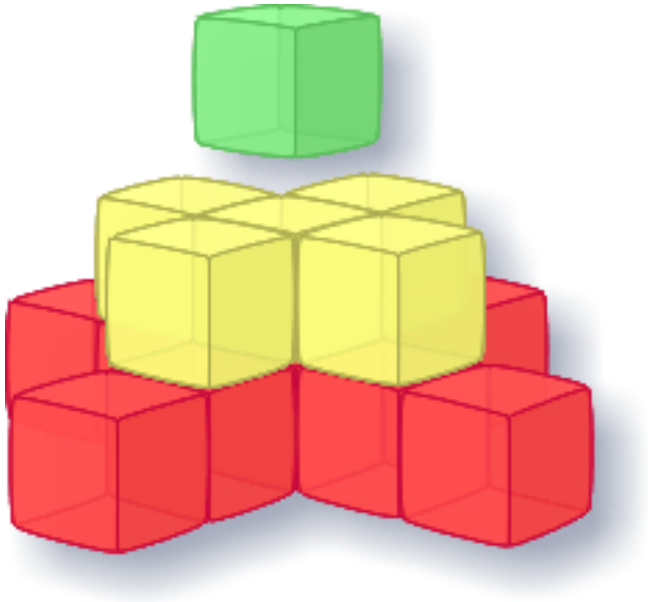
Dipl. Ing. Toralf Riedel
Dipl. Ing. Päd. Alexander Huwaldt

Benutzerhandbuch SiSy®

für die Programmierung von Mikrocontroller- und PC-Anwendungen

gültig ab SiSy-Version 3.4



1. Auflage: April 2013

www.laser-co.de
www.myavr.de
service@myavr.de
Tel: ++49 (0) 3585 470 222
Fax: ++49 (0) 3585 470 233

Inhalt

1 Einleitung

Sie haben eine Ausgabe des Modellierungswerkzeuges Simple System, kurz SiSy, erworben. Bevor auf die verschiedenen Funktionen des Programms eingegangen wird, noch einige Worte zum vorliegenden Handbuch. Mit Hilfe des Handbuchs werden dem Nutzer die Grundlagen der Bedienung von SiSy erläutert. Der Inhalt, die Gestalt und die Regeln der Modelle werden nur am Rand betrachtet. Das genaue Vorgehen für die Programmierung eines Mikroprozessors wird an einem Beispiel ausführlich beschrieben. Auf die Grundlagen der Mikroprozessorprogrammierung wird im Rahmen dieses Handbuches nicht eingegangen. Dazu dienen die myAVR Lehrbücher.
Dem Nutzer wird in diesem Handbuch der Einstieg in das Programm erleichtert und die umfangreichen Funktionen von SiSy kurz und verständlich beschrieben. Bei der Arbeit mit SiSy erstellt der Anwender Modelle in Form von Diagrammen und in ihnen enthaltene Symbole. Die Grundlagen der Entstehung und Bearbeitung solcher Diagramme sind Gegenstand der Betrachtung dieses Handbuchs.
Folgende Darstellungs- und Gestaltungsmittel sind für den Nutzer bei der Arbeit mit SiSy besonders wichtig:

- die Diagramme als Fenster zur Ansicht und Bearbeitung von Modellen;
- der Navigator als Fenster zur Steuerung und Bewegung in Modellen;
- der Assistent mit Hilfestellungen zum jeweils geöffneten Diagramm und mit Diagrammvorlagen (wenn vorhanden);
- die Menüs und Schalter für Befehle an Navigator, Diagramm und Objekt im Kontext mit der Modellierung.

Zu den Bezeichnungen im Text:

- Falls ein Menübefehl nur über Untermenüs zu erreichen ist, werden die einzelnen Menübezeichnungen kursiv geschrieben und durch Schrägstriche voneinander getrennt.
  Beispiel: Menü *Hilfe/über SiSy*

- Titel von Dialogboxen, Schaltflächen und Menüpunkten werden in Anführungs-Zeichen gesetzt.
  Beispiel: Dialogbox „Definition“, Schaltfläche „OK“

Zu den beschriebenen SiSy Ausgaben und Referenzhardware

Die Beschreibungen in diesem Handbuch beziehen sich auf die SiSy Ausgaben

- SiSy AVR: Mikrocontrollerprogrammierung für AVR-Programmierung in Assembler und C/C++ sowie UML; die verwendete Referenzhardware für die Beispielprogramme ist ein myAVR Board MK2
- SiSy ARM: Mikrocontrollerprogrammierung für ARM- Programmierung in C/C++ sowie UML; die verwendete Referenzhardware für die Beispielprogramme ist ein Board STM32F4-Discovery
- SiSy Mikrocontroller++: Mikrocontrollerprogrammierung für AVR-, ARM- und Windows-Programmierung

Abbildungen im Handbuch

Die Abbildungen in diesem Handbuch beziehen sich auf die jeweils gestellte Aufgabe und sind mit der entsprechenden Ausgabe erstellt.

2 Vorbereitung

In diesem Kapitel werden Sie über notwendige Schritte zur Installation, Konfiguration und Aufbau einer funktionsfähigen Entwicklungsumgebung informiert.

2.1 Installation der Software

Für die Bearbeitung der Übungen und Aufgaben steht Ihnen die Entwicklungsumgebung SiSy AVR, SiSy ARM bzw. SiSy Microcontroller ++ zur Verfügung. Sollten Sie SiSy bereits installiert haben, können Sie dieses Kapitel überspringen.
Die Installation und der erste Start müssen mit Administratorrechten ausgeführt werden.

2.1.1 Voraussetzungen

Für die Installation benötigen Sie einen Freischaltcode (Lizenzangaben). Falls Sie diese Angaben nicht mit der Software erhalten haben, können Sie diese online abrufen von

www.sisy.de → Meine SiSy-Lizenz

oder

www.myAVR.de → Online-Shop→ Kontakt/Service

oder fordern Sie diese beim Hersteller an:

Tel: 03585-470222
Fax: 03585-470233
e-Mail: support@myAVR.de.

Bitte prüfen Sie, ob die Systemvoraussetzungen für die Installation und die Arbeit mit SiSy für die Mikrocontrollerprogrammierung gewährleistet sind.

- PC-Arbeitsplatz oder Notebook mit USB-Anschluss
- PC mit Windows XP, Vista oder Windows 7
- Microsoft Internet-Explorer 7 oder höher
- Maus oder ähnliches Zeigegerät
- Assembler bzw. C/C++ Entwicklungsumgebung (in SiSy bereits integriert)
- myAVR Board MK2 bzw. STM32F4-Discovery
- Programmierkabel
  - USB Kabel (myAVR Board MK2)
  - Mini-USB-Kabel (STM32F4-Discovery)
- Bei Bedarf (z.B. autonomer Einsatz des Boards) geeignete Spannungsversorgung z.B. 9 V Batterie / stabilisiertes 9 V Netzteil

Des Weiteren sollten Sie Grundkenntnisse in einer beliebigen Programmiersprache besitzen.

2.1.2 Setup von der SiSy-CD

Legen Sie die CD „SiSy“ in Ihr CD-ROM-Laufwerk ein. Falls die CD nicht automatisch startet, wählen Sie bitte im Explorer das CD-ROM-Laufwerk und starten die „setup.exe“ aus dem Pfad *CD-Laufwerk:\Ausgabe\SiSy*.
Auf dem Startbildschirm stehen Schaltflächen zur Verfügung zum Installieren der Software und zum Öffnen von Begleitdokumenten.
Für die Installation der Software betätigen Sie die entsprechende Schaltfläche. In Abhängigkeit Ihrer Rechnerkonfiguration kann der Start des Setup-Programms einige Sekunden dauern. Das gestartete Setup-Programm wird Sie durch die weitere Installation führen.

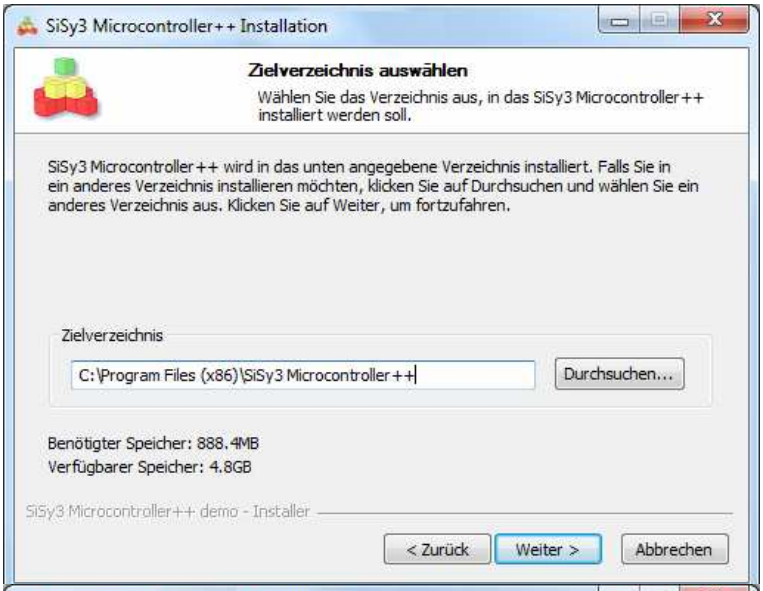
Beginn der Installation

Betätigen Sie im Setup-Programm die Schaltfläche „Weiter“. Sie erhalten die Lizenzbestimmungen. Bitte lesen Sie diese sorgfältig durch. Wenn Sie sich mit diesen Bestimmungen einverstanden erklären, bestätigen Sie die Lizenzbestimmungen mit der Schaltfläche „Annehmen“.

Sie werden im folgenden Dialog dazu aufgefordert, Ihre Lizenzangaben einzugeben.

Danach erscheint die Dialogbox „Komponenten auswählen“, welche Sie mit „Weiter“ bestätigen.

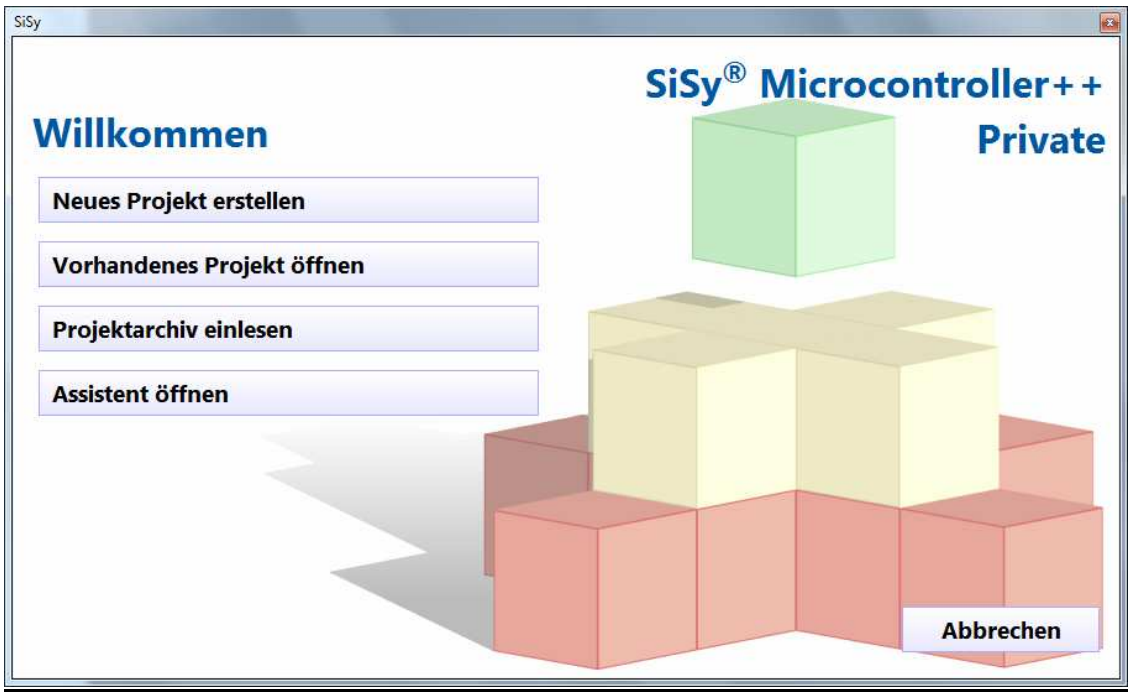
Im darauf folgenden Fenster können Sie festlegen, unter welchem Pfad SiSy installiert werden soll. Wenn ein anderer Pfad (bzw. ein anderes Laufwerk) gewünscht wird, aktivieren Sie die Schaltfläche „Durchsuchen“. Eine Dialogbox erscheint, in der Sie Laufwerk und Verzeichnis auswählen können.



Bestimmen Sie danach den Startmenü-Ordner, in dem die Verknüpfungen von SiSy eingefügt werden. Sie können den Zielordner ändern; Sie können dies durch auswählen von „Keine Verknüpfungen erstellen“ unterbinden.

Beginnen Sie nun die Installation durch Betätigen der Schaltfläche „Installieren“. Während der Installation erhalten Sie ggf. Hinweise, dass verschiedene Treiber installiert bzw. aktualisiert werden sollten. Es wird stets eine Eingabe erwartet. In Abhängigkeit Ihrer Rechnerkonfiguration kann die komplette Installation einige Minuten in Anspruch nehmen.
Zum Abschluß der Installation wird ein entsprechendes Dialogfeld angezeigt. Mit dem Aktivieren der Schaltfläche „Fertig stellen“ ist die Installation abgeschlossen.

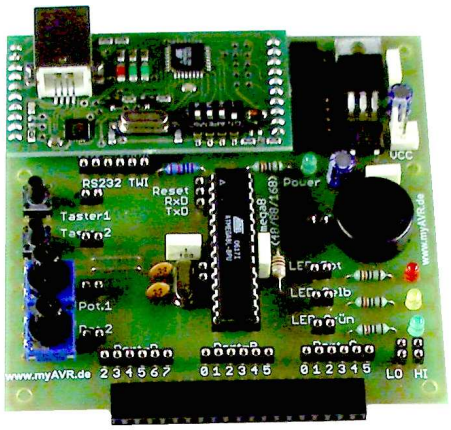
Sie können nun SiSy starten. Es erscheint auf Ihrem Bildschirm der Dialog „Willkommen in SiSy“. Folgen Sie dann den Hinweisen des Assistenten, indem Sie „Assistent öffnen“ auswählen.



Hinweis:
In SiSy sind 2 Dateien enthalten, die Makros beinhalten („handbuch.doc“, „multi.doc"). Von einigen Virenscannern werden diese Makros als „Virus“ erkannt und entsprechend behandelt. In den Heuristik-Einstellungen des Virenscanners kann diese Behandlung unterdrückt werden.

2.2 Beschaffen bzw. Herstellen der Hardware

Alle Ausführungen, Übungen und Aufgabenstellungen in diesem Benutzerhandbuch beziehen sich für die AVR-Programmierung auf ein myAVR Board MK2 und für die ARM-Programmierung auf ein STM32F4-Discovery als Referenzhardware. Dokumente sowie Anwendungsbeispiele zu myAVR Boards stehen zum Download bereit unter www.myAVR.de. Für das STM32F4-Discovery sind frei verfügbare, fertig lauffähige Beispiele auf www.st.com erhältlich.



myAVR Board MK2 mit USB-Port

STM32F4-Discovery mit Mini USB-Port

Für die Bearbeitung der Beispiele in diesem Handbuch benötigen Sie die nachfolgend aufgeführte Hard- bzw. Software; sie ist exemplarisch zu verstehen.

	PC-Programme	**AVR-Programme**	**ARM-Programme**
Hardware		• bestücktes myAVR Board • Programmierkabel (USB) • Patchkabel • 9 V Netzteil / Batterie bei Bedarf (z.B.: autonomer Einsatz)	• STM32F4-Discovery • Programmierkabel (Mini-USB) • STM-Patchkabel • STM32-Board-F4D (optional)
Software	SiSy Microcontroller ++ oder SiSy Professional	SiSy Microcontroller ++ oder SiSy AVR oder eine andere SiSy Ausgabe mit dem Add-On AVR	SiSy Microcontroller ++ oder SiSy ARM oder eine andere SiSy Ausgabe mit dem Add-On ARM

Ausführliche Beschreibungen zur Programmierung mit Assembler sowie C/C++ sind nicht Inhalt dieses Handbuches. Weiterführende Erklärungen dazu sind im „myAVR Lehrbuch Mikrocontrollerprogrammierung“ enthalten.

3 Arbeiten mit SiSy, allgemein

Um dem Nutzer zu erläutern, wie er in SiSy modellieren kann, werden zweckentsprechende Definitionen der Begriffe gegeben.

3.1 Projektarbeit

3.1.1 Was ist ein SiSy-Projekt?

Ein SiSy-Projekt ist eine abgegrenzte Menge von verknüpften Elementen für ein zu bearbeitendes Problem. Alle aktuellen Daten sind in einer Projektdatenbank gespeichert. Die Projektdatenbank besteht aus einer Anzahl von Dateien im Projektverzeichnis, wobei jedes Projekt sein eigenes Verzeichnis hat. Durch das Anlegen eines Projektarchivs können diese in einer Datei komprimiert werden.

3.1.2 Neues Projekt erstellen

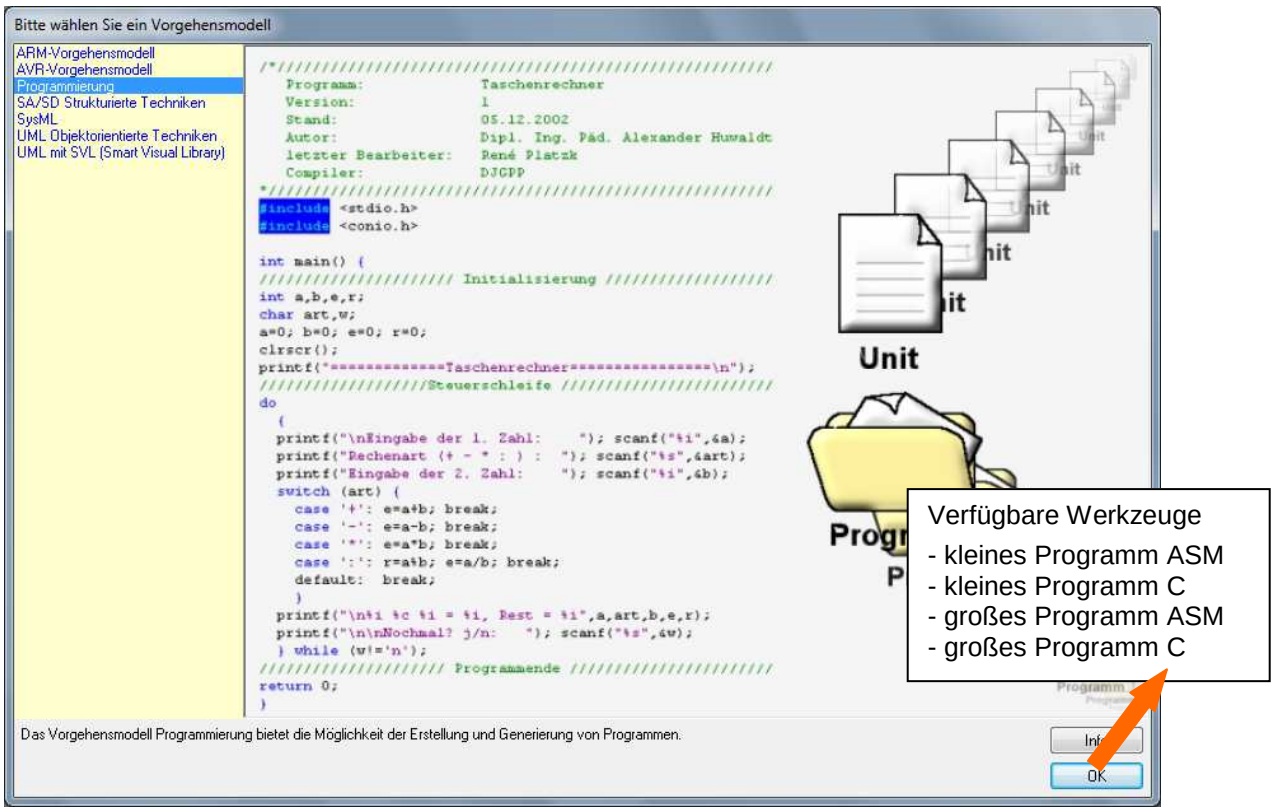
In SiSy legen Sie stets ein Projekt an. In dieses Projekt integrieren Sie Ihr Programm bzw. mehrere Programme. Unabhängig vom Programmnamen benötigt jedes Projekt einen Namen.
Ein neues Projekt wird eingerichtet und für die Bearbeitung bereitgestellt. Die Definition (Erstellung) eines neuen Projektes erfolgt durch Vergabe eines Projektdateinamens und/oder durch Überschreiben eines alten Projektes gleichen Namens. Über die Schaltfläche „Ordner für Projekte ändern“ kann die vorgeschlagene Pfadangabe geändert werden.

Hinweis:
Falls ein bereits vorhandener Projekt- oder Verzeichnisname gewählt wird, erscheint folgende Meldung:

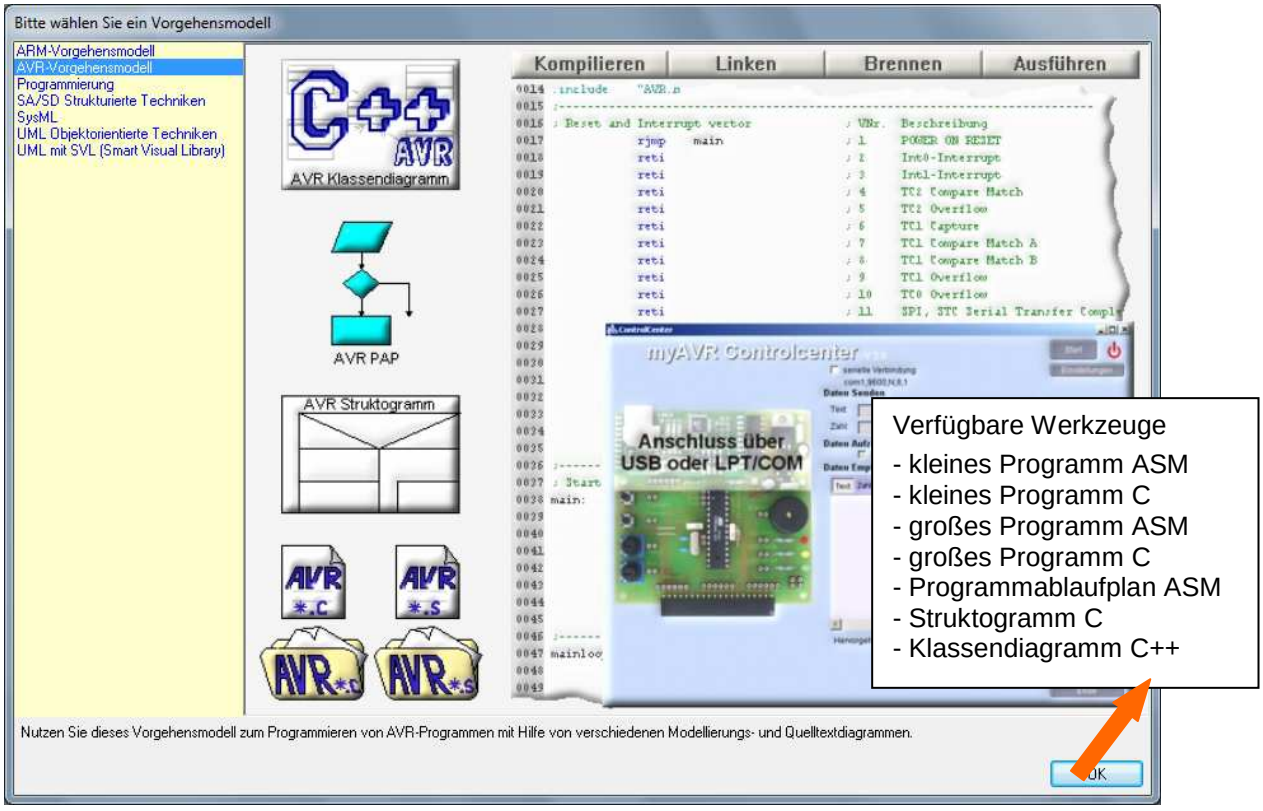
> *„Achtung: ein Projekt dieses Namens existiert bereits. Ändern Sie den Projektnamen oder das vorhandene Projekt wird gelöscht und durch das neue ersetzt.“*

Vorgehensmodell auswählen

Nach der Vergabe eines Projektnamens kann im nachfolgenden Fenster ein Vorgehensmodell ausgewählt werden. Die Auswahl des Vorgehensmodells bestimmt im weiteren Projektverlauf die von SiSy zur Verfügung gestellten Werkzeuge. Je nach Vorgehensmodell können Vorlagen oder Assistenten das Erstellen eines neuen Projektes unterstützen.
Stehen in Ihrer SiSy Ausgabe mehrere Vorgehensmodelle (Hauptebenen) zur Verfügung, wird eine Auswahl der Hauptebene angeboten. Mit der Auswahl des Vorgehensmodells (VGM) entscheiden Sie über das Profil des Projektes.
Ist ein Projekt klein und die Modellierung von Programmablaufplänen nicht erforderlich, wählt man ein einfaches Vorgehensmodell; in den gezeigten Beispielen im Abschnitt 4 das Vorgehensmodell „Programmierung“. Damit sind die Menüs und Objektbibliotheken entsprechend übersichtlicher.
Bei komplexen Projekten, für die neben kleinen Programmen auch Programmablaufpläne, Struktogramme oder Klassendiagramme nötig sind, wird ein komplexeres Vorgehensmodell wie z.B. „AVR-Vorgehensmodell“ bzw. „ARM-Vorgehensmodell“ ausgewählt.
Die folgenden Abbildungen zeigen für 2 verschiedene Vorgehensmodelle die verfügbaren Werkzeuge.



Vorgehensmodell „Programmierung“ mit den verfügbaren Werkzeugen



Vorgehensmodell „AVR Vorgehensmodell“ mit den verfügbaren Werkzeugen

Im weiteren Verlauf werden entsprechend der Auswahl des Vorgehensmodells unterschiedliche Hilfen zum Erstellen eines Programms angeboten. Für Einsteiger empfiehlt sich zuerst die Nutzung des Vorgehensmodells „Programmierung“.

3.1.3 Vorhandenes Projekt öffnen

Ein vorhandenes Projekt wird geöffnet und die abgespeicherten Diagramme und Objekte sowie alle Möglichkeiten für die weitere Bearbeitung werden verfügbar gemacht. Die Wahl des Projektes erfolgt durch Klicken auf den entsprechenden Namen oder über die Schaltfläche „Projekt suchen“.
Ist der Mauszeiger auf dem Projektnamen oder auf den Symbolen, dann werden Informationen angezeigt.

3.1.4 Projekt archivieren

Menü *Projekt/Archiv/Anlegen.*
Es kann ein komprimiertes Archiv des Projektes erzeugt werden. Dies ist besonders aus Gründen der Datensicherheit sinnvoll. Zielverzeichnis und Dateiname für die Archivdatei werden vorgeschlagen und können korrigiert werden. Beim Aktivieren der Checkbox „Datum anfügen“ wird das Projekt mit Änderungsdatum gespeichert. Wenn ein Projekt unter einem bereits vorhandenen Archivnamen angelegt werden soll, wird eine Warnung vor dem Überschreiben angezeigt. Bei Auswahl von „Nein“ wird die Erstellung des Archivs abgebrochen, bei „Ja“ wird das Projekt archiviert.

Hinweis:
SiSy bietet die Möglichkeit des regelmäßigen Abspeicherns verschiedener Arbeitsstände, d.h. ein archiviertes Projekt wird nicht überschrieben. Ein Projektstand kann in einer neuen Archivdatei abgelegt werden.

3.1.5 Projektarchiv einlesen

Menü *Projekt/Archiv/Einlesen*.
Hierunter versteht man das Einlesen eines Archivs zum Zweck der Rekonstruktion des Projektes.
Einlesen bedeutet Entpacken eines archivierten Projektes. Dazu muss das Archiv ausgewählt sein. Beim Aktivieren der Checkbox „Datum entfernen“ wird das Datum nicht im Projektnamen geschrieben.

Hinweis:
Wenn im Zielpfad des Entpackens bereits ein Projekt existiert, erscheint eine Überschreibwarnung.

3.1.6 Projektarchiv als Email versenden

Menü *Projekt/Archiv/Als Email versenden….*
Um Projekte per Email zu versenden, sollten diese zu einem Archiv zusammengefasst werden. Die Funktion „Projektarchiv als Email versenden“ erzeugt von dem aktuell geöffneten Projekt ein Archiv, startet das Standard-Email-Programm Ihres Systems, legt eine neue Email an und fügt das Archiv als Anlage ein. Sie können jetzt die Email schreiben und sofort das Projektarchiv senden.

Hinweis:
Das Standard-Email-Programm Ihres Systems muss ordnungsgemäß eingerichtet und registriert sein, damit diese Funktion korrekt ausgeführt werden kann. Es ist nicht möglich, diese Funktion mit einem WEB-Browser-Email-Programm zu nutzen.

3.2 Die Modellierungselemente von SiSy

Werkzeug

SiSy stellt für die Bearbeitung der Modelle und Teilmodelle Werkzeuge der entsprechenden Methodik bereit. Werkzeuge sind Editoren, mit denen in einem Fenster die grafische Darstellung (Diagramme) der Modelle bearbeitet werden kann.

Modellelemente

Modelle bestehen aus Elementen, welche zueinander in Beziehung stehen. Diese Modellelemente werden nach bestimmten Regeln (Metamodell) durch SiSy verarbeitet und dargestellt (Notation). Modellelemente sind in der Regel Knoten (Objekte), Kanten (Verbindungen), Rahmen, Referenzen auf Modellelemente und Texte. Prinzipiell kann jedes Modellelement verfeinert werden. Typisch ist es Knoten (Objekte) dadurch zu verfeinern, dass „unter“ diesen Objekten weitere Modellelemente angelegt werden. Dabei entsteht eine Modellhierarchie. Jedes Modellelement verfügt über einen Kurznamen, einen Langnamen, eine Beschreibung und modellinterne Attribute (Inhalt).

Diagramme

Diagramme sind grafische Darstellungen von Modellen oder Teilmodellen, die mit einem bestimmten Werkzeug erstellt werden. Die Modellierungselemente werden als Objekte in den Diagrammen unter Einhaltung von Regeln zusammengestellt.

Objekte

Objekte sind mögliche Modellelemente in Diagrammen, z.B. „kleines Programm“ in der „Programmierung“. Objekttypen sind konkrete Ausprägungen von Objekten, die in einem Diagramm angelegt wurden, z.B. Objekttyp „Lauflicht“ vom Objekt „kleines Programm“.

Referenzen

Die Objekte eines Diagramms können in anderen Diagrammen wieder verwendet werden. Durch das Hineinziehen aus dem Navigator oder aus einem offenen Diagramm wird eine Referenz vom Originalobjekt erzeugt. Die Referenz ist nur ein Verweis auf das Original, alle angezeigten Informationen wie Kurzname, Langname und Beschreibung werden vom Original bezogen. Somit sind die Informationen in allen Referenzen eines Objektes identisch mit dem Original. Dadurch werden Änderungen dieser Informationen automatisch auf alle Referenzen übertragen. Weiterhin ist es möglich, diese Referenzierung über einen so genannten Report auszuwerten.

Kanten

Kanten sind Verbindungselemente zwischen Objekten. Eine Verbindung wird durch Ziehen mit der Maus (linke Maustaste) vom Verteiler des selektierten Objektes auf das gewünschte Objekt erreicht. Nach Loslassen der Maustaste und Prüfung der Verbindungszulässigkeit durch SiSy erscheint ein Kanten-Dialog, in dem das Element definiert und individuelle Einstellungen getroffen werden können.

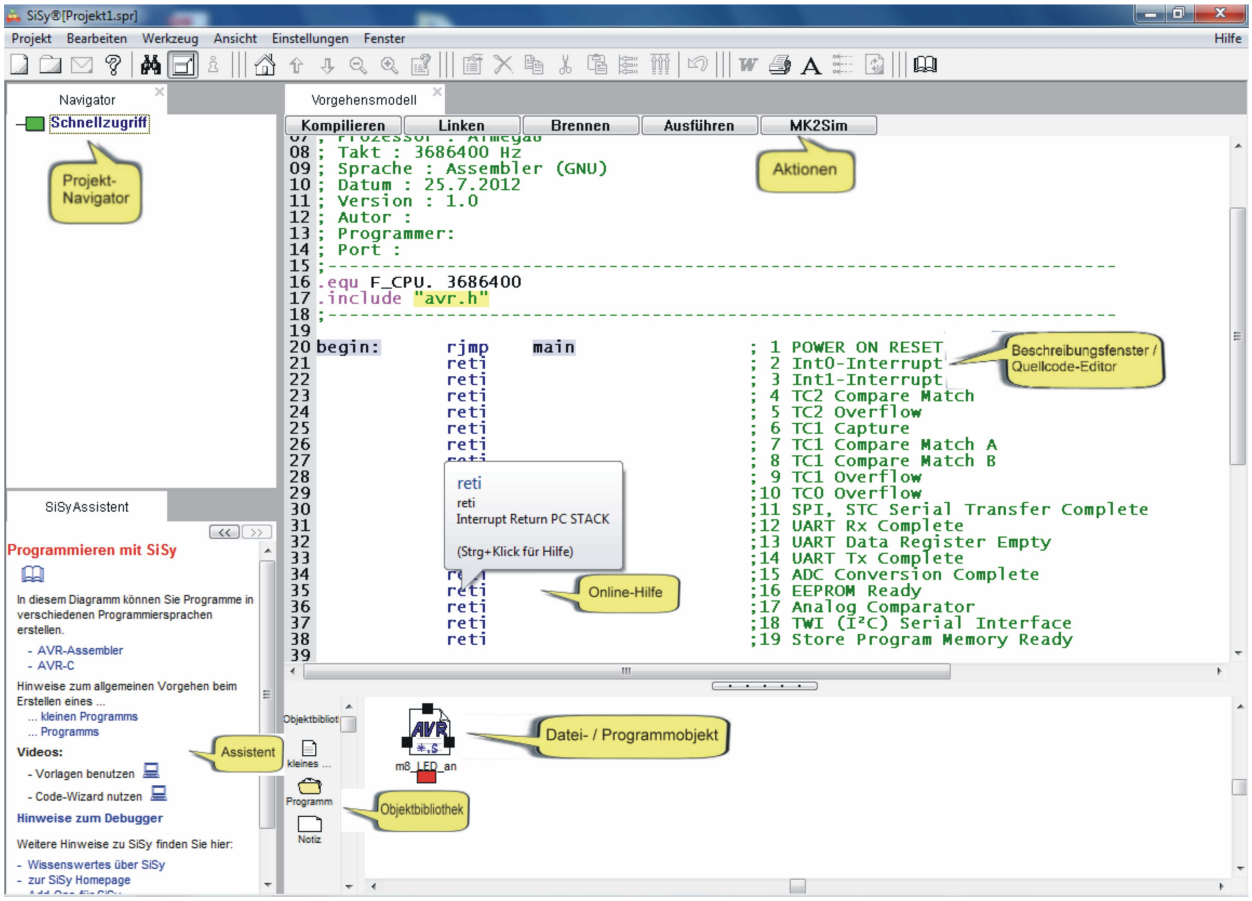
Hinweis:

Bei Verbindung mit gehaltener STRG-Taste wird die Prüfung vernachlässigt und eine Zwangsverbindung erreicht.

