QuickReport

version 1.0d for Borland C++Builder

QuSoft AS

web http://www.qusoft.no

e-mail support qusoft.no

swreg #8644

Postal address:

Fred Olsensgt. 1
N-0152 Oslo
Norway

Fax number: +47 22 41 74 91

QRDBCalc
__property bool AlignToBand;
__property long AsInteger;
__property double AsReal;
__property TFont Font;
__property TQRCalcOperation Operation;
property TFont ParentFont;
__property System::AnsiString PrintMask;
__property TQRBand* ResetBand;
__property TQRLabelOnPrintEvent OnPrint;
QRDetailLink
__property Db::TDataSource* DataSource;
__property TQRBand* DetailBand;
__property TQRBand* FooterBand;
__property TQRBand* HeaderBand;
__property TQRController* Master;
__property bool PrintBefore;
__property TQRFilterEvent OnFilter;
__property TReportNeedDataEvent OnNeedData;
QRDBText
__property TAlignment Alignment;
__property bool AlignToBand;
__property bool AutoSize;
__property TColor Color;
__property Db::TDataSource* DataSource;
__property System::AnsiString DataField;
__property TFont Font;
__property Boolean ParentFont;
__property bool Transparent;
__property TQRLabelOnPrintEvent OnPrint;
QRGroup
__property Db::TDataSource* DataSource;
__property System::AnsiString DataField;
__property TQRBand* FooterBand;
__property TQRBand* HeaderBand;
__property int Level;
__property TReportNeedDataEvent OnNeedData;
QRLabel
__property TAlignment Alignment;
__property bool AlignToBand;
__property bool AutoSize;
__property System::AnsiString Caption;
__property TColor Color;
__property TFont Font;
__property bool ParentFont;
__property bool Transparent;
__property TQRLabelOnPrintEvent OnPrint;
QRMemo
__property TAlignment Alignment;
__property TColor Color;
__property TFont Font;
__property Classes::TStringList* Lines;
__property bool ParentFont;
__property bool WordWrap;
QRPreview
__property long PageNumber;

__property int Zoom;
void __fastcall ZoomToFit(void);
void __fastcall ZoomToWidth(void);
QRPrinter
__property bool Canceled;
__property Graphics::TMetafileCanvas* Canvas;
__property int Copies;
__property bool EnableOpenBtn;
__property bool EnableSaveBtn;
__property int FromPage;
__property int LeftWaste;
__property Printers::TPrinterOrientation Orientation;
__property Graphics::TMetafile* Page;
__property int PageCount;
__property int PageHeight;
__property int PageNumber;
__property int PageWidth;
__property int PaperBin;
__property int PaperLength;
__property TQRPaperSize PaperSize;
__property int PaperWidth;
__property System::AnsiString PreviewCaption;
__property bool PrinterOK;
__property System::AnsiString ProgressCaption
__property bool ShowProgress;
__property TQRPrinterStatus Status;
__property int Thumbs;
__property System::AnsiString Title;
__property int ToPage;
__property int TopWaste;
__property int TotalPageHeight;
__property int TotalPageWidth;
__property bool ZoomButtons;
__property TOnPreviewEvent OnPreview
void __fastcall BeginDoc(void);
void __fastcall Cancel(void);
void __fastcall Cleanup(void);
void __fastcall EndDoc(void);
void __fastcall Load(System::AnsiString Filename);
void __fastcall NewPage(void);
void __fastcall Preview(void);
void __fastcall Print(void);
void __fastcall Save(System::AnsiString Filename);
QRSysData
__property TAlignment Alignment;
__property bool AlignToBand;
__property bool AutoSize;
__property TColor Color;
__property TQRSysDataType Data;
__property TFont Font;
__property bool ParentFont;
__property TQRLabelOnPrintEvent OnPrint;
__property System::AnsiString Text;
__property bool Transparent;
QRShape
__property Graphics::TBrush* Brush;

Introduction

About this document

This document describes the 32 bit version of QuickReport for C++Builder. You use QuickReport and the features described in this document entirely at your own risk. If you have any questions regarding QuickReport, this document, or related topics please contact Borland Technical support or QuSoft.

About QuickReport

Welcome to QuickReport for C++Builder. QuickReport is designed to be a tightly integrated visual reporting tool, that also gives you great code control over your reports. Here is a listing of some of QuickReport's features:

- Banded report generator, using the C++Builder form designer as the report designing environment
- Event handlers can be attached to virtually any part of the report generation process to give you great control of the finished report
- Powerful on-screen preview, including preview component for easy creation of your own preview forms. Reports can even be previewed from within the C++Builder IDE
- Create reports from tables and queries, in addition to arrays, lists, text files, or any source you can imagine
- Powerful master/detail functions with unlimited levels and unlimited detail tables at the same level
- Create mailing labels and multi column reports
- Automatic printing of thumbnails
- Print labels, database fields, calculations, memo fields, pictures, icons, and shapes
- Can be compiled to work without any dataawareness at all, working on PC's without the BDE
- Use any Windows font, size, style, and color
- Finished reports can be saved to file for later viewing or printing
- Portrait or Landscape printing of reports
- Build in support for custom paper sizes
- On line help file describing all methods, properties, and events

Licensing and Registration

QuickReport 1.0d for C++Builder

C++Builder includes the full 32 bit version of QuickReport 1.0. No additional registration is needed to use the version included with C++Builder 2.0. The full 32 bit version of QuickReport 1.0d is available for free use by C++Builder users.

QuickReport 2.0 for C++Builder

QuickReport 2.0 is now available for C++Builder. This is a greatly improved version with easier and faster user interface and many new features. More information and an order form can be found at the end of this document.

Copyright and Distribution

 QuSoft can be contacted on the web at http://www.qusoft.no, via fax at +47 22 41 64 91 or postal mail at:
QuSoft AS, Fred Olsensgt. 1, N-0152 Oslo, Norway

The latest version of QuickReport can be downloaded from our ftp server at ftp.qusoft.no. Go to the /pub/quickrep directory. A pointer to the latest version can also be found on our web page.

Use of QuickReport is entirely at your own risk. Under no events shall QuSoft AS be responsible for any loss or damage caused by the use or distribution of QuickReport.

Getting Technical Support

Users of the QuickReport included in C++Builder will get QuickReport support through Borland Technical Support .

Users who have purchased QuickReport 2.0 / source code from QuSoft will receive technical support via e-mail. Please send your questions to *support@qusoft.no*. Depending on the amount of registrations we might also set up a technical support phone line but details are not set yet. Announcements will be made through our Web site.

Using QuickReport

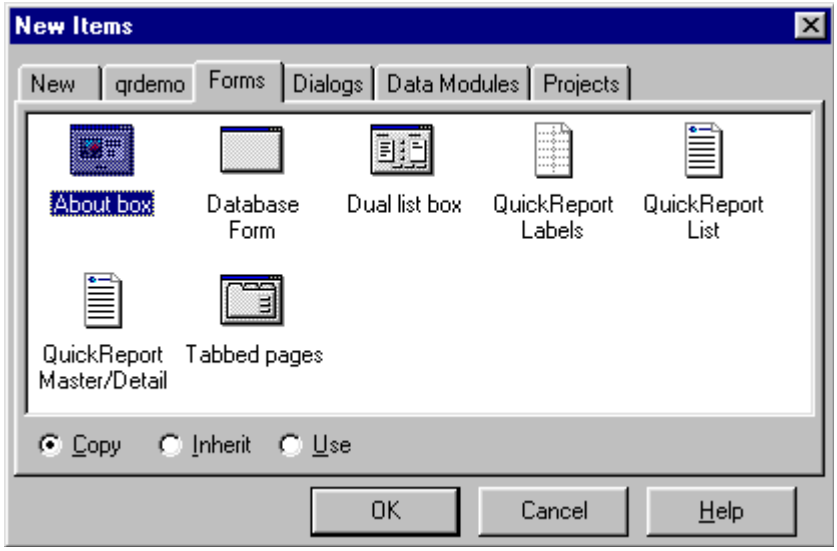
Getting Started

QuickReport is a fast, easy, native component reporting solution for C++Builder. To help you understand how to use QuickReport, C++Builder includes both templates in the Object Repository and a full sample reporting application in Cbuilder\Examples\Dbtasks\Quickrpt directory.

