxtHead*** : Tcolor;
Farbe der Überschrift

property ***DisplayTyp*** : tDispType;
DisplayTyp stellt die Erscheinungsform von Oscillo ein. Es gibt 4 Möglichkeiten:

complete : Alle Teile von Logic werden dargestellt.
dispOnly : Nur das Display selbst wird dargestellt.
dispTop : Das Display, die Kanalnamen und „NameHead“ wird dargestellt.
dispCenter : Nur das Display und die Kanalnamen werden dargestellt.
dispBottom : Das Display, die Kanalnamen und die Tasten werden dargestellt.

property ***Modus*** : tModus;
Diagram : die Daten werden grafisch dargestellt.
TextDisp : die Daten werden zeilenweise als Text dargestellt.

property ***ChanFirst*** : boolean;
property ***ChanLast*** : boolean;
Diese beiden Eigenschaften bestimmen im Grafik-Modus den ersten bzw. letzten sichtbaren Kanal. Ist z.B. ChanFirst = 5 und ChanLast = 8 werden die Kanäle 5..8 dargestellt.

property ***ColDisplay*** : Tcolor;
Hintergrund Farbe des Displays im Grafik Mode

property ***ColCurve*** : Tcolor;
Farbe der Linien/Kurven bei der grafischen Darstellung

property ***ColTextDisplay*** : Tcolor;
Hintergrund Farbe des Displays im Text Mode

property ***ColText*** : Tcolor;
Schrift-Farbe des Displays im Text Mode

property ***ColGrid*** : Tcolor;
Farbe des Rasters des Displays im Grafik Mode

property ***ColPointer*** : Tcolor;
Farbe der Messlinien des Displays

property ***ColMarker*** : Tcolor;

Farbe der Bezugslinien für die Messlinien des Displays

property ***ColTrigger*** : Tcolor;
Farbe der eingeblendeten Triggerlinie

property ***TimeBase*** : double;
Gibt den zeitlichen Abstand zwischen zwei Messwerten an. Dieser Wert dient nur für die Mess-Operationen.

Achtung:
Die Komponente hat keinerlei Timer oder ähnliches. Es liegt deshalb beim Anwendungs Programm, dass bei der Aufzeichnung die Daten im entsprechenden Zeitabstand mittels SetBuffer eingeschrieben werden. TimeBase dient nur zur korrekten Darstellung der Daten der t-Achse und bei den Messfunktionen.

property ***DispStartTime*** : double;
DispStartTime dient zum Setzen der Display Startzeit innerhalb des Buffers/Aufzeichnung und damit zum zeitlichen horizontalen verschieben/scrollen des Display.

property ***DispEndTime*** : double;
Ähnliche Funktion wie DispStartTime.

property ***DispStartPos*** : LongInt;
DispStartPos dient, wie auch DispStartTime zum Setzen der Display Startposition innerhalb des Buffers/Aufzeichnung und damit zum zeitlichen horizontalen verschieben/scrollen des Display.

property ***DispEndPos*** : LongInt;
Ähnliche Funktion wie DispStartPos.

property ***PointerEnable*** : boolean;
Mit PointerEnable wird die Messlinie „Pointer“ sichtbar bzw. unsichtbar geschaltet.

property ***MarkerEnable*** : boolean;
Mit MarkerEnable wird die Messlinie „Marker“ sichtbar bzw. unsichtbar geschaltet.

property ***PointerTime*** : double;
Die Pointer Messlinie ist die eigentliche Messlinie. Die X-Koordinate (Zeit) dieser Linie wird relativ zur x-Koordinate der Markerlinie und absolut zum Buffer Anfang in dem entsprechenden Feld angezeigt. Mit PointerTime kann die absolute Zeit vom Programm ausgelesen bzw. gesetzt werden. Die Einheit ist Sekunden.

property ***PointerPos*** : LongInt;
Wie PointerTime nur dass die Eigenschaft nicht in sec sondern in Buffer Schritten angegeben ist.

property ***MarkerTime*** : double;
Die Marker Messlinie dient zur relativen Messung der t Werte im Zusammenhang mit der Pointer Messlinie. Mit MarkerTime kann die absolute Zeit vom Programm ausgelesen bzw. gesetzt werden. Die Einheit ist Sekunden.

property ***MarkerPos*** : LongInt;
Wie MarkerTime nur dass die Eigenschaft nicht in sec sondern in Buffer Schritten angegeben ist.

property ***TriggerPos*** : LongInt;
Die Trigger Linie ist nur zur Orientierung und kann vom Hauptprogramm beliebig verändert werden. Sie hat keinen Einfuss auf die Mess Operationen.

property ***ChanNamen*** : TStringList;

Diese StringList enthält die Kanalname, die am linken Rand des Displays erscheinen. Der Namen für Kanal 0 steht an der Position 0 der Liste. Beispiel in Logicana.dpr

property ***LogSize*** : TLogSize; (8, 16 bzw. 32 bits)

Speicherbreite des Buffers. Sollte vor der Veränderung von BufferSize auf den gewünschten Wert eingestellt werden. Die Anzahl der Samples errechnet sich aus BufferSize div LogSize (in Bytes). Sämtliche Lese- und Schreiboperationen beziehen sich auf dieses Property. Die zugrundeliegende Prozedur erwartet zwar per Definition immer ein 32-Bit Wert, es werden aber nur die niederwertigen bits, abhängig von LogSize, in den Buffer geschrieben. Das gleiche gilt für Leseoperationen. Der Rückgabewert ist zwar immer 32bit breit, aber nur die per LogSize definierten niederwertigen bytes werden besetzt. Der Rest ist don't care.

property ***BufferCount*** : LongInt;

Read only! Nach korrekter Einstellung von LogSize und BufferSize enthält BufferCount die max einschreibbare Anzahl von Samples.

property ***BufferSize*** : LongInt;

Grösse des Aufzeichnungs Buffers bzw. Anzahl der möglichen Aufzeichnung Werte. Ein Verändern dieses Wertes hat ein Löschen bzw. Freigabe der Buffer zur Folge und eine anschliessende Neuinitialisierung. BufferSize wird immer in Bytes angegeben. Maximale Grösse ist 64000 Bytes. Die mögliche Sample Anzahl ergibt sich aus Buffersize div Speicherbreite (8, 16 bzw. 32 bit). Die Speicherbreite wird durch das property „LogSize" eingestellt.

```
GetMem(Buffer, BufferSize);
```

Mit der function

```
GetBuffer(adrs : LongInt) : LongInt;
```

kann der Buffer ausgelesen werden.

Mit der procedure

```
SetBuffer(adrs : LongInt; val : LongInt);
```

werden 32 Bits in den Buffer geschrieben, und zwar an die Stelle auf die „adrs" zeigt. Im Normalfall wird der Buffer, beginnend mit der Adresse 0 forlaufend beschrieben bis zur maximalen Adresse BufferCount -1. Dann wird ein DispRefresh aufgerufen um das Display neu zu zeichnen. Bei sehr langsamer Aufzeichnungsrate kann auch nach jedem Schreibvorgang ein DispRefresh aufgerufen werden.

property ***ZoomEnable*** : boolean;

Sperrt die Zoomfunktion der mittleren Maustaste (oder Shift + Linke Maustaste) oder gibt diese frei.

property ***onZoomSet*** : TNotifyEvent;

Ein Event/Ereignis, das auftritt, wenn der Zoom sich ändert. Dieses Erreignis ist dann wichtig, wenn die Anwendung „private Malereien" oder Texte auf dem Display ausgeben will.

property ***onRefresh*** : TNotifyEvent;

Ein Event/Erreignis, das auftritt, wenn das Display sich neu zeichnet. Sonst wie unter onZoomSet.

Methoden/Prozeduren:

function ***LoadFromFile***(FName : string) : boolean;

Lädt eine zuvor abgespeicherte oder durch ein anderes Programm generierte Konserve aus einer Datei. FName muss u.U. den kompletten Pfad und den Dateinamen nach DOS-Konventionen enthalten. Bei Dateien, die durch andere Programme generiert werden, muss die Dateistruktur folgendermassen beschaffen sein bzw. generiert werden:

Datei-Header

```
str:= 'LogicAna';
str:= str + '|' + FloatToStr(TimeBase);
str:= str + '|' + FloatToStr(TriggerPos);
str:= str + '|' + IntToStr(integer(FLogSize));
str:= str + '|' + IntToStr(BufferSize);
str:= str + '@' + ^Z;
BlockWrite(DestFile, Str, ord(Str[0])+1);
```

Channel-Namen

```
for idx:= 0 to 31 do begin
   str:= ChanNamen.strings[idx];
   if str = '' then str:= 'Chan ' + IntToStr(idx);
   BlockWrite(DestFile, Str, ord(Str[0])+1);
end;
```

Daten-Block

```
for idx:= 0 to BufferSize-1 do begin
   val:= Buffer8^[idx];
   BlockWrite(DestFile, val, 1);
end;
```

function ***SaveToFile***(FName : string) : boolean;
Schreibt den kompletten Buffer mit den zugehörigen Record Parametern in die Datei „FName“. Dateistruktur siehe „LoadFromFile“.

procedure ***DispRefresh***;
Erzwingt ein Neuzeichnen des Displays mit Standard Parametern.

procedure ***ZoomRefresh***;
Erzwingt ein Neuzeichnen des Displays mit den Zoomparametern.

function ***GetDispRect*** : Trect;
Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche. Siehe Beispiel bei „property canvas“.

function ***GetBuffer***(adrs : LongInt) : LongInt;
Mit der function kann der Buffer an der Adresse „adrs“ ausgelesen werden. Siehe auch property „LogSize“ !!

procedure ***SetBuffer***(adrs : LongInt; val : LongInt);
Mit SetBuffer werden 8/16/32 Bits in den Buffer geschrieben, und zwar an die Stelle auf die „adrs“ zeigt. Im Normalfall wird der Buffer, beginnend mit der Adresse 0 forlaufend beschrieben bis zur maximalen Adresse BufferCount -1. Dann wird ein DispRefresh aufgerufen um das Display neu zu zeichnen. Bei sehr langsamer Aufzeichnungsrate kann auch nach jedem Schreibvorgang ein DispRefresh aufgerufen werden. Siehe auch property „LogSize“ !!

Procedure ***DoStepLeft***;
Horizontal Scroll einen Schritt bzw. 1x TimeBase nach links bzw. Richtung Aufzeichnungs-start. In beiden Modi Grafik und Text wirksam.

Procedure ***DoStepRight***;
Horizontal Scroll einen Schritt bzw. 1x TimeBase nach rechts bzw. Richtung Aufzeichnungs-start. In beiden Modi Grafik und Text wirksam.

Procedure ***DoPageLeft***;

Horizontal Scroll eine Bildschirmseite nach links bzw. Richtung Aufzeichnungsstart. In beiden Modi Grafik und Text wirksam.

Procedure ***DoPageRight***;
Horizontal Scroll eine Bildschirmseite nach rechts bzw. Richtung Aufzeichnungsstart. In beiden Modi Grafik und Text wirksam.

Variable

```
ZoomStartX  : LongInt;
ZoomEndX    : LongInt;
ZoomStartY  : LongInt;
ZoomEndY    : LongInt;
```

Mit diesen 4 Variablen kann ein Zoomausschnit aus dem Buffer gewählt werden. Dabei ist unbedingt zu beachten, dass sowohl die y-Grenzen (0 = Kanal0 und 31 = Kanal31) als auch die t-Grenzen (>= 0 und < BufferLen) nicht überschritten werden. Mit ZoomRefresh kann der gewählte Ausschnitt sichtbar gemacht werden.

Es ist zu beachten, dass die Komponente, im Gegensatz zu den anderen grafischen Komponenten, keinen Standard Zoom besitzt. Das bedeutet, dass ein einmal gewählter Zoomausschnitt nur durch einen neuen Zoomauschnitt verändert werden kann. Soll der ursprüngliche Zoom wieder hergestellt werden, so muss das Hauptprogramm diese 4 Parameter zuerst sichern, dann den neuen Zoom einstellen und anschliessend wieder mit den zuvor gesicherten Werten den alten Zustand wieder herstellen.

Plotter

HPGL-Interpreter mit komfortablen Druckmöglichkeiten. Kann die wesentlichen HPGL-Plotterbefehle graphisch auf dem Bildschirm oder jedem Windows-tauglichen Drucker ausgeben. Einstellbare Blattformate, Stiftstärken und Farben. Keine Beschränkung in der Dateigrösse.

Komponenten Deklaration

```
type
  TPlotter   = class(TRahmen)
  private
    { Private-Deklarationen }
    painted     : Boolean;
    FOnZoomSet  : TNotifyEvent;
    FOnRefresh  : TNotifyEvent;
    FZoomEna    : Boolean;
    FPen1Col    : TColor;
    FPen1Width  : double;
    FPen2Col    : TColor;
    FPen2Width  : double;
    FPen3Col    : TColor;
    FPen3Width  : double;
    FPen4Col    : TColor;
    FPen4Width  : double;
    FPen5Col    : TColor;
    FPen5Width  : double;
    FPen6Col    : TColor;
    FPen6Width  : double;
    ZoomPos1    : TPoint;
    ZoomPos2    : TPoint;
    CopyBmp     : TBitMap;
    dispRect    : TRect;
    SheetX      : LongInt;
    SheetY      : LongInt;
    PlotList    : TList;
{    PlotList    : SingleList; }

    Procedure SetAspect;
    procedure SetFullSize(pValue: Boolean);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetNormBlatt(pValue : TNormBlatt);
    function  FHardCopy : TBitmap;
    procedure FClearBmp(b : TBitmap);
    procedure FreePlotList;
    procedure FLoadHPGL(fn : string);
    function  splitHpgl(var str : string; var pen : Integer;
                        var down : boolean;
                        var hpgx, hpgy : LongInt) : Boolean;
    procedure ScalePnt(XL,YL : double; var XP,YP : LongInt);
  protected
    { Protected-Deklarationen }
    XStartP    : LongInt;
    YStartP    : LongInt;
    XAbsP      : LongInt;
    YAbsP      : LongInt;
    XStartL    : double;
    YStartL    : double;
    XAbsL      : double;
```

```
    YAbsL      : double;
    zooming    : boolean;
    MouseX,
    MouseY     : LongInt;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    FColDisp    : TColor;
    ZoomStartX  : double;
    ZoomStartY  : double;
    ZoomEndX    : double;
    ZoomEndY    : double;
    FNormBlatt  : TNormBlatt;
    FFileName   : string;
    FFullSize   : boolean;

    procedure SetZoomDim;
    procedure PlotAll;
    procedure PrintAll;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property canvas;
    property Align;
    procedure StyleChanged(Sender : TObject);
    property LoadHPGL   : String      read FFileName  write FLoadHPGL;
    property NormBlatt  : TNormBlatt  read FNormBlatt write SetNormBlatt;
    property ColDisplay : TColor      read FColDisp   write SetColDisp;
    property Pen1Color  : TColor      read FPen1Col   write FPen1Col;
    property Pen1Width  : double      read FPen1Width write FPen1Width;
    property Pen2Color  : TColor      read FPen2Col   write FPen2Col;
    property Pen2Width  : double      read FPen2Width write FPen2Width;
    property Pen3Color  : TColor      read FPen3Col   write FPen3Col;
    property Pen3Width  : double      read FPen3Width write FPen3Width;
    property Pen4Color  : TColor      read FPen4Col   write FPen4Col;
    property Pen4Width  : double      read FPen4Width write FPen4Width;
    property Pen5Color  : TColor      read FPen5Col   write FPen5Col;
    property Pen5Width  : double      read FPen5Width write FPen5Width;
    property Pen6Color  : TColor      read FPen6Col   write FPen6Col;
    property Pen6Width  : double      read FPen6Width write FPen6Width;
    property ZoomEnable: boolean      read FZoomEna   write FZoomEna;
    property onZoomSet : TNotifyEvent read FOnZoomSet write FOnZoomSet;
    property onRefresh : TNotifyEvent read FOnRefresh write FOnRefresh;
    Property HardCopy  : TBitMap      read FHardCopy  write FClearBmp;
    Property FullSize  : Boolean      read FFullSize  write SetFullSize;
  end;
```

Oscillo

Komponente mit allen Funktionen eines SpeicherOszilloskops. 4 Kanäle, Zoomfunktion, Speichern, Laden und Drucken der Daten. Sehr leistungsfähig und komfortabel. Einstellbar von einfacher Kurven-Ausgabe bis zum kompletten Oszillographen.

TOscillo ist in erster Linie zur Darstellung von sog. Schnappschüssen gedacht, wie z.B. ein echtes Digital Scope. Eine kontinuierliche Darstellung wie z.B. bei einem analogen Oscilloscope ist zur Zeit nur eingeschränkt möglich.

