Build Reports using QuickReport 3

for Borland Delphi

Distributed Worldwide by QBS Software Ltd

File: C:\WINDOWS\TEMP\QuickReport 3.doc

Contents

What is QuickReport 3?

QuickReport 3 is a set of Delphi components designed to let you produce database output quickly and easily. As well as allowing you to fling together printed reports, QuickReport lets you create print previews where the user can check the result of a printout without wasting paper, and export data to other file formats, such as plain ASCII, comma separated values (CSV) and HTML.

QuickReport is itself written in Delphi and knows all about the Delphi model for handling databases. So you can use it to report on traditional BDE-based databases such as Paradox and dBase, client datasets used in multi-tier environments, the new Delphi 5 ADO and Interbase Express components and third party alternatives such as Apollo. You can even use QuickReport formatting facilities to print out non-database data, if you need to.

This manual is designed to get you up to speed quickly with QuickReport, so that you can start to use it in your own applications at once.

There's more

QuickReport is a fine product – but if you need even more versatility, you might consider upgrading to QuickReport Pro. Naturally, the Pro version is offers everything in the standard product plus:

- Three extra export filters:
  Excel XLS: The XLS filter is compatible with Excel 4 and later, and provides a simple and robust mechanism for exporting unformatted data into spreadsheets.
  Rich Text RTF: The RTF filter, based on Microsoft's RTF version 1.5 spec, supports more RTF features than *TRichEdit* itself.
  Windows Metafile WMF: The WMF filter lets you capture report output in a convenient graphical format.
- Some powerful extra components. Let the user do the work:
  TQREditor is an end user report editor that you can ship royalty-free with your app.

TQuickAbstractRep is a descendant of the *TCustomQuickRep* base class that does not use *TDataset* - use it to build your own report systems.
TQRLoopBand prints the number of times set in its *PrintCount* property - great for creating blank forms.
TQRListWizard will create an instant report based on the fields of a table.

- Expert technical support via email.
- Full source code. *Use the source, Luke!* The user can easily modify the code to localise the language, adopt it to local interface standards, add new features and so on.
- More demos with more depth, including examples of how to make use of all the Pro edition features, and advanced techniques such as writing custom functions for the expression evaluator.

You can upgrade to QuickReport Professional by ordering from our web site, that of our distributor QBS Software Ltd at http://www.qbss.com or from your local Delphi add-on reseller.

A first report

The best way to get the hang of the QuickReport library is to see it in action. So this section explains how to set up a very basic report. With the Delphi IDE running, follow these steps:

1. Choose File | New Application.
2. Drop a *TTable* component onto the main form.
3. Use the Object Inspector to set its *DatabaseName* property to 'DBDemos', *TableName* to 'CUSTOMER.DB' and *Active* to *True*.
4. Drop a *TQuickRep* component on the main form. Its size and position don't matter.
5. Set its *DataSet* property to 'Table1'. This is a key step. The report object to iterates through all the records in it *DataSet*, in this case *Table1*, whenever it is printed or previewed.
6. If necessary, expand the *Bands* property in the Object Inspector by clicking on the + symbol to its left. Set the *HasDetail* item to *True*. You will see the detail band appear inside the report; changing the property actually creates the *DetailBand1* object.
7. Drop a *TQRDBText* component onto the newly created detail band.
8. Set its *DataSet* to 'Table1' and *DataField* to 'Company'.

At this point your form should look something like Figure 1 below.

Figure 1 – Setting up a basic report

To check that you have set up the properties correctly, preview the report by right-clicking somewhere on the *TQuickRep* component and selecting the Preview item from the popup menu. If you did everything right you should now see a preview window containing your report, as shown in Figure 2.

Figure 2 – The preview window

If all has gone well, you now have a report that works at design time. Of course all may *not* have gone well. If you are now mournfully gazing at an entirely blank report, please check that you have completed all the steps – a likely explanation is that you forgot to set *TTable1*'s *Active* property to *True*. Similarly, if you are looking at a report with only one line – 'Kauai Dive Shoppe' – the problem is probably that you failed to connect *QuickRep1*'s *Dataset* property to *TTable1*.

One other problem which may bite you is that the buttons on the toolbar above the report preview area fail to appear. This is nobody's fault: you have become a victim of what the manufacturer of your PC's operating system is pleased to call, in its technical documents, 'DLL Hell'. Specifically, your machine's copy of the common control library (comctrl32.dll) is before 4.72, and needs updating.

You can download a later version of comctrl32.dll from the Microsoft website at http://www.microsoft.com. But since this is one of those files that invariably turns up in new versions of Internet Explorer and Windows Service Packs, you may well find it on one of those CDs that they give away with PC magazines, and save a download. (In fact, it is unlikely that this bug will bite you the developer. We describe it here so that you will recognise the problem if one of your users is caught by it.)

Now lets make the report work as part of a compiled program. You need to write code to call *TQuickRep.Preview*:

1. Drop a button to your form and set its *Caption* property to 'Preview'
2. Double click on the button to add an *OnClick* event. Add a line of code, so that it looks like this:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  QuickRep1.Preview;
end;
```

Now run your application and click the Preview button. As before, you should see the preview window appear. If you want to try printing the report directly to the default printer, simply change the call to the *Preview* method to a call to *Print*, ie

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  QuickRep1.Print;
end;
```

At this point I must admit to taking a slightly dirty shortcut. Our test application a *TQuickRep* component on its main form and, as you can see, this looks pretty odd. In real applications you never display a form containing a *TQuickRep* component. Instead you use them from other forms.

So what we should really do to finish off, if this little example were going to be a real application, is:

1. Create another form – it will be called *Form2*
2. Make the new form into the main form of the project by setting Project | Options | Main form to *Form2*
3. Drop a button on *Form2*

4. Write code like this in the button's event handler
```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Form1.QuickRep1.Preview;
end;
```
5. Compile the project. The compiler will complain that *Unit1* is not in *Unit2*'s *Uses* list, and offer to fix the code. Accept the offer.

