Using components and packages

With HBasic you can add precompiled components that add new features like Qt widgets or access to C++ written libraries. For an introduction to components and object oriented programming in HBasic see *Shared libraries* and *Using OO in HBasic* . In this document we will show how to call methods in a component, access the properties of a component or connect to the events of a component.

Overview

A component defined within a HBasic package is a more complex type of C++ code which will be derived from the Qt class QWidget or QObject. Both types of components may export methods, events and properties. The component name itself may be used like a user defined type within HBasic source code. You can create an instance of a component within the HBasic formdesigner or in the sourcecode with a Dim statement like

Dim varname **As** component_name

In the second case the component will be created at runtime when the runtime execution reaches this source code line.

Links to other documents about components

- Using object oriented programming in HBasic
- List of HBasic packages
- Examples with predefined packages

Contents

Overview

The first part of this document will show how you can use properties, methods and events derived from QT widgets that may be the base class of HBasic components. A simple method to create a component is to create for example a button or a line edit widget in the form designer. Each icon in the select window on the left side of the HBasic mainwindow represents a component type.

HBasic can read the metainformation of the QT widget and use the properties, methods and events of it. When looking at the metainformation provided with the packages you can see that you can also set up your own methods, properties and events in the component code. The following examples show how you can access this parts from your HBsic program.

The example_package for HBasic component examples

For the component example programs I added a example_package which has a visible component xbutton and an unvisible component named nogui. This components both export a method "add" which adds two values and a property "prop1". Be sure to import this package with the package manager into your project if you don't load the predefined program examples. Other examples in this document use components from the hbasic_stdgui package.

Creating components in the formdesigner

If the component description defines an icon this will be displayed in the HBasic select window after loading the package which contains the component.

To create for example an instance of the Button component which will be defined in the hbasic_stdgui package click on the Button icon in the select window and click on the formdesigner window at the position where the button should be created. This will create a button component and you can now edit the properties exported by the component in the property editor.

After you have created an instance of a component either with a Dim statement or in the formdesigner you can access the methods, catch the events and read or write the properties of the component. If your component inherits the Qt class QPushButton (which inherits QWidget in Qt) you may call methods defined within QPushButton or you may extend the class with your own methods. You can call this exported methods from HBasic no matter if you call a Qt method or a slot that you have defined yourself. In the following examples we assume that you have created a component named *comp1* in the formdesigner or in the code with a statement like

Dim comp1 **As** component_type

Using predefined Qt properties

This example shows how you can access a property within a QT component. The text property of the lineedit component may be used to read or write the text that will be displayed in the QT lineedit widget that is the basetype of the HBasic component.

When you look at the source code and meta description of the component you can see that there is no code and no information mentioned in the component description. HBasic can read and access this directly from the QT widget. Have a look at the package manager to get a list of QT properties that may be used.

```
' Access property in GUI component

Sub btn_start_clicked()
   lineedit1.text = "Hello HBasic"
End Sub
```

Example ex_guicomp_qt_property: Access text property of QT component.

Calling predefined QT methods

Like getting some metainformation about the properties of a QT widget it is also possible to query a QT widget for methods (slots) that may be called. You can call this methods from HBasic without any user defined metainformation in the HBasic package about this method.

The following example calls the methods hide() and show() for the lineedit widget which will be displayed in the form window of the example. Since I didn't find a method to pass HBasic parameter values to a predefined QT method this method calls currently may only be used for methods without parameters. Any ideas how methods with parameters may be called in this environment are welcome.

```
Sub btnhide_clicked()
   lineedit1.hide()
End Sub

Sub btnshow_clicked()
   lineedit1.show()
End Sub
```

Example ex_guicomp_qt_method: Call predefined QT method.

Connecting to predefined QT events

Later in this document you will see how you can define and use your own events in a component definition. This example shows how you can connect a Hbasic subroutine to predefined QT widget event. This works because HBasic sets up a QT event filter for a component if the component is derived from a QT widget. You may than connect to this event with a subroutine with the name compname_eventname. Currently this works for the following event types (but may be extended later to other events):

- ButtonPress
- ButtonRelease
- ButtonDblClick
- Enter
- Leave
- Show

```
Sub button1_buttonpress()
    Print "Button has been pressed"
End Sub
```

Example ex_qt_comp_event.bas: Example how to use predefined QT events

Calling a method of a component

The following examples use parts of the component that have been defined by the developer of the component source code (See document how to create a new package or the source code of the packages delivered with HBasic). This example calls a user defined method within a component. This method comes from the nogui component of the example_package which only adds two numbers. Since the nogui component is not derived from a QT widget we have to create an instance of it with a Dim statement. Remember that you don't have to use a *New* statement to create an instance of a QT component.

```
 ' Read and write property of dim component

 Sub btn_start_clicked()
    Dim c As nogui

    ' Call method in component
    Print c.add( 444,555 )
 End Sub
```

Example ex_dimcomp_method.bas: Call method of component that's created with Dim

Overloaded component method

You may define more than one component method with the same name within one component definition. This is only useful if the method implementations use different parameter types. HBasic takes the first method with matching parameter types for a method call. This might be used if overloading a method name is important for yout component.

The xbutton component in the example_package has two methods with the name add. One for integer values and one for double values. This example calls the same method name with different parameter types.

```
 Sub xbutton1_clicked()
    Print xbutton1.add( 111, 222 )

    ' overloaded method for type double
    Print xbutton1.add( 11.111, 22.222 )
 End Sub
```

Example ex_guicomp_method.bas: Call overloaded component method.

Read and write properties

Components may export properties which can be used like variables. This example reads and writes the property prop1 of the nogui component.

```
 ' Read and write property of dim component

 Sub btn_start_clicked()
    Dim c As nogui

    c.prop1 = 1122
    Print c.prop1
 End Sub
```

Example ex_dimcomp_prop.bas: Read and write the value of a component property

GUI component property

This example shows the same thing also with a property of the xbutton GUI component.

```
 ' Use property in gui component

 Public c As nogui

 Sub xbutton1_clicked()
    xbutton1.prop1 = 888
    Print xbutton1.prop1

    ' property in dim component
    c.prop1 = 444
    Print c.prop1

    nogui1.prop1 = 1234
    Print nogui1.prop1
 End Sub
```

Catching events from a component

A predefined component may also throw events during runtime to inform the user that something special happened.

We already used the event *clicked* of the button widget within several HBasic examples. If a subroutine should be called if a component a throws the event evt you have to define this subroutine with the name a_evt. If you created a component *button1* which throws an event *clicked* you have to define a subroutine with name button1_clicked to catch the event.

```
Sub button1_clicked()
   Print "Button1 has been clicked"
End Sub
```

We have already seen some examples of how to catch an event from a component. Whenever we connect to the clicked event of a button widget to start execution of a test program we use component events since the button widget is a component of the hbasic_stdgui package.

To connect to an event you define a subroutine with the name *compname_eventname*. The HBasic compiler will connect this subroutine as an event handler to the event with name *eventname*. This will for example be used to call a subroutine if a button widget has been clicked (subname button1_clicked) or if you connect a subroutine to an action that will be started on menubar or toolbar events (subname action_xxx).