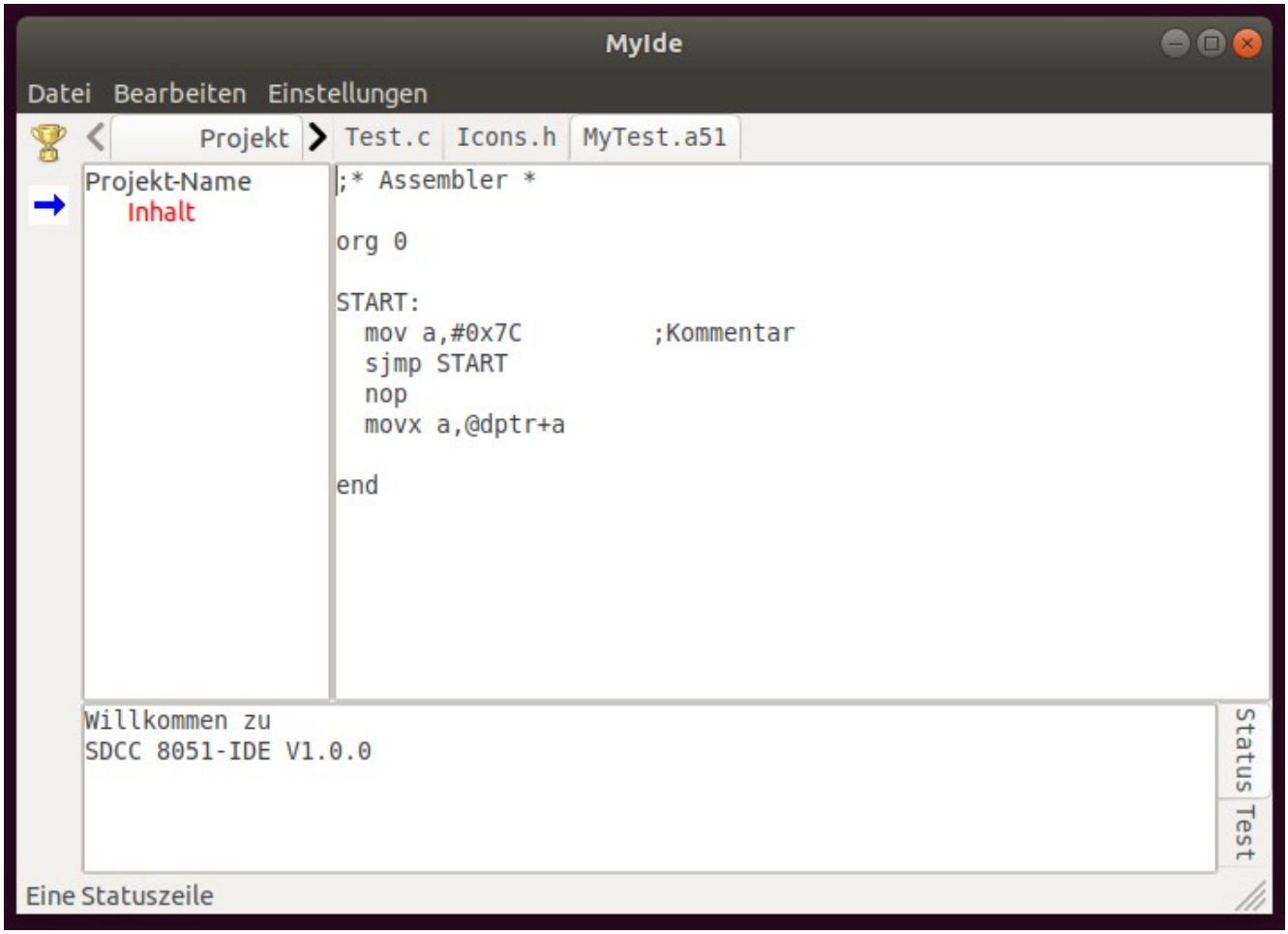
wxWidget mit QT-Creator

Entwicklernotizen

Teil 1 – Eine Basisapplikation



Stand: 31. März 2021
Tom Amann

Inhaltsverzeichnis

1 Einleitung

Nach einigen Versuchen, welche Programmierumgebung zum Erstellen von GUI-Anwendungen (Graphic User Interface) mit „wxWidgets“ gut geeignet ist, bin ich beim Qt-Creator gelandet. Der Qt-Creator ist in der Lage auch völlig von Qt losgelöste Programme für die Konsole zu verwalten. Die verwendete Programmiersprache ist C++, wobei der Qt-Creator alle benötigten Funktionen zum schreiben, testen und debugen der Programme ermöglicht.

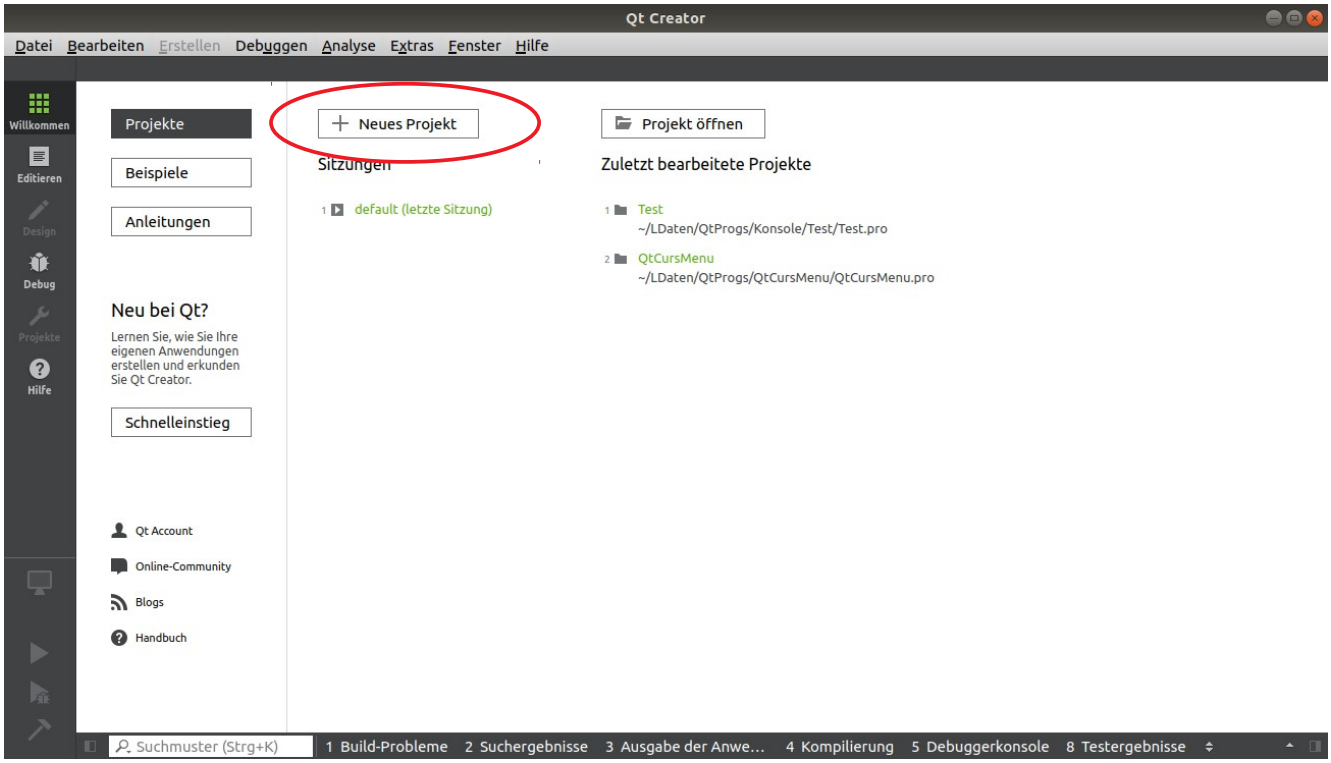
Die Entwicklungsumgebung ist Ubuntu Linux 18.04.5 LTS mit Qt-Creator 4.5.2.

Dies sind meine persönlichen Notizen, um auch nach einiger Zeit wieder nachzulesen, wie die Dinge zusammen hängen und funktionieren.

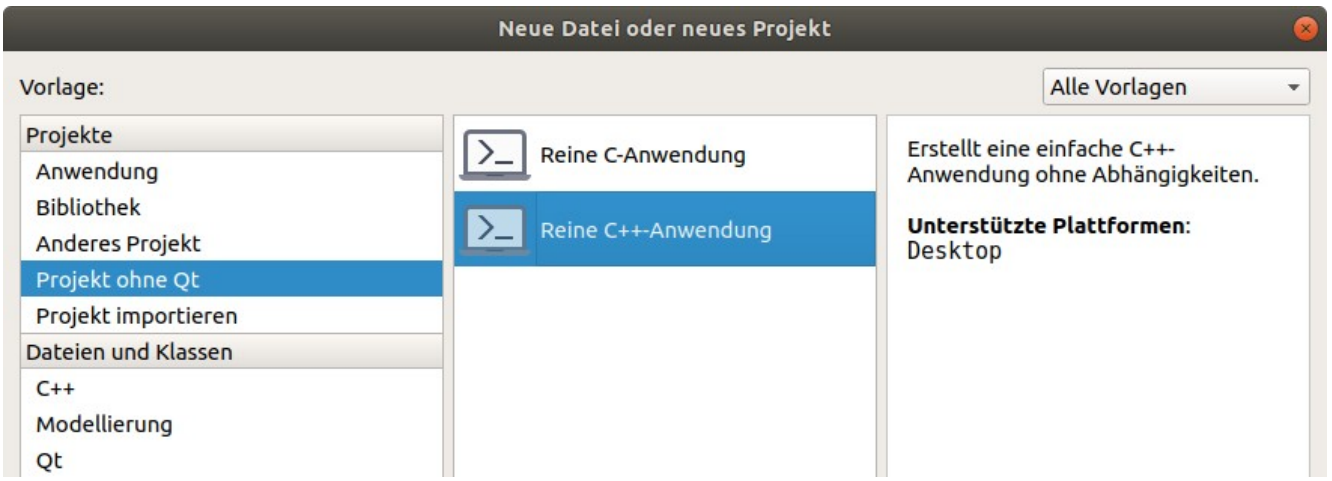
Zu den einzelnen Komponenten sind hier keine Details aufgeführt, diese sind der ausführlichen Dokumentation zu wxWidgets im Internet zu entnehmen und müssen hier nicht wiederholt werden.

2 Ein Projekt erzeugen

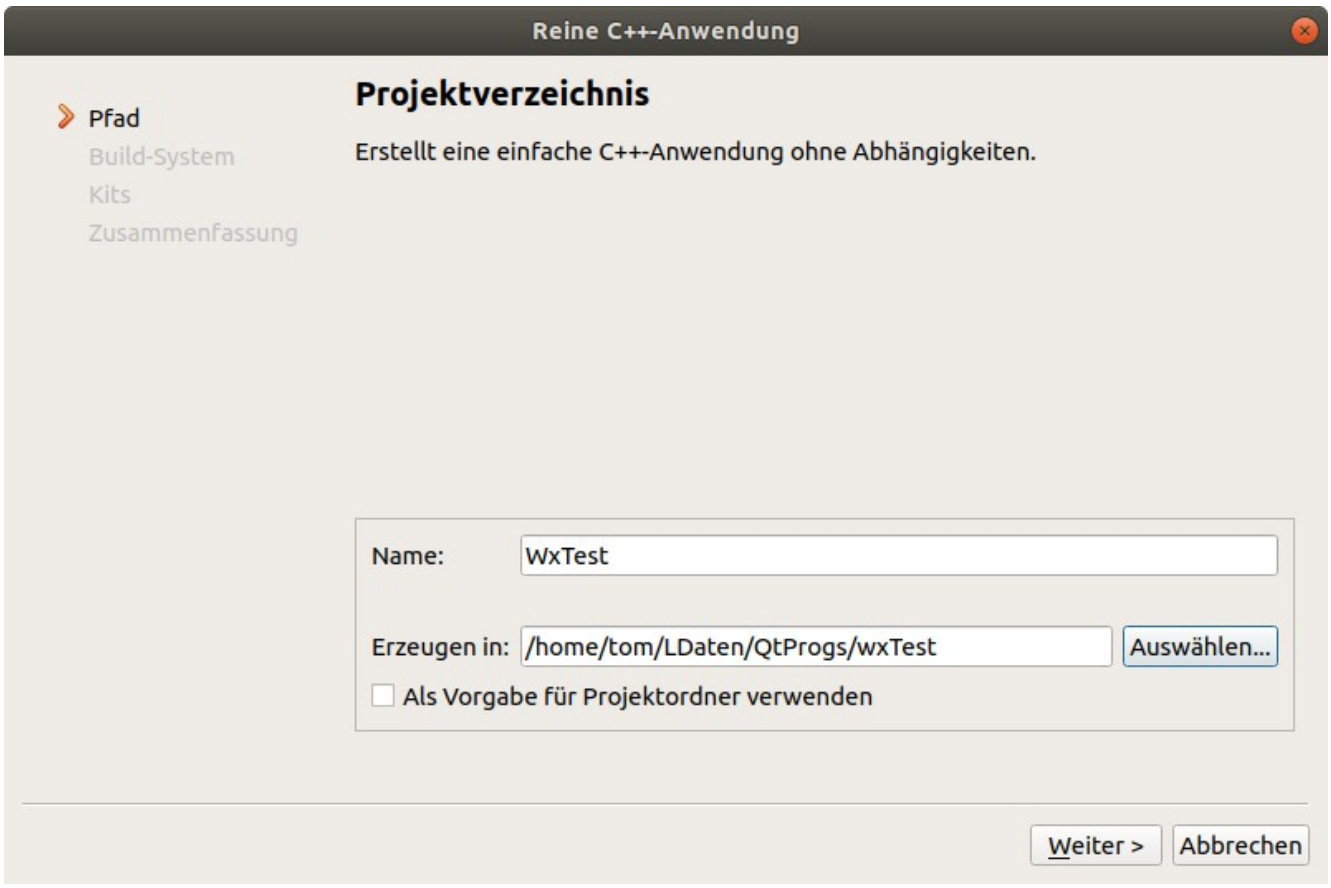
Zunächst ein neues Projekt anlegen. Dazu im Menü „Datei“ und anschließend „Neu“ wählen. Oder beim Start des Qt-Creator „Neues Projekt“ wählen



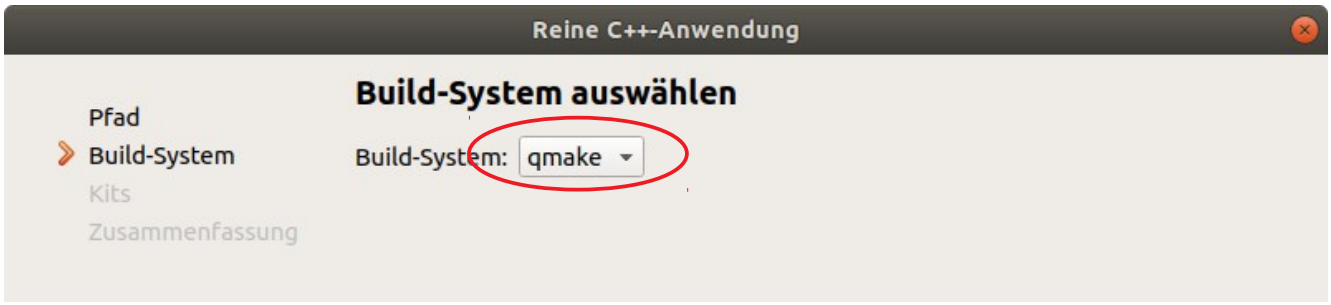
Hier ein „Projekt ohne Qt“ und die entsprechende Programmiersprache wählen. Meine Wahl ist „Reine C++-Anwendung“.



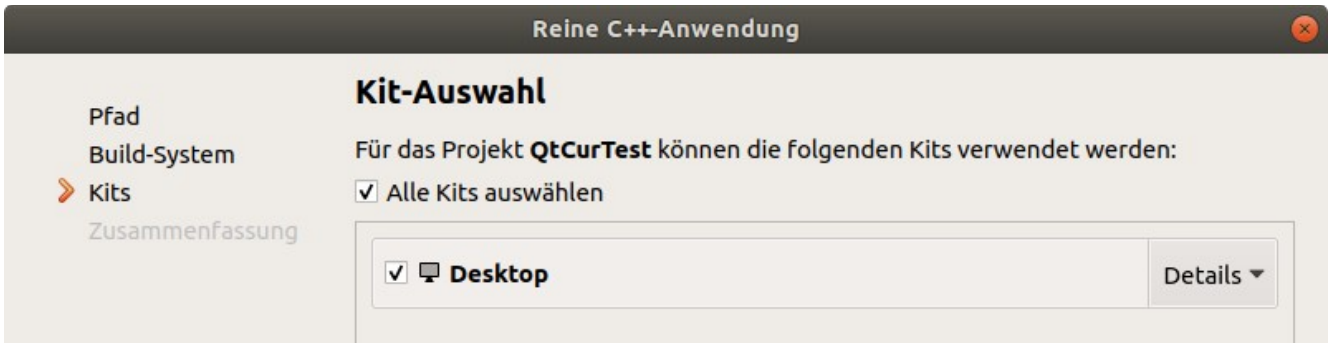
Nun den Projektnamen und ein Projektverzeichnis wählen.



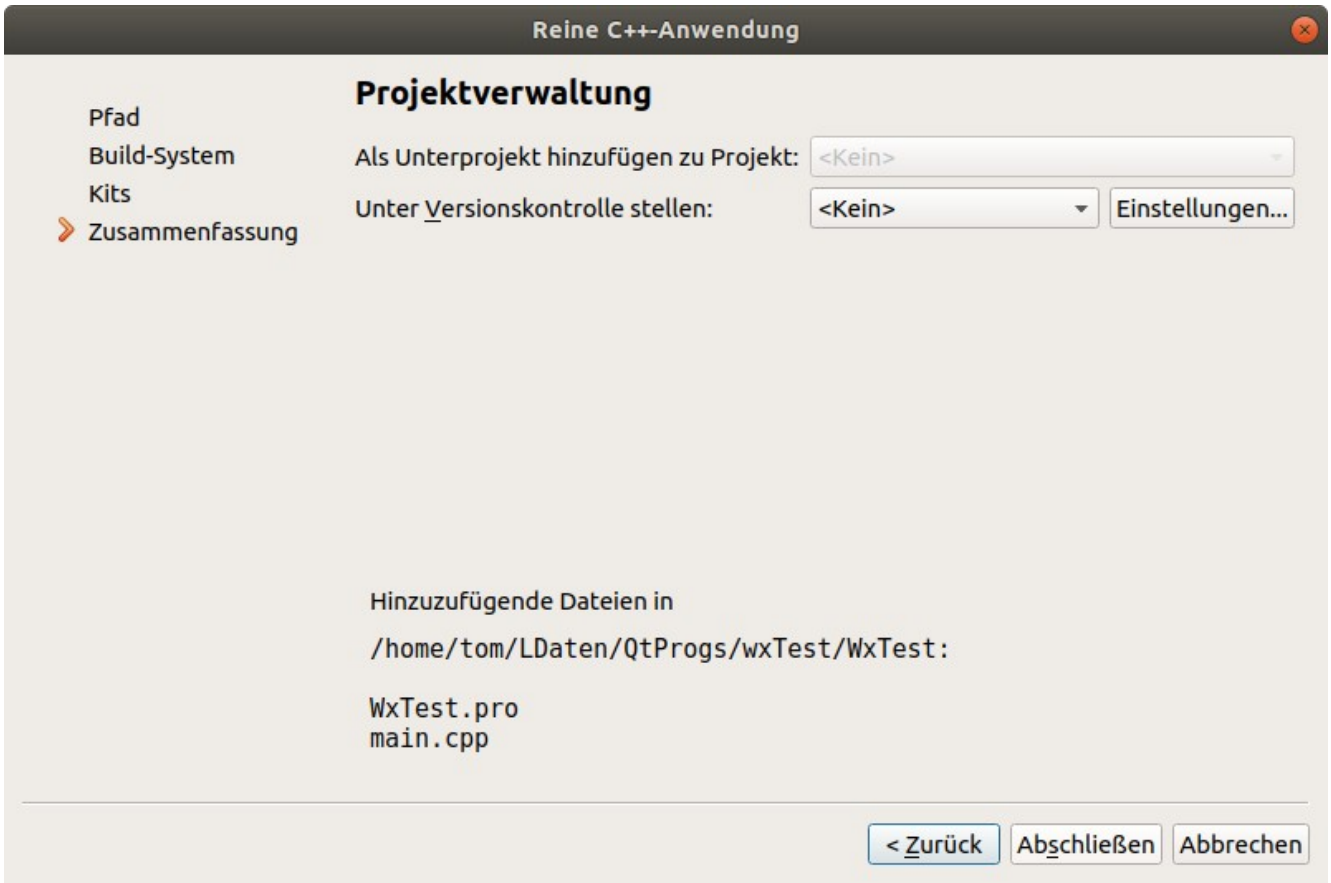
Jetzt noch das Build-System wählen, für wxWidgets ist „qmake“ gut geeignet.