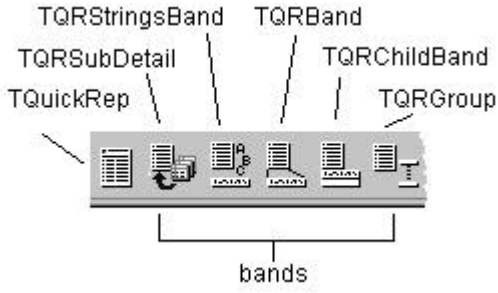
The application should now compile and run, and looks prettier and more 'realistic'. The end user doesn't get to see any bewildering *TQuickRep* components.

But doing this aesthetic polishing doesn't get us any further with QuickReport. So I am going to leave out the need to have a second form from all the examples from this point onwards, and trust you will remember when making real applications.

The components

The QuickReport components are all contained in the QReport tab of the Delphi component palette. Here is a whistle stop tour of what they are and what they do to help you get your bearings.

Figure 3 - TQuickRep and band components



TQuickRep. This is the most important component of them all, a container for all the other printing components. It represents the paper on which your report will be printed. Its *Page* property lets you set up the dimensions of the paper you are going to print on, while the *Dataset* property specifies a source of data that the report will iterate through.

Note that, instead of dropping a *TQuickRep* component onto an ordinary form, you can instead add a *TQuickReport* module to your project:

1. Choose File | New… to display the New Items dialog box.
2. Choose the New tab
3. Select the Report item (middle of the bottom row)

A *TQuickReport* is a bit like a *TDataModule* – it is a specialist type of form, which is never displayed to the user. If you wish you can use *TQuickReport* pseudo-forms instead of *TQuickRep* components on ordinary forms – there is no difference in their methods, properties and events. But we recommend, from experience, that you put a *TQuickRep* component on a form: it's the more versatile approach. For example, having the *TQuickRep* component on a form lets you use the form's *OnCreate* event if you want to create additional objects to be used by the report programmatically.

Band components

These are also container components, representing horizontal strips across report. Bands can be associated with a physical position on a page – for example the top – and also reflect the master/detail relationships in the database that is being displayed. For example, in the same way that there might be many sales records for a given customer record, so a band containing data about an individual sale might appear many times for each occurrence of a band containing customer data.

TQRSubDetail. This is the detail band in a master/detail relationship. You can also make it the master of another detail band, and so create multiple levels of subdetails.

TQRStringsBand. This band provides one mechanism to report on data without using a *TDataSet*. It encapsulates a *TStrings* container; instead of retrieving a sequence of records from a database, it retrieves a sequence of strings from its container.

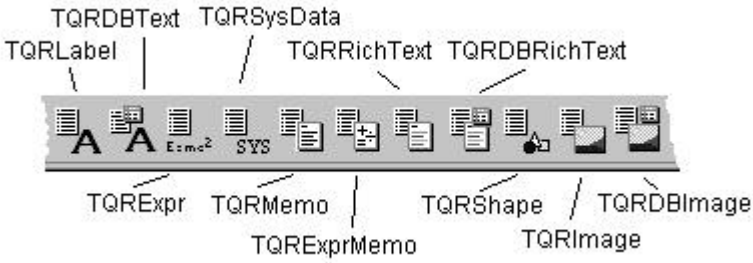
TQRBand. A generic band type, which can act in different roles according to it *BandType* property. Usually there is no need to drag a *TQRBand* onto a report. Instead use the Bands property of *TQuickRep*, which creates *TQRBand* objects and sets their band type in one go.

TQRChildBand. Use *TQRChildBand* objects when you need to extend an existing band. For example, suppose you have placed some *TQRMemo* components in a band, and wish to add, say, a *TQRLabel*, which should always appear below. Since *TQRMemo* objects can expand themselves according to their contents, it is not sufficient to arrange the label within the band. Instead add in a *TQRChildBand* object, and put your label on that. The easiest way to add a child band, by the way, is to double-click the *HasChild* property of its parent in the Object Inspector.

TQRGroup. A band that prints whenever an expression changes, usually a database field. This band is used to group like records together. For example, given a database table containing US addresses, one could sort them by a State Code field and add a group band with it Expression property set to the State Code field. When the report is printed, the contents of the group band will be printed between records belonging to a given state.

Printable components

Figure 4 - Printable components



The QuickReport printable components are mostly equivalents of standard controls that you use on forms. These are the controls, which actually get printed on the paper. Position them within bands to define the layout of your report.

TQRLabel. Place some static text on the page.

TQRDBText. Equivalent of a *TDBText* control – use it to display the contents of a linked database field. Unlike ordinary data-aware controls, but in common with all QuickReport controls, *TQRDBText* uses a *DataSet* property to specify its source of data. Normal data-aware controls use a *DataSource* property, which requires you to supply an extra *TDataSource* component to 'wire' controls to a dataset. QuickReport controls have no such requirement.

TQRExpr. Use this to display an 'expression'. Typically you use one of these when you need to massage the appearance of your data before printing it out. The best way to think of these is as *ad hoc* calculated fields, used only in the report. For example, you might use it to concatenate the parts of a customer name, held in a customer table as string fields called "Title", "Forename" and "Surname". To do this simply set the Expression property of *TQRExpr* to

```
Title + " " + Forename + " " + Surname
```

In real life, you would probably use a more complex expression to cope with blank fields elegantly, but you get the idea.

TQRSysData. A control to display 'system data', by which we mean things like the current page number within the report, and the current date and/or time.

TQRMemo. Very much like its standard control cousin the *TMemo*; use this to display multiple lines of text. As you would expect, the text to be printed is held in a *TStrings* type property called *Lines*.

TQRExprMemo. A composite of *TQRExpr* and *TQRMemo*. You can use this to include {braced} expressions in multi-line blocks. This makes it an absolute natural for doing addresses, especially since it includes a boolean property *RemoveBlankLines*. For example:

```
Company : {CompanyName}
Address : {Address1}
          {Address2}
Contact : {Contact + ' ' + Phone number}
```

TQRRichText. Place some rich text (ie multi-line text with RTF formatting) on the page. One use of this component is to print the contents of a *TRichEdit* control – simply assign it to *TQRRichText's ParentRichEdit* property.