Rahmen

Ein Rahmen fasst ausgewählte Objekte des Diagramms optisch zusammen. Er besitzt einen Kurz- sowie Langnamen und eine Objektbeschreibung, kann also als Objekt aufgefasst werden. Inhalte von Rahmen sind in Reports oder einer Dokumentengenerierung auswertbar.

3.3 Die Fenster für die Modellierung



Navigator
Dieser befindet sich im linken, oberen Teil des Fensters. Er ermöglicht dem Anwender die Orientierung über die in der Projektdatenbank gespeicherten Objekte sowie deren Bereitstellung für die weitere Verwendung. Nach dem Start von SiSy werden neben dem Vorgehensmodell eine Reihe leicht zu handhabender Schaltflächen, Menüs und weitere Hilfsmittel angezeigt.

SiSy Assistent
Am linken unteren Bildschirmrand befindet sich diese Nutzerunterstützung.

- Er begleitet den Anwender durch das gesamte Projekt und hält immer passende Informationen zur aktuellen Sicht parat
- Er ist auf die jeweilige Ausgabe von SiSy bezogen.
- Oft können Beispiele als Vorlagen vom Assistenten geladen werden.

Diagrammfenster
Am oberen Bildrand befinden sich das Hauptmenü, das hilfreiche Funktionen zu SiSy bereithält, und eine Werkzeugleiste, mit deren Hilfe schnell auf nützliche Anwendungen zugegriffen werden kann. Das Diagrammfenster nimmt die rechte Bildschirmseite ein und ist der Raum, in dem der Nutzer modelliert. Es enthält:

- das ausgewählte Vorgehensmodell,
- die Objektbibliothek mit den möglichen Objekten des aktuellen Diagramms sowie
- ein Fenster zur Beschreibung des markierten Objekts, in diesem Fall zum Editieren des Quelltextes.

Die Bedienelemente/Objektbibliothek

SiSy bietet, wie bei Windows-Anwendungen üblich, die Steuerung von Befehlen über das Hauptmenü, über die Werkzeugleisten, die Tastatur oder die Objektbibliothek an. Darüber hinaus enthalten das Kontextmenü und der Navigator Steuerfunktionen.

Die Anzahl der möglichen Befehle in der Menüleiste ist abhängig davon, ob ein Projekt geöffnet ist. Ist das nicht der Fall, erscheint ein Menü mit wenigen Befehlen. Bei einem geöffneten Projekt hält SiSy umfangreichere Menüs bereit. Die wichtigsten Menübefehle befinden sich auch als grafische Schaltfläche in der Werkzeugleiste, die eine schnelle und effiziente Bedienung des Programms ermöglicht. Die Toolbox-Darstellung bietet dem Anwender wichtige Programmfunktionen als direkten Link an.
Ein weiteres Bedienelement ist die Objektbibliothek. Sie unterstützt das Anlegen neuer Objekte.

3.4 Modelle bearbeiten

3.4.1 Modellelemente anlegen

Modellelemente können auf verschiedene Art und Weisen erzeugt werden. Der häufigste Weg ist das Anlegen über die Objektbibliothek. Dabei wird das gewünschte Element per Drag und Drop von der Objektbibliothek in das dazugehörige Diagrammfenster gezogen. Referenzen auf vorhandene Modellelemente werden üblicherweise dadurch erzeugt, dass diese per Drag und Drop aus dem Navigator in das gewünschte Diagramm gezogen werden. Verbindungen lassen sich am Einfachsten über den Verteiler herstellen. Sie können aber Modellelemente auch durch kopieren oder importieren anlegen.

3.4.2 Modellelemente auswählen

Die meisten Modellelemente lassen sich in Diagrammen auswählen (selektieren). Das Selektieren erfolgt in der Regel durch Anklicken (linke Maustaste) mit dem Mauscursor. Eine Selektion per Klick mit der rechten Maustaste ist ebenfalls möglich, dabei öffnet sich nach dem Auswählen sofort das Kontextmenü für das selektierte Modellelement. Das Betätigen der Tabulatortaste bewirkt das Auswählen des nächsten Modellelementes in der Reihenfolge des Anlegens der Elemente. Wird über die Auswahl mit der Tabulatortaste das letzte Element erreicht, beginnt die nächste Iteration wieder beim ersten Modellelement im Diagramm. Durch Betätigen der Umschalttaste kann die Iterationsrichtung umgeschaltet werden. Es ist ebenfalls möglich, mehrere Modellelemente auszuwählen. Dafür muss beim Selektieren mit der Maus die Umschalttaste gehalten werden. Bei gehaltener Umschalttaste lassen sich mehrere Objekte (keine Kanten und Rahmen) durch Ziehen einer Markise mit dem Mauscursor auswählen. Über den Menüpunkt *Bearbeiten/Diagramm/Alles markieren* können Sie alle Objekte des aktiven Diagramms selektieren.

3.4.3 Modellelemente untersuchen (Report)

Modellelemente sind in den meisten Fällen nicht isoliert sondern haben vielfältige Beziehungen zu anderen Modellelementen. Um bei Änderungen, wie zum Beispiel Umbenennen oder Löschen, mögliche Auswirkungen auf das gesamte Modell zu überblicken, bietet der Objektreport die Möglichkeit die Modellbeziehungen des ausgewählten Objektes auszuwerten. Wählen Sie ein Objekt aus und öffnen Sie den Reportdialog über das Hauptmenü, die Werkzeugleiste oder das Kontextmenü (diagrammabhängig). Wählen Sie die erforderliche Modellanfrage und starten den Report. Das Ergebnis der Modellanfrage (z.B. Alle angebundenen Objekte) wird als Liste angezeigt. Sie können dann zu diesen Modellelementen navigieren.

3.4.4 Modellelemente verschieben

Bestimmte Modellelemente (z.B. Knoten/Objekte) lassen sich im Diagramm frei anordnen. Das Positionieren erfolgt in der Regel durch Drag und Drop mit der Maus. Die Modellelemente werden dabei an einem Raster ausgerichtet. Das Raster erleichtert die Positionierung. Um die Rasterfangfunktion zu unterbinden, muss beim Drag und Drop eines Elementes die Umschalttaste betätigt werden. Viele Modellelemente lassen sich auch über die Cursortasten verschieben. Dabei bewegen sich die Modellelemente in Richtung der betätigten Taste auf dem Raster. Auch dabei lässt sich durch die Umschalttaste die Rasterfangfunktion unterbinden. Bei Kanten wird ein Stützpunkt verschoben, der in Abhängigkeit der ausgewählten Kantenform zum gewünschten Ergebnis der Darstellung führt.
Rahmen haben nur eine eigene Position, solange diese leer sind. Ein Rahmen bezieht sich in Position und Größe immer auf die darin enthaltenen Modellelemente (umschließen). Beim Verschieben eines Rahmens mit enthaltenen Elementen wird nicht der Rahmen, sondern die enthaltenen Modellelemente verschoben.

3.4.5 Die Größe von Modellelementen verändern

In manchen Diagrammen verfügen Modellelemente über Anfassmarken zur Änderung der Größe. Die manuelle Größenänderung von Modellelementen ist jedoch in vielen Diagrammtypen nicht erwünscht. In diesen Diagrammen stehen keine Anfassmarken zu Größenänderung zur Verfügung. Sollte in Ausnahmefällen eine manuelle Anpassung der Objektgröße gewünscht sein, kann dies für selektierte Objekte durch die Kombination von gehaltener Steuerungstaste und den Cursortasten bzw. die Tasten Plus, Minus und Mal des Numerikfeldes erfolgen.

3.4.6 Modellelemente verbinden

Knoten und Rahmen können über Kanten verbunden werden (Verbindungen). Verbindungen werden als Linien mit unterschiedlichen Linientypen und -formen sowie Symbolen (Anfang, Ende, Mitte) und diagrammspezifischen Beschriftungen dargestellt. Das Verbinden erfolgt in der Regel über den Verteiler (rotes Rechteck an der Unterkante) eines selektierten Objektes. Dabei wird per Drag und Drop vom Verteiler die Verbindung vom Quellelement zum Zielelement gezogen. Es ist nicht nötig exakt den Rand des Zielelementes zu treffen sondern besser, die Verbindung weit in das Zielobjekt hinein zu ziehen. Nach dem Loslassen der linken Maustaste wird die Verbindung erzeugt und die Randpunkte bis zu denen die Verbindung gezeichnet wird (Clipping) automatisch ermittelt. Sie können während des Drag und Drop durch Klick mit der rechten Maustaste bereits den zukünftigen Stützpunkt der Verbindung festlegen.

3.4.7 Modellelemente löschen

Zu löschende Modellelemente müssen zuerst selektiert werden. Die Löschaufforderung erfolgt mittels der Taste „Entfernen“, der Werkzeugleiste oder über das Hauptmenü *Bearbeiten/Objekt/Löschen* bzw. das Kontextmenü. Der Löschvorgang selbst ist abhängig von der Art des zu löschenden Modellelementes. Zum Beispiel werden frei stehende Referenzen ohne Rückfrage gelöscht, Originale werden in zwei Stufen gelöscht. Dabei erfolgt zuerst das Entfernen aus dem Diagramm und dann auf Anfrage auch das Löschen aus der Datenbank. Objekte die verfeinert oder verknüpft sind lassen sich nicht ohne erhöhten Aufwand aus dem Modell entfernen. Zuerst müssen die Modellbeziehungen des Elementes aufgelöst werden bis es für den Löschvorgang isoliert ist. Dazu springt in entsprechenden Fällen der Löschassistent an. Dieser gibt dem Anwender die Möglichkeit zuerst die Auswirkungen auf das Gesamtmodell zu beurteilen und kann dann auf Anforderung komplexe Löschvorgänge, zum Beispiel von ganzen Teilmodellen in der Modellhierarchie, durchführen.

3.4.8 Modellelemente kopieren

Um eine Kopie eines Modellelementes zu erzeugen wählen Sie im Menü oder in der Werkzeugleiste den Befehl „Kopieren“. Sie können dieses Objekt jetzt innerhalb des Projektes in ein Diagramm passenden Typs einfügen. Sollte das Diagramm den Objekttyp nicht akzeptieren, erhalten Sie dazu eine Meldung. Rahmenelemente wie zum Beispiel Klassen können mit deren Inhalt kopiert werden. Dabei werden jedoch Modellelemente in der Verfeinerung nach unten nicht berücksichtigt (flache Kopie). Innerhalb eines Diagramms können Sie von einzelnen Objekten durch Drag und Drop bei gleichzeitig gedrückter Steuerungstaste Kopien erzeugen.

Hinweis:
Beachten Sie, dass eine Kopie immer dem kopierten Element entspricht. Das heißt, die Kopie eines Originals ist wieder ein neues Original, die Kopie einer Referenz ist eine neue Referenz auf das ursprüngliche Original.

3.4.9 Modellelemente ausschneiden

Zum Ausschneiden eines Modellelements wählen Sie im Menü oder der Werkzeugleiste *Bearbeiten/Objekt/Ausschneiden*. Sie können dieses Objekt jetzt innerhalb des Projektes in ein Diagramm passenden Typs einfügen. Sollte das Diagramm den Objekttyp nicht akzeptieren, erhalten Sie dazu eine Meldung. Rahmenelemente wie zum Beispiel Klassen können mit deren Inhalt verschoben werden. Dabei werden auch Modell-Elemente in der Verfeinerung nach unten berücksichtigt.

Hinweis:
Beachten Sie, dass dieser Vorgang starke Auswirkungen auf die Modellstruktur hat. Vergewissern Sie sich über den Objektreport über die möglichen Auswirkungen.

3.4.10 Diagramme kopieren (flache Kopie)

Sie können Kopien von gesamten Diagrammen anlegen und einfügen. Dabei unterscheidet man zwischen einer flachen Kopie bei der alle Elemente der gezeigten Ebene (Diagramm) kopiert werden und einer tiefen Kopie bei der alle Elemente dieser Ebene und der darunter liegenden Ebenen (Teilhierarchie) kopiert werden. Eine flache Kopie erzeugen Sie über die Menübefehle *Bearbeiten/Diagramm/Diagrammkopie anlegen (flache Kopie)* und *Bearbeiten/Diagramm/Diagrammkopie einfügen (flache Kopie).*

Hinweis:
Beachten Sie, dass das Einfügen der Diagrammkopie nur in einem Diagramm des gleichen Typs und innerhalb des gleichen Projektes möglich ist.

3.4.11 Modelle kopieren (tiefe Kopie)

Sie können Kopien von gesamten Teilmodellen anlegen und einfügen. Dabei unterscheidet man zwischen einer flachen Kopie bei der alle Elemente der gezeigten Ebene (Diagramm) kopiert werden und einer tiefen Kopie bei der alle Elemente dieser Ebene und der darunter liegenden Ebenen (Teilhierarchie) kopiert werden. Eine tiefe Kopie erzeugen Sie über die Menübefehle *Bearbeiten/Diagramm/Diagrammkopie anlegen (tiefe Kopie)* und *Bearbeiten/Diagramm/Diagrammkopie einfügen (tiefe Kopie).*

Hinweis:
Beachten Sie, dass eine tiefe Kopie dem Wesen nach ein Export und Import von Teilmodellen darstellt. Das Einfügen eines solchen Imports in ein anderes Projekt kann zu Modellkonflikten führen.

3.4.12 Die Modellhierarchie bearbeiten (Jo-Jo)

Die Modellhierarchie kann durch Einfügen neuer Ebenen oder Auflösen von Modell-Ebenen verändert werden (Jo-Jo). Diese Funktion erreichen Sie am besten über das Haupt- oder Kontextmenü. Um eine gewünschte Anzahl von Modellelementen zu einer neuen Ebene zusammenzufassen, selektieren Sie diese und wählen im Menü *Bearbeiten/Objekt/Jo-Jo/Objekte zusammenfassen*. Es wird ein neues Objekt erzeugt und in das Diagramm eingefügt. Die ausgewählten Objekte wurden dem neuen Objekt untergeordnet. Bei der Funktion *Jo-Jo/Objekt auflösen* werden die untergeordneten Objekte in das aktuelle Diagramm eingefügt und das ausgewählte Objekt gelöscht.

3.5 Die Ansicht verändern

3.5.1 Fensterinhalt verschieben

Modelle werden oft viel größer als der sichtbare Bereich des Diagrammfensters. Um den sichtbaren Teil eines Diagramms zu verändern (Scroll) haben Sie folgende Möglichkeiten:

- die Scrollleisten des Fensters benutzen
- den Diagrammhintergrund per Drag und Drop schieben
- wenn kein Modellelement ausgewählt ist, mit den Cursortasten scrollen
- Menüpunkt *Ansicht/Fenstermitte* um zum Diagrammursprung zurück zu kehren

Hinweis:
Beachten Sie, dass der Diagrammursprung nicht wie bei einer Textverarbeitung oben links des Arbeitsbereiches liegt sondern wie bei einer MindMap oder vielen Konstruktionsprogrammen (CAD) in der Mitte (Nullpunkt, Fenstermitte) und das Modell sich somit in alle Richtungen entwickeln lässt. Vergleichen Sie dazu die Position der Scrollbalken. Versuchen Sie auf keinen Fall zu Beginn der Arbeit die Scrollposition oben links herzustellen!

3.5.2 Fensterinhalt vergrößern oder verkleinern

Um einen Überblick über größere Diagramme zu gewinnen, können Sie die Darstellung vergrößern oder verkleinern (Zoom). Eine elegante Möglichkeit das aktuelle Diagramm zu zoomen ist das Scrollrad der Maus. Sie können aber auch über das Kontextmenü oder den Menüpunkt *Ansicht* feste Zoom-Faktoren einstellen.
Die Funktion *Ansicht/Zoom/Alles* aus dem Menü oder der Werkzeugleiste passt das Diagramm an die Fenstergröße an. Eine weitere Möglichkeit das Diagramm zu zoomen, erhalten Sie über die Tasten des Numerikfeldes:

- Numerikfeld **+** ... Ansicht vergrößern
- Numerikfeld **-** ... Ansicht verkleinern
- Numerikfeld ***** ... Ansicht alles (einpassen)

3.5.3 Fensterinhalt farbig oder als Kontur darstellen

Temporär kann die farbige Darstellung von Elementen unterbunden werden. Dazu können Sie über das Kontextmenü oder das Menü *Ansicht/Darstellung/Konturen* zwischen farbiger oder Konturdarstellung umschalten.

3.5.4 Die Schriftart ändern

Die in Diagrammen verwendete Schriftart lässt sich über die Werkzeugleiste oder den Menüpunkt *Einstellungen/Schriftart…* verändern. Dabei sollte die Schriftgröße 10 bis 12 eingehalten werden.

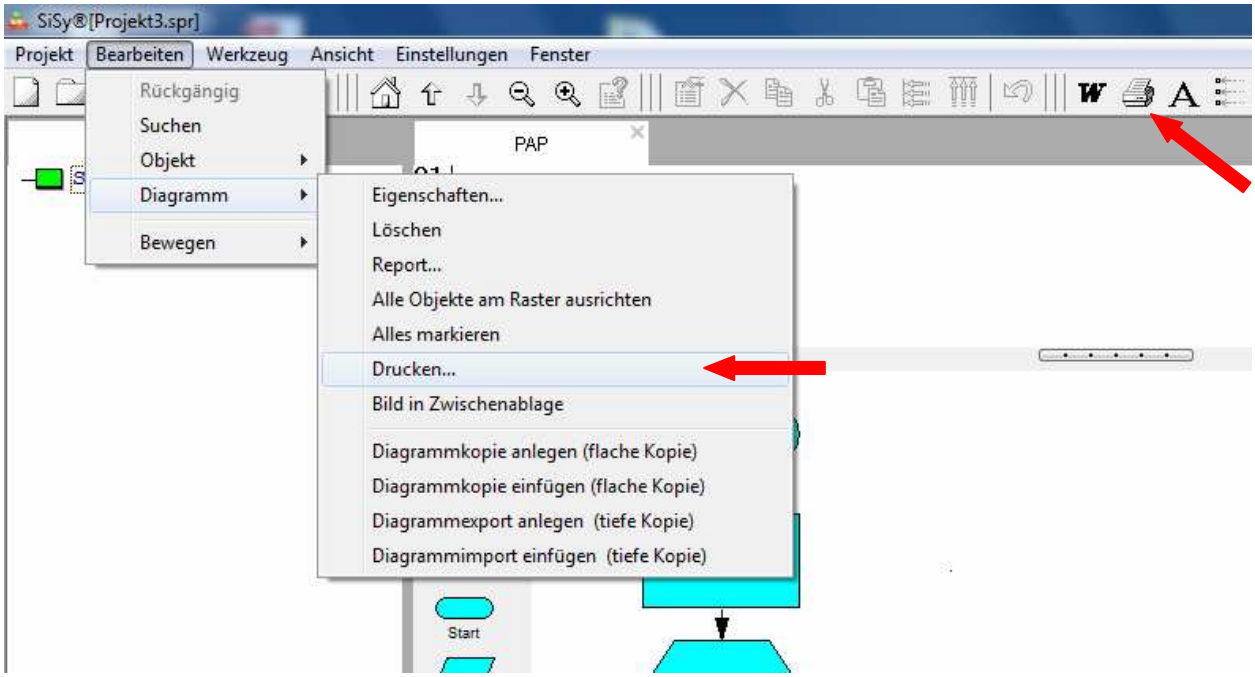
3.6 Druckfunktionen in SiSy

Sie haben in SiSy verschieden Möglichkeiten Projektinhalte wie Grafiken, Übersichten, Quellcodes oder ganze Projektdokumentationen zu drucken. Dabei ist zu berücksichtigen, dass in SiSy bestimmte Informationen wie zum Beispiel die Darstellung eines Programmablaufplanes sichtbare Elemente eines Diagramms sind und andere Teile wie zum Beispiel der Quellcode eines Elementes nur über Dialoge oder bei Selektierung des Elementes sichtbar sind. Je nachdem, welchen Inhalt Sie dokumentieren wollen, richtet sich die Auswahl der betreffenden Druckfunktion.

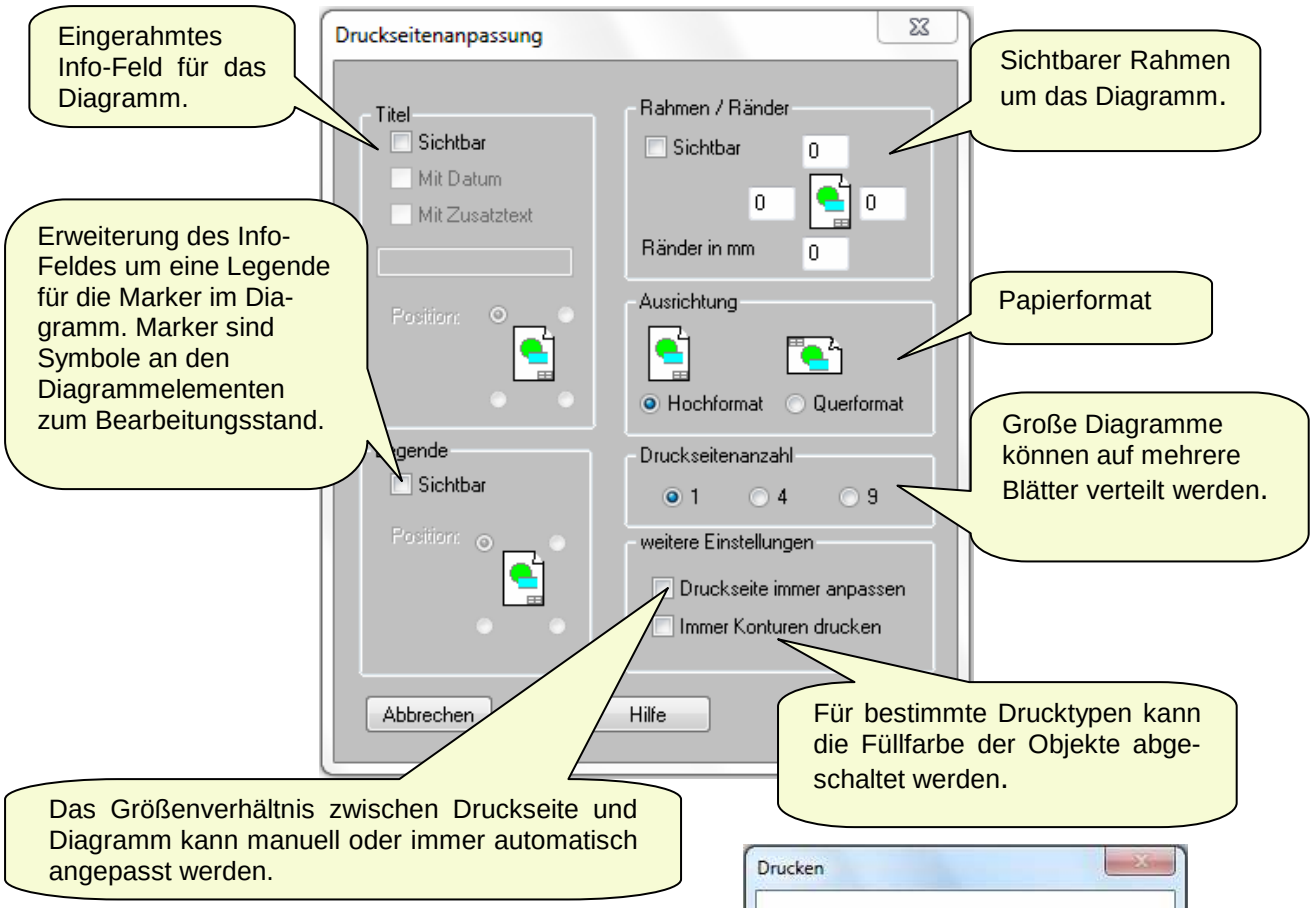
3.6.1 Diagramme drucken

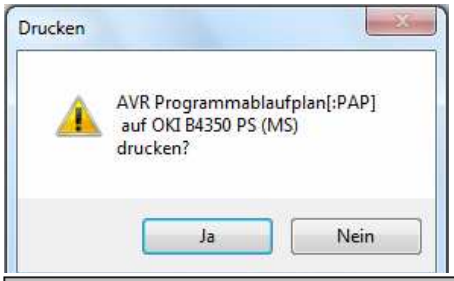
Wenn Sie ein einzelnes Diagramm, also den sichtbaren Inhalt eines Diagrammfensters drucken wollen, gehen Sie wie folgt vor:

- ggf. Projekt öffnen
- das gewünschte Diagramm öffnen
- die Menüfolge *Bearbeiten/Diagramm/Drucken…* wählen oder das Druckersymbol in der Werkzeugleiste aktivieren



- Sie erhalten den Dialog zum Einrichten der Druckseite, wählen Sie die gewünschten Optionen aus!



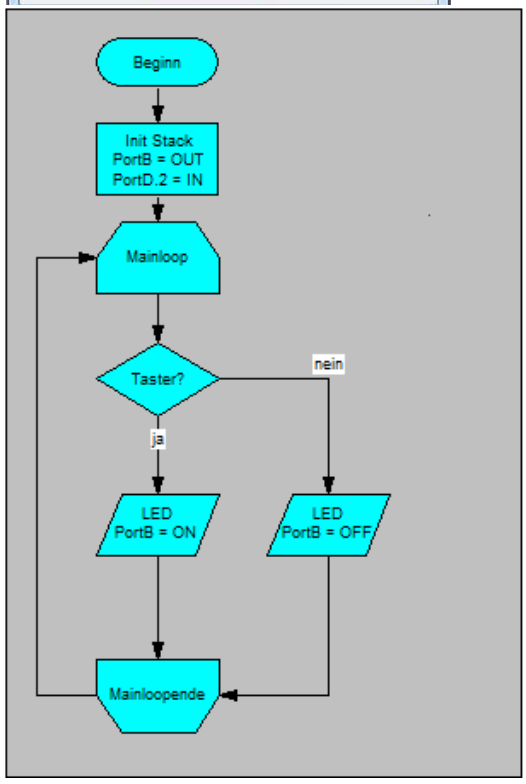


- Zum Drucken wählen Sie die Schaltfläche „OK“. Der Ausdruck erfolgt auf dem Standarddrucker des Systems

- Im Hintergrund ist die Druckvorschau zu sehen. Der Druckvorgang kann hier abgebrochen werden, um die Einstellungen zu überarbeiten. Dabei kann die relative Position und das Größenverhältnis der Druckseite zum Diagramm verändert werden. Der Dialog zum Verändern der Einstellungen lässt sich per Doppelklick auf den Selektierungsmarken der Druckseite öffnen.

- Die Druckseitenansicht lässt sich über die Menüfolge *Ansicht/Druckseite* ein- und ausblenden

- Der Druckvorgang kann über das Druckersymbol in der Werkzeugleiste jederzeit gestartet werden



3.6.2 Grafiken und Inhalte drucken (QuickDok)

Viele Projektinformationen sind kein sichtbarer Teil von erstellten Diagrammen. Diese wurden über Dialoge und Masken eingegeben und stehen als Attribute in der Projektdatenbank zur Verfügung. Um diese Informationen auszudrucken, stellt SiSy für jeden Diagrammtyp eine Reportfunktion zur Verfügung, mit der die wesentlichen Informationen, Inhalte und Attribute des Diagramms und der Objekte in einem Diagramm, als Word-Dokument generiert werden.

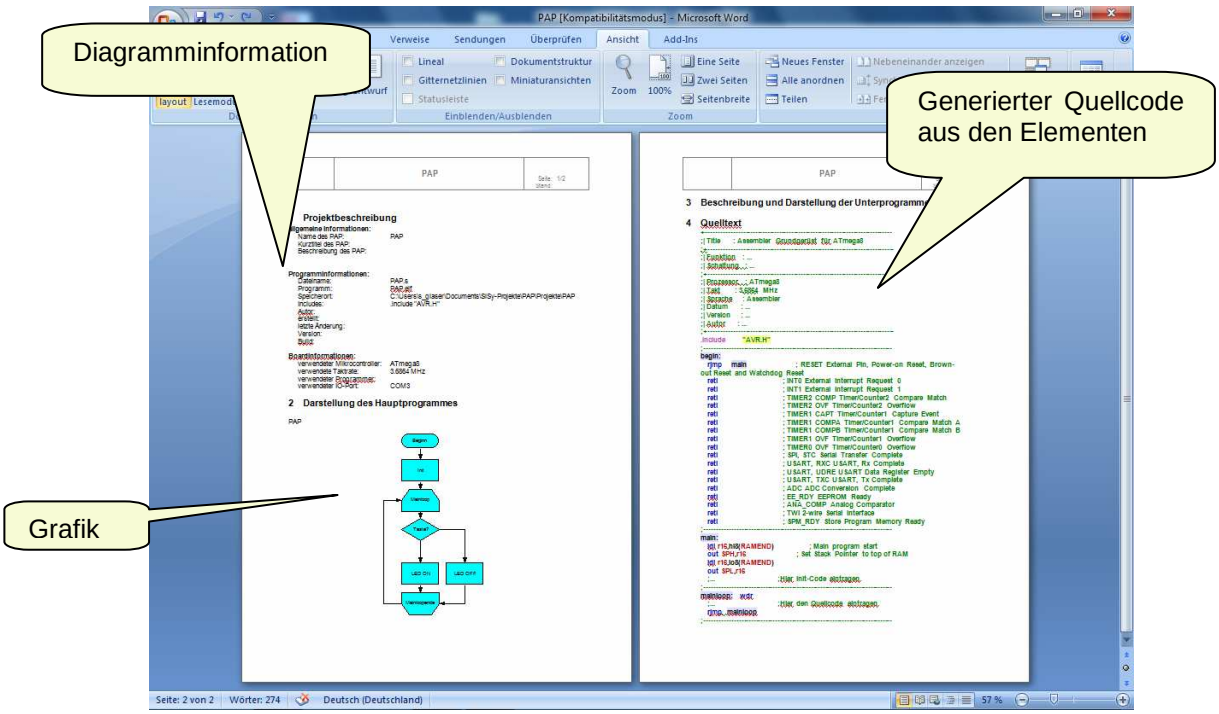
Für das Generieren des Word-Dokumentes muss auf dem PC MS Word installiert sein. Die Dokumentengenerierung basiert auf Makros; unterstützt wird Version Word 2007.
Vorgehensweise in Word:

- Wählen Sie die Schaltfläche „Office“
- Menüfolge: *Word-Optionen / Vertrauensstellungscenter / Einstellungen für das Vertrauensstellungscenter / Einstellungen für Makros*
- Aktivieren Sie unter „Einstellungen für Makros“ den Auswahlpunkt „Alle Makros aktivieren“
- Setzen Sie im Unterpunkt „Entwicklermakroeinstellungen“ den Haken bei „Zugriff auf das VBA-Projektobjektmodell vertrauen“

Hinweis:
Für die Nutzung anderer Word-Versionen finden Sie Hinweise in unseren FAQs (www.sisy.de → SiSy-Wiki → weiteres →FAQ → DokGen)

Die Reportfunktion zum Generieren des Word-Dokumentes aktivieren Sie über das Symbol „QuickDok“ in der Werkzeugleiste.