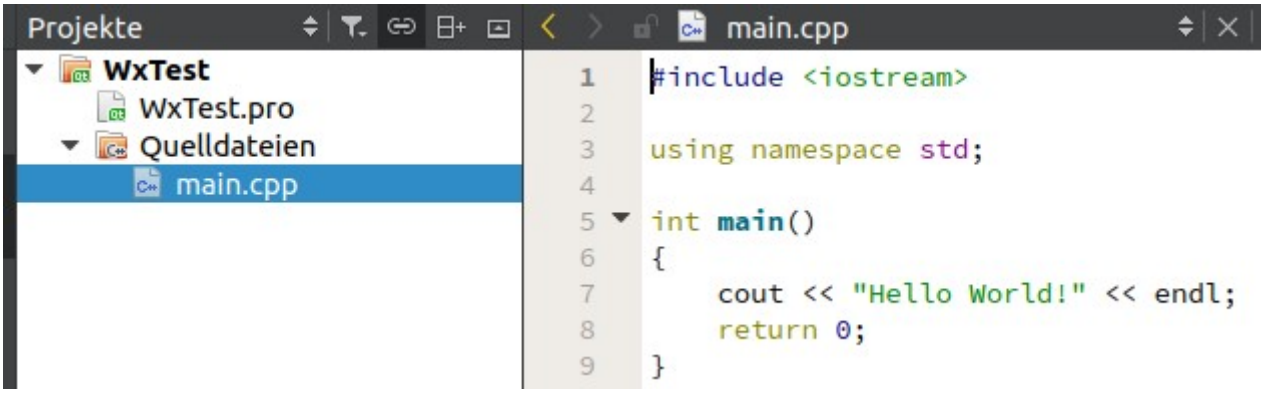
Hier muss nichts verändert werden.



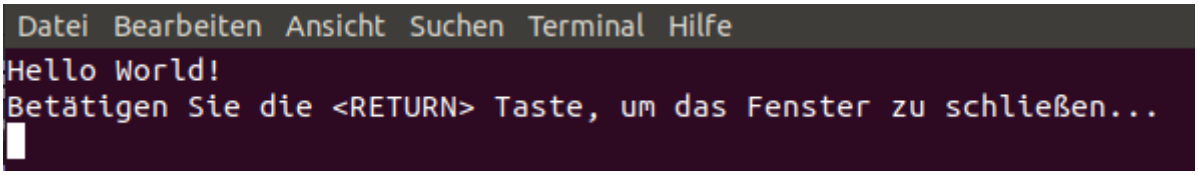
Auch hier gibt es nichts zu ändern und die Einstellungen können abgeschlossen werden.



Der Creator hat zwei Dateien erzeugt. Die Datei „main.cpp“ mit dem automatisch erzeugten Programmcode und die Projektdatei „WxTest.pro“.



Dieses kleine Programm schreibt nur den Text „Hello World!“ in die Konsole. Es kann bereits im QtCreator compiliert und ausgeführt werden (grüner Pfeil unten links, oder im Menü Erstellen → ausführen, oder Tastatur Strg+r).

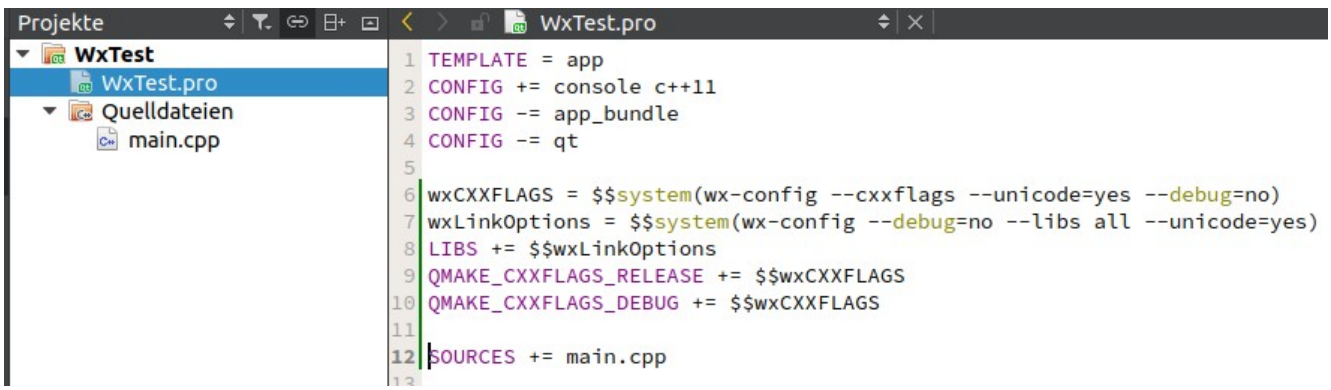


Um mit wxWidgets zu arbeiten sind noch die wxWidgets-Bibliotheken in das Projekt aufzunehmen. Dazu in der Projektdatei „WxTest.pro“ durch die Zeilen:

```
wxCXXFLAGS = $$system(wx-config --cxxflags --unicode=yes --debug=no)
wxLinkOptions = $$system(wx-config --debug=no --libs all --unicode=yes)
LIBS += $$wxLinkOptions
QMAKE_CXXFLAGS_RELEASE += $$wxCXXFLAGS
QMAKE_CXXFLAGS_DEBUG += $$wxCXXFLAGS
```

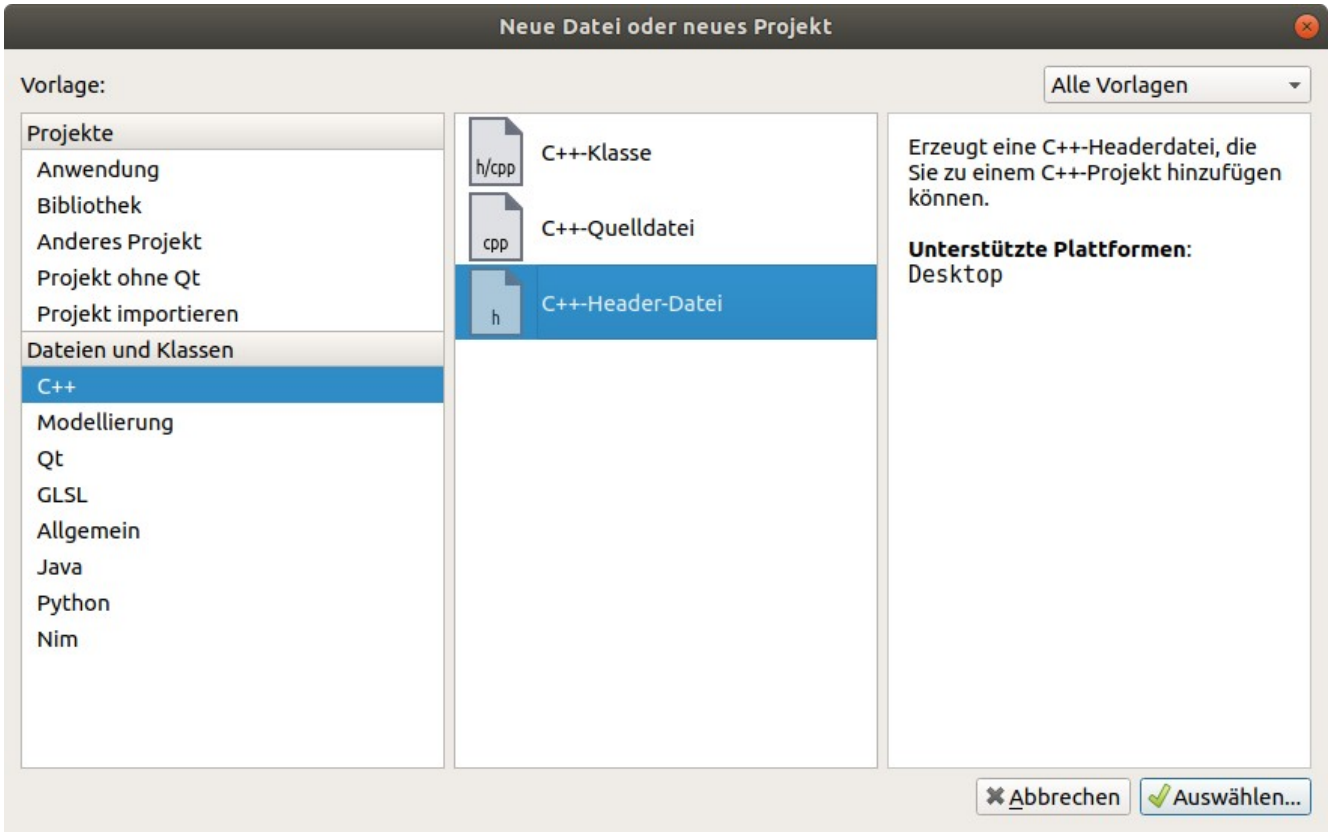
die Pfade zu den Bibliothek angeben und die Library zum linken aufnehmen. Werden noch andere Bibliotheken benötigt, werden von diesen auch die Header in die Quelldateien und die Libs ins Projekt aufgenommen.

Der Eintrag „--libs all“ wird benötigt, auch die wxStyledTextCtrl lib aufzunehmen, diese Lib entspricht der Scintilla EditorLib. Wird diese nicht benötigt, reicht „--libs“.



Nun das Programm umschreiben zur wxWidget-Application.

Zunächst eine Header-Datei für main erstellen.



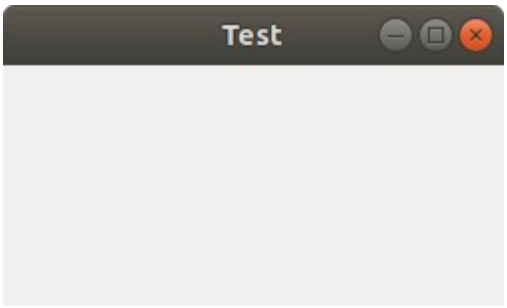
Inhalt der Header Datei **Main.h:**

```
#ifndef MAIN_H
#define MAIN_H

#include <wx/wx.h>

class MyApp : public wxApp
{
  public:
    virtual bool OnInit();
};

class Test : public wxFrame
{
public:
    Test(const wxString& title);

};

#endif // MAIN_H
```

Jetzt die **Main.cpp** Datei für wxWidgets ändern:

```
#include "main.h"

IMPLEMENT_APP(MyApp)

Test::Test(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 150))
{
  Centre();
}

bool MyApp::OnInit()
{
    Test *test = new Test(wxT("Test"));
    test->Show(true);
    return true;
}
```

Das Projekt nun compilieren und ausführen sollte zu diesem Ergebnis führen:



Damit ist die Entwicklungsumgebung fertig und es können jetzt Programme mit wxWidgets unter Qt-Creator geschrieben und getestet werden.

3 Eine Applikation erstellen

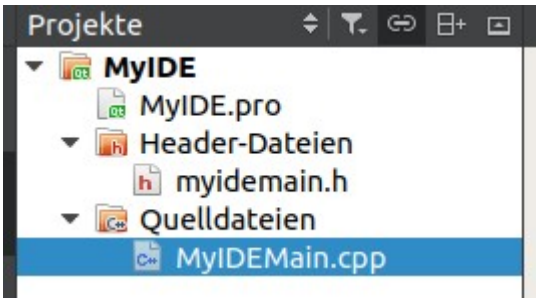
Im folgenden werde ich mir schrittweise eine Applikation aufbauen, welche als Grundgerüst für Mikrocontroller-IDE‘s verwendet werden kann. Letztlich wird daraus eine IDE für meine MCS51-Platinen IS51 bis IS5122.

Ich beginne mir, wie unter Kapitel 2 beschrieben, ein neues Projekt anzulegen. Ich werde die meisten Schritte nicht detailiert beschreiben, da sie in der Dokumentation zu wxWidgets gut erklärt sind.

Zuerst im gewünschten Pfad einen Ordner für das Projekt anlegen. Ich wähle: /home/tom/Ldaten/QtProgs/MyIDE. Nicht vergessen die Projektdatei „xxxxxx.pro“ wie beschrieben anzupassen, eine Headerdatei erzeugen und die Maindatei zu ändern.

3.1 Ein Frame für die Applikation

Jede Applikation besteht aus einem Frame mit einem oder mehreren Panel‘s zur Aufnahme der Komponenten. Zunächst besteht die Applikation nur aus einem Frame (Rahmen), wie oben beschrieben. Das Projekt besteht aus den Dateien „MyIdeMain.cpp“, MyIdeMain.h“ und der Projekt-Datei „MyIDE.pro“.



Der Inhalt von **MyIDE.pro** ist:

```
TEMPLATE = app
CONFIG += console c++11
CONFIG -= app_bundle
CONFIG -= qt

wxCXXFLAGS = $$system(wx-config --cxxflags --unicode=yes --debug=no)
wxLinkOptions = $$system(wx-config --debug=no --libs all --unicode=yes)
LIBS += $$wxLinkOptions
QMAKE_CXXFLAGS_RELEASE += $$wxCXXFLAGS
QMAKE_CXXFLAGS_DEBUG += $$wxCXXFLAGS

SOURCES += \
    MyIDEMain.cpp

HEADERS += \
    myidemain.h
```

Der Inhalt von **MyIdeMain.h** ist:

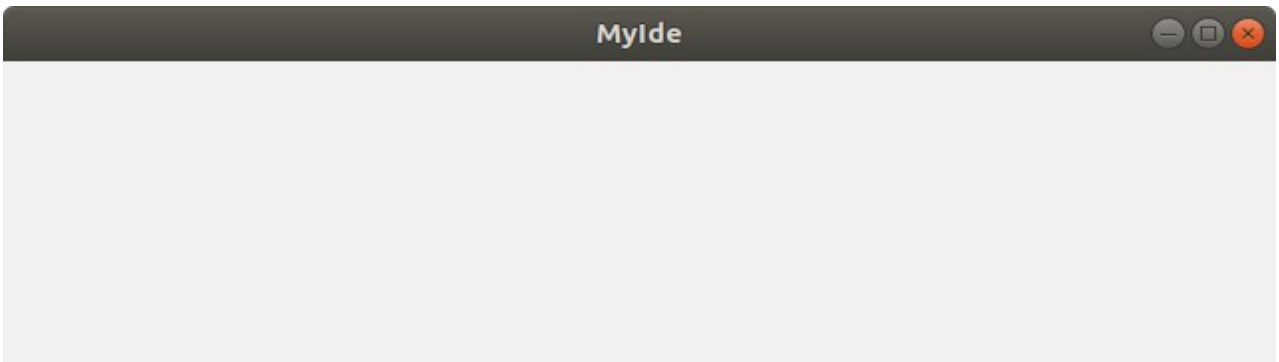
```
#ifndef MYIDEMAIN_H
#define MYIDEMAIN_H

#include <wx/wx.h>

class IDEFrame : public wxFrame
{
public:
    IDEFrame(const wxString& title);
```

```
};

class MyIdeApp : public wxApp
{
  public:
    virtual bool OnInit();
};

#endif // MYIDEMAIN_H
```

Und der Inhalt von **MyIdeMain.cpp** ist:

```
#include "MyIdeMain.h"

IDEFrame::IDEFrame(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
{
  Centre();
}

IMPLEMENT_APP(MyIdeApp)

bool MyIdeApp::OnInit()
{
    IDEFrame *myframe = new IDEFrame(wxT("MyIde"));
    myframe->Show(true);

    return true;
}
```

Dies ergibt folgende Applikation:



3.2 Ein Main-Menü

Ein Menü besteht aus dem Menübar, den Menüs und den Items der Menüs.

Menübar			
Menü	**Menü**	**Menü**	**Menü**
Menü-Item	Menü-Item	Menü-Item	Menü-Item
Menü-Item	Menü-Item	Menü-Item	Menü-Item

Die meisten Programme verfügen über ein oder mehrere Menübars, so auch meine Applikation. Und so will ich dem nackten Rahmen jetzt einen Menübar hinzufügen.

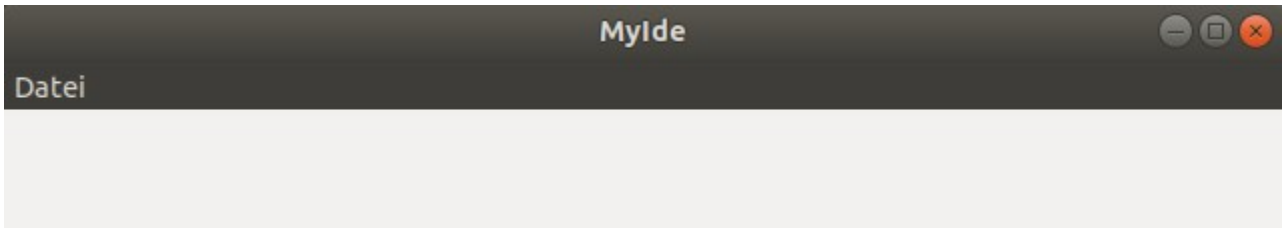
In die Header-Datei MyIdeMain.h wird

```
#include <wx/menu.h>
      class IDEFrame : public wxFrame
      { public:
          IDEFrame(const wxString& title);
wxMenuBar *menubar;        //Menübar
wxMenu *datei;             //Datei-Menü
void OnQuit(wxCommandEvent& event);   //Ereignisbehandlung Quit
      };
```

aufgenommen. Und in MyIdeMain.cpp

```
      IDEFrame::IDEFrame(const wxString& title)
            : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
      {
menubar = new wxMenuBar;       //Menübar
datei = new wxMenu;            //Datei-Menü
datei->Append(wxID_EXIT, wxT("&Beenden"));
menubar->Append(datei, wxT("&Datei"));
SetMenuBar(menubar);

Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnQuit));

        Centre();
      }

void IDEFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
{
  Close(true);
}
```

Nun sieht die Anwendung so aus und kann über Menü beendet werden.



Nun werde ich dem Menübar noch weitere Menüs hinzufügen.