QuickReport Templates and the C++Builder Object Repository

In C++Builder you will find three report templates on the Forms page of the Object Repository,. including:

QuickReport Mailing Labels	A basic mailing label template
QuickReport List	Simple list template with page header and footer
QuickReport Master/detail	A Master/Detail template, using the QRDetailLink component



Instructions on customizing the templates are included in the source code automatically created when you use one of the templates.

QuickReport Sample Applications

QuickReport includes a sample application to get you started with your own reports. To use the demo program:

- Open the project QRDEMO.MAK in the CBUILDER\EXAMPLES\DBTASKS\QUICKRPT directory and run it.

There are five reports in this program plus a sample custom preview form. Each of them demonstrates some basic QuickReport features. The main form itself demonstrates how to call reports, both for print and preview. It also demonstrates how to connect a custom preview for general use in your application.

Simple Report	Shows you how to create a basic report with a title band, column headers, detail band and a page footer with page number. In addition it shows a simple use of grouping with the QRGroup component and event the use of the OnNeedData event.
Memos and Pictures	This report shows how to include images and memo fields in your reports.
Master Detail Report	The QRDetailLink component is a powerful helper when you want to create master detail reports. This report includes a total of four tables and also demonstrates the how to avoid orphan bands.
Mailing Labels	This report demonstrates how to create a mailing label report, in addition to the use of a overlay band.
Text File Lister	One powerful feature of QuickReport is the ability to create reports from datasources other than TTable and TQuery. It is a text file lister, using the QuickReport->OnNeedData event to retrieve lines from a text file. The report can very well be used as a source code lister, since it displays two columns of small text to fit a lot of information on one page.
Custom Preview form	The custom preview form shows how you can incorporate the preview totally in your application's look and feel using the QRPreview component. This form is very simple but can act as a starting point for your own preview form.

QuickReport Basics

When using QuickReport all reports are designed as C++Builder forms using the C++Builder form designer. Advanced users can also generate forms/reports programmatically on the fly but that is not covered in this document.

QuickReport is a *banded* report generator. This means that you build your reports using *QRBands*, a special type of the C++Builder TPanel, as the different parts of your report. A report can have several bands of different types:

- The most important bands are the ***Detail Band, Page header and footer band and Title band***. ***Group Headers, Footers and SubDetail bands*** work in conjunction with other components.

For a complete description of all band types please look up TQRBand in the Component Reference chapter.

Creating a very simple report

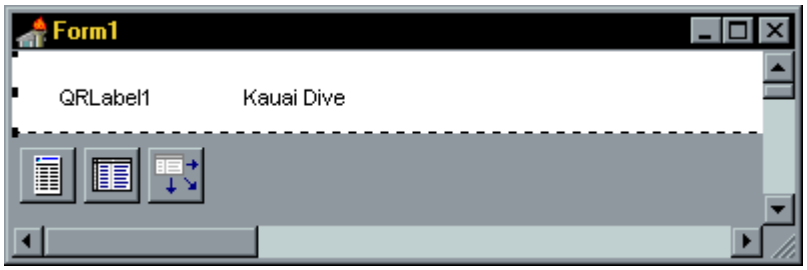
The simplest report you can create is a form having the following components:

- A **TQRBand** with BandType set to rbTitle.
- A **TQuickReport** component.
- A **QRLabel** on the QRBand. Put some text in the Caption property.
- Set the Form.Font to Arial or another TrueType font.

Now you can right click the TQuickReport component and select *Preview report* from the menu. A Preview form with a blank page, except for the tiny QRLabel you put on it, should now appear.

To make the report a bit more interesting, you do the following:

- Change the **QRBand::BandType property** to **rbDetail.**
- Add a TTable component to the form. Set the DataBase property to BCDemos and the TableName to Customers. Set Active to true.
- Add a TDataSource component and set the DataSet property to point to the TTable you just added.
- Set the DataSource property of QuickReport to point to the newly added DataSource.
- Add a QRDBText component to the QRBand. Set the DataSource property to the correct value and pick a field for the DataField property.



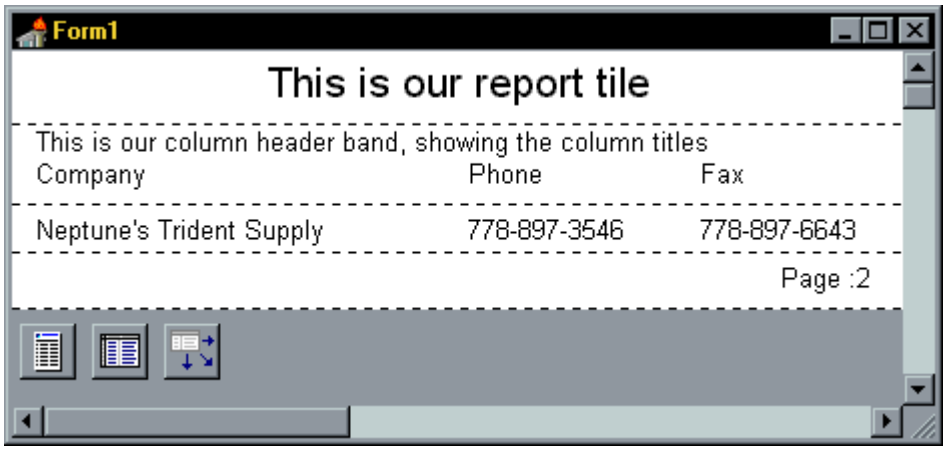
Now you can right click the QuickReport component and select *Preview report* again. You have now finished your first data-aware report using QuickReport!

Customizing Report Display

We will now create a customized report with a title band, column heading, detail band and page footer. Close any existing reports and create a new form.

- Create a new form.
- Set the form font to Arial.
- Put four bands on the report.
- Set their respective band types to: rbTitle, rbColumnHeader, rbDetail and rbPageFooter.
- Put a TTable component on the form, set DataBase to BCDemos, TableName to Customer and Active to true.
- Put a TDataSource on the form and set the DataSet property to Table1.
- Put a TQuickReport component on the form. Set the DataSource property to DataSource1.

- Put one TQRLabel on the title band and three TQRLabels on the column header band as show below, and set their Caption properties to This is our report title, and to Company, Phone, and Fax, respectively.
- Put three QRDBText components on the Detail Band. Set their DataSource property to DataSource1 and the DataField properties to Company, Phone, and Fax.
- Put a QRSysData on the Page footer band. Set the Data property to qrsPageNumber. Set the Text property to "Page: ".
- Your report should now look approximately like the one below.

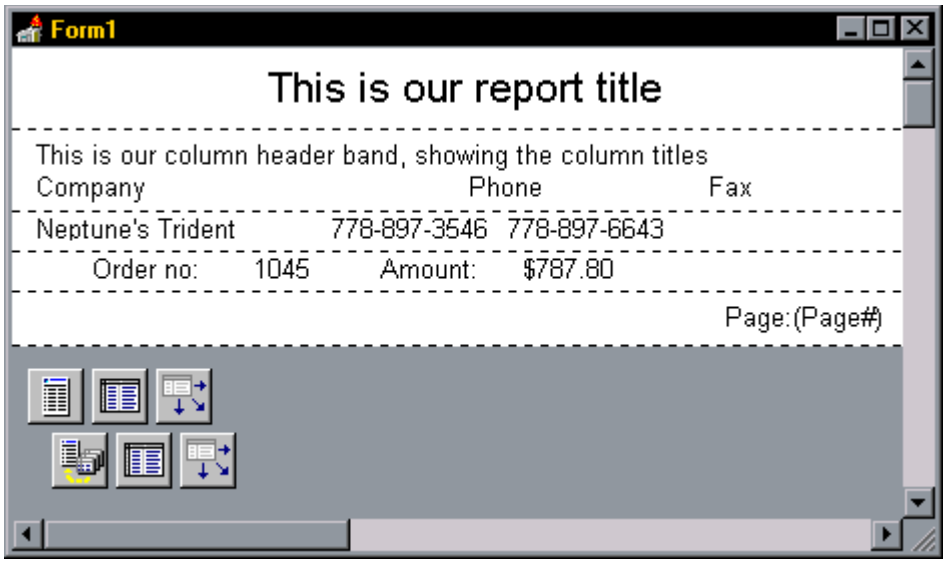


Again you can right click the QuickReport component and select *Preview report*.

