Other code examples

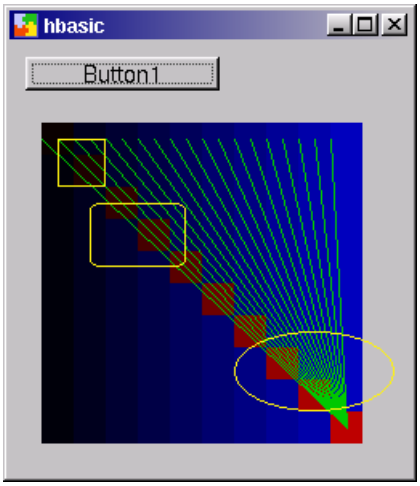
Contents

Overview

This document shows some examples for using methods of different packages, how to use the debugger and how runtime exceptions may happen or may be caught within a try..catch statement.

Painter package

The painter package exposes a component that may be used to draw pixel graphics like the one created with this example.




```
Public i As Integer

Sub button1_clicked()

   i = 10
   While i < 200
      painter1.setColor( 0,0,i )
      painter1.fillrect( i, 10, 20, 200)

      i = i + 20
   Wend

   i = 10
   While i < 200
      painter1.setColor( i, 0, 0)
      painter1.fillrect( i,i,20,20)

      i = i + 20
   Wend

   painter1.setColor( 0,200,0)
   i = 10
   While i < 200
      painter1.line( i, 20, 200, 200)
      i = i + 10
   Wend

   painter1.setColor( 255, 243,15)
   painter1.drawRect( 20,20,30,30)
   painter1.roundRect( 40,60,60,40)
   painter1.drawEllipse( 130,140,100,50)
```

```
End Sub
```

Example ex_comp_painter.bas: Draw lines and Rectangels in a painter component

HTML widget

This example shows how the HTML widget of the standard controls may be used to display a html file (The start-page of the HBasic docu) in a similar way as in a HTML browser.

```
Sub button1_clicked()
    htmlwgt1.setsource( "/usr/local/hbasic/doc/hbasic_index.html")
End Sub
```

Example ex_html_widget.bas: Example how to use the HTML widget.

Scrollbar / slider widget

This example shows how the scrollbar and slider widgets (horizontal and vertical) may be used. This widgets may be initialized with the methods min(int_value) and max(int_value) and the current value may be set with the setValue(int_value) method. Furthermore these widgets provide an event "valueChanged" which will be triggered whenever the user changes the value of a scrollbar or slider.

```
Sub button1_clicked()
   hscroll1.min = 0
   hscroll1.max = 20
   hscroll1.value = 10

   vscroll1.min = 0
   vscroll1.max = 20
   vscroll1.value = 5
End Sub

Sub hscroll1_changed()
   vscroll1 . value = hscroll1.value
End Sub

Sub vscroll1_changed()
   hscroll1 . value = vscroll1.value
End Sub
```

Example ex_scrollbar.bas: Example how to use the scrollbar widget.

These examples change the value of the horizontal widget if the value of the vertical widget will be changed by the user and set the vertical widget if the user changes the horizontal widget. This should show how to connect to *changed* events and set slider values.

```
Sub button1_clicked()
   hslider1.min = 0
   hslider1.max = 20
   hslider1.value = 10

   vslider1.min = 0
   vslider1.max = 20
   vslider1.value = 5
End Sub

Sub hslider1_changed()
   vslider1.value = hslider1.value
   lineedit1.text = str( hslider1.value )
End Sub

Sub vslider1_changed()
   hslider1.value = vslider1.value
   lineedit1.text = str( vslider1.value )
End Sub
```

Example ex_slider.bas: Example how to use the slider widget.

Progressbar widget

This example shows how the progressbar widget may be used. You may set the number of steps for the progressbar widget with the steps(int_value) method and the current position with the progress(int_value) method.

```
Sub button1_clicked()
   progressbar1.steps = 20
   progressbar1.progress = progressbar1.progress + 1
```

```
   If progressbar1.progress > 20 Then
      progressbar1.progress = 1
   End If
End Sub
```

Example ex_progressbar.bas: Example how to use the progressbar widget.

Datetime package

The Datetime package includes components with name date and time which may be used to read the current system date or system time. Later this may be extended to replace all functions needed for values of type date and time.

```
Dim d As date
Dim t As time

Sub button1_clicked()
   d = date.currentdate()

   Print "Year = " + str( d.year())
   Print "Month = " + str( d.month())
   Print "Day = " + str( d.day())

   t = time.currenttime()

   Print "Hour = " + str( t.hour())
   Print "Minute = " + str( t.minute())
   Print "Second = " + str( t . second())
End Sub
```

Example ex_comp_datetime.bas: Access methods in datetime package

Standard dialog package

The dialog package includes components to show QT standard dialogs. Currently it can show dialogs of the following type:

filedialog	Component to create QFileDialog widget
fontdialog	Component to create QFontDialog widget
colordialog	Component to create QColorDialog widget
printdialog	Component to create QPrintDialog widget

```
Dim f As filedialog

Sub button1_clicked()
   f.show()
End Sub
```

Example ex_comp_dialog.bas: Show filedialog component

File package

The file package will include all components that may be used to read or write UNIX files. This includes the following components:

FILE	Read and write access to files (see example)
FILEDLG	Call QFileDialog to ask user for filename
FILEINFO	Get Info about files (QFileInfo)
DIRINFO	Get Info about directory structures and their contents (QDirInfo)