Komponenten Deklaration

```
type
  TBufferA    = array[0..65000] of single;
  osDispType  = (complete, fullSize, topScreen, bottomScreen);
  osGraphMode = (gmDraw, gmNone);
  TChanMode   = (DC, AC, GND);

type
  TOscillo = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnZoomSet  : TNotifyEvent;
    FOnRefresh  : TNotifyEvent;
    FOnBuffFull : TNotifyEvent;
    FOnRecStart : TNotifyEvent;

    CopyBmp     : TBitMap;

    FColCapt    : TColor;
    FColTxtCapt : TColor;
    FCaption    : String32;
    FDisplayTyp : osDispType;
    FNormYsize  : double;
    FSamplesXdiv : integer;
    FMeasChan   : integer;
    FMeasEna    : boolean;
    FChanAddMode : double;
    FPointerPosX : double;
    FPointerPosY : double;
    FColPointer  : TColor;
    FColDisplay : TColor;
    FColRaster  : TColor;
    FScalaPartX : Integer;
    FScalaPartY : Integer;
    FMarkerPosX : double;
    FMarkerPosY : double;
    FColMarker  : TColor;
    FZoomEna    : Boolean;
    FZoomFull   : Boolean;
    FZoomLight  : Boolean;
    FRecStart   : Boolean;
    FJoined     : Boolean;
    FTimeBase   : double;
    FTimeStart  : double;
    FTimeEnd    : double;
    FChan1Col   : TColor;
    FChan2Col   : TColor;
```

```
    FChan3Col   : TColor;
    FChan4Col   : TColor;
    FChan1Inv   : Boolean;
    FChan2Inv   : Boolean;
    FChan3Inv   : Boolean;
    FChan4Inv   : Boolean;
    FChan1Mode  : TChanMode;
    FChan2Mode  : TChanMode;
    FChan3Mode  : TChanMode;
    FChan4Mode  : TChanMode;
    FChan1Off   : Boolean;
    FChan2Off   : Boolean;
    FChan3Off   : Boolean;
    FChan4Off   : Boolean;
    FChan1Lock  : Boolean;
    FChan2Lock  : Boolean;
    FChan3Lock  : Boolean;
    FChan4Lock  : Boolean;
    FChan1Gain  : double;
    FChan2Gain  : double;
    FChan3Gain  : double;
    FChan4Gain  : double;
    FChan1Offs  : double;
    FChan2Offs  : double;
    FChan3Offs  : double;
    FChanAddA   : integer;
    FChanAddB   : integer;
    FChan4Offs  : double;
    FBufferLen  : integer;
    FBufferFill : integer;
    FGraphMode  : osGraphMode;
    Chan1Buffer : ^TBufferA;
    Chan2Buffer : ^TBufferA;
    Chan3Buffer : ^TBufferA;
    Chan4Buffer : ^TBufferA;

    painted     : Boolean;
    XStartP     : LongInt;
    YStartP     : LongInt;
    XAbsP       : LongInt;
    YAbsP       : LongInt;
    YStartL     : double;
    YAbsL       : double;
    DspParmRect,
    DiffTimeRect,
    DeltaRect,
    SizeRect,
    CaptRect,
    topRect,
    dispRect,
    bottomRect  : TRect;
    zooming     : boolean;
    MouseX,
    MouseY      : LongInt;
    PointerPx   : integer;
    PointerPy   : integer;
    MarkerPx    : integer;
    MarkerPy    : integer;
    ZoomPosX1   : LongInt;
    ZoomPosX2   : LongInt;
    ZoomPosY1   : LongInt;
    ZoomPosY2   : LongInt;
    oldZoom     : boolean;
```

```
    curTimeBase : double;
    curChan1Gain: double;
    curChan2Gain: double;
    curChan3Gain: double;
    curChan4Gain: double;
    oldFullStart: double;
    StartPos    : LongInt;
    EndPos      : LongInt;
    DispPoints  : LongInt;
    Chan1Invers : double;
    Chan2Invers : double;
    Chan3Invers : double;
    Chan4Invers : double;

    Procedure DrawSize;
    procedure DrawSignals(pos, step : integer);
    procedure SetNormY(pValue: double);
    procedure SetSamplesXdiv(pValue : integer);
    procedure SetJoined(pValue: Boolean);
    procedure SetRecStart(pValue: Boolean);
    procedure SetZoomFull(pValue: Boolean);
    procedure SetZoomLight(pValue: Boolean);
    procedure SetTimeBase(pValue: double);
    procedure SetTimeStart(pValue: double);
    procedure SetTimeEnd(pValue: double);
    procedure SetChan1Offs(pValue: double);
    procedure SetChan2Offs(pValue: double);
    procedure SetChan3Offs(pValue: double);
    procedure SetChan4Offs(pValue: double);
    procedure SetChan1Gain(pValue: double);
    procedure SetChan2Gain(pValue: double);
    procedure SetChan3Gain(pValue: double);
    procedure SetChan4Gain(pValue: double);
    procedure SetChan1Off(pValue: Boolean);
    procedure SetChan2Off(pValue: Boolean);
    procedure SetChan3Off(pValue: Boolean);
    procedure SetChan4Off(pValue: Boolean);
    procedure SetChan1Inv(pValue: Boolean);
    procedure SetChan2Inv(pValue: Boolean);
    procedure SetChan3Inv(pValue: Boolean);
    procedure SetChan4Inv(pValue: Boolean);
    Procedure SetDisplayTyp(pValue : osDispType);
    Procedure SetCaption(pValue : String32);
    Procedure SetChan1Col(pValue : TColor);
    Procedure SetChan2Col(pValue : TColor);
    Procedure SetChan3Col(pValue : TColor);
    Procedure SetChan4Col(pValue : TColor);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColTxtCapt(pValue : TColor);
    procedure DrawTextCapt;
    procedure SetBufferLen(pValue : integer);
    procedure SetBufferFill(pValue : integer);
    procedure SetMeasChan(pValue: integer);
    procedure SetMeasEna(pValue: Boolean);
    procedure SetChanAddMode(pValue: double);
    procedure SetPointerPosX(pValue: double);
    procedure SetMarkerPosX(pValue: double);
    procedure SetPointerPosY(pValue: double);
    procedure SetMarkerPosY(pValue: double);
    procedure DrawPointer;
    procedure DrawMarker;
    procedure ShowPos;
    Procedure SetScalaPartX(pValue : integer);
```

```
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColMarker(pValue : TColor);
    Procedure SetColDisplay(pValue : TColor);
    Procedure SetColRaster(pValue : TColor);
    Procedure SetScalaPartY(pValue : integer);
    function  FHardCopy : TBitmap;
    procedure FClearBmp(b : TBitmap);
  protected
    { Protected-Deklarationen }
    PROCEDURE ClipMouse(clip : boolean);
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    minStepX    : double;
    ZoomStartX  : double;
    ZoomEndX    : double;
    ZoomStartY  : double;
    ZoomEndY    : double;
    DisplayTime : double;
    DisplayYs   : double;

    function  LoadFromFile(FileName : string) : boolean;
    function  SaveToFile(FileName : string) : boolean;
    procedure ChanAdd(ChanA, ChanB : Integer);
    procedure DispRefresh;
    function  GetDispRect : TRect;
    procedure ScalePnt(XL,YL : double; var XP,YP : LongInt);
    procedure PointToScale(px, py : integer; var sx, sy : double);
    function  GetBuffer(adrs : integer; var val1, val2, val3, val4 : single) :
boolean;
    procedure SetBuffer(val1, val2, val3, val4 : single);
    function  ChangeBuffer(adrs : integer; val1, val2, val3, val4 : single) :
boolean;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property canvas;
    property Align;
    property DrawMode;
    procedure StyleChanged(Sender : TObject);
    Property HardCopy   : TBitMap   read FHardCopy    write FClearBmp;
    Property Caption    : String32  read FCaption     write SetCaption;
    property DisplayTyp : osDispType read FDisplayTyp write SetDisplayTyp;
    property Chan1Off   : boolean   read FChan1Off    write SetChan1Off;
    property Chan2Off   : boolean   read FChan2Off    write SetChan2Off;
    property Chan3Off   : boolean   read FChan3Off    write SetChan3Off;
    property Chan4Off   : boolean   read FChan4Off    write SetChan4Off;
    property Chan1Col   : TColor    read FChan1Col    write SetChan1Col;
    property Chan2Col   : TColor    read FChan2Col    write SetChan2Col;
    property Chan3Col   : TColor    read FChan3Col    write SetChan3Col;
    property Chan4Col   : TColor    read FChan4Col    write SetChan4Col;
    property Chan1Inv   : boolean   read FChan1Inv    write SetChan1Inv;
    property Chan2Inv   : boolean   read FChan2Inv    write SetChan2Inv;
    property Chan3Inv   : boolean   read FChan3Inv    write SetChan3Inv;
    property Chan4Inv   : boolean   read FChan4Inv    write SetChan4Inv;
    property Chan1Lock  : boolean   read FChan1Lock   write FChan1Lock;
    property Chan2Lock  : boolean   read FChan2Lock   write FChan2Lock;
```

```
    property Chan3Lock  : boolean   read FChan3Lock   write FChan3Lock;
    property Chan4Lock  : boolean   read FChan4Lock   write FChan4Lock;
    property Chan1Mode  : TChanMode read FChan1Mode   write FChan1Mode;
    property Chan2Mode  : TChanMode read FChan2Mode   write FChan2Mode;
    property Chan3Mode  : TChanMode read FChan3Mode   write FChan3Mode;
    property Chan4Mode  : TChanMode read FChan4Mode   write FChan4Mode;
    property Chan1Gain  : double    read FChan1Gain   write SetChan1Gain;
    property Chan2Gain  : double    read FChan2Gain   write SetChan2Gain;
    property Chan3Gain  : double    read FChan3Gain   write SetChan3Gain;
    property Chan4Gain  : double    read FChan4Gain   write SetChan4Gain;
    property Chan1Offs  : double    read FChan1Offs   write SetChan1Offs;
    property Chan2Offs  : double    read FChan2Offs   write SetChan2Offs;
    property Chan3Offs  : double    read FChan3Offs   write SetChan3Offs;
    property Chan4Offs  : double    read FChan4Offs   write SetChan4Offs;
    property TimeBase   : double    read FTimeBase    write SetTimeBase;
    property TimeStart  : double    read FTimeStart   write SetTimeStart;
    property TimeEnd    : double    read FTimeEnd     write SetTimeEnd;
    property ColCapt    : TColor    read FColCapt     write SetColCapt;
    property ColTxtCapt : TColor    read FColTxtCapt  write SetColTxtCapt;
    property ColDisplay : TColor    read FColDisplay  write SetColDisplay;
    property ColRaster  : TColor    read FColRaster   write SetColRaster;
    property ColPointer : TColor    read FColPointer  write SetColPointer;
    property ColMarker  : TColor    read FColMarker   write SetColMarker;
    property ScalaPartX : Integer   read FScalaPartX  write SetScalaPartX;
    Property MeasChan   : integer   read FMeasChan    write SetMeasChan;
    Property MeasEnable : boolean   read FMeasEna     write SetMeasEna;
    Property ChanAddMode : double   read FChanAddMode write SetChanAddMode;
    property ChanAddA   : Integer   read FChanAddA    write FChanAddA;
    property ChanAddB   : Integer   read FChanAddB    write FChanAddB;
    Property PointerPosX : double   read FPointerPosX write SetPointerPosX;
    Property MarkerPosX : double    read FMarkerPosX  write SetMarkerPosX;
    Property PointerPosY : double   read FPointerPosY write SetPointerPosY;
    Property MarkerPosY : double    read FMarkerPosY  write SetMarkerPosY;
    property ScalaPartY : Integer   read FScalaPartY  write SetScalaPartY;
    property BufferLen  : integer   read FBufferLen   write SetBufferLen;
    property BufferFill : integer   read FBufferFill  write SetBufferFill;
    property ZoomEnable : boolean   read FZoomEna     write FZoomEna;
    property ZoomFull   : boolean   read FZoomFull    write SetZoomFull;
    property ZoomLight  : boolean   read FZoomLight   write SetZoomLight;
    property RecStart   : boolean   read FRecStart    write SetRecStart;
    property Joined     : boolean   read FJoined      write SetJoined;
    property GraphMode  : osGraphMode read FGraphMode write FGraphMode;
    property NormYsize  : double    read FNormYsize   write SetNormY;
    property SamplesXdiv : integer   read FSamplesXdiv write SetSamplesXdiv;
    property onZoomSet : TNotifyEvent read FOnZoomSet write FOnZoomSet;
    property onRefresh : TNotifyEvent read FOnRefresh write FOnRefresh;
    property onBuffFull: TNotifyEvent read FOnBuffFull write FOnBuffFull;
    property onRecStart : TNotifyEvent read FOnRecStart write FOnRecStart;
end;
```

Eigenschaften/Properties:

property ***Align***;

Delphi kompatibel: automatische Anpassung etc.

property ***canvas*** : TCanvas;

Zeichenfläche der Komponente. Der Entwickler kann mit den vorhandenen Properties von „canvas“ eigene Zeichenoperationen durchführen und Texte plazieren. Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche. Siehe auch ScalePnt.

Beispiel:

```
var r : TRect;
begin
  r:= Oscillo1.GetDispRect;
  with r do begin
    Oscillo1.Canvas.MoveTo(left, top);
    Oscillo1.Canvas.LineTo(right, bottom);
    Oscillo1.Canvas.TextOut(left, top + (bottom-top) div 2, 'Text');
  end;
end;
```

property ***HardCopy*** : TBitMap;

HardCopy liefert die Komponente als BitMap. Hiermit lassen sich diverse Datei- oder Print Operationen durchführen. Das folgende Beispiel speichert die Komponente als BitMap File auf die Disk. Weitere Beispiele sind im Projekt „Oscar.dpr“ im Formular „Omain“ zu finden.

Beispiel:

```
Oscillo1.HardCopy.SaveToFile('FileName.bmp');
Oscillo1.HardCopy:= nil;                          (* gibt Resource frei *)
```

property ***Caption*** : String32;

Caption ist die Überschrift der Komponente und erscheint im obersten Textfenster.

property ***ColCapt*** : Tcolor;

Hintergrund Farbe der Überschrift

property ***ColTxtCapt*** : Tcolor;

Farbe der Überschrift

property ***DisplayTyp*** : osDispType;

DisplayTyp stellt die Erscheinungsform von Oscillo ein. Es gibt 4 Möglichkeiten:

complete : Alle Teile von Oscillo werden dargestellt.
fullSize : Nur das Display selbst wird dargestellt.
topScreen : Das Display und „Caption“ wird dargestellt.
bottomScreen Das Display und die Kanalnamen werden dargestellt.

property ***GraphMode*** : osGraphMode;

bestimmt die Darstellung während der Aufzeichnung. Mit gmNone kann der Buffer mit SetBuffer komplett neu beschrieben werden, wobei das alte Bild bestehen bleibt, bis ein DispRefresh durchgeführt wird. Es existieren 2 Modi:

gmNone : Während des Beschreibens des Buffers mit SetBuffer wird nicht gezeichnet
gmDraw : Während des Beschreibens des Buffers mit SetBuffer wird gleichzeitig gezeichnet.

property ***Chan1Col*** : Tcolor;
property ***Chan2Col*** : Tcolor;
property ***Chan3Col*** : Tcolor;
property ***Chan4Col*** : Tcolor;

Farbe des Text/Namen Kanal x sowie der Kurve von Kanal x

property ***Chan1Off*** : boolean;
property ***Chan2Off*** : boolean;
property ***Chan3Off*** : boolean;
property ***Chan4Vis*** : boolean;

Schaltet die Darstellung der Kurve Kanal x ein bzw. aus

property ***Chan1Inv*** : boolean;
property ***Chan2Inv*** : boolean;

property ***Chan3Inv*** : boolean;
property ***Chan4Inv*** : boolean;

Invertiert die Darstellung des entsprechenden Kanals. D.h. aus einer y-Koordinate von +1.5 wird in der Darstellung -1.5.

property ***Chan1Lock*** : boolean;
property ***Chan2Lock*** : boolean;
property ***Chan3Lock*** : boolean;
property ***Chan4Lock*** : boolean;

Sperrt beim Neubeschreiben des Buffers mit „SetBuffer“ den zugehörigen Kanal gegen Überschreiben. Der alte Wert bleibt erhalten.

property ***Chan1Mode*** : TChanMode;
property ***Chan2Mode*** : TChanMode;
property ***Chan3Mode*** : TChanMode;
property ***Chan4Mode*** : TChanMode;

Es existieren 3 Modi:
DC, AC : Beide Modi werden von der Komponente identisch behandelt bzw. ignoriert
GND : Beim Schreiben mit SetBuffer wird in den jeweiligen Kanal eine 0 eingeschrieben.

property ***Chan1Gain*** : double;
property ***Chan2Gain*** : double;
property ***Chan3Gain*** : double;
property ***Chan4Gain*** : double;

Diese Eigenschaft gibt den y-Masstab an in Volt/div. Diese Werte werden nur für die Mess-Operationen benötigt. Intern wird mit skalierten Werten gerechnet (siehe NormYsize). Die gesamte logische Displayhöhe für einen einzelnen Kanal ergibt sich aus ScalaPartY x ChanxGain.
Beispiel: ScalaPartY = 10, Chan1Gain = 1Volt Display „Y“ = 10Volt

property ***Chan1Offs*** : double;
property ***Chan2Offs*** : double;
property ***Chan2Offs*** : double;
property ***Chan2Offs*** : double;

Die Eigenschaft ChanxOffs bestimmt die aktuelle y-Lage des entsprechenden Kanals. Dabei kann sich der gewählte Offset im Bereich (-2 * NormYsize) bis (+2 * NormYsize) bewegen. NormYsize x 2 ist die skalierte Display Höhe. Beispiel für Offset Manipulation in „Omain.pas“.

property ***ChanAddA*** : Integer;
property ***ChanAddB*** : Integer;

Die beiden properties gehören absolut zusammen, denn sie bestimmen gemeinsam die Kanäle, die addiert werden sollen:

ChanAddA	ChanAddB	Addition
0	0	keine
1	2	ch1 + ch2
1	3	ch1 + ch3
1	4	ch1 + ch4
2	3	ch2 + ch3
2	4	ch2 + ch4
3	4	ch3 + ch4

property ***ChanAddMode*** : double;

Dieses property ist ein Multiplikator für den mit ChanAddB angewählten Kanal. Im Normalfall hat diese Eigenschaft den Wert +1.0 oder -1.0. Damit wird eine Addition oder Subtraktion erzielt. Für spezielle Anwendungungen ist jedoch auch ein beliebiger anderer Wert möglich.

property ***BufferFill*** : Integer;
Gibt die Anzahl der eingeschriebenen yt-Werte zurück, Read Only. Die Eigenschaft ist gleichzeitig der Schreib-Pointer bei „SetBuffer“.

property ***ColDisplay*** : Tcolor;
Hintergrund Farbe des Displays

property ***ColRaster*** : Tcolor;
Farbe des Rasters des Displays

property ***ColPointer*** : Tcolor;
Farbe der Messlinien des Displays

property ***ColMarker*** : Tcolor;
Farbe der Bezugslinien für die Messlinien des Displays

property ***ScalaPartX*** : Integer;
Anzahl der vertikalen Orientierungslinien auf der t-Achse. (1..20)

property ***ScalaPartY*** : Integer;
Anzahl der vertikalen Orientierungslinien auf der y-Achse. (1..20)

property ***NormYsize*** : double;
Um die Zeichengeschwindigkeit der Komponente nicht zu sehr zu beeinträchtigen und um möglichst universell zu sein, wird intern mit normierten Werten gearbeitet. NormYsize beschreibt den y-Masstab, d.h. das Display läuft in der y-Dimension von -NormYsize bis +NormYsize. Im Beispiel „Oscar“ von -4095 bis +4095. Alle Datenoperationen mit den Buffern (Lesen/Schreiben) laufen auf dieser Basis ab. Ein Punkt mit der y-Koordinate 0 liegt exakt in der vertikalen Display Mitte, ein Punkt an der Stelle 4095 am obersten Displayrand und einer mit -4095 am unteren.
Die Komponente selbst rechnet grundsätzlich nur mit diesen Werten. Die Eigenschaften **ChanxGain** dienen nur für die Messoperation.
Beispiel:
NormYsize = 2000
ScalaPartY = 10
Chan1Gain = 0.1 (= 100mV/Div)

Um einen Punkt an die y-Position +0.25Volt zu setzen muss z.B. bei SetBuffer die entsprechende y-Koordinate den Wert +1000 haben :
(2 x NormYsize / ScalaPartY) x (0.25 / Chan1Gain) = 1000
(2 x 2000 / 10) x (0.25V / 0.1V) = 1000
Da NormYsize und ScalaPartY in der Regel konstant sind, kann man vereinfachen :
400 x (0.25V / 0.1V)

property ***SamplesXdiv*** : integer;
Gibt an wieviel Messpunkte in der horizontalen x- bzw. t-Achse pro Teilung (ScalaPartX) dargestellt werden sollen. ScalaPartX x TimeBase = Zeitintervall für ein komplettes Display bzw. TimeEnd - TimeStart.
Im Demo Programm „Oscar“ ist SamplesXdiv = 100, ScalaPartX = 10 und TimeBase = 1msec.
Das Display hat deshalb eine Anzeige Spannweite von 10 x 1msec -> 10msec.
Das Hauptprogramm muss, um ein Raster zu füllen, Anzahl SamplesXdiv Messpunkte liefern, das bedeutet alle 10usec einen Wert. (TimeBase / SamplesXdiv).

property ***TimeBase*** : double;
Gibt den zeitlichen Abstand zwischen zwei vertikalen Orientierungslinien auf der t-Achse an.
TimeBase / SamplesXdiv ergibt die Zeit zwischen 2 Messpunkte (Messintervall).
Achtung:
Die Komponente hat keinerlei Timer oder ähnliches. Es liegt deshalb beim Anwendungs Programm, dass im Aufzeichnungsmodus die Daten im entsprechenden Zeitabstand mittels

SetBuffer eingeschrieben werden. TimeBase dient nur zur korrekten Darstellung der Daten der x- bzw. t-Achse und bei den Messfunktionen.

property ***TimeStart*** : double;
TimeStart dient zum Setzen der Display Startzeit innerhalb des Buffers/Aufzeichnung und damit zum zeitlichen horizontalen verschieben/scrollen des Display.

property ***TimeEnd*** : double;
Ähnliche Funktion wie TimeStart.

property ***MeasChan*** : integer;
Da jeder Kanal beim Messen sein eigenes Y-Koordinatensystem mitbringt, ist es notwendig, bevor die Fadenkreuze mit MeasEnable freigeschaltet werden, Mit MeasChan den gewünschten Kanal (1..4) zu bestimmen.

property ***MeasEnable*** : boolean;
Mit MeasEnable wird die Möglichkeit, mit der linken bzw. rechten Maustaste die Mess-Fadenkreuze zu bewegen freigeschaltet bzw. gesperrt. Ebenso werden die Messfadenkreuze sichtbar/unsichtbar geschaltet.

property ***PointerPosX*** : double;
Das Pointer Fadenkreuz ist das eigentliche Messfadenkreuz. Die X-Koordinate (Zeit) dieses Kreuzes wird relativ zur x-Koordinate des Markerkreuzes und absolut zum Aufzeichnungsstart in dem entsprechenden Feld angezeigt. Mit PointerPosX kann die absolute x-Position vom Programm ausgelesen bzw. gesetzt werden. Die Einheit ist Sekunden.

property ***MarkerPosX*** : double;
Das Marker Fadenkreuz dient zur relativen Messung der y und t Werte im Zusammenhang mit dem Pointer Fadenkreuz. Mit MarkerPosX kann die absolute x-Position vom Programm ausgelesen bzw. gesetzt werden. Die Einheit ist Sekunden.

property ***PointerPosY*** : double;
Das Pointer Fadenkreuz ist das eigentliche Messfadenkreuz. Die Y-Koordinate dieses Kreuzes wird relativ zur y-Koordinate des Markerkreuzes und absolut in dem entsprechenden Feld angezeigt. Mit PointerPosY kann die absolute y-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***MarkerPosY*** : double;
Das Marker Fadenkreuz dient zur relativen Messung der y und t Werte im Zusammenhang mit dem Pointer Fadenkreuz. Mit MarkerPosY kann die absolute y-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***BufferLen*** : Integer;
Grösse des Aufzeichnungs Buffers bzw. Anzahl der möglichen Aufzeichnungs Records. Ein Verändern dieses Wertes hat ein Löschen bzw. Freigabe der Buffer zur Folge und eine anschliessende Neuinitialisierung. Der tatsächliche Speicherverbrauch des Buffers kann aus der Initialisierung errechnet werden: Maximaler Grösse ist 16000 Samples.

GetMem(Chan1Buffer, BufferLen * SizeOf(Single));

Mit der function
```
GetBuffer(adrs : integer; var val1, val2, val3, val4 : single) : boolean;
```
kann der Buffer ausgelesen werden. GetBuffer wird true, wenn „adrs“ eine gültigen Wert hat.

Mit der procedure
```
SetBuffer(val1, val2, val3, val4 : single);
```
werden die 4 Werte in den Buffer geschrieben, und zwar an die Stelle auf die der Startpointer „BufferFill“ zeigt. Der Pointer wird automatisch weitergeführt. Da gleichzeitig ein Zeichnungs-vorgang auf das Display erfolgt, ist diese Operation nur im „Record“ Modus zulässig.

Mit der function

```
ChangeBuffer(adrs : integer; val1, val2, val3, val4 : single) : boolean;
```

werden 4 Werte an die Stelle „adrs“ geschrieben. Der Pointer wird nicht verändert, solange in einen schon beschriebenen Teil des Buffers geschrieben wird. Ein Schreiben auf einen noch nicht beschriebenen Teil des Buffers ist möglich, setzt aber dann den Schreib-Pointer „BufferFill“ auf die neue Adresse. Ein gleichzeitiges Zeichnen der neuen Werte erfolgt nicht.

property ***StartRec*** : boolean;

Wie schon angeführt, ist ein Schreiben von Werten in den Buffer mit zugehöriger Pointer-Fortführung und Anzeige auf dem Display nur sinnvoll im Record Modus. Durch setzen von StartRec auf true wird dieser Modus initialisiert, der Schreib-Pointer wird zurückgesetzt, die Buffer gelöscht etc. etc. Nach dem Aufzeichnungsvorgang mittels Aufrufe der Methode SetBuffer muss der Record Modus wieder zurückgesetzt werden durch StartRec:= false. Erst dann können Zoom-, Mess- und Speicherfunction ausgeführt werden.

Beispiel: Oscar.dpr Omain.pas

property ***ZoomEnable*** : boolean;

Sperrt die Zoomfunktion der mittleren Maustaste (oder Shift + Linke Maustaste) oder gibt diese frei.

property ***ZoomLight*** : boolean;

Im nicht-gezoomten Modus wird die zuletzt eingestellte Zoom-Area mittels eines Rahmens dargestellt. Ist wichtig zum Grössenvergleich, Auffinden etc.

property ***ZoomFull*** : boolean;

Schaltet um zwischen Normaldarstellung und gezoomter Darstellung.
Beispiel in Oscar.dpr.

property ***Joined*** : boolean;

Bestimmt die Zeichnungsart der Kurven auf dem Display. Wenn Joined = true dann werden die einzelnen Messpunkte mittels einer Linie miteinander verbunden. Im anderen Fall werden die Messpunkte nur als Pixel/Punkt gezeichnet. Joined ist relativ langsam, hat aber den Vorteil der besseren Darstellung.

property ***onBuffFull***: TNotifyEvent;

Der Event (Ereignis) wird aufgerufen, wenn beim kontinuierlichen Schreiben in die Buffer mit SetBuffer das Bufferende erreicht ist.

property ***onRecStart*** : TNotifyEvent;

Das Ereignis zeigt an, dass der Aufnahme Modus (StartRec = true) gestartet wurde.

property ***onZoomSet*** : TNotifyEvent;

Ein Event/Ereignis, das auftritt, wenn der ZoomMode sich ändert. Dieses Erreignis ist dann wichtig, wenn die Anwendung „private Malereien“ oder Texte auf dem Display ausgeben will.

property ***onRefresh*** : TNotifyEvent;

Ein Event/Erreignis, das auftritt, wenn das Display sich neu zeichnet. Sonst wie unter onZoomSet.