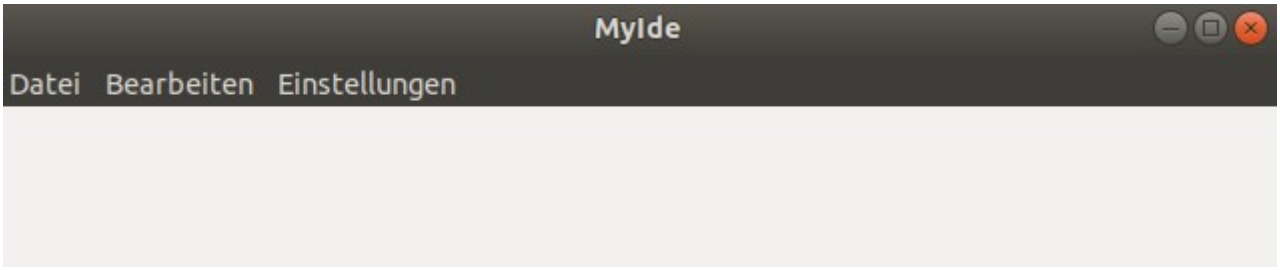
In der Header-Datei:

```
      class IDEFrame : public wxFrame
      { public:
          IDEFrame(const wxString& title);
          wxMenuBar *menubar;       //Menübar
          wxMenu *datei;            //Datei-Menü
wxMenu *bearbeiten;       //Bearbeiten-Menü
wxMenu *einstellungen;    //Einstellungen-Menü
          void OnQuit(wxCommandEvent& event);   //Ereignisbehandlung Quit
      };
```

In der CPP-Datei:

```
        datei = new wxMenu;           //Datei-Menü
bearbeiten = new wxMenu;      //Bearbeiten-Menü
einstellungen = new wxMenu;   //Einstellungen-Menü
        datei->Append(wxID_EXIT, wxT("&Beenden"));
        menubar->Append(datei, wxT("&Datei"));
menubar->Append(bearbeiten, wxT("&Bearbeiten"));
menubar->Append(einstellungen, wxT("&Einstellungen"));
        SetMenuBar(menubar);
```

Nun sieht es so aus:



Es braucht noch einige Items in den Menüs, doch zunächst werden Zugriffs-ID's für die Items definiert. Während für das Beenden des Programms die vordefinierte ID „wxID_EXIT" verwendet wird, bekommen die Anderen Menü-Items selbstdefinierte ID's.

Header:

```
enum
{ ID_MDOpen   = 100,    //Menü-Datei öffnen
  ID_MDNew    = 101,    //Menü-Datei Neu
  ID_MBBack   = 120,    //Menü-Bearbeiten Rückgängig
  ID_MEDevice = 140     //Menü-Einstellungen Baustein
};
```

Main:

```
        einstellungen = new wxMenu;   //Einstellungen-Menü
datei->Append(ID_MDOpen, wxT("&Oeffnen"), wxT("Bestehende Datei öffnen"));
datei->Append(ID_MDNew, wxT("&Neu"), wxT("Neue Datei öffnen"));
datei->AppendSeparator();
        datei->Append(wxID_EXIT, wxT("&Beenden"));
bearbeiten->Append(ID_MBBack, wxT("&Zurück"), wxT("Rueckgängig"));
einstellungen->Append(ID_MEDevice, wxT("&Baustein"), wxT("CPU-Baustein wählen"));
        menubar->Append(datei, wxT("&Datei"));
```

Damit werden im Datei-Menü die Items „Oeffnen“ und „Neu“ eingerichtet. Es folgt ein Trennstrich (datei→AppendSeparator();). Das Bearbeiten-Menü erhält einen Item „Zurück“ mit der ID „ID_MBBack“ und das Einstellungen-Menü den Item „Baustein“ mit der ID „ID_MEDevice“. Der zweite Text bei Append soll später in der Statuszeile erscheinen, aber diese habe ich noch nicht. Dieser Text kann auch entfallen.

Nun die Ereignisbehandlung der neuen Items. Da noch keine wirkliche Ereignisbehandlung vorhanden ist, sollen sie einfach nur Messageboxen anzeigen.

Header:

```
          wxMenu *einstellungen;    //Einstellungen-Menü
void OnOpen(wxCommandEvent& event);   //Event Datei-Öffnen
void OnNew(wxCommandEvent& event);    //Event Datei-Neu
          void OnQuit(wxCommandEvent& event);   //Event Datei-Quit
void OnBack(wxCommandEvent& event);   //Event Bearbeiten-Rückgängig
void OnDevice(wxCommandEvent& event); //Event Einstellungen-Baustein
      };
```

Main:

```
        Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
                wxCommandEventHandler(IDEFrame::OnQuit));
Connect(ID_MDOpen, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnOpen));
Connect(ID_MDNew, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnNew));
Connect(ID_MBBack, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnBack));
Connect(ID_MEDevice, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnDevice));

void IDEFrame::OnNew(wxCommandEvent& WXUNUSED(event))
{ wxMessageBox(wxT("Datei Neu gewählt"),
                wxT("Meldung"), wxOK | wxICON_INFORMATION);
}

void IDEFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
{ wxMessageBox(wxT("Datei öffnen gewählt"),
                wxT("Meldung"), wxOK | wxICON_INFORMATION);
}

void IDEFrame::OnBack(wxCommandEvent& WXUNUSED(event))
{ wxMessageBox(wxT("Bearbeiten - Rückgängig gewählt"),
                wxT("Meldung"), wxOK | wxICON_INFORMATION);
}

void IDEFrame::OnDevice(wxCommandEvent& WXUNUSED(event))
{ wxMessageBox(wxT("Einstellungen - Baustein gewählt"),
                wxT("Meldung"), wxOK | wxICON_INFORMATION);
}
```

3.3 Ein Toolbar

Viele Programme nutzen den Vorteil häufig benötigte Funktionen nicht mehrstufig in den Menüs zu erreichen, sondern direkt durch einen Klick auf einen Schnellstarter. Dafür gibt es den Toolbar mit seinen Schnellstart-Knöpfen. Ich sollte gut überlegen welche Funtionen darüber erreichbar sind. Ein Beenden-Knopf scheint mir da wenig Sinn zu machen, da er im Programmlauf nur ein einziges mal benötigt wird. Ein Knopf das Quellprogramm zu übersetzen wird dagegen oft benötigt.

Ein Problem, die Icons müssen sich in einem Pfad befinden, wo sie vom Programm gefunden werden. Das ist am sichersten erfüllt, wenn sie sich im gleichen Verzeichnis wie das Programm befinden.

Header:

Eine neue Event-ID und einen Zeiger auf einen Toolbar hinzufügen.

```
      enum
      { ID_MDOpen   = 100,    //Menü-Datei öffnen
        ...
ID_TBTest   = 201     //Toolbar Test
      };

      class IDEFrame : public wxFrame
      { public:
          IDEFrame(const wxString& title);
        ...
wxToolBar* toolBar;
      };
```

Main:

Das Icon **"trophy-gold.png"** wurde in das Arbeitsverzeichnis kopiert. SetToolShortHelp setzt den Text, welcher erscheint wenn der Mauscursor über dem Icon steht. **Bind** verbindet die ID „ID_TBTest“ mit der Event-Routine „OnNew“.

```
      IDEFrame::IDEFrame(const wxString& title)
             : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
      {
        ......
wxImage::AddHandler(new wxPNGHandler);
wxBitmap test(wxT("trophy-gold.png"), wxBITMAP_TYPE_PNG);
toolBar = new wxToolBar(this, wxID_ANY);
toolBar->AddTool(ID_TBTest, wxT("Ein Test"), test);
toolBar->SetToolShortHelp(ID_TBTest, wxT("Test"));
Bind(wxEVT_COMMAND_TOOL_CLICKED, &IDEFrame::OnNew, this, ID_TBTest);
toolBar->Realize ();
SetToolBar(toolBar);
```

So sieht es aus, wenn das Icon angewählt wurde:

Eine andere Möglichkeit ist, die Icons mit in den Programmcode aufzunehmen. Damit ist gleichzeitig sichergestellt, dass das Icon auf jedem Rechner verfügbar ist. Dazu muss das Icon im XML-Format vorliegen und so in eine Include-Datei aufgenommen werden. Zum bearbeiten / erstellen der Icons benutze ich das Programm „KolourPaint“, das direkt mit dem XML-Format umgehen kann. Als Header nutze ich eine Datei „Icons.h“ in die ich die Icons aufnehme.

XML eines Icon in Header **Icons.h:**

```
static const char *Run22[]={
  "22 22 2 1",
  "# c #0000ff",
  ". c #ffffff",
  "......................",
  "......................",
  "......................",
  "......................",
  "......................",
  "............#.......",
  "............##......",
  "............###.....",
  "............####....",
  "...################...",
  "...#################..",
  "...#################..",
  "...################...",
  "............####....",
  "............###.....",
  "............##......",
  "............#.......",
  "......................",
  "......................",
  "......................",
  "......................",
  "......................"};
```

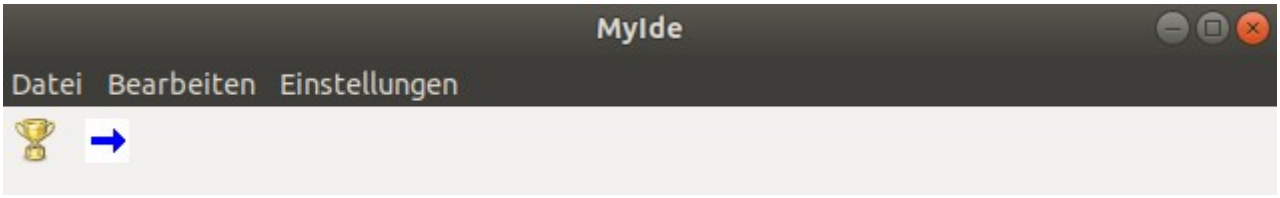
Im **Header:**

```
#include "Icons.h"
```

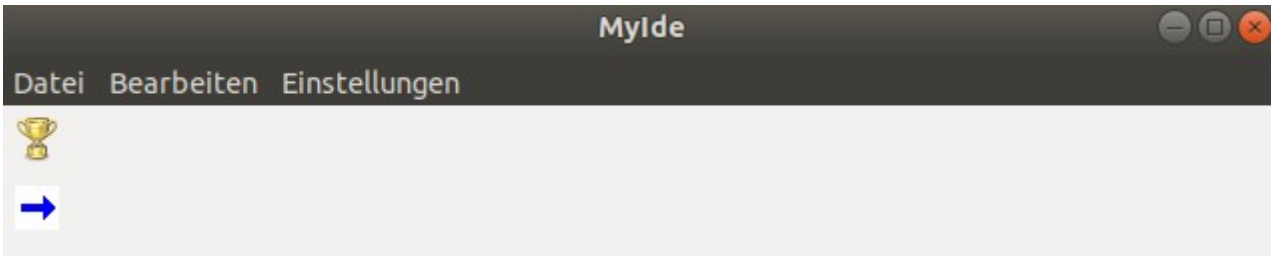
In Main:

```
      IDEFrame::IDEFrame(const wxString& title)
             : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
      {
        ……
toolBar = new wxToolBar(this, wxID_ANY, wxDefaultPosition, wxDefaultSize,
    wxTB_HORIZONTAL|wxNO_BORDER|wxTB_FLAT);
//  toolBar = new wxToolBar(this, wxID_ANY);
        toolBar->AddTool(ID_TBTest, wxT("Ein Test"), test);
        toolBar->SetToolShortHelp(ID_TBTest, wxT("Test"));
toolBar->AddTool(ID_TBRun, wxT("Run"), Run22);
toolBar->SetToolShortHelp(ID_TBRun, wxT("Run"));

        Bind(wxEVT_COMMAND_TOOL_CLICKED, &IDEFrame::OnNew, this, ID_TBTest);
Bind(wxEVT_COMMAND_TOOL_CLICKED, &IDEFrame::OnOpen, this, ID_TBRun);
```

Hier wird der Toolbar anders aufgenommen, um ihn auch vertical (wxTB_HORIZONTAL / wxTB_VERTICAL) darstellen zu können. AddTool nimmt das neue Icon aus Icons.h auf. Mit „Bind“ wird das Ereignis „ID_TBRun“ an die Routine „OnOpen“ gebunden.




```
toolBar = new wxToolBar(this, wxID_ANY, wxDefaultPosition, wxDefaultSize,
    wxTB_VERTICAL|wxNO_BORDER|wxTB_FLAT);
```



Durch vertikale Darstellung des Toolbar bleiben mehr Zeilen für Text in einem Texteditor des Hauptfensters.

3.4 Ein Panel

Ein Panel dient der Aufnahme anderer Komponenten als gemeinsamer Rahmen. Ein Programm verfügt meist über ein oder mehrere Panels.

Hier wird auch gleich ein Sizer für das Panel mit aufgenommen.

Header:

```
        wxToolBar* toolBar;
wxPanel* panel;
wxBoxSizer* panelSizer;
    };
```

Main:

```
    IDEFrame::IDEFrame(const wxString& title)
            : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
    {
// Ein Top-Level Panel zur Aufnahme des Inhalts des Frame
panel = new wxPanel(this, wxID_ANY);
```

3.4.1 Sizer für das Panel

?

Main:

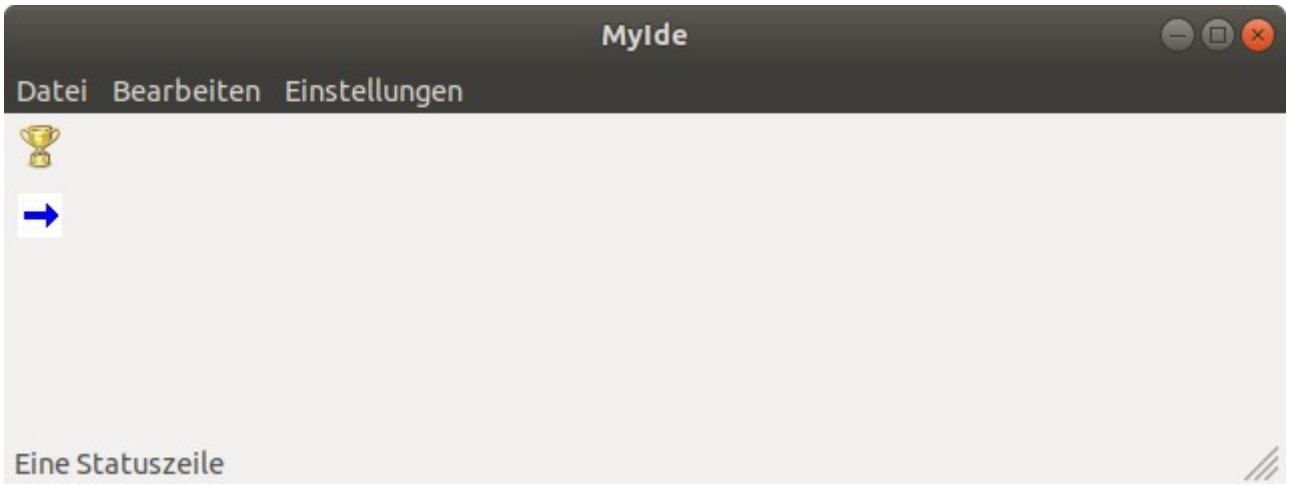
```
panelSizer = new wxBoxSizer(wxHORIZONTAL);
panel->SetSizer(panelSizer);
```

3.5 Eine Statuszeile

Eine Statuszeile am unteren Bildrand kann zusätzliche Informationen anzeigen. In ihr werden die bereits definierten Meldungen nun angezeigt.

Main:

```
	IDEFrame::IDEFrame(const wxString& title)
		: wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
	{
	……..
CreateStatusBar();
SetStatusText("Eine Statuszeile");
	……..
```



3.6 Ein Projekt-Notebook

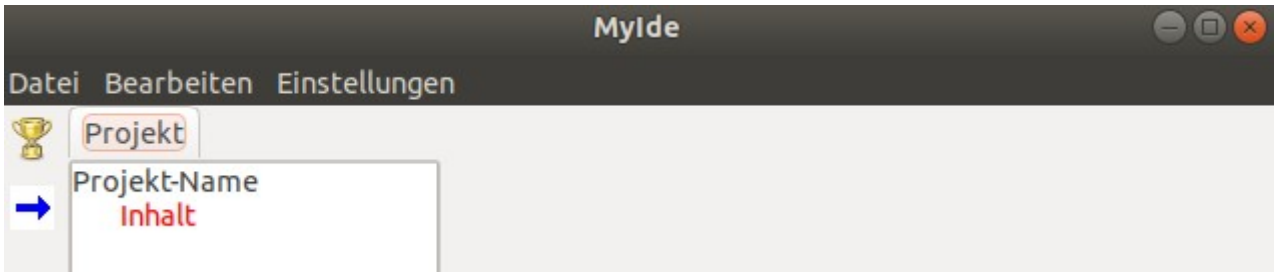
In einem Notebook werden mehrere Kompnenten, z.B: Editoren, unter einem gemeinsamen Rahmen dargestellt. Über Reiter wird die gewünschte Komponente ausgewählt.
Hier wird ein Notebook erstellt und darin eine Textanzeige (wxTextCtrl) dargestellt.

Header:

```
#include <wx/notebook.h>
#include <wx/textctrl.h>

    wxNotebook* ProNBook;           //Notebook für Projekt (links)
    wxTextCtrl* projektCtrl;        //Seite für Projekt-Notebook
```

Main:

```
//Das Projekt-Notebook erzeugen
    ProNBook = new wxNotebook(panel, wxID_ANY);

//Eine Seite zu Projekt-Notebook hinzufügen
    projektCtrl = new wxTextCtrl(ProNBook, wxID_ANY, L"Projekt-Name\n",
       wxDefaultPosition, wxSize(180, 300), wxTE_MULTILINE);
    ProNBook->AddPage(projektCtrl, L"Projekt");
    projektCtrl->SetDefaultStyle(wxTextAttr(*wxRED));
    projektCtrl->AppendText("\tInhalt\n");

        panelSizer = new wxBoxSizer(wxHORIZONTAL);
        panelSizer->Add(ProNBook, 0, wxEXPAND);      //Sizer des Notebook
        panel->SetSizer(panelSizer);
```



3.6.1 Weitere Textfenster einfügen

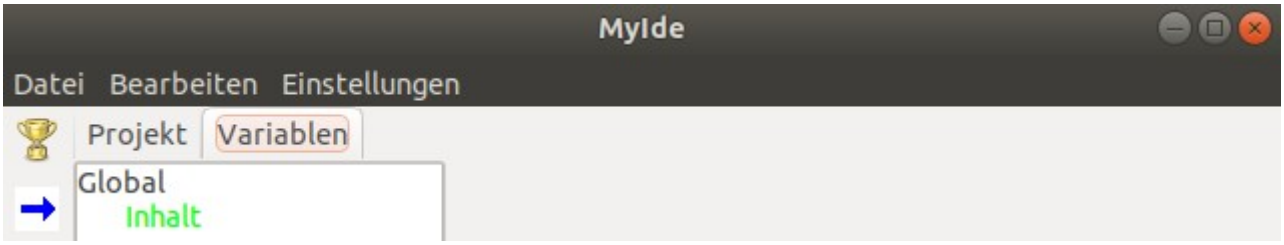
Ein Notebook kann mehrere Fenster aufnehmen, so soll hier ein weiteres Textfenster hinzugefügt werden.

Header:

```
    wxTextCtrl* VarsCtrl;        //Weitere Seite für Projekt-Notebook
```

Main:

```
//Eine weitere Seite zu Projekt-Notebook hinzufügen
    VarsCtrl = new wxTextCtrl(ProNBook, wxID_ANY, L"Global\n",
       wxDefaultPosition, wxSize(180, 300), wxTE_MULTILINE);
    ProNBook->AddPage(VarsCtrl, L"Variablen");
    VarsCtrl->SetDefaultStyle(wxTextAttr(*wxGREEN));
    VarsCtrl->AppendText("\tInhalt\n");
```


MyIde
Datei
Bearbeiten
Einstellungen
Projekt
Variablen
Global
Inhalt

3.7 Ein Editor-Notebook

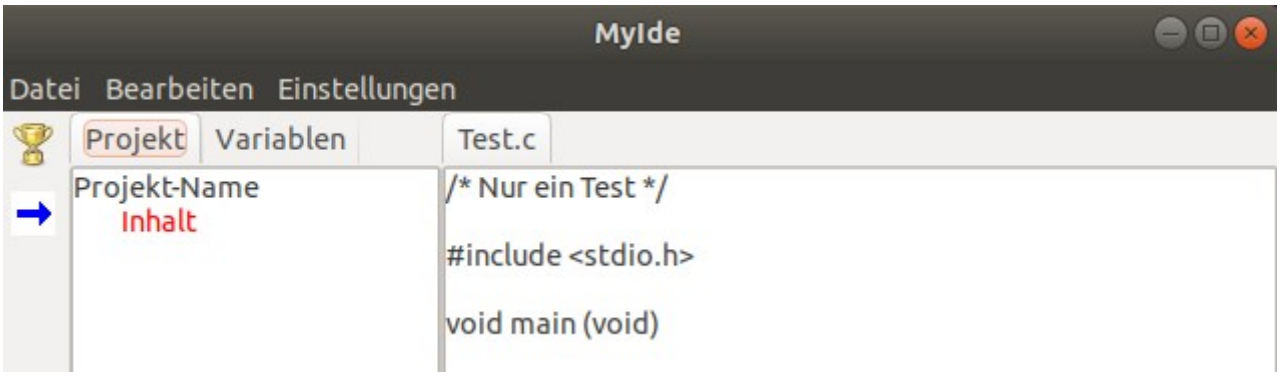
Im Editor-Notebook befinden sich die Sourcedateien der zu erstellenden Programme um hier bearbeitet zu werden.

