CalendarWorks

Version 1.01

Calendar Component for Delphi

Manual

Installation

Installing The Shareware Version of CalendarWorks

The files included in the demo version of CalendarWorks are:

CWBASIC.DCU	- Compiled CalendarWorks Basic Calendar Component
CWBASIC.DCR	- Basic Calendar Component resource file
CWORKS.DOC	- This documentation
DEMO.EXE	- CalendarWorks demo program

To run the demo just execute DEMO.EXE.

To install the CalendarWorks Basic Calendar component:

Copy CWBASIC.DCU and CWBASIC.DCR to \DELPHI\LIB
Run Delphi
Select "Options|Install Components"
Select "Add" then type "CWBASIC"
Select OK

Installing The Registered Version of CalendarWorks

The files included in the registered version of CalendarWorks are:

CWBASIC.DCU	- Compiled CalendarWorks Basic Calendar Component
CWBASIC.DCR	- Basic Calendar Component resource file
CWCAL.DCU	- Full Featured CalendarWorks Calendar
CWCAL.DCR	- CalendarWorks Calendar resource file
CWTOOLS.DCU	- CalendarWorks additional tool components
CWTOOLS.DCR	- CalendarWorks additional tools resource file
CWSMCAL.DCU	- Calendar used by CWDropDownEdit
CWSMCAL.DFM	- Form for calendar used by CWDropDownEdit
CWQKCAL.DCU	- Quick Calendar dialog / CWQuickCalendar
CWQKCAL.DFM	- Form used by CWQuickCalendar
CWORKS1.RES	- Resource file used by CWCAL
CWORKS2.RES	- Resource file used by CWTOOLS
CWBASIC.PAS	- Source code for CWBASIC
CWCAL.PAS	- Source code for CWCAL
CWTOOLS.PAS	- Source code for CWTOOLS
CWSMCAL.PAS	- Source code for SWSMCAL
CWQKCAL.PAS	- Source code for CWQKCAL
CWORKS.DOC	- Manual

Installing The Registered Version of CalendarWorks *continued*

To install all CalendarWorks components:

Copy all *DCU, *.DCR and *.RES files to \DELPHI\LIB
Run Delphi
Select "Options|Install Components"
Select "Add" then type "CWBASIC"
Select "Add" then type "CWCAL"
Select "Add" then type "CWTOOLS"
Select OK

When finished there will be a new section called CalendarWorks added to the VCL toolbar.

Version Information

Version 1.01 - May '96
Added the following functions and procedures

```
function Bitmap(ADay:integer):string
function TextCount(ADay:integer):integer;
function  Text(ADay:integer;Index:integer):string;
procedure DeleteBitmap(ADay:integer);
procedure DeleteText(ADay,Index:integer);
```

Version 1.00 - May '96
No known bugs at this time.

If users are having difficulties please let me know by contacting me at **hsw@sympatico.ca**

Registration

Getting a Registered Copy of CalendarWorks

The demo version of CalendarWorks comes with the fully functional Basic Calendar component. To obtain the other components, including the highly powerful, full featured CalendarWorks Calendar, you must download the cworks16.zip or cworks32.zip file.

The Calendarworks components are free but you can register the components if you wish. With registration you will receive the source code and the knownledge that you are contributing to the development of this set of components.

Cost to register CalendarWorks is $34,95.

Ordering CalendarWorks Through the Mail

If ordering by mail, please send $34.95 in US funds to HSoftWare. If ordering within the US or Canada include $3.00 in US funds for shipping and handling. If ordering from outside the US and Canada, include $5.00 in US funds for shipping and handling. You can save yourself the shipping and handling charge if you request the registered version sent via e-mail to a Compuserve or Internet address. When ordering within Canada, add GST and when ordering within Ontario add GST and PST.

Please send cheque or money order to:

HSoftWare
Howard Walsh
45 Milfoil Crescent
Kitchener, Ontario N2E 3L2

Ordering CalendarWorks with a Credit Card

Credit card orders are also accepted through PsL. (Public software Library) - a credit card processing service only.

To order by MasterCard, Visa, American Express, or Discover, call the Public (software) Library at 1-800-2424-PsL or 713-524-6394 or send your order by FAX to 1-713-524-6398 or by CompuServe E-Mail to 71355,470 or Internet mail to 71355.470@compuserve.com. You can also mail credit card orders to PsL at P.O. Box 35705, Houston, TX 77235-5705.

When ordering please specify the product, CalendarWorks, and the product number, **14629**.

HSoftWare cannot be reached at the numbers above.
These numbers are for PsL, a credit care order taking service ONLY!

Please be sure to include your credit card number and expiration date on all credit card orders.

Ordering CalendarWorks with a Credit Card *continued*

Any questions about the status of the shipment or an order, refunds, registration options, product details, technical support, volume discounts, dealer pricing, site licenses, etc., must be directed to HSoftWare, 45 Milfoil Crescent, Kitchener, Ontario N2E 3L2, or by e-mail to hsw@sympatico.ca§ on the Internet.

What You Get

With the registered version you will receive all the latest CalendarWorks components, plus the full source code.

This source code may be altered by you as you wish to make CalendarWorks function to your personal needs. There are no run-time royalties when using CalendarWorks, however, you cannot resell CalendarWorks as your own, altered or unaltered in component or other non-executable form.