Creating a Master Detail Report

We will now change this report into a master detail report, showing all the orders for each customer.

- Add a TTable, set the Database property to BCDemos and TableName to Orders, set Active to true. Set the MasterSource to DataSource1 and click on the MasterFields property to bring up the dialog box to create a master/detail relationship. Click on the index combo box and select CustNo, then click CustNo in both the list boxes and click *Add*. Click *OK*.
- Add a TDataSource, set the Dataset property to Table2 (the new table).
- Add a new QRBand to your report.
- Add a QRDetailLink component. Set the DataSource property to DataSource2 (The new one). Set the Master property to QuickReport1 and DetailBand to QRBand5 (The new band). The BandType property of this band will now be set to rbSubDetail automatically.
- Put two QRDBText components on the new band, connect them both to DataSource2 and then to OrderNo and AmountPaid. Set the Alignment properties to alRighJustify.
- Right click your QuickReport component and select *Preview report.*



The finished Master Detail form

Printing single record reports

When your QuickReport component doesn't have a DataSource connected it will print only one set of detail bands, effectively creating a report listing only one record. This makes it very easy to print the current record using a Data Module in C++Builder.

Creating reports from non BDE data sources

QuickReport can create reports from virtually any type of data. By not specifying a DataSource, but instead creating an OnNeedData event handler, you can make QuickReport print whatever data you manage to put into your QRLabel components. Both the QuickReport and the QRDetailLink components have the OnNeedData event, so you might very well create reports where some parts (master information) are taken from a table and other information (like detail information) is taken from some other source using the QRDetailLink->OnNeedData event. For a complete example on how to use the OnNeedData event take a look at the Text File Lister in the demonstration program. It will demonstrate how to print a text file with QuickReport, reading the file line by line and feeding the lines into a report.

Creating a Custom Preview Form

A print preview is essential to any application. In addition to providing a full featured preview form automatically, QuickReport makes it very easy to create preview forms with your own look and feel. The QRPreview component takes all the hassle out of displaying the actual page, providing you with properties such as PageNumber and Zoom to give you total control.

There are basically two steps needed to create your own preview form.
The first is to actually create the form, using a QRPreview component and probably several buttons for navigating between the pages, zooming, printing or whatever you want include.
The second step is to hook this form into QuickReport. This is done by writing an event handler for the QRPrinter::OnPreview event. At first sight it might look strange that this isn't an event in QuickReport component, but that would require you connect your preview form to all reports in your form. Putting this event in the QRPrinter class will usually mean you have to write one line of code to attach the event handler, then it will be active for *all* reports in your application. A typical event handler might look like this for the QRPrinter->OnPreview event:

```
void __fastcall TMainForm::OnPreview()
{
   PreviewForm->ShowModal();
}
```

And the code you would use to attach this preview form to the QRPrinter class could be put in the OnCreate event of MyMainForm and would look like this:

```
QRPrinter1->OnPreview = OnPreview;
```

Your preview form would now be used any time you call the Preview() method.
For a complete example on both a preview form and how to connect it please take a look at the QuickReport demonstration program.

Printing Images

QuickReport has no image component by itself. Instead it is aware of any standard TImage or TDBImage component placed on the report and will print them.

Important Issues when using QuickReport

Using Small and Large fonts when designing reports

Please always have your Windows font size set to small fonts when designing reports with QuickReport. If your reports are designed using Large Fonts you should set the global qrOrgRes variable to 120, indicating 120 pixels per inch. QuickReport will then scale all controls correctly no matter what resolution and font size is used when the report is printed.

Limitations of the design time preview

During the design time preview, QuickReport cannot call any event handlers you have created. This limits the design time preview to reports where the dataset is available at design time, and where events are not critical to the working of the report.

QuickReport and DataSources

QuickReport will not restore the current record after it has finished creating a report. If you want your datasource to be restored you will have to do so manually.
Also note that a lot of screen flashing might appear if you run QuickReport on a dataset used by a form visible at the time you run the report. If you run reports on the same dataset as you use in your forms you should hide those forms before running the report. DisableControls will not work, since that will also disable the data-aware QuickReport controls, causing all records in the report to print the same data.

Answers to some Frequently Asked Questions

How do I format output of a data field?

Formatting the output of a data field can be done in two ways. The best is to use the DisplayMask property of the field in question; double-click your TTable or TQuery component and make sure the field is added as a component. Select the field component and set the DisplayMask property.

You can also use the OnPrint event of the QRDBText or QRLabel components to format the contents just prior to printing. Note that you should change the Value parameter passed to the event handler, not the actual caption of the component.

How do I combine several reports into one?

Using the TQRDetailLink component you can create reports with many 'master' tables. Do not connect the TQuickReport component to a datasource, but instead create two or more TQRDetailLink components and set the TQuickReport component as the master for all of them. Connect detail bands, headers and footers as you wish. You can even attach any number of subDetail bands to these 'master' tables. When preparing the report, QuickReport will find out that no master table is present, and only the detail tables will be printed.

How do I filter records?

QuickReport has a build in data filtering event which can be used to accept or reject a record. Both TQuickReport and TQRDetailLink has an OnFilter event, and in this event handler you can check the values of the current record. Set the PrintRecord parameter to false if you want to reject the current record.

How do I make sure several bands stay on the same page?

The TQRBand component has a LinkBand property which is used to make several bands stick together. The LinkBand property can be set connected to another QRBand and QuickReport will then make a page break if there is not enough space left on the page for *both* bands. The second band can in turn be linked to a third band, and so on.

Why are my fonts clipped or displayed at a wrong size during preview?

QuickReport relies on Windows to perform any scaling of fonts during preview and thumbnail printing. Windows is not able to scale all fonts. Typically TrueType fonts and PostScript fonts work very well, while Bitmap fonts (like MS SYSTEM) have difficulty. Please make sure you use a TrueType or PostScript font if you want accurately scaled previews and thumbnail printouts.

Why are my reports suddenly rescaled and all the fonts have changed size?

C++Builder sometimes rescale controls on a form, typically if you change screen resolution or Windows font size. Make sure the Scaled property of your report form is set to false, as this will minimize the problem. Also see the *Important Issues on using QuickReport* paragraph. Starting with version 1.0c the Scaled property is automatically set to false when you drop a QuickReport component on a form.

How do I change the layout of mailing labels to fit any particular form ?

Select the number of labels across the page with the Columns property of TQuickReport. Put a page header band on the report to change top spacing. Use the LeftMargin property to change the left spacing and use ColumnMargin to adjust column spacing. Using these three properties, you should be able to create mailing label reports that fit most needs.

How do I eliminate the large black areas on the HP LaserJet 4 reports?

Try selecting an HP LaserJet III driver instead or get an updated LaserJet 4 driver from HP.

Why are some bands printed at the top of the report, even if they shouldn't?

The default BandType of is rbTitle. You have probably forgotten to change this to the correct band type.

How do I print one record per page?

If you want to print a report with one record on each page you set the ForceNewPage property of your Detail band to True.

Why doesn't my modeless custom preview form work?

When you call the QuickReport->Preview() method, QuickReport first prepares the report, then displays it (modally) and finally frees the report from memory. But when you call up the preview modeless, QuickReport continues to execute, freeing the report you are trying to view on screen at the same time. So a different approach must be used to show a report modeless. First call QuickReport->Prepare() instead of Preview(). Then call QRPrinter->Preview() (make sure you have set the QRPrinter->OnPreview event to an event handler which calls up your preview form, modeless). The report will now stay in memory and your preview form will work correctly. Before you prepare another report you must now call a QRPrinter->Cleanup() to free the allocated memory and make the QRPrinter ready to accept a new print job.