Header:

```
#include <wx/stc/stc.h>

    wxNotebook* EditNBook;    //Notebook für Editoren (rechts)
    wxTextCtrl* EditCtrl;     //Seite für Editor-Notebook
```

Main:

```
//Das Projekt-Notebook und Editor-Notebook erzeugen
  ProNBook = new wxNotebook(panel, wxID_ANY);
  EditNBook = new wxNotebook(panel, wxID_ANY);

//Eine Seite zu Editor-Notebook hinzufügen
  EditCtrl = new wxTextCtrl(EditNBook, wxID_ANY, L"/* Nur ein Test */\n\n",
     wxDefaultPosition, wxSize(180, 300), wxTE_MULTILINE);
  EditNBook->AddPage(EditCtrl, L"Test.c");
  EditCtrl->AppendText(L"#include <stdio.h>\n\nvoid main (void)\n");

  panelSizer->Add(ProNBook, 0, wxEXPAND);       //Sizer des Notebook
  panelSizer->Add(EditNBook, 1, wxEXPAND);
```



3.8 Ein Status-Notebook

Da ein einfacher wxBoxSizer nur Komponenten neben- oder unter-einander darstellen kann und mir ein wxFlexGridSizer hier zu aufwändig erscheint, entscheide ich mich im Frame ein MainPanel (mPanel) aufzunehmen, das ein weiteres Panel (ePanel) für das Projekt- und Editor-Notebook enthält. Es befindet sich oben, direkt unter dem Menübar. Im Main-Panel befindet sich dann auch das Status-Notebook, am unteren Bildrand (über der Statuszeile).

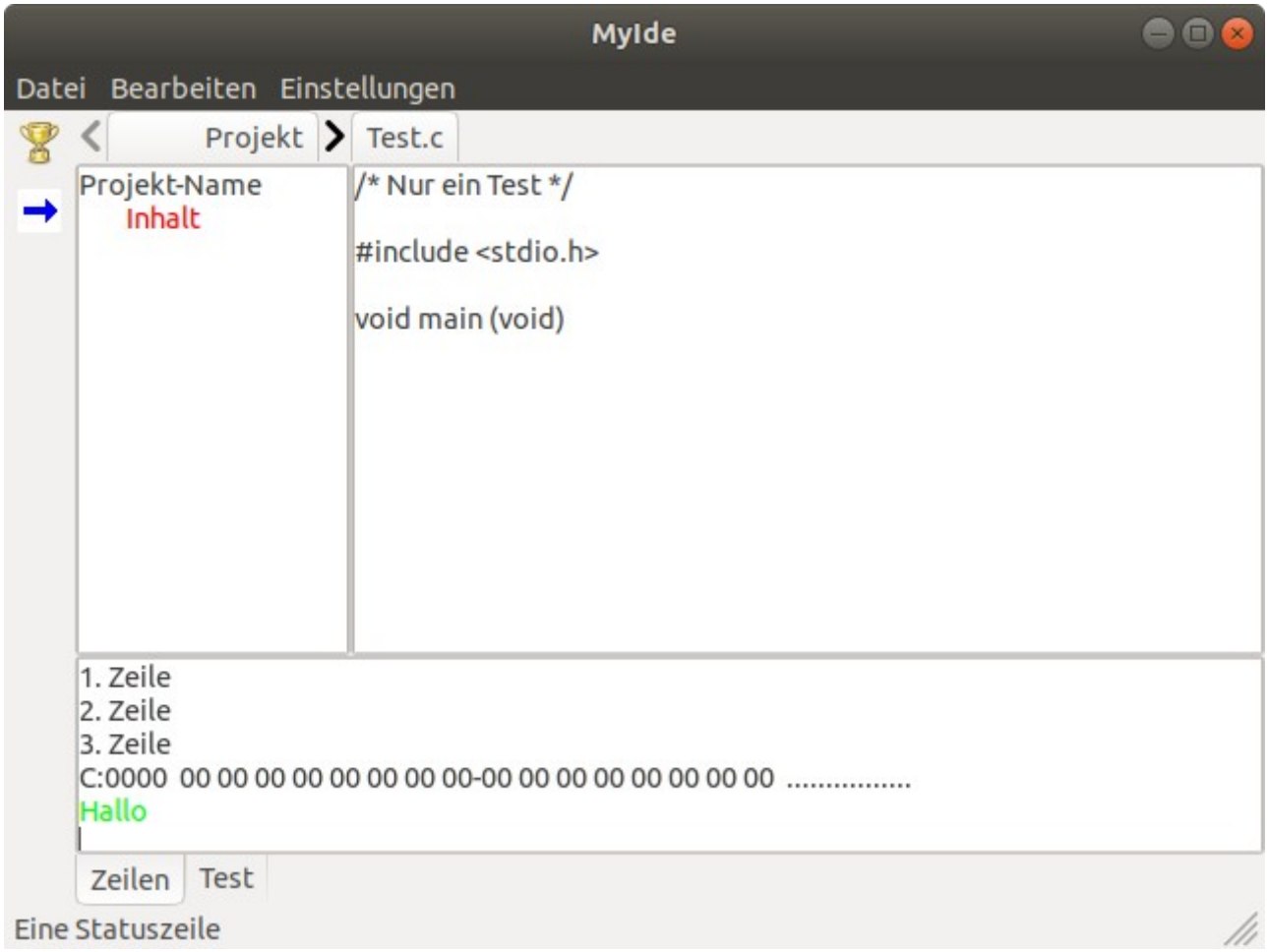
Header:

```
  wxPanel *mPanel;            //Main-Panel (auch für Status-Notebook)
  wxBoxSizer *MSSizer;        //Main/Status-Sizerj
  wxPanel *ePanel;            //Projekt/Editor-Panel
  wxBoxSizer *PESizer;        //Projekt/Editor-Sizer
  wxNotebook* StatusNBook;    //Notebook für Status (unten)
  wxTextCtrl* StatusCtrl;     //Seite für Status-Notebook
  wxTextCtrl* NextCtrl;       //Seite für Status-Notebook
```

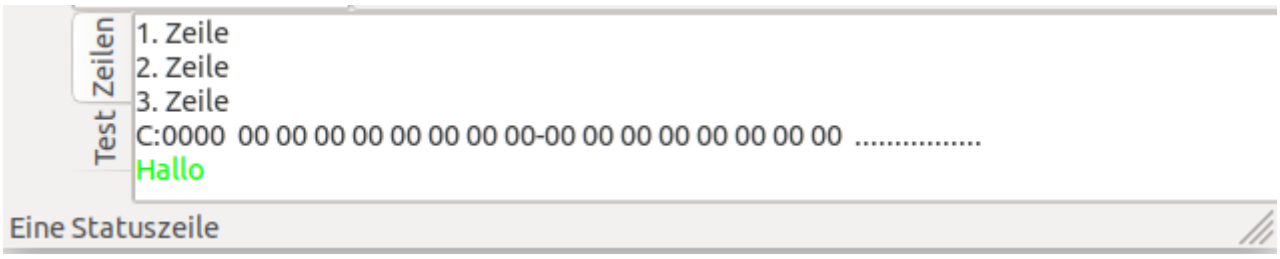
Main:

```
//Ein Top-Level Panel zur Aufnahme aller Inhalte des Frame erzeugen.
  mPanel = new wxPanel(this, wxID_ANY);
//Ein Panel im Panel
  ePanel = new wxPanel(mPanel, wxID_ANY);
//Das Projekt-Notebook und Editor-Notebook erzeugen
  ProNBook = new wxNotebook(ePanel, wxID_ANY, wxDefaultPosition, wxSize(140, -1));
  EditNBook = new wxNotebook(ePanel, wxID_ANY);
  StatusNBook = new wxNotebook(mPanel, wxID_ANY, wxDefaultPosition,
    wxDefaultSize , wxNB_BOTTOM | wxNB_FIXEDWIDTH);

//Eine Seite zu Projekt-Notebook hinzufügen
    projektCtrl = new wxTextCtrl(ProNBook, wxID_ANY, L"Projekt-Name\n",
       wxPoint(0, 0), wxSize(200, 100), wxTE_MULTILINE);
    ProNBook->AddPage(projektCtrl, L"Projekt");
    projektCtrl->SetDefaultStyle(wxTextAttr(*wxRED));
    projektCtrl->AppendText("\tInhalt\n");
//Eine weitere Seite zu Projekt-Notebook hinzufügen
    VarsCtrl = new wxTextCtrl(ProNBook, wxID_ANY, "Global\n",
       wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
    ProNBook->AddPage(VarsCtrl, "Variablen");
    VarsCtrl->SetDefaultStyle(wxTextAttr(*wxGREEN));
    VarsCtrl->AppendText("\tInhalt\n");

//Eine Seite zu Editor-Notebook hinzufügen
  EditCtrl = new wxTextCtrl(EditNBook, wxID_ANY, L"/* Nur ein Test */\n\n",
     wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
  EditNBook->AddPage(EditCtrl, L"Test.c");
  EditCtrl->AppendText(L"#include <stdio.h>\n\nvoid main (void)\n");

//Eine Seite zu Status-Notebook hinzufügen
  StatusCtrl = new wxTextCtrl(StatusNBook, wxID_ANY, wxT("1. Zeile\n2. Zeile\n3.
      Zeile\n"), wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
  StatusNBook->AddPage(StatusCtrl, wxT("Zeilen"));
  NextCtrl = new wxTextCtrl(StatusNBook, wxID_ANY, wxT("Dies\nist\nein\nTest\n"),
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
  StatusNBook->AddPage(NextCtrl, wxT("Test"));
  StatusCtrl->AppendText( wxT("C:0000  00 00 00 00 00 00 00 00-
      00 00 00 00 00 00  00 00  ................\n"));
  StatusCtrl->SetDefaultStyle(wxTextAttr(*wxGREEN));
  StatusCtrl->AppendText(wxT("Hallo\n"));

// Sizer für das Projekt/Editor-Notebook
```

```
PESizer = new wxBoxSizer(wxHORIZONTAL);
PESizer->Add(ProNBook, 0, wxEXPAND);        //Sizer des Projekt-Notebook
PESizer->Add(EditNBook, 5, wxEXPAND);       //Sizer des Editor-Notebook
ePanel->SetSizer(PESizer);
```



Die Reiter des Status-Notebook (unten „Zeilen“ / „Test“) befinden sich am unteren Rand des Panel. Ich hätte sie gerne links oder rechts davon angeordnet, dadurch wird der Titel des Reiters aber vertikal dargestellt. Wie ich ihn horizontal angezeigt bekomme, konnte ich nicht herausfinden. Möchte mich jetzt nicht zu lange damit aufhalten, ist etwas für später.



3.9 Dialoge

Es gibt drei Arten von Dialogen, Nachrichten- (Message), vordefinierte- (predfined) und benutzerdefinierte- (custom) Dialoge.

Message-Dialoge

sollen Meldungen an den Benutzer anzeigen. Sie sind teilweise vordefiniert und sind an die Bedürfnisse anpassbar.

Main:

```
void IDEFrame::OnNew(wxCommandEvent& WXUNUSED(event))
{ wxMessageBox(wxT("Datei Neu gewählt"),
                  wxT("Meldung"), wxOK | wxICON_INFORMATION);
}
```

wxOK wählt den Button und wxICON_INFORMATION das dargestellte Symbol.

Weitere Icon: `wxICON_ERROR, wxICON_QUESTION, wxICON_EXCLAMATION`

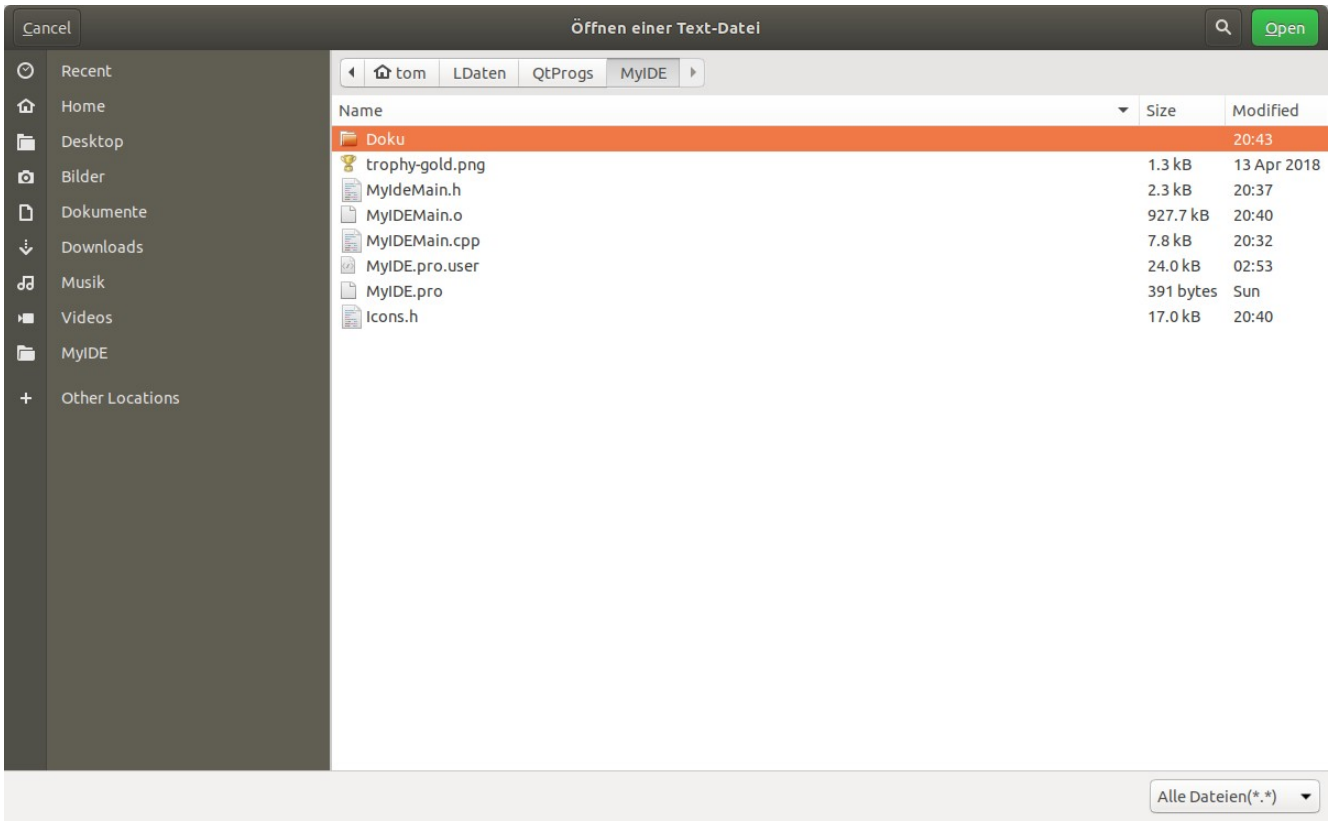
Vordefinierte Dialoge

sind bereits fertig definiert, um eine bestimmte, häufig vorkommende Aufgabe, zu erledigen. Das kann das öffnen und speichern von Dateien oder die Auswahl einer Farbe sein. Hier ein Beispiel für das öffnen einer Datei im Editor.

Header: `void OnOpen(wxCommandEvent& event);  //Event: Datei-Öffnen`

Main:

```
// Datei->Öffnen Dialog
void IDEFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
{ wxFileDialog *fdlg = new wxFileDialog(this, wxT("Datei öffnen"),wxT(""),wxT(""),
    wxT("Alle Dateien(*.*)|*.*|C Dateien(*.c*)|*.c*|Header(*.h)|*.h|ASM
Dateien(*.a*)|*.a*"),
    wxFC_OPEN, wxDefaultPosition);
  if(fdlg->ShowModal() == wxID_OK )
  { EditCtrl->LoadFile(fdlg->GetFilename());
    EditNBook->SetPageText(EditNBook->GetSelection(), fdlg->GetFilename());
  }
  fdlg->Destroy();
}
```



Benutzerdefinierte Dialoge

sind vollständig vom Programmentwickler definiert, um seine speziellen Aufgaben zu erledigen. Dazu können viele der Komponenten von wxWidgets innerhalb des Dialogs Verwendung finden. Jeder Dialog ist eine eigene Klasse.

Im Beispiel wird ein Dialog zum eingeben einer Taktfrequenz erzeugt. Er besteht aus drei RadioButton für vorgegebene Frequenzen und einem Textfeld zur Eingabe einer Zahl.

Header:

```
#include <wx/dialog.h>

  uint32_t GetClock() { return this->ui32Clock; }
  void SetClock(uint32_t uClock) { this->ui32Clock = uClock; }
  protected:
    uint32_t ui32Clock;          //8051 Taktfrequenz
```

Im Header wird die Variable der Frequenz protected deklariert mit zwei public Zugriffs-Methoden (Get und Set).

Main:

```
// Ereignisbehandlung
void IDEFrame::OnDevice(wxCommandEvent& WXUNUSED(event))
{ wxString mystring;
  TaktDialog* bd = new TaktDialog(wxT("Benutzer-Dialog"));
  bd->ui32Takt = ui32Clock;
//Richtigen Button aktivieren
  switch(ui32Clock)
  { case 24000000:
      bd->rb->SetValue(true);
      break;
    case 12000000:
      bd->rb1->SetValue(true);
```

```
      break;
    case 11059200:
      bd->rb2->SetValue(true);
      break;
    default:
      bd->rb3->SetValue(true);
      break;
  }
  mystring = wxString::Format(wxT("%d"), ui32Clock);
  bd->tc->SetValue(mystring);
  if(bd->ShowModal() == wxID_OK)
  { if(bd->rb->GetValue())
      ui32Clock = 24000000;
    else if(bd->rb1->GetValue())
      ui32Clock = 12000000;
    else if(bd->rb2->GetValue())
      ui32Clock = 11059200;
    else if(bd->rb3->GetValue())
      ui32Clock = wxAtoi(bd->tc->GetValue());     //Wert übernehmen
//Zahl(ui32Clock) in String(mystring) wandeln
    mystring = wxString::Format(wxT("%d Hz"), ui32Clock);
    wxMessageBox(mystring, wxT("Meldung"), wxOK | wxICON_INFORMATION);
    bd->ui32Takt = ui32Clock;
  }
  bd->Destroy();
}

//Konstruktor
TaktDialog::TaktDialog(const wxString & title)
       : wxDialog(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 230))
{
  wxPanel *panel = new wxPanel(this, -1);
    wxBoxSizer *vbox = new wxBoxSizer(wxVERTICAL);
    wxBoxSizer *hbox = new wxBoxSizer(wxHORIZONTAL);
    st = new wxStaticBox(panel, -1, wxT("Takt in Hz"),
        wxPoint(5, 5), wxSize(240, 150));
    rb = new wxRadioButton(panel, -1,
        wxT("24000000"), wxPoint(15, 30), wxDefaultSize, wxRB_GROUP);
    rb1 = new wxRadioButton(panel, -1,
        wxT("12000000"), wxPoint(15, 55));
    rb2 = new wxRadioButton(panel, -1,
        wxT("11059200"), wxPoint(15, 80));
    rb3 = new wxRadioButton(panel, -1,
        wxT("Benutzer"), wxPoint(15, 105));
    tc = new wxTextCtrl(panel, -1, wxString::Format(wxT("%d"), ui32Takt),
        wxPoint(100, 103));
    wxButton *okButton = new wxButton(this, wxID_OK, wxT("Ok"),
        wxDefaultPosition, wxSize(70, 30));
    wxButton *closeButton = new wxButton(this, wxID_CANCEL, wxT("Abbruch"),
        wxDefaultPosition, wxSize(70, 30));
    hbox->Add(okButton, 1);
    hbox->Add(closeButton, 1, wxLEFT, 5);
    vbox->Add(panel, 1);
    vbox->Add(hbox, 0, wxALIGN_CENTER | wxTOP | wxBOTTOM, 10);
    SetSizer(vbox);
    Centre();
}
```

In Main befindet sich der Konstruktor und die Ereignisbehandlung.