Methoden/Prozeduren:

function ***LoadFromFile***(FName : string) : boolean;

Lädt eine zuvor abgespeicherte oder durch ein anderes Programm generierte Konserve aus einer Datei. FName muss u.U. den kompletten Pfad und den Dateinamen nach DOS-Konventionen enthalten. Bei Dateien, die durch andere Programme generiert werden, muss die Dateistruktur folgendermassen beschaffen sein bzw. generiert werden:

Datei-Header

```
str:= 'Oscillo';
str:= str + '|' + FloatToStr(TimeBase);
str:= str + '|' + FloatToStr(Chan1Gain);
str:= str + '|' + FloatToStr(Chan2Gain);
str:= str + '|' + FloatToStr(Chan3Gain);
str:= str + '|' + FloatToStr(Chan4Gain);
str:= str + '|' + IntToStr(FBufferLen);
str:= str + '@' + ^Z;
BlockWrite(DestFile, Str, ord(Str[0])+1);
```

Daten-Block:

```
tc:= sizeOf(single);
for idx:= 0 to BufferLen-1 do begin
     GetBuffer(idx, ch1, ch2, ch3, ch4);
     BlockWrite(DestFile, ch1, tc);
     BlockWrite(DestFile, ch2, tc);
     BlockWrite(DestFile, ch3, tc);
     BlockWrite(DestFile, ch4, tc);
end;
```

function ***SaveToFile***(FName : string) : boolean;
Schreibt den kompletten Buffer mit den zugehörigen Record Parametern in die Datei „FName“. Dateistruktur siehe „LoadFromFile“.

procedure ***DispRefresh***;
Erzwingt ein Neuzeichnen des Displays.

function ***GetDispRect*** : Trect;
Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche. Siehe Beispiel bei „property canvas“.

function ***GetBuffer***(adrs : Integer; var val1, val2, val3, val4 : single) : boolean;
Mit der function kann der Buffer ausgelesen werden. GetBuffer wird true, wenn „adrs“ eine gültigen Wert hat.

procedure ***SetBuffer***(val1, val2, val3, val4 : single);
Mit der procedure werden die 4 Werte in den Buffer geschrieben, und zwar an die Stelle auf die der Startpointer „BufferFill“ zeigt. Der Pointer wird automatisch weitergeführt. Da gleichzeitig ein Zeichnungsvorgang auf das Display erfolgt, ist diese Operation nur im „Record“ Modus zulässig. (StartRec = true)

function ***ChangeBuffer***(adrs : Integer; val1, val2, val3, val4 : single) : boolean;
Mit der function werden 4 Werte an die Stelle „adrs“ geschrieben. Der Pointer wird nicht verändert, solange in einen schon beschriebenen Teil des Buffers geschrieben wird. Ein Schreiben auf einen noch nicht beschriebenen Teil des Buffers ist möglich, setzt aber dann den Schreib-Pointer „BufferFill“ auf die neue Adresse. Ein gleichzeitiges Zeichnen der neuen Werte erfolgt nicht.

procedure ***ScalePnt***(XL,YL : double; var XP,YP : LongInt);
Skalierungs Routine. Dient vor allem dem „privaten“ Zeichnen auf dem Display. Beispiel bei „property canvas“. Um die absoluten Koordinaten für MoveTo, LineTo, TextOut etc. zu erhalten, wird ScalePnt in XL, YL die logische Koordinate im gewählten Koordinatensystem (Time, NormYsize) übergeben. In XP und YP erhält man die physikalische Koordinate zurück.

procedure ***ChanAdd***(ChanA, ChanB : Integer);
Die Pozedur ChanAdd ist eine Kombination der Properties ChanAddA und ChanAddB und wird genau so angewandt wie diese. Beispiel in „Oscar.dpr“.

Variable

DisplayYs : double; Y-Display Höhe absolut, niemals verändern!!
minStepX : double; kann zum horizontal Scrollen herangezogen werden. Nicht ändern!!
DisplayTime : double; kann zum horizontal Scrollen herangezogen werden. Nicht ändern!!

ZoomStartX : double;
ZoomEndX : double;
ZoomStartY : double;
ZoomEndY : double;

Mit diesen 4 Variablen kann ein Zoomausschnit aus dem Buffer gewählt werden. Dabei ist unbedingt zu beachten, dass sowohl die y-Grenzen (-NormYsize und +NormYsizeY) als auch die t-Grenzen (>= 0 und < BufferLen x (TimeBase / SamplesXdiv)) nicht überschritten werden. Mit ZoomFull **=** true/false kann der gewählte Ausschnitt sichtbar gemacht bzw. rückgesetzt werden.

xyGraph

Komponente zur graphischen Darstellung von bis zu 4 Messreihen mit jeweils bis zu 8000 xy-Koordinaten in einem frei wählbaren Koordinatensystem. Die x und y-Achsen sind unabhängig voneinander Linear oder Logarithmisch einstellbar. Alle Parameter über Properties zugänglich. Eingebaute Datei-Operationen (Laden, Speichern, BitMap). Mehrere Printer/Hardcopy Funktionen. Zoom und Messfunktionen.

Komponenten Deklaration

```
type
  xyCoo     = record
                xc : single;
                yc : single;
              end;
  TBufferA  = array[0..32000] of xyCoo;

type
  TxyGraph = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnZoomSet  : TNotifyEvent;
    FOnRefresh  : TNotifyEvent;

    CopyBmp     : TBitMap;

    FColCapt    : TColor;
    FColTxtCapt : TColor;
    FCaption    : String32;
    FUnitNameX  : String32;
    FUnitNameY  : String32;
    FUnitColX   : TColor;
    FUnitColY   : TColor;
    FUnitAutoX  : boolean;
    FUnitAutoY  : boolean;
    FNonLinearX : Boolean;
    FLinStartX  : double;
    FLinEndX    : double;
    FLogStartX  : double;
    FLogEndX    : double;
    FNonLinearY : Boolean;
    FLinStartY  : double;
    FLinEndY    : double;
    FLogStartY  : double;
    FLogEndY    : double;
    FMeasEna    : boolean;
    FPositionShow : Boolean;
    FPointerPosX : double;
    FPointerPosY : double;
    FColPointer : TColor;
    FColDisplay : TColor;
    FColGrid    : TColor;
    FColZeroLine: TColor;
    FScalaPartX : Integer;
    FScalaPartY : Integer;
    fScalaDigsY : Integer;
    FMarkerPosX : double;
    FMarkerPosY : double;
```

```
    FColMarker  : TColor;
    FZoomEna    : Boolean;
    FZoomFull   : Boolean;
    FZoomLight  : Boolean;
    FJoined     : Boolean;
    FChan1Name  : string;
    FChan2Name  : string;
    FChan3Name  : string;
    FChan4Name  : string;
    FChan1Col   : TColor;
    FChan2Col   : TColor;
    FChan3Col   : TColor;
    FChan4Col   : TColor;
    FChan1Ena   : Boolean;
    FChan2Ena   : Boolean;
    FChan3Ena   : Boolean;
    FChan4Ena   : Boolean;
    FChan1Vis   : Boolean;
    FChan2Vis   : Boolean;
    FChan3Vis   : Boolean;
    FChan4Vis   : Boolean;
    FBufferLen  : integer;
    FChan1Fill  : integer;
    FChan2Fill  : integer;
    FChan3Fill  : integer;
    FChan4Fill  : integer;
    Ch1Buff     : ^TBufferA;
    Ch2Buff     : ^TBufferA;
    Ch3Buff     : ^TBufferA;
    Ch4Buff     : ^TBufferA;

    painted     : Boolean;
    XStartP     : LongInt;
    YStartP     : LongInt;
    XAbsP       : LongInt;
    YAbsP       : LongInt;
    XStartL     : double;
    XAbsL       : double;
    XEndL       : double;
    YStartL     : double;
    YAbsL       : double;
    YEndL       : double;
    Log10StartY : double;
    Log10EndY   : double;
    Log10StartX : double;
    Log10EndX   : double;
    GridOffs    : integer;
    TextYpos    : integer;
    axRect,
    ayRect,
    dxRect,
    dyRect,
    CaptRect,
    leftRect,
    topRect,
    dispRect,
    bottomRect  : TRect;
    zooming     : boolean;
    MouseX,
    MouseY      : LongInt;
    PointerPx   : integer;
    PointerPy   : integer;
    MarkerPx    : integer;
```

```
    MarkerPy    : integer;
    ZoomPosX1   : LongInt;
    ZoomPosX2   : LongInt;
    ZoomPosY1   : LongInt;
    ZoomPosY2   : LongInt;
    oldZoom     : boolean;

    PROCEDURE CalcMeasPoints(X, Y: Integer; Shift: TShiftState);
    Procedure SetPositionShow(pValue : Boolean);
    Procedure SetUnitColX(pValue : TColor);
    Procedure SetUnitColY(pValue : TColor);
    Procedure SetUnitNameX(pValue : string32);
    Procedure SetUnitNameY(pValue : string32);
    procedure DrawSignal1(pos : integer);
    procedure DrawSignal2(pos : integer);
    procedure DrawSignal3(pos : integer);
    procedure DrawSignal4(pos : integer);
    Procedure SetNonLinearX(pValue : Boolean);
    Procedure SetNonLinearY(pValue : Boolean);
    Procedure SetLinStartX(pValue : double);
    Procedure SetLinEndX(pValue : double);
    Procedure SetLinStartY(pValue : double);
    Procedure SetLinEndY(pValue : double);
    Procedure SetLogStartX(pValue : double);
    Procedure SetLogEndX(pValue : double);
    Procedure SetLogStartY(pValue : double);
    Procedure SetLogEndY(pValue : double);
    procedure SetJoined(pValue: Boolean);
    procedure SetZoomFull(pValue: Boolean);
    procedure SetZoomLight(pValue: Boolean);
    procedure SetChan1Name(pValue: string);
    procedure SetChan2Name(pValue: string);
    procedure SetChan3Name(pValue: string);
    procedure SetChan4Name(pValue: string);
    procedure SetChan1Ena(pValue: Boolean);
    procedure SetChan2Ena(pValue: Boolean);
    procedure SetChan3Ena(pValue: Boolean);
    procedure SetChan4Ena(pValue: Boolean);
    procedure SetChan1Vis(pValue: Boolean);
    procedure SetChan2Vis(pValue: Boolean);
    procedure SetChan3Vis(pValue: Boolean);
    procedure SetChan4Vis(pValue: Boolean);
    procedure SetChan1Res(pValue: Boolean);
    procedure SetChan2Res(pValue: Boolean);
    procedure SetChan3Res(pValue: Boolean);
    procedure SetChan4Res(pValue: Boolean);
    function  GetChan1Res : boolean;
    function  GetChan2Res : boolean;
    function  GetChan3Res : boolean;
    function  GetChan4Res : boolean;
    Procedure SetCaption(pValue : String32);
    Procedure SetChan1Col(pValue : TColor);
    Procedure SetChan2Col(pValue : TColor);
    Procedure SetChan3Col(pValue : TColor);
    Procedure SetChan4Col(pValue : TColor);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColTxtCapt(pValue : TColor);
    procedure DrawTextCapt;
    procedure SetBufferLen(pValue : integer);
    procedure SetChan1Fill(pValue : integer);
    procedure SetChan2Fill(pValue : integer);
    procedure SetChan3Fill(pValue : integer);
    procedure SetChan4Fill(pValue : integer);
```

```
    procedure SetMeasEna(pValue: Boolean);
    procedure SetPointerPosX(pValue: double);
    procedure SetMarkerPosX(pValue: double);
    procedure SetPointerPosY(pValue: double);
    procedure SetMarkerPosY(pValue: double);
    Procedure SetScalaPartX(pValue : integer);
    Procedure SetScalaDigsY(pValue : integer);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColMarker(pValue : TColor);
    Procedure SetColDisplay(pValue : TColor);
    Procedure SetColGrid(pValue : TColor);
    Procedure SetColZeroLine(pValue : TColor);
    Procedure SetScalaPartY(pValue : integer);
    function  FHardCopy : TBitmap;
    procedure FClearBmp(b : TBitmap);
    procedure DrawPointer;
    procedure DrawMarker;
    procedure ShowPos;
    procedure ClearCh1Buff;
    procedure ClearCh2Buff;
    procedure ClearCh3Buff;
    procedure ClearCh4Buff;
    procedure NewCh1Buff;
    procedure NewCh2Buff;
    procedure NewCh3Buff;
    procedure NewCh4Buff;
  protected
    { Protected-Deklarationen }
    PROCEDURE ClipMouse(clip : boolean);
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    ZoomStartX  : double;
    ZoomEndX    : double;
    ZoomStartY  : double;
    ZoomEndY    : double;

    function  LoadFromFile(FileName : string) : boolean;
    function  SaveToFile(FileName : string) : boolean;
    procedure ParamChange;
    procedure DispRefresh;
    procedure Repaint;
    procedure DrawScala;
    function  GetDispRect : TRect;
    function  CheckCoos(coo : xyCoo) : boolean;
    procedure ScalePnt(XL,YL : double; var XP,YP : LongInt);
    procedure PointToScale(px, py : integer; var sx, sy : double);
    function  GetCh1Buff(adrs : integer; var valx, valy : single) : boolean;
    function  GetCh2Buff(adrs : integer; var valx, valy : single) : boolean;
    function  GetCh3Buff(adrs : integer; var valx, valy : single) : boolean;
    function  GetCh4Buff(adrs : integer; var valx, valy : single) : boolean;
    function  ChangeCh1Buff(adrs : integer; valx, valy : single) : boolean;
    function  ChangeCh2Buff(adrs : integer; valx, valy : single) : boolean;
    function  ChangeCh3Buff(adrs : integer; valx, valy : single) : boolean;
    function  ChangeCh4Buff(adrs : integer; valx, valy : single) : boolean;
    function  SetCh1Buff(valx, valy : single) : boolean;
    function  SetCh2Buff(valx, valy : single) : boolean;
    function  SetCh3Buff(valx, valy : single) : boolean;
```

```
    function  SetCh4Buff(valx, valy : single) : boolean;
    procedure ResetChan1;
    procedure ResetChan2;
    procedure ResetChan3;
    procedure ResetChan4;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property canvas;
    property Align;
    procedure StyleChanged(Sender : TObject);
    Property HardCopy   : TBitMap   read FHardCopy    write FClearBmp;
    Property Caption    : String32  read FCaption     write SetCaption;
    Property UnitColX   : TColor    Read FUnitColX    Write SetUnitColX;
    Property UnitColY   : TColor    Read FUnitColY    Write SetUnitColY;
    Property UnitNameX  : String32  read FUnitNameX   write SetUnitNameX;
    Property UnitNameY  : String32  read FUnitNameY   write SetUnitNameY;
    property UnitAutoX  : boolean   read FUnitAutoX   write FUnitAutoX;
    property UnitAutoY  : boolean   read FUnitAutoY   write FUnitAutoY;
    property Chan1Ena   : boolean   read FChan1Ena    write SetChan1Ena;
    property Chan2Ena   : boolean   read FChan2Ena    write SetChan2Ena;
    property Chan3Ena   : boolean   read FChan3Ena    write SetChan3Ena;
    property Chan4Ena   : boolean   read FChan4Ena    write SetChan4Ena;
    property Chan1Res   : boolean   read GetChan1Res  write SetChan1Res;
    property Chan2Res   : boolean   read GetChan2Res  write SetChan2Res;
    property Chan3Res   : boolean   read GetChan3Res  write SetChan3Res;
    property Chan4Res   : boolean   read GetChan4Res  write SetChan4Res;
    property Chan1Name  : string    read FChan1Name   write SetChan1Name;
    property Chan2Name  : string    read FChan2Name   write SetChan2Name;
    property Chan3Name  : string    read FChan3Name   write SetChan3Name;
    property Chan4Name  : string    read FChan4Name   write SetChan4Name;
    property Chan1Col   : TColor    read FChan1Col    write SetChan1Col;
    property Chan2Col   : TColor    read FChan2Col    write SetChan2Col;
    property Chan3Col   : TColor    read FChan3Col    write SetChan3Col;
    property Chan4Col   : TColor    read FChan4Col    write SetChan4Col;
    property Chan1Vis   : boolean   read FChan1Vis    write SetChan1Vis;
    property Chan2Vis   : boolean   read FChan2Vis    write SetChan2Vis;
    property Chan3Vis   : boolean   read FChan3Vis    write SetChan3Vis;
    property Chan4Vis   : boolean   read FChan4Vis    write SetChan4Vis;
    property Chan1Fill  : integer   read FChan1Fill   write SetChan1Fill;
    property Chan2Fill  : integer   read FChan2Fill   write SetChan2Fill;
    property Chan3Fill  : integer   read FChan3Fill   write SetChan3Fill;
    property Chan4Fill  : integer   read FChan4Fill   write SetChan4Fill;
    property CaptBackCol: TColor    read FColCapt     write SetColCapt;
    property CaptionCol : TColor    read FColTxtCapt  write SetColTxtCapt;
    property ColDisplay : TColor    read FColDisplay  write SetColDisplay;
    property ColGrid    : TColor    read FColGrid     write SetColGrid;
    property ColZeroLine: TColor    read FColZeroLine write SetColZeroLine;
    property PointerCol : TColor    read FColPointer  write SetColPointer;
    property MarkerCol  : TColor    read FColMarker   write SetColMarker;
    property ScalaPartX : Integer   read FScalaPartX  write SetScalaPartX;
    Property MeasEnable : boolean   read FMeasEna     write SetMeasEna;
    Property PositionShow : Boolean Read FPositionShow Write SetPositionShow;
    Property PointerPosX : double   read FPointerPosX write SetPointerPosX;
    Property MarkerPosX : double    read FMarkerPosX  write SetMarkerPosX;
    Property PointerPosY : double   read FPointerPosY write SetPointerPosY;
    Property MarkerPosY : double    read FMarkerPosY  write SetMarkerPosY;
    property ScalaPartY : Integer   read FScalaPartY  write SetScalaPartY;
    property ScalaDigsY : Integer   read FScalaDigsY  write SetScalaDigsY;
    property BufferLen  : integer   read FBufferLen   write SetBufferLen;
    property ZoomEnable : boolean   read FZoomEna     write FZoomEna;
    property ZoomFull   : boolean   read FZoomFull    write SetZoomFull;
```

```
    property ZoomLight  : boolean   read FZoomLight   write SetZoomLight;
    property Joined     : boolean   read FJoined      write SetJoined;
    Property NonLinearX : Boolean   read FNonLinearX  write SetNonLinearX;
    Property NonLinearY : Boolean   read FNonLinearY  write SetNonLinearY;
    property LinStartX  : double    read FLinStartX   write SetLinStartX;
    property LinStartY  : double    read FLinStartY   write SetLinStartY;
    property LinEndX    : double    read FLinEndX     write SetLinEndX;
    property LinEndY    : double    read FLinEndY     write SetLinEndY;
    property LogStartX  : double    read FLogStartX   write SetLogStartX;
    property LogStartY  : double    read FLogStartY   write SetLogStartY;
    property LogEndX    : double    read FLogEndX     write SetLogEndX;
    property LogEndY    : double    read FLogEndY     write SetLogEndY;
    property onZoomSet : TNotifyEvent read FOnZoomSet write FOnZoomSet;
    property onRefresh : TNotifyEvent read FOnRefresh write FOnRefresh;
  end;


function TxyGraph.SaveToFile(FileName : string) : boolean;
 var idx, tc    : integer;
     DestFile   : file of byte;
     NumWritten : integer;
     st         : string;
     xyVal      : xyCoo;
 begin
   AssignFile(DestFile, FileName);
   Rewrite(DestFile);
   st:= 'xyGraph';
   st:= st + '|' + IntToStr(Chan1Fill);
   st:= st + '|' + IntToStr(Chan2Fill);
   st:= st + '|' + IntToStr(Chan3Fill);
   st:= st + '|' + IntToStr(Chan4Fill);
   st:= st + '@' + ^Z;
   NumWritten:= length(St);
   BlockWrite(DestFile, NumWritten, 1);
   BlockWrite(DestFile, St[1], NumWritten);
   tc:= sizeOf(xyCoo);
   for idx:= 0 to Chan1Fill-1 do begin
     GetCh1Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
   end;
   for idx:= 0 to Chan2Fill-1 do begin
     GetCh2Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
   end;
   for idx:= 0 to Chan3Fill-1 do begin
     GetCh3Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
   end;
   for idx:= 0 to Chan4Fill-1 do begin
     GetCh4Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
   end;
   System.CloseFile(DestFile);
   Result:= true;
 end;
```

Eigenschaften/Properties

property ***Align***;
Delphi kompatibel: automatische Anpassung etc.

property ***canvas*** : TCanvas;
Zeichenfläche der Komponente. Der Entwickler kann mit den vorhandenen Properties von „canvas“ eigene Zeichenoperationen durchführen und Texte plazieren. Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche.

Beispiel:
```
var r : TRect;
begin
  r:= xyGraph1.GetDispRect;
  with r do begin
    xyGraph1.Canvas.MoveTo(left, top);
    xyGraph1.Canvas.LineTo(right, bottom);
    xyGraph1.Canvas.TextOut(left, top + (bottom-top) div 2, 'Text');
  end;
end;
```

property ***HardCopy*** : TBitMap;
HardCopy liefert den Inhalt des Displays als BitMap. Hiermit lassen sich diverse Datei- oder Print Operationen durchführen. Das folgende Beispiel speichert die Komponente als BitMap File auf die Disk. Weitere Beispiele sind im Projekt „xyDemo.dpr“ zu finden.

