AutoSizeControlBar

Type: **Boolean**
Default: **True**

Description:
Use this property when you are presenting the user with a resizable calendar (on a form with the Align property set to alClient for example). If True the proportion of the control bar height to the height of the calendar will remain constant. If False the control bar height will remain constant (as set with ControlBarHeight)

Example:

```
{turn autosizing off}
KSCalendar1.AutoSizeControlBar := False;
```

See Also:
ControlBarHeight
ControlBarPosition
ControlBarStyle

BackColor

Type: **TColor**
Default: **clWhite**

Description:
The back color is the color that appears behind the day text and the day border (if you are showing it).

Example:

```
{set the BackColor to red}
KSCalendar1.BackColor := clRed;
```

See Also:
CalendarColor
NavBackColor
ThreeDLightColor
ThreeDShadowColor

BackColorMarked

Type: **TColor**
Default: **clYellow**

Description:
The back color is the color that appears behind the day text of days specified by the MarkedDays property.

Example:

```
{set the BackColorMarked to silver}
KSCalendar1.BackColorMarked := clSilver;
```

See Also:
MarkedDays

Border

Type: **Boolean**
Default: **True**

Description:
Determines if the control will draw a thin black frame around the entire calendar.

Example:

```
{turn border off}
KSCalendar1.Border:= False;
```

See Also:
BorderBevel
BorderColor

BorderBevel

Type: **Integer**
Default: **1**

Description:
This is the bevel width around the edges of the calendar. Legal values are from 0 (2D) to 5. If you choose a larger bevel size you might have to tweak the margin value a little to improve the look of the calendar.

Example:

```
{set the border bevel to a width of 3 pixels}
KSCalendar1.BorderBevel := 3;
```

See Also:
Border
BorderColor

BorderColor

Type: **TColor**
Default: **clBlack**

Description:
This is the color of the border (See Border Property) around the calendar.

Example:

```
{set the BorderColor to yellow}
KSCalendar1.BorderColor := clYellow;
```

See Also:
Border
BorderBevel

CalendarColor

Type: **TColor**
Default: **clSilver**

Description:
This is the surface color of the calendar.
Example:

```
{set the CalendarColor to dark gray}
KSCalendar1.CalendarColor := clGray;
```

See Also:
BackColor
ForeColor
ThreeDLightColor
ThreeDShadowColor

CellBevel

Type: **Integer**
Default: **1**

Description:
This is the bevel width around the edges of the day cells. Legal values are from 0 (2D) to 5. If you choose a larger bevel size you might have to increase the size of the calendar to improve the look of the calendar.

Example:

```
{set the CellBevel to a width of 2 pixels}
KSCalendar1.CellBevel := 2;
```

See Also:
ThreeDLightColor
ThreeDShadowColor

Technical Support & Contacting Kinetic Software

We welcome any and all contact with our customers. Please drop us a note with your feedback or questions!

You can contact Kinetic Software via:

Internet: **kinetic@usit.net**
World Wide Web: **http://www.esper.com/kinetic**

Compuserve Address: **102065,563**
Compuserve via Internet: **102065.563@Compuserve.Com**

Snail Mail: **Kinetic Software**
8404 Swathmore Court
Knoxville, TN
37919

For information on other Delphi components from Kinetic Software please visit our WWW site (address above)!

KSCalendar Date Kit v2.0

The following topics are available in this help file:

ControlBarHeight

Type: **Integer**
Default: **23**

Description:
The height of the control bar in pixels. See the Glyph properties for notes on adjusting this height to suit custom bitmaps.

Example:

```
{set the ControlBarHeight to 30 pixels}
KSCalendar1.ControlBarHeight := 30;
```

See Also:
AutoSizeControlBar
ControlBarPosition
ControlBarStyle

ControlBarHintText

Type: **TStringList**
Default: **Last Year, Last Month, Next Month, Next Year, Cancel, Select**

Description:
Use this property to adjust the hint text for the control bar. KSCalendar will only utilize the first 4 strings you enter here (6 if you are using the advanced control bar). You can also use this property to help internationalize the calendar (see International Support). If you are switching control bar styles a lot it is advisable to make sure that there are always 6 strings available to be used as hints!!

Example:

```
{change the hint text}
KSCalendar1.ControlBarHintText.Strings[0] := 'Back Year';
KSCalendar1.ControlBarHintText.Strings[1] := 'Back Month';
KSCalendar1.ControlBarHintText.Strings[2] := 'Forward Month';
KSCalendar1.ControlBarHintText.Strings[3] := 'Forward Year';
KSCalendar1.ControlBarHintText.Strings[4] := 'Cancel';
KSCalendar1.ControlBarHintText.Strings[5] := 'OK';
```

See Also:
ControlBarHints

ControlBarHints

Type: **Boolean**
Default: **True**

Description:
Determines whether the control bar will show hints to the user.

Example:

```
{turn ControlBarHints off}
KSCalendar1.ControlBarHints := False;
```

See Also:
ControlBarHintText
ControlBarStyle

ControlBarPosition

Type: **Enumerated (cbTop, cbBottom)**
Default: **cbBottom**

Description:
The control bar can be placed at the bottom of the component or just below the title (month and year text line).

Example:

```
{set the ControlBarPosition to the top part of the calendar}
KSCalendar1.ControlBarPosition := cbTop
```

See Also:
ControlBarHeight
ControlBarStyle

ControlBarStyle

Type: **Enumerated (cbsNone, cbsNormal, cbsAdvanced)**
Default: **cbsNormal**

Description:
The advanced control bar (cbsAdvanced) offers you two extra control bar buttons that you can use. The two buttons fire the custome events: OnFunction1Click and OnFunction2Click. Think of these two buttoms as "programmable". For example you can use these two buttons when you are building a "popup" KSCalendar (see DEMO.PAS for a sample popup).

Example:

```
{set the ControlBarStyle to advanced}
KSCalendar1.ControlBarStyle := cbsAdvanced;
```

See Also:
ControlBarHeight
ControlBarPosition

Date

Type: **TDateTime (Read Only)**
Default: **N/A**

Description:
This is the main date property of the KSCalendar. This property reflects the calendar's currently selected date. Note: This is a read only property ... if you want to set the calendar's date use the public method SetDate.