Writing your own printable components

If the included printable controls don't include just the control you need, you can easily create your own, using the TQRCustomControl as a base class. This class has the following class declaration:

```
class __declspec(delphiclass) TQRCustomControl;
class __declspec(pascalimplementation) TQRCustomControl : public Stdctrls::TCustomLabel
{
   typedef Stdctrls::TCustomLabel inherited;

private:
  bool FAlignToBand;
  TQRLabelOnPrintEvent FOnPrintEvent;
  TQuickReport* FReport;
  __property bool AlignToBand = {read=FAlignToBand, write=FAlignToBand, nodefault};


protected:
  __property TQRLabelOnPrintEvent OnPrint = {read=FOnPrintEvent, write=FOnPrintEvent};
public:
  __fastcall virtual TQRCustomControl(Classes::TComponent* AOwner);
  virtual void __fastcall Paint(void);
  virtual void __fastcall Print(int X, int Y);
  virtual void __fastcall Stretch(int &size);
  __property AutoSize  = {stored=true, default=1};
  __property TQuickReport* ParentReport = {read=FReport, write=FReport, nodefault};
__published:
  __property Height ;
  __property Left ;
  __property Top ;
  __property Visible ;
  __property Width ;
public:
  __fastcall virtual ~TQRCustomControl(void) { }
}
```

There will typically be two types of components. If you want to print some text, you will override the Paint() and Print() methods, setting the Caption property (inherited from TCustomLabel) to whatever text you want to print before calling the *inherited Print()* or *Paint()*.

Graphical controls will probably not call the inherited methods but rather provide their own drawing and painting. Printing is accomplished by writing to the QRPrinter->Canvas at the coordinates passed to the Print() method. Before any coordinate value can be used you must translate them into the correct values for the current form resolution. QuickReport takes care of this for you in the QuickReport->XPos() and YPos() functions. The QuickReport component you will use can be referenced through the ParentReport property:

```
ParentReport->QRPrinter->Canvas->Draw(ParentReport->XPos(X),
   ParentReport->YPos(Y), ParentReport->XPos(X+Width),
    ParentReport->YPos(Y+Height), ABitmap);
```

If your control will have a dynamic vertical size you must also override the Stretch() method. This method should return the *additional* vertical size of the control for the instance about to be printed.

Several printable components with source code are available from our web site at *http://www.qusoft.no*.

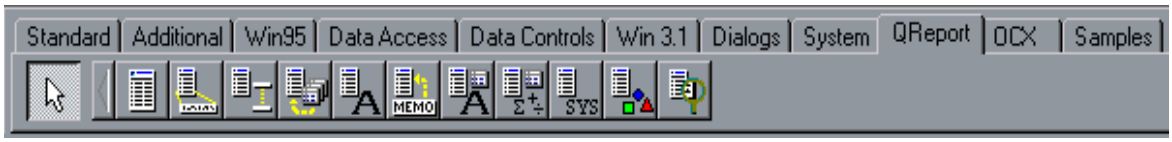
NOTE: If you have a printable component written for QuickReport version 1.0b or earlier and you reference the XPos() and YPos() functions you must probably change your code for version 1.0c and later. The X and Y coordinate passed to the Print method are now not already converted to correct coordinates so if your code looked like this:

```
ParentReport->QRPrinter->Canvas->Draw(X,Y,X+ParentReport->XPos(Width),
   Y+ParentReport->YPos(Height), ABitmap);
```

It should be changed to:

```
ParentReport->QRPrinter->Canvas->Draw(ParentReport->XPos(X),
   ParentReport->YPos(Y), ParentReport->Pos(X+Width),
  ParentReport->YPos(Y+Height), ABitmap);
```

QuickReport Reference



QRBand


QRBands make up the different parts of your report, such as Detail Band for the data part, Page Header band for page headers and so on. QuickReport then replicates these bands everywhere they are supposed to appear.

Some band types are printed automatically by QuickReport, while others need to be connected to a QRGroup or QRDetailLink component to be printed. For the bands that print automatically you can have multiple bands of the same type. QuickReport will print them all (those with the Enabled property set to true) in their creation order. If you want to rearrange the order that they print, use the Send To Back() and Bring To Front() methods of the C++Builder form designer.

__property TAlign Align;

The Align property of a QRBand is usually set to alTop, making the bands appear after one another in the form designer. If you want to visually rearrange the order of the bands you can do so by changing the properties to alNone, rearrange them, then set Align back to alTop. The order of the bands in the form designer has no impact on printing order.

__property TQRBandType BandType;

The BandType property is used to tell QuickReport how to handle each band. There are several different band types, and picking the right one for each band is important. When you connect a band to a QRGroup or QRDetailLink component the BandType is automatically set to the right type. The possible values of BandType are:

rbTitle	Report title band, printed once at the beginning of the report. You can have several bands of this type and they will all be printed one after another. Title bands are printed automatically
rbPageHeader	Page Header, printed once at the top of each page. You can have several bands of this type and all will be printed after another on top of the page. Page header bands are printed automatically.
rbColumnHeader	The Column Header band is much like the Page header, except that it is printed on top of every column in a multi-column report. The Column Header bands are printed automatically
rbDetail	Detail bands are replicated for all records in your dataset. You can have many detail bands but for basic reports you usually have only one. The detail bands are printed automatically
rbPageFooter	The Page Footer band is printed at the bottom of each page. The page footers are printed automatically

rbOverlay	Overlay bands are printed as an overlay of the page. It is always positioned from the top left corner, independent of all other bands on the page.
rbGroupHeader	Group header bands are usually used as group headers in master/detail reports. They must be connected to a QRGroup component or a QRDetailLink component to be printed.
rbSubDetail	Sub Detail bands are used as detail bands for the detail tables in a master/detail report when you use the QRDetailLink component and must be connected to this component to be printed
rbGroupFooter	Group footers are used mostly as footers in master/detail reports. They must be connected to a QRGroup or QRDetailLink component to be printed.
rbSummary	Summary bands are automatically printed at the end of the report.

__property TColor Color;

Sets the background color of the band

__property bool Enabled;

If set to false, the band will not be printed. This property can be changed during report creation to print different bands depending of the data being printed.

__property TFont Font;

The parent font to use for controls on this band.

__property bool ForceNewPage;

If true, a page break will be made before this band is printed. This can be used if you want a group header to always appear on a new page or any other situations. Please see also the QuickReport->NewPage() method.

__property TQRFrame* Frame;

The pen to be used to frame the band. Set Pen width to 0 for no frame (default)

__property Extctrls::TCustomPanel* LinkBand;

The LinkBand property is used to make sure several bands always appear on the same page. When QuickReport checks if there is enough space left on the page to print the current band, it will include the size of a band connected to this property. This band can again be connected to another band, and so on. Please note that you should always link to the *next* band on the report, never backwards to the previous.

__property bool ParentFont;

The ParentFont property determines where a band looks for its font information. If ParentFont is true, the Band uses the font in its parent form's Font property. If ParentFont is false, the Band uses its own Font property.

By using ParentFont, you can ensure that all the bands on a report have a uniform appearance. For example, if you want all the bands in a form to use 12-point Courier for their font, you can set the form's Font property to that font. By default, all the controls on report form will use the same font.
To specify a different font for a particular control, specify the desired font as the value of the band's Font property, and ParentFont becomes false automatically.

__property TQRRuler Ruler;

QuickReport can paint a ruler in the background of the band during design time. This will help you position the controls on the bands. One of the following values can be set:

qrrNone	No ruler is displayed
qrrInchesH	Horizontal ruler lines every inch
qrrInchesV	Vertical ruler lines every inch
qrrInchesHV	Horizontal and vertical ruler lines every inch
qrrCmH	Horizontal rulers every centimeter
qrrCmV	Vertical ruler lines every centimeter
qrrCmHV	Horizontal and vertical ruler lines every centimeter

typedef void __fastcall (__closure *TQRBandAfterPrintEvent)(bool BandPrinted);

__property TQRBandAfterPrintEvent AfterPrint;

This event is called whenever the band has been printed or tried to be printed but stopped in the BeforePrint event. The BandPrinted parameter will tell if the band was actually printed or not.

typedef void __fastcall (__closure *TQRBandBeforePrintEvent)(bool &PrintBand);

__property TQRBandBeforePrintEvent BeforePrint;

The BeforePrint event is called before the band is about to be printed. You can set the PrintBand parameter to false if you want the band to be skipped for this time.