- Symbol  (QuickDok) in der Werkzeugleiste wählen
- Aus der Liste der angebotenen Reportfunktionen auswählen
- Dateinamen für den Report vergeben
- Makros aktivieren

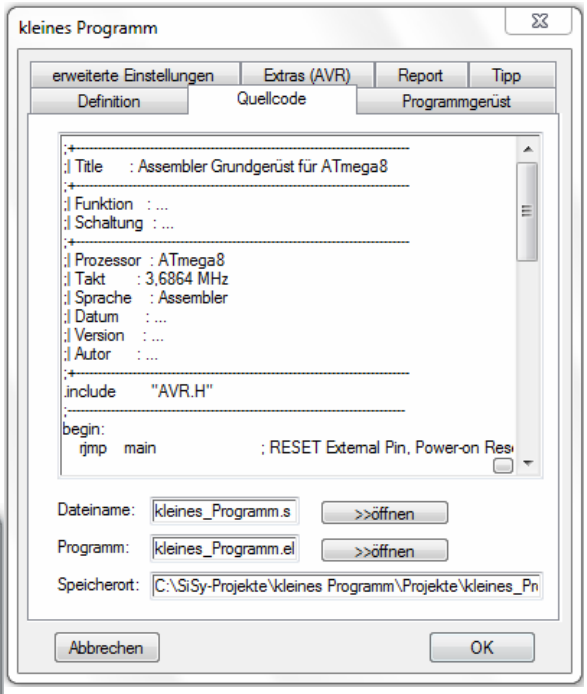


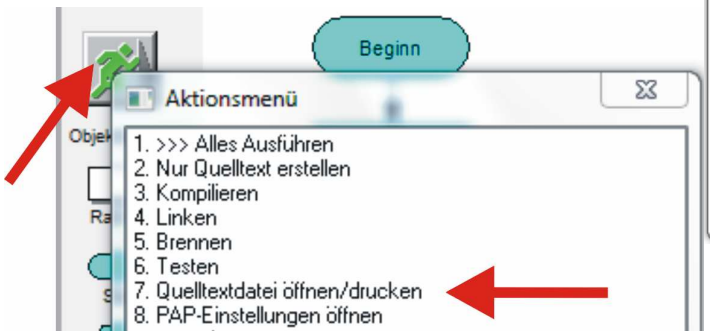
3.6.3 Nur Quellcodes drucken

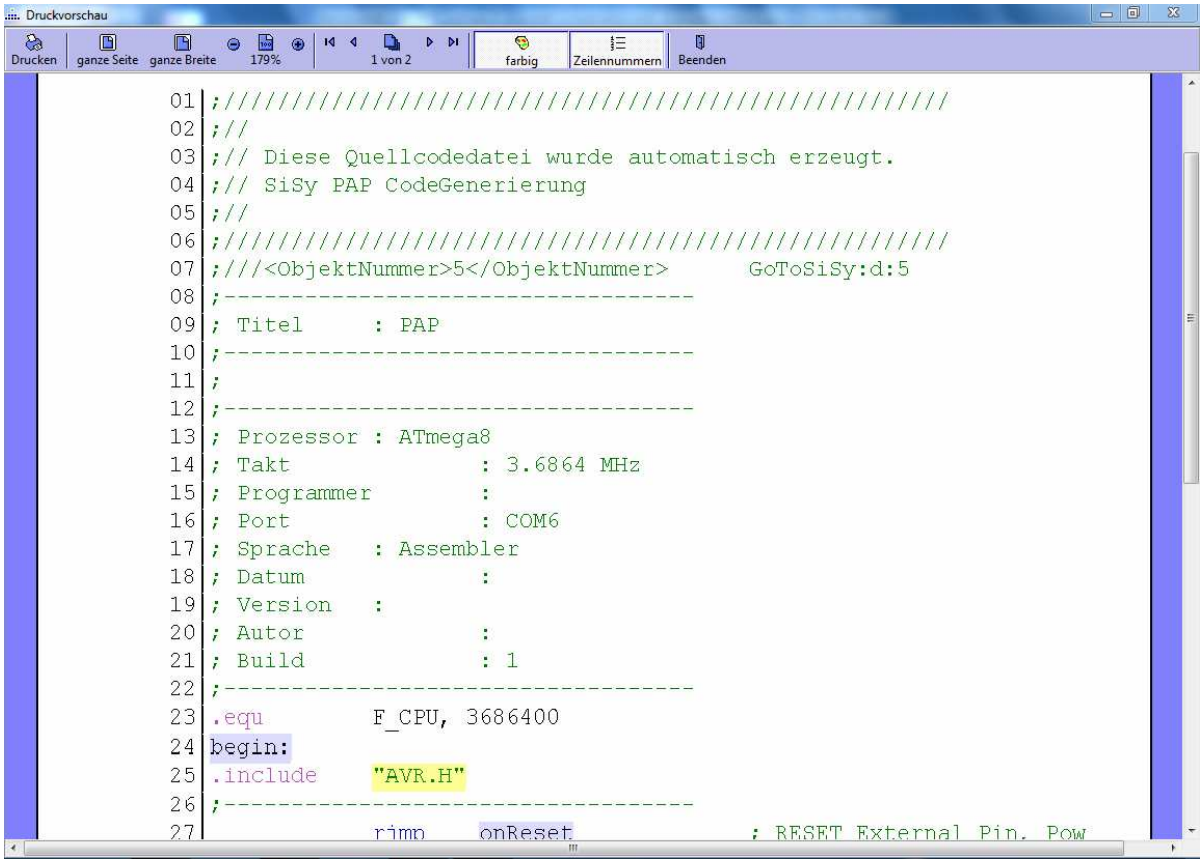
Für das Ausdrucken von Quellcodes bietet SiSy einen speziellen Quellcode-Druckassistenten. Damit ist es möglich Quellcodes formatiert, in Syntaxfarben und mit Zeilennummern auszudrucken.

- kleines Programm, Quellcode drucken
  - kleines Programm selektieren
  - rechte Maustaste „Definieren…“
  - Dialogseite „Quellcode“
  - Schaltfläche „>>öffnen“

- Programmablaufplan, Quellcode drucken
  - PAP öffnen
  - Aktionsmenü aktivieren
  - „Quellcodedatei öffnen/drucken“



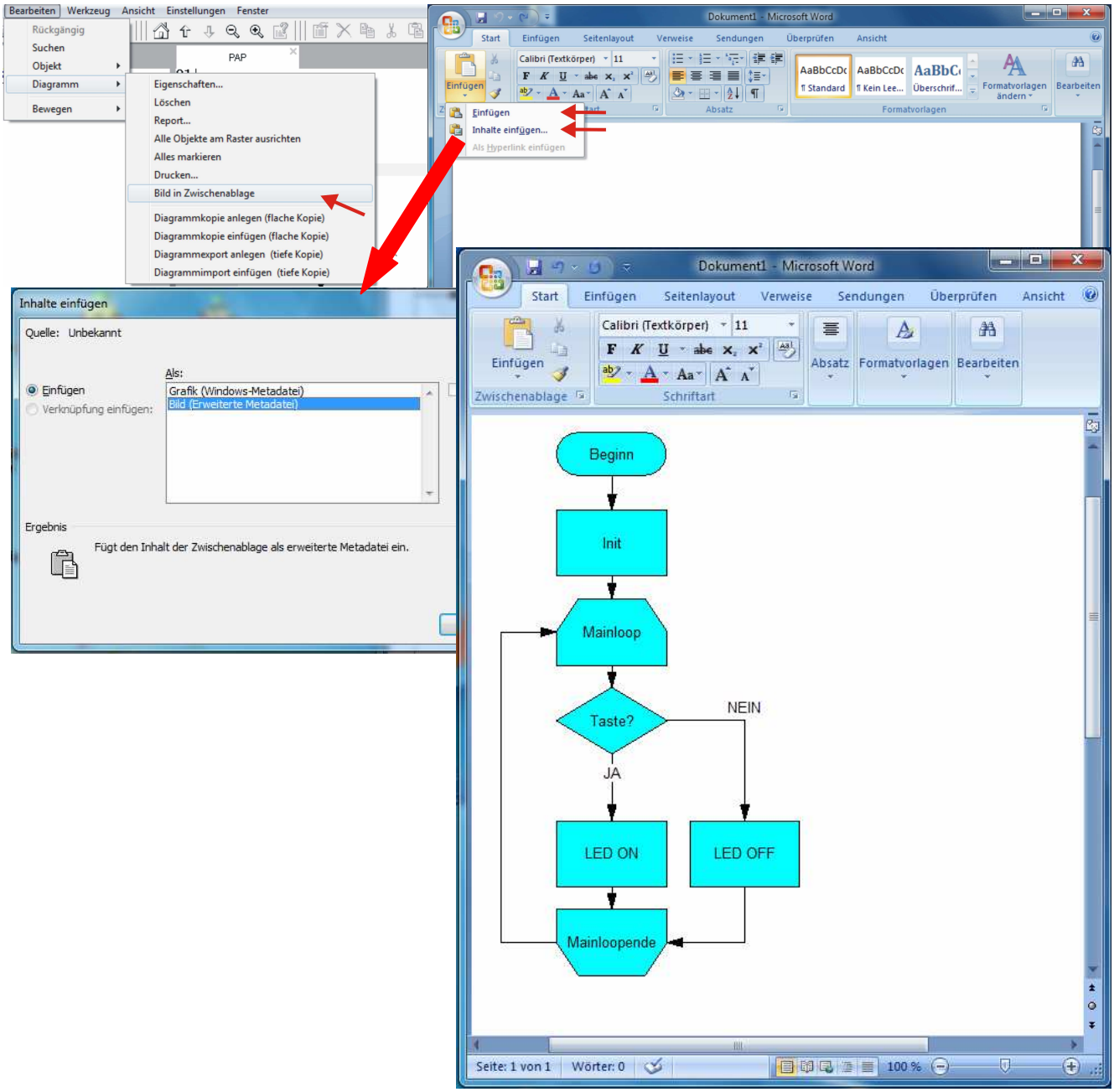




3.6.4 Nutzen der Zwischenablage

Oft ist es erforderlich in Projektdokumentationen die Diagramme als Bilder einzufügen. In SiSy werden die Diagramme nicht als Bilder gespeichert sondern zur Laufzeit aus den Modellinformationen generiert. Um die Bilder der Diagramme weiter zu verwenden, steht dem Anwender die Funktion „Bild in Zwischenablage“ zur Verfügung. Dabei erstellt SiSy eine skalierbare Vektorgrafik (WMF) und legt diese in die Zwischenablage (Copy). Die Grafik kann nun von anderen Anwendungen über den Befehl „Einfügen“ (Paste) beliebig weiter verwendet werden.

- gewünschtes Diagramm öffnen
- Menüfolge *Bearbeiten/Diagramm/Bild in Zwischenablage* wählen
- Zielanwendung, zum Beispiel Word öffnen
- Menüfolge *Start/Einfügen* oder *Start/Einfügen/Inhalte einfügen* wählen
- Gegebenenfalls einzufügendes Grafikformat wählen



3.7 SiSy Add-Ons verwalten

3.7.1 Einführung

Das Modellierungswerkzeug SiSy besteht aus seinen Kernkomponenten:

- Anwendungssystem SiSy mit
  - Rahmenanwendung
  - Assistent
  - Diagrammfenster
- Laufzeitbibliotheken
  - Datenbanktreiber (Repository)
  - Grafikbibliotheken
  - Metamodell-Engine
  - SiSy BASIC Interpreter
- Allgemeine Hilfe (dieses Benutzerhandbuch)

… und installierten Add-Ons.

Add-Ons enthalten zusätzliche Komponenten, SiSy-BASIC-Skripte, erweiterte Hilfen und spezielle Informationen für die Modellierung (Metamodell). Dabei können Add-Ons wiederum aus Add-Ons thematisch zusammengestellt werden. Diese zusätzlichen Bausteine werden während der Laufzeit vor allem durch die Metamodell-Engine verarbeitet. Dadurch können beliebige Modellierungsaufgaben durch SiSy verarbeitet werden. Es ist nur das entsprechende Add-On erforderlich. Für die aktuelle Version von SiSy sind zum Beispiel folgende Add-On Zusammenstellungen verfügbar:

- **AVR**: Programmierung von AVR Mikrocontrollern
  - einfache Programmierung mit dem Zeileneditor in Assembler und C
  - grafische Programmierung mit dem Programmablaufplan in Assembler, dem Struktogramm in C, dem UML-Klassendiagramm in C++ und dem UML-State chart in C++
  - Codegenerierung mit dem myAVR Code-Wizard für Assembler und C
  - myAVR Werkzeuge für Test und Kommunikation mit AVR Controllern
- **ARM:** Programmierung von ARM Mikrocontrollern
  - einfache Programmierung mit dem Zeileneditor in C
  - grafische Programmierung mit dem Struktogramm in C, dem UML-Klassendiagramm in C++ und dem UML-Statechart in C++
  - Integrierter Debugger für ARM Mikrocontroller
- **SPRG**: einfache Programmierung für Konsolen- und GUI-Anwendungen
  - Integrierte Programmiersprachen C/C++, JAVA, C#, PASCAL, Assembler
  - Integrierter Debugger
- **UML**: objektorientierte Systementwicklung für Konsolen- und GUI-Anwendungen
  - Typische UML-Diagramme z.B.: Klassendiagramm, Anwendungsfalldiagramm, Aktivitätsdiagramm, Zustandsdiagramm, Sequenzdiagramm, u.a.
  - Codegenerierung für C++, C#, JAVA
  - Integrierter Debugger mit Modellnachführung im Schrittbetrieb
- **SVL**: Smart Visual Library
  - Einfache bausteinartige Entwicklung von Windows-Anwendungen (RAD Rapid Application Development) im UML Klassendiagramm
- **SysML**: System Modeling Language
  - Fachübergreifende Systementwicklung mit dem OMG Sprachstandard
- **SA/SD**: Strukturierte Techniken zur Systementwicklung
  - Round Trip Engineering mit den klassischen Analyse und Entwurfswerkzeugen bis zur Codegenerierung, Strukturierte Analyse, Entity Relationship Diagramm mit SQL-Generierung und Struktogramme mit Codegenerierung
- **DOKGEN**: Dokumentengenerierung
  - Generieren von Dokumentationen direkt aus den Modellen heraus.
- **GPM / QM**: Geschäftsprozessmodellierung und Qualitätsmanagement
  - Modellierung von integrierten Managementsystemen

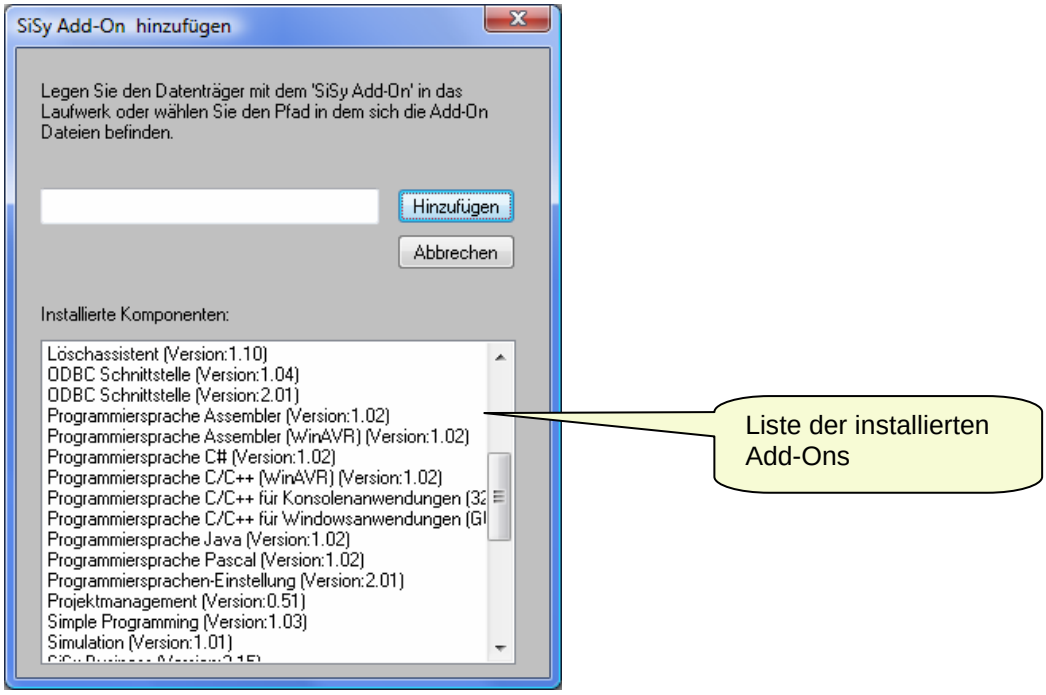
Aus den verfügbaren Add-Ons werden entsprechend der unterschiedlichen Einsatzgebiete des Modellierungswerkzeuges SiSy spezielle Ausgaben zusammengestellt. Die jeweiligen Ausgaben können jederzeit durch weitere Add-Ons ergänzt werden. Es sind unter anderem folgende SiSy-Ausgaben verfügbar (Stand Februar 2013):

Ausgabe (Auswahl)

- SiSy Professional
  umfasst alle verfügbaren Add-Ons
- SiSy Business
  umfasst Add-Ons für Prozessmodellierung, Qualitäts- und Projektmanagement
- SiSy Developer
  umfasst umfangreiche Add-Ons zur Systementwicklung
- SiSy Microcontroller ++
  umfasst die Add-Ons AVR, ARM, ausgewählte Teile der UML, SVL und SysML
- SiSy AVR
  beinhaltet nur das Add-On AVR
- SiSy ARM
  umfasst die Add-Ons ARM, ausgewählte Teile der UML und SysML

3.7.2 Add-Ons anzeigen

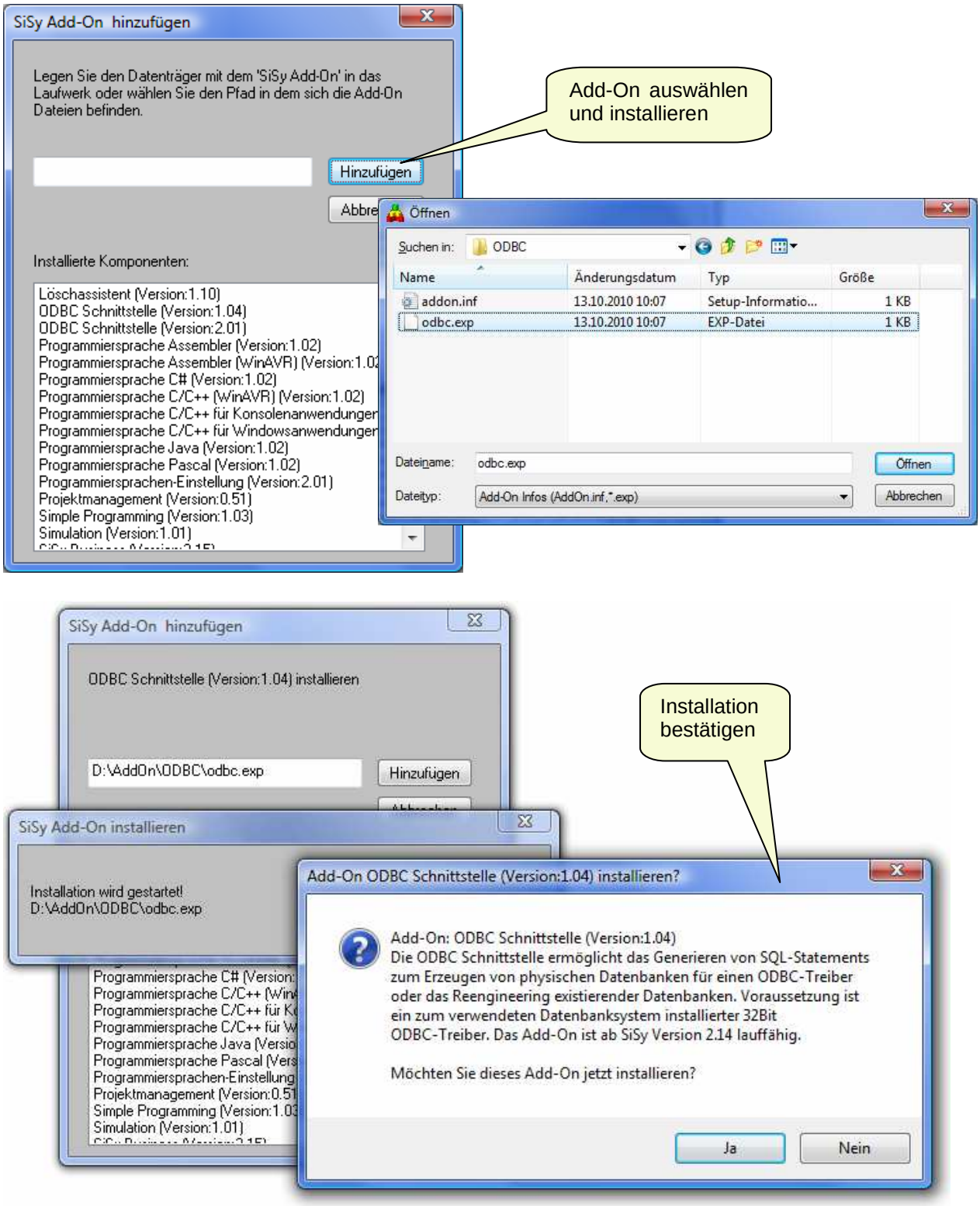
Für die Anzeige der installierten Add-Ons nutzen Sie den Menüpunkt *Einstellungen/ Add-On hinzufügen*.



Hinweis:
Beachten Sie, Änderungen an den Add-Ons haben in jedem Fall Auswirkungen auf die Funktionalität von SiSy und abhängiger Add-Ons.

3.7.3 Add-Ons hinzufügen

Für das Hinzufügen von Add-Ons nutzen Sie den Menüpunkt *Einstellungen/Add-On hinzufügen*. Wählen Sie im Add-On Dialog die Schaltfläche „Hinzufügen“. SiSy Add-Ons besitzen die Dateierweiterung *.exp.



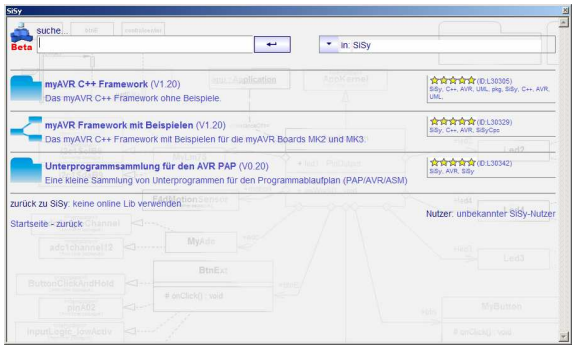
<u>Hinweis:</u>
Für die Installation von Add-Ons sollten Sie Administratorrechte besitzen. Beachten Sie, dass Änderungen an den Add-Ons in jedem Fall Auswirkungen auf die Funktionalität von SiSy und abhängiger Add-Ons haben.

3.8 LibStore

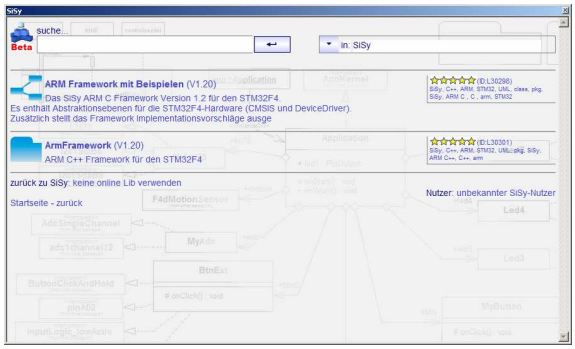
LibStore ist eine online-Sammlung von Vorlagen, Mustern, Beispielen und Bibliotheken. Diese werden Ihnen bei der Arbeit mit SiSy angeboten, sobald bei der Modellierung im jeweiligen Diagramm LibStore verfügbar ist und Sie online sind.
Die Nutzung der im SiSy-LibStore zur Verfügung gestellten Inhalte ist, wenn durch den Autor nicht anderweitig geregelt, an die Lizenzbedingungen Ihrer SiSy-Version gebunden.

3.8.1 Handhabung: ein Projekt mit LibStore anlegen

Nach dem Start von SiSy legen Sie ein neues Projekt an und wählen ein Vorgehensmodell aus. Es öffnet bereits LibStore und bietet, soweit für dieses VGM vorhanden, eine Auswahl von Vorlagen an.

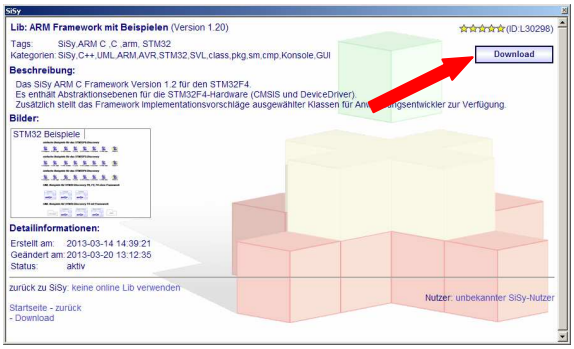


LibStore aus AVR-Vorgehensmodell



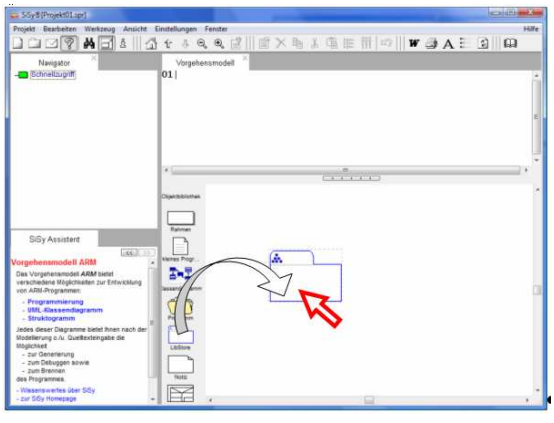
LibStore aus ARM-Vorgehensmodell

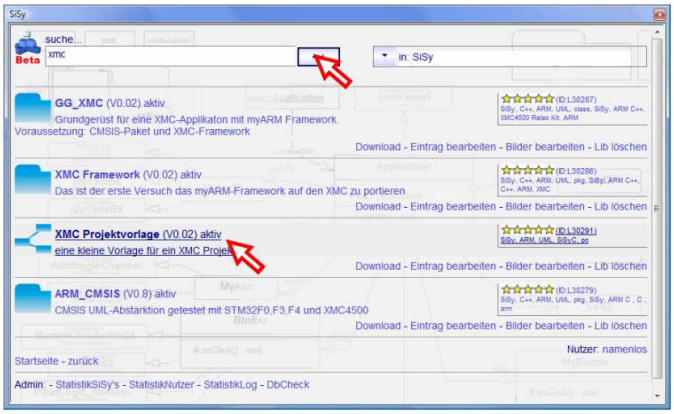
Aktivieren Sie Ihre gewünschte Vorlage und aktivieren Sie im nächsten Fenster die Schaltfläche „Download“. Die ausgewählte Vorlage wird in ein leeres Diagramm geladen; sie steht Ihnen uneingeschränkt zur weiteren Arbeit zur Verfügung.



3.8.2 Handhabung: LibStore in einem Projekt nutzen

Aus der Objektbibliothek ziehen Sie ein neues Objekt vom Typ „LibStore“ in das geöffnete Diagramm. In Abhängigkeit von den verfügbaren Add-Ons in Ihrer SiSy-Ausgabe und dem Diagrammtyp werden Ihnen von LibStore Vorlagen, Beispiele, usw. angeboten. Zur Einschränkung können Sie in der Suchzeile Begriffe eingeben, wie z.B. Controllernamen, dann Ihre Auswahl treffen und den Download ausführen.





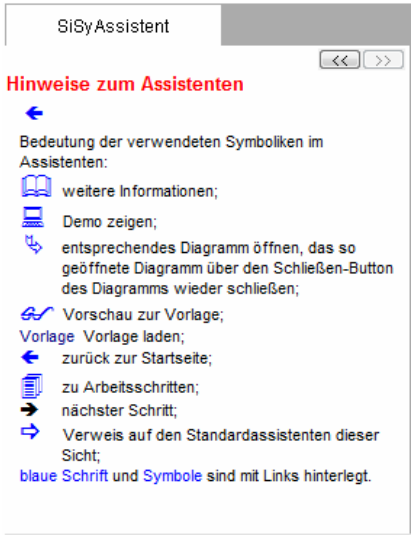
3.9 Die Hilfefunktionen in SiSy

Nutzen Sie die zahlreichen Hilfen und Vorlagen, die SiSy dem Entwickler bietet!

3.9.1 Der Assistent

Der Assistent ist hilfreich bei der Unterstützung und Führung des Nutzers im Umgang mit SiSy. Er befindet sich standardmäßig im linken, unteren Bildschirmbereich. Der Assistent kann über die Werkzeugleiste (-Symbol) geöffnet werden, falls dies nicht beim Programmstart erfolgte. Der Assistent begleitet Sie im gesamten Projekt. Sie erhalten immer passende Informationen zum aktuellen Diagrammtyp und haben die Möglichkeit, durch verschiedene Links weitere Hilfethemen aufzurufen oder Vorlagen in Ihr Projekt zu laden.

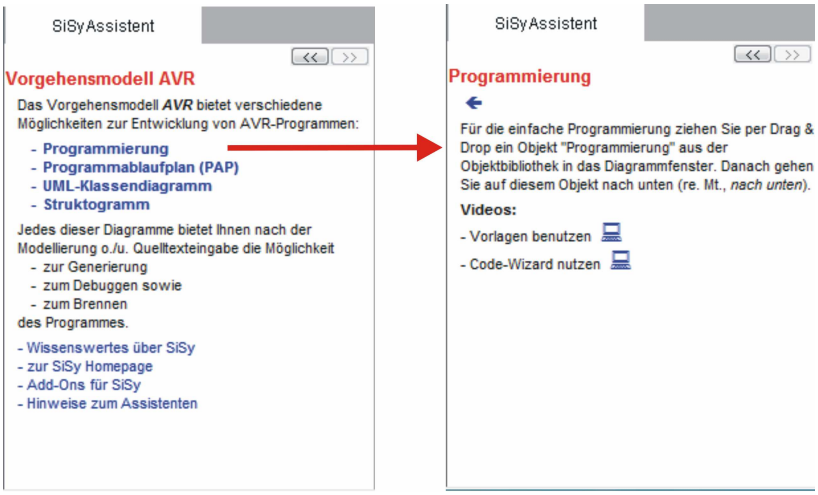
Beachte: Der Assistent ist auf die jeweilige Ausgabe von SiSy, die verfügbaren Add-Ons und das gewählte Modell bezogen.



Bedeutung der verwendeten Symboliken im Assistenten:

	weitere Informationen; öffnet eine Hilfedatei (*.chm, *.hlp, *htm)
	Demovideo zur Handhabung zeigen; öffnet eine Animation oder Videomitschnitt der Bildschirmarbeit (AVI, ScreenCam- oder FLASH-Film)
	entsprechendes Diagramm öffnen, das so geöffnete Diagramm kann über die Schließen-Schaltfläche des Diagramms wieder geschlossen werden;
Beispiel 1	eine Diagrammvorlage laden;
	Vorschau zur Diagrammvorlage;
←	zurück zur Startseite des Assistenten;
	ein kleines Skript zu Arbeitsschritten anzeigen;
➔	nächster Schritt (Arbeitsschritt);
⇨	Verweis auf den Standardassistenten dieser Sicht;
blaue Schrift	und
blaue Symbole	sind mit Links hinterlegt und können per Mausklick aktiviert werden.

Beispiele für Assistenten:

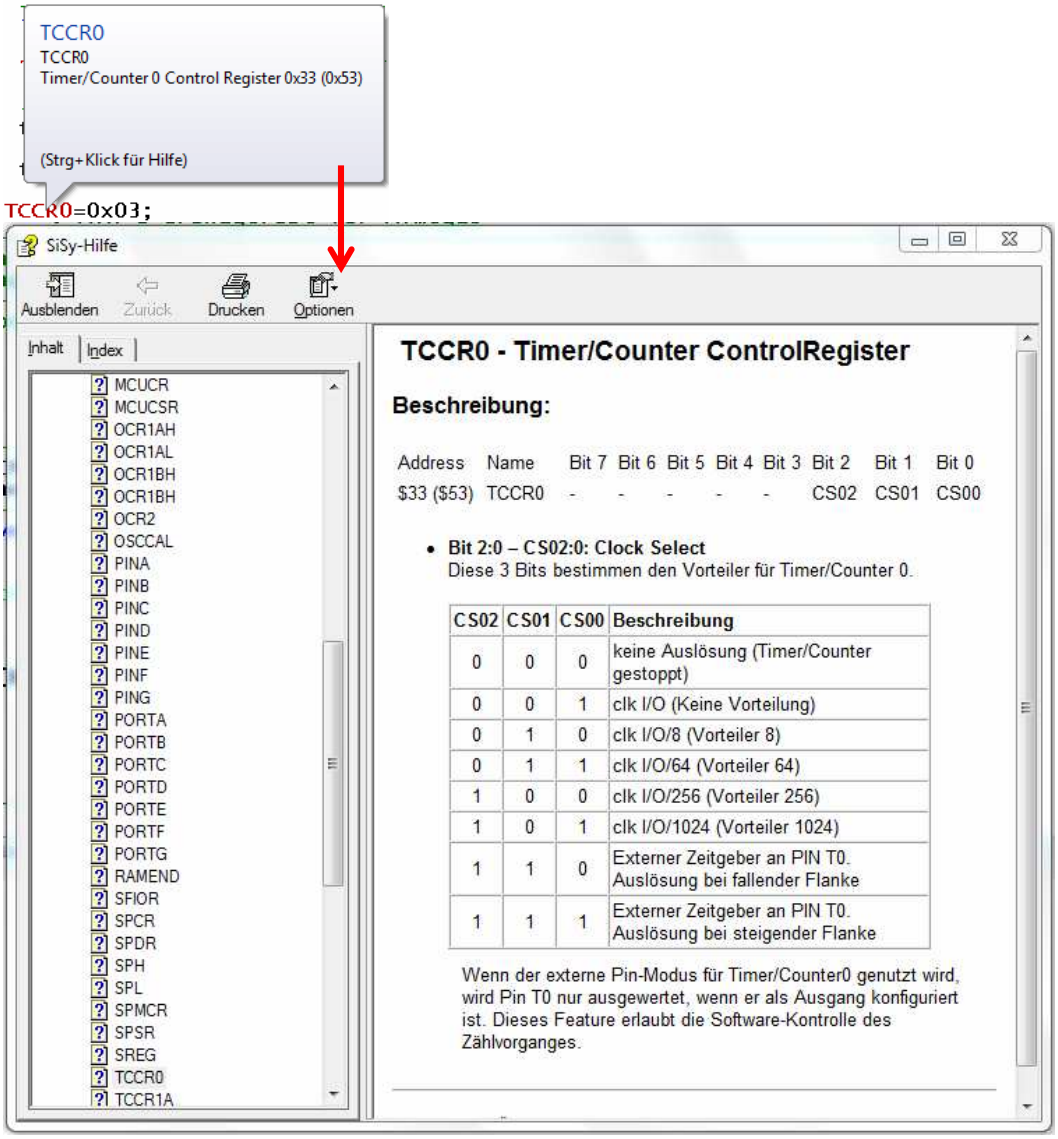


3.9.2 Die Online-Hilfe

Bei der Eingabe von Quellcode im dafür vorgesehenen Editorfenster werden reservierte Worte (Bezeichner, Schlüsselworte) der gewählten Programmiersprache durch verschiedenfarbiges Einfärben (Syntaxfarben) hervorgehoben. Zu den hervorgehobenen Bezeichnern existiert in der Regel auch eine kurze Online-Hilfe und eine ausführlichere Hilfe. Die Online-Hilfe ist ein Pop-Up-Fenster, welches automatisch eingeblendet wird, wenn Sie mit dem Mauszeiger auf einen Befehl zeigen. In dem Pop-Up ist eine kurze Hilfestellung zu dem Bezeichner eingeblendet. Steht eine Hilfe mit ausführlicheren Informationen zur Verfügung, wird diese in dem Pop-Up (STRG+Klick für Hilfe) angezeigt.