Example:

```
{read the KSCalendar's Date}
var
    MyDate: TDateTime;
begin
    MyDate := KSCalendar1.Date;
```

See Also:
DateText
Day
Month
Year
SetDate

DateFormatString

Type: **TCaption**
Default: **"MM/DD/YY"**

Description:
Enter the format string you would like to use to output text to the DateText property (above). For more information on how to use this property see the FormatDateTime function in Delphi. The KSCalendar uses the same formatting protocol.

For more information on this format string see the FormatDateTime function in Delphi.

Example:

```
{change the DateFormatString to display like: "Sep 09, 1995"}
KSCalendar1.DateFormatString := 'mmm dd, yyyy'

{change the DateFormatString to display like: "Saturday September 9, 1995"}
KSCalendar1.DateFormatString := 'dddd mmmm d, yyy'
```

See Also:
Date
DateText
TitleStyle

DateText

Type: **TCaption (Read Only)**
Default: **N/A**

Description:
This property contains the date as a formatted string. The formatting is controlled by the DateFormatString property (below).

Example:

```
{place the calendar's DateText into an edit control}
Edit1.Text := KSCalendar1.DateText;
```

See Also:
Date
DateFormatString
TitleStyle

DateToDayNumber

Declaration: **function DateToDayNumber(ADate: TDateTime): Integer;**

Description:

This is the opposite of the above function. Pass in a Delphi date and this method will return the day number. This function of course takes leap years into consideration.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DayNumber
DayNumberToDate

DateToWeekNumber

Declaration: function DateToWeekNumber(ADate: TDateTime): Integer;

Description:

Use this function to find out what week number a particular date will fall into. Week numbers returned can range from 1 to 53.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
WeekNumber
WeekNumberToDate

Day

Type: **Integer**
Default: **N/A**

Description:
Contains the currently selected calendar day. Values will range from 1 to 31.

Example:

```
{read the currently selected Day on the calendar}
var
    MyDay: Integer;
begin
    MyDay := KSCalendar1.Day;
```

See Also:
Date
FontHighLight
HighlightBackColor
Month
Year
SetDate

DayNumber

Type: **Integer**
Default: **N/A**

Description:
This property stores the day number of the year. For example if the calendar date (see Date property) is January 1, 1995...DayNumber will be 1. If the date is December 31, 1990 then the DayNumber will be 365 (or 366 if 1990 was a leap year).

Example:

```
{change the currently selected day in the calendar to January 20 of the current year}
KSCalendar1.DayNumber := 20;
```

See Also:
DayNumberToDate
DateToDayNumber

DayNumberToDate

Declaration: **function DayNumberToDate(AYear,ADayNum: Integer): TDateTime;**

Description:

This function will return a Delphi date (TDateTime) for a given day number. Because leap years screw things up you have to pass in the year you are interested in as well as the day number. If you pass in an illegal ADayNum (like 370) this method will return a -1.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DayNumber
DateToDayNumber

DayRowHeight

Type: **Integer**
Default: **15**

Description:
This is the height of the area of the rectangle that contains the day headings (S M T W T F S). You can increase this value if you plan to use a larger font for the day headings. Setting this value to 0 is the same as setting ShowDayRow to False.

Example:

```
{set the DayRowHeight to 20 pixels}
KSCalendar1.DayRowHeight := 20;
```

See Also:
ShowDayRow
ShowDayRowBorder

DayTextHorzPos

Type: **Enumerated (hLeft, hCenter, hRight)**
Default: **hRight**

Description:
This property allows you to designate the position of the DayText (day numbers) in each day cell.

Example:

```
{move the day text to the right of each cell}
KSCalendar1.DayTextVertPos := hRight;
```

See Also:
DayTextVertPos

DayTextVertPos

Type: **Enumerated (vTop, vCenter, vBottom)**
Default: **vCenter**

Description:
This property allows you to designate the position of the DayText (day numbers) in each day cell.

Example:

```
{move the day text to the bottom of each cell}
KSCalendar1.DayTextVertPos := vBottom;
```

See Also:
DayTextHorzPos

DayTitleLetters

Type: **Integer**
Default: **1**

Description:
This property controls how many characters the calendar will use to display the day titles. A setting of 1 would produce S M T W T F S, a setting of 2 would produce Su Mo Tu We Th Fr Sa. A seeting of 9 of produce the full day titles.

Example:

```
{set DayTitleLetters to 3 to display day titles like Sun, Mon Tue...}
KSCalendar1.DayTitleLetters := 3;
```

See Also:
DayTitles
ShowDayRow

DayTitles

Type: **TStringList**
Default: **Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday**

Description:
Use this property to change the text displayed as DayTitles. (see International Support)

Example:

```
{set the calendar for French DayTitles}
KSCalendar1.DayTitles.Strings[0]:='Dimanche';
KSCalendar1.DayTitles.Strings[1]:='Lundi';
KSCalendar1.DayTitles.Strings[2]:='Mardi';
KSCalendar1.DayTitles.Strings[3]:='Mecredi';
KSCalendar1.DayTitles.Strings[4]:='Jeudi';
KSCalendar1.DayTitles.Strings[5]:='Vendredi';
KSCalendar1.DayTitles.Strings[6]:='Samedi';
```

See Also:
DayTitleLetters
International Support

DaysBetween

Declaration: function DaysBetween(ADate, BDate: TDateTime): Integer;

Description:

Use this method to calculate the number of days between two dates. The order of the dates is important since this function will return the difference between the two dates as a positive or negative integer. If the two dates are identical this function will return a 0. If BDate is before ADate this function will return a negative number.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
Date

DaysInMonth

Declaration: **function DaysInMonth(AYear, AMonth: Integer): Integer;**

Description:

Use this function to find out how many days (28,29,30 or 31) are in a given month. Once again you must pass in the year and month because this method will take leap years into account.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DaysInYear
DaysRemaining

DaysInYear

Declaration: function DaysInYear(AYear: Integer): Integer;

Description:

Use this function to find out how many days (28,29,30 or 31) are in a given month. Once again you must pass in the year and month because this method will take leap years into account..

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DaysInMonth
DaysRemaining

DaysRemaining

Declaration: function DaysRemaining(Index: Integer; ADate: TDateTime): Integer;

Description:

Use this function to find out how many days are remaining in a month or in a year. To find the number of days left in a month call this method with Index=0. To find the number of days left in a year call this method with Index=1. Pass in the date you are interested in using ADate!

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DaysInMonth
DaysInYear

Delphi 1.0 Installation (16 bit)

The KSCalendar Date Kit v2.0 diskette comes with both Delphi 1.0 (16 bit) and Delphi 2.0 (32 bit) versions.