Technical Support Addresses

Please report any problems you are having so that I may improve CalendarWorks. Also, if you have an idea to make CalendarWorks better, please let me know. I can be reached at the following addresses:

Internet: hsw@sympatico.ca

Warranty

Disclaimer of Warranties

HSoftWare does not warrant that the CalendarWorks component will meet your requirements or that the operation of the component will be uninterrupted or error free. Nor do we offer any Warranties or Guarantees of any kind. You are free to use the CalendarWorks component in any way you see fit, but at your own risk.

HSoftWare is not responsible for any problems or damage caused by the CalendarWorks component or that may result from using the CalendarWorks component, whether licensed or not. This includes, but is not limited to, computer hardware, computer software, operating systems, and any other computer or computing accessories. Should the CalendarWorks component prove defective, the end user assumes the risk of paying the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will HSoftWare be liable for any damages whatsoever, including without limitation damages for loss of business profits, business interruption, loss of business information and the like, arising out of the use or the inability to use the CalendarWorks component, even if HSoftWare has been advised of the possibility of such damages. The end user agrees to hold HSoftWare and/or any persons associated with the creation of this component harmless for any problems arising from the use of the component.

Use of this product for any period of time constitutes your acceptance of this agreement.

HSoftWare regrets having to place such a disclaimer. We have made every effort to create a software product that is error free, however, in the creation of any piece of software the possibility exists for unforeseen problems.

AddBitmap method

Applies to
CWCalendar component

Declaration

procedure AddBitmap(ADateTime:ADateTime;ABitmap:string);

The *AddBitmapt* method is used to attach a bitmap, defined as **ABitmap**, to the cell defined by the date passed in **ADateTime**. There are ten pre-defined bitmaps which are stored in then cworks1.res file.

If the current month and year displayed on the calendar does not match the month and year passed in **ADateTime**, no bitmap is added. As such, it is probably best to call this method when an **OnUpdate** event occurs.

This method does not automatically refresh the calendar. If not using with method in conjunction with the **OnUpdate** event, the **RefreshDay** or **RefreshDays** methods should be called.

There are ten pre-defined bitmaps stored in cworks1.res. The *AddBitmap* method is designed to work with bitmaps which are 32 pixels wide by 16 pixels high. If larger or smaller bitmaps are desired, the source will have to be altered to accomadate the different size.

The ten pre-defined bitmaps are:

Bitmap	**Resource Name**
Alarm	CWALARM
Checkmark	CWCHECK
Globe	CWGLOBE
Library	CWLIBRARY
Light Bulb	CWBULBON
Mailbox	CWMAILBOX
Night	CWNIGHT
Pencil	CWPENCIL
Phone	CWPHONE
Todo	CWTODO

Example
The following example adds the text, *Meeting*, and the alarm bitmap to today's date.

```
procedure TForm1.CWCalendar1Update(Sender: TObject);
begin
  CWCalendar1.AddText(Date,clRed,'Meeting');
  CWCalendar1.AddBitmap(Date,'CWAlarm');
  CWCalendar1.RefreshDay(Day);
end;
```

AddText method

Applies to
CWCalendar component

Declaration

procedure AddText(ADateTime:ADateTime;AColor:TColor;AText:string);

The *AddText* method is used to attach a string, **AText**, to the cell defined by the date passed in **ADateTime**. Up to six lines of text, each with a maximum length of 64 bytes, can be added to each cell.

If the current month and year displayed on the calendar does not match the month and year passed in **ADateTime**, no text is added. As such, it is probably best to call this method when an **OnUpdate** event occurs.

The text color is determined by the value of **AColor**.

This method does not automatically refresh the calendar. If not using with method in conjunction with the **OnUpdate** event, the **RefreshDay** or **RefreshDays** methods should be called.

Example
The following example adds the text, *Meeting*, and the alarm bitmap to today's date.

```
procedure TForm1.CWCalendar1Update(Sender: TObject);
begin
  CWCalendar1.AddText(Date,clRed,'Meeting');
  CWCalendar1.AddBitmap(Date,'CWAlarm');
  CWCalendar1.RefreshDay(Day);
end;
```

Bitmap method

Applies to
CWCalendar component

Declaration

function Bitmap(ADay:integer):string

The *Bitmap* method returns the name of the bitmap resource for the cell day defined by the value passed in ADay. If no bitmap is present or ADay is an invalid day, then *Bitmap* returns an empty string.

DayNumber method

Applies to
CWBasicCalendar, CWCalendar components

Declaration

function DayNumber(ADate:TDateTime):integer;

The *DayNumber* method returns a numeric value indicating which day of the year on which the date passed in ADate falls.

Example

This example returns 122, indicating that May 1 is the 122nd day the 1996..

```
EncodeDate(ADate, 1996, 5, 1);
Day := DayNumber(ADate);
```

DaysBetween method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

function DaysBetween(ADate,BDate:TDateTime):longint;

The *DaysBetween* method returns a numeric value indicating which how many days lay between the two dates passed in ADate and BDate. ADate can be greater than, equal to or lesser than ADate.

DaysInMonth method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

function DaysInMonth(AYear,AMonth:integer):integer;

The *Days InMonth* method returns a numeric value indicating the number of days in the month and year passed in AMonth and AYear. The AYear value is required to determine if the number of days in February is 28 or 29.

DaysInYear method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

function DayInYear(AYear:integer):integer;

The *DaysInYear* method returns a numeric value indicating the number of days in the year passed in AYear. The value returned is either 365 or 366 depending of whether the passed year is a leap year.