Benutzer-Dialog
Takt in Hz
24000000
12000000
11059200
Benutzer
12000000
Ok
Abbruch

3.10 Eine Konfigurationsdatei

Die Konfigurationsdatei wird beim ersten Start des Programms automatisch angelegt und mit Default-Werten beschrieben. Die Werte sind veränderbar und werden sofort und bei erneuten Programmstarts übernommen. Im Beispiel wird die Taktfrequenz des Zielsystems in der Config-Datei abgespeichert und beim Programmstart von dort gesetzt.

Header:

```
#include <wx/stdpaths.h>
#include <wx/fileconf.h>

   wxFileConfig *config;      //Config-Datei
   bool PrgConfig(void);   //Programm aus Config-Datei konfigurieren
```

Main:

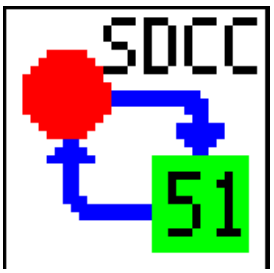
```
//Konstruktor
  SetStatusText("Eine Statuszeile");
   Centre();
PrgConfig();      //Programm aus Config-Datei konfigurieren
}

//Destruktor
IDEFrame::~IDEFrame()
{
  delete config;
}

// Ereignisbehandlung
void IDEFrame::OnDevice(wxCommandEvent& WXUNUSED(event))

.....
   bd->ui32Takt = ui32Clock;
//In Config-Datei übernehmen
config->SetPath("/zielsystem");
config->Write(wxT("Takt"), ui32Clock);
config->Flush();
   }

//Programm aus Config-Datei konfigurieren
bool IDEFrame::PrgConfig(void)
{ wxString cfgFile = wxStandardPaths::Get().GetExecutablePath() + ".cfg";
  config = new wxFileConfig( "", "", cfgFile);
  if(!wxFile::Exists(cfgFile))      //Config existiert?
  { config->SetPath("/zielsystem");     //Falls nein, Defaultwerte setzen
    config->Write(wxT("Takt"), 12000000);
//Nur zun testen
    config->SetPath("/dateien");
    config->Write(wxT("datei"), wxT("unbenannt.txt"));
//Test Ende
    config->Flush(true);
  }
//Programm aus Config-Datei initialisieren
  config->SetPath("/zielsystem");
  SetClock(config->Read(wxT("Takt"), true));
  return true;
}
```

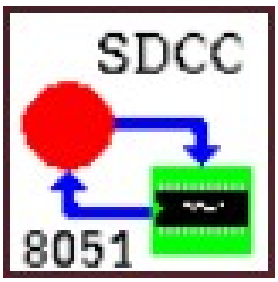
3.11 Ein Icon für das Programm

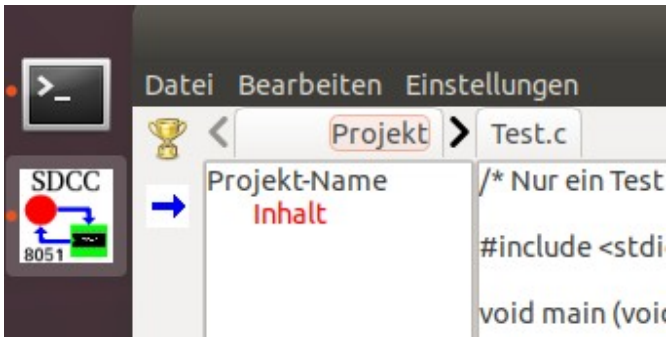
Das Programmicon wird wieder mit KolourPaint erstellt und als „.XML“ in die Icon-Headerdatei aufgenommen. Ich habe hier zwei Icons erstellt, eines mit 32x32 Pixeln (oben) und eines mit 64x64 Pixeln (unten). Der gezeigte Code in „Icons.h“ ist für 32x32.

Header **Icons.h:**

```
static const char *Logo32[]={
"32 32 5 1",
"# c #000000",
"b c #0000ff",
"c c #00ff00",
"a c #ff0000",
". c #ffffff",
"................................",
"............###.###...###..###.",
"...........#....#..#.#....#....",
"...........#....#..#.#....#....",
"............##..#..#.#....#....",
".....aaaaa.....#.#..#.#....#....",
"....aaaaaaa....#.#..#.#....#....",
"...aaaaaaaaa###..###...###..###.",
"..aaaaaaaaaa....................",
"..aaaaaaaaaaa...................",
"..aaaaaaaaaaabbbbbbbbbbb........",
"..aaaaaaaaaaabbbbbbbbbbbb.......",
"..aaaaaaaaaaa.........bbb.......",
"...aaaaaaaaa...........bb.......",
"....aaaaaaa...........bbbbbb.....",
".....aaaaa...........bbbbbb.....",
".......bb.............bbbb......",
"......bbbb.............bb.......",
".....bbbbbb.......ccccccccccc..",
".......bb.........ccccccccccc..",
".......bb.........cc###ccc#cc..",
".......bb.........cc#ccccc##cc..",
".......bb.........cc#cccccc#cc..",
".......bbb........cc###cccc#cc..",
"........bbbbbbbbbbcccccc#ccc#cc..",
".........bbbbbbbbbcccccc#ccc#cc..",
"..................ccccc#ccc#cc..",
"..................cc###ccc###c..",
"..................ccccccccccc..",
"..................ccccccccccc..",
"................................",
"................................"};
```







Main: SetIcon(wxIcon(Logo32)); //oder Logo64

3.12 Programmcode

Hier der Code des bisherigen Programms im Zusammenhang.

Icons.h:

```
#ifndef ICONS_H
#define ICONS_H

/* XPM */
static const char *Run22[]={
  "22 22 2 1",
  "# c #0000ff",
  ". c #ffffff",
  "......................",
  "......................",
  "......................",
  "......................",
  "......................",
  "..............#.......",
  "..............##......",
  "..............###.....",
  "..............####....",
  "...################...",
  "...#################..",
  "...#################..",
  "...################...",
  "..............####....",
  "..............###.....",
  "..............##......",
  "..............#.......",
  "......................",
  "......................",
  "......................",
  "......................",
  "......................"};

static const char *Logo64[]={

    "64 64 6 1",
    "a c #000000",
    "c c #0000ff",
    "d c #00ff00",
    "# c #cccccc",
    "b c #ff0000",
    ". c #ffffff",
    "................................................................",
    "................................................................",
    "................................................................",
    "................................................................",
    "................................................................",
    "......................#aaaa...aaaaaa#.....#aaa#....#aaa#.......",
    "....................#a#.#aa...a##.#a#...#a#.#a#..#a#.#a#......",
    "....................aa...#a...a#...#a..#a....##.#a#...##......",
    "....................aa........a#....a#.aa.....#.aa.....#......",
    "....................aaaa#....a#....a#.a#.......a#............",
    "......................#aa#...a#....a#.a#.......a#............",
    "........................#a#..a#....a#.aa.......aa............",
    "....................a#....a#..a#...#a..#a#...##.#a#...##......",
```

```
    "......................aa#.#aa...a##.#a#...aa#.#a#..aa#.#a#......",
    ".......................aaaaa#..aaaaaa#.....#aaa#....#aaa#.......",
    "............................................................",
    "............................................................",
    "............bbbbb...........................................",
    ".........bbbbbbbbbb.........................................",
    ".......bbbbbbbbbbbb.........................................",
    ".....bbbbbbbbbbbbbb.........................................",
    "....bbbbbbbbbbbbbbbb........................................",
    "...bbbbbbbbbbbbbbbbbb.......................................",
    "...bbbbbbbbbbbbbbbbbb.......................................",
    "..bbbbbbbbbbbbbbbbbbbb......................................",
    "..bbbbbbbbbbbbbbbbbbbbbccccccccccccccccccccc................",
    "..bbbbbbbbbbbbbbbbbbbbbbccccccccccccccccccccc...............",
    "..bbbbbbbbbbbbbbbbbbbbbbccccccccccccccccccccc...............",
    "..bbbbbbbbbbbbbbbbbbbb..................cccc................",
    "...bbbbbbbbbbbbbbbbbbb..................ccc.................",
    "...bbbbbbbbbbbbbbbbbbb..................ccc.................",
    "...bbbbbbbbbbbbbbbbbbb..................ccc.................",
    "....bbbbbbbbbbbbbbbbb..................ccccccccc.............",
    ".....bbbbbbbbbbbbbbbb..................ccccccc...............",
    ".....bbbbbbbbbbbbbb.....................ccccc...............",
    "......bbbbbbbbbbbb.......................ccc................",
    "........bbbbbbbbbb.........................c.................",
    "...........bbbbb.................ddddddddddddddddddddddddd....",
    "............c...................ddddddddddddddddddddddddd....",
    "............ccc.................ddddddddddddddddddddddddd....",
    "...........ccccc.................ddddddddddddddddddddddddd....",
    "..........ccccccc................dd#d#d#d#d#d#d#d#d#d#d#dd....",
    "..........ccccccccc..............dd#d#d#d#d#d#d#d#d#d#d#dd....",
    "............ccc.................daaaaaaaaaaaaaaaaaaaaaad....",
    "............ccc.................daaaaaaaaaaaaaaaaaaaaaad....",
    "............ccc.................daaaaaaaaaaaaaaaaaaaaaad....",
    "............cccc................ddaaaaaa..a..aaaa.aaaaad....",
    "............ccccccccccccccccccccdddaaaaaaa.aa...aaaaaaad....",
    "............ccccccccccccccccccccddaaaaaaaaaaaaaaaaaaaaad....",
    ".............ccccccccccccccccccccdaaaaaaaaaaaaaaaaaaaaad....",
    "................................daaaaaaaaaaaaaaaaaaaaad....",
    "................................daaaaaaaaaaaaaaaaaaaaad....",
    "................................dd#d#d#d#d#d#d#d#d#d#d#dd....",
    "....#aaa...#aaa...aaaa#....aa.....dd#d#d#d#d#d#d#d#d#d#dd....",
    "....a#.#a..aa#aa..a####..#aaa.....ddddddddddddddddddddddd....",
    "...#a..#a.#a#.#a..a.....aa#a.....ddddddddddddddddddddddd....",
    "....a#.#a.#a..#a..aaaa#....#a.....ddddddddddddddddddddddd....",
    "....#aaa..#a..#a..a#.aa....#a.....ddddddddddddddddddddddd....",
    "....a###a.#a..#a.....#a....#a.....ddddddddddddddddddddddd....",
    "...#a..#a.#a#.#a..#..#a....#a.................................",
    "...#a#.#a..aa#aa..a#.a#....#a.................................",
    "....#aaa...#aaa...#aaa....aaaa................................",
    "............................................................"};

#endif // ICONS_H
```

MyIdeMain.h:

```
#ifndef MYIDEMAIN_H
#define MYIDEMAIN_H

#include <wx/wx.h>
#include <wx/menu.h>
#include "Icons.h"
#include <wx/notebook.h>
#include <wx/textctrl.h>
#include <wx/dialog.h>
#include <wx/stdpaths.h>
#include <wx/fileconf.h>

enum
{ ID_MDNew      = 101,    //Menü-Datei Neu
  ID_MDOpen     = 102,    //Menü-Datei öffnen
  ID_MDSave     = 103,    //Menü-Datei speichern
  ID_MDSaveAs   = 104,    //Menü-Datei speichern unter
  ID_MDClose    = 105,    //Menü-Datei schließen
  ID_MBBack     = 121,    //Menü-Bearbeiten Rückgängig
  ID_MEDevice   = 141,    //Menü-Einstellungen Baustein
  ID_MEAsm      = 161,    //Menü-Erstellen Assemblieren
  ID_MEComp     = 162,    //Menü-Erstellen Kompilieren
  ID_TBTest     = 201,    //Toolbar Test
  ID_TBRun      = 202,    //Toolbar Run
  ID_TBStep     = 203,    //Toolbar Step
  ID_TBBack     = 204,    //Toolbar Backstep
  ID_TBStepOut  = 205,    //Toolbar StepOut
  ID_TBStepOver = 206     //Toolbar StepOver
};

class IDEFrame : public wxFrame
{ public:
    IDEFrame(const wxString& title);
    ~IDEFrame();
    wxMenuBar *menubar;       //Menübar
    wxMenu *datei;            //Datei-Menü
    wxMenu *bearbeiten;       //Bearbeiten-Menü
    wxMenu *einstellungen;    //Einstellungen-Menü
    wxMenu *erstellen;        //Erstellen-Menü
    void OnNew(wxCommandEvent& event);    //Event: Datei-Neu
    void OnOpen(wxCommandEvent& event);   //Event: Datei-Öffnen
    void OnSave(wxCommandEvent& event);   //Event: Datei-Speichern
    void OnSaveAs(wxCommandEvent& event); //Event: Datei-Speichern unter
    void OnClose(wxCommandEvent& event);  //Event: Datei-Schließen
    void OnQuit(wxCommandEvent& event);   //Event: Datei-Quit
    void OnBack(wxCommandEvent& event);   //Event: Bearbeiten-Rückgängig
    void OnDevice(wxCommandEvent& event); //Event: Einstellungen-Baustein
    wxFileConfig *config;       //Config-Datei
    bool PrgConfig(void);       //Programm aus Config-Datei konfigurieren
    wxToolBar *toolBar;
    wxPanel *mPanel;            //Main-Panel (auch für Status-Notebook)
    wxPanel *ePanel;            //Projekt/Editor-Panel
    wxBoxSizer *PESizer;        //Projekt/Editor-Sizer
    wxBoxSizer *MSSizer;        //Main/Status-Sizer
    wxNotebook* ProNBook;       //Notebook für Projekt (links)
    wxTextCtrl* projektCtrl;    //Seite für Projekt-Notebook
    wxTextCtrl* VarsCtrl;       //Weitere Seite für Projekt-Notebook
    wxNotebook* EditNBook;      //Notebook für Editoren (rechts)
    wxTextCtrl* EditCtrl;       //Seite für Editor-Notebook
```

```
    wxNotebook* StatusNBook;    //Notebook für Status (unten)
    wxTextCtrl* StatusCtrl;     //Seite für Status-Notebook
    wxTextCtrl* NextCtrl;       //Seite für Status-Notebook
    uint32_t GetClock() { return this->ui32Clock; }
    void SetClock(uint32_t uClock) { this->ui32Clock = uClock; }
  protected:
    uint32_t ui32Clock;         //8051 Taktfrequenz
};

class MyIdeApp : public wxApp
{
  public:
    virtual bool OnInit();
};

class TaktDialog : public wxDialog
{ public:
    TaktDialog(const wxString& title);
  wxStaticBox *st;
  wxTextCtrl *tc;
  wxRadioButton *rb;
  wxRadioButton *rb1;
  wxRadioButton *rb2;
  wxRadioButton *rb3;
  uint32_t ui32Takt;
};

#endif // MYIDEMAIN_H
```