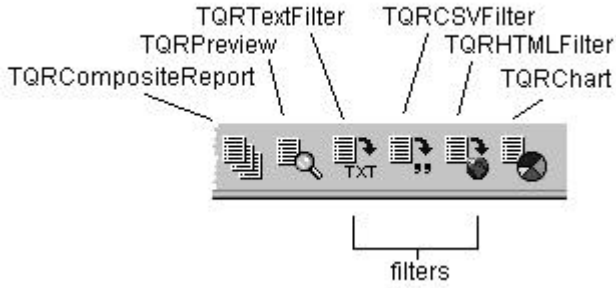
TQRDBRichText. As you'd expect, this is a data-aware version of *TQRRichText*. Use it to print formatted memos stored in BLOB fields.

TQRShape. A cousin of the little-used *TShape* control from Delphi's 'Additional' palette. Actually the QuickReport version is very useful for placing 'furniture' into report layouts such as dividing lines above totals and grouping rectangles.

TQRImage. Display a picture or logo on a report using this control. Supports the same vector and bitmap image formats as *TImage*, and can be loaded at design time using the *Picture* property.

TQRDBImage. A data-aware image control for displaying images stored in BLOB fields in the database.

Figure 5 - Filters and miscellaneous components



TQRCompositeReport. Sometimes you need to group together separate reports into a single print run. For example, maybe you need to print out all the new customers obtained in the last week, together with a summary of all orders in the last week and also a list of stock that needs reordering. As far as your customer is concerned these things belong together and should be printed together. But from the database point of view you will want to use three separate *TQuickRep* components to do the job.

The way to handle this situation is to use a *TQRCompositeReport* component. Drop one on the form where you want to kick off the printing. First you need to define a handler for its *OnAddReports* event, which calls the *TQRCompositeReport.Add* method to add all the *TQuickRep* components you need to print. Suppose the reports you want to print are held on forms called *RepNewCust*, *RepOrderSummary* and *RepStockReorder*, and in each case the *TQuickRep* component on the form is called 'Report' (see the section 'TQuickRep in detail' below for why you might do this). Then your *OnAddReports* event handler should look like this

```
procedure TForm1.QRCompositeReport1AddReports(
  Sender: TObject);
begin
  QRCompositeReport1.Reports.Add(RepNewCust.Report);
  QRCompositeReport1.Reports.Add(RepOrderSummary.Report);
  QRCompositeReport1.Reports.Add(RepStockReorder.Report);
end;
```

(If you don't mind using the *with* statement in your code, you can tidy up this fragment considerably by wrapping it up in

```
with QRCompositeReport1.Reports do
begin
  ...
end;
```

and knocking out the ugly repetitive *QRCompositeReport1.Reports* from the middle three lines.)

Now you can call *QRCompositeReport1.Print* to print out all three reports in a single batch, and *QRCompositeReport1.Preview* to preview them together. There are also T*QRCompositeReport* component properties that let you set up paper sizes and set an overall title for the composite report – basically everything you need to handle the output from the multiple reports in one place.

TQRPreview. To preview a report before it is printed for real, all you need do is call *TQuickRep.Preview* and a standard preview window will appear. There are times, however, when you want to have more control over the exact appearance of the preview.

The *TQRPreview* control lets you do this. Drop it on one of your own forms and, after you have added a line of code to the *TQuickRep.OnPreview* event, the control will act as a frame for the previewed report. If you are more ambitious, or you want to change the preview of a composite report, you can register your own preview form as the default. See the section '**Error! Reference source not found.**' later on for details.

Filters

Sometimes, instead of printing or displaying it directly, you need to export data from your database to another format. QuickReport comes complete with three filter components that let you do this quickly and easily for their respective formats. Simply drop the export filter onto the same form as the report, and the file format appears in the drop-down list of the Save To file dialog in the preview. Registration is automatic, and you don't need to code a thing! (Don't worry - you can export a report programmatically too if you wish – see below.)

Note that not all printable components are exported by filters. Specifically, only the contents of the following text-producing components appear in exported data: *TQRLabel, TQRDBText, TQRExpr, TQRMemo, TQRSysdata* and *TQRExprMemo*.

TQRTextFilter. 'Text' format: exports the contents of the report in plain ASCII, using spaces to separate fields.

TQRCSVFilter. CSV format: exports the report as 'Comma Separated Variables'. As well as using a comma to separate fields, this filter places "double quotes" around them, which allows you to have commas within the fields themselves. This format is easily imported into spreadsheets such as Microsoft Excel. By the way, the component has a *Separator* property that specifies the character used to separate the fields. By default this value is set to ',' comma, but it can be changed to match your requirements.

TQRHTMLFilter. HTML format: exports the report to a HyperText Markup Language file, as used in web browsers, some emailers, help systems and many other places.

It is also possible to call filters explicitly from code. This fragment uses the HTML filter.

```
quickrep1.ExportToFilter(
 TQRHTMLDocumentFilter.Create('c:\report.txt'));
```

To use the Text or CSV filters in this way, use the same *ExportToFilter* call but instantiate a *TQRAsciiExportFilter* or *TQRCommaSeparatedFilter* object as appropriate.

Chart

TQRChart is a version of *TChart* adapted to work with QuickReport. This allows you to add complex charts to your reports – the combination is very powerful indeed. *TQRChart* is used in the same way as the ordinary *TChart* control – double click it to bring up its extensive property editor. For details of how to accomplish tasks such as setting up series and adjusting the look of a chart, please see the TeeChart documentation.

Version incompatibility

Because of dependency issues beyond our control, certain versions of TeeChart are incompatible with newer versions of QuickReport, and the *TQRChart* control and the whole of TeeChart can be unloaded when you upgrade QuickReport. For example, at time of writing the Delphi 3 version of QuickReport 3 (or higher) will not work with TeeChart, because the version of TeeChart that is shipped with Delphi 3 is coded to depend on the QuickReport 2 package. A workaround is to download the (free) TeeChart 4 evaluation version from the TeeMach site at http://www.teemach.com/.

This issue extends to the Decision Cube components found in Client/Sever and Enterprise Editions of Delphi – these depend on TeeChart, and get unloaded when it does. At present, there is no way to use the Delphi 3 Decision Cube with both QuickReport 3 and TeeChart.