DaysRemaining method

Applies to
CWBasicCalendar, CWCalendar components

Declaration

function DaysRemaining(Index:integer;ADate:TDateTime):integer;

The *DaysRemaining* method returns a numeric value indicating the number of days remaining in the month or year of the date passed in ADate. When Index is set to 0, *DaysRemaining* returns the number of days remaining in the year. When Index is set to 1, *DaysRemaining* returns the number of days remaining in the month.

Example
This example returns the number of days remaining in the month of May when the passed date is May 12, 1996.

```
EncodeDate(ADate, 1996, 5, 12);
DaysLeft := DaysRemaining(1,ADate);
```

DeleteBitmap method

Applies to
CWCalendar component

Declaration

procedure DeleteBitmap(ADay:integer);

The *DeleteBitmap* method is used to remove a bitmap which is attached to the cell date defined by ADay. If no bitmap is present or ADay is an invalid day, no action is taken.

Example
The following example removes the bitmap from the selected cell with the OnDblClick event.

```
procedure TForm1.CWCalendar1OnDblClick(Sender: TObject);
begin
  CWCalendar1.DeleteBitmap(Day);
  CWCalendar1.RefreshDay(Day);
end;
```

DeleteText method

Applies to
CWCalendar component

Declaration

procedure DeleteText(ADay,Index:integer);

The *DeleteText* method is used to remove the text line specified by Index which is attached to the cell date defined by ADay. If no text is present or ADay is an invalid day, no action is taken.

Example
The following example removes the first line of text from the selected cell with the OnDblClick event.

```
procedure TForm1.CWCalendar1OnDblClick(Sender: TObject);
begin
  CWCalendar1.DeleteText(Day,1);
  CWCalendar1.RefreshDay(Day);
end;
```

FirstOfMonth method

Applies to
CWBasicCalendar, CWCalendar components

Declaration

function FirstOfMonth(AYear,AMonth:integer):integer;

The *FirstOfMonth* method returns a numeric value indicating which day of the week the first of the month passed in AMonth and AYear falls on. *FirstOfMonth* returns 1 for Sunday and 7 for Saturday.

GetDayRect method

Applies to
CWCalendar component

Declaration

function GetDayRect(ADay:integer):TRect;

The *GetDayRect*method returns a **TRect** object defining the cell rectangle for the day of the currently displayed month. If **ADay** is not a valid day, then *GetDayRect* returns Rect(0,0,0,0).

GetDayRect may be used when creating your own user draw calendar.

GetDayXY method

Applies to
CWBasicCalendar, *CWCalendar* components

Declaration

function GetDayXY(x,y:integer):integer;

The *GetDayXY* method returns a numeric value indicating the day of the current month displayed on the calendar for the values of x and y. These values, x and y, are best obtaining by catching the OnMouseDown event.

If the co-ordinates x and y do not fall within a valid cell date then *GetDayXY* returns a value of 0.

Example
This example determines if the left mouse button was pressed while the cursor was on a valid date. If so, then that date is selected.

```
procedure TForm1.CWBasicCalendar1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var ADay:integer;
begin
  if mbLeft in Button then
    begin
    ADay := GetDayXY(x,y);
    if ADay <> 0 then
        SelectDay(ADay);
end;
```

IsLeapYear method

Applies to
CWBasicCalendar, *CWCalendar* components

Declaration

function IsLeapYear(AYear:integer):boolean;

The *IsLeapYear* method returns true if the year passed in AYear is a leap year, otherwise, it returns false.

NextDay method

Applies to
CWBasicCalendar, *CWCalendar* components

Declaration

procedure NextDay;

The *NextDay* method moves the calendars selected day to the next calendar day. If the calendar is on the last day of the month, the calendar moves to the first day of the next month.

NextMonth method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

procedure NextMonth;

The *NextMonth*method updates the displayed calendar to show the following month.

NextYear method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

procedure NextYear;

The *NextYear* method updates the displayed calendar to show the following year.

PrevDay method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

procedure PrevDay;

The *PrevDay* method moves the calendars selected day to the previous calendar day. If the current day is the first of the month, the calendar moves to the last day of the previous month.

PrevMonth method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

procedure PrevMonth;

The *PrevMonth* method updates the displayed calendar to show the previous month.

PrevYear method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

procedure PrevMonth;

The *PrevYear* method updates the displayed calendar to show the previous year.

RefreshDay method

Applies to

CWCalendar component

Declaration

procedure RefreshDay(ADay:integer);

The *RefreshDay* method repaints only the calendar day as defined by **ADay**. If **ADay** is not a valid day for the current month, no cells are refreshed. The calendar header and grid is not repainted.

As calendar cells are not automatically repainted when using **AddCellText** or **AddCellBitmap**, the *RefreshDay* method could be called to repaint just the cell which has been updated.

Example

The following example uses RefreshDay to update the calendar after adding the text, *Meeting*, and the alarm bitmap to today's date.

```
procedure TForm1.CWCalendar1Update(Sender: TObject);
begin
  CWCalendar1.AddCellText(Date,clRed,'Meeting');
  CWCalendar1.AddCellBitmap(Date,'CWAlarm');
  CWCalendar1.RefreshDay(Day);
end;
```

RefreshDays method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

procedure RefreshDays;

The *RefreshDays* method repaints only the calendar day cells. Calendar header and grid is not repainted.