Beispiel:
```
xyGraph1.HardCopy.SaveToFile('FileName.bmp');
xyGraph1.HardCopy:= nil;                          (* gibt Resource frei *)
```

property ***Caption*** : String32;
Caption ist die Überschrift der Komponente und erscheint im obersten Textfenster.

property ***CaptBackCol***: Tcolor;
Hintergrund Farbe der Überschrift

property ***CaptionCol*** : Tcolor;
Farbe der Überschrift

property ***UnitColX*** : Tcolor;
Farbe des Textes EinheitX, z.B. „Volt“

property ***UnitColY*** : Tcolor;
Farbe des Textes EinheitY, z.B. „Amp“

property ***UnitAutoX*** : Boolean;
Sperrt bzw. gibt die automatische Umformung der EinheitX, z.B. „Amp“ frei. Bei false wird der Text im Property UnitNameX direkt angezeigt. Bei true erfolgt ggf. eine Umformung, dabei wird z.B. aus „Amp“ -> „mAmp“ oder „kAmp“. Die Beschriftung der X-Scala wird jedoch nicht beeinflusst.

property ***UnitAutoY*** : Boolean;
Sperrt bzw. gibt die automatische Umformung der EinheitY, z.B. „Volt“ frei. Bei false wird der Text im Property UnitNameY direkt angezeigt. Bei true erfolgt ggf. eine Umformung, dabei wird z.B. aus „Volt“ -> „mVolt“ oder „kVolt“. Die Beschriftung der Y-Scala wird jedoch nicht beeinflusst.

property ***ScalaDigsY*** : Integer; (0..9)
Gibt die max. darstellbaren Ziffern der Y-Scala an. Der Wert 0 schaltet die Beschränkung aus.

Überschreitet der aktuelle Skalenwert diesen Vorgabewert, so wird in Exponentendarstellung umgeschaltet.

property ***UnitNameX*** : String32;
Text/Name EinheitX, z.B. „Volt“. Wird durch „UnitAutoX“ manipuliert.

property ***UnitNameY*** : String32;
Text/Name EinheitY, z.B. „Amp“. Wird durch „UnitAutoY“ manipuliert.

property ***Chan1Ena*** : boolean;
property ***Chan2Ena*** : boolean;
property ***Chan3Ena*** : boolean;
property ***Chan4Ena*** : boolean;
Beim Wechsel von false nach true wird der zugehörige Buffer aquiriert und initialisiert. Das Beschreiben mit Daten wird zugelassen. Beim Wechsel von true nach false wird der requirierte Speicher des Buffers freigegeben und ein Lesen und Schreiben ist nicht mehr möglich.

property ***Chan1Res*** : boolean;
property ***Chan2Res*** : boolean;
property ***Chan3Res*** : boolean;
property ***Chan4Res*** : boolean;
Löscht beim Schreiben den zugehörigen Buffer mit 0, dabei spielt der Wert des boolean (true/false) keine Rolle. Der Lesezugriff ergibt ChanxRes:= ChanxFill = 0;

property ***Chan1Vis*** : boolean;
property ***Chan2Vis*** : boolean;
property ***Chan3Vis*** : boolean;
property ***Chan4Vis*** : boolean;
Schaltet die Kurve des entsprechenden Kanals sichtbar bzw. unsichtbar.

property ***Chan1Name*** : string;
property ***Chan2Name*** : string;
property ***Chan3Name*** : string;
property ***Chan4Name*** : string;
Text/Name für jeweiligen Kanal, wird im zugehörigen Feld in der eingestellten Kanalfarbe angezeigt.

property ***Chan1Col*** : Tcolor;
property ***Chan2Col*** : Tcolor;
property ***Chan3Col*** : Tcolor;
property ***Chan4Col*** : Tcolor;
Farbe des Text/Namen Kanals sowie der Kurve des zugehörigen Kanals.

property ***Chan1Fill*** : Integer;
property ***Chan2Fill*** : Integer;
property ***Chan3Fill*** : Integer;
property ***Chan4Fill*** : Integer;
Gibt die Anzahl der eingeschriebenen xy-Werte zurück und dient gleichzeitig als Schreibpointer für SetChanxBuff. Nur Lesen möglich.

property ***ColDisplay*** : Tcolor;
Hintergrund Farbe des Displays

property ***ColGrid*** : Tcolor;
Farbe des Rasters des Displays

property ***ColZeroLine***: Tcolor;
Farbe der Nullinie des Displays

property ***PointerCol*** : Tcolor;
Farbe der Messlinien des Displays

property ***MarkerCol*** : Tcolor;
Farbe der Bezugslinien für die Messlinien des Displays

property ***ScalaPartX*** : Integer;
Anzahl der vertikalen Orientierungslinien auf der x-Achse. (1..20)
bei NonLinear X : Anzahl der Linien in einer Dekade
bei Linear X : Anzahl aller Rasterlinien des Displays und Beschriftung der X-Achse

property ***ScalaPartY*** : Integer;
Anzahl der horizontalen Orientierungslinien auf der y-Achse. (1..20)
bei NonLinear Y : Anzahl der Linien in einer Dekade
bei Linear Y : Anzahl aller Rasterlinien des Displays und Beschriftung der Y-Achse

property ***MeasEnable*** : boolean;
Hiermit werden die Mess Fadenkreuze sichtbar bzw. unsichtbar geschaltet.
Die beiden Mess-Fadenkreuze können mit der linken bzw. rechten Maustaste bewegt werden.

property ***PointerPosX*** : double;
Das Pointer Fadenkreuz ist das eigentliche Messfadenkreuz. Die X-Koordinate dieses Kreuzes wird relativ zur x-Koordinate des Markerkreuzes und absolut in dem entsprechenden Feld angezeigt. Mit PointerPosX kann die absolute x-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***MarkerPosX*** : double;
Das Marker Fadenkreuz dient zur relativen Messung der x und y Werte im Zusammenhang mit dem Pointer Fadenkreuz. Mit MarkerPosX kann die absolute x-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***PointerPosY*** : double;
Das Pointer Fadenkreuz ist das eigentliche Messfadenkreuz. Die Y-Koordinate dieses Kreuzes wird relativ zur y-Koordinate des Markerkreuzes und absolut in dem entsprechenden Feld angezeigt. Mit PointerPosY kann die absolute y-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***MarkerPosY*** : double;
Das Marker Fadenkreuz dient zur relativen Messung der x und y Werte im Zusammenhang mit dem Pointer Fadenkreuz. Mit MarkerPosY kann die absolute y-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***BufferLen*** : Integer;
Grösse des Aufzeichnungs Buffers bzw. Anzahl der möglichen Aufzeichnungs Records.
Ein Verändern dieses Parameters löscht die Buffer komplett und initialisiert sie neu.
Der tatsächliche Speicherverbrauch eines Buffers kann aus der Initialisierung errechnet werden:

```
GetMem(Ch1Buff, BufferLen * SizeOf(xyCoo));
```

wobei xyCoo so definiert ist:

```
type
   xyCoo     = record
                 xc : single;
                 yc : single;
               end;
```

Mit der function
```
GetCh1Buff(adrs : integer; var valx, valy : single) : boolean;
```
kann der Buffer 1 ausgelesen werden. GetChxBuff wird true, wenn „adrs“ gültig ist.

Mit der function
```
TxyGraph.SetCh2Buff(valx, valy : single) : boolean;
```
wird ein Koordinatenpaar in Buffer 2 geschrieben, und zwar an die Stelle auf die der Startpointer (ChanxFill) zeigt. Der Pointer wird automatisch inkrementiert. Da gleichzeitig ein Zeichnungsvorgang auf das Display erfolgt, ist diese Operation, vor allem beim „Joined“ Modus, wenn der letzte Punkt mit dem neuen mit einer Linie verbunden wird, nur sinnvoll, wenn die erfassten Daten auch sozusagen inkrementell sind. Random geschrieben werden kann nur, wenn joined immer false ist.

Mit der function
```
ChangeCh1Buff(adrs : Integer; valx, valy : single) : boolean;
```
wird ein Koordinatenpaar an die Stelle „adrs“ geschrieben. Der Schreib-Pointer wird nicht verändert, solange die Operation einen schon beschriebenen Teil des Buffers betrifft. Ein Schreiben auf einen noch nicht beschriebenen Teil der Buffer ist möglich, hat aber zur Folge dass der Schreib-Pointer (ChanxFill) dann auf diese Adresse gesetzt wird.

property ***ZoomEnable*** : boolean;
Sperrt die Zoomfunktion der mittleren Maustaste (oder Shift + Linke Maustaste) oder gibt diese frei.

property ***ZoomLight*** : boolean;
Im nicht-gezoomten Modus wird die zuletzt eingestellte Zoom-Area mittels eines Rahmens dargestellt. Ist wichtig zum Grössenvergleich, Auffinden etc.

property ***ZoomFull*** : boolean;
Schaltet um zwischen Normaldarstellung, gezoomter Darstellung. Beispiel in „xyDemo.dpr.“

property ***Joined*** : boolean;
Bestimmt die Zeichnungsart der Kurven auf dem Display. Wenn Joined = true dann werden die einzelnen Messpunkte mittels einer Linie miteinander verbunden. Im anderen Fall werden die Messpunkte nur als Pixel/Punkt gezeichnet. Joined ist relativ langsam, hat aber den Vorteil der besseren Darstellung.

property ***NonLinearX*** : Boolean;
Stellt die Lineare oder Logarithmische Darstellung des Displays für die x-Achse ein.
Siehe auch ScalaPartX.

property ***NonLinearY*** : Boolean;
Stellt die Lineare oder Logarithmische Darstellung des Displays für die y-Achse ein.
Siehe auch ScalaPartY.

property ***LinStartX*** : double;
Definiert die „linkste“ Linie bzw. x-Anfangswert der Darstellung des Displays. Muss kleiner bzw. negativer sein als LinEndX. Wird benutzt wenn NonLinearX = false.

property ***LinEndX*** : double;
Definiert die „rechste“ Linie bzw. x-Endwert der Darstellung des Displays. Muss grösser bzw. positiver sein als LinStartX. Wird benutzt wenn NonLinearX = false.

property ***LogStartX*** : double;
Definiert die „linkste“ Linie bzw. x-Anfangswert der Darstellung des Displays. Muss kleiner bzw. negativer sein als LogEndX aber immer > 0. Wird benutzt wenn NonLinearX = true.

property ***LogEndX*** : double;

Definiert die „oberste“ Linie bzw. x-Endwert der Darstellung des Displays. Muss grösser bzw. positiver sein als LogStartY. Wird benutzt wenn NonLinearY = true.

property ***LinStartY*** : double;
Definiert die „unterste“ Linie bzw. y-Anfangswert der Darstellung des Displays. Muss kleiner bzw. negativer sein als LinEndY. Wird benutzt wenn NonLinearY = false.

property ***LinEndY*** : double;
Definiert die „oberste“ Linie bzw. y-Endwert der Darstellung des Displays. Muss grösser bzw. positiver sein als LinStartY. Wird benutzt wenn NonLinearY = false.

property ***LogStartY*** : double;
Definiert die „unterste“ Linie bzw. y-Anfangswert der Darstellung des Displays. Muss kleiner bzw. negativer sein als LogEndY aber immer > 0. Wird benutzt wenn NonLinearY = true.

property ***LogEndY*** : double;
Definiert die „oberste“ Linie bzw. Endwert der y-Darstellung des Displays. Muss grösser bzw. positiver sein als LogStartY. Wird benutzt wenn NonLinearY = true.

property ***onZoomSet*** : TNotifyEvent;
Ein Event/Ereignis, das auftritt, wenn der ZoomMode sich ändert. Dieses Erreignis ist dann wichtig, wenn die Anwendung „private Malereien“ oder Texte auf dem Display ausgeben will.

property ***onRefresh*** : TNotifyEvent;
Ein Event/Erreignis, das auftritt, wenn das Display sich neu zeichnet. Sonst wie unter onZoomSet.

Methoden/Prozeduren

function ***LoadFromFile***(FName : string) : boolean;
Lädt eine zuvor abgespeicherte oder durch ein anderes Programm generierte Konserve aus einer Datei. „FName“ muss u.U. den kompletten Pfad und den Dateinamen nach DOS-Konventionen enthalten. Bei Dateien, die durch andere Programme generiert werden, muss die Dateistruktur folgendermassen beschaffen sein bzw. generiert werden:

Datei-Header

```
str:= 'xyGraph';
str:= str + '|' + IntToStr(Chan1Fill);
str:= str + '|' + IntToStr(Chan2Fill);
str:= str + '|' + IntToStr(Chan3Fill);
str:= str + '|' + IntToStr(Chan4Fill);
str:= str + '@' + ^Z;
BlockWrite(DestFile, Str, ord(Str[0])+1);
```

Daten-Block:

```
tc:= sizeOf(xyCoo);
for idx:= 0 to Chan1Fill-1 do begin
     GetCh1Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
end;
for idx:= 0 to Chan2Fill-1 do begin
     GetCh2Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
end;
for idx:= 0 to Chan3Fill-1 do begin
     GetCh3Buff(idx, xyVal.xc, xyVal.yc);
     BlockWrite(DestFile, xyVal, tc);
end;
```

```
          for idx:= 0 to Chan4Fill-1 do begin
               GetCh4Buff(idx, xyVal.xc, xyVal.yc);
               BlockWrite(DestFile, xyVal, tc);
          end;

     type
        xyCoo     = record
                      xc : single;
                      yc : single;
                    end;
```

function ***SaveToFile***(FName : string) : boolean;
Schreibt die 4 Buffer mit den zugehörigen Record Parametern in die Datei „Fname“. Dateistruktur siehe „LoadFromFile“.

procedure ***ParamChange***;
Nach verändern von Aufzeichnungs- oder Bufferparametern **muss** ParamChange aufgerufen werden. Dadurch ist es möglich einen komplett neuen Parametersatz zu installieren, ohne dass bei jeder einzelnen Parameteränderung die Komponente sich neu zeichnet.

procedure ***DispRefresh***;
Erzwingt ein Neuzeichnen des Displays.

procedure ***Repaint***;
Erzwingt ein Neuzeichnen der gesamten Komponente.

procedure ***DrawScala***;
Erzwingt ein Neuzeichnen der Scala.

procedure ***ScalePnt***(XL,YL : double; var XP,YP : LongInt);
Skalierungs Routine. Dient vor allem dem „privaten“ Zeichnen auf dem Display. Beispiel bei „property canvas“. Um die absoluten Koordinaten für MoveTo, LineTo, TextOut etc. zu erhalten, wird ScalePnt in XL, YL die logische Koordinate im gewählten Koordinatensystem (LinStartX etc. etc) übergeben. In XP und YP erhält man die physikalische Koordinate zurück.

function ***GetDispRect*** : Trect;
Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche. Siehe Beispiel bei „property canvas“.

function ***GetCh1Buff***(adrs : integer; var valx, valy : single) : boolean;
function ***GetCh2Buff***(adrs : integer; var valx, valy : single) : boolean;
function ***GetCh3Buff***(adrs : integer; var valx, valy : single) : boolean;
function ***GetCh4Buff***(adrs : integer; var valx, valy : single) : boolean;
Mit der function GetChxBuff kann der Buffer ausgelesen werden. GetChxBuff wird true, wenn „adrs“ eine gültigen Wert hat. (Siehe property BufferLen)

function ***SetCh1Buff***(valx, valy : single) : boolean;
function ***SetCh2Buff***(valx, valy : single) : boolean;
function ***SetCh3Buff***(valx, valy : single) : boolean;
function ***SetCh4Buff***(valx, valy : single) : boolean;
Mit der procedure SetChxBuff wird ein Koordinatenpaar in den Buffer x geschrieben, und zwar an die Stelle auf die der pointer ChanxFill zeigt. Der Pointer wird automatisch inkrementiert. Da gleichzeitig ein Zeichnungsvorgang auf das Display erfolgt, ist diese Operation, vor allem beim „Joined“ Modus, wenn der letzte Punkt mit dem neuen mit einer Linie verbunden wird, nur sinnvoll, wenn die erfassten Daten auch sozusagen inkrementell sind. Random geschrieben werden kann nur, wenn joined immer false ist.
(Siehe property BufferLen)

function ***ChangeCh1Buff***(adrs : integer; valx, valy : single) : boolean;
function ***ChangeCh2Buff***(adrs : integer; valx, valy : single) : boolean;
function ***ChangeCh3Buff***(adrs : integer; valx, valy : single) : boolean;
function ***ChangeCh4Buff***(adrs : integer; valx, valy : single) : boolean;

Mit der function ChangeChxBuff wird ein Koordinatenpaar an die Stelle „adrs“ geschrieben. Der Schreib-Pointer wird nicht verändert, solange die Operation einen schon beschriebenen Teil des Buffers betrifft. Ein Schreiben auf einen noch nicht beschriebenen Teil der Buffer ist möglich, hat aber zur Folge dass der Schreib-Pointer (ChanxFill) dann auf diese Adresse gesetzt wird. (Siehe property BufferLen)

procedure ***ResetChan1***;
procedure ***ResetChan2***;
procedure ***ResetChan3***;
procedure ***ResetChan4***;

Die Prozedur RestChanx löscht den enstsprechenden Buffer mit „0“ und setzt den Schreibpointer auf die Adresse „0“. Anschliessend wird das Display neu gezeichnet (Refresh).

Variable

ZoomStartX : double;
ZoomEndX : double;
ZoomStartY : double;
ZoomEndY : double;

Mit diesen 4 Variablen kann ein Zoomausschnit aus dem Buffer gewählt werden. Dabei ist unbedingt zu beachten, dass sowohl die y-Grenzen(LinStartY und LinEndY bzw. LogStartY und LogEndY) als auch die x-Grenzen nicht überschritten werden. Mit ZoomFull **=** true/falsekann der gewählte Ausschnitt sichtbar gemacht bzw. rückgesetzt werden.

xyGraphN

Komponente zur graphischen Darstellung von bis zu 16 Messreihen mit jeweils bis zu 8000 xy-Koordinaten. Jeder Kanal hat sein eigenes frei wählbares Koordinatensystem. Die x und y-Achsen sind unabhängig voneinander Linear oder Logarithmisch einstellbar. Alle Parameter über Properties zugänglich. Eingebaute Datei-Operationen (Laden, Speichern, BitMap). Mehrere Printer/Hardcopy Funktionen. Zoom und Messfunktionen.