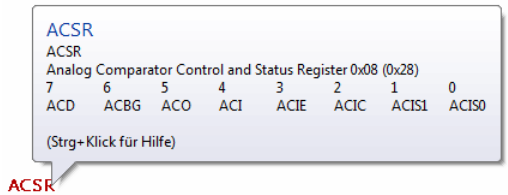
Schlüsselwort-Hilfe

Bei der Eingabe von bekannten Registernamen wird die Bezeichnung des Registers und dessen Adresse eingeblendet. Durch betätigen der Taste „STRG“ und gleichzeitigem Klick auf das Register öffnet eine Hilfedatei mit dem entsprechenden Hilfethema zu dem eingegeben Register.



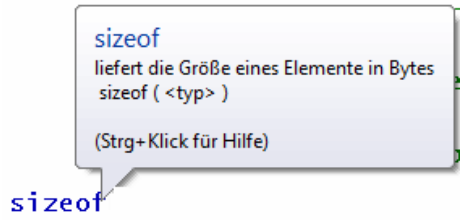
Bit-Hilfe

Wenn für ein Register detaillierte Informationen zu der Bedeutung/Funktion der einzelnen Bits vorliegen, wird zusätzlich im Pop-Up eine Kurzreferenz der Bits angezeigt. Durch drücken der Taste „STRG“ sowie Klick auf das Register öffnet SiSy eine Hilfedatei mit dem entsprechenden ausführlichen Hilfethema.



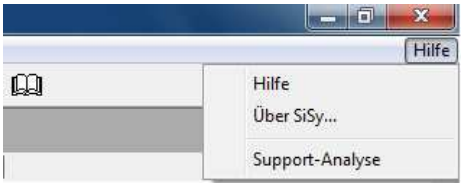
Befehlshilfe

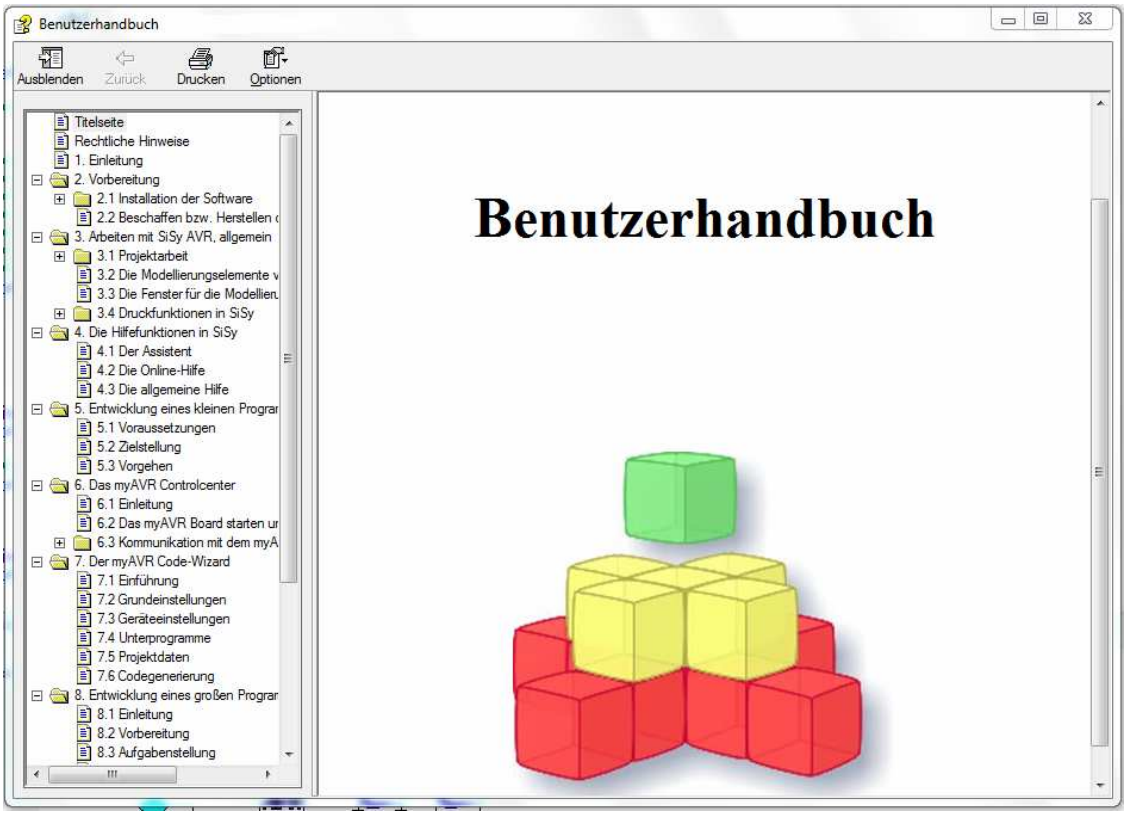
Bei der Eingabe von Befehlen wird in der Regel die Bedeutung bzw. Funktion des Befehls und ein kurzes Syntaxbeispiel eingeblendet. Durch drücken der Taste „STRG“ sowie Klick auf den Befehl öffnet SiSy eine Hilfedatei mit dem entsprechenden ausführlichen Hilfethema.



3.9.3 Die allgemeine Hilfe

SiSy bietet neben der direkten Hilfe bei der Eingabe von Schlüsselworten auch eine allgemeine Hilfe an.

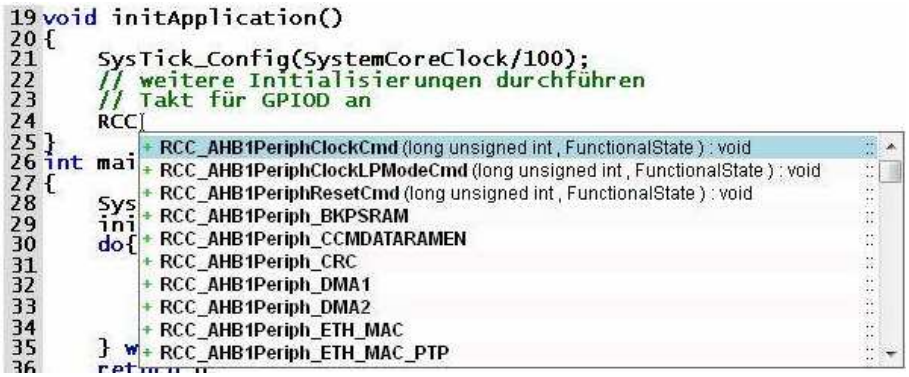




3.9.4 SiSy Code Vervollständigung

Der Codegenerator ist eine integrierte Hilfe in SiSy. Vorteilhaft erweist sich diese Hilfe bei der Generierung des Quellcodes aus einem Programmablaufplan sowie bei der Arbeit mit Klassendiagrammen. Des Weiteren fungiert er als Assistent zum Erstellen von Assembler- und C-Codes für die Programmierung von Mikrocontrollern, was die fehlerhafte Codeeingabe minimiert.

Bei der *Programmierung* springt die Codevervollständigung nach der Eingabe von drei zusammenhängenden Buchstaben an. Aus der angezeigten Liste kann der gewünschte Befehl selektiert werden.



Bedingungen im *Programmablaufplan* haben spezielle Vorlagen. Diese sind so konstruiert, dass eine JA/NEIN Entscheidung erzeugt werden kann. Findet der Codegenerator das Schlüsselwort JA oder NEIN an einer der folgenden Verbindungen, setzt er diese in eine entsprechende Sprunganweisung um.

Im *UML Klassendiagramm* erzeugt der Codegenerator eine Instanz der Klasse und ruft die Main-Methode auf. Das Zustandsdiagramm mit Quellcodegenerierung erhalten Sie über ein spezielles Attribut im Klassendiagramm der UML. Dieses Zustandsattribut kann mit einem Zustandsdiagramm verfeinert werden. Dabei wird der Quellcode in spezielle Klassenmethoden generiert.

In den entsprechenden Kapiteln zur Programmierung, zum Programmablaufplan, zum Klassendiagramm und zum myAVR Code-Wizard wird explizit auf die Code-Vervollständigung eingegangen.

4 Entwicklung eines kleinen Programms

SiSy bietet die Möglichkeit, Programme in einem konventionellen Zeileneditor zu erstellen und zu übersetzen. Für den Einstieg bietet sich die einfache Programmierung an, ein „kleines Programm“.

4.1 Vorgehen für PC Programme

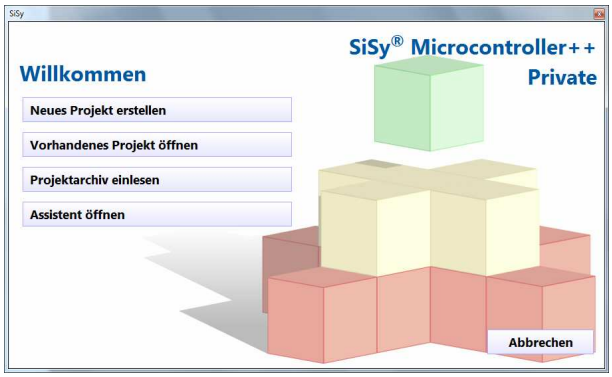
Hinweis:
Dieses Beispiel ist erstellt mit der Ausgabe SiSy Microcontroller ++

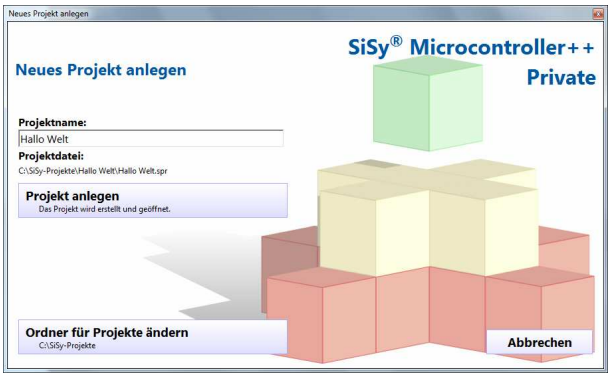
Zielstellung

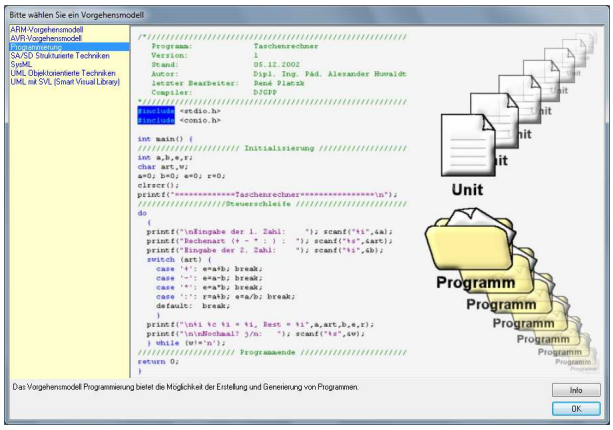
Die erste Aufgabe ist das berühmte „Hallo Welt“, programmiert in C, mit dem jeder Programmierer angefangen hat. Dieser Text soll auf dem Bildschirm ausgegeben werden.

Ein neues Projekt anlegen

Starten Sie SiSy und aktivieren Sie „Assistent öffnen“. Im SiSy-Assistent folgen Sie dem Link „Neues Projekt anlegen“ und vergeben den Projektnamen „Hallo_Welt“. Bestätigen Sie diesen Dialog mit „Projekt anlegen“.
Wählen Sie das Vorgehensmodell „Programmierung“ und bestätigen Sie mit „OK“.



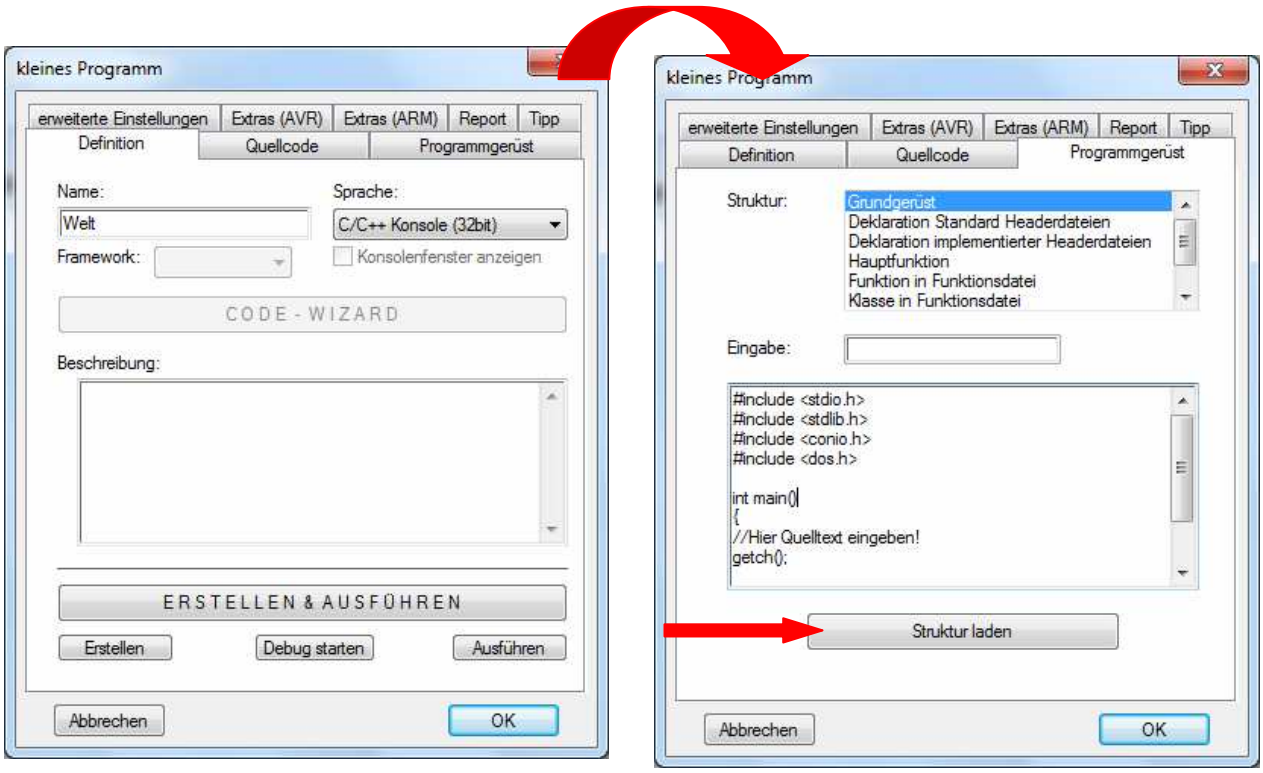




Hinweis:
In SiSy legen Sie stets ein Projekt an. In dieses Projekt integrieren Sie Ihr Programm bzw. mehrere Programme. Unabhängig vom Programmnamen benötigt jedes Projekt einen Namen. Beachten Sie die Erläuterungen im Kapitel 3.1.2.

Das folgende Dialogfeld mit den Informationen zur Hardware-Konfiguration brechen Sie ab; danach bestätigen Sie die voreingestellte Auswahl „leeres Diagramm“.

Ziehen Sie per Drag & Drop ein Objekt "kleines Programm" in das Diagrammfenster. Es öffnet ein Dialogfenster; oder klicken Sie mit der rechten Maustaste auf das Objekt und wählen im Kontextmenü „Definieren“. Vergeben Sie den Namen „Welt“ und wählen die Sprache „C/C++ Konsole (32 Bit)“ aus. Wechseln Sie zur Registerkarte „Programm gerüst“ und laden die Grundstruktur eines C-Programms (Struktur laden). Schließen Sie den Dialog mit „OK“.



Geben Sie im Quellcodefenster den folgenden Programmcode für „Hallo Welt“ ein:

```
printf ("Hallo Welt!");
```

Klicken Sie nacheinander auf „Kompilieren“, „Linken“, „Ausführen“ oder nur auf „>>>Ausführen“. Nun sollte sich ein DOS Fenster öffnen mit der Ausgabe "Hallo Welt". Ist dies nicht der Fall und Sie erhalten eine Fehlermeldung, prüfen Sie bitte die Schreibweise und beachten Sie den Syntaxfehler.

4.2 Vorgehen für AVR Programme

Hinweis:
Dieses Beispiel ist erstellt mit der Ausgabe SiSy AVR.

Zielstellung

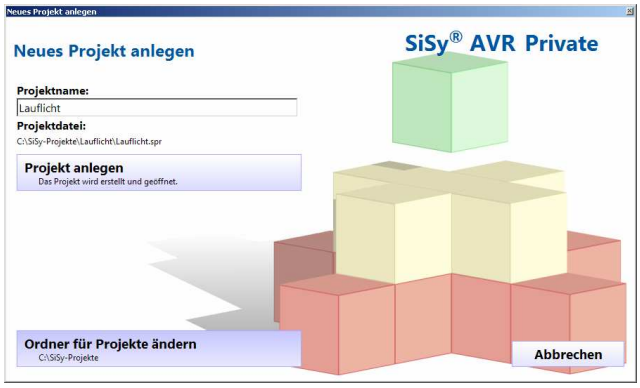
In dem ersten AVR Beispielprogramm sollen die drei LED's auf dem myAVR Board nacheinander aufleuchten und damit ein „Lauflicht" erzeugen. Die Programmiersprache ist Assembler.

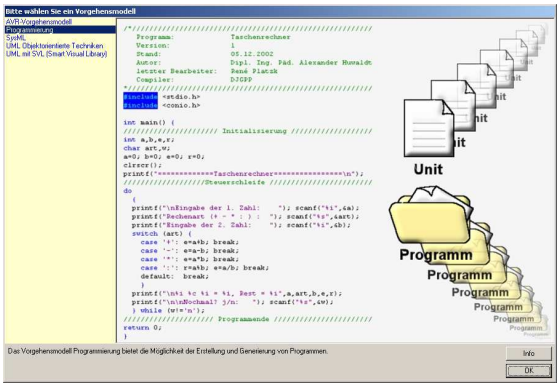
Aufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der auf dem myAVR Board die 3 LEDs nacheinander aufleuchten.

Schaltung:
Port D.2 = rote LED
Port D.3 = grüne LED
Port D.4 = gelbe LED

Ein neues Projekt anlegen

Starten Sie SiSy und wählen Sie „Assistent öffnen". Aktivieren Sie im SiSy-Assistent den Menüpunkt „Neues Projekt anlegen" und vergeben Sie den Projektnamen „Lauflicht". Bestätigen Sie diesen Dialog mit „Projekt anlegen". Wählen Sie das Vorgehensmodell „Programmierung" und bestätigen Sie mit „OK".

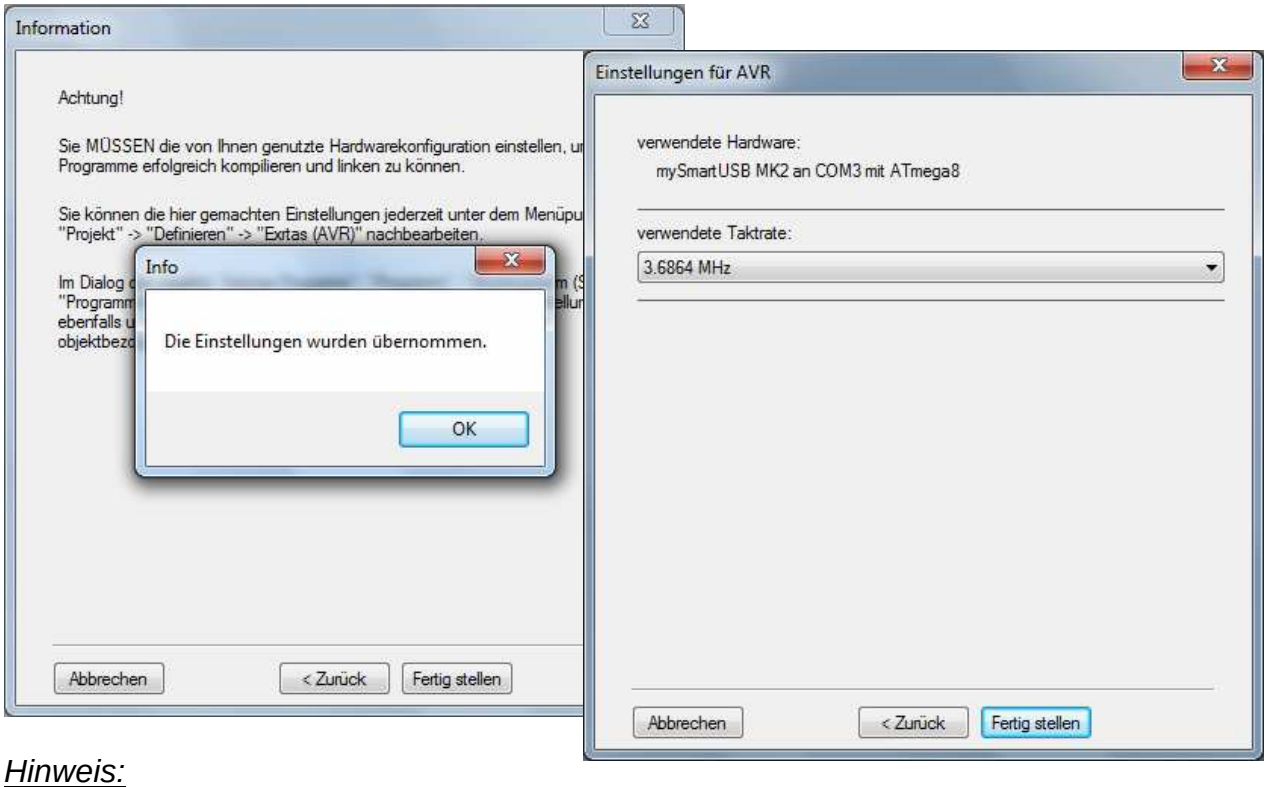




Hinweis:
In SiSy legen Sie stets ein Projekt an. In dieses Projekt integrieren Sie Ihr Programm bzw. mehrere Programme. Unabhängig vom Programmnamen benötigt jedes Projekt einen Namen. Beachten Sie die Erläuterungen im Kapitel 3.1.2.

Es folgt ein Info-Fenster, welches Sie mit „Weiter" bestätigen. Als nächstes wählen Sie Ihren Programmer und Ihren Controller aus; speichern Sie die Auswahl.

Den folgenden Dialog mit der Info „Die Einstellungen wurden übernommen“ bestätigen Sie. Dann wählen Sie die verwendete Taktrate Ihres Mikrocontrollers aus; klicken Sie auf „Fertig stellen“.



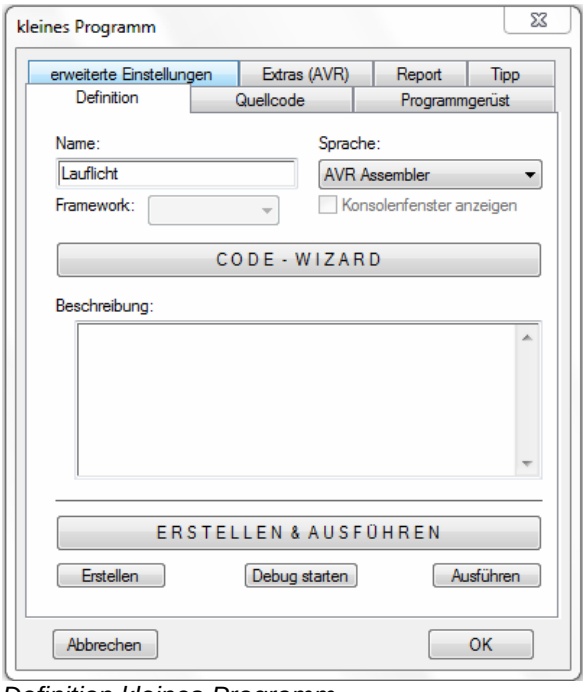
Hinweis:
Bei jedem neuen Projekt müssen die Grundeinstellungen zur verwendeten Zielplattform vorgenommen werden (Mikrocontrollertyp, Taktrate, Programmer und I/O-Port).
Werden keine Einstellungen vorgenommen, so geht SiSy von einem mySmartUSB MK2 Programmer an COM 3 aus. Als Mikrocontroller wird ein Atmega8 mit 3,6864 MHz verwendet.

Im nächsten Fenster wählen Sie „leeres Diagramm“ aus. Beenden Sie das Fenster über „Weiter“ sowie „Fertig stellen“.

Kleines Assembler-Programm anlegen

Erstellen Sie ein Programm für den AVR-Mikrocontroller, indem Sie per Drag & Drop aus der Objektbibliothek ein Objekt „kleines Programm“ in das Diagramm fenster ziehen. Das Kontextmenü öffnet sich automatisch.
Auf der Registerkarte „Definition“ tragen Sie den Programmnamen ein (im Beispiel „Lauflicht“) und wählen die Programmiersprache „AVR Assembler“ aus.

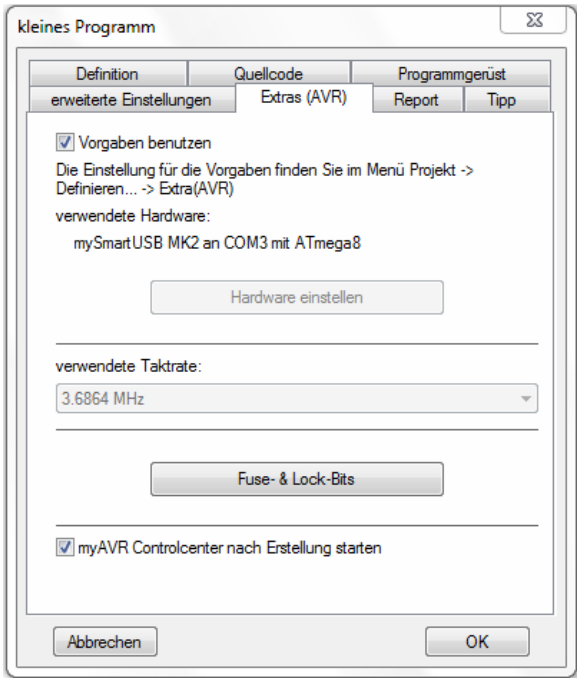
Hinweis:
Für spätere Bearbeitungen, markieren Sie das Objekt und wählen aus dem Kontextmenü (rechte Maustaste) „Definieren“.



Definition kleines Programm

Zielplattform und Programmer einstellen

Kontrollieren Sie auf der Registerkarte „Extras (AVR)“ den ausgewählten Mikrocontroller. Die Option „Vorgaben benutzen“ überträgt automatisch die Grundeinstellungen (Mikrocontrollertyp, Taktrate, Programmer und I/O-Port) des Projektes in die lokalen Einstellungen. Sollen die lokalen Einstellungen unter „Extras (AVR)“ von den Projekteinstellungen abweichen, muss die Option „Vorgaben benutzen“ abgeschaltet werden.



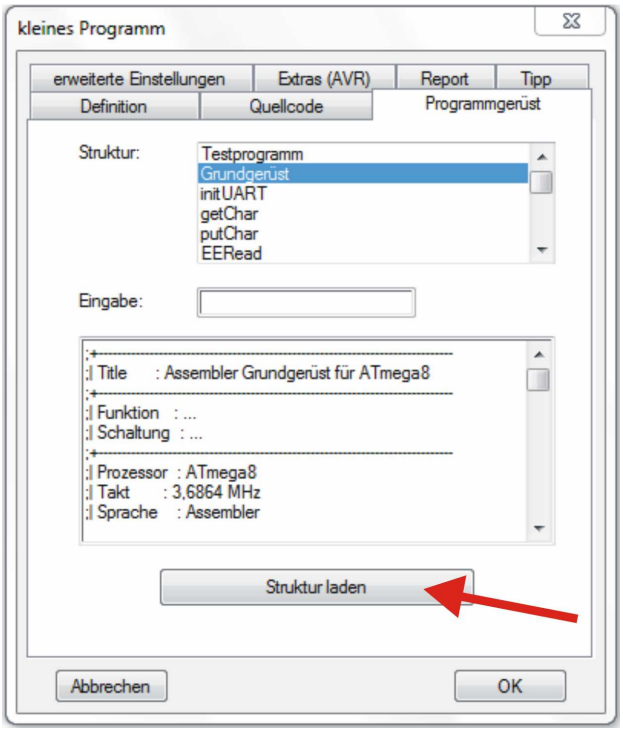
Einstellungen Extras (AVR)

Programmgerüst laden, Quellcode erstellen

Über die Registerkarte „Programmgerüst“ können Sie das „Grundgerüst“ für ein AVR Assemblerprogramm laden; in der Registerkarte „Quellcode“ können Sie den Quellcode eigenständig eintragen.

<u>Hinweis:</u>
Den Quellcode können Sie auch im Beschreibungsfenster/Editorfenster der SiSy-Benutzeroberfläche eintragen bzw. korrigieren.

Laden Sie die Vorlage für das Programmgerüst oder erstellen Sie den folgenden Quellcode. Vergleichen Sie dazu auch den Abschnitt zum myAVR Code-Wizard (Abschnitt 10.4). Das Programmgerüst darf erst geladen werden, wenn der Zielcontroller ausgewählt wurde; die Vorlagen sind controllerspezifisch.



Grundgerüst laden

```
;+----------------------------------------------------------------------------
;| Title                          : Assembler Grundgerüst für ATmega8
;+----------------------------------------------------------------------------
;| Prozessor                      : ATmega8
;| Takt                           : 3,6864 MHz
;| Sprache                        : Assembler
;+----------------------------------------------------------------------------
.include        "AVR.H"
;----------------------------------------------------------------------------
begin:      rjmp   main                ;  Power-on Reset
            reti                       ;  INT0 External Interrupt Request 0
            reti                       ;  INT1 External Interrupt Request 1
            reti                       ;  TIMER2 COMP Timer/Counter2 Compare Match
            reti                       ;  TIMER2 OVF Timer/Counter2 Overflow
            reti                       ;  TIMER1 CAPT Timer/Counter1 Capture Event
            reti                       ;  TIMER1 COMPA Timer/Counter1 Compare Match A
            reti                       ;  TIMER1 COMPB Timer/Counter1 Compare Match B
            reti                       ;  TIMER1 OVF Timer/Counter1 Overflow
            reti                       ;  TIMER0 OVF Timer/Counter0 Overflow
            reti                       ;  SPI, STC Serial Transfer Complete
            reti                       ;  USART, RXC USART, Rx Complete
            reti                       ;  USART, UDRE USART Data Register Empty
            reti                       ;  USART, TXC USART, Tx Complete
            reti                       ;  ADC ADC Conversion Complete
            reti                       ;  EE_RDY EEPROM Ready
            reti                       ;  ANA_COMP Analog Comparator
            reti                       ;  TWI 2-wire Serial Interface
            reti                       ;  SPM_RDY Store Program Memory Ready
;----------------------------------------------------------------------------
main:       ldi          r16,hi8(RAMEND)     ; Main program start
            out          SPH,r16             ; Set Stack Pointer to top of RAM
            ldi          r16,lo8(RAMEND)
            out          SPL,r16
            ;Hier Init-Code eintragen.
;----------------------------------------------------------------------------
mainloop:   wdr
            ;Hier den Quellcode eintragen.
            rjmp   mainloop
```

Quellcode in Assembler erstellen

Das Lauflicht soll über die LEDs angezeigt und von dem Prozessorport D gesteuert werden. Die Realisierung erfolgt über je ein Bit im Register R18. Dieses wird mit dem Befehl Bit-Rotation nach rechts verschoben und an PORT D des Prozessors ausgegeben. Auf Grund der Prozessorgeschwindigkeit muss die Ausgabe des Lauflichtes für unser Auge verzögert werden. Geben Sie folgenden Quellcode ein bzw. ergänzen Sie die Programmvorlage!