QRCustomControl

The QRCustomControl object is the base class for all printable QuickReport components. It's not found on the Component palette; instead you subclass it if you want to create your own printable controls. There are three methods you need to override to create a printable control.

virtual void __fastcall Paint(void);

The Paint() method is used to draw the component at *design time*. That is in the C++Builder Form designer, not in the design-time preview. If you want to print anything but basic text, you must override this method.

virtual void __fastcall Print(int X, int Y);

The Print() method is used to print the method onto the QRPrinter->Canvas. If you want to print anything but basic text you must override this method.

virtual void __fastcall Stretch(int &size);

If your component has a variable height you must override the Stretch() method. Set the Value parameter to the *extra* height your component will occupy, that is, not including the value of the Height property.

QRDBCalc

 The QRDBCalc component eases the job of doing basic calculations during report generation. It can connect to a database field and calculations can be reset after the printing of any particular band.

__property bool AlignToBand;

If true, the control will align horizontally to the entire band with, not only in its given rectangle. So if Alignment is taRightJustify and AlignToBand is true, the text will be right aligned on the band (page) instead of just within the control. Similarly, if Alignment is taCenter and AlignToBand is true, the text will be centered on the band (page). The default is false.

__property long AsInteger;

Returns the current value of the QRDBCalc component as long value.

__property double AsReal;

Returns the current value of the QRDBCalc component as a double value.

__property TFont Font;

The Font property is a font object that controls the attributes of text printed to the report. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font class.

__property TQRCalcOperation Operation;

This property selects the type of calculation for the control to perform. It can be on of the following values:

qrcSUM	Calculate the total of the field
qrcCOUNT	Count the number of times the control is printed
qrcMAX	The maximum value of the database field
qrcMIN	The minimum value of the database field
qrcAVERAGE	Calculates the average of the database field

property TFont ParentFont;

The ParentFont property of a printable control determines if the control will use its own Font property for font selection or if it will get font information from its parent QRBand.

__property System::AnsiString PrintMask;

Use the PrintMast property to format the output of the QRDBCalc component. Look up the FormatFloat routine in C++Builder help for documentation on how the PrintMask works.

__property TQRBand* ResetBand;

Setting the ResetBand property causes the calculation to be reset every time the selected band is printed. The reset is performed after the band is printed.

typedef void __fastcall (__closure *TQRLabelOnPrintEvent)(System::TObject* sender, System::AnsiString &Value);

__property TQRLabelOnPrintEvent OnPrint;

The OnPrint event is called just before the control is printed. The string to be outputted is in the Value property and can be changed. This property is also useful to suppress printing of repeated values.

QRDetailLink


QRDetailLink is used to create different types of master detail reports with QuickReport. Using the QRDetailLink control you can create virtually any type of such reports, including unlimited levels and unlimited number of detail tables on the same level. You connect a QRDetailLink to a QuickReport component or another QRDetailLink using the Master property. In addition you must set up a master/detail relationship between the tables using the MasterSource/MasterField properties of the TTable/TQuery components. For more information on using the QRDetailLink please see *Creating a Master Detail Report* elsewhere in this document.
The QRDetailLink component is not intended to be used together with the QRGroup component. They are two different ways of creating master/detail reports and mixing them might give a wrong result.

__property Db::TDataSource* DataSource;

The DataSource to connect this QRDetailLink to. A master/detail relationship must be set up for this to work. If you don't specify a DataSource you can use the OnNeedData event to print detail information from other sources, such as arrays or text files.

__property TQRBand* DetailBand;

Selects the DetailBand for this DetailLink. The detail band is printed for each record in the DataSource. The BandType of the selected band is automatically set to rbSubDetail.

__property TQRBand* FooterBand;

Selects the band to be used as this QRDetailLink's footer band. The footer band is printed after any detail records for this DetailLink. It might contain any summary information for the detail table. The BandType property of the selected band is automatically set to rbGroupFooter.

__property TQRBand* HeaderBand;

The HeaderBand is printed before the detail records of the DetailLink component. The BandType property of the selected band is automatically set to rbGroupHeader.

__property TQRController* Master;

The Master property connects to a TQuickReport component or another QRDetailLink. A master detail relationship must be setup between the tables involved.

__property bool PrintBefore;

If PrintBefore is false (Default) the QRDetailLink component and its bands will be printed after any detail band of the Master. If true, this QRDetailLink will be printed before the Master information.

typedef void __fastcall (__closure *TQRFilterEvent)(bool &PrintRecord);

__property TQRFilterEvent OnFilter;

The OnFilter event is used for local filtering of tables. The event handler will be called each time the record pointer is advanced through the dataset, allowing you to reject the record by setting the PrintRecord parameter to False. Please see *How do I filter Records* for more information.

typedef void __fastcall (__closure *TReportNeedDataEvent)(bool &MoreData);

__property TReportNeedDataEvent OnNeedData;

The OnNeedData event is used if you want to print detail data from other sources than C++Builder DataSources. To print such a report, leave the QRDetailLink->DataSource property blank and provide a event handler for this event. The event handler would then retrieve the data from whatever source you want and place the result into the Caption property of QRLabels on your report. When there are no more data to print you set the MoreData parameter to False.
For more information on event driven reports please see *Creating Reports from non BDE Datasources*.

QRDBText

 The QRDBText component is a data-aware text control. It can print text fields, numeric fields, and memo fields. If you connect it to a memo field, it automatically stretches to make room for the entire text. The band it's put on will also stretch accordingly. It the memo field is too large to fit on a page, it is clipped.

__property TAlignment Alignment;

The Alignment property specifies how text is aligned within the component.
These are the possible values:

taLeftJustify Align text to the left side of the control
taCenter Center text horizontally in the control
taRightJustify Align text to the right side of the control

__property bool AlignToBand;

If true, the control aligns horizontally to the entire band with, not only in its given rectangle. So if Alignment is taRightJustify and AlignToBand is true, the text will be right aligned on the band (page) instead of just within the control. Similarly, if Alignment is taCenter and AlignToBand is true, the text will be centered on the band (page). The default is false.

__property bool AutoSize;

When the AutoSize property is true, the component resizes to the width and length of the current string.

__property TColor Color;

Selects the background color for the control

__property Db::TDataSource* DataSource;

Selects the DataSource from which you want to print a data field.

__property System::AnsiString DataField;

Selects the Data field to be printed by the control.

__property TFont Font;

The Font property is a font object that controls the attributes of text printed to the report. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font object.

__property Boolean ParentFont;

The ParentFont property of a printable control determines if the control will use its own Font property for font selection or if it will get font information from its parent QRBand.

__property bool Transparent;

The Transparent property determines if the control is transparent. You could place a transparent label or text control on top of a bitmap, and the control won't hide part of the bitmap. If Transparent is True, the Color property (for background color) is ignored.

typedef void __fastcall (__closure *TQRLabelOnPrintEvent)(System::TObject* sender, System::AnsiString &Value);

__property TQRLabelOnPrintEvent OnPrint;

The OnPrint event is called just before the control is printed. The string to be outputted is in the Value property and can be changed. This property is also useful to suppress printing of repeated values.

QRGroup

 The QRGroup component is typically used to print group headers and footers when some data field change during report creation. This can be used to create master detail reports form SQL queries or simply to print a group footer when some data changes. The Simple Report in the demo program demonstrates how a QRGroup can be used to put a spacer band whenever the first letter of the company name changes.

You can nest QRGroup components up to ten levels. Whenever a group breaks, all higher level groups will break too.

__property Db::TDataSource* DataSource;

Select the datasource you want this group connected to.

__property System::AnsiString DataField;

Select the data field you want this group to be connected to. The group will reprint when the value of the selected field change or a lower level group is reprinted.

__property TQRBand* FooterBand;

Selects the band to be used as this group's footer band. The band type of the selected band is automatically set to rbGroupFooter.

__property TQRBand* HeaderBand;

Selects the band to be used as the header band for this group. The BandType property of the selected band is automatically set to rbGroupHeader.

__property int Level;

Sets the level of this group. Group 0 is the lower group (outer) and 9 is the inner group. Please make sure no two groups are on the same level.

typedef void __fastcall (__closure *TReportNeedDataEvent)(bool &MoreData);

__property TReportNeedDataEvent OnNeedData;

The OnNeedData event can be used to modify the value of the group data before the decision is taken whether to break the group or not. The Simple Report in the demo program demonstrates how to use this event to make the group depend on the first letter in a company name instead of the entire field.