The following example opens a file in write-mode and writes the numbers from 1 to 20 to it.

```
Public f As file
Public i As Integer

Sub button1_clicked()
   f.open( "test.dat", "w" )
   i = 1
   While i < 20 Do
      f.write( "Value = "+str(i))
      i = i + 1
   Wend
   f.close()
End Sub
```

Example ex_comp_file.bas: Open file and write data to it

Menu example

The menu example shows how a subroutine may be connected to a menu or toolbar action. Have a look at the description how to set up a menubar or toolbar with the HBasic menueditor to create a new menubar or toolbar.

```
Sub button1_clicked()
   Print "Button clicked"
End Sub

Sub action_act_load()
   Print "Load selected"
End Sub

Sub action_act_save()
   Print "Save selected"
End Sub
```

Example ex_menu.bas: Start subroutine if user clicks menu entry or toolbar button

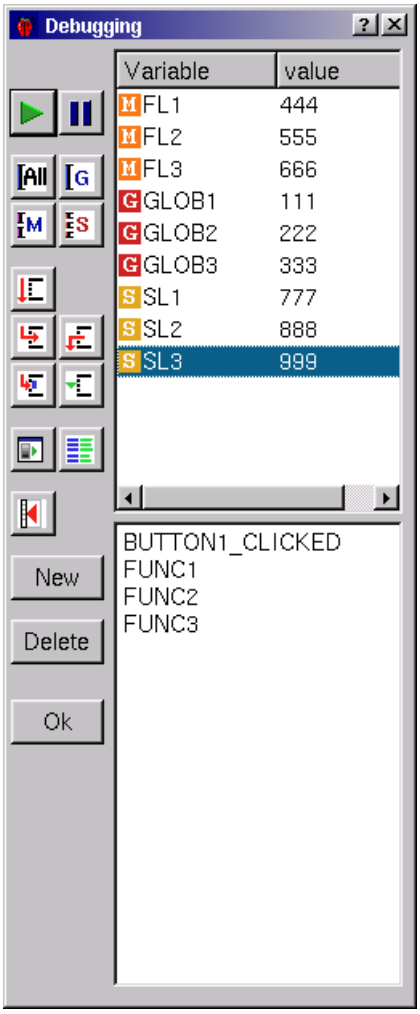
Debugging

This example defines some variables (global formlocal and sublocal) that can be shows in the debugging window.

```
Project Project1

Source Form1

Public glob1 As Integer
Public glob2 As Integer
Public glob3 As Integer

Dim fl1 As Integer
Dim fl2 As Integer
Dim fl3 As Integer

Sub func3()
   Dim sl1 As Integer
   Dim sl2 As Integer
   Dim sl3 As Integer

   glob1 = 111
   glob2 = 222
   glob3 = 333

   fl1 = 444
   fl2 = 555
   fl3 = 666

   sl1 = 777
   sl2 = 888
   sl3 = 999

   Print "Program started"
End Sub

Sub func2()
   func3()
End Sub

Sub func1()
   func2()
End Sub

Sub button1_clicked()
   func1()
End Sub
```

Example ex_debug.bas: Example that may be used to test debugging features.

Debugging window showing variable values after program has been aborted with a breakpoint. You can see the values of formlocal (M= modul) variables, G= global variables and S= SUBLOCAL variables and the callstack in the bottom window.



Error handling

Example that shows a parser error message because of illegal syntax in the code.

```
Sub button1_clicked()
   For While
End Sub
```

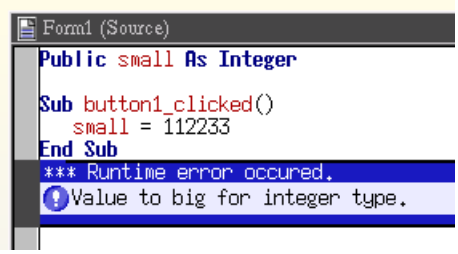
When you start the compiler for this example you can see how HBasic shows error messages at compile time.

Runtime exceptions

The next example forces a runtime error because of an overflow in an integer variable.

```
Project Project1

Source Form1

Public small As Integer

Sub button1_clicked()
   small = 112233
End Sub
```

Example ex_exception.bas: Program that throws runtime exception

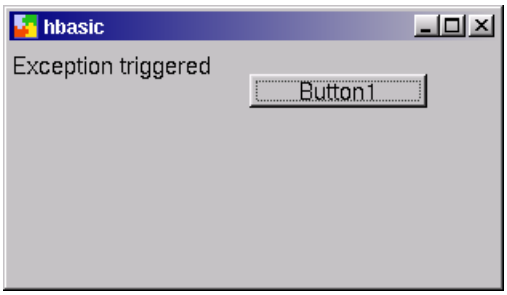


Caught runtime exception

If you don't want your program to stop with an emergency exit any time a runtime error occures you can use a "try .. catch .. end try" structure to catch the runtime error.

```
Public small As Short

Sub button1_clicked()
   Try
      small = 112233
   Catch( IntegerOverflow )
      Print "Exception triggered"
   End Try
End Sub
```

Example ex_caught_exception.bas: Catch runtime exception



There is no need for the "Try ... Catch" structure to be in the same subroutine where the runtime error occures. If there is no runtime handler in the current subroutine HBasic continues to search for a "Try...Catch" structure in the calling subroutine.

Example to show fallback to calling subroutine for runtime error handling.

```
Dim i As Short

Sub button1_clicked()
   i = 1111
   Try
      test2()
   Catch( IntegerOverflov )
      Print "Overflow error ..."
   End Try
End Sub

' Second subroutine where runtime error occurs.
Sub test2()
   i = i * 2222
End Sub
```