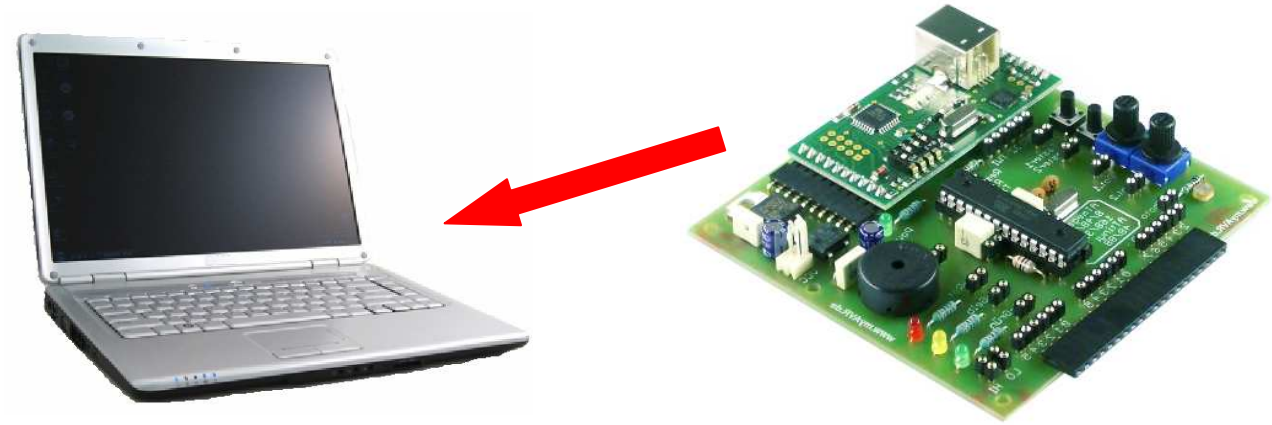
```
;-------------------------------------------------------------------------------
;* Titel            :Lauflicht für myAVR Board
;* Prozessor        :ATmega8 mit 3,6864 MHz
;* Schaltung        :PORT D.2 bis PORT D.4 an LED 1 bis 3
;* Datum            :31.01.2011
;* Autor            :Dipl. Ing. Päd. Alexander Huwaldt
;-------------------------------------------------------------------------------
.include            "avr.h"
;-------------------------------------------------------------------------------

; Reset and Interruptvectoren    ; VNr. Beschreibung
begin:        rjmp    main       ; 1    POWER ON RESET
              reti               ; 2    Int0-Interrupt
              reti               ; 3    Int1-Interrupt
              reti               ; 4    TC2 Compare Match
              reti               ; 5    TC2 Overflow
              reti               ; 6    TC1 Capture
              reti               ; 7    TC1 Compare Match A
              reti               ; 8    TC1 Compare Match B
              reti               ; 9    TC1 Overflow
              reti               ; 10   TC0 Overflow
              reti               ; 11   SPI, STC Serial Transfer Complete
              reti               ; 12   UART Rx Complete
              reti               ; 13   UART Data Register Empty
              reti               ; 14   UART Tx complete
              reti               ; 15   ADC Conversion Complete
              reti               ; 16   EEPROM Ready
              reti               ; 17   Analog Comparator
              reti               ; 18   TWI (I²C) Serial Interface
              reti               ; 19   Store Program Memory Redy
;-------------------------------------------------------------------------------
; Start, Power ON, Reset
main:         ldi    r16 , lo8(RAMEND)
              out    SPL , r16                ; Init Stackpointer LO
              ldi    r16 , hi8(RAMEND)
              out    SPH , r16                ; Init Stackpointer HI
              ldi    r16 , 0b11111111         ; PORT D auf Ausgang
              out    DDRD , r16
              ldi    r16 , 0b00000000         ; Alle Bits auf LOW
              out    PORTD , r16
              ldi    r17 , 0b00000000
              ldi    r18 , 0b00000001         ; 1 Lauflicht-Bit
;-------------------------------------------------------------------------------
mainloop:     wdr
              inc    r16                      ; Wait
              brne   skip
              inc    r17                      ; Wait
              brne   skip
              rcall  up1                      ; Lauflicht
skip:         rjmp   mainloop
;-------------------------------------------------------------------------------
up1:          rol    r18                      ; Bit-Rotation
              out    PORTD , r18
              ret
;-------------------------------------------------------------------------------
```

Kompilieren und Linken

Der eingegebene Quellcode muss nun in Maschinencode für den AVR-Prozessor übersetzt werden. Wählen Sie dazu die Schaltflächen „Kompilieren“ und „Linken“. Bei fehlerfreier Übersetzung liegt das Programm als „Lauflicht.hex“ vor und kann auf den FLASH-Programmspeicher des Prozessors gebrannt werden.

Hardware anschließen und brennen

Das myAVR Board verfügt über eine ISP (In System Programming) Schnittstelle. Der Prozessor muss also nicht für die Programmierung aus dem System entfernt werden, um ihn in einem gesonderten Programmiergerät zu brennen, sondern kann in dem myAVR Board direkt programmiert werden. Dazu verbinden Sie das myAVR Board über das Programmierkabel mit dem USB-Port Ihres Rechners.



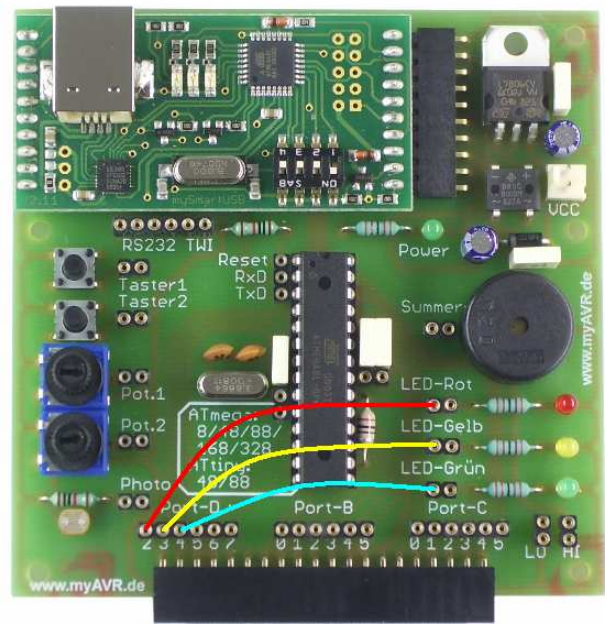
Zum Brennen wählen Sie die Schaltfläche „Brennen“. Bei erfolgreichem Brennvorgang erhalten Sie im Ausgabefenster vom myAVR ProgTool folgende Meldung:

Ausgabefenster myAVR ProgTool

Mikrocontrollerlösung testen

Für den Test des Programms ist es nötig, den Port D mit den LEDs zu verbinden.

- Wenn vorhanden, ziehen Sie die Batterie bzw. das Netzteil und das Programmierkabel ab.
- Verbinden Sie die LEDs mit dem Prozessorport D entsprechend der Abbildung. Nutzen Sie die Patchkabel!
- Prüfen Sie die Verbindungen und schließen Sie die Batterie/das Netzteil oder das Programmierkabel wieder an und nehmen die Mikrocontrollerlösung in Betrieb.
- Es ist jetzt an den LEDs ein Lauflicht zu sehen.
- Gratulation!
  Das ist Ihre erste Mikrocontrollerlösung mit dem myAVR Board.



Beim Kompilieren, Linken und Brennen des Schnellstart-Beispiels öffnet sich ein Ausgabefenster und zeigt Protokollausgaben der Aktionen an. Beim Brennen öffnet sich zusätzlich das myAVR ProgTool. Wenn die Hardware ordnungsgemäß angeschlossen, von der Software erkannt und das Programm erfolgreich auf den Programmspeicher des Mikrocontrollers übertragen wurde, schließen Sie das myAVR ProgTool. Die letzte Ausschrift hat folgenden Inhalt:

```
brenne Daten neu
Ende.
```

Ausgabefenster mit "Brenn-Protokoll“

4.3 Vorgehen für ARM Produkte

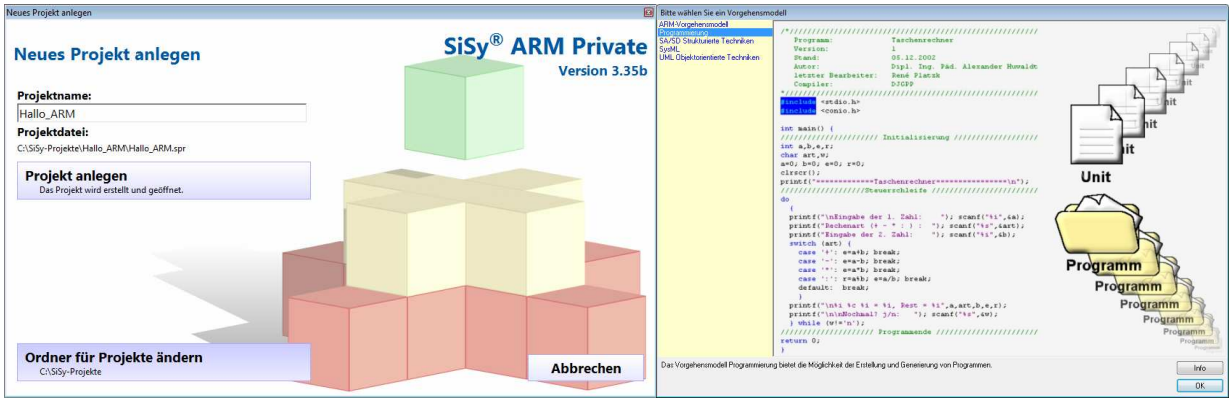
Hinweis:
Dieses Beispiel ist erstellt mit der Ausgabe SiSy ARM.

Zielstellung

In dem ersten C-Programm sollen zwei LEDs auf dem STM32F4-Discovery in kurzen Zeitabständen aufleuchten und damit ein „Blinklicht“ erzeugen.

Ein neues Projekt anlegen

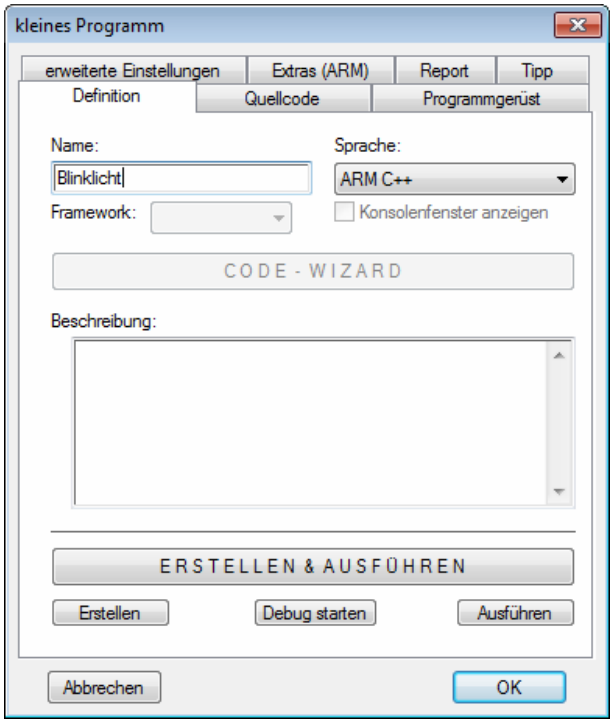
Starten Sie SiSy und wählen Sie „Assistent öffnen“. Aktivieren Sie im SiSy-Assistent den Menüpunkt „Neues Projekt anlegen“ und vergeben Sie den Projektnamen „Hallo_ARM“. Bestätigen Sie diesen Dialog mit „Projekt anlegen“.
Wählen Sie das Vorgehensmodell „Programmierung“ und bestätigen Sie mit „OK“. Es öffnet das Diagrammfenster für die weitere Bearbeitung.



Hinweis:
In SiSy legen Sie stets ein Projekt an. In dieses Projekt integrieren Sie Ihr Programm bzw. mehrere Programme. Unabhängig vom Programmnamen benötigt jedes Projekt einen Namen. Beachten Sie die Erläuterungen im Abschnitt 3.1.2.

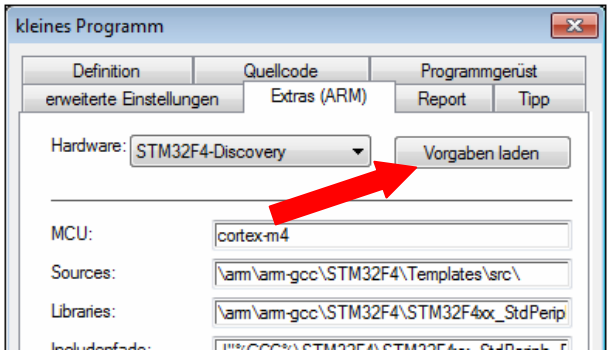
Kleines C-Programm anlegen

Erstellen Sie ein Programm für den ARM-Mikrocontroller, indem Sie per Drag & Drop aus der Objektbibliothek ein Objekt „kleines Programm“ in das Diagrammfenster ziehen. Das Kontextmenü öffnet sich automatisch.
Auf der Registerkarte „Definition“ tragen Sie den Programmnamen ein (hier „Blinklicht“) und wählen die Sprache „ARM C++“.



Hinweis:
Für spätere Bearbeitungen, markieren Sie das Objekt und wählen aus dem Kontext-menü (rechte Maustaste) „Definieren“.

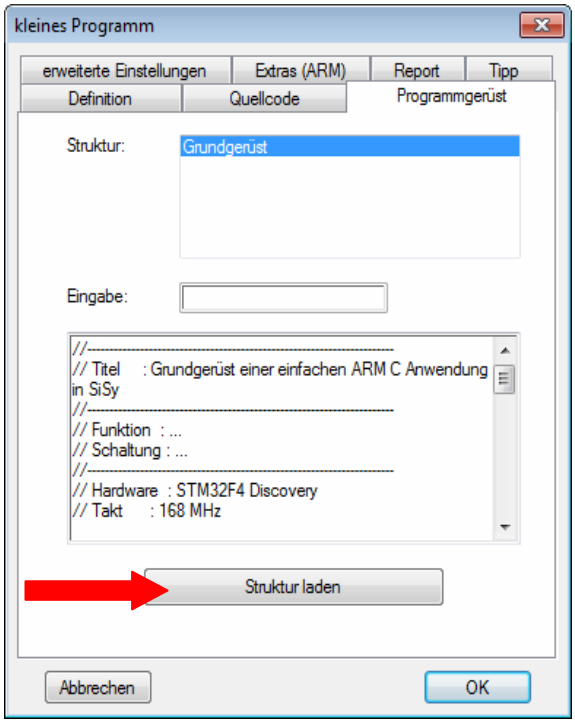
Über die Registerkarte „Extras (ARM)“ treffen Sie die Hardware-Auswahl und aktivieren die Schaltfläche „Vorgabe laden“. In unserem Beispiel ist die Hardware „STM32F4-Discovery“.



Programmgerüst laden, Quellcode erstellen

Über die Registerkarte „Programmgerüst“ können Sie das folgende Grundgerüst für ein ARM Programm laden; in der Registerkarte „Quellcode“ können Sie den Quellcode eigenständig eintragen.

Hinweis:
Den Quellcode können Sie im Beschreibungsfenster/Editorfenster der SiSy-Benutzeroberfläche eintragen bzw. korrigieren.



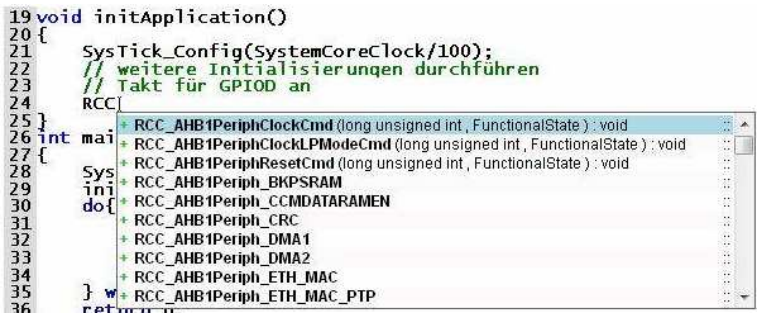
Laden Sie die Vorlage („Struktur laden“) für das Programmgerüst oder erstellen Sie den folgenden Quellcode.

```
#include <stddef.h>
#include <stdlib.h>
#include "stm32f4xx.h"

void initApplication()
{
    SysTick_Config(SystemCoreClock/100);
    // weitere Initialisierungen durchführen
}

int main(void)
{
    SystemInit();
    initApplication();
    do{
        // Eingabe
        // Ausgabe
        // Verarbeitung
    } while (true);
    return 0;
}

extern "C" void SysTickFunction(void)
{
    // Application SysTick

}
```

Quellcode in C erstellen

Das Blinklicht soll über die LEDs angezeigt und von dem Prozessorport GPIOD gesteuert werden. Die Realisierung erfolgt über GPIO Pin 12 und 13. Dieses wird mit dem Befehl Bit-Rotation nach rechts verschoben und an den Port GPIOD des Prozessors ausgegeben. Auf Grund der Prozessorgeschwindigkeit muss die Ausgabe des Blinklichtes für unser Auge verzögert werden. Geben Sie folgenden Quellcode ein bzw. ergänzen Sie die Programmvorlage!

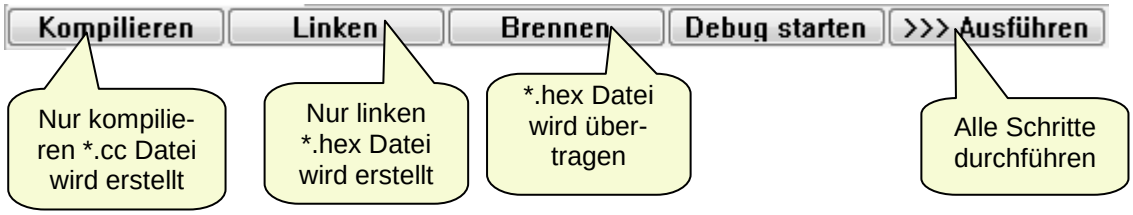
Bei der Eingabe des Quellcodes springt nach drei zusammenhängenden Buchstaben die Codevervollständigung an und listet alle Bezeichner fortlaufend gefiltert. Wählen Sie jetzt die Cursor-Taste mit Pfeil nach unten, Sie können in der Liste rollen und per Enter einen Eintrag auswählen.




```
#include <stddef.h>
#include <stdlib.h>
#include "stm32f4xx.h"

void initApplication()
{
    SysTick_Config(SystemCoreClock/100);
    // weitere Initialisierungen durchführen

    /* GPIOG Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

    /* Configure PG6 and PG8 in output pushpull mode */
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}

void delay(vu32 nCount)
{
      while(nCount--);
}

int main(void)
{
    SystemInit();
    initApplication();
    GPIO_SetBits(GPIOD,GPIO_Pin_13);
    do{
        // Eingabe
        // Ausgabe
        // Verarbeitung
        GPIO_ToggleBits(GPIOD,GPIO_Pin_13|GPIO_Pin_12);
        delay(10000000);
        } while (true);
    return 0;
}
extern "C" void SysTickFunction(void)
{
    // Application SysTick
}
```

Kompilieren und Linken

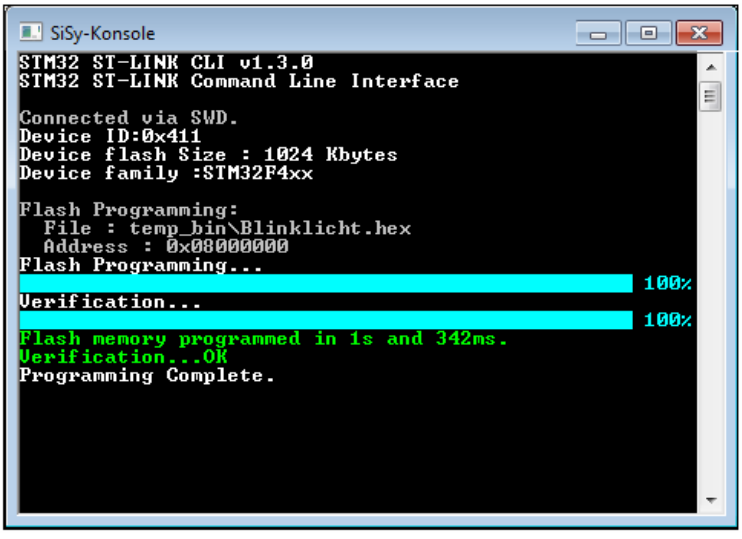
Der eingegebene Quellcode muss nun in Maschinencode für den ARM-Prozessor übersetzt werden. Wählen Sie dazu die Schaltflächen „Kompilieren“ und „Linken“. Bei fehlerfreier Übersetzung liegt das Programm als „Blinklicht.hex“ vor und kann auf den FLASH-Programmspeicher des Prozessors gebrannt werden.



Hardware anschließen und brennen

Das STM32F4-Discovery verfügt über eine ISP (In System Programming) Schnittstelle. Der Prozessor muss also nicht für die Programmierung aus dem System entfernt werden, um ihn in einem gesonderten Programmiergerät zu brennen, sondern kann im STM32F4 direkt programmiert werden. Dazu verbinden Sie das STM32F4-Discovery über das Programmierkabel mit dem USB-Port Ihres Rechners.

Zum Brennen wählen Sie die Schaltfläche „Brennen“.
Beim Kompilieren, Linken und Brennen des Beispiels öffnet sich ein Ausgabefenster und zeigt Protokollausgaben der Aktionen an. Beim Brennen öffnet sich zusätzlich das ST-Link.

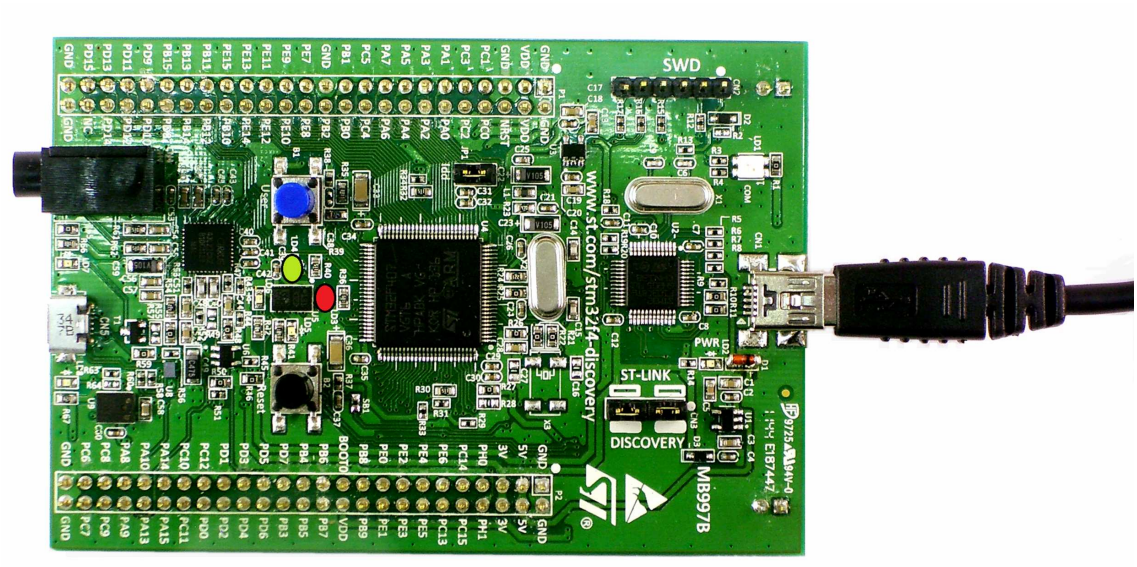


Bei erfolgreichem Brennvorgang erhalten Sie im Ausgabefenster folgende Meldung:

Mikrocontrollerlösung testen

Für den Programmtest ist es nötig, dass das STM32F4-Discovery mit dem PC verbunden ist.

- Es ist an den LEDs ein Blinklicht zu sehen.
- Gratulation!
  Das ist Ihre erste Mikrocontrollerlösung mit dem STM32F4-Discovery.



Die Datenkommunikation mit der Mikrocontrollerlösung erfolgt über das ControlCenter. Vergleichen Sie dazu den Abschnitt 10.2 zum ControlCenter.

5 Entwicklung eines großen Programms

5.1 Einleitung

Für die Entwicklung eines größeren Programms ist es unzweckmäßig, alle Befehle in eine Datei (Unit) zu schreiben. Der Grund dafür ist, dass bei mehr als 60 bis 80 Zeilen Quellcode die Übersicht über die Programmstruktur verloren geht. Selbst die Unterteilung in Unterprogramme reicht ab einer bestimmten Größe von Programmen nicht mehr aus. SiSy erlaubt zwar in kleinen Programmen bzw. je Unit 10 Kilobyte Code. Das sind in Assembler zum Beispiel über 1000 Zeilen. Ein kleines Programm bzw. eine einzelne Unit sollte jedoch nie mehr als 80 bis 120 Zeilen haben. Wird diese Grenze erreicht, sollte das Programm in logische Einheiten (Units, Module) gegliedert werden. Dabei fasst man alles zusammen, was zu einer bestimmten Funktionalität oder einer bestimmten Baugruppe wie zum Beispiel dem AD-Wandler gehört. Physisch entstehen dabei mehrere Dateien, die für sich genommen wieder übersichtlich sind, da diese dann nur 80 bis 120 Zeilen Code enthalten. Das Übersetzungsprogramm (Assembler, Compiler, Linker) sorgt dann dafür, dass alle einzelnen Units zu einem vollständigen Programm zusammengefügt werden.
Die folgenden Kapitel erläutern jeweils an einem sehr kleinen Beispiel die Handhabung der Komponenten eines großen Programms, welches in mehrere Units zerlegt wird.

5.2 Vorgehen für PC Programme

Hinweis:
Dieses Beispiel ist erstellt mit der Ausgabe SiSy Microcontroller ++.

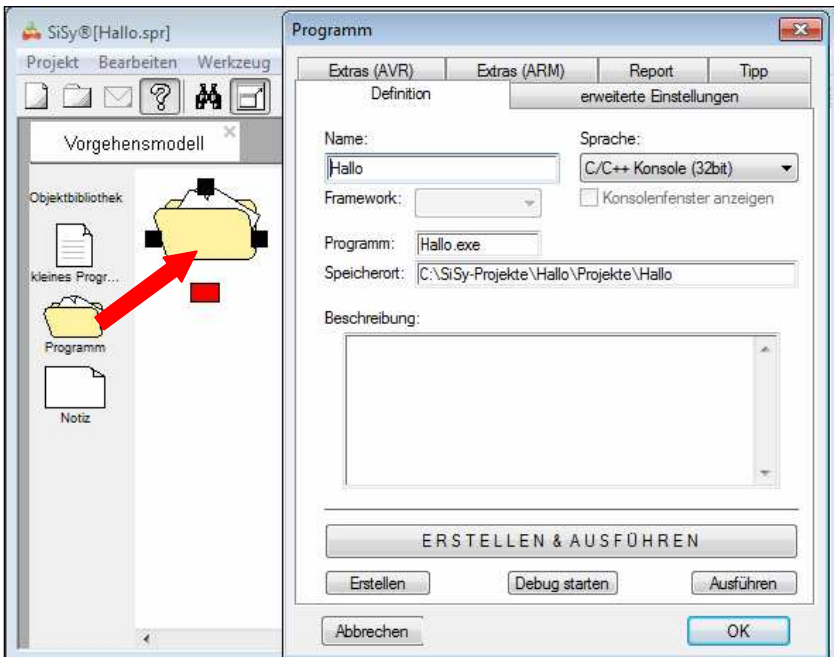
5.2.1 Zielstellung

Im folgenden Beispiel wird ein Programm mit mehreren Modulen veranschaulicht. In einer einfachen Anwendung wird vom Hauptprogramm ein Unterprogramm aufgerufen und ins Hauptprogramm zurückgekehrt. Auf dem Bildschirm werden die Texte dieser Aktionen ausgegeben.

5.2.2 Hauptprogramm erstellen

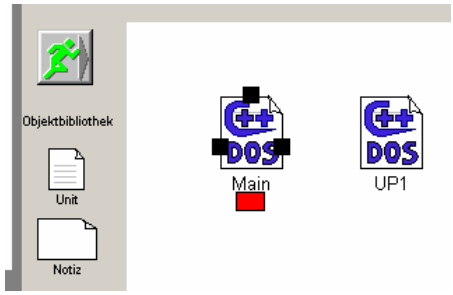
Starten Sie SiSy und legen Sie ein neues Projekt an. Vergeben Sie den Namen „Hallo„ und wählen Sie das Vorgehensmodell „Programmierung“. Das folgende Dialogfeld mit den Infos zur Hardware-Konfiguration brechen Sie ab; danach bestätigen Sie die voreingestellte Auswahl „leeres Diagramm“.

Für ein Programm mit mehreren Modulen ziehen Sie per Drag & Drop das Symbol für „Programm“ in das Diagrammfenster. Es öffnet ein Dialogfenster. Geben Sie für das Beispiel den Namen „Hallo“ ein und wählen Sie die Sprache „C/C++ Konsole“; schließen Sie mit „OK“.



Klicken Sie mit der rechten Maustaste auf das Symbol und wählen im Kontextmenü „Nach unten“. Hier ziehen Sie zwei *Units* in das Diagramm und nennen diese „Main“ und „UP1“.
In dem Dialogfenster zur Definition von „UP1“ setzen Sie einen Haken in den beiden Zeilen

- „Headerdef-Datei erzeugen,
  Datei mit Deklaration wird generiert“
- „Funktionsdef-Datei erzeugen (.cpp)“



5.2.3 Units (Unterprogramme) anlegen und verknüpfen

Übernehmen Sie nun folgende Quelltexte in die jeweiligen Units.

In der Unit „Main“ fügen Sie folgenden Quelltext ein:

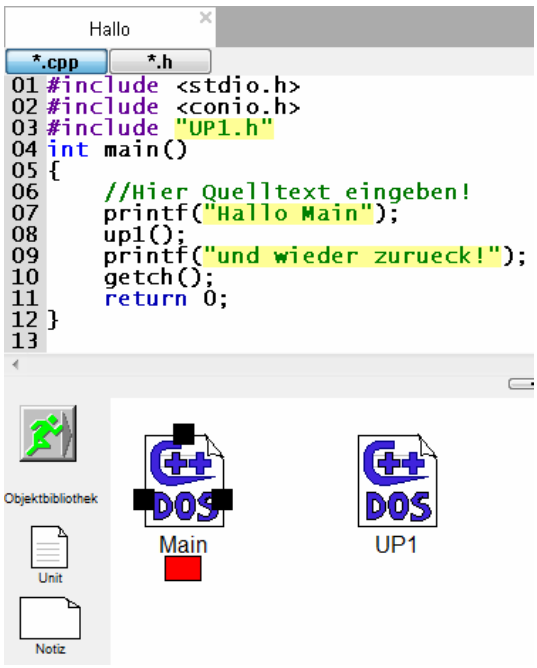
```
#include <stdio.h>
#include <conio.h>
#include "UP1.h"
int main()
{
//Hier Quelltext eingeben!
printf("Hallo Main");
up1();
printf("und wieder zurueck!");
getch();
return 0;
}
```

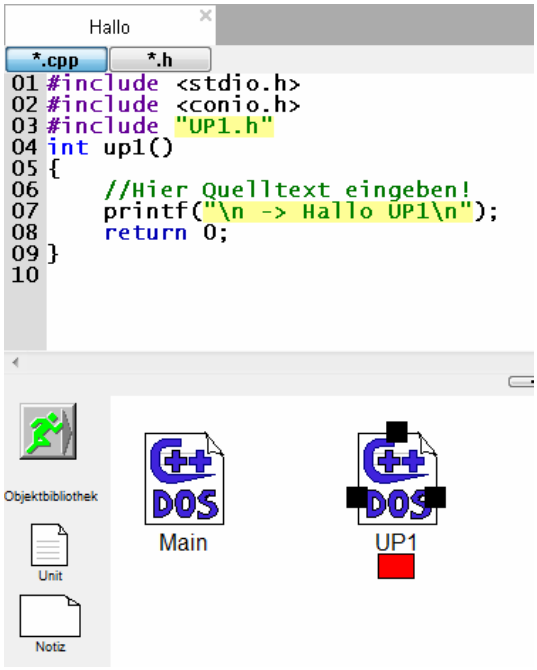
In "UP1" fügen Sie den folgenden Quelltext ein:

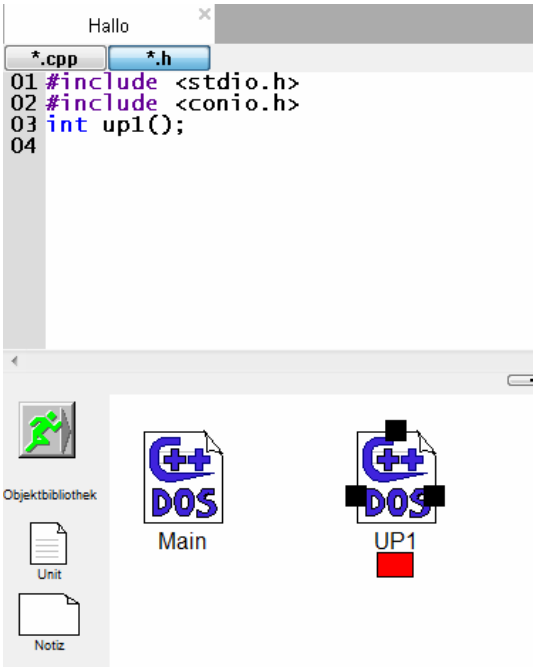
```
#include <stdio.h>
#include <conio.h>
#include "UP1.h"
int up1()
{
//Hier Quelltext eingeben!
printf("\n -> Hallo UP1\n");
return 0;
}
```

Danach wählen Sie in „UP1“ die Schaltfläche „*.h“ und tragen die Bibliotheken `stdio` und `conio`, sowie die Definition von „UP1“ ein:

```
#include <stdio.h>
#include <conio.h>
int up1();
```







Zum Ausführen des Programms klicken Sie auf das Aktionsmenü in der Objektbibliothek. Es stehen verschiedene Möglichkeiten zur Auswahl.
Nach „>>> Erstellen und Ausführen“ erscheint folgende Ausgabe auf dem Bildschirm.

5.3 Vorgehen für AVR Programme

Hinweis:
Dieses Beispiel ist erstellt mit der Ausgabe SiSy AVR.

5.3.1 Zielstellung

Es ist eine Mikrocontroller-Anwendung mit der Technik des großen Programms zu entwerfen und in der Sprache Assembler zu realisieren. Das Programm ist in mehrere Units zu teilen.