QRLabel

 Static text on reports are printed using the QRLabel component. Set the Caption property to the text you want to print. QRLabel with not wrap the text on several lines if it doesn't fit in the designed width. You can also change the caption property during report generation.

__property TAlignment Alignment;

The Alignment property specifies how text is aligned within the component.
These are the possible values:

taLeftJustify Align text to the left side of the control
taCenter Center text horizontally in the control
taRightJustify Align text to the right side of the control

__property bool AlignToBand;

If true, the control will align horizontally to the entire band with, not only in its given rectangle. So if Alignment is taRightJustify and AlignToBand is true, the text will be right aligned on the band (page)

instead of just within the control. Similarly, if Alignment is taCenter and AlignToBand is true, the text will be centered on the band (page). The default is false.

__property bool AutoSize;

When the AutoSize property is true, the component resizes to the width and length of the current string.

__property System::AnsiString Caption;

The caption property contains the text string to be printed on the report.

__property TColor Color;

Selects the background color for the control

__property TFont Font;

The Font property is a font object that controls the attributes of text printed to the report. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font class.

__property bool ParentFont;

The ParentFont property of a printable control determines if the control will use its own Font property for font selection or if it will get font information from its parent QRBand.

__property bool Transparent;

The Transparent property determines if the control is transparent. You could place a transparent label or text control on top of a bitmap, and the control won't hide part of the bitmap. If Transparent is true, the Color property (for background color) is ignored.

typedef void __fastcall (__closure *TQRLabelOnPrintEvent)(System::TObject* sender, System::AnsiString &Value);

__property TQRLabelOnPrintEvent OnPrint;

The OnPrint event is called just before the control is printed. The string to be outputted is in the Value property and can be changed. This property is also useful to suppress printing of repeated values.

QRMemo

The QRMemo control is used to print multi-line text not bound to a data field. QRMemo will *not* stretch to fit the height of the text automatically.

__property TAlignment Alignment;

The Alignment property specifies how text is aligned within the component.
These are the possible values:

taLeftJustify Align text to the left side of the control

taCenter Center text horizontally in the control
taRightJustify Align text to the right side of the control

__property TColor Color;

Selects the background color for the control

__property TFont Font;

The Font property is a font object that controls the attributes of text printed to the report. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font object.

__property Classes::TStringList* Lines;

The Lines property contains all the text of the QRMemo control. It's a standard StringList and you can use methods like Clear and Add to enter text into the control.

__property bool ParentFont;

The ParentFont property of a printable control determines if the control will use its own Font property for font selection or if it will get font information from its parent QRBand.

__property bool WordWrap;

If WordWrap is true lines in the QRMemo will be wrapped to fit within the width of the control. If false, lines will be printed as they are entered in the Lines property and all text might not show.

QRPreview

 Using the QRPreview component it is very easy to make preview forms with your own look and feel. All you need to do to create a basic preview form is to drop a QRPreview on a form, make an event handler for the QRPrinter->OnPreview event and call the ShowModal() method of your preview form. For more information on creating custom preview forms see *Creating a Custom Preview Form* on page 11.

__property long PageNumber;

PageNumber selects the page to show in the preview component. The selected page will automatically be painted when setting this property.

__property int Zoom;

Select the current zoom level with this property. It can be set to virtually any value, setting it to 100 will give you approximately a 1 to 1 reproduction of your page.

void __fastcall ZoomToFit(void);

Calling ZoomToFit will automatically zoom the entire page will fit in the QRPreview control. If you resize the control, the zoom value will be recalculated automatically.

void __fastcall ZoomToWidth(void);

Calling ZoomToWidth will automatically zoom the page so it will fit just in the width of the QRPreview control. If you resize the control, the zoom value will be recalculated automatically.

QRPrinter

The QRPrinter is not a component on the Component palette. Instead, one instance is created on application startup and is used throughout your application. It's the QRPrinter that takes care of all the low level stuff like recording your report as it's created, printing it, calling up the preview and even saving and loading reports.

QRPrinter can even be used independent of the QuickReport component. Using the QRPrinter->Canvas like any other canvas, you can print just about anything, then call the QRPrinter->Preview() method to bring up a preview before printing.

__property bool Canceled;

The QRPrinter->Canceled property tells if the Cancel method has been called or not.

__property Graphics::TMetafileCanvas* Canvas;

The Canvas is actually a Windows metafile used for creating the pages.

__property int Copies;

Number of copies to print. Note that not all printer driver support this option.

__property bool EnableOpenBtn;

If true (default), the Print report button will appear on the standard preview form.

__property bool EnableSaveBtn;

If true (default), the Save report button will appear on the standard preview form

__property int FromPage;

The FromPage property can be used to set the starting page if you want QRPrinter to print only a range of the prepared pages.

__property int LeftWaste;

LeftWaste contains the unprintable area on the left edge of the page. It is only valid during report creation.

__property Printers::TPrinterOrientation Orientation;

Use the Orientation property to select portrait (poPortrait) or landscape (poLandscape) orientation of the page. This property is set automatically when you choose QuickReport print.

__property Graphics::TMetafile* Page;

When the report is prepared, the Page property points to a metafile containing the currently selected page.

__property int PageCount;

The PageCount property returns the total number of pages send to the QRPrinter object in this document.

__property int PageHeight;

QRPrinter->PageHeight is the height of the current printable area, in screen resolution.

__property int PageNumber;

The PageNumber property is used to select what page QRPrinter makes available via the Page property. It is set automatically when you change the value of a QRPreview->PageNumber property. Generally you will not change this property directly, but access the PageNumber property of the QRPrinter object, since this will also force a repaint of the preview.

__property int PageWidth;

QRPrinter->PageWidth is the width of the current printable area, in screen resolution.

__property int PaperBin;

Selects the output bin to be used for the print job. The possible options are:

dmbin_Upper	dmbin_OnlyOne
dmbin_Lower	dmbin_Middle
dmbin_Manual	dmbin_Envelope
dmbin_EnvManual	dmbin_Auto
dmbin_Tractor	dmbin_SmallFmt
dmbin_LargeFmt	dmbin_LargeCapacity
dmbin_Cassette	

__property int PaperLength;

Select a non zero value for this property will set up a custom paper length. The unit is $1/10^{th}$ mm, making a value of 254 equal one inch. Setting this property automatically sets the PaperSize property to rpCustom. Selecting a predefined paper size will set this property to 0 again. You should always set the PaperWidth property if you set PaperLength. Setting the QuickReport->PaperLength property will automatically set this property when the report is printed.

__property TQRPaperSize PaperSize;

Selects the paper size to be used for this print job. Setting this to other than qrpDefault will override the setting in your printer driver. Note that not all printer drivers support all paper sizes. A paper size selected for a QuickReport component will automatically be carried over to QRPrinter when the report is printed.

qrpDefault	Use the default paper size for the currently selected printer driver.
qrpLetter	Letter size paper (8 1/2 x 11 in)
qrpLegal	Legal size paper (8 1/2 x 14 in)
qrpA3	A3 size paper (297 x 420 mm)

qrpA4	A4 size paper (210 x 297 mm)
qrpA5	A5 size paper (148 x 210 mm)
qrpCustom	Use the paper size set by the PaperWidth and PaperLength properties

__property int PaperWidth;

Select a non zero value for this property will set up a custom paper width. The unit is $1/10^{th}$ mm, making a value of 254 equal one inch. Setting this property automatically sets the PaperSize property to rpCustom. Selecting a predefined paper size will set this property to 0 again. You should always set the PaperLength property if you set PaperWidth.

__property System::AnsiString PreviewCaption;

Sets the caption of the standard preview form. Note that this property is read only in the shareware version.

__property bool PrinterOK;

If a printer is installed and available on your system the PrinterOK property is true, otherwise it's false.

__property System::AnsiString ProgressCaption

Sets the caption of the progress form. Note that this property is read only in the shareware version.

__property bool ShowProgress;

If True, a progress bar will appear during report printing, saving and loading. This property is automatically set by the QuickReport component.