By default the installation program installs the Delphi 2.0 (32 bit) compatible version...if you are using the KSCalendar Date Kit with Delphi 1.0 do the following:

1. Use the default install program (setup.exe) to install the Delphi 2.0 version.
2. Locate the DELPHI10 directory (created in the install destination directory).
3. Replace the installed Delphi 2.0 files in the install directory with the files in the DELPHI10 directory.

If you have any trouble building projects in Delphi 1.0 try removing all of the KSCalendar Date Kit v2.0 DCU (compiled Delphi unit) files and then rebuilding your project.

Sample Projects
The sample projects supplied with the KSCalendar Date Kit v2.0 are all native Delphi 1.0 (16 bit) projects.

Events Reference

The KSCalendar utilizes the following custom events:

OnDateChange
OnDrawCell
OnFunction1Click
OnFunction2Click

FontDay

Type: **TFont**
Default: **Arial, Regular, 8, clBlack**

Description:
Set the font for all the non-selected days on the calendar.

Example:

```
{make the FontDay a little larger}
KSCalendar1.FontDay.Size := 10;
```

See Also:
FontDayTitle
FontHighLight
FontMonth
FontNav
FontYear

FontDayTitle

Type: **TFont**
Default: **Arial, Regular, 8, clBlack**

Description:
Set the font for the day titles.

Example:

```
{make the FontDayTitle a little smaller}
KSCalendar1.FontDayTitle.Size := 7;
```

See Also:
FontDay
FontHighLight
FontMonth
FontNav
FontYear

FontHighLight

Type: **TFont**
Default: **Arial, Regular, 8, clWhite**

Description:
Set the font for the currently higlighted day on the calendar.

Example:

```
{make the FontHighLight a little larger}
KSCalendar1.FontHighLight.Size := 10;
```

See Also:
FontDay
FontDayTitle
FontMonth
FontNav
FontYear

FontMarked

Type: **TFont**
Default: **Arial, Regular, 8, clBlack**

Description:
Set the font for the day text in the marked days. For this property to have any meaning you must set the MarkedDays property.

Example:

```
{make the FontMarked a little larger}
KSCalendar1.FontMonth.Size := 10;
```

See Also:
FontDay
FontDayTitle
FontHighLight
FontMonth
FontNav
FontYear
MarkedDays

FontMonth

Type: **TFont**
Default: **Arial, Regular, 8, clBlack**

Description:
Set the font for the month in the title bar. To accomodate a larger font you might have yo adjust the TitleHeight property.

Example:

```
{make the FontMonth a little smaller}
KSCalendar1.FontMonth.Size := 7;
```

See Also:
FontDay
FontDayTitle
FontHighLight
FontNav
FontYear

FontNav

Type: **TFont**
Default: **Arial, Regular, 8, clBlack**

Description:
Set the font for the day text in the navigation days. For this property to have any meaning you must set ShowNavDays to True.

Example:

```
{make the FontMonth a little smaller}
KSCalendar1.FontMonth.Size := 7;
```

See Also:
FontDay
FontDayTitle
FontHighLight
FontMonth
FontYear
ShowNavDays

FontYear

Type: **TFont**
Default: **Arial, Regular, 8, clBlack**

Description:
Set the font for the year in the title bar. To accomodate a larger font you might have yo adjust the TitleHeight property.

Example:

```
{make the FontYear a lot larger
KSCalendar1.FontYear.Size := 18;
```

See Also:
FontDay
FontDayTitle
FontHighLight
FontMonth
FontNav

ForeColor

Type: **TColor**
Default: **clBlack**

Description:
This property is used when the calendar is in 2D mode (Ctl3D = False). It reflects the color of the grid lines separating the day cells.

Example:

```
{set the ForeColor to navy blue}
KSCalendar1.ForeColor := clNavy;
```

See Also:
BackColor
CalendarColor

Gap

Type: **Integer**
Default: **2**

Description:
This property assists in the internal spacing of the calendars parts. Try experimenting with this value to see the results. It is used to set the amount of space between adjoining regions on the calendar (like the TitleBar and the DayRowBar).

Example:

```
{set the Gap to 5 pixels}
KSCalendar1.Gap := 5;
```

See Also:
Margin

GetDayCellRect

Declaration: function GetDayCellRect(ADay: Integer): TRect;

Description:
Use this powerful method to obtain the rectangle bounding a particular day. If the day you pass in cannot be found on the currently displayed month this function will return a TRect with 0 for all its values. Using this function in conjunction with the calendar's published Canvas property allows users to custom paint into the day cells.

Note: If you are going to custom paint on the calendar we recommend setting the ReadOnly property to True. This will prevent the calendar's painting of the highlighted day from interfering. In this way you can present a "month view" to the user with your custom paint work. If you do not want to make the calendar read only then you can store the previously selected day and repaint it as well as the currently selected day every time the user selects a new day.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
GetDayForXY
Owner Draw Information

GetDayForXY

Declaration: function GetDayForXY(X,Y: Integer): Integer;

Description:

This function will return the calendar day number at a specific X,Y location (relative to the top left corner of the calendar. This is very useful when working with owner draw (see Style property) calendars. Pass in the coordinates of a mouse click and this function will tell you which day was clicked on. If the click is not on a valid day this function will return a 0.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
GetDayCellRect
Owner Draw Information
Style

GlyphFunction1

Type: **TBitmap**
Default: **(None)**

Description:
The bitmap that will display on the first function button (on left) Please not that this bitmap (glyphs) is not sizable. It must match up with the size of the control bar (see ControlBarHeight) to look its best.

Example:

```
{load a newbitmap into GlyphFunction1}
KSCalendar1.GlyphFuntion1.LoadFromFile('c:\mybitmap.bmp');
```

See Also:
ControlBarStyle
GlyphFunction2

GlyphFunction2

Type: **TBitmap**
Default: **(None)**

Description:
The bitmap that will display on the second function button (on right) Please not that this bitmap (glyphs) is not sizable. It must match up with the size of the control bar (see ControlBarHeight) to look its best.

Example:

```
{load a newbitmap into GlyphFunction2}
KSCalendar1.GlyphFuntion2.LoadFromFile('c:\mybitmap.bmp');
```

See Also:
ControlBarStyle
GlyphFunction1

GlyphLastMonth

Type: **TBitmap**
Default: **(None)**

Description:
The bitmap that will display on the first control bar button (on far left) Please not that this bitmap (glyphs) is not sizable. It must match up with the size of the control bar (see ControlBarHeight) to look its best.