We do apologise for this unsatisfactory situation. Since it is caused by design decisions made by other parties in code we cannot access, so that we are not able to fix matters autonomously. If you run into trouble when upgrading QuickReport, please check our website http://www.qusoft.no/ and TeeMach's for the latest information.

Creating reports

The first step when creating a QuickReport is to create a form to store your *TQuickRep* component. We refer to this form as a 'report form' since it's just a container for a report component and is not shown to the end user at runtime. It's a good idea to adopt a naming convention for such reports so that they are easily identifiable in the project manager and in directory listings. For example, you could prefix all report form names with 'rep' or 'rp' to make them stand out. You might want to use a similar scheme with form and data module unit names.

TQuickRep in detail

The next step is to drop a *TQuickRep* component onto the form. Another useful convention you may like to adopt: by naming all *TQuickRep* components 'Report', you can reference them as *repCustomerListing.Report*, *repSalesListing.Report* and so on.

Units and Zoom properties

When dropping the *TQuickRep* component on a form you will se a grid to act as a guide for positioning components. The grid is shown in the current QuickReport *units*. Select the currently active unit by changing the *TQuickRep.Units* property in the property inspector. The grid will be updated when you change this property.

Figure 6 - Adjusting the Units property to alter grid spacing

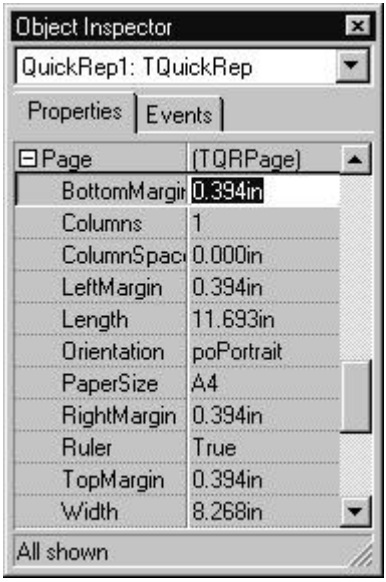
With *Units* set to 'MM', the grid displays at 10mm intervals; if it is set to 'Inches' then the grid displays at 1" intervals. Using the grid you can produce very accurate report layouts, positioning and sizing QuickReport components to 0.01" or 0.01mm.

Usually your screen is too small to display an entire *TQuickRep* component, since it sizes itself based on the actual paper size selected. To get a clearer picture of the whole report, change the *Zoom* property to 50% or less. Changing the zoom causes the *TQuickRep* component and all the printable controls it contains to be redrawn at once to the requested scale. This feature can also be used to enlarge important details for accurate positioning and sizing.

Paper size and margins

You can set up page layout accurately by expanding the *Page* property of the *TQuickRep* component. Double click on the + sign to the left of 'Page' in the Object Inspector to expand the sub properties. You will now see all the settings controlling the page layout.

Figure 7 - Page sub properties



The values given are in the currently selected *Units*, in this case inches. The margin settings can be seen as blue dotted lines on the *TQuickRep* component. All bands are sized to fit inside the margins.

The sub properties are described in Table 1 below:

Table 1 - Sub properties of Page

Property	Type/Values	Notes
BottomMargin	*Extended*	*Units* property determines interpretation.
Columns	*Integer*	Number of columns when printing multi-column detail bands.
ColumnSpace	*Extended*	Space inserted between each column in a multi column report. *Units* property determines interpretation.
LeftMargin	*Extended*	*Units* property determines interpretation.
Length	*Extended*	Read-only, unless *PaperSize* is set to *Custom. Units* property determines interpretation.
Orientation	*TPrinterOrientation = (poPortrait, poLandscape)*	
PaperSize	*TQRPaperSize = (Default, Letter, LetterSmall, Tabloid, Ledger, Legal, Statement, Executive, A3, A4, A4Small, A5, B4, B5, Folio, Quarto, qr10X14, qr11X17, Note, Env9, Env10, Env11, Env12, Env14, Sheet, DSheet, ESheet, Custom)*	These are all the default paper sizes supported by Windows. To use another paper size, set this property to *Custom* and set *Length* and *Width* appropriately – but see also note below.
RightMargin	*Extended*	*Units* property determines interpretation.
Ruler	*Boolean*	Enables display of grid.
TopMargin	*Extended*	*Units* property determines interpretation.
Width	*Extended*	*Units* property determines interpretation.

Not all printer drivers support setting custom paper sizes through the *Custom* setting of *Papersize*. In these cases you must select 'Custom paper size' in the printer driver's own dialog (accessed from the Windows Control Panel) and define the paper's dimensions there. Set this custom paper size to be the default paper size for that printer and finally set the *TQuickRep.Page.PaperSize* property to *Default*. Your custom size will now be picked up at runtime.

Alternatively, and perhaps more robustly, use the next largest standard paper size, and set the margins to keep the printing within the custom area.

Selecting a font

As you would expect, you can set the default font for your report in the *TQuickRep.Font* property. Double click on the property to get the standard Delphi font dialog.

The fonts listed are the Windows system fonts, True Type fonts and any PostScript fonts (if Adobe TypeManager is installed). You can use any combination of fonts in your reports but we advise the use of TrueType or PostScript fonts if you intend to allow the user to preview the report. The system fonts do not scale very well in preview.

Some dot matrix printers print much faster if you select a font already build into the printer hardware, called a 'printer font'. Such fonts are not listed by the font dialog, but can be set programmatically:

```
repCustomerListing.Report.Font.Name := 'CG TIMES';
```

The readability of your report depends very much on your font selection. You should consider this carefully when selecting fonts. Using many different fonts, colours and styles in a report can easily make it look cluttered and difficult to read.

Title and Description

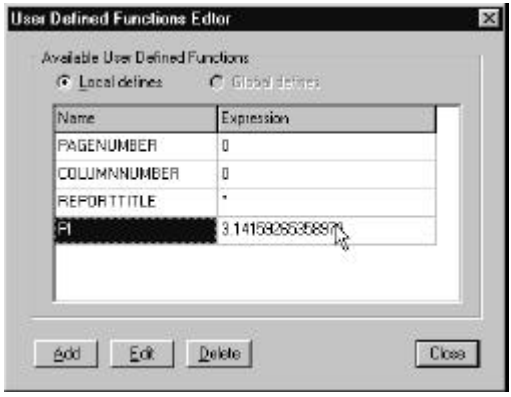
The *TQuickRep* component has *Title* and *Description* string properties to identify and describe the report. These are provided for your convenience, so you can make report selection a data-driven procedure. For example, you might have generate a menu that lists all your reports by title and shows the description when the user selects a report. An example of this can be seen in the QuickReport example project.