Komponenten Deklaration

```
type
  xyCoo     = record
                xc : single;
                yc : single;
              end;
  TBufferA  = array[0..32000] of xyCoo;

{ Forward declarations }
  TxyGraphN = class;

  TxyChan = class(TPersistent)
  private
    FControl    : TXYgraphN;
    cChanName   : string;
    cChanFill   : integer;
    cBufferLen  : integer;
    ChanBuff    : ^TBufferA;
    cChanCol    : TColor;
    cZeroLineCol: TColor;
    cZeroLineStyl : TPenStyle;
    cChanEna    : Boolean;
    cChanVis    : Boolean;
    cUnitNameX  : String32;
    cUnitNameY  : String32;
    cUnitAutoX  : boolean;
    cUnitAutoY  : boolean;
    cUnitColX   : TColor;
    cUnitColY   : TColor;
    cNonLinearX : Boolean;
    cLinStartX  : double;
    cLinEndX    : double;
    cLogStartX  : double;
    cLogEndX    : double;
    cNonLinearY : Boolean;
    cLinStartY  : double;
    cLinEndY    : double;
    cLogStartY  : double;
    cLogEndY    : double;
    cScalaDigsY : Integer;
    cScalaPartX : Integer;
    cScalaPartY : Integer;
    cScalaPLogX : Integer;
    cScalaPLogY : Integer;
    cLog10StartY: double;
    cLog10EndY  : double;
    cLog10StartX: double;
    cLog10EndX  : double;
    cXStartL    : double;
```

```
    cXAbsL      : double;
    cXEndL      : double;
    cYStartL    : double;
    cYAbsL      : double;
    cYEndL      : double;
    ChanNum     : integer;

    procedure ParamChange(doInval : boolean);
    procedure DispRefresh;
    procedure ClearChanBuff;
    procedure NewChanBuff;
    procedure SetChanEna(pValue: Boolean);
    procedure SetChanVis(pValue: Boolean);
    procedure SetChanRes(pValue: Boolean);
    function  GetChanRes : boolean;
    Procedure SetChanCol(pValue : TColor);
    Procedure SetZeroLineCol(pValue : TColor);
    procedure SetZeroLineStyl(pValue : tPenStyle);
    procedure SetChanFill(pValue : integer);
    Procedure SetScalaDigsY(pValue : integer);
    Procedure SetScalaPartX(pValue : integer);
    Procedure SetScalaPartY(pValue : integer);
    Procedure SetScalaPLogX(pValue : integer);
    Procedure SetScalaPLogY(pValue : integer);
    procedure SetChanName(pValue: string);
    Procedure SetNonLinearX(pValue : Boolean);
    Procedure SetNonLinearY(pValue : Boolean);
    Procedure SetLinStartX(pValue : double);
    Procedure SetLinEndX(pValue : double);
    Procedure SetLinStartY(pValue : double);
    Procedure SetLinEndY(pValue : double);
    Procedure SetLogStartX(pValue : double);
    Procedure SetLogEndX(pValue : double);
    Procedure SetLogStartY(pValue : double);
    Procedure SetLogEndY(pValue : double);
    Procedure SetUnitColX(pValue : TColor);
    Procedure SetUnitColY(pValue : TColor);
    Procedure SetUnitNameX(pValue : string32);
    Procedure SetUnitNameY(pValue : string32);
    Procedure UpdateParam;
    constructor Create(AControl: TXYgraphN);
  public
    procedure ResetChan;
    procedure Assign(Source: TPersistent); override;
  published
    property ZeroLineStyl: TPenStyle read cZeroLineStyl write SetZeroLineStyl;
    property ZeroLineCol: TColor    read cZeroLineCol write SetZeroLineCol;
    Property UnitColX   : TColor    Read cUnitColX    Write SetUnitColX;
    Property UnitColY   : TColor    Read cUnitColY    Write SetUnitColY;
    Property UnitNameX  : String32  read cUnitNameX   write SetUnitNameX;
    Property UnitNameY  : String32  read cUnitNameY   write SetUnitNameY;
    property UnitAutoX  : boolean   read cUnitAutoX   write cUnitAutoX;
    property UnitAutoY  : boolean   read cUnitAutoY   write cUnitAutoY;
    property ChanRes    : boolean   read GetChanRes   write SetChanRes;
    property ChanName   : string    read cChanName    write SetChanName;
    property ChanCol    : TColor    read cChanCol     write SetChanCol;
    property ChanVis    : boolean   read cChanVis     write SetChanVis;
    property ChanFill   : integer   read cChanFill    write SetChanFill;
    property ScalaDigsY : Integer   read cScalaDigsY  write SetScalaDigsY;
    property ScalaPartX : Integer   read cScalaPartX  write SetScalaPartX;
    property ScalaPartY : Integer   read cScalaPartY  write SetScalaPartY;
    property ScalaPartNX: Integer   read cScalaPLogX  write SetScalaPLogX;
    property ScalaPartNY: Integer   read cScalaPLogY  write SetScalaPLogY;
```

```
    Property NonLinearX : Boolean   read cNonLinearX  write SetNonLinearX;
    Property NonLinearY : Boolean   read cNonLinearY  write SetNonLinearY;
    property LinStartX  : double    read cLinStartX   write SetLinStartX;
    property LinStartY  : double    read cLinStartY   write SetLinStartY;
    property LinEndX    : double    read cLinEndX     write SetLinEndX;
    property LinEndY    : double    read cLinEndY     write SetLinEndY;
    property LogStartX  : double    read cLogStartX   write SetLogStartX;
    property LogStartY  : double    read cLogStartY   write SetLogStartY;
    property LogEndX    : double    read cLogEndX     write SetLogEndX;
    property LogEndY    : double    read cLogEndY     write SetLogEndY;
  end;


  TxyGraphN = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnZoomSet  : TNotifyEvent;
    FOnRefresh  : TNotifyEvent;
    FOnChChange : TNotifyEvent;

    FColCapt    : TColor;
    FColTxtCapt : TColor;
    FCaption    : String32;
    FMeasEna    : boolean;
    FPosShow    : Boolean;
    FPointerPosX : double;
    FPointerPosY : double;
    FColPointer : TColor;
    FColDisplay : TColor;
    FColGrid    : TColor;
    FZeroLineCol: TColor;
    fZeroLineStyl : TPenStyle;
    FMarkerPosX : double;
    FMarkerPosY : double;
    FColMarker  : TColor;
    FZoomEna    : Boolean;
    FZoomFull   : Boolean;
    FZoomLight  : Boolean;
    FJoined     : Boolean;
    FBufferLen  : integer;
    FSpaceRight : integer;
    FSpaceLeft  : integer;
    FChanCount  : integer;

    fScalaDigsY : Integer;
    fScalaPartX : Integer;
    fScalaPartY : Integer;
    fScalaPLogX : Integer;
    fScalaPLogY : Integer;
    FUnitNameX  : String32;
    FUnitNameY  : String32;
    FUnitAutoX  : boolean;
    FUnitAutoY  : boolean;
    FUnitColX   : TColor;
    FUnitColY   : TColor;
    FNonLinearX : Boolean;
    FLinStartX  : double;
    FLinEndX    : double;
    FLogStartX  : double;
    FLogEndX    : double;
    FNonLinearY : Boolean;
    FLinStartY  : double;
    FLinEndY    : double;
```

```
    FLogStartY  : double;
    FLogEndY    : double;
    fXStartL    : double;
    fXAbsL      : double;
    fXEndL      : double;
    fYStartL    : double;
    fYAbsL      : double;
    fYEndL      : double;
    fLog10StartY: double;
    fLog10EndY  : double;
    fLog10StartX: double;
    fLog10EndX  : double;
    FComGrid    : boolean;

    CopyBmp     : TBitMap;
    axBmp       : TBitMap;
    ayBmp       : TBitMap;
    dxBmp       : TBitMap;
    dyBmp       : TBitMap;
    axBmpsav    : TBitMap;
    ayBmpsav    : TBitMap;
    dxBmpsav    : TBitMap;
    dyBmpsav    : TBitMap;
    ChanArr     : Array[1..16] of TxyChan;
    CurChan     : integer;
    ChanKeyDown : integer;
    painted     : Boolean;
    XStartP     : LongInt;
    YStartP     : LongInt;
    XAbsP       : LongInt;
    YAbsP       : LongInt;
    GridOffs    : integer;
    TextYpos    : integer;
    ComBtn      : TRect;
    KeyArr      : array[1..16] of TRect;
    axRect,
    ayRect,
    dxRect,
    dyRect,
    CaptRect,
    leftRect,
    topRect,
    dispRect,
    bottomRect  : TRect;
    zooming     : boolean;
    MouseX,
    MouseY      : LongInt;
    PointerPx   : integer;
    PointerPy   : integer;
    MarkerPx    : integer;
    MarkerPy    : integer;
    ZoomPosX1   : LongInt;
    ZoomPosX2   : LongInt;
    ZoomPosY1   : LongInt;
    ZoomPosY2   : LongInt;
    oldZoom     : boolean;

    procedure DrawSignal(chan, pos : integer);
    procedure DrawComBtn;
    PROCEDURE CalcMeasPoints(X, Y: Integer; Shift: TShiftState);
    procedure SetComGrid(pValue: boolean);
    Procedure SetPosShow(pValue : Boolean);
    procedure SetChan1(pValue: TxyChan);
```

```
    procedure SetChan2(pValue: TxyChan);
    procedure SetChan3(pValue: TxyChan);
    procedure SetChan4(pValue: TxyChan);
    procedure SetChan5(pValue: TxyChan);
    procedure SetChan6(pValue: TxyChan);
    procedure SetChan7(pValue: TxyChan);
    procedure SetChan8(pValue: TxyChan);
    procedure SetChan9(pValue: TxyChan);
    procedure SetChan10(pValue: TxyChan);
    procedure SetChan11(pValue: TxyChan);
    procedure SetChan12(pValue: TxyChan);
    procedure SetChan13(pValue: TxyChan);
    procedure SetChan14(pValue: TxyChan);
    procedure SetChan15(pValue: TxyChan);
    procedure SetChan16(pValue: TxyChan);
    function  GetChan1 : TxyChan;
    function  GetChan2 : TxyChan;
    function  GetChan3 : TxyChan;
    function  GetChan4 : TxyChan;
    function  GetChan5 : TxyChan;
    function  GetChan6 : TxyChan;
    function  GetChan7 : TxyChan;
    function  GetChan8 : TxyChan;
    function  GetChan9 : TxyChan;
    function  GetChan10 : TxyChan;
    function  GetChan11 : TxyChan;
    function  GetChan12 : TxyChan;
    function  GetChan13 : TxyChan;
    function  GetChan14 : TxyChan;
    function  GetChan15 : TxyChan;
    function  GetChan16 : TxyChan;
    procedure SetChanCount(pValue : integer);
    procedure SetJoined(pValue: Boolean);
    procedure SetZoomFull(pValue: Boolean);
    procedure SetZoomLight(pValue: Boolean);
    Procedure SetCaption(pValue : String32);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColTxtCapt(pValue : TColor);
    procedure DrawTextCapt;
    procedure SetSpaceRight(pValue : integer);
    procedure SetSpaceLeft(pValue : integer);
    procedure SetBufferLen(pValue : integer);
    procedure SetMeasEna(pValue: Boolean);
    procedure SetPointerPosX(pValue: double);
    procedure SetMarkerPosX(pValue: double);
    procedure SetPointerPosY(pValue: double);
    procedure SetMarkerPosY(pValue: double);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColMarker(pValue : TColor);
    Procedure SetColDisplay(pValue : TColor);
    Procedure SetColGrid(pValue : TColor);
    procedure SetCurChan(pValue : integer);
    function  FHardCopy : TBitmap;
    procedure FClearBmp(b : TBitmap);
    procedure DrawPointer;
    procedure DrawMarker;
    procedure ShowPos;
  protected
    { Protected-Deklarationen }
    PROCEDURE ClipMouse(clip : boolean);
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
```

```
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    ZoomStartX  : double;
    ZoomEndX    : double;
    ZoomStartY  : double;
    ZoomEndY    : double;

    function  LoadFromFile(FileName : string) : boolean;
    function  SaveToFile(FileName : string) : boolean;
    procedure ParamChange;
    procedure DispRefresh;
    procedure Repaint;
    procedure DrawScala;
    function  GetDispRect : TRect;
    function  CheckCoos(chan : integer; coo : xyCoo) : boolean;
    procedure ScalePnt(chan : integer; XL,YL : double; var XP,YP : LongInt);
    procedure PointToScale(px, py : integer; var sx, sy : double);
    function  GetChanBuff(chan, adrs : integer; var valx, valy : single) :
boolean;
    function  ChangeChanBuff(chan, adrs : integer; valx, valy : single) :
boolean;
    function  SetChanBuff(chan : integer; valx, valy : single) : boolean;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property canvas;
    property Align;
    procedure StyleChanged(Sender : TObject);
    property SpaceRight : integer   read FSpaceRight  write SetSpaceRight;
    property SpaceLeft  : integer   read FSpaceLeft   write SetSpaceLeft;
    property ChanShow   : integer   read CurChan      write SetCurChan;
    Property HardCopy   : TBitMap   read FHardCopy    write FClearBmp;
    Property Caption    : String32  read FCaption     write SetCaption;
    property CaptBackCol: TColor    read FColCapt     write SetColCapt;
    property CaptionCol : TColor    read FColTxtCapt  write SetColTxtCapt;
    property ColDisplay : TColor    read FColDisplay  write SetColDisplay;
    property ColGrid    : TColor    read FColGrid     write SetColGrid;
    property PointerCol : TColor    read FColPointer  write SetColPointer;
    property MarkerCol  : TColor    read FColMarker   write SetColMarker;
    Property MeasEnable : boolean   read FMeasEna     write SetMeasEna;
    Property PositionShow : Boolean Read FPosShow     Write SetPosShow;
    Property PointerPosX : double   read FPointerPosX write SetPointerPosX;
    Property MarkerPosX : double    read FMarkerPosX  write SetMarkerPosX;
    Property PointerPosY : double   read FPointerPosY write SetPointerPosY;
    Property MarkerPosY : double    read FMarkerPosY  write SetMarkerPosY;
    property BufferLen  : integer   read FBufferLen   write SetBufferLen;
    property ZoomEnable : boolean   read FZoomEna     write FZoomEna;
    property ZoomFull   : boolean   read FZoomFull    write SetZoomFull;
    property ZoomLight  : boolean   read FZoomLight   write SetZoomLight;
    property Joined     : boolean   read FJoined      write SetJoined;
    property CommonGrid : boolean   read FComGrid     write SetComGrid;
    property ChanCount  : integer   read FChanCount   write SetChanCount;
    property Channel1   : TxyChan   read GetChan1     write SetChan1;
    property Channel2   : TxyChan   read GetChan2     write SetChan2;
    property Channel3   : TxyChan   read GetChan3     write SetChan3;
    property Channel4   : TxyChan   read GetChan4     write SetChan4;
    property Channel5   : TxyChan   read GetChan5     write SetChan5;
    property Channel6   : TxyChan   read GetChan6     write SetChan6;
    property Channel7   : TxyChan   read GetChan7     write SetChan7;
```

```
    property Channel8   : TxyChan   read GetChan8     write SetChan8;
    property Channel9   : TxyChan   read GetChan9     write SetChan9;
    property Channel10  : TxyChan   read GetChan10    write SetChan10;
    property Channel11  : TxyChan   read GetChan11    write SetChan11;
    property Channel12  : TxyChan   read GetChan12    write SetChan12;
    property Channel13  : TxyChan   read GetChan13    write SetChan13;
    property Channel14  : TxyChan   read GetChan14    write SetChan14;
    property Channel15  : TxyChan   read GetChan15    write SetChan15;
    property Channel16  : TxyChan   read GetChan16    write SetChan16;
    property onZoomSet : TNotifyEvent read FOnZoomSet write FOnZoomSet;
    property onRefresh : TNotifyEvent read FOnRefresh write FOnRefresh;
    property onChanChange : TNotifyEvent read FOnChChange write FOnChChange;
  end;


function TxyGraphN.SaveToFile(FileName : string) : boolean;
 var idx, tc, i : integer;
     DestFile   : file of byte;
     NumWritten : integer;
     st         : string;
     xyVal      : xyCoo;
 begin
   AssignFile(DestFile, FileName);
   Rewrite(DestFile);
   st:= 'xyGraphN';
   for idx:= 1 to 16 do
     st:= st + '|' + IntToStr(ChanArr[idx].ChanFill);
   st:= st + '@' + ^Z;
   NumWritten:= length(St);
   BlockWrite(DestFile, NumWritten, 1);
   BlockWrite(DestFile, St[1], NumWritten);
   tc:= sizeOf(xyCoo);
   for i:= 1 to 16 do begin
     for idx:= 0 to ChanArr[i].ChanFill-1 do begin
       GetChanBuff(i, idx, xyVal.xc, xyVal.yc);
       BlockWrite(DestFile, xyVal, tc);
     end;
   end;
   System.CloseFile(DestFile);
   Result:= true;
 end;
```

ytGraph

Fortlaufende graphische Darstellung ankommender Daten/Messwerte. Bis zu einer gewissen Geschwindigkeit echtzeitfähig. Auch abspielen von gespeicherten Daten. Zoom, Print und Messfunktionen. 4 Kanäle. Y-Achse lin/log. Eingebaute Datei-Operationen (Laden, Speichern, BitMap).

Die Komponente TRecCtrl ist speziell für die Steuerung des ytGraph entwickelt worden. Im Demo **Programm ytDemo.dpr** ist das Zusammenspiel gut zu erkennen.

Bitte beachten:
Die Komponente zeichnet aus Geschwindigkeitsgründen direkt auf den Bildschirm ohne Zwischen Pufferung. Das bedeutet, dass während eines Aufzeichnungvorgangs die Komponente nicht durch andere Windows Komponenten überdeckt werden sollte. Dies hätte ein Neuzeichnen zur Folge, was u.U. mit Datenverlust bezahlt werden muss. Der Kompromiss wurde gewählt, um die Zeichnungsgeschwindigkeit während der Aufzeichnung hoch zu halten. Ein zerstören der Komponente z.B. durch einen fremden Dialog wird während der Aufzeichnung nicht repariert, was auf die Aufzeichnung selbst jedoch keinen Einfluss hat. Die Daten werden sicher gehalten, nur das gelöschte Teil des Displays bleibt weiss.

Komponenten Deklaration

```
type
  TytGraph = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnZoomSet   : TNotifyEvent;
    FOnRefresh   : TNotifyEvent;
    FOnParmChange : TNotifyEvent;

    CopyBmp      : TBitMap;
    yBmp         : TBitMap;

    FColCapt     : TColor;
    FColTxtCapt  : TColor;
    FCaption     : String32;
    FUnitNameY   : String32;
    FUnitColX    : TColor;
    FUnitColY    : TColor;
    FDisplayTyp  : ytDispType;
    FGraphMode   : ytGraphMode;
    FTimeBase    : double;
    FSamples     : LongInt;
    FNonLinearY  : Boolean;
    FLinStartY   : double;
    FLinEndY     : double;
    FLogStartY   : double;
    FLogEndY     : double;
    FDateTime    : boolean;
    FMeasEna     : boolean;
    FMeasShow    : Boolean;
    FPointerPosX : double;
```

```
    FPointerPosY : double;
    FColPointer : TColor;
    FColDisplay : TColor;
    FColGrid    : TColor;
    FColZeroLine: TColor;
    FScalaPartX : Integer;
    FScalaPartY : Integer;
    FMarkerPosX : double;
    FMarkerPosY : double;
    FColMarker  : TColor;
    FStartRec   : Boolean;
    FStartPlay  : Boolean;
    FZoomEna    : Boolean;
    FZoomLight  : Boolean;
    FZoomMode   : ytZoomMode;
    FJoined     : Boolean;
    FChan1Col   : TColor;
    FChan2Col   : TColor;
    FChan3Col   : TColor;
    FChan4Col   : TColor;
    FChan1Name  : string;
    FChan2Name  : string;
    FChan3Name  : string;
    FChan4Name  : string;
    FChan1Vis   : Boolean;
    FChan2Vis   : Boolean;
    FChan3Vis   : Boolean;
    FChan4Vis   : Boolean;
    FBufferLen  : LongInt;
    FBuffFill   : LongInt;
    StartPtr    : PytRec;
    CurPtr      : PytRec;
    StartTime   : double;
    EndTime     : double;
    DispTime    : double;
    curTime     : double;
    RecDate     : TDateTime;
    IntPol      : double;
    curFrac     : double;
    minTime     : double;
    stepping    : boolean;
    zooming     : boolean;
    MouseX,
    MouseY      : LongInt;
    PointerPx   : integer;
    PointerPy   : integer;
    MarkerPx    : integer;
    MarkerPy    : integer;
    ZoomPosX1   : LongInt;
    ZoomPosX2   : LongInt;
    ZoomPosY1   : LongInt;
    ZoomPosY2   : LongInt;
    InterLeave  : LongInt;
    oldZoom     : ytZoomMode;
    savMeasEna  : boolean;
    RecEndTime  : double;
    RecStartTime: double;

    painted     : Boolean;
    XStartP     : LongInt;
    YStartP     : LongInt;
    XAbsP       : LongInt;
    YAbsP       : LongInt;
```

```
    XEndL       : double;
    YStartL     : double;
    YAbsL       : double;
    YEndL       : double;
    Log10StartY : double;
    Log10EndY   : double;
    oldNormStart: double;
    gridStep    : double;
    GridOffs    : integer;
    TextYpos    : integer;
    ScalaYpos   : Integer;
    axRect,
    ayRect,
    dxRect,
    dyRect,
    SizeRect,
    CaptRect,
    leftRect,
    topRect,
    dispRect,
    sourceRect,
    destRect,
    ScalaXrect,
    TimeRectA,
    DateRect,
    TimeRectE,
    bottomRect  : TRect;

    PROCEDURE CalcMeasPoints(X, Y: Integer; Shift: TShiftState);
    Procedure SetDisplayTyp(pValue : ytDispType);
    Procedure SetGraphMode(pValue : ytGraphMode);
    Procedure SetMeasShow(pValue : Boolean);
    Procedure SetDateTime(pValue : Boolean);
    Procedure SetUnitColX(pValue : TColor);
    Procedure SetUnitColY(pValue : TColor);
    Procedure SetUnitNameY(pValue : string32);
    procedure DrawSignals;
    procedure SetStartRec(pValue: Boolean);
    procedure SetStartPlay(pValue: Boolean);
    Procedure SetNonLinearY(pValue : Boolean);
    Procedure SetLinStartY(pValue : double);
    Procedure SetLinEndY(pValue : double);
    Procedure SetLogStartY(pValue : double);
    Procedure SetLogEndY(pValue : double);
    procedure SetJoined(pValue: Boolean);
    procedure SetZoomMode(pValue: ytZoomMode);
    procedure SetZoomLight(pValue: Boolean);
    procedure SetChan1Vis(pValue: Boolean);
    procedure SetChan2Vis(pValue: Boolean);
    procedure SetChan3Vis(pValue: Boolean);
    procedure SetChan4Vis(pValue: Boolean);
    Procedure SetCaption(pValue : String32);
    Procedure SetChan1Col(pValue : TColor);
    Procedure SetChan2Col(pValue : TColor);
    Procedure SetChan3Col(pValue : TColor);
    Procedure SetChan4Col(pValue : TColor);
    Procedure SetChan1Name(pValue : string);
    Procedure SetChan2Name(pValue : string);
    Procedure SetChan3Name(pValue : string);
    Procedure SetChan4Name(pValue : string);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColTxtCapt(pValue : TColor);
    procedure DrawTextCapt;
```

```
    function  ScaleX(XL : double) : LongInt;
    function  ScaleY(YL : double)  : LongInt;
    procedure PointToScale(px, py : integer; var sx, sy : double);
    procedure SetBufferLen(pValue : LongInt);
    procedure SetBuffFill(pValue : LongInt);
    procedure SetMeasEna(pValue: Boolean);
    procedure SetSamples(pValue: LongInt);
    procedure SetTimeBase(pValue: double);
    procedure SetPointerPosX(pValue: double);
    procedure SetMarkerPosX(pValue: double);
    procedure SetPointerPosY(pValue: double);
    procedure SetMarkerPosY(pValue: double);
    Procedure SetScalaPartX(pValue : integer);
    Procedure SetColPointer(pValue : TColor);
    Procedure SetColMarker(pValue : TColor);
    Procedure SetColDisplay(pValue : TColor);
    Procedure SetColGrid(pValue : TColor);
    Procedure SetColZeroLine(pValue : TColor);
    Procedure SetScalaPartY(pValue : integer);
    function  FHardCopy : TBitmap;
    procedure FClearBmp(b : TBitmap);
    procedure DrawPointer;
    procedure DrawMarker;
    procedure ShowPos;
    Procedure DrawSize;
    procedure ScrollForw;
    procedure ScrollBack;
    procedure FreeBuffer;
  protected
    { Protected-Deklarationen }
    PROCEDURE ClipMouse(clip : boolean);
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    ZoomStartX  : double;
    ZoomEndX    : double;
    ZoomStartY  : double;
    ZoomEndY    : double;

    function  LoadFromFile(FName : string) : boolean;
    function  SaveToFile(FName : string) : boolean;
    procedure StartForw(pValue : boolean);
    function  StepForw : boolean;
    procedure StartBack(pValue : boolean);
    function  StepBack : boolean;
    procedure PageBack;
    procedure PageForw;
    procedure SetDispStart(pValue : double);
    procedure ParamChange;
    procedure DispRefresh;
    procedure Repaint;
    procedure DrawScala;
    Procedure ShowTime;
    function  GetDispRect : TRect;
    function  GetBuffer(adrs : LongInt; var val1, val2, val3, val4 : single) :
boolean;
    procedure SetBuffer(val1, val2, val3, val4 : single);
```

```
    function  ChangeBuffer(adrs : LongInt; val1, val2, val3, val4 : single) :
boolean;
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    property canvas;
    property DrawMode;
    procedure StyleChanged(Sender : TObject);
    Property HardCopy   : TBitMap   read FHardCopy    write FClearBmp;
    Property Caption    : String32  read FCaption     write SetCaption;
    property DisplayTyp : ytDispType read FDisplayTyp write SetDisplayTyp;
    Property GraphMode : ytGraphMode Read FGraphMode  Write SetGraphMode;
    Property UnitColX   : TColor    Read FUnitColX    Write SetUnitColX;
    Property UnitColY   : TColor    Read FUnitColY    Write SetUnitColY;
    Property UnitNameY  : String32  read FUnitNameY   write SetUnitNameY;
    property Chan1Name  : string    read FChan1Name   write SetChan1Name;
    property Chan2Name  : string    read FChan2Name   write SetChan2Name;
    property Chan3Name  : string    read FChan3Name   write SetChan3Name;
    property Chan4Name  : string    read FChan4Name   write SetChan4Name;
    property Chan1Col   : TColor    read FChan1Col    write SetChan1Col;
    property Chan2Col   : TColor    read FChan2Col    write SetChan2Col;
    property Chan3Col   : TColor    read FChan3Col    write SetChan3Col;
    property Chan4Col   : TColor    read FChan4Col    write SetChan4Col;
    property Chan1Vis   : boolean   read FChan1Vis    write SetChan1Vis;
    property Chan2Vis   : boolean   read FChan2Vis    write SetChan2Vis;
    property Chan3Vis   : boolean   read FChan3Vis    write SetChan3Vis;
    property Chan4Vis   : boolean   read FChan4Vis    write SetChan4Vis;
    property BufferFill : LongInt   read FBuffFill    write SetBuffFill;
    property CaptBackCol: TColor    read FColCapt     write SetColCapt;
    property CaptionCol : TColor    read FColTxtCapt  write SetColTxtCapt;
    property ColDisplay : TColor    read FColDisplay  write SetColDisplay;
    property ColGrid    : TColor    read FColGrid     write SetColGrid;
    property ColZeroLine: TColor    read FColZeroLine write SetColZeroLine;
    property PointerCol : TColor    read FColPointer  write SetColPointer;
    property MarkerCol  : TColor    read FColMarker   write SetColMarker;
    property ScalaPartX : Integer   read FScalaPartX  write SetScalaPartX;
    property DispSamples: LongInt   read FSamples     write SetSamples;
    property TimeBase   : double    read FTimeBase    write SetTimeBase;
    Property DateTime   : boolean   read FDateTime    write SetDateTime;
    Property MeasEnable : boolean   read FMeasEna     write SetMeasEna;
    Property MeasShow   : Boolean   Read FMeasShow    Write SetMeasShow;
    Property PointerPosX : double   read FPointerPosX write SetPointerPosX;
    Property MarkerPosX : double    read FMarkerPosX  write SetMarkerPosX;
    Property PointerPosY : double   read FPointerPosY write SetPointerPosY;
    Property MarkerPosY : double    read FMarkerPosY  write SetMarkerPosY;
    property ScalaPartY : Integer   read FScalaPartY  write SetScalaPartY;
    property BufferLen  : LongInt   read FBufferLen   write SetBufferLen;
    property StartRec   : boolean   read FStartRec    write SetStartRec;
    property StartPlay  : boolean   read FStartPlay   write SetStartPlay;
    property ZoomEnable : boolean   read FZoomEna     write FZoomEna;
    property ZoomLight  : boolean   read FZoomLight   write SetZoomLight;
    property ZoomMode   : ytZoomMode read FZoomMode   write SetZoomMode;
    property Joined     : boolean   read FJoined      write SetJoined;
    Property NonLinearY : Boolean   read FNonLinearY  write SetNonLinearY;
    property LinStartY  : double    read FLinStartY   write SetLinStartY;
    property LinEndY    : double    read FLinEndY     write SetLinEndY;
    property LogStartY  : double    read FLogStartY   write SetLogStartY;
    property LogEndY    : double    read FLogEndY     write SetLogEndY;
    property onZoomSet  : TNotifyEvent   read FOnZoomSet    write FOnZoomSet;
    property onRefresh  : TNotifyEvent   read FOnRefresh    write FOnRefresh;
    property OnParmChange : TNotifyEvent read FOnParmChange write FOnParmChange;
```

```
  end;


function TytGraph.SaveToFile(FName : string) : boolean;
 var idx, tc    : integer;
     DestFile   : file;
     NumWritten : integer;
     str        : string;
     Val1, Val2,
     Val3, Val4 : single;
     GetRec     : PytRec;
 begin
   if BufferFill = 0 then exit;
   Result:= true;
   AssignFile(DestFile, FName);
   Rewrite(DestFile, 1);
   str:= 'ytGraph';
   str:= str + '|' + DateToStr(RecDate);
   str:= str + '|' + IntToStr(BufferFill);
   if NonLinearY then str:= str + '|N'
   else str:= str + '|L';
   str:= str + '|' + FloatToStr(FLogStartY);
   str:= str + '|' + FloatToStr(FLogEndY);
   str:= str + '|' + FloatToStr(FLinStartY);
   str:= str + '|' + FloatToStr(FLinEndY);
   str:= str + '|' + FloatToStr(TimeBase);
   str:= str + '|' + FloatToStr(RecStartTime);
   str:= str + '|' + FloatToStr(RecEndTime);
   str:= str + '@' + ^Z;
   NumWritten:= length(Str);
   BlockWrite(DestFile, NumWritten, 1);
   BlockWrite(DestFile, Str[1], NumWritten);
   tc:= sizeOf(single);
   GetRec:= StartPtr;
   for idx:= 0 to BufferFill-1 do begin
     val1:= GetRec^.v1;
     val2:= GetRec^.v2;
     val3:= GetRec^.v3;
     val4:= GetRec^.v4;
     BlockWrite(DestFile, val1, tc);
     BlockWrite(DestFile, val2, tc);
     BlockWrite(DestFile, val3, tc);
     BlockWrite(DestFile, val4, tc);
     GetRec:= GetRec^.next;
   end;
   System.CloseFile(DestFile);
   Result:= true;
 end;
```