__property TQRPrinterStatus Status;

Read Only. The Status property tells the current status of the QRPrinter class. As long as you don't use the QRPrinter directly you should not have to worry very much about the setting of this property. The possible values are:

mpReady	The QRPrinter object is ready to accept new jobs (BeginDoc())
mpBusy	The QRPrinter is currently active receiving a job
mpFinished	A job is finished (EndDoc() is called) and can be previewed, printed or saved
mpPrinting	A job is currently being sent to the printer
mpPreviewing	A preview is currently active

__property int Thumbs;

The Thumbs property selects how many report pages are to be printed across each sheet of paper. The default is one, causing one report page to be printed on each sheet. If you set it to two, four report pages (2x2) will be printed, three will print nine (3x3) and so on.

__property System::AnsiString Title;

The title property is displayed in the Print Manager and in any network printer queue. It's set automatically by the QuickReport component if you not printing directly to the QRPrinter yourself.

__property int ToPage;

The ToPage property can be used to set the ending page if you want QRPrinter to print only a range of the prepared pages.

__property int TopWaste;

TopWaste contains the unprintable area on the top edge of the page. It is only valid during report creation.

__property int TotalPageHeight;

TotalPageHeight contains the total paper height, including the unprintable areas at top and bottom. The value is in screen resolution.

__property int TotalPageWidth;

TotalPageWidth contains the total paper width, including the unprintable areas at left and right edge. The value is in screen resolution.

__property bool ZoomButtons;

If true (default) the standard preview form will have three buttons for selecting the zoom level, Fit In Window, 100% and Page Width. If false, the preview form will instead have a combo box with some more choices.

typedef void __fastcall (__closure *TOnPreviewEvent)(void);

__property TOnPreviewEvent OnPreview

The OnPreview event is used if you want to create your own preview forms instead of using the standard preview. Please see *Creating a Custom preview form* on page 11 for more information on how to use this event handler.

void __fastcall BeginDoc(void);

If you want to use the QRPrinter object directly you must call BeginDoc() before printing anything to the Canvas property.

void __fastcall Cancel(void);

Call the Cancel() method if you want to cancel the current job.

void __fastcall Cleanup(void);

If you use the QRPrinter object directly you must call the CleanUp() method when you are finished with a job to free it from memory.

void __fastcall EndDoc(void);

If you use the QRPrinter object directly call EndDoc() to indicate that you have reached the end of the job and now want to print, preview or save the job.

void __fastcall Load(System::AnsiString Filename);

If you want to load a previously saved report, call the QRPrinter->Load method. The file must be a .QRP file of the correct format.

void __fastcall NewPage(void);

If you use the QRPrinter object directly, you use the NewPage() method to indicate that you want to create a new page. *Do not call this method in any events during a regular QuickReport creation, instead use the QuickReport->NewPage() method which will make sure any page footers and headers are printed correctly.*

void __fastcall Preview(void);

The Preview() method brings up a print preview of the current job. The job has to be completed (EndDoc()) before you can call the Preview() method. Usually you will just call the preview method of your QuickReport component, which will take care of all preparation of the job.

void __fastcall Print(void);

The Print() method prints the current job. The job has to be completed (EndDoc()) before you can call the Print() method. Usually you will just call the Print() method of your QuickReport component, which will take care of all preparation of the job.

void __fastcall Save(System::AnsiString Filename);

If you want to save the current report to a file you can call the QRPrinter->Save() method. The file will be saved as a .QRP file. This is a proprietary file format, but it will be documented in a future version.

QRSysData

The QRSysData component is used to print various system information, like page number, date and time and report title.

__property TAlignment Alignment;

The Alignment property specifies how text is aligned within the component.
These are the possible values:

taLeftJustify Align text to the left side of the control
taCenter Center text horizontally in the control
taRightJustify Align text to the right side of the control

__property bool AlignToBand;

If true, the control aligns horizontally to the entire band with, not only in its given rectangle. So if Alignment is taRightJustify and AlignToBand is true, the text will be right aligned on the band (page) instead of just within the control. Similarly, if Alignment is taCenter and AlignToBand is true, the text will be centered on the band (page). The default is false.

__property bool AutoSize;

When the AutoSize property is true, the component resizes to the width and length of the current string.

__property TColor Color;

Selects the background color for the control

__property TQRSysDataType Data;

The Data property is used to select the information displayed by the QRSysData component. The possible values are:

qrsTime	Current time in the format HH:MM
qrsDate	Current date in the format set by the global ShortDateFormat variable
qrsDateTime	Current date and time separated by a comma
qrsPageNumber	The current page number
qrsReportTitle	The report title as it appears in the TQuickReport->ReportTitle property
qrsDetailCount	Total number of records to be printed in report
qrsDetailNo	Current record number to be printed

__property TFont Font;

The Font property is a font object that controls the attributes of text printed to the report. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font object.

__property bool ParentFont;

The ParentFont property of a printable control determines if the control will use its own Font property for font selection or if it will get font information from its parent QRBand.

typedef void __fastcall (__closure *TQRLabelOnPrintEvent)(System::TObject* sender, System::AnsiString &Value);

__property TQRLabelOnPrintEvent OnPrint;

The OnPrint event is called just before the control is printed. The string to be outputted is in the Value property and can be changed. This property is also useful to suppress printing of repeated values.

__property System::AnsiString Text;

Any text you want to print immediately in front of the data can be put into the Text property. Typically you would set this to 'Printed at : ' for a QRDBCalc component with the Data property set to qrsDateTime.

__property bool Transparent;

The Transparent property determines if the control is transparent. You could place a transparent label or text control on top of a bitmap, and the control won't hide part of the bitmap. If Transparent is true, the Color property (for background color) is ignored.

QRShape

 The QRShape component is used to draw simple shapes on your reports.

__property Graphics::TBrush* Brush;

Selects the Brush to be used for painting the shape. The Brush property has two sub properties, Color and Style. The Brush->Color property selects the color to use when painting and the Brush->Style property selects the actual Brush to use. The possible styles are:

bsSolid	Solid brush with the current color
bsClear	No brush is used
bsBDiagonal	Horizontal lines upwards to the right
bsFDiagonal	Horizontal lines upwards to the left
bsCross	Cross pattern
bsDiagCross	Diagonal cross pattern
bsHorizontal	Horizontal lines
bsVertical	Vertical lines

__property int Height;

The Height of the shape

__property Graphics::TPen* Pen;

The pen to use when drawing the outline of the shape.

__property TQRShapeType Shape;

Use the Shape property to select what kind of shape you want to print. The possible values are:

qrsRectangle	Draws rectangles on the report
qrsCircle	Draws circles and ellipses on the report
qrsVertLine	To draw vertical lines on the report
qrsHorLine	To draw horizontal lines
qrsTopAndBottom	Draws horizontal lines on the top and bottom of the shape
qrsRightAndLeft	Draws vertical lines on the right and left edge of the control

__property int Width;

The width of the shape

QuickReport

 The QuickReport component is your report controller. Its job is to transform your C++Builder form into a report. After you have dropped this component on the form you can call its *Print()* and *Preview()* methods to bring up the report. During design time you can right click the QuickReport component to bring up a menu. From here you can preview your report at design time, without running your application.

Please note that dropping a QuickReport component on your form will resize it to add scrollbars to your form. This is done to make it easier to do report layouts.

__property long PageCount;

Runtime and read only. This property tells the total page count of the report. Note that it is not a valid value before the report is prepared.

__property long PageNumber;

Runtime and Read Only. PageNumber tells the current page number during report creation. This is the same value displayed by a QRSysData with the Data property set to PageNumber.

__property long RecordNo;

Runtime and Read Only. RecordNo indicates the current record number in your (master) table. This is the same value displayed by a QRSysData component with the Data property set to RecordNo.

__property long ColumnMarginInches;

The ColumnMarginInches property sets the spacing put between columns in a multi-column report. The spacing is set in 1/10th of inches. Use this property together with the LeftMarginInches to make a mailing labels report fit on your label layout.

__property long ColumnMarginMM;

This is the same as the ColumnMarginInches property, except the value is set in MM rather than inches.

__property int Columns;

Sets the number of columns to print on the page. The column width is set automatically depended on the Columns property, the LeftMargin and the ColumnsMargin properties.

__property bool DisplayPrintDialog;

If true, a print dialog will be displayed when you call the QuickReport->Print method. *Note that the print dialog will not be displayed if you click the Print button from the standard preview form, regardless of the setting of this property.*

__property long LeftMarginInches;

The LeftMarginInches property sets the left margin of the page in 1/10th of inches. The unprintable area of the printer is added to this value automatically.