The *Title* property can be printed on the report itself using a *TQRSysData* component.

Functions

The *Functions* property of a *TQuickRep* allows you to set up constants and functions that can be used by QuickReport expressions contained in any *TQRExpr*, *TQRExprMemo* and *TQRGroup* components that you drop on the report. Double-click the property the '…' button in the object inspector to bring up the special property editor:

Figure 8 - Functions property editor



Use this dialog, and the expression builder that underlies it, to define constants that you expect to require in multiple expressions. For example, you can see that in Figure 8 above I have defined, with a regrettable lack of originality, the constant `PI` as 3.14159265358979. The other functions you see, `PAGENUMBER`, `COLUMNNUMBER` and `REPORTTITLE`, are automatically predefined.

Working with bands

QuickReport is a *banded* report generator. If you are unfamiliar with banded report generators you can think of them as small paper templates, which are arranged horizontally on a page and filled with data. Different templates are copied into different parts of the page/report. The printable components, *TQRLabel, TQRDBText* and so on, are designed to be placed on these bands. Placing these components directly on the report is not supported.

The easiest way to add bands is via the *TQuickRep.Bands* property in the Property Inspector. Click the '+' sign to the left of the word 'Bands' to expand the list of common bands:

Figure 9 - Bands sub properties

The Object Inspector shows if a given band type exists in the report or not, and you can add or delete a band simply by changing the relevant property. Bands created this way get names that describe their function: *DetailBand1*, *PageHeaderBand1* and so on. The *BandType* property of each band is also set automatically.

You can also add bands by selecting the *TQRBand* component on the component palette and dropping it on the report. Note that if you do it this way you must take care to set the *BandType* property to the desired band type, and you should also give the band a descriptive name. The *Bands* property of the container *TQuickRep* will update itself to reflect bands added to the report this way.

> While it is possible to add a band manually and set its *BandType* to *rbSubDetail* or *rbGroupHeader*, this is not recommended. These band types are intended for use only with *TQRSubDetail* and *TQRGroup* components. Using them elsewhere may cause unexpected and undesirable effects when the report is printed.

Here are the simple band types you can add to a report:

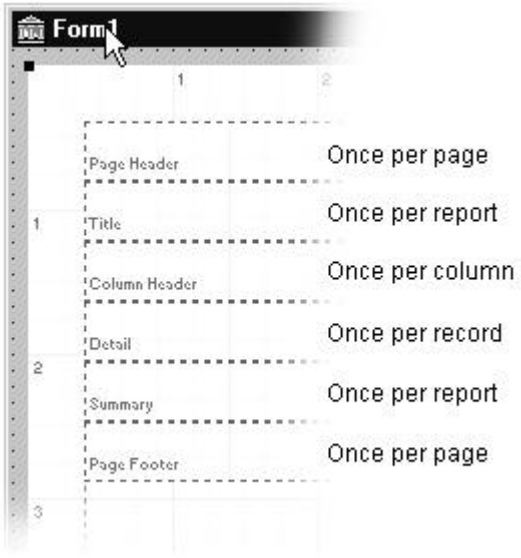
Table 2 - Simple band types

Band type	Purpose
Page Header	The first band usually printed on all pages. Whether it is printed on the first page printing is governed by the report's *Options.FirstPageHeader* property. The default is to print the first page header.
Title	A title band is the first band printed to a report (after the very first page header, if any). It's useful for printing the report title, data selection criteria, creation date and time and so on.
Column Header	The column header band is printed on top of each report column. In a regular single column report, this band is printed once per page, following the page header (and the title band for the first page). In a multi column report it's printed once for each column. It's most useful for printing field names.

Detail	One detail band is printed for each record (row) of data in your dataset. This is perhaps the most important band in your report and is usually what takes most of the space on the final output. You would typically put data-aware printable controls such as *TQRDBText* on this band.
Summary	After all detail bands has been printed you can print a summary band. This is often used for printing totals of numeric fields.
Page Footer	The last band printed on all pages. Last page printing is governed by the report's *Options.LastPageFooter* property. The default is to print the last page footer.

As you add new bands to a report, you will notice that they automatically position themselves in the actual printing order. You will see that the Page Header band is on top, followed by the Title band, column header band and so on, as shown in Figure 10 below.

Figure 10 - Simple band types



Each band has its band type printed in small letters in its lower left corner. This allows you to identify the bands while designing the report. This text is not printed on the final report.

Bands appear on the *TQuickRep* component in the order in which they are printed. It is helpful to understand why the bands line up the way they do. Generally bands will print in at the frequency shown in Figure 10, although things become more complicated when you start to add sub details and group bands.

Sizing the bands

Bands derive their horizontal size from the containing *TQuickRep* object. Their *Size.Width* properties should be considered read only; values written to them are ignored. For a single column report, the width of all bands is set to the page width minus the left and right margins. In multi-column reports, the width of certain band types (Column Header, Detail, Sub Detail, Group Header and Group Footer) is adjusted to reflect the width available for a single column.

However you can adjust the vertical size of the bands. Select a band and resize it with the mouse in the usual way or, if you want more accurate control, setting an exact value in the *Size.Height* property.

Turning bands on and off

You might sometimes want to disable printing of a certain band. This can be done, either at design time or at run time, by setting the *TQRBand.Enabled* property to *False*.

During report generation you can also temporarily disable printing of a band by creating an event handler for the band's *BeforePrint* event. This event handler takes a boolean parameter *PrintBand* that can be set to *False* to disable band printing – but just for that single instance. This feature can be used to perform simple filtering:

```
procedure TrepCusList.RepDetailBeforePrint
 (Sender: TQRCustomBand;
  var PrintBand: Boolean);
begin
  PrintBand := CustTableTotalSales > 3000000;
end;
```

Note: When *PrintBand* is set to *False* for a detail band, the values for that record are not included in any aggregate *TQRExr* function, for example the `SUM` function. This is a behaviour change between QuickReport 2 and QuickReport 3.