Eigenschaften/Properties

property ***Align***;
Delphi kompatibel: automatische Anpassung etc.

property ***canvas*** : TCanvas;
Zeichenfläche der Komponente. Der Entwickler kann mit den vorhandenen Properties von „canvas“ eigene Zeichenoperationen durchführen und Texte plazieren. Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche.

Beispiel:

```
var r : TRect;
begin
  r:= ytGraph1.GetDispRect;
  with r do begin
     ytGraph1.Canvas.MoveTo(left, top);
     ytGraph1.Canvas.LineTo(right, bottom);
     ytGraph1.Canvas.TextOut(left, top + (bottom-top) div 2, 'Text');
  end;
end;
```

property ***HardCopy*** : TBitMap;
HardCopy liefert die Komponente als BitMap. Hiermit lassen sich diverse Datei- oder Print Operationen durchführen. Das folgende Beispiel speichert die Komponente als BitMap File auf die Disk. Weitere Beispiele sind im Projekt „ytDemo.dpr“ im Formular „ytPrint“ zu finden.

Beispiel:

```
ytGraph1.HardCopy.SaveToFile('FileName.bmp');
ytGraph1.HardCopy:= nil;                          (* gibt Resource frei *)
```

property ***Caption*** : String32;
Caption ist die Überschrift der Komponente und erscheint im obersten Textfenster.

property ***CaptBackCol***: Tcolor;
Hintergrund Farbe der Überschrift

property ***CaptionCol*** : Tcolor;
Farbe der Überschrift

property ***DisplayTyp*** : ytDispType;
DisplayTyp stellt die Erscheinungsform von ytGraph ein. Es gibt 4 Möglichkeiten:

complete : Alle Teile von ytGraph werden dargestellt.
fullSize : Nur das Display selbst wird dargestellt.
topScreen : Das Display und „Caption“ wird dargestellt.
bottomScreen Das Display und die Kanalnamen werden dargestellt.

property ***GraphMode*** : ytGraphMode;
bestimmt die Darstellung während der Aufzeichnung bzw. Wiedergabe. Es existieren 3 Modi:

gmNone : Es wird nur das erste Bild/Seite gezeichnet, dann keine weiteren mehr
gmPage : Seitenweises Schieben von rechts nach links, ist z.Zeit noch nicht implementiert.
gmScroll : Die erste Seite wird komplett gezeichnet, dann erfolgt ein kontinuierliches Durchschieben des Bildes von rechts nach links.

property ***UnitColX*** : Tcolor;
Farbe des Textes EinheitX, z.B. „Time“

property ***UnitColY*** : Tcolor;
Farbe des Textes EinheitY, z.B. „Amp“

property ***UnitNameY*** : String32;
Text/Name EinheitY, z.B. „Amp“

property ***Chan1Name*** : string;
Text/Name für Kanal 1, wird im unteren linken Feld bei DisplayTyp „complete“ oder „bottomScreen“ angezeigt.

property ***Chan2Name*** : string;
Text/Name für Kanal 2, wird im zweiten Feld von links bei DisplayTyp „complete“ oder „bottomScreen“ angezeigt.

property ***Chan3Name*** : string;
Text/Name für Kanal 3, wird im zweiten Feld von rechts bei DisplayTyp „complete“ oder „bottomScreen“ angezeigt.

property ***Chan4Name*** : string;
Text/Name für Kanal 4, wird im unteren rechten Feld bei DisplayTyp „complete“ oder „bottomScreen“ angezeigt.

property ***Chan1Col*** : Tcolor;
Farbe des Text/Namen Kanal 1 sowie der Kurve von Kanal 1

property ***Chan2Col*** : Tcolor;
Farbe des Text/Namen Kanal 2 sowie der Kurve von Kanal 2

property ***Chan3Col*** : Tcolor;
Farbe des Text/Namen Kanal 3 sowie der Kurve von Kanal 3

property ***Chan4Col*** : Tcolor;
Farbe des Text/Namen Kanal 4 sowie der Kurve von Kanal 4

property ***Chan1Vis*** : boolean;
Schaltet die Darstellung der Kurve Kanal 1 ein bzw. aus

property ***Chan2Vis*** : boolean;
Schaltet die Darstellung der Kurve Kanal 2 ein bzw. aus

property ***Chan3Vis*** : boolean;
Schaltet die Darstellung der Kurve Kanal 3 ein bzw. aus

property ***Chan4Vis*** : boolean;
Schaltet die Darstellung der Kurve Kanal 4 ein bzw. aus

property ***BufferFill*** : LongInt;
Gibt die Anzahl der eingeschriebenen yt-Werte zurück, kann auf 0 gesetzt werden.

property ***ColDisplay*** : Tcolor;
Hintergrund Farbe des Displays

property ***ColGrid*** : Tcolor;
Farbe des Rasters des Displays

property ***ColZeroLine***: Tcolor;
Farbe der Nullinie des Displays

property ***PointerCol*** : Tcolor;
Farbe der Messlinien des Displays

property ***MarkerCol*** : Tcolor;
Farbe der Bezugslinien für die Messlinien des Displays

property ***ScalaPartX*** : Integer;
Anzahl der vertikalen Orientierungslinien auf der t-Achse. (1..20)

property ***DispSamples***: LongInt;
Gibt an wieviel Messpunkte in der horizontalen x- bzw. t-Achse im ZoomMode = zmNormal dargestellt werden sollen. DispSamples x TimeBase = Zeitintervall für ein komplettes Display bzw. DispEndTime - DispStartTime.
Im Demo Programm „ytDemo“ ist DispSamples = 1000 und TimeBase = 10msec. Das Display hat deshalb eine Anzeige Spannweite von 1000 x 10msec -> 10sec.

property ***TimeBase*** : double;
Gibt den zeitlichen Abstand zwischen zwei aufgezeichneten Werten an.
Achtung:
Die Komponente hat keinerlei Timer oder ähnliches. Es liegt deshalb beim Anwendungs Programm, dass im Aufzeichnungsmodus die Daten im entsprechenden Zeitabstand mittels SetBuffer eingeschrieben werden. Das gleiche gilt umgekehrt bei der Wiedergabe mittels StartPlay etc. TimeBase dient nur zur korrekten Darstellung der Daten der x- bzw. t-Achse und bei den Messfunktionen.

property ***DateTime*** : boolean;
Ist DateTime = true, dann wird mit dem aktuellen Datum und Uhrzeit als t-Masstab aufgezeichnet, ansonsten beginnt die Aufzeichnung mit der Zeit 0.
Ein ändern von DateTime ist nur unmittelbar vor RecStart sinnvoll und zulässig.

property ***MeasEnable*** : boolean;
Mit MeasEnable wird die Möglichkeit, mit der linken bzw. rechten Maustaste die Mess-Fadenkreuze zu bewegen freigeschaltet bzw. gesperrt.

property ***MeasShow*** : Boolean;
Hiermit werden die Mess Fadenkreuze sichtbar bzw. unsichtbar geschaltet.

property ***PointerPosX*** : double;
Das Pointer Fadenkreuz ist das eigentliche Messfadenkreuz. Die X-Koordinate (Zeit) dieses Kreuzes wird relativ zur x-Koordinate des Markerkreuzes und absolut zum Aufzeichnungsstart in dem entsprechenden Feld angezeigt. Mit PointerPosX kann die absolute x-Position vom Programm ausgelesen bzw. gesetzt werden. Die Einheit ist Sekunden.

property ***MarkerPosX*** : double;
Das Marker Fadenkreuz dient zur relativen Messung der y und t Werte im Zusammenhang mit dem Pointer Fadenkreuz. Mit MarkerPosX kann die absolute x-Position vom Programm ausgelesen bzw. gesetzt werden. Die Einheit ist Sekunden.

property ***PointerPosY*** : double;
Das Pointer Fadenkreuz ist das eigentliche Messfadenkreuz. Die Y-Koordinate dieses Kreuzes wird relativ zur y-Koordinate des Markerkreuzes und absolut in dem entsprechenden Feld angezeigt. Mit PointerPosY kann die absolute y-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***MarkerPosY*** : double;

Das Marker Fadenkreuz dient zur relativen Messung der y und t Werte im Zusammenhang mit dem Pointer Fadenkreuz. Mit MarkerPosY kann die absolute y-Position vom Programm ausgelesen bzw. gesetzt werden.

property ***ScalaPartY*** : Integer;

Anzahl der horizontalen Orientierungslinien auf der y-Achse. (1..20).
bei NonLinear Y : Anzahl der Linien in einer Dekade
bei Linear Y : Anzahl aller Rasterlinien des Displays und Beschriftung der Y-Achse

property ***BufferLen*** : LongInt;

Grösse des Aufzeichnungs Buffers bzw. Anzahl der möglichen Aufzeichnungs Record. Der Buffer ist als RingBuffer in Form einer Linked List organisiert, d.h. ist beim Aufzeichnen der Buffer komplett gefüllt, wird der nächste Wert an die Stelle des ersten Werts geschrieben und dieser damit überschrieben. Die zugehörigen Start- und Endpointer werden automatisch mitgeführt. Durch die RingBuffer Konstruktion werden immer die letzten [BufferLen] Parameter gehalten. Der tatsächliche Speicherverbrauch des Buffers kann aus der Initialisierung errechnet werden: Maximaler Grösse ist 50000 Samples.

```
for idx:= 2 to BufferLen do begin
   GetMem(StartPtr, SizeOf(TytRec));
end;
```

wobei TytRec so definiert ist:

```
type
  PytRec  = ^TytRec;
  TytRec  = Record
              prev : PytRec;
              v1   : single;
              v2   : single;
              v3   : single;
              v4   : single;
            next : PytRec;
          end;
```

Mit der function

```
GetBuffer(adrs : LongInt; var val1, val2, val3, val4 : single) : boolean;
```

kann der Buffer ausgelesen werden. GetBuffer wird true, wenn „adrs“ eine gültigen Wert hat, d.h. innerhalb des beschriebenen Teils Buffers zeigt.

Mit der procedure

```
SetBuffer(val1, val2, val3, val4 : single);
```

werden die 4 Werte in den Buffer geschrieben, und zwar an die Stelle auf die der Startpointer zeigt. Die Pointer werden automatisch weitergeführt. Da gleichzeitig ein Zeichnungsvorgang auf das Display erfolgt, ist diese Operation nur im „Record“ Modus zulässig.

Mit der function

```
ChangeBuffer(adrs : LongInt; val1, val2, val3, val4 : single) : boolean;
```

werden 4 Werte an die Stelle „adrs“ geschrieben. Die Pointer werden nicht verändert, d.h. ein Schreiben auf einen noch nicht beschriebenen Teil des Buffers ist möglich, hat aber dann keine weiteren Folgen bzw. ist ohne Wirkung.

property ***StartRec*** : boolean;

Wie schon angeführt, ist ein Schreiben von Werten in den Buffer mit zugehöriger Pointer-Fortführung und Anzeige auf dem Display nur sinnvoll im Record Modus. Durch setzen von StartRec auf true wird dieser Modus initialisiert, alle Pointer werden zurückgesetzt, der Buffer gelöscht etc. etc. Nach dem Aufzeichnungsvorgang mittels Aufrufe der Methode SetBuffer muss der Record Modus wieder zurückgesetzt werden durch StartRec:= false. Erst dann können Zoom-, Mess- und Speicherfunction ausgeführt werden.

Beispiel: ytDemo procedure FillBuffer

property ***StartPlay*** : boolean;
StartPlay initialisiert die Komponente zur kontinuierlichen Wiedergabe des Buffers. Die eigentliche Wiedergabe unterliegt jedoch dem Anwendungsprogrammierer. Nach dem beenden der Wiedergabe muss StartPlay auf false gesetzt werden. Während der Wiedergabe sollten Zoom- und Messfunktionen unterbunden werden.

Beispiel für einen Wiedergabe Ablauf aus „ytDemo.dpr“ :

```
ytGraph.StartPlay:= true;
Repeat
    ytGraph.StepForw;
    Application.ProcessMessages;
    (* ... Delay ... *)
until (* ... Abbruchbedingung ... *)
ytGraph.StartPlay:= false;
```

property ***ZoomEnable*** : boolean;
Sperrt die Zoomfunktion der mittleren Maustaste (oder Shift + Linke Maustaste) oder gibt diese frei.

property ***ZoomLight*** : boolean;
Im nicht-gezoomten Modus wird die zuletzt eingestellte Zoom-Area mittels eines Rahmens dargestellt. Ist wichtig zum Grössenvergleich, Auffinden etc.

property ***ZoomMode*** : ytZoomMode;
Schaltet um zwischen Normaldarstellung, gezoomter Darstellung und ganzer Buffer Darstellung. Beispiel in ytDemo.dpr.
zmNormal, zmFull, zmZoomed

property ***Joined*** : boolean;
Bestimmt die Zeichnungsart der Kurven auf dem Display. Wenn Joined = true dann werden die einzelnen Messpunkte mittels einer Linie miteinander verbunden. Im anderen Fall werden die Messpunkte nur als Pixel/Punkt gezeichnet. Joined ist relativ langsam, hat aber den Vorteil der besseren Darstellung.

property ***NonLinearY*** : Boolean;
Stellt die Lineare oder Logarithmische Darstellung des Displays ein.
Siehe auch ScalaPartY.

property ***LinStartY*** : double;
Definiert die „unterste“ Linie bzw. Anfangswert der Darstellung des Displays. Muss kleiner bzw. negativer sein als LinEndY. Wird benutzt wenn NonLinearY = false.

property ***LinEndY*** : double;
Definiert die „oberste“ Linie bzw. Endwert der Darstellung des Displays. Muss grösser bzw. positiver sein als LinStartY. Wird benutzt wenn NonLinearY = false.

property ***LogStartY*** : double;
Definiert die „unterste“ Linie bzw. Anfangswert der Darstellung des Displays. Muss kleiner bzw. negativer sein als LogEndY aber immer > 0. Wird benutzt wenn NonLinearY = true.

property ***LogEndY*** : double;
Definiert die „oberste“ Linie bzw. Endwert der Darstellung des Displays. Muss grösser bzw. positiver sein als LogStartY. Wird benutzt wenn NonLinearY = true.

property ***onZoomSet*** : TNotifyEvent;
Ein Event/Erreignis, das auftritt, wenn der ZoomMode sich ändert. Dieses Erreignis ist dann wichtig, wenn die Anwendung „private Malereien“ oder Texte auf dem Display ausgeben will.

property ***onRefresh*** : TNotifyEvent;
Ein Event/Erreignis, das auftritt, wenn das Display sich neu zeichnet. Sonst wie unter onZoomSet.

property ***OnParmChange*** : TNotifyEvent;
Das Erreignis wird durch verändern von Parameter, z.B. durch den Anwender oder durch Laden einer Konserve von einer Datei, ausgelöst.

Methoden/Prozeduren

function ***LoadFromFile***(FName : string) : boolean;
Lädt eine zuvor abgespeicherte oder durch ein anderes Programm generierte Konserve aus einer Datei. Fname muss u.U. den kompletten Pfad und den Dateinamen nach DOS-Konventionen enthalten. Bei Dateien, die durch andere Programme generiert werden, muss die Dateistruktur folgendermassen beschaffen sein bzw. generiert werden:

Datei-Header

```
str:= 'ytGraph';
str:= str + '|' + DateToStr(RecDate);
str:= str + '|' + IntToStr(BufferFill);
if NonLinearY then str:= str + '|N'
else str:= str + '|L';
str:= str + '|' + FloatToStr(FLogStartY);
str:= str + '|' + FloatToStr(FLogEndY);
str:= str + '|' + FloatToStr(FLinStartY);
str:= str + '|' + FloatToStr(FLinEndY);
str:= str + '|' + FloatToStr(TimeBase);
str:= str + '|' + FloatToStr(RecStartTime);
str:= str + '|' + FloatToStr(RecEndTime);
str:= str + '@' + ^Z;
BlockWrite(DestFile, Str, ord(Str[0])+1);
```

Daten-Block:

```
tc:= sizeOf(single);
for idx:= 0 to BufferFill-1 do begin
   val1:= Rec.v1;
   val2:= Rec.v2;
   val3:= Rec.v3;
   val4:= Rec.v4;
   BlockWrite(DestFile, val1, tc);
   BlockWrite(DestFile, val2, tc);
   BlockWrite(DestFile, val3, tc);
   BlockWrite(DestFile, val4, tc);
   (* nächster Record holen *)
   Rec:= ...;
end;
```

function ***SaveToFile***(FName : string) : boolean;
Schreibt den kompletten Buffer mit den zugehörigen Record Parametern in die Datei „Fname“. Dateistruktur siehe „LoadFromFile“.

procedure ***StartForw***(pValue : boolean);
StartForw bereitet die Komponente auf nachfolgende StepForw Aktionen vor. Nach erfolgten Steps muss StartForw auf false gesetzt werden.

function ***StepForw*** : boolean;
Schiebt das Display um einen Schritt nach links. Die Funktion kann mehrmals hintereinander aufgerufen werden. Ein oder mehrere Steps müssen mit der Ankündigung StartForw eingeleitet werden.