__property long LeftMarginMM;

This is the same as the LeftMarginInches property, except that the value is given in MM rather than inches.

__property Classes::TNotifyEvent OnStartPage

The OnStartPage event is called whenever a new page is started.

__property Printers::TPrinterOrientation Orientation;

The Orientation property selects the paper orientation to use when printing the report. The value can be poPortrait (default) or poLandscape. Note that to set this value from your source code you need to include the Printer header, because the TPrinterOrientation type is declared in that header.

__property TQRFrame* PageFrame;

Controls the drawing of a frame around the entire page. You can set the top, bottom, right and left sub-properties to true or false to select what part of the frame you want to draw. The line style can be set with the Style and Color sub-properties.

__property int PaperLength;

Select a non zero value for this property will set up a custom paper length. The unit is 1/10th mm, making a value of 254 equal one inch. Setting this property automatically sets the PaperSize property to rpCustom. Selecting a predefined paper size will set this property to 0 again. You should always set the PaperWidth property if you set PaperLength.

__property TQRPaperSize PaperSize;

Selects the paper size to be used for this report. Setting this to other than qrpDefault will override the setting in your printer driver. Note that not all printer drivers support all paper sizes.

qrpDefault	Use the default paper size for the currently selected printer driver.
qrpLetter	Letter size paper (8 1/2 x 11 in)
qrpLegal	Legal size paper (8 1/2 x 14 in)
qrpA3	A3 size paper (297 x 420 mm)
qrpA4	A4 size paper (210 x 297 mm)
qrpA5	A5 size paper (148 x 210 mm)
qrpCustom	Use the paper size set by the PaperWidth and PaperLength properties

__property int PaperWidth;

Select a non zero value for this property will set up a custom paper width. The unit is 1/10th mm, making a value of 254 equal one inch. Setting this property automatically sets the PaperSize property to rpCustom. Selecting a predefined paper size will set this property to 0 again. You should always set the PaperLength property if you set PaperWidth.

__property System::AnsiString ReportTitle;

Sets the title for this report. The report title will appear in the Print Manager and in any network printer queue. It is also the value printed by a QRSysData component with the Data property set to ReportTitle.

__property bool RestartData;

If true (default), QuickReport will move to the first record in the dataset before printing the report. If false, report creation will start at the current record.

__property bool ShowProgress;

If true (default), a progress form will appear while the report is being prepared and printed. The progress form has a Cancel button the user can press to abort the current process.

__property bool SQLCompatible;

Not all databases allow the retrieval of the number of records in the table, and the BDE will create an exception if the operation is tried with a database that doesn't support it. Setting the SQLCompatible to true will allow QuickReport to work with such databases. Note that the progress gauge will not move during report creation if SQLCompatible is true.

__property bool TitleBeforeHeader;

If false (default), the Page header band is printed first also on page one. If you want the Title band to appear before the page header band on the first page, set this property to true.

void __fastcall PrinterNewPage(void);

Calling the NewPage() method during report creation forces QuickReport to insert a page break at the current position. This can typically be done in the AfterPrint event of a band.

void __fastcall Prepare(void);

The Prepare() method is used to prepare the report and send it to the QRPrinter object. It is called automatically by the Print() or Preview() methods, but you might want to call it yourself if you want to create a report without printing or previewing it. If you do so, remember to call the QRPrinter->CleanUp() method when you are finished using the report.

void __fastcall Preview(void);

Call the Preview() method to bring up a print preview of your report. If you haven't created an event handler for the QRPrinter->OnPreview event a default preview form will be displayed. Otherwise it's up to you to display a preview form, usually one with a QRPreview component on it.

void __fastcall Print(void);

The Print() method will prepare and print your report. If the DisplayPrintDialog property is true, a Print Dialog will be displayed before the report is prepared.

__property Classes::TNotifyEvent OnEndPage

The OnEndPage event is called when QuickReport is about to move to a new page.

__property Classes::TNotifyEvent AfterDetail;

The AfterDetail event is called when the detail bands are printed for a (master) record.

__property Classes::TNotifyEvent AfterPrint;

The AfterPrint event is called when the report creation is finished. You can use it to close any datasets you opened in the BeforePrint event.

__property Classes::TNotifyEvent BeforeDetail;

The BeforeDetail event is called just before the detail bands for a (master) record are printed.

typedef void __fastcall (__closure *TQRReportBeforePrintEvent)(bool &PrintReport);

__property TQRReportBeforePrintEvent BeforePrint

The BeforePrint event is called before any preparation of the report is started. Actually, the operation can be canceled by setting the PrintReport parameter to false.

typedef void __fastcall (__closure *TQRFilterEvent)(bool &PrintRecord);

__property TQRFilterEvent OnFilter;

The OnFilter event is used for local filtering of tables. The event handler will be called each time the record pointer is advanced through the dataset, allowing you to reject the record by setting the PrintRecord parameter to false. Note that the QRDetailLink component has it's own OnFilter event to filter records in a detail section of a report. Please see *How do I filter Records* for more information.

typedef void __fastcall (__closure *TReportNeedDataEvent)(bool &MoreData);

__property TReportNeedDataEvent OnNeedData

The OnNeedData event is used if you want to print data from other sources than C++Builder DataSources. To print such a report you would leave the QuickReport->DataSource property blank and provide a event handler for this event. The event handler would then retrieve the data from whatever source you want and place the result into the Caption property of QRLabels on your report. When there are no more data to print you set the MoreData parameter to false.
Note that the QRDetailLink component also has a OnNeedData event, which can be used to print event driven data in a *detail section* of a report.
For more information on event driven reports please see *Creating Reports from non BDE Datasources*.

QuickReport Registration Form

To purchase QuickReport 2.0 with full source code fill out the registration form bellow and send or fax it to QuSoft AS

PLEASE USE CAPTIAL LETTERS. DON'T USE A RED PEN IF YOU INTEND TO FAX THE FORM.

_____
Date

_____
Company Name

_____
Contact

_____
Postal address

_____
Zip Code

_____
State, Country

_____ _____
Phone Number Fax Number

Send by Postal mail (Add $10) :____ E-mail :____

Printed documentation (Add $10, requires postal delivery):_____

E-mail address :_____

Payment method:

Included Check :____ Included Cash/Money Order :____

Visa :____ MasterCard :____ CompuServe SWREG :____

Credit Card number :_____ Exp. Date :_____

Signature :_____

The registration fee is **$99.00** US and registration can be done by sending in this form. Registered users can download the registered version and source code from Internet or get it by postal mail for an additional US $10.

The CompuServe SWREG number for QuickReport is 8644.
For payment with Credit Card or through SWREG this registration form can be faxed to QuSoft at fax:
+47 22 41 74 91
For Cash or Money order the Registration form must be send together with payment to:
QuSoft AS, Fred Olsensgt. 1, N-0152 Oslo, Norway
Full registration information and an online registration form can also be found at our web site.
Full registration information and an online registration form can also be found at our web site.

QuickReport Mailing List

If you want to receive an e-mail notifications when new versions of QuickReport are released please send a e-mail to *qmail@qsd.no* with *SUBSCRIBE QUICKREPORT* as message subject. Please do not use this address for questions or comments about QuickReport.

Files you can download from our internet server

Over the coming weeks the following will be made available for download from our Web site, *http://www.qsd.no*:

- International versions of the help file and resource files (German, French, Italian)
- The QuickReport FAQ
- Updated 16 and 32 bit versions of QuickReport
- The documentation in Word and HTML format

3rd party QuickReport printable components

Several QuickReport users have created some nice printable components based on the TQRCustomControl class and are willing to share those components with other QuickReport users. You can download these components from our Web site, *http://www.qsd.no* or our ftp server, *ftp.qsd.no*. If you want to submit a component you have created please follow the instructions on the web site.

Credits

QuickReport and the documentation is written by Allan Lochert.

Thanks to hundreds of users on the Internet and CompuServe who have submitted bugs and feature requests, struggling with the not-always-so-good beta versions of QuickReport. Without their help the creation of QuickReport would not have been possible.

Thanks to Magne Nilsen of Vega Software for all the useful hints.

Thanks also to Borland International for their great support and for making the 32 bit version of QuickReport available with C++Builder.