MyIdeMain.cpp:

```
/**********************************
 * Grundgerüst für µController-IDE  *
 **********************************/
#include "MyIdeMain.h"

IDEFrame::IDEFrame(const wxString& title)
       : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(700, 500))
{
//Ein Top-Level Panel zur Aufnahme aller Inhalte des Frame erzeugen.
  mPanel = new wxPanel(this, wxID_ANY);
  ePanel = new wxPanel(mPanel, wxID_ANY);  //Ein Panel im Panel

//Das Projekt-Notebook und Editor-Notebook erzeugen
  ProNBook = new wxNotebook(ePanel, wxID_ANY, wxDefaultPosition, wxSize(140, -1));
  EditNBook = new wxNotebook(ePanel, wxID_ANY);
  EditNBook->SetFont(wxFont(10, wxFONTFAMILY_MODERN,
                            wxFONTSTYLE_NORMAL, wxFONTWEIGHT_LIGHT));

  StatusNBook = new wxNotebook(mPanel, wxID_ANY, wxDefaultPosition,
      wxDefaultSize , wxNB_RIGHT);
  StatusNBook->SetFont(wxFont(10, wxFONTFAMILY_MODERN,
                            wxFONTSTYLE_NORMAL, wxFONTWEIGHT_LIGHT));

//Eine Seite zu Projekt-Notebook hinzufügen
    projektCtrl = new wxTextCtrl(ProNBook, wxID_ANY, L"Projekt-Name\n",
       wxPoint(0, 0), wxSize(200, 100), wxTE_MULTILINE);
    ProNBook->AddPage(projektCtrl, L"Projekt");
    projektCtrl->SetDefaultStyle(wxTextAttr(*wxRED));
    projektCtrl->AppendText("\tInhalt\n");
//Eine weitere Seite zu Projekt-Notebook hinzufügen
    VarsCtrl = new wxTextCtrl(ProNBook, wxID_ANY, "Global\n",
       wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
    ProNBook->AddPage(VarsCtrl, "Variablen");
    VarsCtrl->SetDefaultStyle(wxTextAttr(*wxGREEN));
    VarsCtrl->AppendText("\tInhalt\n");

// Sizer für Projekt/Editor- und Status-Notebook
  MSSizer = new wxBoxSizer(wxVERTICAL);
  MSSizer->Add(ePanel, 1, wxEXPAND);       //Sizer des Projekt/Editor-Notebook
  MSSizer->Add(StatusNBook, 0, wxEXPAND);  //Sizer des Status-Notebook
  mPanel->SetSizer(MSSizer);
  StatusCtrl = new wxTextCtrl(StatusNBook, wxID_ANY, wxT("1. Zeile\n2. Zeile\n3.
Zeile\n"),
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
  StatusNBook->AddPage(StatusCtrl, wxT("Zeilen"));
  NextCtrl = new wxTextCtrl(StatusNBook, wxID_ANY, wxT("Dies\nist\nein\nTest\n"),
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
  StatusNBook->AddPage(NextCtrl, wxT("Test"));
StatusCtrl->AppendText( wxT("C:0000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00
00  ................\n"));
StatusCtrl->SetDefaultStyle(wxTextAttr(*wxGREEN));
  StatusCtrl->AppendText(wxT("Hallo\n"));

// Sizer für das Projekt/Editor-Notebook
  PESizer = new wxBoxSizer(wxHORIZONTAL);
  PESizer->Add(ProNBook, 0, wxEXPAND);       //Sizer des Projekt-Notebook
  PESizer->Add(EditNBook, 5, wxEXPAND);
  ePanel->SetSizer(PESizer);
```

```
  menubar = new wxMenuBar;      //Menübar
  datei = new wxMenu;           //Datei-Menü
  bearbeiten = new wxMenu;      //Bearbeiten-Menü
  einstellungen = new wxMenu;   //Einstellungen-Menü
  erstellen = new wxMenu;       //Erstellen-Menü
  datei->Append(ID_MDNew, wxT("&Neu"), wxT("Neue Datei öffnen"));
  datei->Append(ID_MDOpen, wxT("&Oeffnen"), wxT("Bestehende Datei öffnen"));
  datei->Append(ID_MDSave, wxT("&Speichern"), wxT("Datei speichern"));
  datei->Append(ID_MDSaveAs, wxT("Speichern &unter"), wxT("Datei speichern
unter"));
  datei->Append(ID_MDClose, wxT("S&chließen"), wxT("Datei schließen"));
  datei->AppendSeparator();
  datei->Append(wxID_EXIT, wxT("&Beenden"));
  bearbeiten->Append(ID_MBBack, wxT("&Zurück"), wxT("Rueckgängig"));
  einstellungen->Append(ID_MEDevice, wxT("&Baustein"), wxT("CPU-Baustein
wählen"));
  erstellen->Append(ID_MEAsm, wxT("&Assemblieren"), wxT("Quelldatei
assemblieren"));
  menubar->Append(datei, wxT("&Datei"));
  menubar->Append(bearbeiten, wxT("&Bearbeiten"));
  menubar->Append(einstellungen, wxT("&Einstellungen"));
  SetMenuBar(menubar);

  Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnQuit));
  Connect(ID_MDNew, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnNew));
  Connect(ID_MDOpen, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnOpen));
  Connect(ID_MDSave, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnSave));
  Connect(ID_MDSaveAs, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnSaveAs));
  Connect(ID_MDClose, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnClose));
  Connect(ID_MBBack, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnBack));
  Connect(ID_MEDevice, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnDevice));

  wxImage::AddHandler(new wxPNGHandler);
  wxBitmap test(wxT("/home/tom/LDaten/QtProgs/MyIDE/trophy-gold.png"),
wxBITMAP_TYPE_PNG);

  toolBar = new wxToolBar(this, wxID_ANY, wxDefaultPosition, wxDefaultSize,
      wxTB_VERTICAL|wxNO_BORDER);
  toolBar->AddTool(ID_TBTest, wxT("Ein Test"), test);
  toolBar->SetToolShortHelp(ID_TBTest, wxT("Test"));
  toolBar->AddTool(ID_TBRun, wxT("Run"), Run22);
  toolBar->SetToolShortHelp(ID_TBRun, wxT("Run"));

  Bind(wxEVT_COMMAND_TOOL_CLICKED, &IDEFrame::OnNew, this, ID_TBTest);
  Bind(wxEVT_COMMAND_TOOL_CLICKED, &IDEFrame::OnOpen, this, ID_TBRun);

  toolBar->Realize ();
  SetToolBar(toolBar);
  SetIcon(wxIcon(Logo64));

  CreateStatusBar();
  SetStatusText("Eine Statuszeile");
  Centre();
```

```
  PrgConfig();      //Programm aus Config-Datei konfigurieren
}

//Destruktor
IDEFrame::~IDEFrame()
{  delete config;  }

IMPLEMENT_APP(MyIdeApp)

bool MyIdeApp::OnInit()
{ IDEFrame *myframe = new IDEFrame(wxT("MyIde"));
  myframe->Show(true);
  return true;
}

void IDEFrame::OnNew(wxCommandEvent& WXUNUSED(event))
{ EditCtrl = new wxTextCtrl(EditNBook, wxID_ANY, wxT(""),
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE, wxDefaultValidator,
wxT("unbenannt"));
  wxString FileName = "unbenannt";
  EditNBook->AddPage(EditCtrl, FileName, true, -1);
}

// Datei->Öffnen Dialog
void IDEFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
{ wxFileDialog *fdlg = new wxFileDialog(this, wxT("Datei öffnen"),
wxT(""),wxT(""),
    wxT("Alle Dateien(*.*)|*.*|C Dateien(*.c*)|*.c*|Header(*.h)|*.h|ASM
Dateien(*.a*)|*.a*"),
    wxFC_OPEN, wxDefaultPosition);
  wxTextCtrl* NeuCtrl;
  if(fdlg->ShowModal() == wxID_OK )
  { NeuCtrl = new wxTextCtrl(EditNBook, wxID_ANY, wxT(""),
      wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE, wxDefaultValidator,
wxT("NeuCtrl"));
    wxString FileName = fdlg->GetFilename();
    EditNBook->AddPage(NeuCtrl, FileName, true, -1);
    FileName = fdlg->GetDirectory() + "/" + fdlg->GetFilename();
//Pfad/Dateiname
    NeuCtrl->SetName(FileName);
    NeuCtrl->LoadFile(FileName, wxTEXT_TYPE_ANY);
    EditNBook->SetPageText(EditNBook->GetSelection(), fdlg->GetFilename());
  }
  fdlg->Destroy();
}

// Datei->Speichern Dialog
void IDEFrame::OnSave(wxCommandEvent& WXUNUSED(event))   //Event: Datei-Speichern
{ wxString mystring;
  wxTextCtrl* wxTC;     //Zeiger auf wxTextCtrl
  wxTC = (wxTextCtrl*)EditNBook->GetCurrentPage();
  mystring = wxTC->GetName();
  wxTC->SaveFile(mystring);     //In Datei speichern
}

// Datei->Speichern unter Dialog
void IDEFrame::OnSaveAs(wxCommandEvent& WXUNUSED(event)) //Event: Datei-Speichern
unter
{ uint32_t iPage;
  wxString mystring;
```

```
  wxTextCtrl* wxTC;     //Zeiger auf wxTextCtrl
  iPage = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
  wxTC = (wxTextCtrl*)EditNBook->GetCurrentPage();
  wxFileDialog *SaveDialog = new wxFileDialog(this, wxT("Datei speichern unter"),
wxT(""),wxT(""),
    wxT("Alle Dateien(*.*)|*.*|C Dateien(*.c*)|*.c*|Header(*.h)|*.h|ASM
Dateien(*.a*)|*.a*"),
    wxFD_SAVE | wxFD_OVERWRITE_PROMPT, wxDefaultPosition);
  if (SaveDialog->ShowModal() == wxID_OK) // If the user clicked "OK"
  { mystring = SaveDialog->GetDirectory() + "/" + SaveDialog->GetFilename();
//Dateiname
    wxTC->SetName(mystring);
    EditNBook->SetPageText(iPage, SaveDialog->GetFilename());
    wxTC->SaveFile(mystring); //Datei im Pfad speichern
  }
}

// Datei->Schliesen Dialog
void IDEFrame::OnClose(wxCommandEvent& WXUNUSED(event))  //Event: Datei-Schliesen
{ EditNBook->RemovePage(EditNBook->GetSelection());
}

void IDEFrame::OnBack(wxCommandEvent& WXUNUSED(event))
{ wxMessageBox(wxT("Bearbeiten - Rückgängig gewählt"),
                wxT("Meldung"), wxOK | wxICON_INFORMATION);
}

// Ereignisbehandlung
void IDEFrame::OnDevice(wxCommandEvent& WXUNUSED(event))
{ wxString mystring;
  TaktDialog* bd = new TaktDialog(wxT("Benutzer-Dialog"));
  bd->ui32Takt = ui32Clock;
  switch(ui32Clock)     //Richtigen Button aktivieren
  { case 24000000:
      bd->rb->SetValue(true);
      break;
    case 12000000:
      bd->rb1->SetValue(true);
      break;
    case 11059200:
      bd->rb2->SetValue(true);
      break;
    default:
      bd->rb3->SetValue(true);
      break;
  }
  mystring = wxString::Format(wxT("%d"), ui32Clock);
  bd->tc->SetValue(mystring);
  if(bd->ShowModal() == wxID_OK)
  { if(bd->rb->GetValue())
      ui32Clock = 24000000;
    else if(bd->rb1->GetValue())
      ui32Clock = 12000000;
    else if(bd->rb2->GetValue())
      ui32Clock = 11059200;
    else if(bd->rb3->GetValue())
      ui32Clock = wxAtoi(bd->tc->GetValue());     //Wert übernehmen
    bd->ui32Takt = ui32Clock;
//In Config-Datei übernehmen
    config->SetPath("/zielsystem");
    config->Write(wxT("Takt"), ui32Clock);
```

```
    config->Flush();
  }
  bd->Destroy();
}

void IDEFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
{ Close(true); }

//Konstruktor
TaktDialog::TaktDialog(const wxString & title)
       : wxDialog(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 230))
{ wxPanel *panel = new wxPanel(this, -1);
  wxBoxSizer *vbox = new wxBoxSizer(wxVERTICAL);
  wxBoxSizer *hbox = new wxBoxSizer(wxHORIZONTAL);
  st = new wxStaticBox(panel, -1, wxT("Takt in Hz"),
      wxPoint(5, 5), wxSize(240, 150));
  rb = new wxRadioButton(panel, -1,
      wxT("24000000"), wxPoint(15, 30), wxDefaultSize, wxRB_GROUP);
  rb1 = new wxRadioButton(panel, -1,
      wxT("12000000"), wxPoint(15, 55));
  rb2 = new wxRadioButton(panel, -1,
      wxT("11059200"), wxPoint(15, 80));
  rb3 = new wxRadioButton(panel, -1,
      wxT("Benutzer"), wxPoint(15, 105));
  tc = new wxTextCtrl(panel, -1, wxString::Format(wxT("%d"), ui32Takt),
      wxPoint(100, 103));
  wxButton *okButton = new wxButton(this, wxID_OK, wxT("Ok"),
      wxDefaultPosition, wxSize(70, 30));
  wxButton *closeButton = new wxButton(this, wxID_CANCEL, wxT("Abbruch"),
      wxDefaultPosition, wxSize(70, 30));
  hbox->Add(okButton, 1);
  hbox->Add(closeButton, 1, wxLEFT, 5);
  vbox->Add(panel, 1);
  vbox->Add(hbox, 0, wxALIGN_CENTER | wxTOP | wxBOTTOM, 10);
  SetSizer(vbox);
  Centre();
}

//Programm aus Config-Datei konfigurieren
bool IDEFrame::PrgConfig(void)
{ wxString cfgFile = wxStandardPaths::Get().GetExecutablePath() + ".cfg";
  config = new wxFileConfig( "", "", cfgFile);
  if(!wxFile::Exists(cfgFile))      //Config existiert?
  { config->SetPath("/zielsystem");     //Falls nein, Defaultwerte setzen
    config->Write(wxT("Takt"), 12000000);
//Nur zum testen
    config->SetPath("/dateien");
    config->Write(wxT("datei"), wxT("unbenannt.txt"));
//Test Ende
    config->Flush(true);
  }
//Programm aus Config-Datei initialisieren
  config->SetPath("/zielsystem");
  SetClock(config->Read(wxT("Takt"), true));
  return true;
}
```

4 Erweiterungen

Hier finden sich nützliche Erweiterungen der Basisapplikation.

4.1 Menü Datei

Das Datei-Menü besteht bislang nur aus den Items „Neu, Öffnen und Beenden“, jetzt wird es um „Speichern“, „Speichern unter“ und „Schließen“ ergänzt.

Header:

```
enum
{ ID_MDNew      = 101,    //Menü-Datei Neu
  ID_MDOpen     = 102,    //Menü-Datei öffnen
ID_MDSave     = 103,    //Menü-Datei speichern
ID_MDSaveAs   = 104,    //Menü-Datei speichern unter
ID_MDClose    = 105,    //Menü-Datei schließen

  void OnNew(wxCommandEvent& event);    //Event: Datei-Neu
  void OnOpen(wxCommandEvent& event);   //Event: Datei-Öffnen
void OnSave(wxCommandEvent& event);   //Event: Datei-Speichern
void OnSaveAs(wxCommandEvent& event); //Event: Datei-Speichern unter
void OnClose(wxCommandEvent& event);  //Event: Datei-Schließen
```

Main:
Dem Editor- und Status-Notebook wird je ein Font in Proportionalschrift (wxFONTFAMILY_MODERN) gegeben, damit die einzelnen Zeichen gleiche Breite aufweisen. Das verbessert die Lesbarkeit von Programmen und Tabellen.

```
//Das Projekt-Notebook und Editor-Notebook erzeugen
  ProNBook = new wxNotebook(ePanel, wxID_ANY, wxDefaultPosition, wxSize(140, -1));
  EditNBook = new wxNotebook(ePanel, wxID_ANY);
EditNBook->SetFont(wxFont(10, wxFONTFAMILY_MODERN,
                          wxFONTSTYLE_NORMAL, wxFONTWEIGHT_LIGHT));

  StatusNBook = new wxNotebook(mPanel, wxID_ANY, wxDefaultPosition,
      wxDefaultSize , wxNB_RIGHT);
StatusNBook->SetFont(wxFont(10, wxFONTFAMILY_MODERN,
                          wxFONTSTYLE_NORMAL, wxFONTWEIGHT_LIGHT));
```

Es wird zu Beginn keine Seite mehr zum Editor-Notebook hinzugefügt.

```
//Eine Seite zu Editor-Notebook hinzufügen
//  EditCtrl = new wxTextCtrl(EditNBook, wxID_ANY, L"/* Nur ein Test */\n\n",
//  wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE, wxDefaultValidator,
wxT("EditCtrl"));
//  EditNBook->AddPage(EditCtrl, wxT("Test.c"));
//  EditCtrl->AppendText(L"#include <stdio.h>\n\nvoid main (void)\n");
```

Die Ereignisbehandlung wird verbunden.

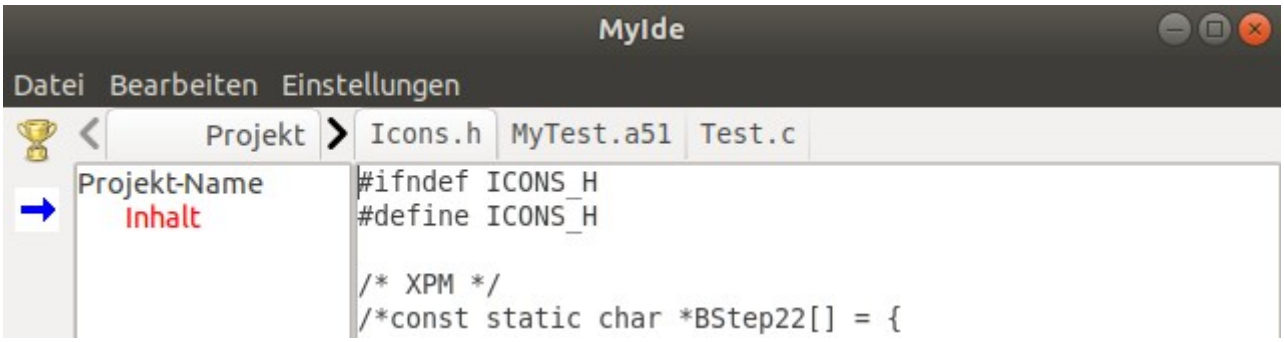
```
  Connect(ID_MDOpen, wxEVT_COMMAND_MENU_SELECTED,
          wxCommandEventHandler(IDEFrame::OnOpen));
Connect(ID_MDSave, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnSave));
Connect(ID_MDSaveAs, wxEVT_COMMAND_MENU_SELECTED,
```

```
        wxCommandEventHandler(IDEFrame::OnSaveAs));
Connect(ID_MDClose, wxEVT_COMMAND_MENU_SELECTED,
        wxCommandEventHandler(IDEFrame::OnClose));
```

Die Ereignisbehandlung wird um die benötigten Methoden erweitert.

```
// Datei->Speichern Dialog
void IDEFrame::OnSave(wxCommandEvent& WXUNUSED(event))   //Event: Datei-Speichern
{ wxString mystring;
  wxTextCtrl* wxTC;     //Zeiger auf wxTextCtrl
  wxTC = (wxTextCtrl*)EditNBook->GetCurrentPage();
  mystring = wxTC->GetName();
  wxTC->SaveFile(mystring);     //In Datei speichern
}

// Datei->Speichern unter Dialog
void IDEFrame::OnSaveAs(wxCommandEvent& WXUNUSED(event)) //Event: Datei-Speichern
unter
{ uint32_t iPage;
  wxString mystring;
  wxTextCtrl* wxTC;     //Zeiger auf wxTextCtrl
  iPage = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
  wxTC = (wxTextCtrl*)EditNBook->GetCurrentPage();
  wxFileDialog *SaveDialog = new wxFileDialog(this, wxT("Datei speichern unter"),
wxT(""),wxT(""),
    wxT("Alle Dateien(*.*)|*.*|C Dateien(*.c*)|*.c*|Header(*.h)|*.h|ASM
Dateien(*.a*)|*.a*"),
    wxFD_SAVE | wxFD_OVERWRITE_PROMPT, wxDefaultPosition);
  if (SaveDialog->ShowModal() == wxID_OK) // If the user clicked "OK"
  { mystring = SaveDialog->GetDirectory() + "/" + SaveDialog->GetFilename();
    wxTC->SetName(mystring);  //Pfad/Dateiname
    EditNBook->SetPageText(iPage, SaveDialog->GetFilename());
    wxTC->SaveFile(mystring); //Datei im Pfad speichern
  }
}

// Datei->Schliesen Dialog
void IDEFrame::OnClose(wxCommandEvent& WXUNUSED(event))
{ EditNBook->RemovePage(EditNBook->GetSelection());
}
```

Jetzt können im Editor-Notebook mehrere Dateien aus unterschiedlichen Verzeichnissen gleichzeitig dargestellt und einzeln bearbeitet werden.



4.2 Config-Datei

Werden die geöffneten Dateien im Editor-Notebook mit in der Config-Datei verwaltet, so können beim neuen Programmstart die zuletzt geöffneten Dateien automatisch wieder geöffnet werden.

Header:

```
  wxFileConfig *config;     //Config-Datei
  wxString GetCFileName();     //Name für Datei in .cfg erzeugen
  bool PrgConfig(void);        //Programm aus Config-Datei konfigurieren
```

Main:

```
void IDEFrame::OnNew(wxCommandEvent& WXUNUSED(event))
{ config->SetPath("/dateien");
//  int iCount = config->GetNumberOfEntries(false);
  wxString Eintrag = wxT("unbenannt.c");
  wxMessageBox(Eintrag, wxT("Meldung"), wxOK | wxICON_INFORMATION);
  wxString FileName = wxStandardPaths::Get().GetExecutablePath();
  FileName = wxFileName(FileName).GetPath() + wxT("/unbenannt.c");
config->Write(GetCFileName(), FileName);
config->Flush(true);

  EditCtrl = new wxTextCtrl(EditNBook, wxID_ANY, wxT(""),
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE, wxDefaultValidator,
wxT(""));
  EditNBook->AddPage(EditCtrl, wxT("unbenannt.c"), true, -1);
}

// Datei->Öffnen Dialog
void IDEFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
{ wxString FileName;
……
//Pfad/Dateiname
    config->Write(Eintrag, FileName);
    config->Flush(true);
    NeuCtrl = new wxTextCtrl(EditNBook, wxID_ANY, wxT(""),
      wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE, wxDefaultValidator,
wxT(""));
……..
  }
  fdlg->Destroy();
}

// Datei->Schliesen Dialog
void IDEFrame::OnClose(wxCommandEvent& WXUNUSED(event))  //Event: Datei-Schliesen
{ uint32_t iPage;
  wxChar wCz, wCq;
  wxString fValue, eQuel, eZiel = wxT("AFile");
//Eintrag in Config-Datei löschen
  iPage = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
  eZiel.SetChar(0, 'A'+iPage);
//  wxMessageBox(eZiel, wxT("Meldung"), wxOK | wxICON_INFORMATION);
  config->SetPath("/dateien");
  config->DeleteEntry(eZiel, true);
  config->Flush(true);
//Aus dem Notebook entfernen
  EditNBook->RemovePage(iPage);

  eZiel = wxT("AFile");
```

```
  eQuel = wxT("AFile");
  for(iPage=config->GetNumberOfEntries(); iPage; iPage--)
  { fValue = config->Read(eQuel);
    if(!fValue)
    { wCq = eQuel.GetChar(0);
      eQuel.SetChar(0, ++wCq);
      fValue = config->Read(eQuel);
      config->Write(eZiel, fValue);
      config->DeleteEntry(eQuel, true);
      wCz = eZiel.GetChar(0);
      eZiel.SetChar(0, ++wCz);
      continue;
    }
    config->Write(eZiel, fValue);
    wCz = eZiel.GetChar(0);
    eZiel.SetChar(0, ++wCz);
    eQuel = eZiel;
  }
  config->Flush(true);
}

// Ereignisbehandlung
void IDEFrame::OnDevice(wxCommandEvent& WXUNUSED(event))
{ wxString mystring;
.....
  bd->ui32Takt = ui32Clock;
//In Config-Datei übernehmen
    config->SetPath("/zielsystem");
    config->Write(wxT("Takt"), ui32Clock);
    config->Flush();
  }
  bd->Destroy();
}

//Filename für Datei in .cfg erzeugen. Damit können die Dateien
//in der richtigen Reihenfolge wieder geöffnet werden, da config
//alphabetisch vorgeht.
wxString IDEFrame::GetCFileName(void)
{ long lVar;
  bool boCnt;
  wxChar xCH = 'A';
  wxString fName = wxT("AFile");
  config->SetPath("/dateien");
  boCnt = config->GetFirstEntry(fName, lVar);
  if(boCnt)
  { do{
      fName.SetChar(0, ++xCH);
    }while(config->GetNextEntry(fName, lVar));
  }
  return fName;
}

//Programm aus Config-Datei konfigurieren
bool IDEFrame::PrgConfig(void)
{ long lVar;
  uint32_t iCo;
  wxInt32 iPages;
  wxString fPath, fName, fExt;
  wxChar xCH = 'A';
  wxString fFile = wxStandardPaths::Get().GetExecutablePath() + ".cfg";
  config = new wxFileConfig( "", "", fFile);
```

```
  config->SetPath("/zielsystem");
  if(!wxFile::Exists(fFile))      //Config existiert?
  { config->Write(wxT("Takt"), 12000000);
    config->Flush(true);
  }
  SetClock(config->Read(wxT("Takt"), true));

//Editortabs wieder einrichten
  config->SetPath("/dateien");
  iPages = config->GetNumberOfEntries();
  std::vector<wxTextCtrl*> wTCPtr[iPages];
  wxString sEntry = wxT("AFile");
  if(config->GetFirstEntry(fName, lVar))
  { for(iCo=0; iPages; iPages--, iCo++)
    { wTCPtr[iCo].push_back(new wxTextCtrl(EditNBook, wxID_ANY));
      fFile = config->Read(sEntry);
      sEntry.SetChar(0, ++xCH);
      wxFileName::SplitPath(fFile, &fPath, &fName, &fExt);
      fName += "." + fExt;
      wTCPtr[iCo].at(0) = new wxTextCtrl(EditNBook, wxID_ANY, wxT(""),
        wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE, wxDefaultValidator,
wxT(""));
      EditCtrl = wTCPtr[iCo].at(0);
      EditNBook->AddPage(EditCtrl, fName, true, -1);
      EditCtrl->LoadFile(fFile, wxTEXT_TYPE_ANY);
    wTCPtr[iCo].pop_back();
      if(!config->GetNextEntry(fName, lVar))
        break;
    }
  }
  return true;
}
```

4.2.1 Zuletzt aktive Datei reaktivieren

Nachdem beim beenden des Programms auch die gerade aktive Datei gespeichert wird, läßt sie sich beim Neustart wieder aktivieren. Leider funktioniert das anzeigen des zuletzt gewählten Textbereiches (um die Cursorposition) nicht, aber ist bereits vorbereitet.