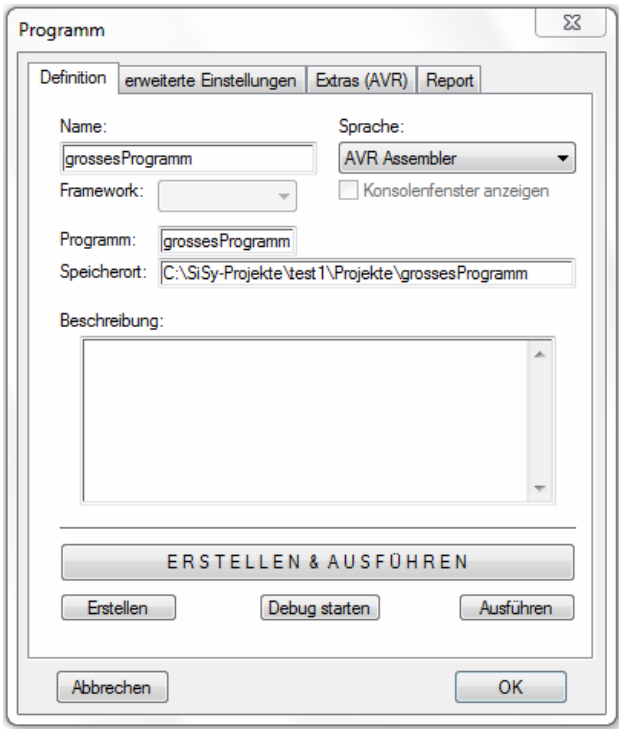
Aufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der ein Taster eine LED schaltet.

Schaltung:
Port D.2 = Taster 1
Port B.0 = LED

5.3.2 Neues Projekt anlegen

Starten Sie SiSy und legen Sie ein neues Projekt an. Wählen Sie das Vorgehensmodell „Programmierung“.
Nehmen Sie die Grundeinstellungen für die verwendete myAVR-Hardware vor oder lassen Sie die myAVR-Hardware automatisch suchen. Erstellen Sie ein leeres Diagramm!

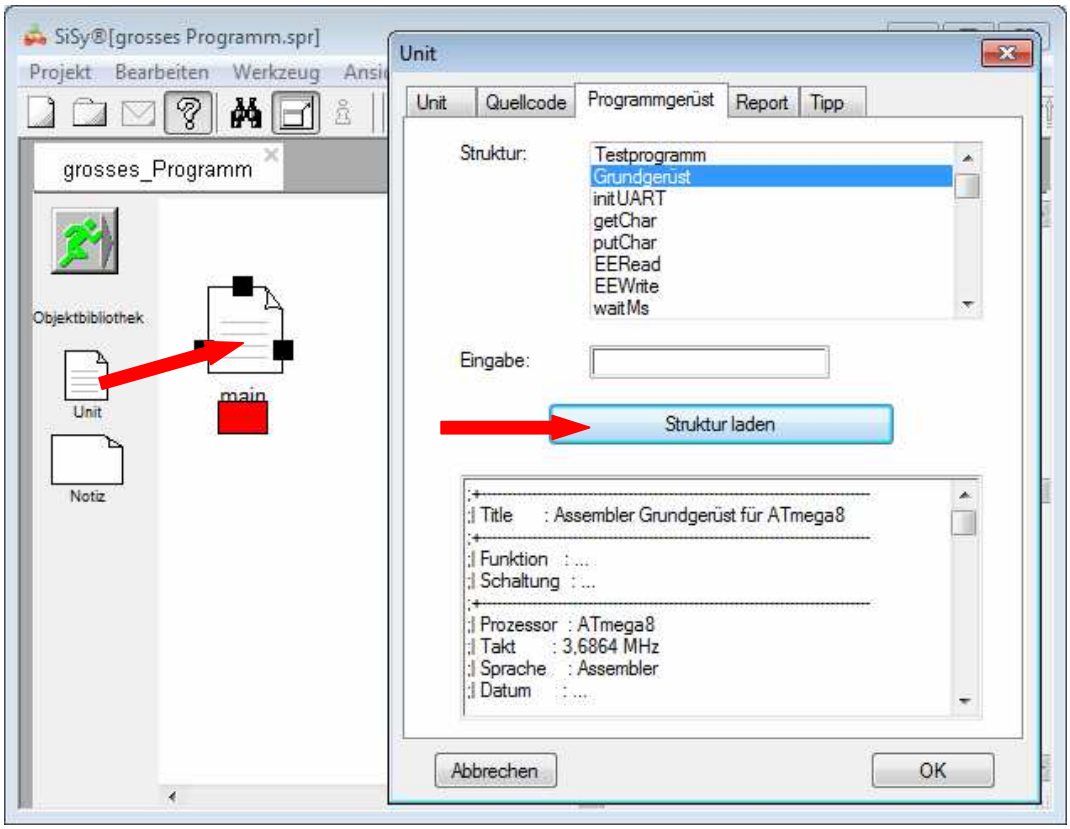
Ziehen Sie aus der Objektbibliothek ein Objekt vom Typ „Programm“.



Legen Sie einen Namen für das Programm fest und die Sprache „AVR Assembler“. Überprüfen Sie gegebenenfalls die Einstellungen unter „Extras AVR“.

5.3.3 Hauptprogramm erstellen

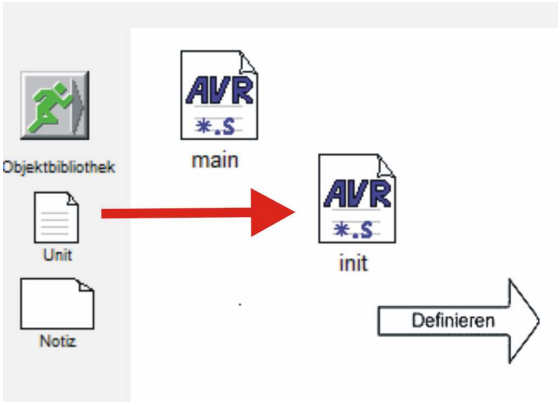
Öffnen Sie das Diagrammfenster für ein großes Programm, indem Sie mit der rechten Maustaste auf das Symbol klicken und aus dem Kontextmenü „Nach unten (öffnen)“ wählen. Legen Sie eine *Unit* an. Diese Unit bildet das Hauptprogramm. Nennen Sie die Unit „main“. Damit wird durch die Entwicklungsumgebung erkannt, dass es sich hierbei um das Hauptmodul handelt. Erstellen Sie hier das Hauptprogramm, nutzen Sie die angebotene Vorlage „Grundgerüst“ und aktivieren Sie „Struktur laden“. Der unten stehende Quellcode wird damit übernommen.

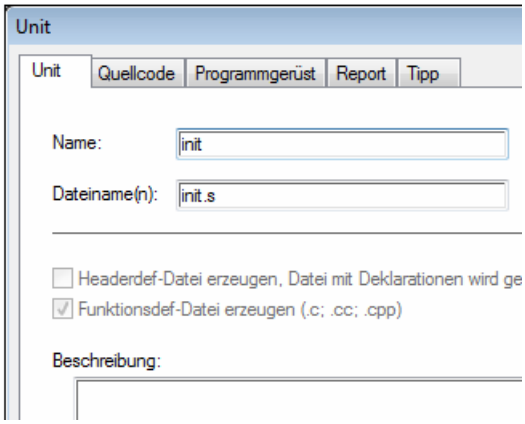



```
;-----------------------------------
.include "AVR.H"
;-----------------------------------
; Reset and Interrupt vectoren
begin:   rjmp    main
         reti
         reti
         reti
         reti
;...
main:    ldi     r16, lo8(RAMEND)
         out     SPL, r16
         ldi     r16, hi8(RAMEND)
         out     SPH, r16
         ; Hier Init-Code eintragen.
;-----------------------------------
mainloop: wdr
          ; Hier Quellcode eintragen.
          rjmp mainloop
```

5.3.4 Units (Unterprogramme) anlegen und verknüpfen

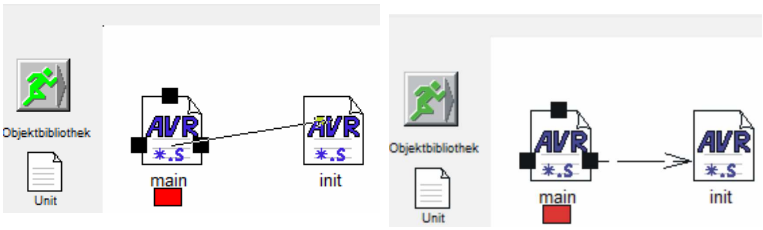
Zur Gliederung des großen Programms wird dieses in mehrere kleine Einheiten (Units bzw. Module) zerlegt. Diese Einheiten werden nach fachlichen/inhaltlichen Gesichtspunkten gebildet. So kann man alle Initialisierungsaufgaben in der Unit „init“ zusammenfassen. Eine Unit kann aus einer Funktion/Unterprogramm oder mehreren Funktionen/Unterprogrammen bestehen. Im einfachsten Fall enthält jede Unit ein Unterprogramm. Legen Sie zusätzlich zur Hauptunit „main“ die Unit „init“ an.





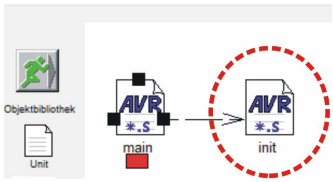
Die Hauptunit benutzt die Unit „init“. Daher ist eine Verbindung von der Hauptunit „main“ zur Unit „init“ zu ziehen. Selektieren Sie die Unit „main“ und ziehen vom Verteiler (rot) eine Verbindung auf die Unit „init“. Dabei wird in der Hauptunit „main“ ein Include-Eintrag für die Unit „init“ erzeugt:

```
.include "init.s"
```



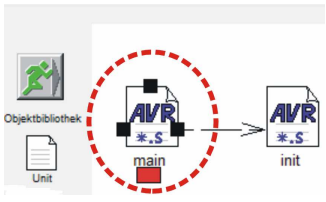
Erstellen Sie die Initialisierungsroutine für die benötigten digitalen Ein und Ausgänge in der Unit „init“:

```
;---------------------------------------
init:       push    r16
            sbi     DDRB,0    ; B.0 Ausgang
            cbi     PORTB,0   ; B.0 LED aus
            cbi     DDRD,2    ; D.2 Eingang
            sbi     PORTD,2   ; D.2 PullUp
            pop     r16
            ret
;---------------------------------------
```



Ergänzen Sie den Code des Hauptprogramms:

```
            ...
            rcall   init
;---------------------------------
mainloop:  sbic    PIND,2
           rjmp    led_aus
led_an:    sbi     PORTB,0
           rjmp    mainloop
led_aus:   cbi     PORTB,0
           rjmp    mainloop
;---------------------------------
.include “init.s“
```



Die Unit „main“ enthält damit nachfolgenden Quellcode:

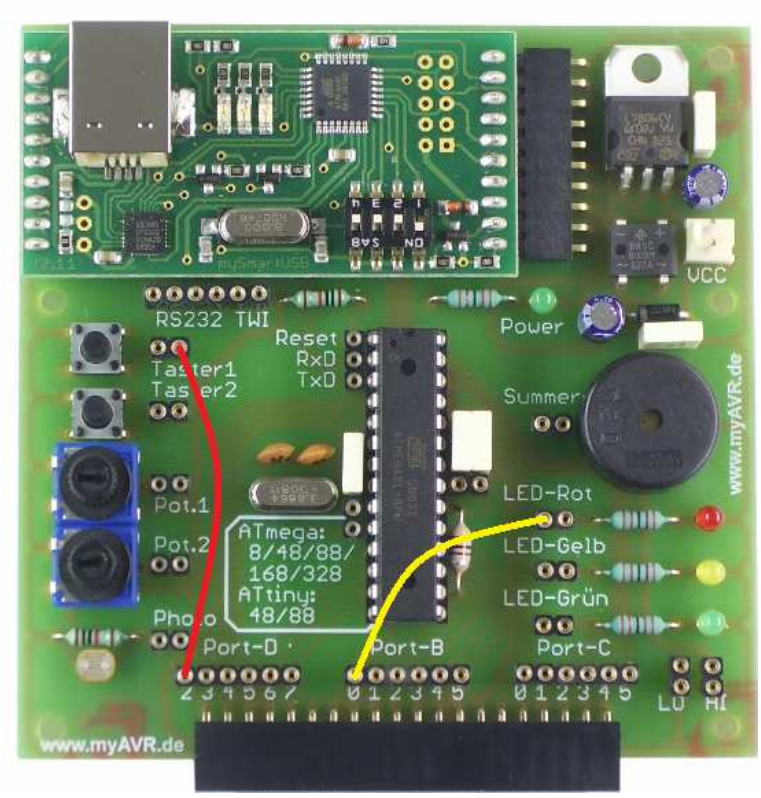
```
;+---------------------------------------------------------------
;| Title      : Assembler Grundgerüst für ATmega8
;+---------------------------------------------------------------
;| Funktion   : ...
;| Schaltung  : ...
;+---------------------------------------------------------------
;| Prozessor  : ATmega8
;| Takt       : 3,6864 MHz
;| Sprache    : Assembler
;| Datum      : ...
;| Version    : ...
;| Autor      : ...
;+---------------------------------------------------------------
.include        "AVR.H"
;---------------------------------------------------------------
begin:
    rjmp    main        ; RESET External Pin, Power-on Reset,
                        ; Brown-out Reset and Watchdog Reset
    reti                ; INT0 External Interrupt Request 0
    reti                ; INT1 External Interrupt Request 1
    reti                ; TIMER2 COMP Timer/Counter2 Compare Match
    reti                ; TIMER2 OVF Timer/Counter2 Overflow
    reti                ; TIMER1 CAPT Timer/Counter1 Capture Event
    reti                ; TIMER1 COMPA Timer/Counter1 Compare Match A
    reti                ; TIMER1 COMPB Timer/Counter1 Compare Match B
    reti                ; TIMER1 OVF Timer/Counter1 Overflow
    reti                ; TIMER0 OVF Timer/Counter0 Overflow
    reti                ; SPI, STC Serial Transfer Complete
    reti                ; USART, RXC USART, Rx Complete
    reti                ; USART, UDRE USART Data Register Empty
    reti                ; USART, TXC USART, Tx Complete
    reti                ; ADC ADC Conversion Complete
    reti                ; EE_RDY EEPROM Ready
    reti                ; ANA_COMP Analog Comparator
    reti                ; TWI 2-wire Serial Interface
    reti                ; SPM_RDY Store Program Memory Ready
;---------------------------------------------------------------------------
main:
    ldi r16,hi8(RAMEND)             ; Main program start
    out SPH,r16                     ; Set Stack Pointer to top of RAM
    ldi r16,lo8(RAMEND)
    out SPL,r16
    rcall   init                    ; Hier Init-Code eingetragen.
;---------------------------------------------------------------------------
mainloop: sbic    PIND,2            ; Hier Quellcode eingetragen
          rjmp    led_aus
led_an:   sbi     PORTB,0
          rjmp    mainloop
led_aus:  cbi     PORTB,0
          rjmp    mainloop

;---------------------------------------------------------------------------
.include "init.s"         ; Eintrag bei Verbindung von "main" auf "init"
```

Übersetzen, Brennen und Test

Zum Übersetzen, Brennen und Testen wählen Sie im Aktionsmenü  den entsprechenden Menüpunkt. Im Ausgabefenster erscheint das Protokoll der ausgeführten Aktionen. Des Weiteren öffnet sich für kurze Zeit das myAVR ProgTool.

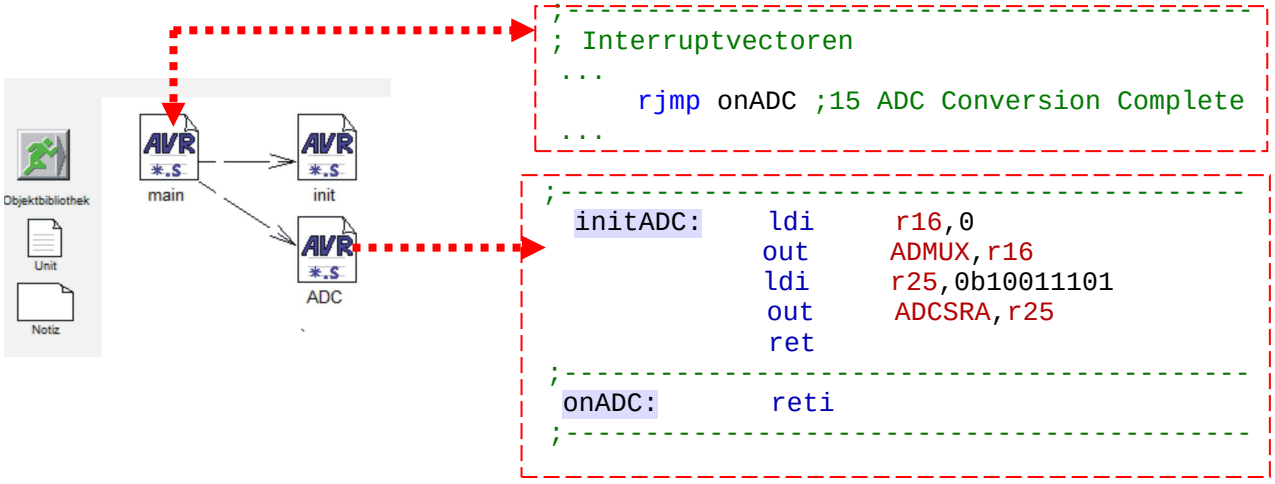
Kompiliere die Datei main.s.
Linke die Datei grossesProgramm.elf.
brenne Daten neu
Öffne myAVR ControlCenter
Ende.

Zum Test des Programms stecken Sie auf dem Board die Verbindungen entsprechende der Vorgabe, nutzen Sie die Patchkabel.
Die Datenkommunikation mit dem Board kann über das ControlCenter erfolgen. Weitere Informationen dazu im Absatz 10.2, (ControlCenter).



5.3.5 Interrupt-Service-Routine (ISR) im großen Programm

Interrupt-Service-Routinen (im weiteren ISR) sind besondere Formen von Unterprogrammen. Diese werden von einer Interruptquelle des Mikrocontrollers (Timer, ADC, UART, usw.) bei entsprechenden Ereignissen automatisch an beliebiger Stelle im Programmfluss aufgerufen (Unterbrechung, engl. Interrupt). Es ist nötig, die Interruptquelle entsprechend zu konfigurieren. Es empfiehlt sich für jedes interruptfähige Gerät eine eigene Unit anzulegen. Diese ist mit der Hauptunit zu verbinden.



5.4 Vorgehen für ARM Programme

Hinweis:
Dieses Beispiel ist erstellt mit der Ausgabe SiSy ARM.

5.4.1 Zielstellung

Das Beispiel aus Kapitel 4.3 soll jetzt mit der Technik des großen Programms (Zerlegen in mehrere Units) erstellt werden, ebenfalls in der Programmiersprache C.

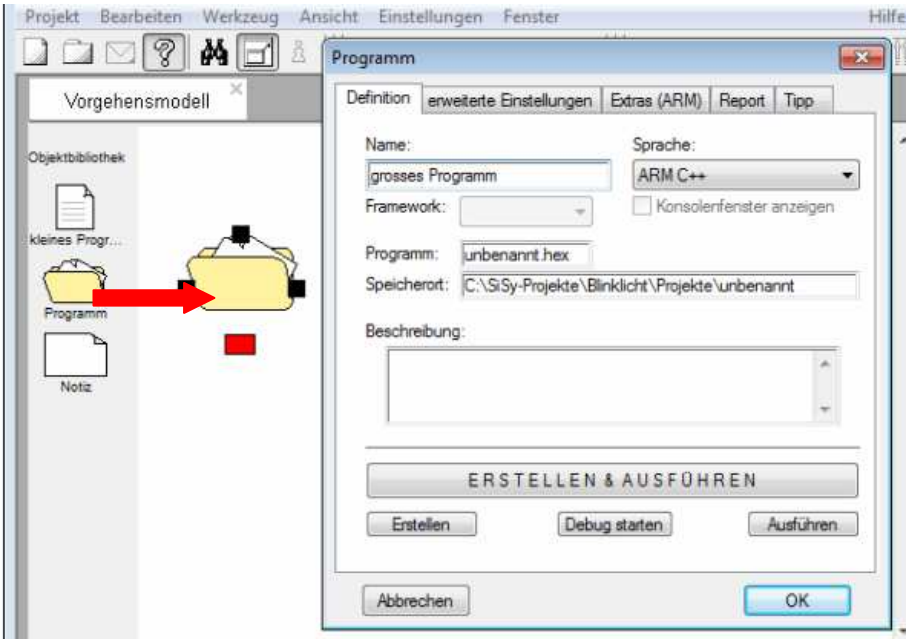
Aufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der ein Blinklicht erzeugt wird.

Schaltung:
GPIOD.13 = LED
GPIOD.12 = LED

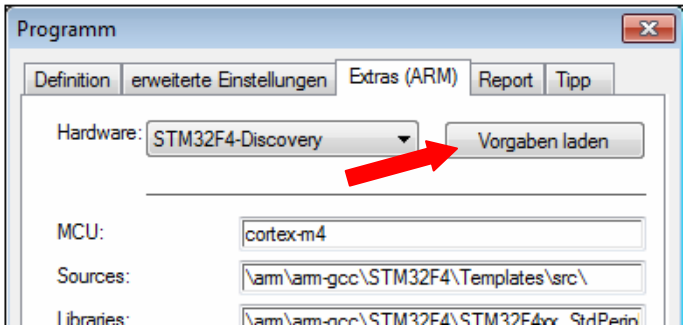
Das Blinklicht soll über die LEDs angezeigt und von dem Prozessorport GPIOD gesteuert werden. Die Realisierung erfolgt über GPIO Pin 12 und 13. Dieses wird mit dem Befehl Bit-Rotation nach rechts verschoben und an den Port GPIOD des Prozessors ausgegeben. Auf Grund der Prozessorgeschwindigkeit muss die Ausgabe des Blinklichtes für unser Auge verzögert werden.

5.4.2 Neues Projekt anlegen

Starten Sie SiSy und legen Sie ein neues Projekt an. Wählen Sie das Vorgehensmodell „Programmierung“ aus. Ziehen Sie aus der Objektbibliothek ein Objekt vom Typ „Programm“. Vergeben Sie für das Programm einen Namen und legen Sie die Sprache „ARM C++“ fest.



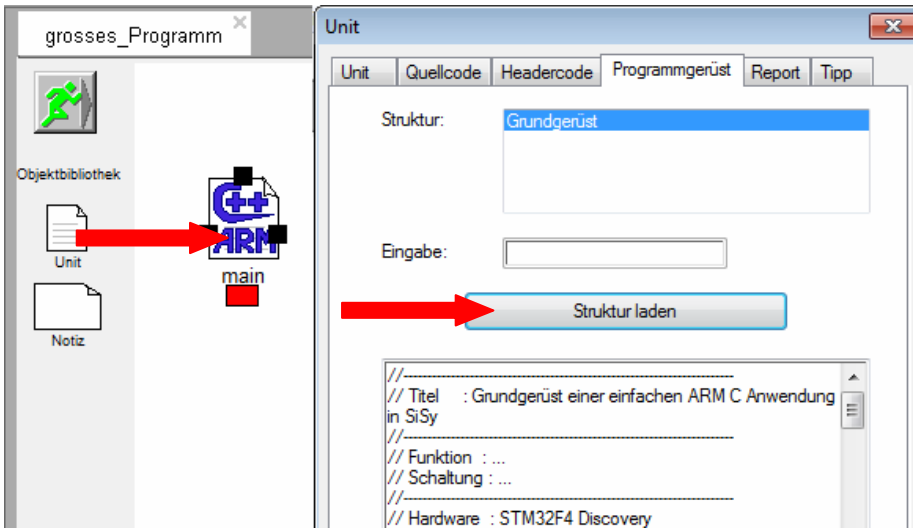
Über die Registerkarte „Extras (ARM)“ treffen Sie die Hardware-Auswahl und aktivieren die Schaltfläche „Vorgabe laden“. In unserem Beispiel ist die Hardware „STM32F4-Discovery“.



5.4.3 Hauptprogramm erstellen

Öffnen Sie das Diagrammfenster für ein großes Programm (rechte Maustaste -> „Nach unten“). Legen Sie eine Unit „main“ an. Diese Unit bildet das Hauptprogramm. Damit wird durch die Entwicklungsumgebung erkannt, dass es sich hierbei um das Hauptmodul handelt.

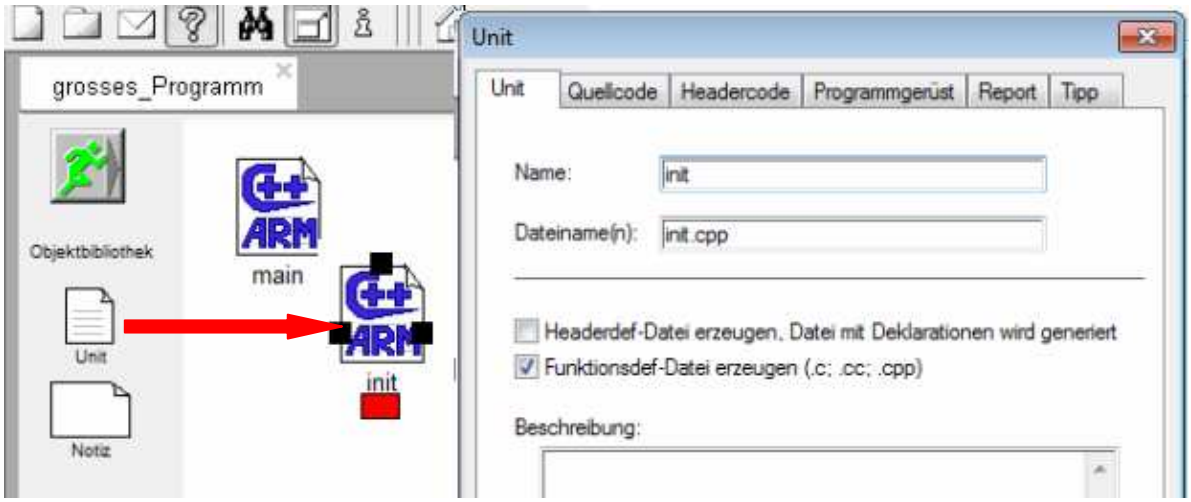
Erstellen Sie hier das Hauptprogramm, nutzen Sie die angebotene Vorlage „Grundgerüst“ und aktivieren Sie „Struktur laden“.



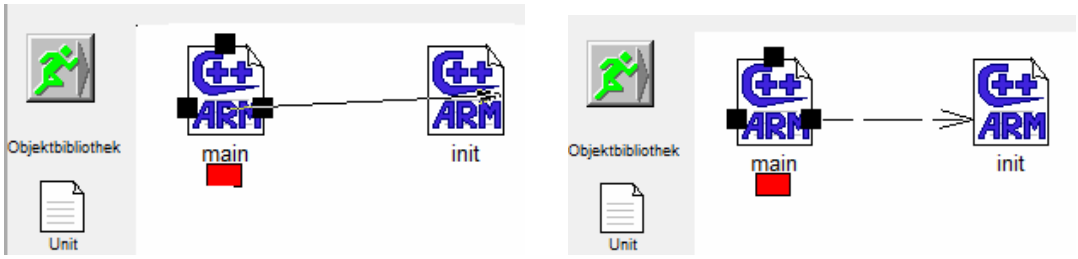

```
//-------------------------------------------------------------------------
// Titel     : Grundgerüst einer einfachen ARM C Anwendung in SiSy
//-------------------------------------------------------------------------
// Funktion  : ...
// Schaltung : ...
//-------------------------------------------------------------------------
// Hardware  : STM32F4 Discovery
// Takt      : 168 MHz
// Sprache   : ARM C
// Datum     : ...
// Version   : ...
// Autor     : ...
//-------------------------------------------------------------------------
#include <stddef.h>
#include <stdlib.h>
#include "stm32f4xx.h"
//#include "stm32f30x.h"
//#include "stm32f0xx.h"

void initApplication()
{
    SysTick_Config(SystemCoreClock/100);
    // weitere Initialisierungen durchführen
}
int main(void)
{
    SystemInit();
    initApplication();
    do{
        // Eingabe
        // Ausgabe
        // Verarbeitung
    } while (true);
    return 0;
}
extern "C" void SysTickFunction(void)
{
    // Application SysTick
}
```

5.4.4 Units (Unterprogramme) anlegen und verknüpfen

Zur Gliederung des großen Programms wird dieses in mehrere kleine Einheiten (Units/ Module) zerlegt. Diese Einheiten werden nach fachlichen/inhaltlichen Gesichtspunkten gebildet. So kann man alle Initialisierungsaufgaben in der Unit „init“ zusammenfassen. Eine Unit kann aus einer Funktion/Unterprogramm oder mehreren Funktionen/Unter programmen bestehen. Im einfachsten Fall enthält jede Unit ein Unterprogramm.
Legen Sie zusätzlich zur Hauptunit „main“ die Unit „init“ an.



Die Hauptunit benutzt die Unit „init“. Daher ist eine Verbindung von „main“ zur „init“ zu ziehen. Selektieren Sie „main“ und ziehen vom Verteiler (rot) eine Verbindung auf „init“.



Dabei wird in der Hauptunit „main“ ein Include-Eintrag für die Unit „init“ erzeugt. Prüfen Sie diese Eintragung; ggf. ergänzen Sie diese.
Erstellen Sie die Initialisierungsroutine für die benötigten digitalen Ein- und Ausgänge in der Unit „init“. Nutzen Sie bei der Quellcode-Eingabe die Codevervollständigung, wie bereits im Kapitel 4.3 („Vorgehen für ARM Produkte“ – kleines Programm) beschrieben.

```
#include "init.h"

void initApplication()
{
    SysTick_Config(SystemCoreClock/100);
    // weitere Initialisierungen durchführen
    /* GPIOG Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
    /* Configure PG6 and PG8 in output pushpull mode */
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

Ergänzen Sie die Unit „sysTick“ und die Unit „delay“. Tragen Sie in diese folgenden Quellcode ein.

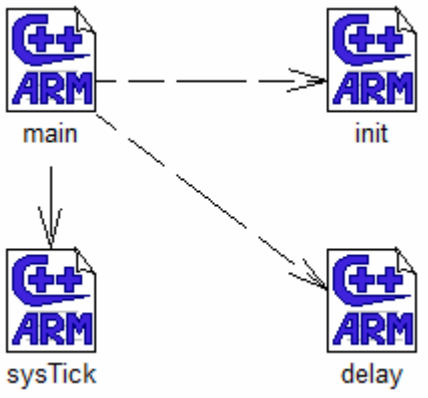
In die Unit „sysTick“:

```
//---------------------------------------
#include "sysTick.h"

extern "C" void SysTickFunction(void)
{
    // Application SysTick
    GPIO_ToggleBits(GPIOD,GPIO_Pin_13);
}
//------------------------------
```

In die Unit „delay“:

```
//------------------------------
#include "delay.h"

void delay(vu32 nCount)
{
    while(nCount)
    {
        nCount--;
    }
}
//------------------------------
```

Diese beiden Units sind noch mit der Hauptunit „main“ zu verbinden. Prüfen Sie in der „main“ die include-Eintäge zu diesen Units; ggf. tragen Sie diese manuell ein.



Die Methoden `void initApplication()` und `void SysTickFunction(void)` sind jetzt in den Units hinterlegt und müssen demzufolge in der Hauptunit entfernt werden. Ergänzen Sie noch in dem Code des Hauptprogramms:

```
#include "main.h"

int main(void)
{
    SystemInit();
    initApplication();
    GPIO_SetBits(GPIOD,GPIO_Pin_12);
    do{
        // Eingabe
        // Ausgabe
        // Verarbeitung
        GPIO_ToggleBits(GPIOD,GPIO_Pin_12);
        delay(10000000);
        } while (true);
    return 0;
}
```

Damit hat das Programm nachfolgenden Quellcode:

Unit „main“

```
//-------------------------------------------------------------------
// Titel     : Grundgerüst einer einfachen ARM C Anwendung in SiSy
//-------------------------------------------------------------------
// Funktion  : ...
// Schaltung : ...
//-------------------------------------------------------------------
// Hardware  : STM32F4 Discovery
// Takt      : 168 MHz
// Sprache   : ARM C
// Datum     : ...
// Version   : ...
// Autor     : ...
//-------------------------------------------------------------------
#include <stddef.h>
#include <stdlib.h>
#include "stm32f4xx.h"
//#include "stm32f30x.h"
//#include "stm32f0xx.h"
#include "init.h"
#include "delay.h"
#include "sysTick.h"
#include "main.h"

int main(void)
{
    SystemInit();
    initApplication();
    GPIO_SetBits(GPIOD,GPIO_Pin_12);
    do{
        // Eingabe
        // Ausgabe
        // Verarbeitung
        GPIO_ToggleBits(GPIOD,GPIO_Pin_12);
        delay(10000000);
        } while (true);
    return 0;
}
```

Unit init.cc

```
#include "init.h"

void initApplication()
{
    SysTick_Config(SystemCoreClock/100);
    // weitere Initialisierungen durchführen
    /* GPIOG Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
    /* Configure PG6 and PG8 in output pushpull mode */
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

Unit init.h

```
#include <stddef.h>
#include "stm32f4xx.h"

void initApplication();
```

Unit delay.cc

```
#include "delay.h"

void delay(vu32 nCount)
{
    while(nCount)
    {
        nCount--;
    }
}
```

Unit delay.h

```
#include <stddef.h>
#include "stm32f4xx.h"

void delay(vu32);
```

Unit sysTick.cc

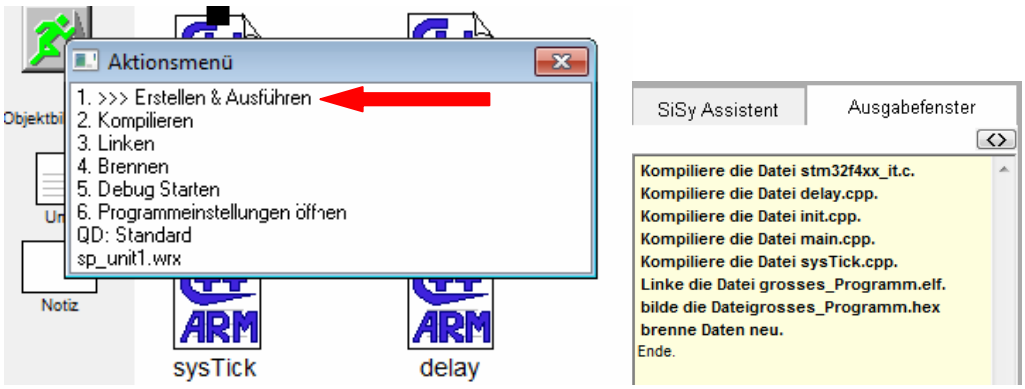
```
#include "sysTick.h"

extern "C" void SysTickFunction(void)
{
    // Application SysTick
    GPIO_ToggleBits(GPIOD,GPIO_Pin_13);
}
```

Unit sysTick.h

```
#include <stddef.h>
#include "stm32f4xx.h"

extern "C" void SysTickFunction();
```

5.4.5 Übersetzen, Brennen und Test

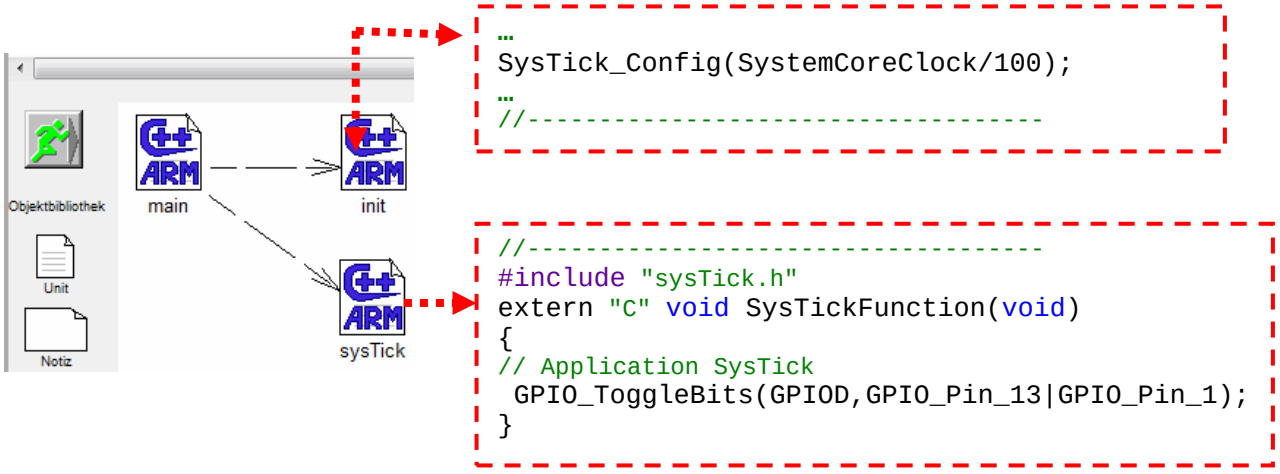
Zum Übersetzen, Brennen und Testen wählen Sie im Aktionsmenü  den entsprechenden Menüpunkt. Im Ausgabefenster erscheint das Protokoll der ausgeführten Aktionen. Des Weiteren öffnet die SiSy-Konsole.