Example:

```
{load a newbitmap into GlyphLastMonth}
KSCalendar1.GlyphLastMonth.LoadFromFile('c:\mybitmap.bmp');
```

See Also:
ControlBarStyle
GlyphLastYear
GlyphNextMonth
GlyphNextYear

GlyphLastYear

Type: **TBitmap**
Default: **(None)**

Description:
The bitmap that will display on the second control bar button (second from far left) Please not that this bitmap (glyphs) is not sizable. It must match up with the size of the control bar (see ControlBarHeight) to look its best.

Example:

```
{load a newbitmap into GlyphLastYear}
KSCalendar1.GlyphLastYear.LoadFromFile('c:\mybitmap.bmp');
```

See Also:
ControlBarStyle
GlyphLastMonth
GlyphNextMonth
GlyphNextYear

GlyphNextMonth

Type: **TBitmap**
Default: **(None)**

Description:
The bitmap that will display on the third control bar button (third from far left) Please not that this bitmap (glyphs) is not sizable. It must match up with the size of the control bar (see ControlBarHeight) to look its best.

Example:

```
{load a newbitmap into GlyphNextMonth}
KSCalendar1.GlyphNextMonth.LoadFromFile('c:\mybitmap.bmp');
```

See Also:
ControlBarStyle
GlyphLastMonth
GlyphLastYear
GlyphNextYear

GlyphNextYear

Type: **TBitmap**
Default: **(None)**

Description:
The bitmap that will display on the third control bar button (fourth from far left) Please not that this bitmap (glyphs) is not sizable. It must match up with the size of the control bar (see ControlBarHeight) to look its best.

Example:

```
{load a newbitmap into GlyphNextYear}
KSCalendar1.GlyphNextYear.LoadFromFile('c:\mybitmap.bmp');
```

See Also:
ControlBarStyle
GlyphLastMonth
GlyphLastYear
GlyphNextMonth

GotoNavClick

Type: **Boolean**
Default: **True**

Description:
If GotoNavClick is True the calendar will select the exact date that was clicked on when the user clicks on a navigation day. For this property to have any effect both ShowNavDays and UseNavigationDays must be True.

Example:

```
{show the navigation days but only navigate to the month clicked on}}
KSCalendar1.ShowNavDays := True;
KSCalendar1.UseNavigationDays := True;
KSCalendar1.GotoNavClick := False;
```

See Also:
ShowNavDays
UseNavigationDays

HighlightBackColor

Type: **TColor**
Default: **clBlue**

Description:
This is the color that appears behind the currently selected (highlighted) day. Combine this property with FontHighLight to customize the appearance of the selected date.

Example:

```
{change the HighlightBackColor to red}
KSCalendar1.HighlightBackColor := clRed;
```

See Also:
BackColor
NavBackColor

How Do I Purchase The KSCalendar Date Kit v2.0?

What Do I Get?

Both Delphi 1.0 (16bit) and Delphi 2.0 (32 bit) versions are included!
You get the KSCalendar v2.0 and 11 add on components that work seamlessly with the KSCalendar All source code for the 12 components is included! This help file is also included along with the source code for the Demo project demonstrating how to use the KSCalendar. You also get lifetime technical support via Compuserve and a warm, fuzzy feeling from contributing to the growth of the Delphi 3rd party tools marketplace.

Price: $75.00

There are three ways to purchase KSCalendar:

Option 1 SWREG

KSCalendar Date Kit v2.0
On Compuserve 'GO SWREG' and register shareware **#10901.** This results in Compuserve charging your account $75.00 and Compuserve then sends us your order. Orders are usually filled the same day we get them from Compuserve.

Option 2 ZAC Delphi Only Tools Catalog

KSCalendar Date Kit v2.0
Call ZAC at 1-800-GO-DELPHI (1-800-463-3574) and order using Visa, MasterCard or American Express. M-F 9AM to 8PM. SAT 10AM to 5PM (All EST).

Option 3 Direct

KSCalendar Date Kit v2.0
Print the following information on a sheet of paper:

Product Name (KSCalendar Date Kit v2.0)
Your Name
Company
Your Job Title
Your E-Mail Address (please indicate if you want KSCalendar Date Kit sent to you on-line)
Street Address
City, State
Zip
Daytime & Evening Phone Number

Send the above information and the exact amount of $75.00 to the address below.
If you are located in the state of Tennesee please add 8.5% sales tax to the total.

Kinetic Software
8404 Swathmore Court
Knoxville, TN
37919

Acceptable forms of payment are:
- Personal or business check drawn on a US Bank in US funds
- Any type of money order (domestic or international) we can easily redeem in US funds

Make check or money order payable to Kinetic Software.

How do I install the KSCalendar Date Kit v2.0?

Which Palette?
The KSCalendar Date Kit comes with a published source file REGCAL.PAS, that will help you register the KSCalendar and its associated components and place them on the Delphi component palette of your choice.

By default the KSCalendar Date Kit will install onto a palette called "Kinetic" and Delphi will create this palette if it does not exist. If you would like to install the KSCalendar Date Kit on a different palette simply edit the file REGCAL.PAS. Simply replace all references to "kinetic" with the name of the palette you would like to install to.

Installation Instructions

Delphi 1.0
1. Make sure the KSCalendar Date Kit files are all in one directory.
2. From the Delphi menu select Options | Install Components...
3. Make sure the search path includes the path to the KSCalendar Date Kit files.
4. Press the Add button and select the path and filename of REGCAL.PAS (use Browse)
5. Back on the Install Components dialog press OK.

Delphi 2.0
1. Make sure the KSCalendar Date Kit files are all in one directory.
2. From the Delphi menu select Component | Install...
3. Make sure the search path includes the path to the KSCalendar Date Kit files.
4. Press the Add button and select the path and filename of REGCAL.PAS (use Browse)
5. Back on the Install Components dialog press OK.

If installation succeeds you should see the KSCalendar and its 11 associated components appear on the palette! If you have any difficulty installing the KSCalendar Date Kit refer to the Delphi Help system.

How do I install this help file into the Delphi help system?

This help file can be fully integrated into Delphi's extensible help system. Integrating this help file will allow you to access context sensitive help when using the KSCalendar Date Kit in the Delphi development environment.

