MASTAPP Help Contents

Click the underlined text that matches a button's label to learn more about the parts of MASTAPP, the sample application that demonstrates Borland C++Builder's database capabilities.

Help for each button describes the button's purpose in the application, lists the MASTAPP project files used to build that part of the application, and details significant database and C++Builder features that the button illustrates.

- New Orders enters new customer orders.
- Browse examines existing orders.
- Parts maintains the MAST inventory.
- Reports prints MASTAPP reports using Quick Reports.
- Close exits MASTAPP.
- Help describes how MASTAPP help is implemented.

Click Splash Screen to learn about the banner that appears when you start MASTAPP.

Select Customer

This form enables you to select a customer by name for placing new orders, or for browsing existing orders. Select the desired customer, and click OK. The Select Customer form is the same form used by Select Part, except that the data source and data fields are different.

Source code for this form can be found in:

SRCHDLG.CPP

SRCHDLG.H

SRCHDLG.DFM

Select Part

This form enables you to select a part by name when entering customer orders. Select the desired part, and click OK. The Select Part form is the same form used by Select Customer, except that the data source and data fields are different.

Source code for this form can be found in:

SRCHDLG.CPP

SRCHDLG.H

SRCHDLG.DFM

New Orders

New Orders enables you to enter new customer orders, modify existing customer orders, browse existing orders, or print the current order.

The form contains two *TDBNavigator* controls, one for editing and inserting records, and another for navigating among records. When you enter a new record, the navigation buttons are disabled, and only the Post and Insert buttons are enabled on the other control. When you browse existing records, the navigation buttons are enabled. When you edit a record, all navigation buttons on both navigator controls are enabled as appropriate.

A mode indicator in the upper right corner of the form indicates whether you are inserting, editing, or browsing.

Source code for the New Orders form can be found in:

EDORDERS.CPP

EDORDERS.H

EDORDERS.DFM

Insert Mode

The following behavior can be expected on the New Orders form in insert mode.

The OrderNo is automatically filled in with the next available order number and displayed in the upper right hand corner of the form. The Date *TDBEdit* control is automatically filled in with today's date. Select a customer from the CompanyCombo *TDBLookupComboBox.* When you select a customer from the list, the application fills in the rest of the customer information in the Bill To edit controls. You can cut and paste from the Bill To edit controls to the Ship To edit controls, or enter different values as appropriate.

The Sold By, Terms, Payment, and Ship Via fields are TDBComboBox controls tied to the ORDERS table. You can select values for these fields from the drop-down lists.

You can enter parts ordered by the customer in the TDBGrid control. You can double-click in the grid, or click the elipsis button in the grid to select parts from the MAST inventory. Both methods invoke the Select Part form.

As parts are entered in the grid, calculations are automatically performed to update the values in the Subtotal, Tax Rate, Tax Total, Freight, and Due database edit controls. These controls, and the Paid control, are connected to the ORDERS table. When the new record is posted, these values are inserted into the ORDERS table.

Edit Mode

The following behavior can be expected on the New Orders form in edit mode.

The first record in the ORDERS table is displayed in the form. Use the navigation controls to scroll through all records. You can modify fields within records as long as those fields are not read-only.

You can enter parts ordered by the customer in the TDBGrid control. You can double-click in the grid, or click the elipsis button in the grid to select parts from the MAST inventory. Both methods invoke the Select Part form.

As parts are entered in the grid, calculations are automatically performed to update the values in the Subtotal, Tax Rate, Tax Total, Freight, and Due database edit controls. These controls, and the Paid control, are connected to the ORDERS table. When the new record is posted, these values are inserted into the ORDERS table.

Browse Mode

The following behavior can be expected on the New Orders form in edit mode.

The first record in the ORDERS table is displayed in the form. Use the navigation controls to scroll through all records. If you attempt to modify any field in a record you enter Edit Mode.

Browse

To examine orders placed by customers, click the Browse button. It invokes the Orders By Customer form.

Orders By Customer displays two *TDBGrid* controls. The top control displays all MAST customers, ordered by customer number. When the form first appears, the first customer number in the CUSTOMER table is selected. The bottom grid displays all orders associated with the current customer number.

Use the *TDBNavigator* control to scroll on either grid. To establish which grid the navigator affects, click on the desired grid. If you scroll to a different record on the customer grid, the orders grid changes to reflect those orders associated with the current customer number. Scrolling the bottom grid lets you see all orders associated with the current customer number.

The Define Query button invokes the Specify Range dialog box, where you can enter a narrower range of dates against which to check for customer invoices, in order to reduce the number of records returned in customer grid. After you define a query, use the Activate Query button to execute the query. When a query is active, the Activate Query button remains in a depressed state. To deactivate the query and restore the default customer list display, click the Activate Query button again.

The Orders By Customer form demonstrates how to switch data sources for a control programmatically at run time. The Define Query and Activate Query code demonstrates how to construct parameterized SQL queries from user input at run time, and how to switch between data sets programmatically.