Über das ControlCenter kann die Datenkommunikation mit dem STM32F4-Discovery erfolgen. Überprüfen Sie gegebenenfalls die Einstellungen entsprechend Absatz 10.2, „Das ControlCenter“.

5.4.6 Interrupt-Service-Routine (ISR) im großen Programm

Interrupt-Service-Routinen (im weiteren ISR) sind besondere Formen von Unterprogrammen. Diese werden von einer Interruptquelle des Mikrocontrollers (Timer, ADC, UART, usw.) bei entsprechenden Ereignissen automatisch an beliebiger Stelle im Programmfluss aufgerufen (Unterbrechung, engl. Interrupt). Es ist nötig die Interruptquelle entsprechend zu konfigurieren. Es empfiehlt sich für jedes interruptfähige Gerät eine eigene Unit anzulegen. Diese ist mit der HauptUnit zu verbinden.



Hinweis:
Die Parameter zum Aufruf der Interrupt-Routinen sind controllerabhängig. Sie sind dem jeweiligen Referenzblatt des Controllers zu entnehmen.

6 Entwicklung Programmablaufplan für AVR Programme

Hinweis:
Dieses Beispiel kann mit der Ausgabe SiSy Microcontroller ++ und SiSy AVR erstellt werden.

6.1 Einleitung

Für die Entwicklung eines Programmablaufplans (PAP) sind konkrete Vorstellungen über die Systemlösung und Kenntnis der Hardware nötig. Ein Programmablaufplan kann aus einer genauen Aufgabenstellung abgeleitet werden.
In dieser Beispielanwendung wird die Aufgabenstellung in einem Programmablaufplan modelliert und aus diesem PAP der Quellcode in Assembler generiert. Das aus dem Quellcode erzeugte Programm kann sofort auf den Controller gebrannt werden.

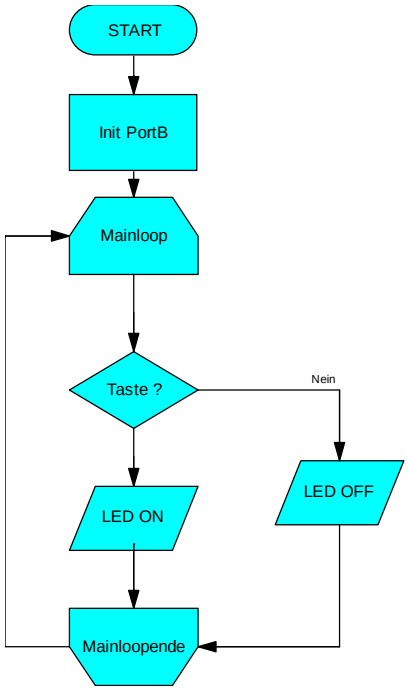
6.2 Einfache Programmentwicklung aus einem PAP

6.2.1 Zielstellung

Es ist im ersten Schritt eine einfache Mikrocontroller-Anwendung mit der Technik des Programmablaufplanes zu entwerfen und in der Sprache Assembler zu realisieren.

Aufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der ein Taster eine LED schaltet.

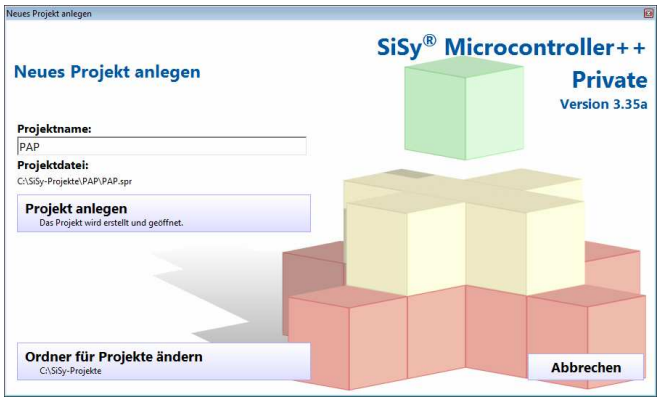
Schaltung:
Port B.0 = Taster 1
Port B.1 = LED

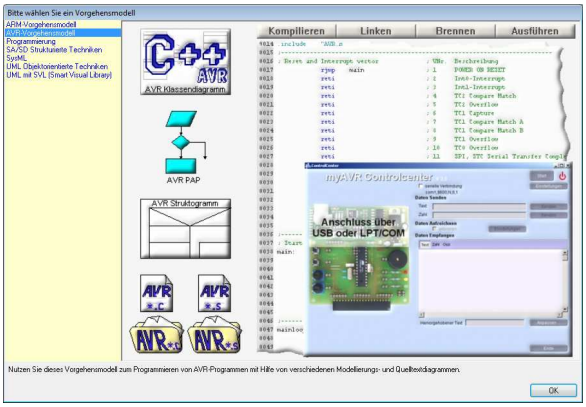


PAP zur Beispielaufgabe

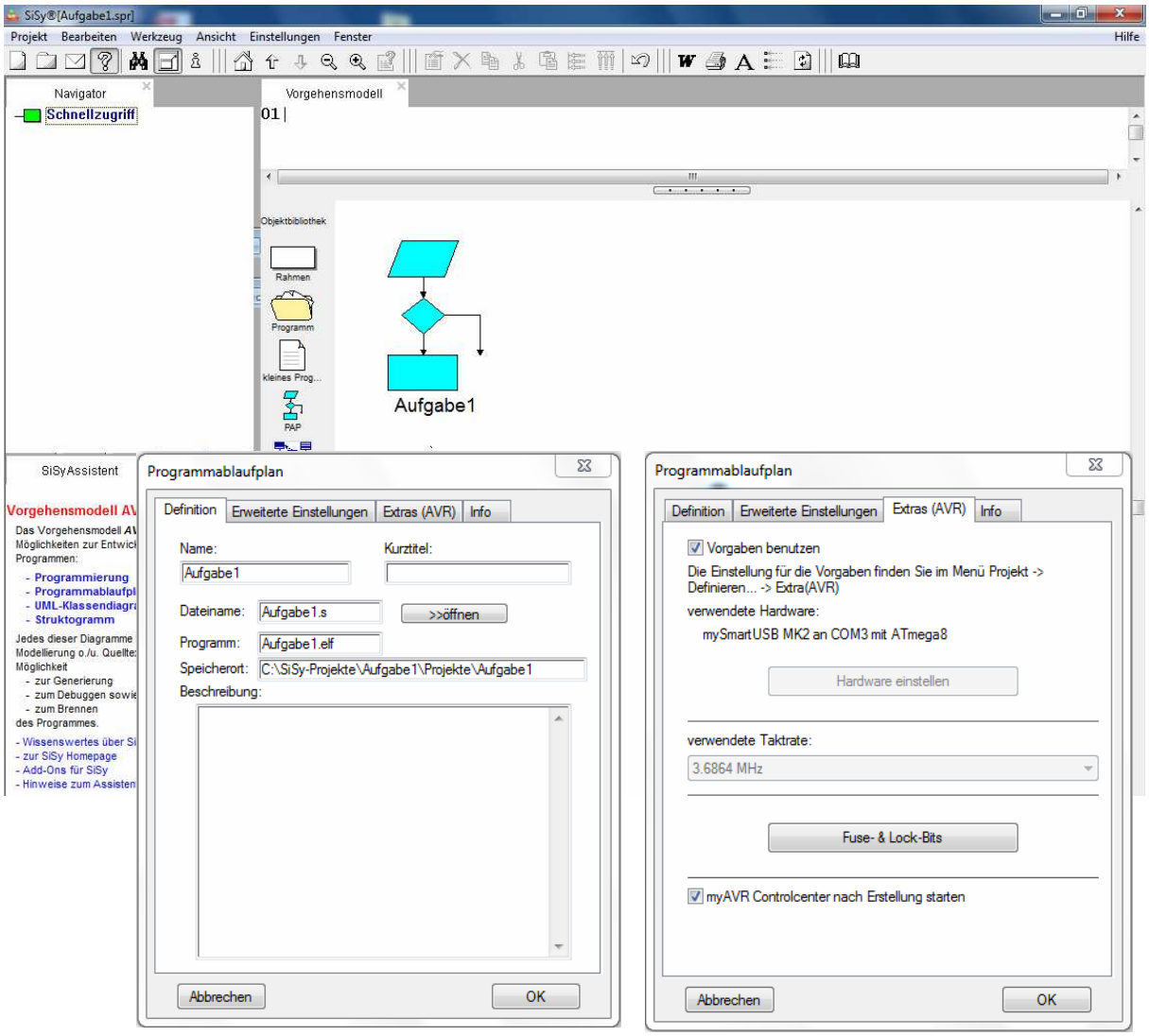
6.2.2 Vorbereitung

Starten Sie SiSy und legen Sie ein neues Projekt an. Wählen Sie das AVR-Vorgehensmodell. Nehmen Sie die Grundeinstellungen für die verwendete AVR Hardware vor oder lassen Sie die myAVR-Hardware automatisch suchen.



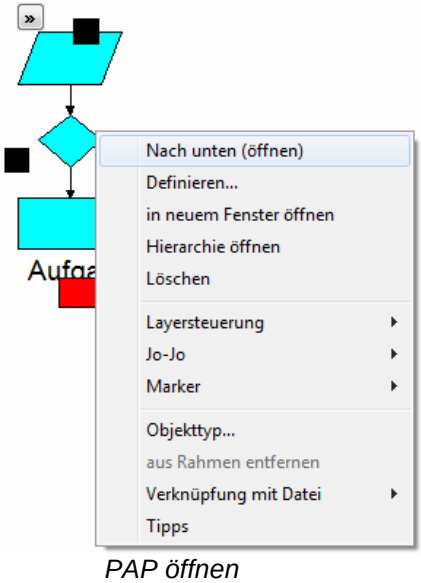


Danach öffnet die typische Benutzeroberfläche von SiSy mit einem leeren Vorgehensmodell und Sie können mit der Arbeit beginnen.
Ziehen Sie als nächstes aus der Objektbibliothek ein Objekt vom Typ „PAP“ in das leere Diagramm. Benennen Sie den PAP mit „Aufgabe1“. Beachten Sie die Einstellungen zum Controllertyp und Programmieradapter unter „Extras (AVR)“; vgl. Abbildung.



Anlegen des Objektes „PAP“ und Einstellungen

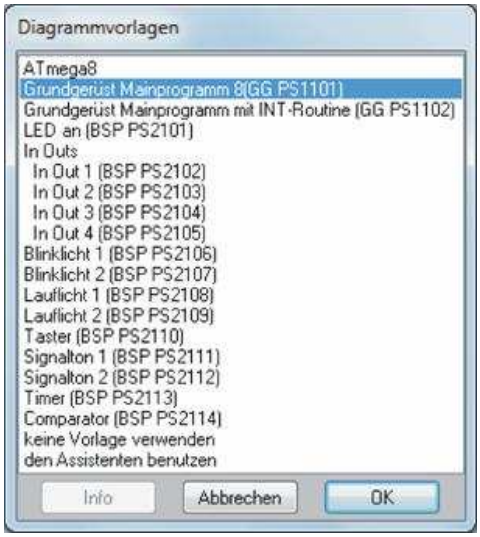
Der nächste Schritt ist das Aufstellen des Programmablaufplanes. Dazu muss das Diagramm unter dem Symbol geöffnet werden. Wählen Sie rechte Maustaste „Nach unten (öffnen)“, um in dieses Diagramm zu gelangen.



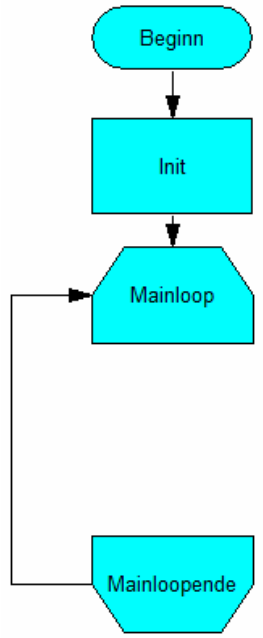
PAP öffnen

6.2.3 Grundstruktur laden

Wenn ein Diagramm leer ist, bietet SiSy typische Vorlagen zum Importieren an. Diese können dann weiterbearbeitet werden.
Wählen Sie die Diagrammvorlage „Grundgerüst Mainprogramm …“.
Die Abbildung *„Grundgerüst PAP“* zeigt den PAP zu diesem Grundgerüst.



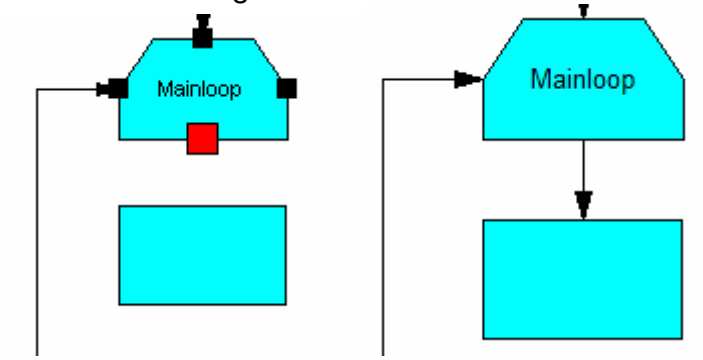
Diagrammvorlagen



Grundgerüst PAP

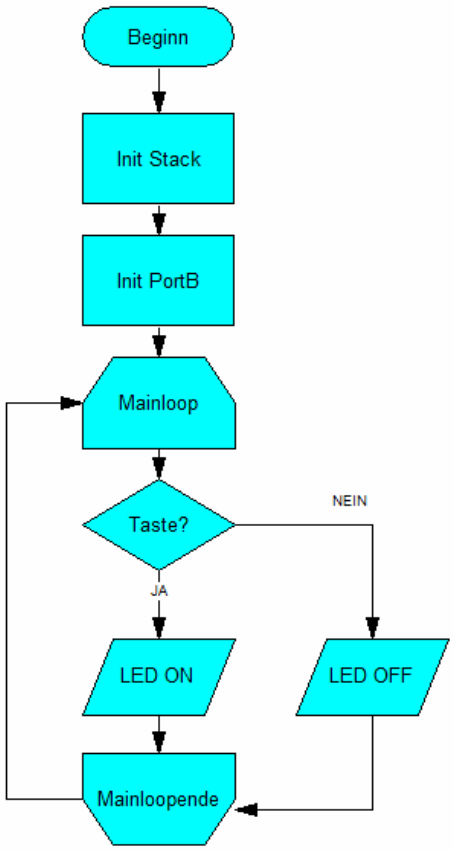
6.2.4 Logik entwerfen

Für die Abbildung der Programmlogik im PAP muss die Vorlage um die fehlenden Elemente ergänzt werden. Des Weiteren sind die Elemente durch gerichtete Verbindungen (Kanten) in der Reihenfolge ihrer Abarbeitung zu verbinden.
Ein Objekt wird im Diagramm ergänzt, indem der entsprechende Objekttyp in der Objektbibliothek mit der Maus ausgewählt und per Drag & Drop an die entsprechende Position im Diagramm gezogen wird.
Verbindungen zwischen den Objekten können über den rot markierten Verteiler von selektierten Objekten hergestellt werden. Dazu ist das Ausgangsobjekt zu selektieren und mit dem Mauscursor von dem roten Verteiler bei gedrückter linker Maustaste eine Verbindung zum Zielobjekt zu ziehen. Für die Benennung der Objekte öffnen Sie den Definieren-Dialog aus dem Kontextmenü.



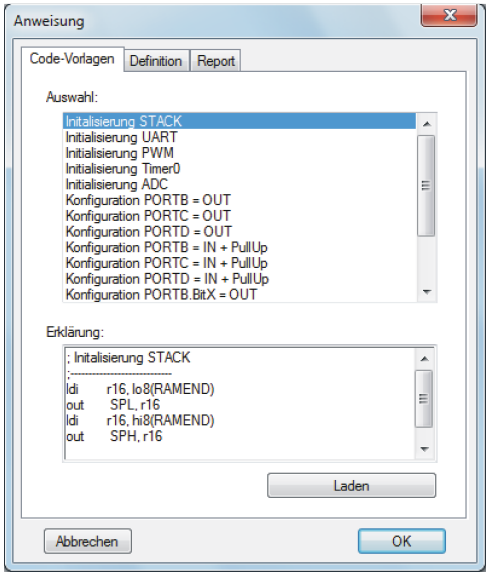
Objekte in das Diagramm einfügen und verbinden

Zeichnen Sie den nebenstehenden Programmablaufplan (vgl. Abbildung „Logikentwurf im PAP“):

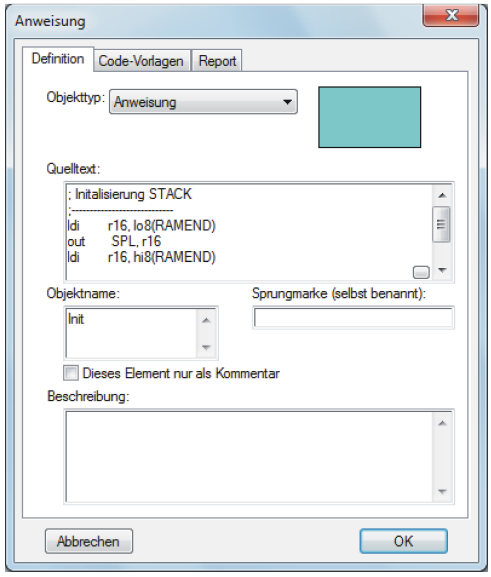


6.2.5 Befehle eingeben

Nachfolgend soll aus dem Programmablaufplan Assemblerquellcode generiert werden. Dazu ist es nötig, die einzelnen Elemente des PAP mit den entsprechenden Assembleranweisungen zu versehen. Dafür gibt es mehrere Möglichkeiten. Zum einen bietet SiSy beim ersten Öffnen eines jeden Elementes typische Code-Vorlagen an, die über die Schaltfläche „Laden“ dem Element zugewiesen werden können. Wird der Definieren-Dialog mit „OK“ beendet, so wird die Auswahl im Objekt gespeichert und beim nächsten Aufruf des Dialoges „Definieren“ erscheinen die Code-Vorlagen nicht mehr; das Element kann ganz normal bearbeitet werden. In den beiden folgenden Abbildungen sind beide Varianten des Dialoges „Definieren“ zu sehen.

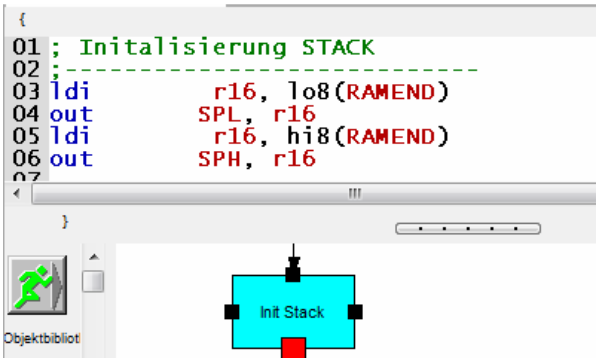


Codevorlage PAP

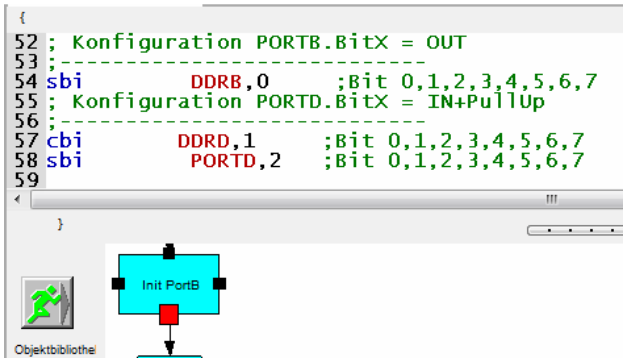


Definition PAP-Element

Die zweite Möglichkeit besteht beim Selektieren von Elementen über den Quellcode-editor oberhalb des Diagrammfensters, vgl. Abbildung „Quellcodefenster im PAP“ und Abbildung „Quellcodeeingabe PAP“.



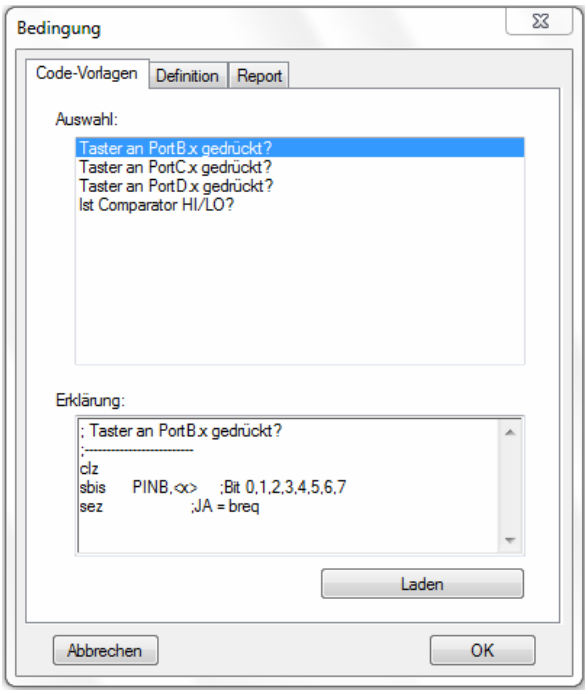
Quellcodefenster im PAP



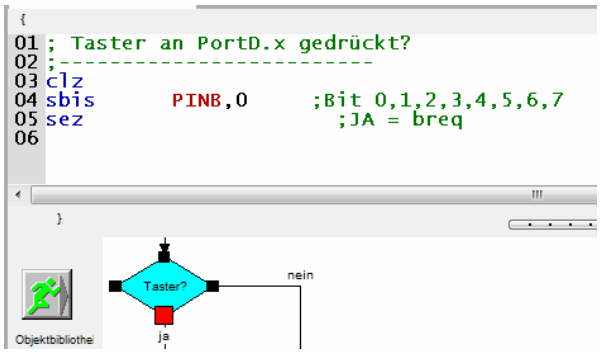
Quellcodeeingabe PAP

Geben Sie die gezeigten Quellcodes in die Objekte ein!

Bedingungen haben spezielle Vorlagen, die eine Codegenerierung bei übersichtlichem Programmablaufplan vereinfachen.
Jede Bedingungsvorlage ist so konstruiert, dass eine JA/NEIN Entscheidung erzeugt werden kann. Findet der Codegenerator das Schlüsselwort JA an einer der folgenden Verbindungen, setzt er diese in eine Sprunganweisung `breq` um. Das Schlüsselwort NEIN wird in `brne` umgewandelt. Alternativ können statt dieser Schlüsselworte auch der Sprungbefehl selber an eine der Kanten geschrieben werden (`breq, brne, brge, brlo`, usw.)

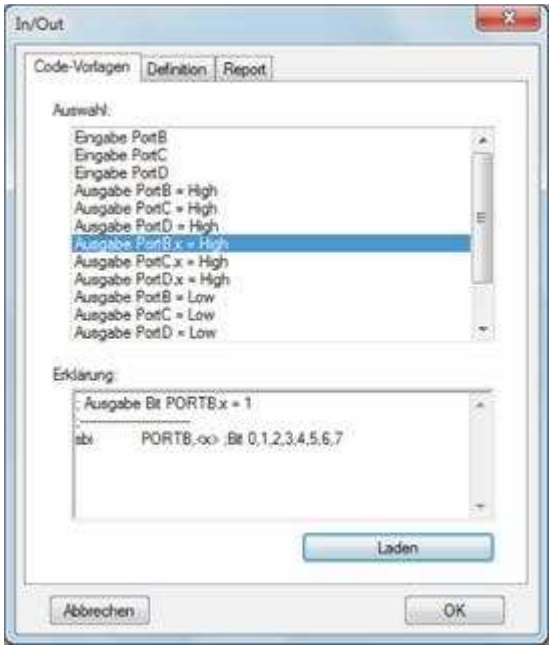


Codevorlage Bedingung



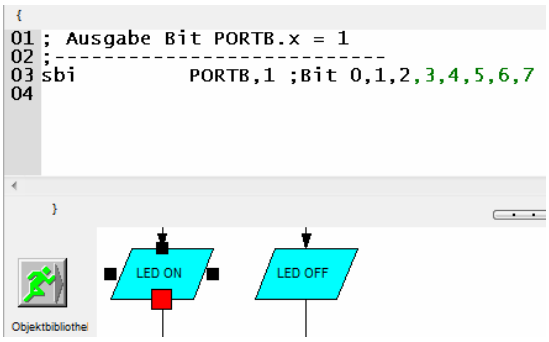
Quellcode einer Bedingung

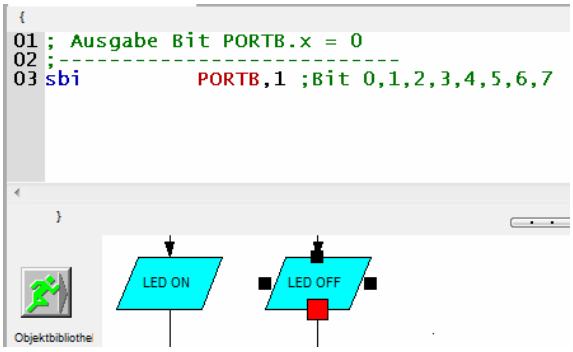
Auch das Element „IN/OUT“ verfügt über spezifische Vorlagen. Diese sind gegebenenfalls mit zu ergänzen. Dazu sind spitze Klammern als Platzhalter in den Vorlagen eingefügt.



Vorlagen für IN/OUT

Ergänzen Sie den Quellcode der gezeigten Elemente!

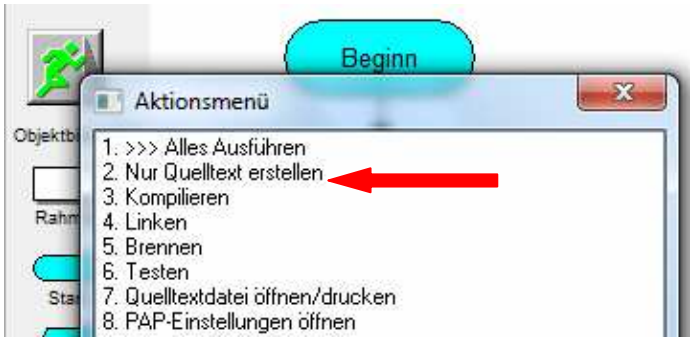




6.2.6 Übersetzen, Brennen und Test

Sind alle betreffenden Elemente mit Quellcode hinterlegt, kann aus dem Programmablaufplan der komplette Quellcode generiert, kompiliert, gelinkt und auf den Mikrocontroller übertragen werden. Die gewünschte Funktion kann aus dem Aktionsmenü ausgewählt werden.

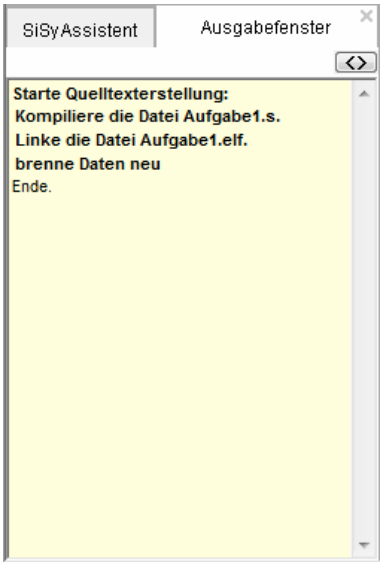
Hinweis:
Beachten Sie, dass für das Brennen des Controllers das Programmierkabel angeschlossen sein muss und bei Bedarf eine geeignete Spannungsquelle anzuschließen ist.



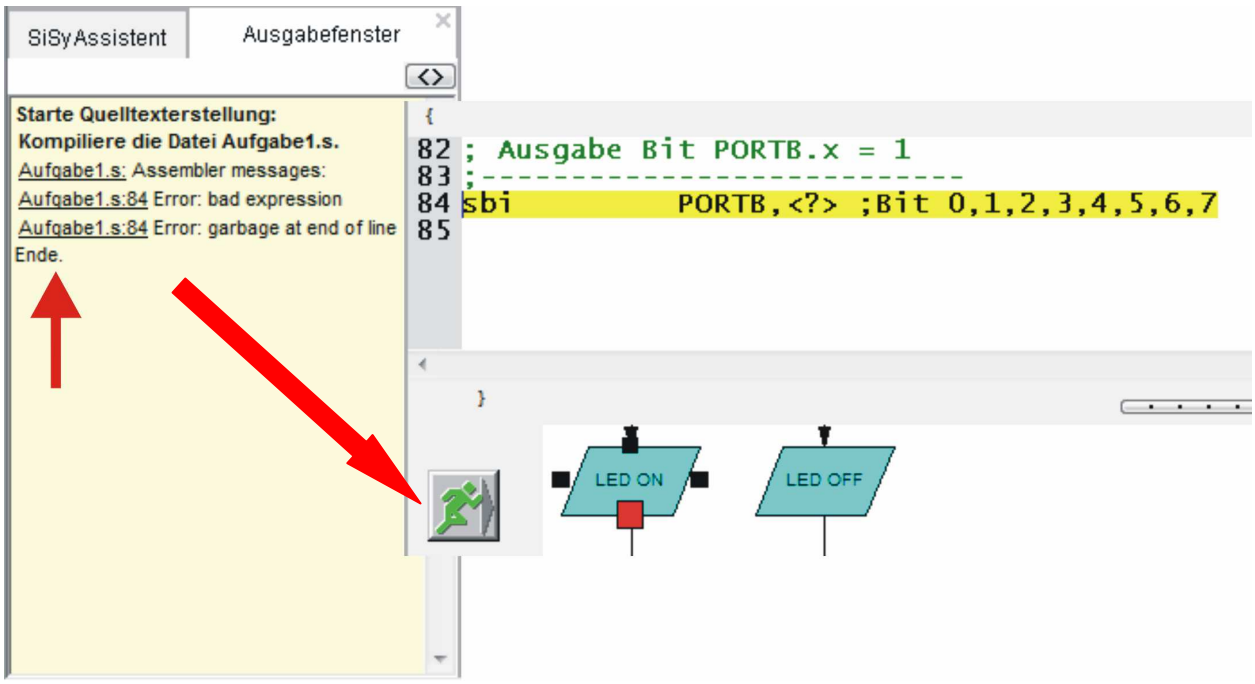
Auswahl aus Aktionsmenü

Funktionalitäten des Aktionsmenüs	
Benennung	**Funktion**
Alles Ausführen	Quellcode generieren, kompilieren, linken, brennen
Nur Quelltext erstellen	Quellcode generieren mit allen Marken
Kompilieren	Quellcode zu Objektdatei übersetzen
Linken	Objektdatei zu ELF-Datei binden
Brennen	ELF-Datei an den Controller übertragen
Testen	ControlCenter öffnen
Quelltextdatei öffnen	Quellcodedatei öffnen
Quelltext bereinigen	Quellcode von überflüssigen Marken bereinigen

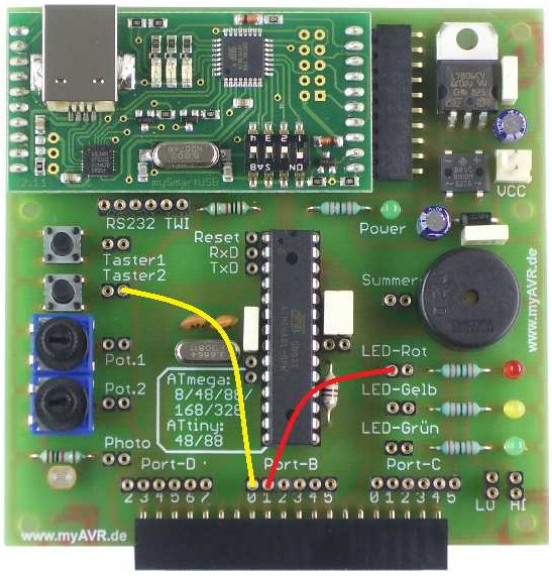
Im Ausgabefenster werden die jeweiligen Aktionen angezeigt. Bei Compilerfehlern werden diese ebenfalls im Ausgabefenster mit der entsprechenden Zeilennummer angezeigt. Um zu dem Fehler zu gelangen, genügt meist ein Klick auf die Fehlermeldung. Das betreffende Objekt wird selektiert und die Zeile hervorgehoben.