Instructions

1. Make sure Delphi is not running.
2. Move the KSCAL.KWF file supplied with the KSCalendar to your \DELPHI\HELP sub-directory.
3. Move the KSCAL.HLP file supplied with the KSCalendar to your \DELPHI\BIN subdirectory.
4. Run the HelpInst application supplied with Delphi.
5. Select File|Open and open \DELPHI\BIN\DELPHI.HDX.
6. Select Keyword|Add Keyword File and select the \DELPHI\HELP\KSCAL.KWF then click OK.
7. Select File|Save.
8. Exit HelpInst.

If the KSCAL.HLP file installed correctly you should be able to click on F1 when the KSCalendar or any of the add on components are selected in the Delphi development environment and jump right into the KSCAL.HLP file. You can also jump to any property's help page by selecting a property in the Object Inspector and pressing F1.

How do I link all the other components to the KSCalendar?

The KSCalendar is made to work as seamlessly as possible with its add-on components.

Add-on components have the Calendar property which connects each add-on to any KSCalendar you choose on the same form.

In most cases the Calendar property is set automatically during the creation and placement of components durin design time. You can verify this by trying a simple experiment:

Place a KSCalendar v2.0 component on a blank form and then place a KSDaySpin component beside it. If you check the Calendar property of the KSDaySpin you will see it is already set to KSCalendar1. If you run the program you will see that the KSDaySpin can be immediately used to control the KSCalendar...without writing any code!!

In fact if you place an add-on component like the KSDateFlip on a blank form and then place a KSCalendar on the same form...you will see that the same automatic connection still takes place!

Many of the add-on components work in this manner. The KSEasyCal, KSMonthView and KSDropDown do not require a connection to the KSCalendar and therefore do not have a Calendar property.

International Support

To achieve internationalization all you have to do is enter the month names and day names for the target language as well as control bar hints (if you are using the control bar). Theoretically the KSCalendar can support any language that uses alphanumeric numbers (1,2,3 etc).

The KSCalendar achieves full international support by publishing all the strings it uses internally as TStringList properties. These properties are as follows:

Property	Required Number of Strings
ControlBarHintText	**6**
DayTitles	**7**
MonthNames	**12**

You can modify these properties at design time (via the TStringList property editor) or at run time (see the source file DEMO.PAS for an example). If you enter more than the required number of strings the KSCalendar will ignore the extra strings.

By default the KSCalendar is English.

IsLeapYear

Declaration: function IsLeapYear(AYear: Integer): Boolean;

Description:

Use this method to determine if a given year is a leap year. Pass in the year and the method will return true if the year is a leap year and false if it is not.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DaysInYear
Year

KSCalendar

The KSCalendar v2.0 is the heart of the KSCalendar Date Kit.
The following subjects are available for the KSCalendar:

KSControlBar

To link the KSControlBar to the KSCalendar see the topic: How do I link all the other components to the KSCalendar?

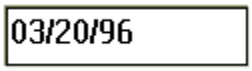
You can use the KSControlBar if you want to implement quick and easy control over your KSCalendar and you don't want to use the optional control bar on the KSCalendar itself.

Use the VisibleButtons property (and its sub-properties) to determine which control buttons are active on the control bar.

Use the ControlBarHintText property to define the hint text for each of the control buttons. Regardless of how many control buttons you have visible you must supply exactly five items in the TStringList property!

The Hint property is not used by this component.

KSDateEdit



To link the KSDateEdit to the KSCalendar see the topic: How do I link all the other components to the KSCalendar?

The KSDateEdit is a simple edit component that can be connected to the KSCalendar to display date information automatically.

The KSDateEdit will display the DateText property of the KSCalendar. For more information on what types of information can be displayed see the following:

DateText
DateFormatString

KSDateFlip

To link the KSDateFlip to the KSCalendar see the topic: How do I link all the other components to the KSCalendar?

The KSDateFlip is a useful little date display component.

To use the KSDateFlip on its own just set its Date (TDateTime) property.

Use the FontDay, FontMonth and FontYear properties to set the font style for each of the three sections.

Use the DayRectHeight, MonthRectHeight and YearRectHeight properties to specify how the KSDateFlip is divided into sections.

Use the MonthLetters property to set how many characters of the month name are displayed in each month cell.

Use the MonthNames property to set the name of the 12 months. This property is useful for International Support.

KSDateLabel

03/12/96

To link the KSDateLabel to the KSCalendar see the topic: How do I link all the other components to the KSCalendar?

The KSDateLabel is a simple label component that can be connected to the KSCalendar to display date information automatically.

The KSDateLabel will display the DateText property of the KSCalendar. For more information on what types of information can be displayed see the following:

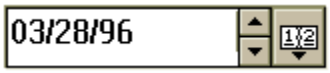
DateText
DateFormatString

KSDaySpin / KSMonthSpin / KSYearSpin

To link the KSDaySpin, KSMonthSpin or KSYearSpin to the KSCalendar see the topic: How do I link all the other components to the KSCalendar?

These three spin buttons allow quick and easy spin button access to the KSCalendar's optimized date switching capabilities.

KSDropDown



The KSDropDown component makes use of the KSCalendar to provide powerful drop down calendar capability. Users can select dates using the spin buttons or by clicking on the drop down button and selecting a date from the KSCalendar.

Use the DropWidth property to set the width of the drop down button and the SpinWidth property to set the width of the spin button.

Since the KSDropDown is derived from TCustomMaskEdit its EditMask property helps to format and restrict user input of dates.

KSEasyCal

Refer to the KSCalendar help for help with KSEasyCal.

While the KSEasyCal seems to "look and feel" like a regular KSCalendar it is actually quite different. The KSEasyCal is provided for those instances when you don't need or want all the full blown power of the KSCalendar in your application.

We have removed some functionality from the KSCalendar to make it smaller (for smaller apps) and faster. KSEasyCal maintains much of KSCalendar's main functionality even though it is about 25K smaller in size!

What's Missing:
Data aware functionality
Owner drawn cells
Navigation days
Auto sized control bar
Advanced control bar style
Control bar position control
ISO Week number support
Day number support
Days remaining in month, year
Day row border
Some date calculation methods

KSMonthBar

The KSMonthBar is a control bar that allows users to quickly and easily select a calendar month.