ResetRange method

Applies to
CWCalendar component

Declaration

procedure ResetRange;

The *ResetRange*method sets the **RangeStartDate** and **RangeEndDate** properties to the current calendar date. Calling this method has no affect on the calendar if the **SelectRange** property is set to false.

SelectCell method

Applies to
CWBasicCalendar, CWCalendar components

Declaration

procedure SelectCell(ACol,ARow:integer):integer;

The *SelectCell* method selects the cell determined by the vales passed in ACol and ARow. If the cell is not a valid day, then the cell is not selected.

ACol can be any value from 0 to 6. ARow can be any value from 0 to 5.

This procedure is used internally by CWBasicCalendar and CWCalendar. It may be of limited use to the developer, but it is presented anyway.

Example
This example selects the cell determined by the third column and second row.

```
SelectCell(3,2);
```

SelectCellXY method

Applies to
CWBasicCalendar, CWCalendar components

Declaration

function SelectCellXY(x,y:integer):integer;

The *SelectCellXY* method will select a cell with a valid date and return that day for the co-ordinates passed in x and y. The values, x and y, are best obtaining by catching the OnMouseDown event.

If the co-ordinates x and y do not fall within a valid cell date then *SelectCellXY* returns a value of 0 and the cell is not selected.

Example
This example determines if the left mouse button was pressed while the cursor was on a valid date. If so, then that date is selected. If not, MessageBeep is called.

```
procedure TForm1.CWBasicCalendar1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var ADay:integer;
begin
  if mbLeft in Button then
    begin
    ADay := SelectCellXY(x,y);
    if ADay <> 0 then
        MessageBeep(0);
end;
```

SelectDay method

Applies to
CWBasicCalendar, CWCalendar components

Declaration

procedure SelectDay(ADay:integer);

The *SelectDay* method selects the cell determined by the value passed in ADay. If ADay is not a valid day number, for example, thirty for the month of February, then the calendar is not updated.

Example
This example updates the calendar to select the fifth day of the current displayed month.

```
SelectDay(5);
```

SetDate method

Applies to

*CWDateEdit, CWDateLabel, CWMonthCombo, CWYearCombo*components

Declaration

procedure SetDate;

The *SetDate* method updates the components displayed date. This method should be called by the calendars OnDateChange or OnRangeChange events.

Example

This example updates a DateLabel whenever the calendars date range changes. The calendars SelectRange property is set to true.

```
procedure TForm1.CWCalendar1RangeChange(Sender: TObject);
begin
  CWDateLabel1.SetDate;
end;
```

Text method

Applies to

CWCalendar component

Declaration

function Text(ADay,Index:integer):**string**

The *Text* method returns the text line specified by Index for the cell date defined by the value passed in ADay. If no text is present or ADay is an invalid day, an empty string is returned.

Example

The following example checks a cells text for the string, *Meeting*, and removes it if found.

```
procedure TForm1.RemoveMeeting(ADay:integer);
begin
  for x = 1 to 6 do
      begin
      if CWCalendar1.Text(ADay,x) = 'Meeting' then
                CWCalendar1.DeleteText(ADay,x);
   end;
end;
```

TextCount method

Applies to

CWCalendar component

Declaration

function TextCount(ADay:integer):**integer**

The *TextCount* method returns the number of text lines which contain text for the cell date defined by the value passed in ADay.

WeekDay method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

function WeekDay(ADate:TDateTime):integer;

The *Weekday* method returns a numeric value indicating which day of the week the date passed in ADate falls on. *Weekday* returns 1 for Sunday, 2 for Monday, 3 for Tuesday, etc..

WeekNumber method

Applies to

CWBasicCalendar, CWCalendar components

Declaration

function WeekNumber(ADate:TDateTime):integer;

The *WeekNumber*method returns a numeric value indicating the week of the year passed in ADate.

BlockColor property

Applies to
CWCalendar component

Declaration

property BlockColor:TColor

The *BlockColor* property defines the background color of calendar cells which appear before the first of the month and after the last of the month. This value is initially set to the same value as **CellColor**, which is clWindow.

Example
The following example sets the block background color to gray.

```
CWCalendar1.BlockColor := clGray;
```

ButtonType property

Applies to
CWArrowBtn, CWNavBtn components

Declaration

property ButtonType:TNavBtn

The *ButtonType* property defines the function of the **CWArrowBtn** and **CWNavBtn** controls. This property takes a value of type **TNavBtn**, which is defined as:

```
type TNavBtn = (btPrevYear, btPrevMonth, btPrevDay, btNextDay, btNextMonth,
                btNextYear, btToday);
```

By default, *ButtonType* is set to btPrevYear. Setting this property automatically changes the glyph of a **CWNavBtn** to represent the action requested. This glyph can be changed by the user.

When pressed, this control will alter the calendar defined by **CWCalendar** as determined by the value of *ButtonType*.

ButtonWidth property

Applies to
CWDropDownEdit component

Declaration

property ButtonWidth:integer

The *ButtonWidth* property determines the control's button width. This value is initially set to fifteen,

but can be change via the Object Inspector, or at run-time, to accomadate glyph's of various sizes.

CalendarCaption property

Applies to
CWQuickCalendar component

Declaration

property CalendarCaption:string

The *CalendarCaption* property set the Quick Calendar title-bar caption. If this property is set to **nil,** the Quick Calendar appears without a title bar.