Beispiel:

```
ytGraph.StartForw(true);
Repeat
  Stop:= not ytGraph.StepForw;
  Application.ProcessMessages;
(* hier kann eine Scrolltaste geprüft werden und ggf. Stop gesetzt werden *)
until stop;
ytGraph.StartForw(false);
```

procedure ***StartBack***(pValue : boolean);
StartBack bereitet die Komponente auf nachfolgende StepBack Aktionen vor. Nach erfolgten Steps muss StartBack auf false gesetzt werden.

function ***StepBack*** : boolean;
Schiebt das Display um einen Schritt nach rechts. Die Funktion kann mehrmals hintereinander aufgerufen werden. Ein oder mehrere Steps müssen mit der Ankündigung StartBack eingeleitet werden.

Beispiel:

```
ytGraph.StartBack(true);
Repeat
  Stop:= not ytGraph.StepBack;
  Application.ProcessMessages;
(* hier kann eine Scrolltaste geprüft werden und ggf. Stop gesetzt werden *)
until stop;
ytGraph.StartBack(false);
```

procedure ***PageBack***;
Beschreibt das Display mit eine neuen, zeitlich zurückliegenden Seite.

procedure ***PageForw***;
Beschreibt das Display mit eine neuen, zeitlich vorneliegender Seite.

procedure **SetDispStart(pValue : double);**
Beschreibt das Display mit dem Buffer Inhalt an der Stelle pValue. Damit kann nach einer Aufzeichnung das Display-Fenster an eine bestimmte Stelle im Aufzeichnungsbuffer gelegt werden. Der Wert pValue bestimmt den Aufsetzpunkt (Zeit) innerhalb des Buffers bzw. die linke Anfangszeit des Displays.

procedure ***ParamChange***;
Nach verändern von Aufzeichnungs- oder Bufferparametern **muss** ParamChange aufgerufen werden. Dadurch ist es möglich einen komplett neuen Parametersatz zu installieren, ohne dass bei jeder einzelnen Parameteränderung die Komponente sich neu zeichnet.

procedure ***DispRefresh***;
Erzwingt ein Neuzeichnen des Displays.

procedure ***Repaint***;
Erzwingt ein Neuzeichnen der gesamten Komponente.

procedure ***DrawScala***;
Erzwingt ein Neuzeichnen der Scala.

procedure ***ShowTime***;
Erzwingt ein Neuzeichnen der Zeitangabe rechts unten.

function ***GetDispRect*** : Trect;
Mit GetDispRect erhält man die Grösse und Position der aktiven Display-Zeichenfläche.
Siehe Beispiel bei „property canvas“.

function ***GetBuffer***(adrs : LongInt; var val1, val2, val3, val4 : single) : boolean;
Mit der function
```
GetBuffer(adrs : LongInt; var val1, val2, val3, val4 : single) : boolean;
```
kann der Buffer ausgelesen werden. GetBuffer wird true, wenn „adrs“ eine gültigen Wert hat, d.h. innerhalb des beschriebenen Teils Buffers zeigt. (Siehe property BufferLen)

procedure ***SetBuffer***(val1, val2, val3, val4 : single);
Mit der procedure
```
SetBuffer(val1, val2, val3, val4 : single);
```
werden die 4 Werte in den Buffer geschrieben, und zwar an die Stelle auf die der Startpointer zeigt. Die Pointer werden automatisch weitergeführt. Da gleichzeitig ein Zeichnungsvorgang auf das Display erfolgt, ist diese Operation ***nur im „Record“ Modus*** zulässig.
(Siehe property BufferLen)

function ***ChangeBuffer***(adrs : LongInt; val1, val2, val3, val4 : single) : boolean;
Mit der function
```
ChangeBuffer(adrs : LongInt; val1, val2, val3, val4 : single) : boolean;
```
werden 4 Werte an die Stelle „adrs“ geschrieben. Die Pointer werden nicht verändert, d.h. ein Schreiben auf einen noch nicht beschriebenen Teil des Buffers ist möglich, hat aber dann keine weiteren Folgen bzw. ist ohne Wirkung. (Siehe property BufferLen)

Variable

ZoomStartX : double;
ZoomEndX : double;
ZoomStartY : double;
ZoomEndY : double;

Mit diesen 4 Variablen kann ein Zoomausschnit aus dem Buffer gewählt werden. Dabei ist unbedingt zu beachten, dass sowohl die y-Grenzen(LinStartY und LinEndY bzw. LogStartY und LogEndY) als auch die t-Grenzen nicht überschritten werden. Mit ZoomMode = zmZoomed bzw. zmNormal kann der gewählte Ausschnitt sichtbar gemacht bzw. rückgesetzt werden.

CurveSynth

Stellt Kurven aus bis 40 Punkten durch Interpolation dar. Lineare, Bezier und Spline Interpolation.

Komponenten Deklaration

```
(* in Curves.dcu *)

const
  MaxPolyPnts   = 100;
  MaxSplinePnts = 400;
  GeoRoundOff   = 0.0000001;
  GeoRoundOff2  = GeoRoundOff + GeoRoundOff;
  maxpolydeg    = 10;

type
  RXYCoord = Record {double X and Y Coordinates}
               X    : double;
               Y    : double;
               r    : double;
               PAttr: integer;
             end;

  RLineSeg = Record
               P1: RXYCoord;
               P2: RXYCoord;
             end;

  R30XYArray  = Array[1..30] of RXYCoord;
  XYPolygon   = Array[1..MaxPolyPnts] of RXYCoord;
  polyxypntr  =  ^PolyXYCoord;
  R30Array    = Array[0..30] of double;
  SplinePntr  = ^SplineRecord;
  SplineIdx   = Array[1..MaxSplinePnts] of SplinePntr;

  PolyXYCoord = Record
                  SeqNr:Integer;
                  Pnt:  RXYCoord;
                  Prev: PolyXYPntr;
                  Nxt:  PolyXYPntr;
                end;

  SplineRecord = Record
                   Idx:   Integer; {Sequence Number}
                   Dx:    double;    {Change in X}
                   Dy:    double;    {Change in Y}
                   Dist:  double;    {Distance Prev to current Pnt}
                   DistX: double;    {X Distance}
                   DistY: double;    {Y Distance}
                   ADist: double;    {Accum Distance First pnt to Curr}
                   CosX:  double;    {Cosine of X}
                   SinY:  double;    {Sin Y}
                   XCoeff:double;    {Spline X Coeff}
                   YCoeff:double;    {Spline Y Coeff}
                   Nxt: SplinePntr;{Pntr to next spline point}
                 end;
```

```
Procedure BezierCurve2D(CtrlPnts : R30XYArray; NrCtrlPnts : Integer;
                        NrCurvePnts : Integer; Var CurvePnts : XYPolygon);

Procedure BSpline(CtrlPnts : XYPolygon; NrCtrlPnts, SP, EP, NrCurvePnts :
Integer;
                  Var SplineCurve : XYPolygon);

Procedure tg_polynomial_pnt(nrctrlpnts : integer; Var ctrlpnts : polyxypntr;
                            xvalue : double; Var yvalue, err_est : double);

 (* in CurvSynth.dcu *)

type
  TCurveSynth = class(TRahmen)
  private
    { Private-Deklarationen }
    FOnRefresh  : TNotifyEvent;
    FOnChan1Err : TNotifyEvent;
    FOnChan2Err : TNotifyEvent;
    FOnChan3Err : TNotifyEvent;
    FOnChan4Err : TNotifyEvent;

    FShowError  : boolean;

    CopyBmp     : TBitMap;
    OrgBmp      : TBitMap;

    FCh1CrvTyp  : TCurveTyp;
    FCh1CrvPts  : integer;
    FCh2CrvTyp  : TCurveTyp;
    FCh2CrvPts  : integer;
    FCh3CrvTyp  : TCurveTyp;
    FCh3CrvPts  : integer;
    FCh4CrvTyp  : TCurveTyp;
    FCh4CrvPts  : integer;
    FColCapt    : TColor;
    FColTxtCapt : TColor;
    FCaption    : String32;
    FUnitNameX  : String32;
    FUnitNameY  : String32;
    FUnitColX   : TColor;
    FUnitColY   : TColor;
    FNonLinearX : Boolean;
    FLinStartX  : double;
    FLinEndX    : double;
    FLogStartX  : double;
    FLogEndX    : double;
    FNonLinearY : Boolean;
    FLinStartY  : double;
    FLinEndY    : double;
    FLogStartY  : double;
    FLogEndY    : double;
    FMouseFunc  : TMouseFunc;
    FPositionShow : Boolean;
    FPointerPosX : double;
    FPointerPosY : double;
    FColPointer : TColor;
    FColDisplay : TColor;
    FColGrid    : TColor;
    FColZeroLine: TColor;
    FScalaPartX : Integer;
```

```
    FScalaPartY : Integer;
    FMarkerPosX : double;
    FMarkerPosY : double;
    FColMarker  : TColor;
    FChan1Name  : string;
    FChan2Name  : string;
    FChan3Name  : string;
    FChan4Name  : string;
    FChan1Col   : TColor;
    FChan2Col   : TColor;
    FChan3Col   : TColor;
    FChan4Col   : TColor;
    FChan1Vis   : Boolean;
    FChan2Vis   : Boolean;
    FChan3Vis   : Boolean;
    FChan4Vis   : Boolean;
    Ch1Buff     : TChanBuff;
    Ch2Buff     : TChanBuff;
    Ch3Buff     : TChanBuff;
    Ch4Buff     : TChanBuff;
    Ch1Pts      : xyPolygon;
    Ch2Pts      : xyPolygon;
    Ch3Pts      : xyPolygon;
    Ch4Pts      : xyPolygon;
    FCh1x_Succ  : Boolean;
    FCh2x_Succ  : Boolean;
    FCh3x_Succ  : Boolean;
    FCh4x_Succ  : Boolean;

    painted     : Boolean;
    XAbsP       : LongInt;
    YAbsP       : LongInt;
    XStartL     : double;
    XAbsL       : double;
    XEndL       : double;
    YStartL     : double;
    YAbsL       : double;
    YEndL       : double;
    Log10StartY : double;
    Log10EndY   : double;
    Log10StartX : double;
    Log10EndX   : double;
    GridOffs    : integer;
    TextYpos    : integer;
    axRect,
    ayRect,
    dxRect,
    dyRect,
    CaptRect,
    leftRect,
    topRect,
    dispRect,
    bottomRect  : TRect;
    MouseX,
    MouseY      : LongInt;
    PointerPx   : integer;
    PointerPy   : integer;
    MarkerPx    : integer;
    MarkerPy    : integer;
    Posx, PosY  : double;
    curChan     : integer;
    curPoint    : integer;
    Ch1Change   : Boolean;
```

```
    Ch2Change   : Boolean;
    Ch3Change   : Boolean;
    Ch4Change   : Boolean;
    Ch1Count    : integer;
    Ch2Count    : integer;
    Ch3Count    : integer;
    Ch4Count    : integer;

    PROCEDURE CalcMeasPoints(X, Y: Integer; Shift: TShiftState);
    Procedure SetPositionShow(pValue : Boolean);
    Procedure SetCh1CrvTyp(pValue : TCurveTyp);
    Procedure SetCh1CrvPts(pValue : integer);
    Procedure SetCh2CrvTyp(pValue : TCurveTyp);
    Procedure SetCh2CrvPts(pValue : integer);
    Procedure SetCh3CrvTyp(pValue : TCurveTyp);
    Procedure SetCh3CrvPts(pValue : integer);
    Procedure SetCh4CrvTyp(pValue : TCurveTyp);
    Procedure SetCh4CrvPts(pValue : integer);
    Procedure SetUnitColX(pValue : TColor);
    Procedure SetUnitColY(pValue : TColor);
    Procedure SetUnitNameX(pValue : string32);
    Procedure SetUnitNameY(pValue : string32);
    procedure DrawSignal1;
    procedure DrawSignal2;
    procedure DrawSignal3;
    procedure DrawSignal4;
    Procedure SetNonLinearX(pValue : Boolean);
    Procedure SetNonLinearY(pValue : Boolean);
    Procedure SetLinStartX(pValue : double);
    Procedure SetLinEndX(pValue : double);
    Procedure SetLinStartY(pValue : double);
    Procedure SetLinEndY(pValue : double);
    Procedure SetLogStartX(pValue : double);
    Procedure SetLogEndX(pValue : double);
    Procedure SetLogStartY(pValue : double);
    Procedure SetLogEndY(pValue : double);
    procedure SetChan1Name(pValue: string);
    procedure SetChan2Name(pValue: string);
    procedure SetChan3Name(pValue: string);
    procedure SetChan4Name(pValue: string);
    procedure SetChan1Vis(pValue: Boolean);
    procedure SetChan2Vis(pValue: Boolean);
    procedure SetChan3Vis(pValue: Boolean);
    procedure SetChan4Vis(pValue: Boolean);
    procedure SetCh1X_Succ(pValue: Boolean);
    procedure SetCh2X_Succ(pValue: Boolean);
    procedure SetCh3X_Succ(pValue: Boolean);
    procedure SetCh4X_Succ(pValue: Boolean);
    Procedure SetCaption(pValue : String32);
    Procedure SetChan1Col(pValue : TColor);
    Procedure SetChan2Col(pValue : TColor);
    Procedure SetChan3Col(pValue : TColor);
    Procedure SetChan4Col(pValue : TColor);
    Procedure SetColCapt(pValue : TColor);
    Procedure SetColTxtCapt(pValue : TColor);
    procedure DrawTextCapt;
    procedure SetMouseFunc(pValue: TMouseFunc);
    procedure SetPointerPosX(pValue: double);
    procedure SetMarkerPosX(pValue: double);
    procedure SetPointerPosY(pValue: double);
    procedure SetMarkerPosY(pValue: double);
    Procedure SetScalaPartX(pValue : integer);
    Procedure SetColPointer(pValue : TColor);
```

```
    Procedure SetColMarker(pValue : TColor);
    Procedure SetColDisplay(pValue : TColor);
    Procedure SetColGrid(pValue : TColor);
    Procedure SetColZeroLine(pValue : TColor);
    Procedure SetScalaPartY(pValue : integer);
    function  FHardCopy : TBitmap;
    procedure FClearBmp(b : TBitmap);
    procedure DrawPointer;
    procedure DrawMarker;
    procedure ShowPos;
    procedure DispMessage(chan : Char);
  protected
    { Protected-Deklarationen }
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    function  LoadFromFile(FileName : string) : boolean;
    function  SaveToFile(FileName : string) : boolean;
    PROCEDURE ClipMouse(clip : boolean);
    procedure ParamChange;
    procedure DispRefresh;
    procedure Repaint;
    procedure DrawScala;
    function  GetDispRect : TRect;
    function  CheckCoos(coo : rxyCoord) : boolean;
    procedure ScalePnt(XL,YL : double; var XP,YP : LongInt);
    procedure PointToScale(px, py : integer; var sx, sy : double);
    function  GetCh1Buff(point : integer; var valx, valy : double) : boolean;
    function  GetCh2Buff(point : integer; var valx, valy : double) : boolean;
    function  GetCh3Buff(point : integer; var valx, valy : double) : boolean;
    function  GetCh4Buff(point : integer; var valx, valy : double) : boolean;
    function  SetCh1Buff(point : integer; valx, valy : double) : boolean;
    function  SetCh2Buff(point : integer; valx, valy : double) : boolean;
    function  SetCh3Buff(point : integer; valx, valy : double) : boolean;
    function  SetCh4Buff(point : integer; valx, valy : double) : boolean;
    function  GetCh1Point(point : integer; var valx, valy : double) : boolean;
    function  GetCh2Point(point : integer; var valx, valy : double) : boolean;
    function  GetCh3Point(point : integer; var valx, valy : double) : boolean;
    function  GetCh4Point(point : integer; var valx, valy : double) : boolean;

    function  GetCh1ValueX(valy : double; var valx : double) : boolean;
    function  GetCh1ValueY(valx : double; var valy : double) : boolean;
    function  GetCh2ValueX(valy : double; var valx : double) : boolean;
    function  GetCh2ValueY(valx : double; var valy : double) : boolean;
    function  GetCh3ValueX(valy : double; var valx : double) : boolean;
    function  GetCh3ValueY(valx : double; var valy : double) : boolean;
    function  GetCh4ValueX(valy : double; var valx : double) : boolean;
    function  GetCh4ValueY(valx : double; var valy : double) : boolean;

    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property canvas;
    property Align;
    procedure StyleChanged(Sender : TObject);
    Property HardCopy   : TBitMap   read FHardCopy    write FClearBmp;
    Property Caption    : String32  read FCaption     write SetCaption;
```

```
    Property UnitColX   : TColor    Read FUnitColX    Write SetUnitColX;
    Property UnitColY   : TColor    Read FUnitColY    Write SetUnitColY;
    Property UnitNameX  : String32  read FUnitNameX   write SetUnitNameX;
    Property UnitNameY  : String32  read FUnitNameY   write SetUnitNameY;
    Property Chan1CurvTyp : TCurveTyp read FCh1CrvTyp write SetCh1CrvTyp;
    Property Chan2CurvTyp : TCurveTyp read FCh2CrvTyp write SetCh2CrvTyp;
    Property Chan3CurvTyp : TCurveTyp read FCh3CrvTyp write SetCh3CrvTyp;
    Property Chan4CurvTyp : TCurveTyp read FCh4CrvTyp write SetCh4CrvTyp;
    Property Chan1CurvPts : Integer read FCh1CrvPts   write SetCh1CrvPts;
    Property Chan2CurvPts : Integer read FCh2CrvPts   write SetCh2CrvPts;
    Property Chan3CurvPts : Integer read FCh3CrvPts   write SetCh3CrvPts;
    Property Chan4CurvPts : Integer read FCh4CrvPts   write SetCh4CrvPts;
    property Chan1Name  : string    read FChan1Name   write SetChan1Name;
    property Chan2Name  : string    read FChan2Name   write SetChan2Name;
    property Chan3Name  : string    read FChan3Name   write SetChan3Name;
    property Chan4Name  : string    read FChan4Name   write SetChan4Name;
    property Chan1Col   : TColor    read FChan1Col    write SetChan1Col;
    property Chan2Col   : TColor    read FChan2Col    write SetChan2Col;
    property Chan3Col   : TColor    read FChan3Col    write SetChan3Col;
    property Chan4Col   : TColor    read FChan4Col    write SetChan4Col;
    property Chan1Vis   : boolean   read FChan1Vis    write SetChan1Vis;
    property Chan2Vis   : boolean   read FChan2Vis    write SetChan2Vis;
    property Chan3Vis   : boolean   read FChan3Vis    write SetChan3Vis;
    property Chan4Vis   : boolean   read FChan4Vis    write SetChan4Vis;
    property Chan1X_Succ: boolean   read FCh1X_Succ   write SetCh1X_Succ;
    property Chan2X_Succ: boolean   read FCh2X_Succ   write SetCh2X_Succ;
    property Chan3X_Succ: boolean   read FCh3X_Succ   write SetCh3X_Succ;
    property Chan4X_Succ: boolean   read FCh4X_Succ   write SetCh4X_Succ;
    property CaptBackCol: TColor    read FColCapt     write SetColCapt;
    property CaptionCol : TColor    read FColTxtCapt  write SetColTxtCapt;
    property ColDisplay : TColor    read FColDisplay  write SetColDisplay;
    property ColGrid    : TColor    read FColGrid     write SetColGrid;
    property ColZeroLine: TColor    read FColZeroLine write SetColZeroLine;
    property PointerCol : TColor    read FColPointer  write SetColPointer;
    property MarkerCol  : TColor    read FColMarker   write SetColMarker;
    property ScalaPartX : Integer   read FScalaPartX  write SetScalaPartX;
    Property MouseFunct : TMouseFunc read FMouseFunc  write SetMouseFunc;
    Property PositionShow : Boolean Read FPositionShow Write SetPositionShow;
    Property ShowError  : Boolean   Read FShowError   Write FShowError;
    Property PointerPosX : double   read FPointerPosX write SetPointerPosX;
    Property MarkerPosX : double    read FMarkerPosX  write SetMarkerPosX;
    Property PointerPosY : double   read FPointerPosY write SetPointerPosY;
    Property MarkerPosY : double    read FMarkerPosY  write SetMarkerPosY;
    property ScalaPartY : Integer   read FScalaPartY  write SetScalaPartY;
    Property NonLinearX : Boolean   read FNonLinearX  write SetNonLinearX;
    Property NonLinearY : Boolean   read FNonLinearY  write SetNonLinearY;
    property LinStartX  : double    read FLinStartX   write SetLinStartX;
    property LinStartY  : double    read FLinStartY   write SetLinStartY;
    property LinEndX    : double    read FLinEndX     write SetLinEndX;
    property LinEndY    : double    read FLinEndY     write SetLinEndY;
    property LogStartX  : double    read FLogStartX   write SetLogStartX;
    property LogStartY  : double    read FLogStartY   write SetLogStartY;
    property LogEndX    : double    read FLogEndX     write SetLogEndX;
    property LogEndY    : double    read FLogEndY     write SetLogEndY;
    property onRefresh : TNotifyEvent read FOnRefresh write FOnRefresh;
    property onChan1Err : TNotifyEvent read FonChan1Err write FonChan1Err;
    property onChan2Err : TNotifyEvent read FonChan2Err write FonChan2Err;
    property onChan3Err : TNotifyEvent read FonChan3Err write FonChan3Err;
    property onChan4Err : TNotifyEvent read FonChan4Err write FonChan4Err;
end;
```

ChartBar

Bargraph Element zum graphischen Darstellen von Daten wie bei Spread-Sheets.
Umrandung, Farben und Transparenz einstellbar.