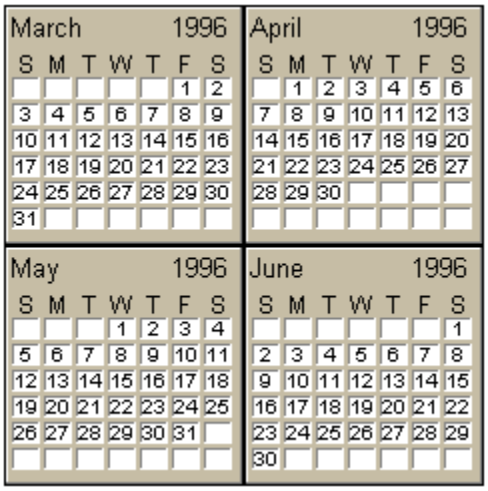
Use the GridCellsAcross property to control the layout grid dimensions. If you set the GridCellsAcross to 4 you'd get a KSMonthBar that displays the months in 4 columns and 3 rows.

Use the MonthLetters property to set how many characters of the month name are displayed in each month cell.

Use the MonthNames property to set the name of the 12 months. This property is useful for International Support.

To link the KSMonthBar to the KSCalendar see the topic: How do I link all the other components to the KSCalendar?

KSMonthView



The KSMonthView component allows you to display multiple months (1 to 12) at one time. A common use for this component is to show a "year at a glance".

Use the GridCellsAcross property to determine how many columns the KSMonthView will have. Valid values for this property are 1,2,3,4,6 and 12.

Use the Months property to select the number of months to display. This property is tightly integrated with the GridCellsAcross property and its possible values depend on the current value of GridCellsAcross.

Use the StartMonth property to select the first month displayed....this property along with the Months property determines the range of months that will be displayed.

Keyboard Control

The KSCalendar can be fully controlled from the keyboard:

Key	Action
Page Up	Last Year
Page Down	Next Year
Up Arrow	Last Month
Down Arrow	Next Month
Left Arrow	Last Day
Right Arrow	Next Day

These key presses actually end up calling the corresponding KSCalendar public methods (LastYear, NextYear etc.)

If you purchased KSCalendar Pro and have access to the source code you can easily remap the above keys. Just take a look at the KSCalendar's KeyDown (TKSCalendar.KeyDown) procedure.

LastDay

Declaration: **procedure LastDay;**

Description:

Use this method to make the calendar decrement visually to the previous calendar day.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
LastMonth
LastYear
NextDay
NextMonth
NextYear

LastMonth

Declaration: **procedure LastMonth;**

Description:

Use this method to make the calendar decrement visually to the previous calendar month.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
LastDay
LastYear
NextDay
NextMonth
NextYear

LastYear

Declaration: **procedure LastYear;**

Description:

Use this method to make the calendar decrement visually to the previous calendar year.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
LastDay
LastMonth
NextDay
NextMonth
NextYear

Margin

Type: **Integer**
Default: **5**

Description:
This value is the distance between the edge (border) of the calendar and the calendar contents. It applies to all four sides of the calendar and can be used to adjust the position of the internal elements (month text, year text, day cells etc.) in relation to the calendar border.

Example:

```
{increase the Marging to 10}
KSCalendar1.Margin := 10;
```

See Also:
Gap

MarkedDays

Type: **TCaption**
Default: **empty string**

Description:
Use this property to mark specific days of the week. Just enter a string containing the days you want marked and those days will be drawn with the background color BackColorMarked.

Setting the MarkedDays property to '01' would result in Sundays and Mondays being marked. Setting the MarkedDays property to '345' would result in Wednesdays, Thursdays and Fridays being marked.

Example:

```
{mark weekends}
KSCalendar1.MarkedDays := '06';
```

See Also:
BackColorMarked

Methods Reference

Custom Methods:

SetDate
GetDayCellRect
IsLeapYear
DayNumberToDate
DateToDayNumber
DaysInMonth
DaysInYear
DaysRemaining
MonthStartsOn
DateToWeekNumber
WeekNumberToDate
GetDayForXY
DaysBetween
NextDay
LastDay
NextMonth
LastMonth
NextYear
LastYear
Refresh
RefreshDays

Month

Type: **Integer**
Default: **N/A**

Description:
Contains the currently selected calendar month. Values will range from 1 to 12.

Example:

```
{change the calendars Month to December}
KSCalendar1.Month := 12;
```

See Also:
Date
Day
Year

MonthNames

Type: **TStringList**
Default: **January, February, March, April, May, June, July, August, September, October, November, December**

Description:
Use this property to change the text for the month names displayed in the title bar. (see International Support)

Example:

```
{set the MonthNames for a French calendar}
KSCalendar1.MonthNames.Strings[0]:='Janvier';
KSCalendar1.MonthNames.Strings[1]:='Février';
KSCalendar1.MonthNames.Strings[2]:='Mars';
KSCalendar1.MonthNames.Strings[3]:='Avril';
KSCalendar1.MonthNames.Strings[4]:='Mai';
KSCalendar1.MonthNames.Strings[5]:='Juin';
KSCalendar1.MonthNames.Strings[6]:='Julliet';
KSCalendar1.MonthNames.Strings[7]:='Août';
KSCalendar1.MonthNames.Strings[8]:='Septembre';
KSCalendar1.MonthNames.Strings[9]:='Octobre';
KSCalendar1.MonthNames.Strings[10]:='Novembre';
KSCalendar1.MonthNames.Strings[11]:='Décembre';
```

See Also:
DayTitles
International Support

MonthStartsOn

Declaration: function MonthStartsOn(AYear,AMonth: Integer) : Integer;

Description:

This is an odd function (used internally so we published it...) that will tell you what day of the week a month starts on. The integer return value can be interpreted as: 0=Sunday, 1=Monday, ... , Saturday=6.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
DaysInMonth
Weekday

NavBackColor

Type: **TColor**
Default: **clSilver**

Description:
This color is the color that appears behind the day text of the navigation days (those days displayed that are in the previous or next month).

Example:

```
{set the NavBackColor to red}
KSCalendar1.NavBackColor := clRed;
```

See Also:
ShowNavDays
UseNavigationDays
BackColor
CalendarColor
ThreeDLightColor
ThreeDShadowColor

NextDay

Declaration: **procedure NextDay;**

Description:

Use this method to make the calendar increment visually to the next calendar day.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
LastDay
LastMonth
LastYear
NextMonth
NextYear

NextMonth

Declaration: **procedure NextMonth;**

Description:

Use this method to make the calendar increment visually to the next calendar month.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
LastDay
LastMonth
LastYear
NextDay
NextYear

NextYear

Declaration: **procedure NextYear;**

Description:

Use this method to make the calendar increment visually to the next calendar year.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
LastDay
LastMonth
LastYear
NextDay
NextMonth

OnDateChange

Type: **TNotifyEvent**

Description:
This event is triggered whenever the day, month or year of the calendar are changed. Use this event to track the users activities and to check for potential problems the user might encounter.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
Day
Month
Year

OnDrawCell

Type: procedure(Control: TWinControl; Day: Integer; Rect: TRect; Coord: TPoint; State: Boolean) of object;

Description:
This event is generated if you have set the Style property to csOwnerDraw. Custom drawing of the calendar should be done during the processing of this event. The KSCalendar will give you a great deal of assistance in your painting process. Day contains the day number that needs to be painted. Rect will contain the day cell rectangle that your painting will occur within. Coord will contain the row and column of the cell and State will contain whether the day is currently selected (True if selected / False if not selected).