CalendarDate property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property CalendarDate:TDateTime

Run-time only. The *CalendarDate* property is used to determine the current selected calendar date and to set the calendar date. To set the calendar date, *CalendarDate* must be a valid date value.

Example
This example checks to see if the current selected date is today's date and if not then sets the calendar date to today's date.

```
with CWCalendar1 do
    begin
    if CalendarDate <> Date then
            CalendarDate := Date;
    end;
```

CalendarLeft property

Applies to
CWQuickCalendar component

Declaration

property CalendarLeft:integer

The *CalendarLeft* property determines the vertical axis on which the Quick Calendar will appear on

the screen. This value is relative to the screen co-ordinates. By default, this value is 150.

CalendarLink property

Applies to
CWDateSpin components

Declaration

property CalendarLink:TCalendarLink

The *CalendarLink* property defines which date element of the calendar set in the CWCalendar property is to be changed by the **CWDateSpin** control when one of its buttons is pressed. This property takes a value of **TCalendarLink**, which is defined as:

```
type TCalendarLink = (clDay, clMonth, clYear);
```

By default, this value is set to clDay.

CalendarTop property

Applies to
CWQuickCalendar component

Declaration

property CalendarTop:integer

The *CalendarTop* property determines the horizontal axis on which the Quick Calendar will appear on the screen. This value is relative to the screen co-ordinates. By default, this value is 150.

Canvas property

Applies to
CWCalendar component

Declaration

property Canvas:TCanvas

Run-time only. The *Canvas* property allows the user to paint directly to the calendar canvas. Individual day cell co-ordinates can be obtained by calling the **GetDayRect** method.

CellColor property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property CellColor:TColor

The *CellColor* property determines the background color of the calendar date cells. This value can be any valid TColor value.

Example
The following example sets the cell background color to yellow.

```
CWCalendar1.CellColor := clYellow;
```

CellFont property

Applies to
CWCalendar component

Declaration

property CellFont:TFont

The *CellFont* property is a font object that controls the attributes of the text used to display text inside the date cells.. The *Color*, *Name*, *Size* and *Style* properites of the *CellFont* property can be changed using the Object Inspector or by code during run-time.

Example
The following example changes the CellFont to Arial Bold in a size 8.

```
CWCalendar1.CellFont.Name := 'Arial';
CWCalendar1.CellFont.Style := [fsBold];
CWCalendar1.CellFont.Size := 8;
```

Col property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property Col:integer

Run-time and read only. The *Col* property holds the column value of the current selected cell.

Used internally. May be of limited user to developers.

CWCalendar property

Applies to
*CWArrowBtn, CWDateEdit, CWDateLabel, CWDateSpin, CWMonthCombo, CWNabBtn, CWYearCombo*components

Declaration

property CWCalendar:TCWCalendar

The *CWCalendar* property sets the calendar which is to interface with the component. Usually, this is in relation to date changes. Specifying the calendar will enable the calendar to change as the date in the control changes. The control itself is updated by a call to **SetDate**.

DateFont property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property DateFont:Font

The *DateFont* property is a font object that controls the attributes of the text used to display the month day values. The *Color, Name, Size* and *Style* properites of the *DateFont* property can be changed using the Object Inspector or by code during run-time.

Example
The following example changes the DateFont to Arial Italic.

```
CWCalendar1.DateFont.Name := 'Arial';
CWCalendar1.DateFont.Style := [fsItalic];
```

DateFormatStr property

Applies to
CWDateEdit, CWDateLabel, CWDropdownEdit components

Declaration

property DateFormatStr:string

The *DateFormatStr* property determines how the control will display calendar dates. This value is initially set by Window's short date format. The string can be changed via the Object Inspector or by code at run time.

DateFormatString property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property DateFormatString:TDateTime

Run-time only. The *DateFormatString* property is used by the DateText property to format the current selected date. Initially, the *DateFormatString* is set to 'mmmm dd, yyyy'.

Example
This example sets the DateFormatString to 'dddd, mmmm d, yyyy' . It will then set the caption of Label1 to display the date in the format of Wednesday, May 1, 1996.

```
with CWCalendar1 do
    begin
    DateFormatString := 'dddd, mmmm d, yyyy';
  Label1.Caption := DateText;
    end;
```

DatePosition property

Applies to
CWCalendar component

Declaration

property DatePosition:TDatePosition

The *DatePosition* property defines where in the date cell the day number is displayed. *DatePosition* takes a value of type **TDatePosition** which is defined as:

```
type TDatePosition = (Centered,LowerLeft,LowerRight,UpperLeft,UpperRight);
```

By default, *DatePosition* is set to Centered. Normally, this property is set using the Object Inspector, however, the value can be changed at run-time, if necessary.

Example
This example positions the day numbers in the bottom right hand corner of the calendar cells.

```
with CWCalendar1 do
    begin
    ...
    DatePosition := LowerRight;
    ...
    end;
```

DateText property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property DateText:string

Run-time and read only. The *DateText* property returns the currently selected date in the format specified by the DateFormatString property.

Example
This example sets the DateFormatString to 'dddd, mmmm d, yyyy' . It will then set the caption of Label1 to display the date in the format of Wednesday, May 1, 1996 using DateText.

```
with CWCalendar1 do
      begin
      DateFormatString := 'dddd, mmmm d, yyyy';
  Label1.Caption := DateText;
      end;
```

DateToShow property

Applies to
CWDateEditl component

Declaration

property DateToShow:string

The *DateToShow* property is a string representation of the date to initially appear in the control's edit window. However, if CWCalendar is set, the actual date text will be determined by the calendar's date.