Source code for Orders By Customer can be found in:

BRCUSTORD.CPP

BRCUSTORD.H

BRCUSTORD.DFM

Parts

The Browse Parts form enables you to browse and edit the MAST inventory

The form contains a *TDBGrid* for displaying all parts and backordered parts, a *TDBNavigator* control for scrolling through records, a Backorder button for querying to see all parts on back order, an Edit button that displays the Edit Parts form for adding or modifying parts, and a Close button for exiting the form.

Browse Parts demonstrates how a single data source can be used by either a TTable or TQuery component. By default, when the form appears, it displays data for all parts from the TTable component. Clicking the Backorders button invokes code that connects TQuery to the data source and executes a query that displays only those parts on back order.

Source code for Browse Parts can be found in:

BRPARTS.CPP

BRPARTS.H

BRPARTS.DFM

Edit Parts

The Edit Parts form enables you to add or modify parts in the MAST inventory, and print information about parts.

The form contains a *TDBNavigator* control for scrolling through records, adding and deleting records, and posting or canceling changes. It contains six *TDBEedit* controls for display, entry, and modification of fields in the PARTS table. It also contains a *TDBLookupComboBox* control for display, entry, and modification of vendor names in the VENDORS table based on vendor numbers in the PARTS table.

Edit Parts demonstrates how to look up, display, and edit field values in one table based on key values in another table by using *TDBLookupComboBox*.

Source code for Edit Parts can be found in:

EDPARTS.CPP

EDPARTS.H

EDPARTS.DFM

Reports

The Reports form lets you specify three different types of reports to print:

- *Customer List* lists MAST customers.
- *Orders History* details the orders placed with MAST.
- *Invoice* provides a full report on a specified invoice.

These reports are built using the Quick Report components on the QReport tab on the component pallet. The main dialog, MainPrintForm.cpp, contains some simple choices for selecting the type of report.

Each report can be previewed on the preview form (in PrevForm.cpp) which has a QRPreview component.

Each report is a form with a QuickReport component, a QRDetailLink component, and various other components from the QReport components tab. Some of which, mainly QRDBText, have DataSource and DataField properties.

Source code for the reports and the selection form can be found in:

PICKREP.CPP PICKREP.H PICKREP.DFM

CUSTRPT.CPP CUSTRPT.H CUSTRPT.DFM

ORDERRPT.CPP ORDERRPT.H ORDERRPT.DFM

INVOICE.CPP INVOICE.H INVOICE.DFM

Close

To exit MASTAPP, click Close, or choose File|Exit from the menu.

Help on Help

MASTAPP help demonstrates how to access a custom help file from a C++Builder application. To link a help file to an application and access it, follow these steps:

1. Set the application's *HelpFile* property to the name of the .HLP file to use. For example, in MASTAPP.CPP, the following code specifies the name of the help file to use for MASTAPP:

   ```
   Application->HelpFile = "MASTAPP.HLP";
   ```

2. Set the main component's *HelpContext* property to *1*. For example, in MASTAPP, MAIN.PAS contains the code for *MainForm*. Select the *MainForm* component, then set its *HelpContext* to *1* in the Object Inspector.
3. To access help from a button, create a help procedure that calls the *HelpCommand* method with a valid WINHELP command. To display the contents for a help file, use the HELP_CONTENTS command, and set the data integer to 0. In MASTAPP, the following help procedure is defined with a context ID that calls the MASTAPP Contents page:

   ```
   void __fastcall TMainForm::HelpBtnClick(TObject *Sender)
   {
     Application->HelpCommand(HELP_CONTENTS, 0);
   }
   ```

Note At design time, set the *HelpContext* property for individual components to specify a context-sensitive help screen to display if the user presses F1 when those components have focus. In MASTAPP, context-sensitive help is implemented for all components on the Parts form, and the Edit Parts form.

4. Create a button that calls a help procedure when clicked. In MASTAPP, MAIN.H contains the following code for creating a help button:

   ```
   TSpeedButton *HelpBtn;
   ```

   In the Object Inspector, set the button's *OnClick* event to the name of the help procedure to call. In MASTAPP, the name of the procedure that is called in *HelpBtnClick*.
5. Create a procedure that calls HelpCommand with the WINHELP command, HELP_QUIT to quit WINHELP when the application ends. IN MASTAPP, MAIN.PAS contains the following procedure for exiting WINHELP:

   ```
   void __fastcall TMainForm::FormDestroy(TObject *Sender)
   {
     Application->HelpCommand(HELP_QUIT,0);
   }
   ```

Splash Screen

The Splash screen appears when MASTAPP starts. MASTAPP.CPP creates a *TSplashForm,* and the instance of the application, calls *SplashForm->Show()* to display the form while all other MASTAPP forms are created, and calls *SplashForm->Update()* to display the application's "Loading ..." message on the form.

When all MASTAPP forms are created, MASTAPP.CPP calls *SplashForm->Hide()* to hide the Splash screen display.

Source code for the splash screen can be found in:

SPLASH.CPP

SPLASH.H

SPLASH.DFM