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
Style

OnFunction1Click

Type: **TNotifyEvent**

Description:
This custom event is triggered when the left function button is pressed. For more information see the property ControlBarStyle

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
OnFunction2Click

OnFunction2Click

Type: **TNotifyEvent**

Description:
This custom event is triggered when the right function button is pressed. For more information see the property ControlBarStyle

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
OnFunction1Click

Owner Draw Information

The KSCalendar uses a powerful Owner Draw paradigm (similar to Delphi's owner draw list boxes) to give developer's almost unlimited freedom in designing their applications. To make the KSCalendar owner drawn you must do two things:

1. Set the Style property to csOwnerDraw (the default is csStandard).
2. Supply painting logic in the OnDrawCell event.

Once you have done this the KSCalendar will generate OnDrawCell events for each day cell it requires painted. You become responsible for painting everything within the day cell, including the actual day number (if you want it there).

The KSCalendar provides a great deal of assistance to you. Lets look at the header for OnDrawCell event handler:

```
procedure TForm1.KSCalendarDrawCell(Control: TWinControl; Day: Integer; Rect: TRect; Coord: TPoint; State: Boolean);
```

The parameters passed to the event handler should provide all the information you require to accomplish your painting:

Parameter	**Description**
Control	The handle to the KSCalendar
Day	The day number (1 to 31) that is being painted
Rect	The rectangle bounding the day cell
Coord	The column (X) and the row (Y) of the cell being painted
State	The state of the cell being painted (True=selected, False=unselected).

For more information on how to build an owner draw KSCalendar refer to the source code of the DEMO project supplied with KSCalendar!

Properties Reference

Any KSCalendar properties you can't find below are standard Delphi properties that can be looked up in the Delphi help system. If you have any properties or features you would like to see please let us know!

Custom Properties:

ThreeDShadowColor
TitleHeight
TitleStyle
UseNavigationDays
Weekday
WeekNumber
WeekStart
Year

Public Properties

The following public properties facilitate access to the buttons (TSpeedButtons) on the control bar:

```
property   BLastMonth: TSpeedButton read FLMButton write FLMButton;
property   BLastYear: TSpeedButton read FLYButton write FLYButton;
property   BNextMonth: TSpeedButton read FNMButton write FNMButton;
property   BNextYear: TSpeedButton read FNYButton write FNYButton;
property   BFunction1: TSpeedButton read FFunc1 write FFunc1;
property   BFunction2: TSpeedButton read FFunc2 write FFunc2;
```

For example, to change the number of glyphs (NumGlyphs) property of the Function1 and Function2 buttons you would use the following syntax:

```
KSCalendar1.BFunction1.NumGlyphs: = 2;
KSCalendar1.BFunction2.NumGlyphs: = 2;
```

Other speedbutton properties like Visible, Caption or Font can be set in a similar manner. For additional information on TSpeedButton refer to the Delphi help system.

ReadOnly

Type: **Boolean**
Default: **False**

Description:
Set this to True to turn off mouse input over the day cells, and the left and right arrow key input. The control bar still operates as do the page up, page down, up arrow and down arrow keys (see Keyboard Control). This is normally used to present users with a "month view" upon which custom paint work has been done (see GetDayCellRect). Setting this property to True will also disable the highlighting of a selected day.

Example:

```
{set the calendar to ReadOnly mode}
KSCalendar1.ReadOnly := True;
```

Refresh

Declaration: **procedure Refresh;**

Description:

Call this method to get the KSCalendar to completely redraw itself.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
RefreshDays

RefreshDays

Declaration: **procedure RefreshDays;**

Description:

Call this method to get the KSCalendar to redraw only the day cells. This can be useful to reduce flicker by not redrawing the entire calendar when only the days need to be updated. Very useful when the calendar is in OwnerDraw mode (see Owner Draw Information)

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
Refresh

SetDate

Declaration: **procedure SetDate(NewDate: TDateTime);**

Description:
Use this method to change the currently selected date of the calendar. This method will result in a fairly optimized repaint of only those parts of the calendar that are effected. This is the only way to change the calendar's date from code! (the Date property is read only)

You can use the KSCalendar to parse dates. Use the SetDate method to set the appropriate date and then read off the values you are interested in from the properties Day, Month, Year, DayNumber, WeekNumber, Weekday. etc.

Example:

For an example of how to implement this event please refer to the Demo project.

See Also:
Day
Month
Year
DayNumber
WeekNumber
Weekday

ShowDayRow

Type: **Boolean**
Default: **True**

Description:
Turn the display of theday row on or off

Example:

```
{set ShowDayRow off}
KSCalendar1.ShowDayRow := False;
```

See Also:
DayTitles
ShowDayRowBorder

ShowDayRowBorder

Type: **Boolean**
Default: **False**

Description:
Turn the display of the border around the day row on or off

Example:

```
{set ShowDayRowBorder on}
KSCalendar1.ShowDayRowBorder := True;
```

See Also:
DayTitles
ShowDayRow

ShowNavDays

Type: **Boolean**
Default: **True**

Description:
This property controls whether or not the navigation days are visible. If they are visible you have the option of using the navigation days to jump to the the previous or next month.
Example:

```
{hide the navigation days}
KSCalendar1.ShowNavDays := False
```

See Also:
NavBackColor
UseNavigationDays
CalendarColor
ThreeDLightColor
ThreeDShadowColor

Style

Type: **Enumerated (csStandard, csOwnerDraw)**
Default: **csStandard**

Description:
Set the style property to csOwnerDraw to obtain complete control of how the KSCalendar looks! Setting the style to csOwnerDraw will result in the KSCalendar generating a OnDrawCell event everytime it needs a day cell painted. You can use this method to do some very advanced custom painting. See Owner Draw Information.