If you turn off a Page Footer band, it will have the effect of leaving a blank space at the bottom of each page – the Detail Bands will not expand to fill the space. To optimise performance, QuickReport doesn't check the length of the page footers all the time. So after you change the *Enabled* property of your Page Footer, call the report object's *ResetPageFooterSize* method to force QuickReport to update its page footer information.

Groups

Groups allow you to generate extra bands between groups of records. For example, if you were listing an address book, you might wish to group all the contacts whose name began with the same capital letter, and to print that letter in large type above each group – in fact this is what we do in the example.

To create a group:

1. Create a simple report as described in 'A first report' above.
2. Set the *IndexName* property of the *TTable* component to 'ByCompany'.
3. Drop a *TQRGroup* component onto an existing *TQuickRep* object, where it appears as a new band. This band will be the *group header*. Every time the group 'breaks', this band will be printed.
4. Set the *Expression* property to

   ```
   COPY(Table1.Company, 1, 1)
   ```

   This extracts the first character from the 'Company' field.
5. Drop a *TQRExpr* control onto the header band. Set its *Expression* property to the same value:

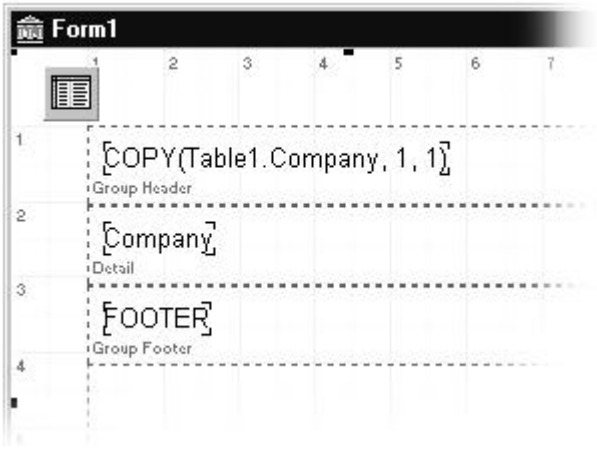
   ```
   COPY(Table1.Company, 1, 1)
   ```

In addition you can also add a *group footer* band. Although we don't really need one here, we'll make one for practice.

6. Select the *TQRBand* component on the palette and drop it on the report. Rename it to *FooterBand1*.
7. Click on the group header band once more. Set the *TQRGroup.FooterBand* property to *FooterBand1*.
8. Drop a *TQRLabel* onto the footer band. Set its *Caption* property to 'FOOTER'.

If all has gone to plan, you should be looking at something like Figure 11:

Figure 11 - Creating a group



Now preview the report, either by running the program, or simply right-clicking on the report object and choosing Preview:

Figure 12 - Preview of a grouped report

As expected, the resulting report shows a list of all the companies grouped in alphabetical order, with each group headed by a line showing the current letter.

Master/detail reports

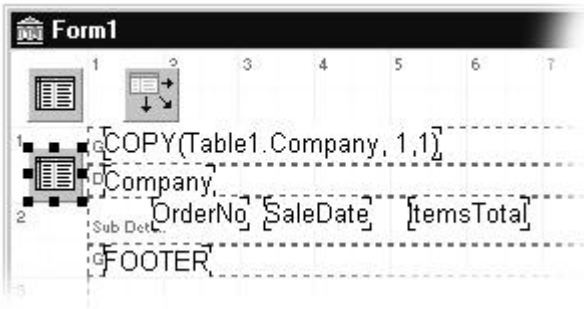
You will very often wish to create a master/detail report – that is one where you are extracting data from two datasets connected by a master/detail relationship. QuickReport allows you to include one or more such relationships in a report using *TQRSubDetail* components.

An obvious example of a master/detail report is to list out all the orders associated with each customer in a database. Here is a quick example:

1. Start with the group report as created in the previous section.
2. Drop a *TDataSource* component on the form, and make its *DataSet* property 'Table1'.
3. Drop a new *TTable* component on the form. Set its *DatabaseName* property to 'DBDemos', *TableName* to 'ORDERS.DB', *IndexName* to 'CustNo', *MasterSource* to 'DataSource1', *MasterFields* to 'CustNo' and *Active* to *True*. The two *TTable* components are now set up in a master/detail relationship.
4. Drop a *TQRSubDetail* component onto the existing *TQuickRep* object, where it appears as a new band. Notice that its *Master* property is automatically set to *QuickRep1*. The master/detail relationship between the two *TTable* objects is mirrored between the report object and its sub detail band.
5. Set the *TQRSubDetail* component *DataSet* property to 'Table2'. The *TQRSubDetail* component iterates all through its *DataSet* for each change in the *Dataset* of its *Master*.
6. Drop three *TQRDBText* components on the sub detail band. Set their *DataSet* properties to 'Table2', and set the *DataField* properties to 'OrderNo', 'SaleDate' and 'ItemsTotal' respectively.

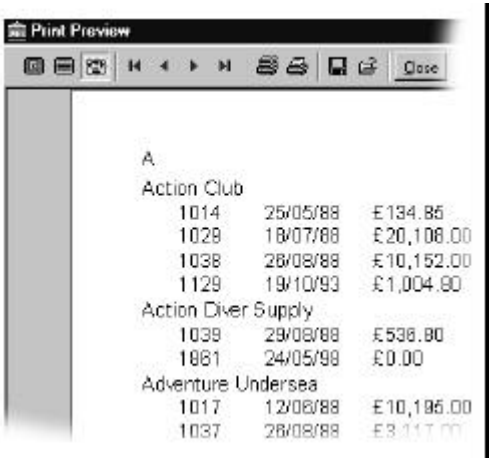
If you have done all that correctly, you should now be looking at something like Figure 13 below.

Figure 13 - Creating a report with sub detail



Now preview the report. The result will look like Figure 14, with each customer's orders listed below the customer name. Note that the format of the date and currency fields depends on Windows' international settings. Mine are set to British, your mileage will obviously vary.