Im Header wird Platz für bis zu 20 Dateien reserviert.

```
  std::vector<wxTextCtrl*> wTCPtr[20];
```

Main:

```
void IDEFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
{ long pos, sel;
//Cursorposition der aktuellen Datei ermitteln
  sel = EditNBook->GetSelection();
  pos = wTCPtr[sel].at(0)->GetInsertionPoint ();
  if(pos != wTCPtr[sel].at(0)->GetLastPosition ())
    wTCPtr[sel].at(0)->GetRange(pos, pos + 1);
//Aktiven Editor und Cursorposition speichern
  config->SetPath("/editor");
  config->Write(wxT("CurrentTab"), EditNBook->GetSelection());    //Aktiver Editor
  config->Write(wxT("CurrentPos"), pos);   //Zeile/Spalte im Editor
  Close(true);
}
```

```
//Programm aus Config-Datei konfigurieren
bool IDEFrame::PrgConfig(void)
{ long lVar;
  uint32_t iCo;


.......
//Aktiven Editor wieder herstellen
  config->SetPath("/editor");
  iCo = atoi(config->Read(wxT("CurrentTab")));
  EditNBook->SetSelection(iCo);    //Aktiver Editor

// ShowPosition funktioniert nicht, etwas für später!
//Cursorposition im Editor wieder herstellen
//  long pos = atol(config->Read(wxT("CurrentPos")));
//  wTCPtr[iCo].at(0)->SetInsertionPoint(pos);  //Curserposition
//  wTCPtr[iCo].at(0)->ShowPosition(pos);       //Textbereich anzeigen
  return true;
}
```

4.3 Editor umstellen

Hier werden die unzureichend funktionierenden wxTextCtrl-Komponenten durch wxStyledTextCtr ersetzt. Dazu sind umfangreiche Änderungen nötig. Ein Vektor von Zeigern erlaubt jetzt die dynamische Verwaltung vieler offener Dateien im Editor Notebook.

Include:

```
      #include <wx/fileconf.h>
#include <wx/stc/stc.h>
#include <vector>

   bool PrgConfig(void);     //Programm aus Config-Datei konfigurieren
std::vector<wxStyledTextCtrl*> wSTCPtr;  //Zeiger auf Editor-Tabs
```

Main:

```
void IDEFrame::OnNew(wxCommandEvent& WXUNUSED(event))
{ int iCo;
  config->SetPath("/dateien");
  iCo = config->GetNumberOfEntries(false);
  wxString FileName = wxStandardPaths::Get().GetExecutablePath();
  FileName = wxFileName(FileName).GetPath() + wxT("/unbenannt.c");
  wSTCPtr.push_back(new wxStyledTextCtrl(EditNBook, wxID_ANY));
  EditNBook->AddPage(wSTCPtr[iCo], wxT("unbenannt.c"), true, -1);
  config->SetPath("/dateien");
  config->Write(GetCFileName(), FileName);
  config->Flush(true);
}

// Datei->Öffnen Dialog
void IDEFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
{ int iCo;
  wxString FileName;
  wxFileDialog *fdlg = new wxFileDialog(this, wxT("Datei öffnen"),
wxT(""),wxT(""),
    wxT("Alle Dateien(*.*)|*.*|C Dateien(*.c*)|*.c*|Header(*.h)|*.h|ASM
Dateien(*.a*)|*.a*"),
    wxFC_OPEN, wxDefaultPosition);
  if(fdlg->ShowModal() == wxID_OK)
  { config->SetPath("/dateien");
    iCo = config->GetNumberOfEntries(false);
    wxString Eintrag = GetCFileName();
    FileName = fdlg->GetDirectory() + "/" + fdlg->GetFilename();
//Pfad/Dateiname
    config->Write(Eintrag, FileName);
    config->Flush(true);
    FileName = fdlg->GetFilename();
    wSTCPtr.push_back(new wxStyledTextCtrl(EditNBook, wxID_ANY));
    EditNBook->AddPage(wSTCPtr[iCo], FileName, true, -1);
    FileName = fdlg->GetDirectory() + "/" + fdlg->GetFilename();
//Pfad/Dateiname
    wSTCPtr[iCo]->SetName(FileName);
    wSTCPtr[iCo]->LoadFile(FileName, wxTEXT_TYPE_ANY);
    EditNBook->SetPageText(EditNBook->GetSelection(), fdlg->GetFilename());
  }
  fdlg->Destroy();
}
```

```
// Datei->Speichern Dialog
void IDEFrame::OnSave(wxCommandEvent& WXUNUSED(event))   //Event: Datei-Speichern
{ int32_t iCo;
  wxChar xCH = 'A';
  wxString mystring;
  iCo = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
  xCH += iCo;
  config->SetPath("/dateien");
  mystring = wxT("AFile");
  mystring.SetChar(0, xCH);
  mystring = config->Read(mystring);
  wSTCPtr[iCo]->SaveFile(mystring);     //In Datei speichern
}

// Datei->Speichern unter Dialog
void IDEFrame::OnSaveAs(wxCommandEvent& WXUNUSED(event)) //Event: Datei-Speichern
unter
{ uint32_t iCo = 0;
  wxString mystring;
  wxString cName = wxT("AFile");
  wxChar xCH = 'A';

  wxFileDialog *SDlg = new wxFileDialog(this, wxT("Datei speichern unter"),
wxT(""),wxT(""),
    wxT("Alle Dateien(*.*)|*.*|C Dateien(*.c*)|*.c*|Header(*.h)|*.h|ASM
Dateien(*.a*)|*.a*"),
    wxFD_SAVE | wxFD_OVERWRITE_PROMPT, wxDefaultPosition);
  if (SDlg->ShowModal() == wxID_OK) //"OK" gewählt
  { mystring = SDlg->GetDirectory() + "/" + SDlg->GetFilename();     //Dateiname
    iCo = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
    EditNBook->SetPageText(iCo, SDlg->GetFilename());
    xCH += iCo;
    cName.SetChar(0, xCH);
    mystring = SDlg->GetDirectory() + "/" + SDlg->GetFilename();
//Pfad/Dateiname
    wSTCPtr[iCo]->SetName(mystring);
    wSTCPtr[iCo]->SaveFile(mystring);     //In Datei speichern
//Eintrag in Config-Datei ändern
    config->SetPath("/dateien");
    config->Write(cName, mystring);
    config->Flush(true);
  }
  SDlg->Destroy();
}

// Datei->Schliesen Dialog
void IDEFrame::OnClose(wxCommandEvent& WXUNUSED(event))  //Event: Datei-Schliesen
{ uint32_t iPage;
  wxChar wCz, wCq;

  wxString fValue, eQuel, eZiel = wxT("AFile");
//Eintrag in Config-Datei löschen
  iPage = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
  eZiel.SetChar(0, 'A'+iPage);
  config->SetPath("/dateien");
  config->DeleteEntry(eZiel, true);
  config->Flush(true);
//Aus dem Notebook entfernen
  EditNBook->RemovePage(iPage);
//Einträge in config-dateien neu ordnen
  eZiel = wxT("AFile");
```

```
  eQuel = wxT("AFile");
  for(iPage=config->GetNumberOfEntries(); iPage; iPage--)
  { fValue = config->Read(eQuel);
    if(!fValue)
    { wCq = eQuel.GetChar(0);
      eQuel.SetChar(0, ++wCq);
      fValue = config->Read(eQuel);
      config->Write(eZiel, fValue);
      config->DeleteEntry(eQuel, true);
      wCz = eZiel.GetChar(0);
      eZiel.SetChar(0, ++wCz);
      continue;
    }
    config->Write(eZiel, fValue);
    wCz = eZiel.GetChar(0);
    eZiel.SetChar(0, ++wCz);
    eQuel = eZiel;
  }
  config->Flush(true);
}

//Programm aus Config-Datei konfigurieren
bool IDEFrame::PrgConfig(void)
{ long lVar;
  uint32_t iCo;
  wxInt32 iPages;
  wxString fPath, fName, fExt;
  wxChar xCH = 'A';
  wxString fFile = wxStandardPaths::Get().GetExecutablePath() + ".cfg";
//Falls noch keine .cfg-Datei existiert, eine anlegen
  config = new wxFileConfig( "", "", fFile);
  config->SetPath("/zielsystem");
  if(!wxFile::Exists(fFile))      //Config existiert?
  { config->Write(wxT("Takt"), 12000000);
    config->SetPath("/dateien");
    config->SetPath("/editor");
    config->Write(wxT("CurrentTab"), 0);    //Aktiver Editor
    config->Write(wxT("CurrentPos"), 0);   //Zeile im Editor
    config->Flush(true);
  }
  SetClock(config->Read(wxT("Takt"), true));
//Editortabs wieder einrichten
  config->SetPath("/dateien");
  wxString sEntry = wxT("AFile");
  if(config->GetFirstEntry(fName, lVar))
  { for(iCo=0; iPages; iPages--, iCo++)
    { wSTCPtr.push_back(new wxStyledTextCtrl(EditNBook, wxID_ANY));
      fFile = config->Read(sEntry);
      sEntry.SetChar(0, ++xCH);
      wxFileName::SplitPath(fFile, &fPath, &fName, &fExt);
      fName += "." + fExt;
      wSTCPtr[iCo] = new wxStyledTextCtrl(EditNBook, wxID_ANY, wxDefaultPosition,
        wxDefaultSize, wxTE_MULTILINE | wxTE_PROCESS_TAB, wxT(""));
      EditNBook->AddPage(wSTCPtr[iCo], fName, true, -1);
      wSTCPtr[iCo]->LoadFile(fFile, wxTEXT_TYPE_ANY);
      if(!config->GetNextEntry(fName, lVar))
        break;
    }
  }
//Aktiven Editor wieder herstellen
  config->SetPath("/editor");
```

```
  iCo = atoi(config->Read(wxT("CurrentTab")));
  EditNBook->SetSelection(iCo);    //Aktiver Editor
//Cursorposition im Editor wieder herstellen
  long pos = atol(config->Read(wxT("CurrentPos")));
  wSTCPtr[iCo]->ShowPosition(pos);       //Textbereich anzeigen
  return true;
}
```

Nun funktioniert alles wie gewollt, auch das positionieren des Cursor und anzeigen des Textbereichs im aktiven Editor-Tab.

4.4 Geänderte Dateien speichern

Um keine Änderung in einer, beim schließen der Datei oder dem beenden des Programms zu verlieren, kann man das Flag „IsModified“ des wxStyledTextCtrl prüfen und falls nötig die Datei abspeichern. Im Beispiel wird in einem Dialog nachgefragt, ob gespeichert wird (Yes). Anderenfalls (No) wird die Datei nicht gespeichert und damit Änderungen seit dem letzten speichern verworfen.

Main:

```
// Datei->Schliesen Dialog
void IDEFrame::OnClose(wxCommandEvent& WXUNUSED(event))  //Event: Datei-Schliesen
{ uint32_t iPage;
  wxChar wCz, wCq;
  wxString cFile, eQuel, eZiel = wxT("AFile");
  wxFileName fValue;
  iPage = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
  config->SetPath("/dateien");
  eZiel = wxT("AFile");
  eZiel.SetChar(0, 'A' + iPage);
//Modifizierte Dateien speichern?
  if(wSTCPtr[iPage]->IsModified())      //Datei geändert?
  { fValue = config->Read(eZiel);       //Pfad und Name
    cFile = fValue.GetFullName();
    if(wxMessageBox(cFile, wxT("Geänderte Datei speichern?"), wxYES_NO
              | wxICON_QUESTION) == wxYES)
      wSTCPtr[iPage]->SaveFile(fValue.GetFullPath());     //Datei speichern
  }
//Eintrag in Config-Datei löschen
    config->DeleteEntry(eZiel, true);
    config->Flush(true);
//Aus dem Notebook entfernen
    EditNBook->RemovePage(iPage);
//Einträge in config-dateien neu ordnen
    eZiel = wxT("AFile");
    eQuel = wxT("AFile");
    for(iPage=config->GetNumberOfEntries(); iPage; iPage--)
    { cFile = config->Read(eQuel);
      if(!cFile)
      { wCq = eQuel.GetChar(0);
        eQuel.SetChar(0, ++wCq);
        cFile = config->Read(eQuel);
        config->Write(eZiel, cFile);
        config->DeleteEntry(eQuel, true);
        wCz = eZiel.GetChar(0);
        eZiel.SetChar(0, ++wCz);
        continue;
      }
      config->Write(eZiel, cFile);
      wCz = eZiel.GetChar(0);
      eZiel.SetChar(0, ++wCz);
      eQuel = eZiel;
    }
  config->Flush(true);
}

void IDEFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
{ long pos;
  int iVar, sel = 0;
  wxString cFile, eZiel = wxT("AFile");
```

```
  wxFileName fValue;
//Modifizierte Dateien speichern?
  iVar = wSTCPtr.size();      //Anzahl offener Dateien
  for(sel = 0; sel < iVar; sel++)
  { if(wSTCPtr[sel]->IsModified())      //Datei geändert?
    { config->SetPath("/dateien");
      eZiel.SetChar(0, 'A' + sel);
      fValue = config->Read(eZiel);     //Pfad und Name
      cFile = fValue.GetFullName();
      if(wxMessageBox(cFile, wxT("Geänderte Datei speichern?"), wxYES_NO
                | wxICON_QUESTION) == wxYES)
       wSTCPtr[sel]->SaveFile(fValue.GetFullPath());     //Datei speichern
    }
  }

//Cursorposition in der aktuellen Datei ermitteln
  sel = EditNBook->GetSelection();      //Nummer der aktiven Datei
  pos = wSTCPtr[sel]->GetInsertionPoint ();
  if(pos != wSTCPtr[sel]->GetLastPosition ())
    wSTCPtr[sel]->GetRange(pos, pos + 1);
//Aktiven Editor und Cursorposition speichern
  config->SetPath("/editor");
  config->Write(wxT("CurrentTab"), EditNBook->GetSelection());    //Aktiver Editor
  config->Write(wxT("CurrentPos"), pos);   //Zeile/Spalte im Editor
  Close(true);
}
```

4.5 Syntaxhighlight

Erleichtert das programmieren ein wenig, da man an der Farbe erkennt, ob ein Element erkannt wird oder nicht. Ich richte hier zwei verschiedene Arten ein, für Assembler- und C-Programme. Der Unterschied wird durch den Dateiextend erkannt.