Example:

```
{set the calendar to OwnerDraw mode}
KSCalendar1.Style := csOwnerDraw;
```

See Also:
Owner Draw information

ThreeDLightColor

Type: **TColor**
Default: **clWhite**

Description:
Use this color property to select the light color for 3D effects on the calendar. Most of the time you would only change this color if you have changed the surface color of the calendar itself (see CalendarColor).

Example:

```
{change the ThreeDLightColor to yellow}
KSCalendar1.ThreeDLightColor := clYellow;
```

See Also:
ThreeDShadowColor
CalendarColor

ThreeDShadowColor

Type: **TColor**
Default: **clGray**

Description:
Use this color property to select the shadow color for 3D effects on the calendar. Most of the time you would only change this color if you have changed the surface color of the calendar itself (see CalendarColor).

Example:

```
{change the ThreeDShadowColor to yellow}
KSCalendar1.ThreeDShadowColor := clYellow;
```

See Also:
ThreeDLightColor
CalendarColor

TitleHeight

Type: **Integer**
Default: **27**

Description:
This is the height of the area of the rectangle that contains the month and year text. You can increase this value if you plan to use a larger font for the either the month or year.

Example:

```
{increase the TitleHeight to 35 pixels}
KSCalendar1.Titleheight := 35;
```

See Also:
Margin
TitleStyle

TitleStyle

Type: **Enumerated (tsNone, tsMonth, tsYear, tsMonthYear, tsFormatted)**
Default: **tsMonthYear**

Description:
Use this property to alter the style of the title bar (at top of calendar). Display month, year, both or display the DateText property as the title! Note: When you set this style to tsFormatted the font that is used to display the title bar is FontMonth.

Example:

```
{set the calendar to display a formatted, centered title}
KSCalendar1.TitleStyle := tsFormatted;
```

See Also:
DateFormatString
DateText
TitleHeight

UseNavigationDays

Type: **Boolean**
Default: **True**

Description:
This property controls whether or not the navigation days can be used to move to the previous or next month. For this property to have any meaning ShowNavDays must be set to True.

Example:

```
{show the navigation days but turn off their navigation use}
KSCalendar1.ShowNavDays := True;
KSCalendar1.UseNavigationDays := False;
```

See Also:
NavBackColor
ShowNavDays
CalendarColor
ThreeDLightColor
ThreeDShadowColor

WeekNumber

Type: **Integer (Read Only)**
Default: **N/A**

Description:
Contains the week number that the currently selected date falls into. According to the International Standards Organization:
1. A week is a period of seven days
2. The first day of the week is a Monday
3. Week number 1 is the first week of the year with 4 or more days

If you use this value a lot you might want to consider setting your WeekStart property to 1 (Monday). This definition implies that you could have 53 weeks in a year and January 1,1995 is not necessarily in week 1.

Example:

```
{read the WeekNumber of the currently selected date}
var
  MyWeek: Integer;
begin}
  MyWeek: := KSCalendar1.WeekNumber;
```

See Also:
DateToWeekNumber
WeekNumberToDate

WeekNumberToDate

Declaration: **function WeekNumberToDate(AYear,AWeek,Day: Integer; var ADate: TDateTime): Boolean;**

Description:

Use this function to calculate the day for a given week and weekday in a given year. Pass in the year via AYear, the week number via AWeek and the day of the week you are interested in via Day (remember 0=Sunday, 1=Monday ... 6=Saturday!). Pass in the Date via ADate (the function will modify the date variable you pass in).
This function will return True if it made a successful calculation or False if it failed.

This function will fail (return False) when the day you requested does not exist in the week you specified. This will generally occur during week 1 or week 52-53.

Example:

```
{if you wanted to calculate the date for the Wednesday of week 32 in 1995 you would use the following syntax}

if (KSCalendar1.WeekNumberToDate(1995,32,3,MyDateVar) then
   {success ... the date is stored in MyDateVar
else
   {failed ... the date stored in MyDateVar is meaningless (0/0/00)
```

See Also:
DateToWeekNumber
WeekNumber

WeekStart

Type: **Integer**
Default: **0**

Description:
Specifies the starting day of the week. The integer value of this property can be interpreted as follows: Sunday = 0, Monday=1, Tuesday=2 ... Saturday=6.

Example:

```
{set the calendar to start on Monday}
KSCalendar1.WeekStart := 1;
```

See Also:
Weekday

Weekday

Type: **Integer (Read Only)**
Default: **N/A**

Description:
Contains the weekday (Sunday, Monday ...etc.) of the currently selected date. The integer value of this property can be interpreted as follows: Sunday = 0, Monday=1, Tuesday=2 ... Saturday=6.

Example:

```
{read the currently selected Weekday
var
   MyDay: Integer;
begin}
   MyDay := KSCalendar1.Weekday;
```

See Also:
Day
WeekStart

What new in version 2.0?

What's new with the KSCalendar:
The KSCalendar has been reworked extensively while retaining full backward compatibility with v1.0. The following capabilities have been added:

1. Navigation days have been added that display days in the previous or following month. These days can be used to navigate from one month to another.
New Properties: FontNav, GotoNavClick, ShowNavDays, UseNavigationDays.

2. Marked days have been added so that you can mark specific days of the week
New Properties: FontMarked, BackColorMarked, MarkedDays

3. The KSCalendar is now data aware.

11 New Components:
The KSCalendar now comes with 11 add on components that work closely with the KSCalendar to provide a full set of development tools. All of these controls are extremely easy to use! In most cases just drop the control on a form that has a KSCalendar and they start to work immediately!

KSDropDown	**Powerful drop down calendar and edit control**
KSDateEdit	**Edit box linked to the KSCalendar**
KSDateLabel	**Label linked to the KSCalendar**
KSDaySpin	**Spin button that automatically controls days**
KSMonthSpin	**Spin button that automatically controls months**
KSYearSpin	**Spin button that automatically controls years**
KSControlBar	**A control bar similar to the one found on the KSCalendar**
KSMonthBar	**A control bar that makes selecting months a snap**
KSDateFlip	**A nifty little graphical control that display dates**
KSMonthView	**View multiple months at a time**
KSEasyCal	**Smaller, faster calendar with less features**

Year

Type: **Integer**
Default: **N/A**

Description:
Contains the currently selected calendar year. Values will range from 1 to **?.**

Example:

```
{read the currently selected Year}
var
   MyYear: Integer;
begin}
   MyYear: := KSCalendar1.Year;
```

See Also:
Day
Month
DaysInYear