Figure 14 - Preview of a master/detail report



More about printable components

By now, you will have a feel for QuickReport's printable components – use them like ordinary controls to define you layout. However, there are a few QuickReport-specific things to learn about them.

Text components

QuickReport's printable text components – *TQRLabel, TQRDBText, TQRExpr, TQRSysData, TQRMemo, TQRExprMemo*, *TQRRichText* and *TQRDBRichText* – share some common properties, which they inherit from a parent, class:

Table 3 - Text component properties

Property	Purpose
AlignToBand	By default components will print/align at the position set in the designer. But sometimes it is more practical to align components to the vertical edges of the band that it is placed on. When *AlignToBand* is *True*, a text components will align itself relative to its parent band instead of its own text rectangle.
AutoSize	Set this property to *True* and a component sizes itself horizontally to fit whatever text is put into it.
AutoStretch	If *AutoStretch* and *WordWrap* are both *True*, a component can expand vertically to accommodate its text. When a component expands in this way it also expands its parent band, provided that band's *CanExpand* property is set to *True*. A band can expand over multiple pages if necessary. Note that if a component expands it will not move other components on the same band down. If you have components whose desired position should depend on the length of a stretching text, you should place these in a child band. Note also that this property cannot be used for components on any band that prints at the bottom of the page. This is typically the page footer, but also applies to any band that has had its *AlignToBottom*

	property set to *True*.
Frame	All text components can display a visible frame around them. This property controls the appearance of the frame, which sides it is drawn on and so on.
Size	All printable components share the *Size* property. If *AutoSize* is *False* you can use this property to set the exact size of the component. This property also contains the position of the component relative to its parent band.
WordWrap	If *WordWrap* is set to *True* text can span multiple lines.

Formatting fields

TQRDBText components use any formatting options defined for the field to which they are connected. Sometimes, however, you need to customise the display of a particular value; this can be achieved using *TQRDBText*'s *Mask* property. This takes the same values as Delphi's own *FormatFloat* function (for numeric fields) and *FormatDateTime* function (for date and time fields) – in fact, QuickReport itself calls these functions to do the work. To give you an example, suppose you wished to print out a numeric value to two decimal places, with the thousands separated by a comma, and negative values shown (in parentheses) in the style favoured by accountants. Then you could use a *Mask* like this

```
#,##0.00;(#,##0.00)
```

which would cause the values

`1234` and `-1234.5`

to be printed as

`1,234.00` and `(1,234.50)`

respectively.

Check out the Delphi help for *FormatFloat* and *FormatDateTime* for details and more examples.

To specifically set a formatting of a field use the *Mask* property. The mask works differently for different field types.

Using expressions

QuickReport includes an advanced expression evaluator, used by the *TQRExpr*, *TQRExprMemo* and *TQRGroup* components. Expressions can be used to combine and manipulate database fields, and perform advance formatting. Their syntax is rather like that of Object Pascal: the expressions can be of boolean, float, integer or string type. Note that date and time fields are converted into strings, and BLOB and memo fields are not supported in expressions.

The evaluator supports the usual set of operators:

Table 4 - Operators supported by the expression evaluator

Operators	Function
`+`	Addition, string concatenation
`- * /`	Subtraction, multiplication, division
`()`	Parentheses
`And Or Not`	Logical operators
`= < >`	Comparison operators
`<= >= <>`	

and a set of standard functions:

Table 5 - Functions supported by the expression evaluator

Function	Description
`AVERAGE(EXPR)`	Aggregate function. Averages the `EXPR`
`COPY(STR,S,L)`	Returns a sub string of `STR` starting at character `S` length `L`
`COUNT`	Aggregate function. Returns the number of iterations of the Master band.
`DATE`	Return current date as a string
`DIV(X, Y)`	Integer division of `X` by `Y`
`FALSE`	Logical value False
`FORMATNUMERIC(F, N)`	Format numeric `N` using string mask `F`. The

	mask takes the values as Delphi's *FormatFloat* function.
FRAC(NUM)	Returns the fractional part of a NUM
IF(EXPR, R1, R2)	Returns R1 or R2 depending on the boolean EXPR
INT(NUM)	Returns the integer part of NUM
LOWER(STR)	Returns STR in lowercase
MAX(EXPR)	Aggregate function. Returns the highest value of EXPR
MIN(EXPR)	Aggregate function. Returns the lowest value of EXPR
PRETTY(STR)	Returns STR in 'pretty' case, ie first latter in uppercase, the remainder lowercase
SQRT(NUM)	Returns the square root of NUM
STR(NUM)	Converts NUM to a string
SUM(EXPR)	Aggregate function. Returns the sum of EXPR
TIME	Return current time as a string
TRUE	Logical value True
TYPEOF(EXPR)	Returns the data type of EXPR as a string, eg 'BOOLEAN'
UPPER(STR)	Returns STR in uppercase

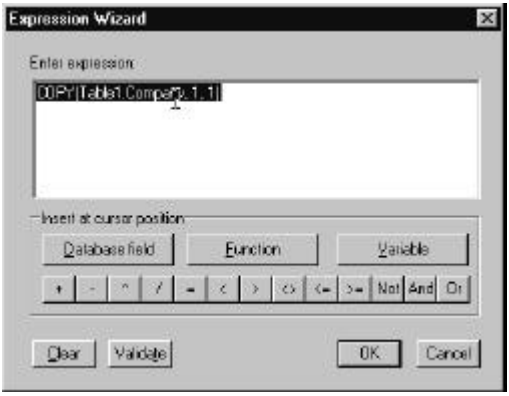
If your expression includes aggregate functions like SUM or COUNT you must link the *Master* property to the component, *TQuickRep* or *TQRSubDetail* that will be used to update the expression. For a simple report this is your *TQuickRep* component, but in a complicated report with many datasets you must take care to link to the correct *TQRSubDetail*. The expression is recalculated each time the record pointer of the linked master is advanced.

The *ResetAfterPrint* property is also useful when working with aggregation functions, and allows you to create, for example, group totals as well as running totals.