Day property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property Day:integer

The *Day* property reads and sets the current day date value. If using *Day* to set the date of the calendar, *Day* must be a valid value for the currently displayed month.

Example
The following example checks to see if the first of the month is selected. If not, then it is set using the Day property.

```
with CWCalendar1 do
    begin
    ...
    if Day <> 1 then
    Day := 1;
    ...
    end;
```

DayTitleFormat property

Applies to
CWBasicCalendar, *CWCalendar* components

Declaration
property DayTitleFormat:TDayTitleFormat

The *DayTitleFormat* property determines how the calendar's day titles will be displayed. *DayTitleformat* takes a value of type **TDayTitleFormat** which is defined as:

```
type TDayTitleFormat = (OneLetter,TwoLetters,ThreeLetters,FullName);
```

Day titles may appear as the first one, two or three letters of the day name or as the full day name.

Example
This example sets the calendar to display the day titles using the first three letters of the day name.

```
with CWCalendar1 do
    begin
    ...
    DayTitleFormat := ThreeLetters;
    ...
    end;
```

DayTitles property

Applies to
CWBasicCalendar, *CWCalendar* components

Declaration
property DayTitles:TStringList

The *DayTitles* property is a string list which holds the names of the days of the week. By default, the titles are in English. These titles can be easily changed to any language using the Object Inspector.

Direction property

Applies to
CWArrowBtn component

Declaration

property Direction:TDirection

The *Direction* property determines the direction in which the **CWArrowBtn** points. This property takes a value of type **TDirection**, which is defined as:

```
type TDirection = (dLeft, dRight, dUp, dDown);
```

By default, the value is dLeft;

DisplayDate property

Applies to
CWQuickCalendar component

Declaration

property DisplayDate:string

The *DisplayDate* property is a string representation of the date to appear in the Quick Calendar control when it first appears.

DownGlyph property

Applies to
CWDateSpin component

Declaration

property DownGlyph:TBitmap

The *DownGlyph* property defines which bitmap is to appear in the Down button of the **CWDateSpin** control. By default, this is a down arrow.

DragDate property

Applies to
CWCalendar component

Declaration

property DragDate:boolean

Run-time only. The *DragDate* property is a boolean value which determines if the calendar will allow the selected date to be dragged off the calendar. The default value is false which allows the user to click the left mouse button, hold it down and move around the calendar highlighting the date cells over which the mouse pointer falls. This is the normal function of the calendar.

With *DragDate* set to true, the left mouse button click causes the calendar to select the cell under the mouse pointer, but does not move the selected cell as the mouse pointer moves around the calender while the button is held down.

EditDate property

Applies to
CWDropDownEdit component

Declaration

property EditDate:TDateTime

Run-time only. The *EditDate* property is a **TDateTime** representation of the date text which appears in the control's edit window. Initially, this value is set to **Date** but changes when the user selects a new date from the drop down calendar.

FocusColor property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property FocusColor:TColor

The *FocusColor* property sets the color used to draw the dotted focus rectangle inside the selected cell when the calendar has the focus. By default, this color is black. If **FocusMode** is set to fm3D or fmNone then changing the *FocusColor* has no effect on the appearance of the calendar.

FocusMode property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property FocusMode:TFocusMode

The *FocusMode* property determines how the calendar will indicate that it has the focus. *FocusMode* takes a value of type **TFocusMode** which is defined as

```
type TFocusMode = (fmFramed,fm3D, fmNone);
```

When *FocusMode* is set to fmFramed, a dotted retangle appears inside the selected cell when the calendar has the focus. When *FocusMode* is set to fm3D and **HiLight3D** is set to either hlRaised or hlLowered, the selected cell toggles its appearance from raised to lowered depending on the state of focus. If *FocusMode* is set to fmNone, the calendar does not indicate visibly when it has the focus.

Example

The following example sets HiLight3D to hlRaised and FocusMode to fm3d. The calendar will now show the selected cell as a raised rectangle when it does not have the focus and will show the selected cell as a lowered rectangle when it does have the focus.

```
with CWCalendar1 do
    begin
    ...
    FocusMode := fm3D;
    HiLight3D := hlRaised;
    ...
    end;
```

GridColor property

Applies to

CWBasicCalendar, CWCalendar components

Declaration

property GridColor:TColor

The *GridColor* property determines the color used to draw the calendar day grid. If **Ctl3D** is true then only the grid inside the frame is painted in the color defined by *GridColor*. When **Ctl3D** is false, all gridlines, including outer border and day title header, are drawn in the color defined by *GridColor*.

GridLines property

Applies to

CWBasicCalendar, CWCalendar components

Declaration

property GridLines:boolean

The *GridLines* property determines if the calendar's grid is shown or not. When set to true, the grid is shown, and when set to false, the grid is not shown. To create a 3D calendar, set the **GridColor** to clSilver, set *GridLines* to false and set **HiLight3D** to hlRaised or hlLowered.

HeaderColor property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property HeaderColor:TColor

The *HeaderColor* property sets the color to be used when painting the cells containing the calendar's day titles. By default, this color is clSilver.