Komponenten Deklaration

```
Type
  TChartBar = class(TGraphicControl)
  private
    FValMin    : double;
    FValMax    : double;
    FValue     : double;
    FColBack   : TColor;
    FColBarPos : TColor;
    FColBarNeg : TColor;
    FColBorder : TColor;
    FOrient    : TOrientation;
    FBorder    : TBordLine;
    FTranspar  : boolean;
    FCaption   : String;
    FColCapt   : TColor;
    XStartL    : double;
    YStartL    : double;
    XAbsL      : double;
    YAbsL      : double;
    painted    : Boolean;
    isSizing   : Boolean;
    isChild    : Boolean;
    CurPos     : integer;
    rAll,
    rPos,
    rNeg       : TRect;


    procedure SetOrientation(pValue : TOrientation);
    procedure SetColBack(pValue : Tcolor);
    procedure SetColBarPos(pValue : Tcolor);
    procedure SetColBarNeg(pValue : Tcolor);
    procedure SetColBorder(pValue : Tcolor);
    procedure SetColCapt(pValue : Tcolor);
    procedure SetCaption(pValue : string);
    procedure SetValue(pValue : double);
    procedure SetValMin(pValue : double);
    procedure SetValMax(pValue : double);
    procedure SetTransp(pValue : boolean);
    procedure SetBorder(pValue : TBordLine);
    procedure DrawValue;
    procedure WMSize(var Message: TWMSize); message WM_SIZE;
  protected
    procedure Paint; Override;
  public
    procedure ScalePnt(XL, YL : double; var XP, YP : integer);
    procedure StyleChanged(Sender : TObject);
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override;
  published
    property OnClick;
```

```
    property OnDblClick;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property Visible;
    property ColCapt      : TColor  read FColCapt   write SetColCapt;
    property Caption      : string  read FCaption   write SetCaption;
    property ValueMin     : double  read FValMin    write SetValMin;
    property ValueMax     : double  read FValMax    write SetValMax;
    property Value        : double  read FValue     write SetValue;
    property ColBarPos    : TColor  read FColBarPos write SetColBarPos;
    property ColBarNeg    : TColor  read FColBarNeg write SetColBarNeg;
    property ColBack      : TColor  read FColBack   write SetColBack;
    property ColBorder    : TColor  read FColBorder write SetColBorder;
    property Border       : TBordLine read FBorder  write SetBorder;
    property Transparent  : boolean read FTranspar  write SetTransp;
    property Orientation : TOrientation read FOrient write SetOrientation;
  end;
```

ChartBase

Basiselement zum plazieren und steuern der ChartBars.

Komponenten Deklaration

```
Type
  TChartBase = class(TCustomPanel)
  protected
    { Protected-Deklarationen }
    dispBmp     : TBitMap;
    FColHead    : TColor;
    FColTxtHead : TColor;
    FNameHead   : String32;
    FDisplayTyp : tdispType;
    FOrient     : TOrientation;
    FValueMin   : double;
    FValueMax   : double;
    FBorder     : TBordLine;
    FScalaPart  : Integer;
    FColGrid    : TColor;
    FColDisp    : TColor;
    FUnitName   : String;
    FUnitCol    : TColor;
    XAbsP       : LongInt;
    YAbsP       : LongInt;
    XStartL     : double;
    YStartL     : double;
    XAbsL       : double;
    YAbsL       : double;
    SkLinks,
    SkRechts,
    SkOben,
    SkUnten    : LongInt;
    headRect,
    leftRect,
    topRect,
    dispRect,
    bottomRect : TRect;
    ChildCount : integer;
    painted    : Boolean;

    Procedure SetColGrid(pValue : TColor);
    Procedure SetColDisp(pValue : TColor);
    Procedure SetUnitName(pValue : string);
    Procedure SetUnitCol(pValue : TColor);
    Procedure SetScalaPart(pValue : integer);
    Procedure SetDisplayTyp(pValue : tdispType);
    Procedure SetNameHead(pValue : String32);
    Procedure SetColHead(pValue : TColor);
    Procedure SetColTxtHead(pValue : TColor);
    Procedure SetOrient(pValue : TOrientation);
    procedure SetValMin(pValue : double);
    procedure SetValMax(pValue : double);
    procedure SetBorder(pValue : TBordLine);
    procedure DrawTextHead;
    procedure NewChild;
    procedure RemoveChild;
    procedure ChildSetup;
```

```
    procedure DrawDisp;
    procedure DrawTransp(fr : TRect);
    procedure DrawScala;
    procedure Paint; override;
  public
    { Public-Deklarationen }
    bRect     : TRect;
    procedure StyleChanged(Sender : TObject);
    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;
  published
    { Published-Deklarationen }
    property Align;
    property BevelInner;
    property BevelOuter;
    property BevelWidth;
    property BorderWidth;
    property BorderStyle;
    property Enabled;
    property Color;
    property Ctl3D;
    property Font;
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property ShowHint;
    property TabOrder;
    property TabStop;
    property Visible;
    property OnClick;
    property OnDblClick;
    property OnEnter;
    property OnExit;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnResize;
    Property NameHead    : String32  read FNameHead    write SetNameHead;
    property DisplayTyp  : tdispType read FDisplayTyp  write SetDisplayTyp;
    property ColHead     : TColor    read FColHead     write SetColHead;
    property ColTxtHead  : TColor    read FColTxtHead  write SetColTxtHead;
    property BarOrient   : TOrientation read FOrient   write SetOrient;
    property BarValueMin : double    read FValueMin    write SetValMin;
    property BarValueMax : double    read FValueMax    write SetValMax;
    property BarBorder   : TBordLine read FBorder      write SetBorder;
    Property UnitName    : String    read FUnitName    write SetUnitName;
    Property UnitCol     : TColor    Read FUnitCol     Write SetUnitCol;
    property ScalaPart   : Integer   read FScalaPart   write SetScalaPart;
    property ColGrid     : TColor    read FColGrid     write SetColGrid;
    property ColDisplay  : TColor    read FColDisp     write SetColDisp;
  end;
```

TextScroll

Textanzeige/Editor. Umschaltbar von Ascii in Hexmodus. Umfangreiche Manipulations-Möglichkeiten des Erscheinungsbildes. Diverse Highlights.

Komponenten Deklaration

```
type
  {Search option flags for editor and viewer}
  TSearchOptions = (
     soFind,          {find  (this option is assumed)    }
     soBackward,    {search backwards                    }
     soMatchCase,   {don't ignore case when searching    }
     soGlobal,        {search globally                   }
     soReplace,       {find and replace         (editor only)}
     soReplaceAll,  {find and replace all     (editor only)}
     soWholeWord,   {match on whole word only (editor only)}
     soSelText);    {search in selected   text  (editor only)}
  TSearchOptionSet = set of TSearchOptions;

type
  {types of tabs supported in the editor}
  TTabType   = (ttReal, ttFixed, ttSmart);

  tsHexSize  = (hsByte, hsWord,     hsLong);

  {entry field flag for display     of number field   with zero value}
  TZeroDisplay = (zdShow, zdHide, zdHideUntilModified);

  TtsCaretShape   = (             {Predefined   caret shapes..}
                csBlock,           {..block over whole cell}
                csHalfBlock,     {..block over bottom part of cell}
                csVertLine,        {..vertical line to left of cell}
                csHorzLine,        {..horizontal line on bottom of cell}
                csCustom,          {..custom width/height}
                csBitmap);         {..bitmap caret, custom width/height}

  TtsCaretAlign   = (             {Alignment of caret in cell..}
                caLeft,      {..left side, centered vertically}
                caTop,       {..top side, centered horizontally}
                caRight,           {..right side, centered vertically}
                caBottom,          {..bottom side, centered horizontally}
                caCenter);         {..centered vertically and     horizontally}

  (* TTextScroll *)
  {user     command     notify event}
  TUserCommandEvent =
    procedure(Sender : TObject;     Command     : Word)
    of object;

  TShowStatusEvent =
    procedure(Sender : TObject;     LineNum     : Integer; ColNum : Integer;
            overwrite, changed : boolean) of object;
    {-event to notify of a viewer or editor caret position change}

  TTopLineChangedEvent =
    procedure(Sender : TObject;     Line : Integer)
    of object;
```

```
    {-event to notify when the top line   changes     for the     editor or
viewer}

  TFontChangeEvent =
    procedure(Sender : TObject;     fHeight, fwidth   : Integer)
    of object;
    {-event to notify when font     size changes for the editor or viewer}

  THexChangeEvent =
    procedure(Sender : TObject;     adr : Integer; b : byte) of object;
    {-event to notify when the user edited the line}

  TTextChangeEvent =
    procedure(Sender : TObject;     Line, Col : Integer; c : char) of object;
    {-event to notify when the user edited the line}


const
  {default property values for viewer and descendants}
  DefBorderStyle     = bsSingle;
  DefBufferPageCount = 16;
  DefColor       = clWhite;
  DefExpandTabs        = True;
  DefFilterChars     = True;
  DefHeight      = 150;
  DefHighlightBack   = clHighlight;
  DefHighlightText   = clHighlightText;
  DefIsOpen      = False;
  DefMarginColor     = clSilver;
  DefScrollBars        = ssBoth;
  DefShowBookmarks   = True;
  DefShowCaret         = True;
  DefTabSize           = 8;
  DefTabStop           = True;
  DefWidth       = 200;

  MaxMarkers           = 10;    {maximum number of text markers}
  MaxdbMarks           = 100;   {maximum number of single   markers}

type
  TTextScroller   = class(TCustomControl)
  protected {private}
    FFileName           :     PString;
    FFileSize           :     Integer;

    {property fields - persistent}
    FBorderStyle      : TBorderStyle;
    FExpandTabs         :     Boolean;
    FColBack            :     TColor;
    FColBack2           :     TColor;
    FColText            :     TColor;
    FColHBack           :     TColor;
    FColHText           :     TColor;
    FMargLeftCol      : TColor;
    FMargTopCol         :     TColor;
    FColMargTxt         :     TColor;
    FMargLeft           :     integer;
    FMargTop            :     integer;
    FMargCapt           :     string;
    FFontSize           :     integer;
    FScrollBars         :     TScrollStyle;
    FShowBookmarks    : Boolean;
    FShowCaret          :     Boolean;
```

```
    FTabSize            :     Byte;
    FPassive            :     boolean;
    FMarkGlyphs         :     TBitMap;     {marker glyphs}
    FdbMarkGlyph      : TBitMap;     {marker glyph}
    FCursorMarg         :     TCursor;
    FNonHexChr          :     char;
    FHexMode            :     Boolean;
    FHexSize            :     tsHexSize;
    FHexBuf       :     integer;
    FHexOffs            :     integer;
    FReadOnly           :     boolean;
    FOverWrite          :     boolean;
    Fdirty        :     boolean;
    FIntegralH          :     boolean;
    FdsgWidth           :     integer;
    FdsgHeight          :     integer;

    {property fields - set at runtime}
    FCaretEffCol      : Integer;
    FCaretPos           :     TtsTextPos;
    FLineCount          :     Integer;
    FMarkers            :     array[0..MaxMarkers-1] of TtsTextPos;
    FdbMarks            :     array[0..MaxdbMarks-1] of integer;
    FTopLine            :     Integer;

    {event variables}
    FOnShowStatus     : TShowStatusEvent;
    FOnTopLineChanged : TTopLineChangedEvent;
    FOnUserCommand    : TUserCommandEvent;
    FOnHexChange      : THexChangeEvent;
    FOnTextChange     : TTextChangeEvent;
    FOnFontChange     : TFontChangeEvent;

    {Other fields}
    fvLines : TtsStringList;

    {other internal fields}
    tsAmFocused         :     Boolean;     {True if focused}
    tsAnchor            :     TtsTextPos;  {anchor for highlighting}
    tsCaret       :     TtsSingleCaret;    {our caret}
    tsCols        :     Integer;     {number of columns in window}
    tsColWid            :     Integer;     {width     of one column}
    tsDefaultChar     : Char;        {replacement Char for current font}
    tsDitheredBG      : Boolean;     {True if a background color is
      dithered}
    tsDivisor           :     Integer;     {divisor for scroll bars}
    tsFile        :     Integer;     {handle of file being browsed}
    tsFirstChar         :     Char;        {first     Char in     current
      font}
    tsHighlight         :     TtsRange;    {highlight range}
    tsRange       :     TtsRange;    {highlight range, also if highlight off}
    tsRange2            :     TtsRange;    {background2 range}
    tsHDelta            :     Integer;     {horizontal scroll delta}
    tsHScroll           :     Boolean;     {True if we have a horizontal scroll
bar}
    tsHSHigh            :     Integer;     {horizontal scroll limit}
    tsLastChar          :     Char;        {last Char in current font}
    tsLineSelCursorOn : Boolean;     {is line selection cursor in use?}
    tsRowHt       :     Integer;     {height of one   row}
    tsRows        :     Integer;     {number of rows in window}
    tsSaveCursor      : HCursor;     {saved     cursor for lengthy waits}
    tsVScroll           :     Boolean;     {True if we have a vertical scroll
bar}
```

```
    tsVSHigh            :     Integer;     {vertical scroll limit}
    tsVSMax       :     Integer;     {max value for   vertical scroll   bar}
    tsWaiting           :     Boolean;     {True if wait cursor is shown}
    tsLeftMargin      : integer;     {MUST be even}
    tsTopMargin         :     integer;
    tsMousePos          :     TtsTextPos;
    HexBufPtr           :     pointer;
    ExternBuf           :     boolean;
    SavShowCaret      : Boolean;
    SavCaretShape     : TtsCaretShape;
    SaveCaretBMP      : TBitMap;
    SavCaretHotX      : integer;
    SavCaretHotY      : integer;
    HexBmp        :     TBitMap;
    HexCapt       :     string;
    painted       :     boolean;
    dsgWidth            :     integer;
    dsgHeight           :     integer;

  protected
    {private message methods}

  public
    {VCL methods}
    constructor   Create(AOwner :   TComponent); override;
    destructor Destroy; override;

    function GetColumnWidth : integer;
    function GetRowHeight : integer;
    function GetHexLineLength :     integer;
    procedure ExternHexBuf(BufPtr : pointer; BufSize : integer);
    procedure RefreshHexLine(LineNum : Integer);

    function ActualColumn(Line : Integer; EffectiveCol : Integer) : Integer;
      {-return the actual column given the line and effective column}
    procedure ClearMarker(N : Integer);
      {-remove the specified text marker}
    procedure CleardbMarker(Line : Integer);
      {-remove the specified text marker}
    procedure ClearAllMarker;
      {-remove all text marker}
    procedure ClearAlldbMarker;
      {-remove all text marker}
    procedure CopyToClipboard;
      {-copy highlighted text to the clipboard}
    function  EffectiveColumn(Line : Integer; ActualCol     : Integer) :
Integer;
      {-return the effective column given the line and actual column}
    function  GetCaretPosition(var Col : Integer) : Integer;
      {-return the current position of the caret}
    function  GetCaretPositionHex : integer;
    function  GetMousePosition(var Col : Integer) : Integer;
      {-return the current position of the mouse}
    function  GetLine(LineNum :     Integer; Dest :   PAnsiChar; DestLen :
Integer) :  PAnsiChar;
      {-get the   specified line}
    function  GetMarkerPosition(N : Integer; var Col : Integer)   : Integer;
      {-return the current position of the specified marker}
    function  isMarkerSet(N : Integer; Line, Col : Integer) : boolean;
      {-return true if the specified marker is set}
    function  isdbMarkerSet(Line : Integer) : boolean;
      {-return true if the specified marker is set}
    function  GetPrintableLine(LineNum : Integer; Dest : PAnsiChar;
```

```
                        DestLen : Integer) : Integer;
      {-get a line in a format suitable   for printing}
    function  GetSelection(var Line1 : Integer; var Col1 : Integer;
                    var Line2 : Integer; var Col2 : Integer) : Boolean;
      {-get a line in a format suitable   for printing}
    function  GetRange(var Line1 : Integer; var Col1 : Integer;
                  var Line2 : Integer; var     Col2 : Integer)   : Boolean;
      {-return True if any text     is currently selected}
    function  GetRange2(var Line1 : Integer; var Line2 : Integer) : Boolean;
      {-return True if any background is currently selected}
    procedure  GotoMarker(N : Integer);
      {-move the caret to the specified   text marker}
    function  Search(const S : string; Options : TSearchOptionSet) : Boolean;
      {-search for a string returning True if found}
      virtual;
    function SearchHex(FindByte1, FindMask1, FindByte2,     FindMask2,
                  FindByte3, FindMask3, FindByte4,   FindMask4 : byte;
                  Options : TSearchOptionSet) : Boolean;

    procedure ResetHighlight;
    procedure SelectAll(CaretAtEnd : Boolean);
      {-select all text in the editor}
    procedure GotoLine(Line : Integer);
      {-move the caret to a specified line }
    procedure SetCaretPosition(Line : Integer; Col : Integer);
      {-move the caret to a specified line and column}
    procedure SetMarker(N : Integer);
      {-set a text marker at the current caret position}
    procedure SetMarkerAt(N : Integer; Line : Integer; Col : Integer);
      {-set a text marker at the specified position}
    procedure SetdbMarker(Line : Integer);
      {-set a text marker at the current caret position}
    procedure SetMiniMarker(Line : Integer; show : boolean);
    function isMiniMarkerSet(Line : Integer) : boolean;

    procedure SetSelection(Line1 : Integer; Col1 : Integer;
                    Line2 : Integer; Col2 : Integer;
                    CaretAtEnd :   Boolean);
      {-select a region of text}
    procedure SetRange2(Line1 :     Integer; Line2 : Integer);
      {-select a region of text     for background2}
    procedure ReSetRange2;
      {-unselect a region of text for background2}

    procedure AppendString(st :     string);
    procedure AppendHexByte(b :     byte; refresh :   BOOLEAN);
    function  ReplaceString(Line : integer; st : string) : boolean;
    function  GetHexByAdr(adr :     integer) : byte;
    procedure SetHexByAdr(adr :     integer; b : byte);
    function  GetHexByLine(Line, Col : integer) : byte;
    procedure SetHexByLine(Line, Col : integer; b : byte);
    function  GetWordAtCaret(var cPos :   TPoint)     : string;
    function  GetCharAtCaret(var cPos :   TPoint)     : char;
    function  GetWordAtMouse(var cPos :   TPoint)     : string;
    function  GetCharAtMouse(var cPos :   TPoint)     : char;
    procedure SetCharAtCaret(ch     :char);

    function  GetLinePtr(LineNum : Integer; var Len : Integer) : PAnsiChar;
    function  GetLinePtrM(LineNum : Integer; var Len : Integer;   var mark :
boolean) : PAnsiChar;

    function  CheckLine(LineNum     : Integer) : Integer;
```

```
    function  SaveToFile(FName : string; canoverwrite :     boolean) : boolean;
    function  LoadFromFile(FName : string) : boolean;
    procedure Flush;


    {public properties}
    property CaretActualPos : TtsTextPos read GetCaretActualPos
                              write SetCaretActualPos;

    property CaretEffectivePos : TtsTextPos read GetCaretEffectivePos
                                 write SetCaretEffectivePos;


    property LineCount : Integer read FLineCount write SetLineCount;

    property LineLength[LineNum     : Integer] : Integer read GetLineLength;

    property Lines[LineNum : Integer] :   string read GetStringLine;

    property TopLine : Integer read GetTopLine write SetTopLine   stored False;

    property dirty     : Boolean   read   Fdirty         write Fdirty;

  published
    {properties}
    property BorderStyle : TBorderStyle   read FBorderStyle
           write SetBorderStyle default DefBorderStyle;
    property Caret : TtsCaret read GetCaretType write SetCaretType;
    property ExpandTabs : Boolean read FExpandTabs
           write SetExpandTabs default DefExpandTabs;
    property FontSize   : integer read FFontSize  write     SetFontSize;
    property ColorHBack : TColor  read FColHBack  write     SetColorHBack;
    property ColorHText : TColor  read FColHText  write     SetColorHText;
    property ColorBack  : TColor  read FColBack   write     SetColorBack;
    property ColorBack2 : TColor  read FColBack2  write     SetColorBack2;
    property ColorText  : TColor  read FColText   write     SetColorText;
    property MarginLeft : integer read FMargLeft  write     SetMargLeft;
    property MarginTop  : integer read FMargTop   write     SetMargTop;
    property MarginCapt : string  read FMargCapt  write     SetMargCapt;
    property MargCursor : TCursor read FCursorMarg write FCursorMarg;
    property MargTxtCol : TColor  read FColMargTxt write SetColMargTxt;
    property MargLeftColor : TColor read FMargLeftCol
           write SetMargLeftCol default DefMarginColor;
    property MargTopColor : TColor read   FMargTopCol
           write SetMargTopCol default DefMarginColor;
    property ScrollBars : TScrollStyle read FScrollBars
           write SetScrollBars default DefScrollBars;
    property ShowBookmarks : Boolean read FShowBookmarks
           write SetShowBookmarks default DefShowBookmarks;
    property ShowCaret : Boolean read FShowCaret
           write SetShowCaret default     DefShowCaret;
    property MarkGlyphs : TBitMap read GetMarkGlyph write SetMarkGlyph;
    property MarkDbGlyph : TBitMap read   GetdbMarkGlyph write SetdbMarkGlyph;
    property Passive    : Boolean   read   FPassive   write SetPassive;
    property ReadOnly   : Boolean   read   FReadOnly  write SetReadOnly;
    property OverWrite : Boolean   read   FOverWrite write SetOverWrite;
    property NonHexChar : Char         read     FNonHexChr write SetNonHexChr;
    property HexMode    : Boolean  read   FHexMode   write SetHexMode;
    property IntegHeight: Boolean  read   FIntegralH write SetIntegralH;

    property HexSize    : tsHexSize read   FHexSize  write   SetHexSize;
    property HexBufLen : integer   read   FHexBuf       write     SetHexBuf;
    property HexOffset : integer   read   FHexOffs  write   SetHexOffs;
```

```
    property TabSize   : Byte    read     FTabSize  write   SetTabSize default
DefTabSize;
    {events}
    property OnShowStatus : TShowStatusEvent read FOnShowStatus   write
FOnShowStatus;
    property OnTopLineChanged :     TTopLineChangedEvent read FOnTopLineChanged
           write FOnTopLineChanged;
    property OnUserCommand : TUserCommandEvent read FOnUserCommand
           write FOnUserCommand;
    property OnHexChange  : THexChangeEvent  read FOnHexChange write
FOnHexChange;
    property OnTextChange : TTextChangeEvent read FOnTextChange   write
FOnTextChange;
    property OnFontChange : TFontChangeEvent read FOnFontChange   write
FOnFontChange;

    property FileName : string read GetFileName write SetFileName;
    property IsOpen : Boolean  read GetIsOpen write SetIsOpen default DefIsOpen;

    property Align;
    property DragCursor;
    property DragMode;
    property ParentShowHint;
    property PopupMenu;
    property ShowHint;
    property TabOrder;
    property TabStop default DefTabStop;
    property Visible;

    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnStartDrag;
  end;
```