The expression builder

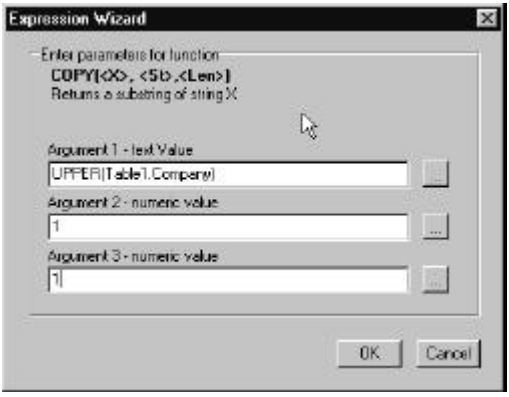
To make it easier to create expressions for your reports, QuickReport includes a special property editor, which appears when you click the '…' button. This is shown in Figure 15.

Figure 15 - Expression builder dialog



The expression builder lets you design your expression by selecting functions and field names from lists – so it makes it a lot easier to avoid typos in identifier names. It also brings up special dialogs to prompt for function arguments.

Figure 16 - Setting function arguments in the expression builder



Creating a default custom preview

We mentioned back in the 'Previews and composite reports' section that it was possible to change the default preview mechanism. It is time to look at how this is done.

The first step when creating a custom default preview is to derive a new class from *TQRPreviewInterface*, like this:

```
// use QRPrntr to get TQRPreviewInterface

TQRCustomPreviewInterface = class(TQRPreviewInterface)
  public
    function Show(AQRPrinter : TQRPrinter)
      : TWinControl; override;
    function ShowModal(AQRPrinter : TQRPrinter)
      : TWinControl; override;
  end;
```

Notice that this is an *interface*[1] class – it serves only to define a couple of functions, and has no data of its own. These two functions are implemented to construct and display your custom preview in non-modal and modal forms.

Lets suppose that the preview form is going to be called *TCustPreview*. Then the implementation of the *TQRCustomPreviewInterface* methods might look like this:

```
function TQRCustomPreviewInterface.Show(
  AQRPrinter: TQRPrinter): TWinControl;
var
  frm : TCustPreview;
begin
  frm := TCustPreview.Create(Application, AQRPrinter);
  frm.Show;
  Result := frm;
end;
```

[1] We have not used Delphi's *interface* keyword to define these classes because this is not currently common practice, and many Delphi programmers are unfamiliar with the syntax. However, the concepts are very similar.

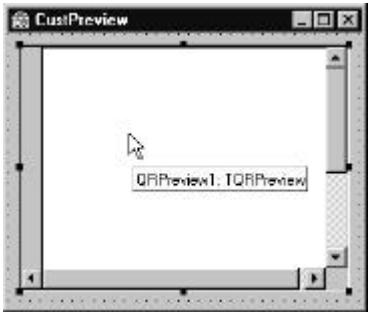
```
function TQRCustomPreviewInterface.ShowModal(
  AQRPrinter: TQRPrinter): TWinControl;
var
  frm : TCustPreview;
begin
  frm := TCustPreview.Create(Application, AQRPrinter);
  frm.ShowModal;
  Result := frm;
end;
```

To register our alternative previewer, we need to call the *RegisterPreviewClass* function, which is in the *QRPrntr* unit. The call looks like this:

```
RegisterPreviewClass(TQRCustomPreviewInterface);
```

Now we are done with the glue code, and can build the actual previewer form. Mine is minimal; just a single *TQRPreview* control stuck onto a form:

Figure 17 - Simple preview form



When you do real previews in your applications, you will probably want to add buttons to call *TQRPreview*'s *Zoom* method and other facilities.

To support the previewing mechanism, I had to write a little more code. Here is the declaration of *TCustPreview*. Notice I have added a new constructor, which expects to receive the *TQRPrinter* argument passed in by the *Show* and *ShowModal* methods of the interface class. Delphi generates a warning message that the new constructor hides the original. In this case it is deliberate, so I have wrapped the class in

```
{$WARNINGS ON} ... {$WARNINGS OFF}
```

compiler directives to make it shut up.

```
{$WARNINGS OFF}
  TCustPreview = class(TForm)
    QRPreview1: TQRPreview;
    procedure CustPreviewClose(Sender: TObject;
                               var Action: TCloseAction);
  private
    { Private declarations }
    fQRPrinter : TQRPrinter;
  public
    { Public declarations }
    constructor Create(AOwner : TComponent;
                       AQRPrinter : TQRPrinter); virtual;
  end;
{$WARNINGS ON}
```

Finally, here is the implementation of the class. Notice in particular the cleanup code held in the form's *OnClose* event. If you don't call *ClosePreview* here, you will get a nasty memory leak. (QuickReport 2 users should note that this is a new requirement. You must modify your existing preview forms when porting them to QuickReport 3 and later.)

```
constructor TCustPreview.Create(AOwner: TComponent;
  AQRPrinter: TQRPrinter);
begin
  inherited Create(AOwner);
  fQRPrinter := AQRPrinter;
  QRPreview1.QRPrinter := AQRPrinter;
end;

procedure TCustPreview.CustPreviewClose(Sender: TObject;
  var Action: TCloseAction);
begin
  fQRPrinter.ClosePreview(Self);
  Action := caFree;
end;
```

Further resources

There are many other resources available to you, to help you make the most of QuickReport in your Delphi applications.

Help files. The QuickReport help files are installed and integrated with Delphi's own help. As well as context-sensitive Reference for components and properties, there are also extensive User Guide and Knowledge Base sections. Since they don't appear in the main Help contents, it's easy to overlook these, so if you get stuck or need to know more, remember to fire up QuickReport help from its Start menu shortcut.

Demo applications. You'll find these in the

`Program Files\Borland\Delphi5\Demos`

directory. The project `Quickrpt\Qr3\qr3demo.dpr` contains lots of examples of different kinds of report. Start here if you want to exploit QuickReport's fancier features, like export filters. Also see the project `Db\Mastapp\mastapp.dpr` contains a good example of QuickReport being integrated into a larger application.

Templates and wizards. Again often overlooked – the standard Delphi repository includes templates for Master/detail, Labels and List reports, plus a report building wizard. Take File | New in the Delphi IDE and have a look at the Forms and Business tabs.

Website. Our website is http://www.qusoft.com Please come and check it out to find more examples, updates, tips and add-ons.