HeaderFont property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property HeaderFont:Font

The *HeaderFont* property is a font object that controls the attributes of the text used to display the day titles. The *Color, Name, Size* and *Style* properites of the *HeaderFont* property can be changed using the Object Inspector or by code during run-time.

Example
The following example changes the HeaderFont to Arial Italic.

```
CWCalendar1.HeaderFont.Name := 'Arial';
CWCalendar1.HeaderFont.Style := [fsItalic];
```

HeaderHeight property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property HeaderHeight:integer

The *HeaderHeight* property sets the height of the day title cells. By default, this height is 18 pixels.

Example
The following example sets the HeaderHeight to 30 in order to accommodate a larger HeaderFont.

```
with CWCalendar1 do
    begin
    HeaderHeight := 30;
    HeaderFont.Name := 'Arial';
    HeaderFont.Size := 20;
    end;
```

HiLight3D property

Applies to
CWBasicCalendar, CWCalendar components

Declaration
property HiLight3D:THiLight3D

The *HiLight3D* property determines if the selected date cell will be displayed in 3D. If displayed in 3D, the seleced cell can be either raised or lowered. *HiLight3D* takes a value of type **THiLight3D**, which is defined as:

```
type THiLight3D = (hlNone, hlRaised, hlLowered);
```

By default, **HiLight3D** is set to hlNone.

Example
The following example sets HiLight3D to hlRaised and FocusMode to fm3d. The calendar will now show the selected cell as a raised rectangle when it does not have the focus and will show the selected cell as a lowered rectangle when it does have the focus.

```
with CWCalendar1 do
    begin
    ...
    FocusMode := fm3D;
    HiLight3D := hlRaised;
    ...
    end;
```

HiLightColor property

Applies to
CWBasicCalendar, CWCalendar components

Declaration
property HiLightColor:TColor

The *HiLightColor* property specifies which color will be used to paint the background of the selected cell. By default, this value is set to Windows highlight color. If creating a 3D highlight, the color is best set to clSilver.

Example

The following example sets HiLightColor to clSilver, HiLight3D to hlRaised and FocusMode to fm3d. The calendar will now show the selected cell as a raised rectangle when it does not have the focus and will show the selected cell as a lowered rectangle when it does have the focus.

```
with CWCalendar1 do
    begin
    ...
    HiLightColor := clSilver;
    FocusMode := fm3D;
    HiLight3D := hlRaised;
    ...
    end;
```

Language property

Applies to

*CWMonthCombo*component

Declaration

property Language:TLanguage

The *Language* property defines the language in which the months of the year are displayed in the **CWMonthCombo** control. This property takes a value of type **TLanguage**, which is defined as:

```
type TLanguage = (English, French, German, Spanish);
```

By default, *Language* is set to English. Any language can be easily added to the source code.

Month property

Applies to

CWBasicCalendar, *CWCalendar* components

Declaration

property Month:integer

The *Month* property reads and sets the current month date value. If using *Month* to set the date of the calendar, *Month* must be a value between 1 and 12 inclusive.

Example

The following example checks to see if if the month is set to December. If so, then the month is set to January and the year is incremented. Otherwise, just the month is incremented.

```
with CWCalendar1 do
    begin
    ...
    if Month = 12 then
      begin
      Month := 1;
      Inc(Year);
      end
    else
      Inc(Month);
    ...
    end;
```

NumYears property

Applies to

*CWYearCombo*component

Declaration

property NumYears:integer

The *NumYears* property sets the number of years which will be displayed in the drop down list of the **CWYearCombo** control when the controls button is pressed. By default, this value is set to eight.

RangeColor property

Applies to

CWCalendar component

Declaration

property RangeColor:TColor

Run-time only. The *RangeColor* property defines the color used to fill the cells of a selected range of dates when the **SelectRange** property is true. This property is set after all other properties have loaded. By default, *RangeColor* is set to the same color as **CellColor**. If **CellColor** is equal to **HiLightColor**, the calendar is likely designed as a 3D calendar with the predominant color of clSilver. In this case, *RangeColor* is set to clGray.

RangeColor can be changed at run-time.

Example

The following example sets RangeColor to clRed..

```
with CWCalendar1 do
```

```
begin
...
RangeColor := clRed;
...
end;
```

RangeEndDate property

Applies to
CWCalendar component

Declaration

property RangeEndDate:TDateTime

Run-time and read only. The *RangeEndDate* property holds the **TDateTime** value for the latest date selected in a range of dates when the **SelectRange** property is true. By default, *RangeEndDate* is set to **Date**.

RangeStartDate property

Applies to
CWCalendar component

Declaration

property RangeStartDate:TDateTime

Run-time and read only. The *RangeStartDate* property holds the **TDateTime** value for the earliest date selected in a range of dates when the **SelectRange** property is true. By default, *RangeStartDate* is set to **Date**.

ReadOnly property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property ReadOnly:boolean

The *ReadOnly* property defines whether the calendar will accept user input. If set to true, the calendar will not highlight the selected date. This mode should be used if the developer wishes to override the built-in paint method and customize the appearance of the calendar. By default, *ReadOnly* is set to false.

RepeatTimer property

Applies to
CWArrowBtn, CWNavBtn components

Declaration

property RepeatTimer:boolean

The *RepeatTimer* property determines if the control will continue to call its Click method while the control remains pressed. Initially, this value is set to false.

Row property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property Row:integer

Run-time and read only. The *Row* property holds the row value of the current selected cell.

Used internally. May be of limited user to developers.