Include:

```
        std::vector<wxStyledTextCtrl*> wSTCPtr;  //Zeiger auf Editor-Tabs
bool SetCStyle(int);         //Style für C-Programme
bool SetAsmStyle(int);       //Style für ASM-Programme
bool SetStyle(int);          //Editor Style wieder herstellen
```

Main:

```
// Datei->Öffnen Dialog
void IDEFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
{ int iCo;


.....
  fdlg->Destroy();
SetStyle(iCo);
}

// Datei->Speichern unter Dialog
void IDEFrame::OnSaveAs(wxCommandEvent& WXUNUSED(event))

{ uint32_t iCo = 0;


.......

  fdlg->Destroy();
SetStyle(iCo);
}

//Programm aus Config-Datei konfigurieren
bool IDEFrame::PrgConfig(void)
{ long lVar;


...........

    EditNBook->AddPage(wSTCPtr[iCo], fName, true, -1);
      wSTCPtr[iCo]->LoadFile(fFile, wxTEXT_TYPE_ANY);
SetStyle(iCo);    //Editor Style wieder herstellen
      if(!config->GetNextEntry(fName, lVar))
        break;
    }
  }

//Style für C-Programme
bool IDEFrame::SetCStyle(int iCo)
{ wSTCPtr[iCo]->StyleClearAll();
// Den Lexer für C++ setzen
  wSTCPtr[iCo]->SetLexer(wxSTC_LEX_CPP);
// Farben für C Syntax-highlight
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_DEFAULT, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_COMMENT, gruen);
```

```
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_COMMENTLINE, gruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_COMMENTDOC, gruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_NUMBER, rot);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_WORD, dunkelblau);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_STRING, dunkelgruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_CHARACTER, dunkelgruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_UUID, gelb);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_PREPROCESSOR, dunkelgruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_OPERATOR, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_IDENTIFIER, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_STRINGEOL, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_VERBATIM, violet);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_REGEX, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_COMMENTLINEDOC, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_WORD2, braun);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_COMMENTDOCKEYWORD, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_COMMENTDOCKEYWORDERROR, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_GLOBALCLASS, braun);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_STRINGRAW, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_TRIPLEVERBATIM, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_HASHQUOTEDSTRING, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_PREPROCESSORCOMMENT, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_PREPROCESSORCOMMENTDOC, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_USERLITERAL, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_TASKMARKER, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_C_ESCAPESEQUENCE, gold);
//wSTCPtr[iCo]->StyleSetBackground(wxSTC_C_ESCAPESEQUENCE, gelb);

  wSTCPtr[iCo]->SetWordChars(
wxT("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMONPQRSTUVWXYZ_\\-:") );

//wxSTC_C_WORD
  wSTCPtr[iCo]->SetKeyWords(0, wxT("auto break case continue default do \
        else enum extern for goto if inline register restrict return \
        CHAR char const double float int long short signed static union unsigned \
        rsizeof struct switch void volatile while"));
//wxSTC_C_WORD2
  wSTCPtr[iCo]->SetKeyWords(1, wxT("P0 p0 SP sp DPL dpl DPH dph PCON pcon \
        TCON tcon TMOD tmod TL0 tl0 TL1 tl1 TH0 th0 TH1 th1 P1 p1 SCON scon \
        SBUF sbuf P2 p2 IE ie P3 p3 IP ip T2CON t2con T2MOD t2mod RCAP2L rcap2l \
        RCAP2H rcap2h TL2 tl2 TH2 th2 PSW psw ACC acc REGB regb"));

  wSTCPtr[iCo]->SetMarginType(1, wxSTC_MARGIN_NUMBER); // Zeilennummer anzeigen
  wSTCPtr[iCo]->SetMarginWidth(1, 40);      //Zweiter Parameter: Randbreite
  wSTCPtr[iCo]->SetMarginLeft(1);           // Linker Rand

  for (int nr = 0; nr < wxSTC_STYLE_LASTPREDEFINED; nr++)
  { wxFont font (10, wxFontFamily::wxFONTFAMILY_MODERN,
    wxFontStyle::wxFONTSTYLE_NORMAL, wxFontWeight::wxFONTWEIGHT_NORMAL);
    wSTCPtr[iCo]->StyleSetFont (nr, font);
  }
  return true;
}

//Style für ASM-Programme
bool IDEFrame::SetAsmStyle(int iCo)
{ wSTCPtr[iCo]->StyleClearAll();
// Den Lexer für ASM setzen
  wSTCPtr[iCo]->SetLexer(wxSTC_LEX_ASM);
// Farben für Assembler Syntax-highlight
```

```
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_DEFAULT, rot);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_COMMENT, gruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_NUMBER, rot);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_STRING, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_OPERATOR, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_IDENTIFIER, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_CPUINSTRUCTION, schwarz);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_MATHINSTRUCTION, dunkelgruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_REGISTER, dunkelgruen);
//wSTCPtr[iCo]->StyleSetBackground(wxSTC_ASM_REGISTER, gelb);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_DIRECTIVE, gelb);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_DIRECTIVEOPERAND, gold);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_COMMENTBLOCK, dunkelgruen);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_CHARACTER, violet);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_EXTINSTRUCTION, dunkelblau);
  wSTCPtr[iCo]->StyleSetForeground(wxSTC_ASM_COMMENTDIRECTIVE, hellblau);

  wSTCPtr[iCo]->SetWordChars(
wxT("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMONPQRSTUVWXYZ_\\-:") );

//wxSTC_ASM_CPUINSTRUCTION
  wSTCPtr[iCo]->SetKeyWords(0, wxT("acall add addc ajmp anl cjne clr cpl da dec \
      div djnz inc jb jbc jc jmp jnb jnc jnz jz lcall ljmp mov movc movx mul nop \
      orl pop push ret reti rl rlc rr rrc setb sjmp subb swap xch xchd xrl"));

//wxSTC_ASM_REGISTER
  wSTCPtr[iCo]->SetKeyWords(2, wxT("p0 sp dpl dph pcon tcon dpl dph tcon \
        tmod tl0 tl1 th0 th1 p1 scon sbuf p2 ie p3 ip t2con t2mod rcap2l rcap2h \
        tl2 th2 psw acc regb"));
//wxSTC_ASM_EXTINSTRUCTION
  wSTCPtr[iCo]->SetKeyWords(5, wxT("equ org end @a ab dptr pc \
        r0 r1 r2 r3 r4 r5 r6 r7"));

  wSTCPtr[iCo]->SetMarginType(1, wxSTC_MARGIN_NUMBER); // Zeilennummer anzeigen
  wSTCPtr[iCo]->SetMarginWidth(1, 40);      //Zweiter Parameter: Randbreite
  wSTCPtr[iCo]->SetMarginLeft(1);           //Linker Rand
  for (int nr = 0; nr < wxSTC_STYLE_LASTPREDEFINED; nr++)
  { wxFont font (10, wxFontFamily::wxFONTFAMILY_MODERN,
    wxFontStyle::wxFONTSTYLE_NORMAL, wxFontWeight::wxFONTWEIGHT_NORMAL);
    wSTCPtr[iCo]->StyleSetFont (nr, font);
  }
  return true;
}

//Editor Style wieder herstellen
bool IDEFrame::SetStyle(int iCo)
{ wxString fExt = wxT("AFile");
  wxFileName fValue;
  config->SetPath("/dateien");
  fExt.SetChar(0, 'A' + iCo);
  fValue = config->Read(fExt);     //Pfad und Name
  fExt = fValue.GetExt();
  if(fExt == "a51" || fExt == "A51" || fExt == "asm" || fExt == "ASM")
  { SetAsmStyle(iCo);
    return true; }
  else
    if(fExt == "c51" || fExt == "C51" || fExt == "c" || fExt == "C")
    { SetCStyle(iCo);
      return true; }
  wSTCPtr[iCo]->SetMarginLeft(0);           //Linker Rand
```

```
  for (int nr = 0; nr < wxSTC_STYLE_LASTPREDEFINED; nr++)
  { wxFont font (10, wxFontFamily::wxFONTFAMILY_MODERN,
    wxFontStyle::wxFONTSTYLE_NORMAL, wxFontWeight::wxFONTWEIGHT_NORMAL);
    wSTCPtr[iCo]->StyleSetFont (nr, font);
  }
  return true;
}
```

4.6 Tastaturabfrage

Um das Programm nicht nur mit der Maus, sondern auch über die Tastatur zu bedienen, braucht es das Bedienen von Tastatur-Events.

Include:

```
#include <wx/event.h>

void OnKeyDown(wxKeyEvent& event);
```

Main:

```
Bind(wxEVT_CHAR_HOOK, &IDEFrame::OnKeyDown, this);

//Tastaturereignisse erfassen
void IDEFrame::OnKeyDown(wxKeyEvent& aEvent)
{ wxChar wxC = aEvent.GetKeyCode();
  if(aEvent.ControlDown())  //Strg-Taste
  { switch(wxC) {
      case 'K':
          wxMessageBox(wxT("Taste 'Strg+K' gedrückt"), wxT("Meldung"),
                       wxOK| wxICON_INFORMATION);
          break;
      case WXK_F5:
          wxMessageBox(wxT("Taste 'Strg+F5' gedrückt"), wxT("Meldung"),
                       wxOK| wxICON_INFORMATION);
          break;
      default:
          break;
      }
  } else {
    switch (wxC) {
      case WXK_F1:
          wxMessageBox(wxT("Taste 'F1' gedrückt"), wxT("Meldung"), wxOK|
wxICON_INFORMATION);
          break;
      default:
          break;
      }
 }
 aEvent.Skip(true);
}
```

4.7 Klammerpaare finden

Es erleichtert das programmieren in C, wenn zusammengehörige Klammern automatisch gefunden und markiert werden können. WxStyledTextCtrl bietet diese Funktion komfortabel vorbereitet.

Include:

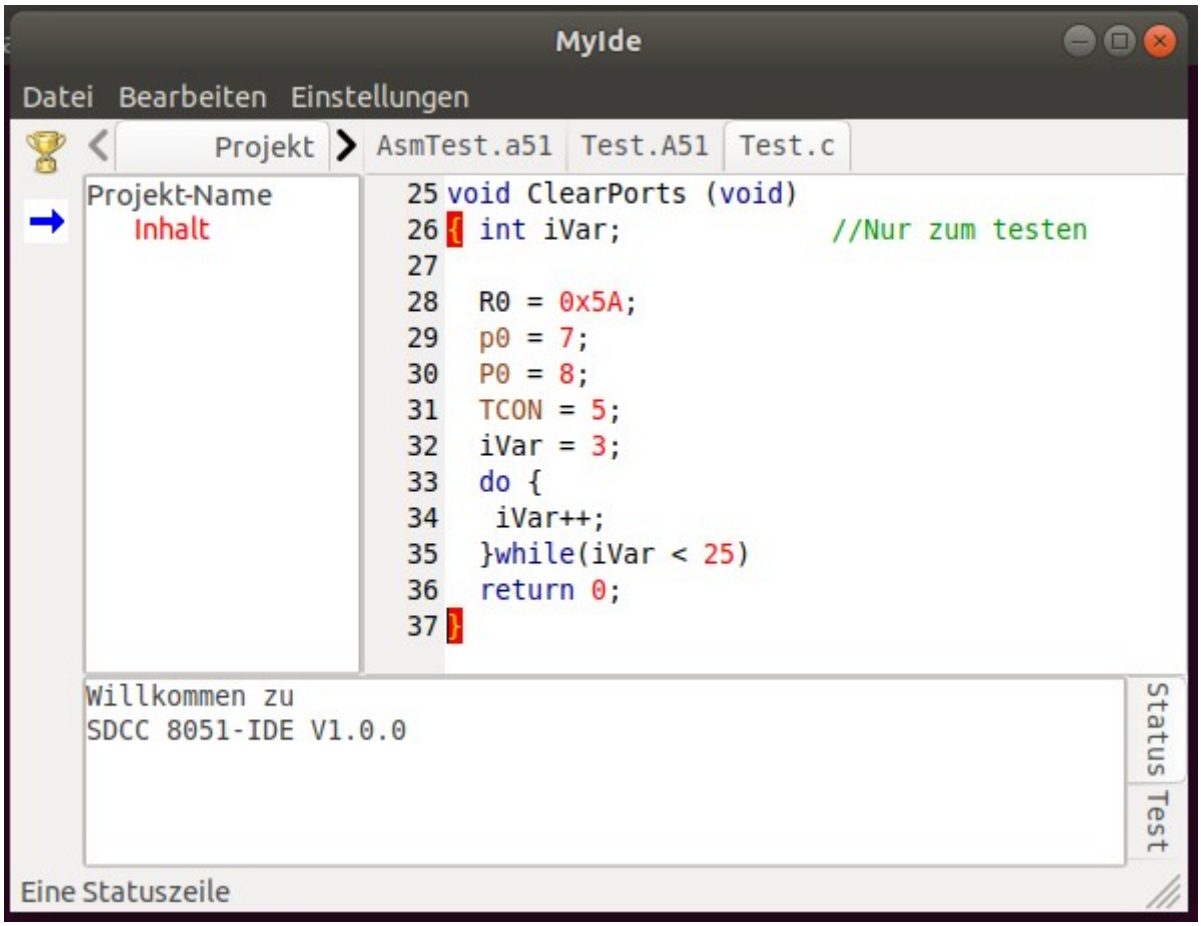
```
   void OnKlammer(wxStyledTextEvent& event);
```

Main:

```
//Klammernpaare finden
void IDEFrame::OnKlammer(wxStyledTextEvent& WXUNUSED(event))
{ int PosA, PosB, sel;
//Cursorposition in der aktuellen Datei ermitteln
  sel = EditNBook->GetSelection();      //Nummer der aktiven Datei
  PosA = wSTCPtr[sel]→GetInsertionPoint();
  PosB = wSTCPtr[sel]→BraceMatch(PosA-1);
  this->wSTCPtr[sel]->BraceHighlight(PosB, PosA-1);
}
```

In SetAsmStyle und SetCStyle die gewünschten Styles einstellen:

```
   wSTCPtr[iCo]->StyleSetForeground(wxSTC_STYLE_BRACELIGHT, gelb);
  wSTCPtr[iCo]->StyleSetBackground(wxSTC_STYLE_BRACELIGHT, rot);
```



4.8 Ein externes Programm starten

Das SDCC-Paket besteht aus, vom eigenen Programm, unabhängigen Prorammen zur Arbeit mit Assembler- und C-Programmen. Diese Dienstprogramme werden innerhalb des eigenen Programmes nach Bedarf aufgerufen. Im Beispiel werden der C-Compiler und Assembler des SDCC-Paket aufgerufen.

Include:

```
   void OnCompile(wxCommandEvent& event);  //Event: übersetzen
```

Main:

```
// Compilieren / Assemblieren
void IDEFrame::OnCompile(wxCommandEvent& event)
{ uint32_t iCo;
  wxArrayString asOut;
  wxArrayString asErr;
  wxFileName fValue;
  wxString wxStr;
  wxProcess* wxProc = new wxProcess(this);
  OnSave(event);    //Datei speichern
  iCo = EditNBook->GetSelection();    //Nummer der aktiven Notebook-Page
//Dateityp (c/asm) erkennen
  fValue = wSTCPtr[iCo]->GetName();
  wxStr = fValue.GetExt();
  if((wxStr[0] == 'a') | (wxStr[0] == 'A'))
    wxStr = "sdas8051 -plosgff ";
  else
    wxStr = "sdcc -mmcs51 --debug --verbose --nolospre ";
//Aufruf vervollständigen
  wxStr = wxStr + wSTCPtr[iCo]->GetName();
//SDCC aufrufen
  wxProc->Redirect();   //Anzeige umleiten
  wxExecute(wxStr, asOut, asErr, wxEXEC_NODISABLE);
  StatusCtrl->Clear();              //Alten Text löschen
  iCo = asErr.GetCount();
  if(iCo)
  { StatusCtrl->SetDefaultStyle(wxTextAttr(*wxRED));
    for(uint32_t i = 0; i < iCo; i++)
    { wxStr = asErr.Item(i) + "\n";
      StatusCtrl->AppendText(wxStr);    //Text schreiben
    }
  } else
  { iCo = asOut.GetCount();
    StatusCtrl->SetDefaultStyle(wxTextAttr(*dunkelgruen));
    if(iCo)
    { for(uint32_t i = 0; i < iCo; i++)
      { wxStr = asOut.Item(i) + "\n";
        StatusCtrl->AppendText(wxStr);    //Text schreiben
      }
    } else
        StatusCtrl->AppendText("Fehlerfrei\n");    //OK-Text
  }
  StatusCtrl->SetDefaultStyle(wxTextAttr(*wxBLACK));
  delete wxProc;
}
```

4.8.1 Programmausgaben im Statusfenster

Am Ende des vorigen Listings ist die Umleitung der Ausgaben des SDCC in das Statusfenster der eigenen Anwendung zu sehen. So wird die Umleitung ermöglicht:

```
  wxProc->Redirect();   //Anzeige umleiten
```

Und dann im eigenen Fenster angezeigt:

```
  StatusCtrl->Clear();              //Alten Text löschen

  wxStr = asOut.Item(i) + "\n";     //Normalausgabe
```

oder:

```
  wxStr = asErr.Item(i) + "\n";     //Fehlerausgabe

  StatusCtrl->AppendText(wxStr);    //Text schreiben
```

Hier wird die Normal-Ausgabe (asOut.Item(i)), oder die Fehlerausgabe (asErr.Item(i)) des SDCC angezeigt!

4.10 Einen Timer einrichten

Manchmal benötigt man ein zyklisch auftretendes Ereignis, um irgendeine Bearbeitung daran zu binden. Dafür sind Timer gut geeignet, sie arbeiten mit einer Auflösung von Millisekunden. Im Beispiel wird der Dialog alle 5 Sekunden angezeigt.

Include:

```
#include <wx/timer.h>

void OnTimer(wxTimerEvent&);
```

Main:

Im Konstruktor:

```
 MyTim = new wxTimer(this, wxID_ANY);
 MyTim->Start(5000, false);


//Millisekunden-Timer der alle 5s. auftritt
void IDEFrame::OnTimer(wxTimerEvent& WXUNUSED(aEvent))
{
  MyTim->Stop();
  wxMessageBox(wxT("Timer-Routine"),
           wxT("Meldung"), wxOK| wxICON_INFORMATION);
  MyTim->Start(5000, false);
}
```

5 Sonstiges

5.1 Programm starten

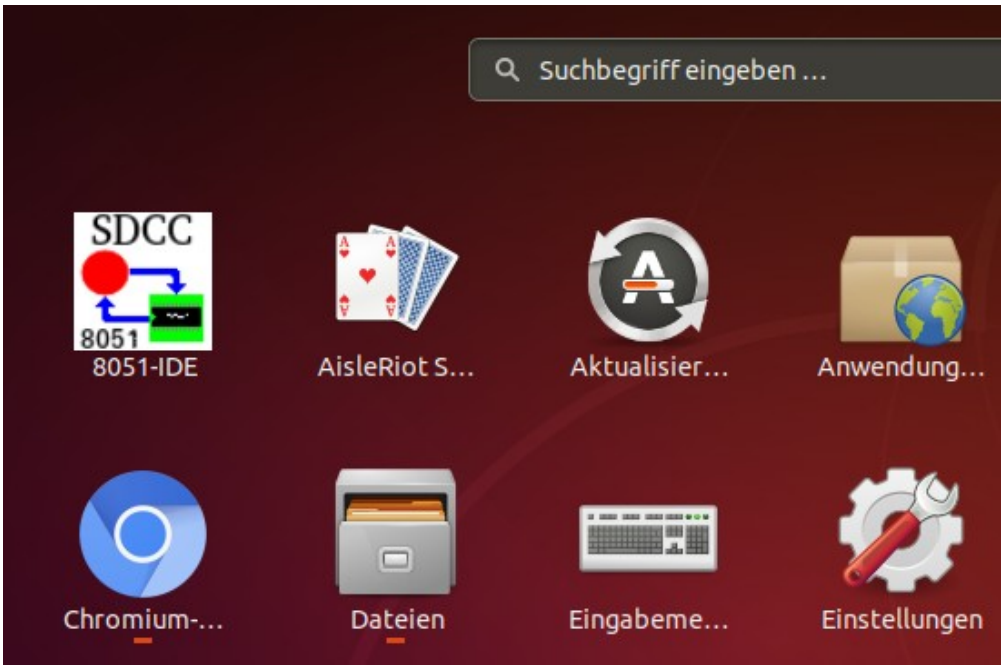
Aus der Entwicklungsumgebung **oder**

Von einem Terminal im Programmverzeichnis durch „./MyIDE“ **oder**

Im versteckten Verzeichnis „/home/BENUTZER/.local/share/applications“ eine Textdatei mit dem Namen „MyIDE.desktop“ mit folgendem Inhalt anlegen:

```
[Desktop Entry]
Type=Application
Icon=/home/tom/LDaten/QtProgs/MyIDE/Logo64.xpm
Name=MyIDE
Name[de]=8051-IDE
Comment=Programme erstellen
Categories=Utility;Core;GTK;
Path=/home/tom/Ldaten/QtProgs/MyIDE/
Exec=/home/tom/LDaten/QtProgs/MyIDE/MyIDE
StartupNotify=true
Terminal=false
MimeType=x-directory/normal;inode/directory;
```

In „Icon“, „Path“ und „Exec“ sind die eigenen Pfade einzutragen. Nun ist die Applikation über ihr Desktop-Icon erreichbar!



5.2 Farbnamen definieren

Anstatt mit nichtssagenden Nummernkombinationen wie „wxColor(0, 0, 255)“ kann man für die Farben auch aussagekräftige Namen definieren. Aus „wxColor(0, 0, 255)“ wird dann „blau“.

Include:

```
//Die Grundfarben
const wxColour schwarz = wxColour(0, 0, 0);
const wxColour weiss = wxColour(255, 255, 255);
const wxColour gruen = wxColour(0,255,0);
const wxColour rot = wxColour(255, 0, 0);
const wxColour blau = wxColour(0, 0, 255);
const wxColour gelb = wxColour(255, 255, 0);
const wxColour grau = wxColour(47,79,79);
const wxColour orange = wxColour(255,140,0);
const wxColour cyan = wxColour(0, 255, 255);
const wxColour magenta = wxColour(255, 0, 255);
const wxColour violet = wxColour(160,32,240);
```

Main:

Aus z.B:

```
wTCPtr->StyleSetForeground(wxSTC_C_DEFAULT, wxColor(0, 0, 255));
```

wird:

```
wTCPtr->StyleSetForeground(wxSTC_C_DEFAULT, blau);
```

Weitere Farbcodes sind in RGB-Farbtabellen im Netz zu finden.

5.x Komandozeilen-Argumente

Befindet sich in den Komandozeilen-Parametern ein Semikolon „;“, so ist diesem ein Backslash voranzustellen (aus „;“ wird „\;“)