SelectRange property

Applies to
CWCalendar component

Declaration

property SelectRange:boolean

The *SelectRange* property sets the calendar in date range mode when its value is true. While in this mode, date ranges are selected by dragging the cursor over the calendar. Date ranges are also selectable by a combination of the Shift-Key and the arrow keys.

ShowDayTitles property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property ShowDayTitles:boolean

The *ShowDayTitles* property defines whether the days of the week are displayed along the top of the calendar. By default, this value is set to true.

Example
The following example removes the DayTitles from the top of the calendar.

```
with CWCalendar1 do
    begin
    ...
    ShowDayTitles := false;
    ...
    end;
```

StartOfWeek property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property StartOfWeek:TDayOfWeek

The *StartOfWeek* property defines which day of the week the calendar will start on. *StartOfWeek* takes a value of **TDayOfWeek**, which is defined as:

```
type TDayOfWeek = (Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday);
```

By default, *StartOfWeek* is set to Sunday. Changing the value of StartOfWeek will cause the calendar to repaint.

Example
The following example sets the calendar to use Monday as the start of the week.

```
with CWCalendar1 do
    begin
    ...
    StartOfWeek := Monday;
    ...
    end;
```

SunDayColor property

Applies to
CWBasicCalendar, CWCalendar, CWQuickCalendar components

Declaration

property SunDayColor:TColor

The *SunDayColor* property sets the color used to paint all dates which fall on a Sunday. By default, this color is set to clRed. If today's date falls on a Sunday and **ToDayColor** is set to a color other that *SunDayColor*, that date will be displayed in the **ToDayColor**.

TextLines property

Applies to
CWCalendar component

Declaration

property TextLines:integer

The *TextLines* property is used to set the maximun number of lines of text which the user can insert into the date cells. This value cannot be more than six. CalendarWorks determines where inside the cell to start the first line of text based on the value of *TextLines*. It is up to the user to make sure the value is set correctly for properly positioned text.

ToDayColor property

Applies to
CWBasicCalendar, CWCalendar, CWQuickCalendar components

Declaration

property ToDayColor:TColor

The *ToDayColor* property sets the color used to paint the current date as returned by the **Date** method. By default, this color is set to clNavy. If today's date falls on a Sunday and *ToDayColor* is set to a color other that **SunDayColor**, that date will be displayed in the *ToDayColor*.

UpGlyph property

Applies to
CWDateSpin component

Declaration

property UpGlyph:TBitmap

The *UpGlyph* property defines which bitmap is to appear in the Up button of the **CWDateSpin** control. By default, this is an up arrow.

Year property

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property Year:integer

The *Year* property reads and sets the current year date value. If using *Month* to set the date of the calendar, *Month* must be a value integer value. There is no bounds checking on this value, therefore, it is up to the developer to make sure this value is reasonable.

Example
The following example checks to see if the calendar is displaying the current year. If not, the calendar is changed.

```
with CWCalendar1 do
    begin
    ...
    DecodeDate(Date,AYear,AMonth,ADay);
    if AYear <> Year then
      Year := AYear;
    ...
    end;
```

OnDateChange event

Applies to
CWBasicCalendar, CWCalendar components

Declaration

property OnDateChange:TNotifyEvent

The *OnDateChange* event is called whenever the calendar's selected date changes. If the **SelectRange** property of the **CWCalendar** component is set to true, the *OnDateChange* event is not called. In this case, the **OnSelectRange** event should be used.

When displaying text and bitmaps within the cells of a CWCalendar, it is best to use the **OnUpdate** event to make changes to the entire month all at once.

The **CWBasicCalendar** component does not allow for display of text and bitmaps inside the date cells.

Example
The following example updates a label to display the current date whenever the calendar date changes.

```
procedure TForm1.CWCalendar1DateChange(Sender: TObject);
begin
    Label1.Caption := CWCalendar1.DateText;
end;
```

OnRangeChange event

Applies to
CWCalendar component

Declaration

property OnRangeChange:TNotifyEvent

The *OnRangeChange* event is called whenever the calendar's selected date range changes while the **SelectRange** property is set to true. If the **SelectRange** property of the **CWCalendar** component is set to true, the *OnDateChange* event is not called. In this case, the *OnRangeChange* event should be used.

The CWDateLabel component, when used in conjunction with CWCalendar, checks to see if the RangeStartDate and RangeEndDate are different and automatically adjusts its display.

Example
The following example updates a CWDateLabel to display the current range whenever the calendar range changes.

```
procedure TForm1.CWCalendar1RangeChange(Sender: TObject);
begin
```

```
  CWDateLabel1.SetDate;
end;
```

OnUpdate event

Applies to

CWBasicCalendar, CWCalendar components

Declaration

property OnUpdate:TNotifyEvent

The *OnUpdate* event is called whenever the calendar month or year changes. This gives the user the chance to update the entire month all at once.

When displaying text and bitmaps within the cells of a CWCalendar, it is best to use the *OnUpdate* event to make changes rather than the **OnDateChange** event.

The **CWBasicCalendar** component does not allow for display of text and bitmaps inside the date cells.

Example

The following example adds the text, *Meeting*, and the alarm bitmap to today's date.

```
procedure TForm1.CWCalendar1Update(Sender: TObject);
begin
  CWCalendar1.AddCellText(Date,clRed,'Meeting');
  CWCalendar1.AddCellBitmap(Date,'CWAlarm');
end;
```