Ausgabefenster



Fehlerbehandlung



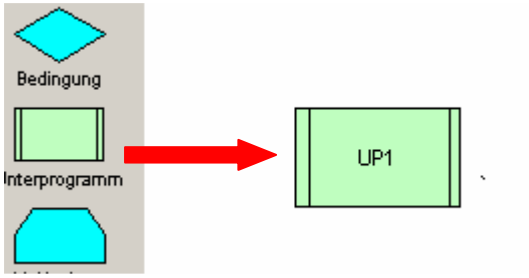
Nachdem das Programm erfolgreich übersetzt und auf den Controller übertragen wurde, kann die Anwendung getestet werden. Stecken Sie auf dem Board die vorgegebenen Verbindungen, nutzen Sie die Patchkabel.

6.3 Unterprogrammtechnik im PAP

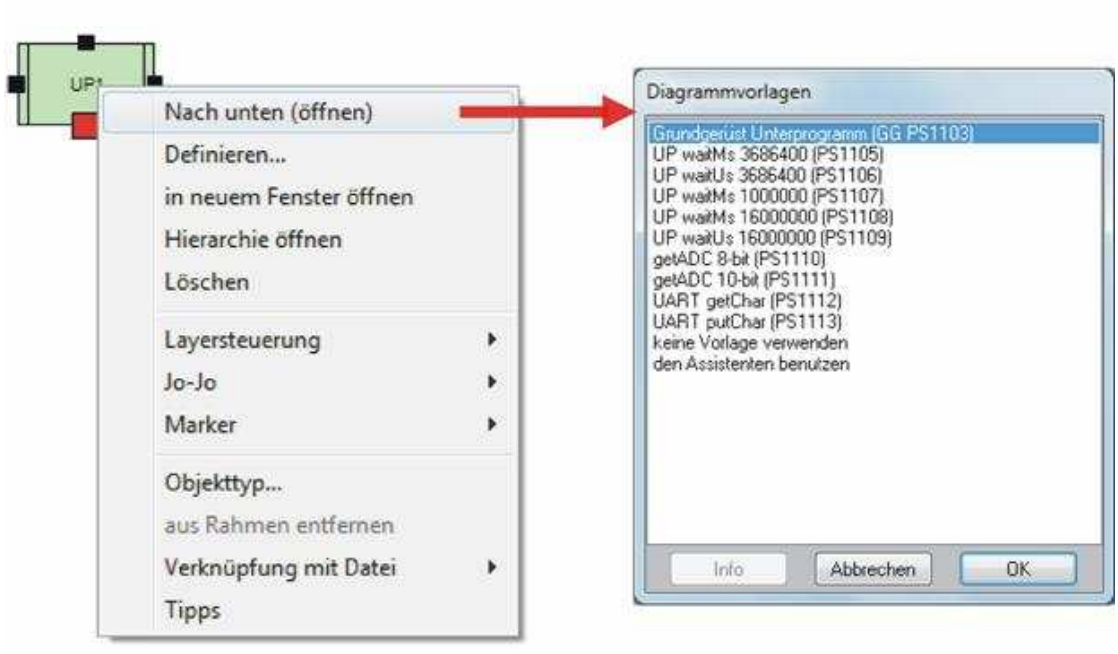
Unterprogramme sind ein wichtiges Gestaltungsmittel für übersichtliche Mikrocontrollerprogramme. Sie werden für in sich abgeschlossene Aufgaben (Verarbeitungsschritte) benutzt, die auch mehrfach im Gesamtprogramm genutzt werden können.

6.3.1 Anlegen eines Unterprogramms

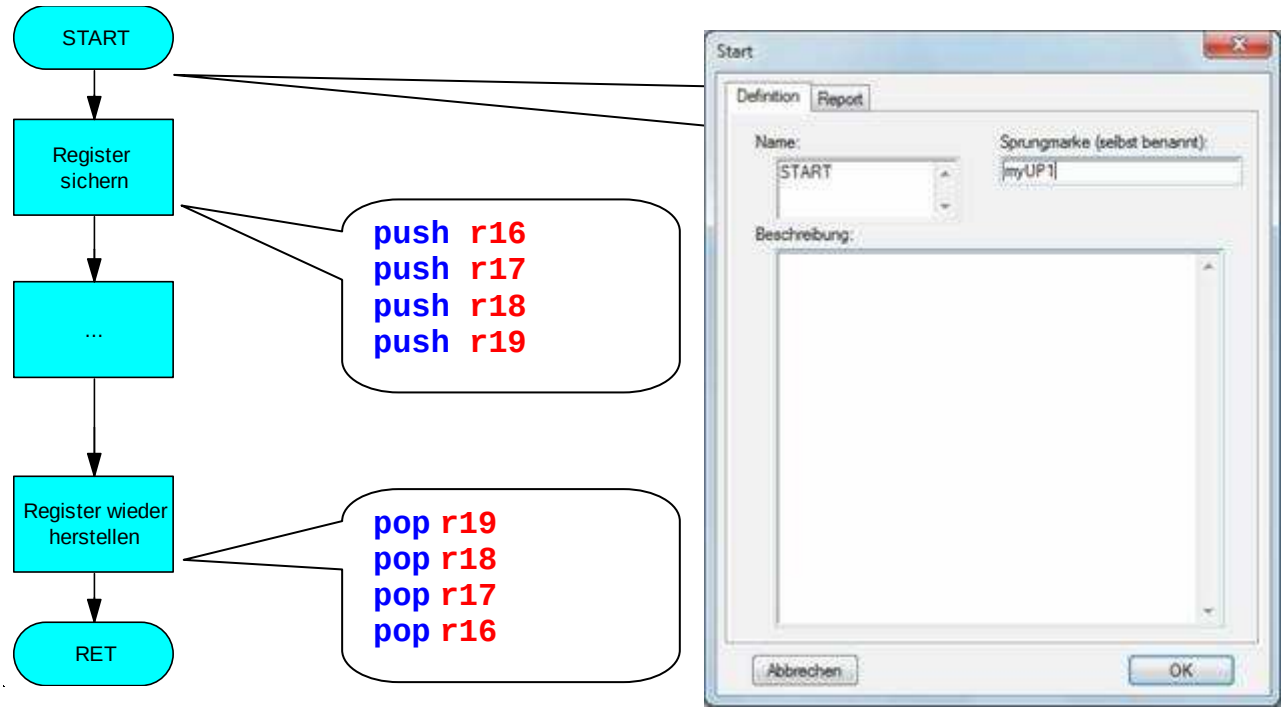
Ziehen Sie den Objekttyp „Unterprogramm“ aus der Objektbibliothek in das gewünschte Diagramm. Mit Doppelklick oder über rechte Maustaste -> Kontextmenü -> Definieren auf dem Element können Sie dem Unterprogramm einen Namen geben.



Damit ist ein Objekt angelegt, welches im aktuellen Diagramm als Aufruf (`call`) des Unterprogramms zu verstehen ist. Die Funktionalität des Unterprogramms wird in einem gesonderten Programmablaufplan für das Unterprogramm entworfen. Dazu ist das Diagramm „unter“ bzw. „hinter“ dem Objekt Unterprogramm zu öffnen. Um das zum Unterprogramm zugehörige Diagramm zu öffnen, wählen Sie auf dem Objekt rechte Maustaste -> Kontextmenü -> Nach unten (öffnen).

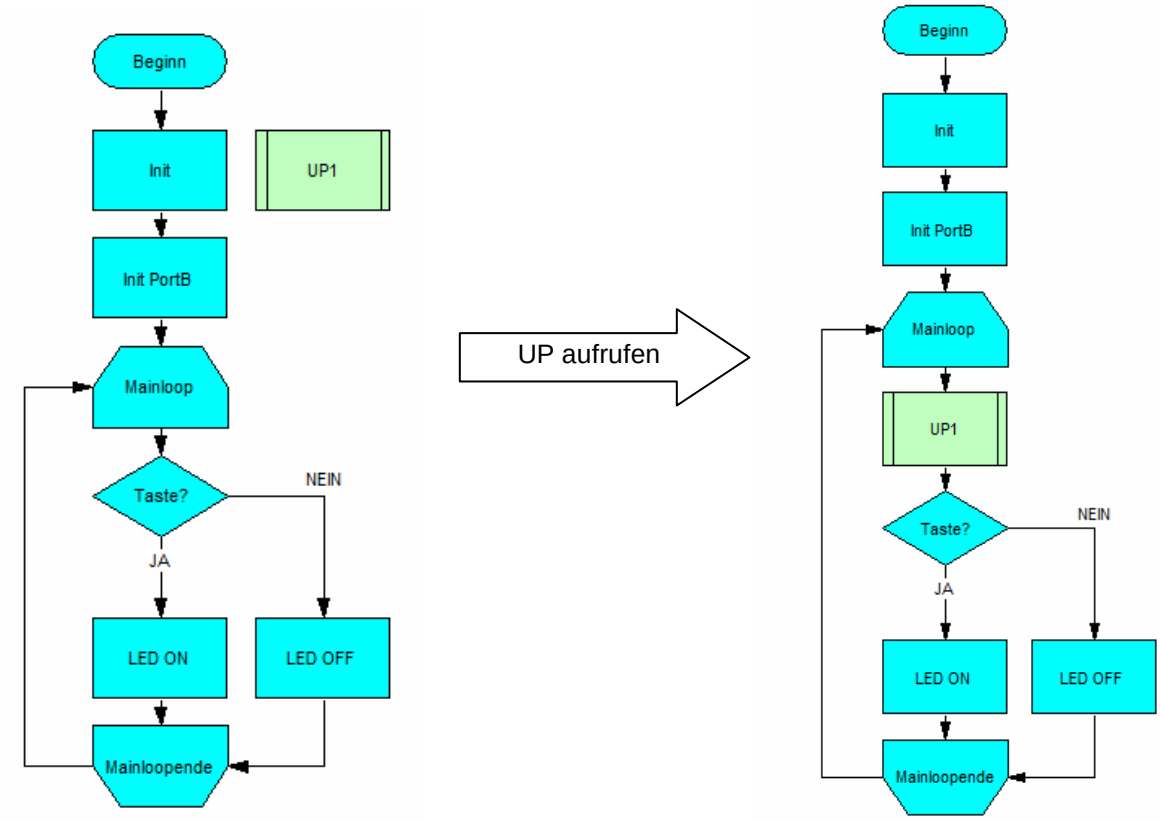


Sie erhalten eine Vorlagenliste für die Grundstruktur von Unterprogrammen. Bitte laden Sie die Vorlage „Grundgerüst Unterprogramm“. Auf dem Objekt „START“ können Sie eine benutzerdefinierte Sprungmarke festlegen (Rechtsklick -> Definieren), die durch den Codegenerator erstellt und verwendet werden soll. Die Vorlage muss entsprechend der vorgesehen Logik abgeändert werden.



6.3.2 Ein Unterprogramm aufrufen

Das Unterprogrammsymbol muss zum Aufruf an der entsprechenden Position im Programmablaufplan eingefügt werden. Der Codegenerator erzeugt dann entsprechend einen Unterprogrammaufruf und den Code für das Unterprogramm selbst. Dazu ist in das Diagramm zurückzukehren, in dem das Objekt „Unterprogramm“ angelegt wurde (zum Beispiel rechte Maustaste -> Kontextmenü-> nach oben…).
Das Unterprogramm ist korrekt eingebunden, wenn es vollständig und eindeutig im Programmfluss integriert ist (mindestens ein eingehender Pfeil und genau ein ausgehender Pfeil).

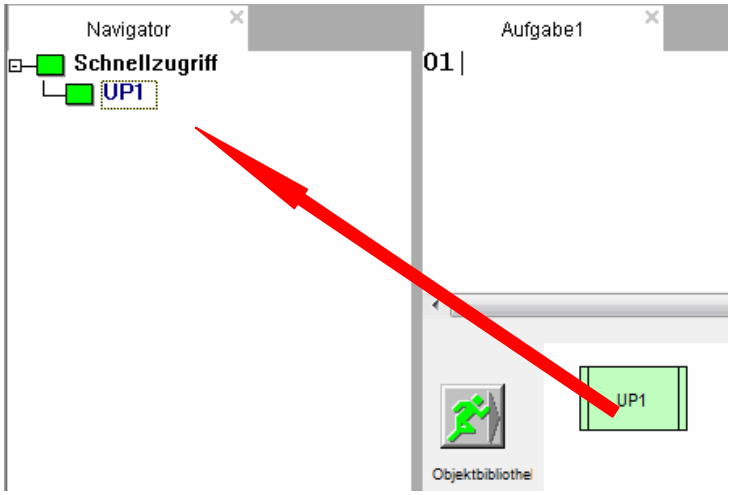


6.3.3 Unterprogramme mehrmals benutzen

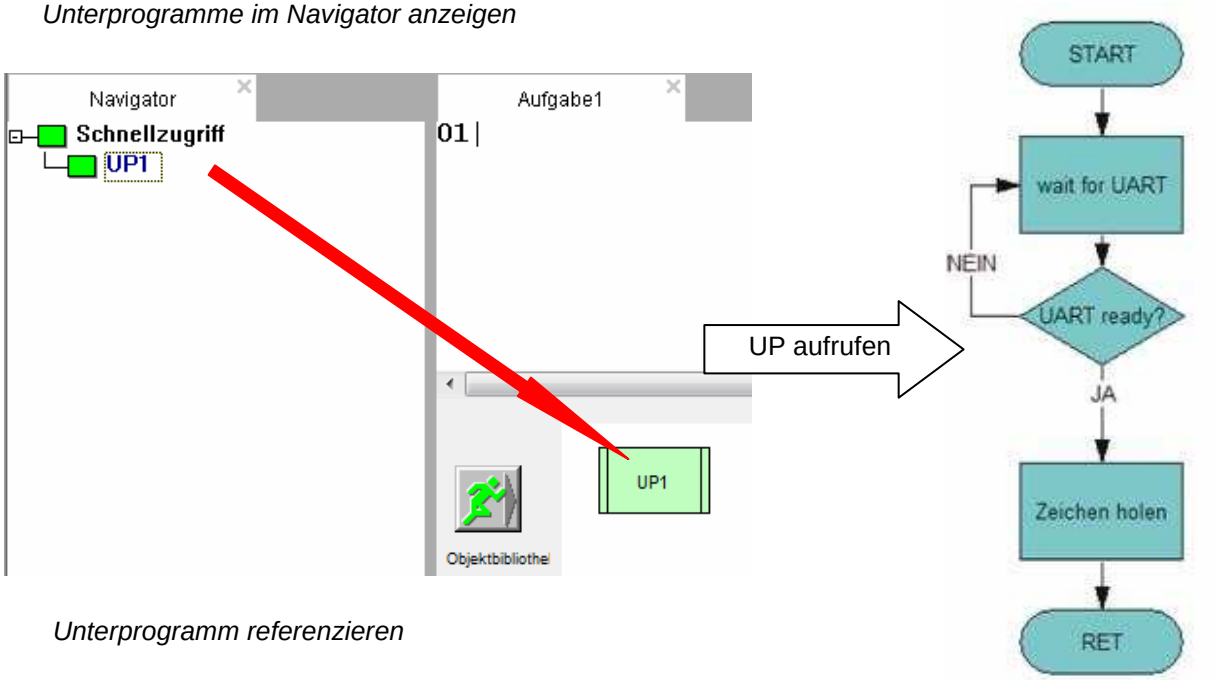
Ein wesentliches Merkmal von Unterprogrammen ist, dass diese von verschiedenen Stellen im Programm aufgerufen (`call`) werden können und auch dorthin zurückkehren (`return`). Um diese Möglichkeit zu nutzen, bietet SiSy das Anlegen von Referenzen. Vergleichen Sie dazu Absatz 3.2.

Um ein Unterprogramm zu referenzieren (wiederholend zeigen und einbinden) gehen Sie wie folgt vor:

1. zeigen Sie im Navigator das gewünschte Unterprogramm an
   a. über den Schnellzugriff,
      dort lässt sich das Original per Drag & Drop ablegen, oder
   b. über die Navigatorsortierung „Unterprogramme“
      Navigator -> rechte Maustaste -> Kontextmenü -> Programmablaufpläne
2. Öffen Sie das gewünschte Zieldiagramm, in dem das Unterprogramm verwendet werden soll
3. ziehen Sie per Drag und Drop das Unterprogramm in das Zieldiagramm, dabei wird nur eine Referenz (Link) auf das Original angelegt.
4. Integrieren Sie Referenz wie oben beschrieben in den Programmfluss



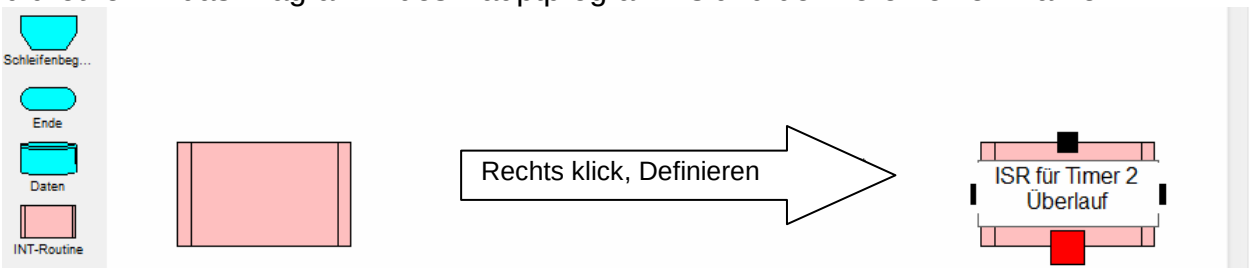
Unterprogramme im Navigator anzeigen



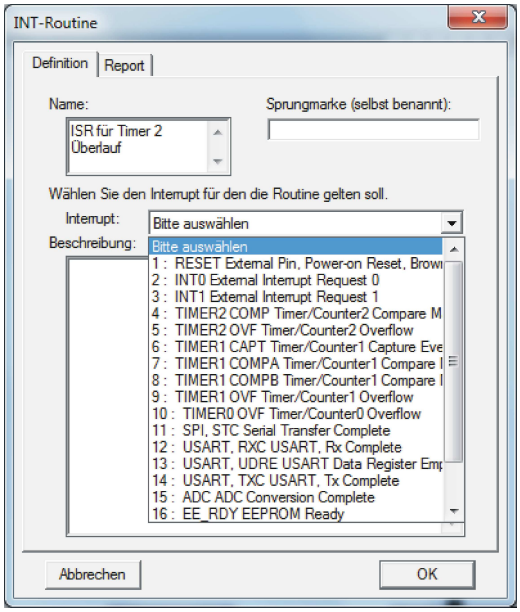
Unterprogramm referenzieren

6.4 Interrupt-Service-Routinen (ISR) im PAP

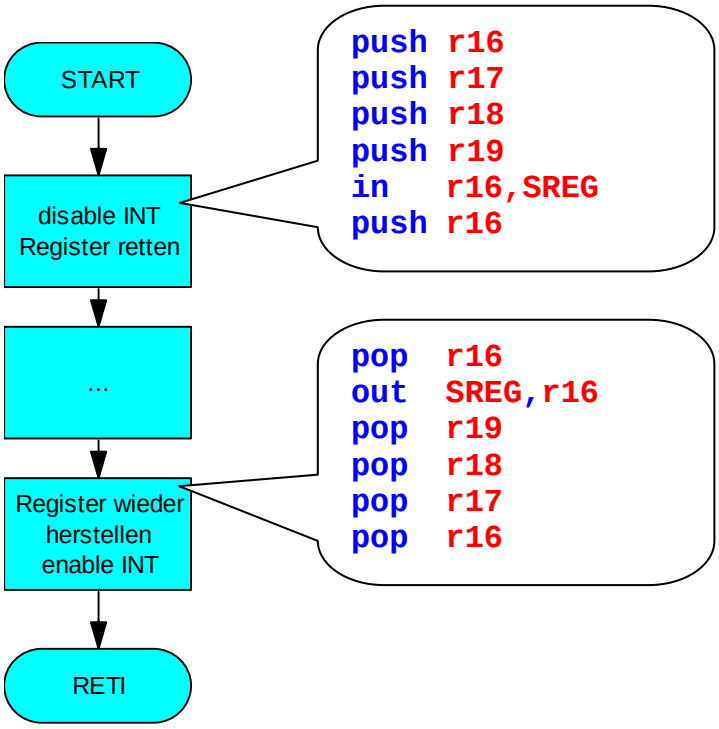
Interrupt-Service-Routinen (im weiteren ISR) sind besondere Formen von Unterprogrammen. Diese werden von einer Interruptquelle des Mikrocontrollers (Timer, ADC, UART, usw.) bei entsprechenden Ereignissen automatisch an beliebiger Stelle im Programmfluss aufgerufen (Unterbrechung, engl. Interrupt). Es ist demzufolge nicht vorgesehen, eine ISR in den Programmfluss zu integrieren.
Um eine ISR zu erzeugen, ziehen Sie ein Objekt vom Typ INT-Routine aus der Objektbibliothek in das Diagramm des Hauptprogramms und definieren einen Namen.

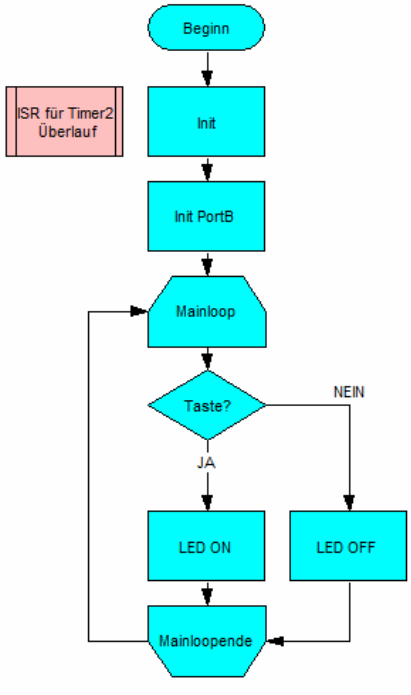


Über rechte Maustaste -> Kontextmenü -> Definieren … legen Sie die Sprungmarke und den Typ des Interrupts fest. Damit erfolgt durch den Codegenerator die Zuordnung der ISR zum entsprechenden Interruptvektor. Beachten Sie, dass die Liste der Interrupts abhängig vom gewählten Controllertyp ist.



Zum Entwerfen der ISR-Logik wählen Sie auf dem Objekt rechte Maustaste -> Kontextmenü -> nach unten …
Ihnen wird das Grundgerüst einer ISR als Diagrammvorlage angeboten. Laden Sie die Diagrammvorlage.
Vervollständigen Sie danach die ISR-Logik. Die ISR wird nicht in den Programmfluss integriert.



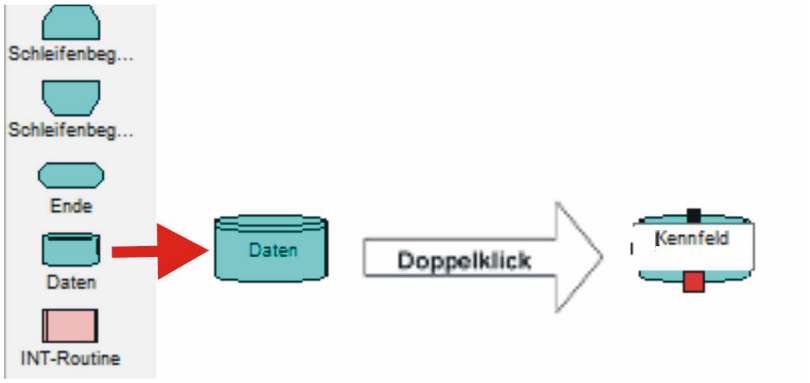


6.5 Daten im PAP

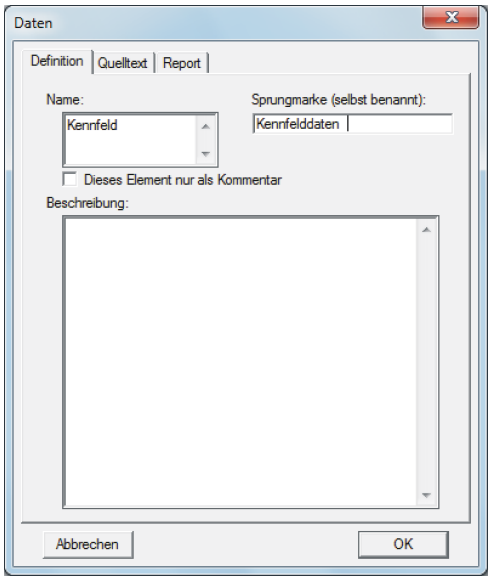
Konstante Daten, die im Programmspeicher (FLASH) des Mikrocontrollers abgelegt werden, können als Datenobjekt deklariert werden. Die im Datenobjekt deklarierten Daten werden durch den Codegenerator immer ans Ende des Quellcodes gestellt. Die zu generierende Marke/Marken für die Datenelemente können vom Entwickler frei gewählt werden.

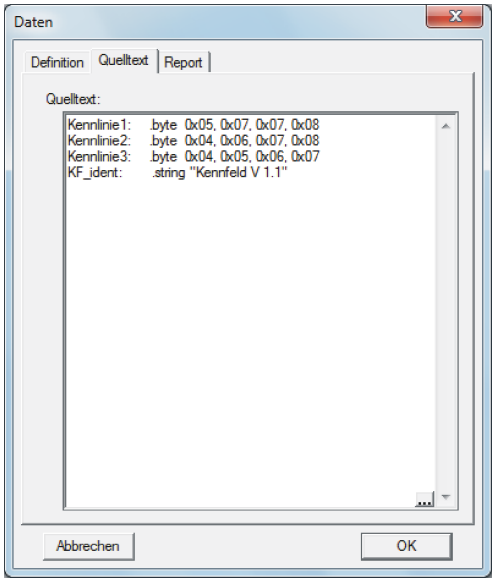
6.5.1 Anlegen eines Datenobjektes

Zum Anlegen eines Datenobjektes ziehen Sie das betreffende Objekt per Drag & Drop aus der Objektbibliothek in das gewünschte Diagramm. Per Doppelklick können Sie einen Namen vergeben (Beachte: Menüfolge *Einstellungen/Menü bei Doppelklick*).



Die Marke und die Daten selbst können über rechte Maustaste -> Kontextmenü -> Definieren… festgelegt werden.





6.5.2 Datenobjekt benutzen

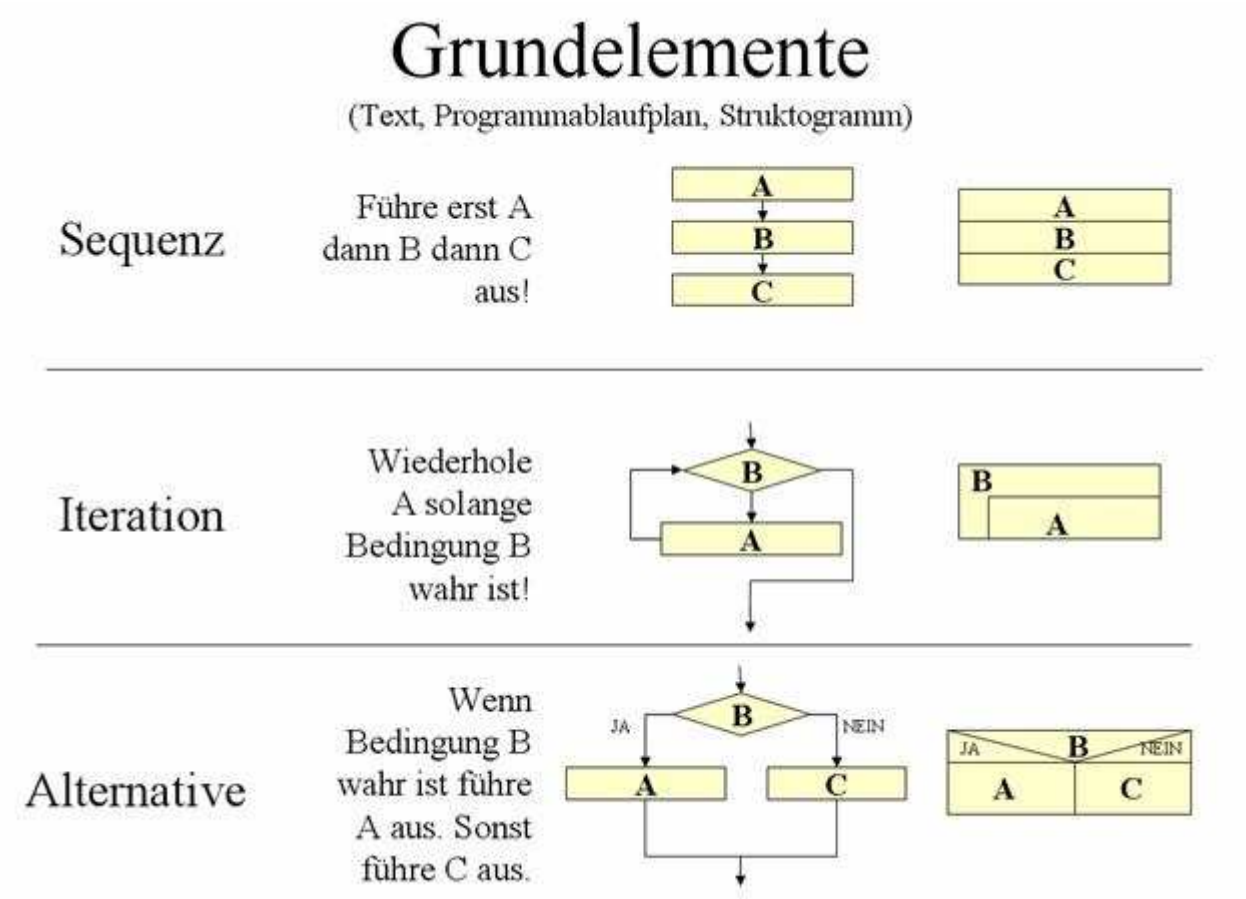
Im Quellcode werden die Daten über die vergegebenen Markennamen angesprochen.

```
{
01 ; Kennfelddaten in Adressregister Laden
02 ldi  r30,lo8 (Kennlinie1)
03 ldi  r31,hi8 (Kennlinie2)
04 ...
05
06
}
```

7 Programmentwicklung aus einem Struktogramm

7.1 Einleitung

Struktogramme (SG) oder auch Nassi-Shneiderman-Diagramme sind ein Entwurfsmittel der strukturierten Programmierung. Strukturiert meint in dem Fall, dass zur Realisierung eines Algorithmus auf das Verwenden von Sprunganweisungen (Goto, Jump) verzichtet wird. Für das Formulieren eines Algorithmus stehen dem Entwickler drei normierte Grundstrukturen zur Verfügung: Sequenz (Folge von Anweisungen), Alternative (Auswahl bzw. bedingte Anweisung), Iteration (Schleife, wiederholte Anweisung).

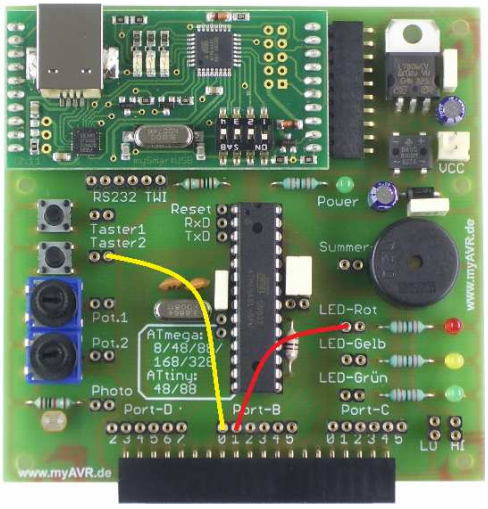


Struktogramme werden als Darstellungsmittel für strukturierte Sprachen wie C oder PASCAL verwendet, da hier im Gegensatz zu Assembler in der Regel auf Sprunganweisungen verzichtet wird.

7.2 Vorgehen für AVR Programme

Die Lösung der folgenden Aufgabe soll die Arbeitsweise und den Umgang mit dem Struktogrammeditor in SiSy verdeutlichen. Die Programmiersprache ist C, Referenzhardware ist ein myAVR Board MK2

Hinweis:
Dieses Beispiel kann mit der Ausgabe SiSy Microcontroller ++ und SiSy AVR erstellt werden.



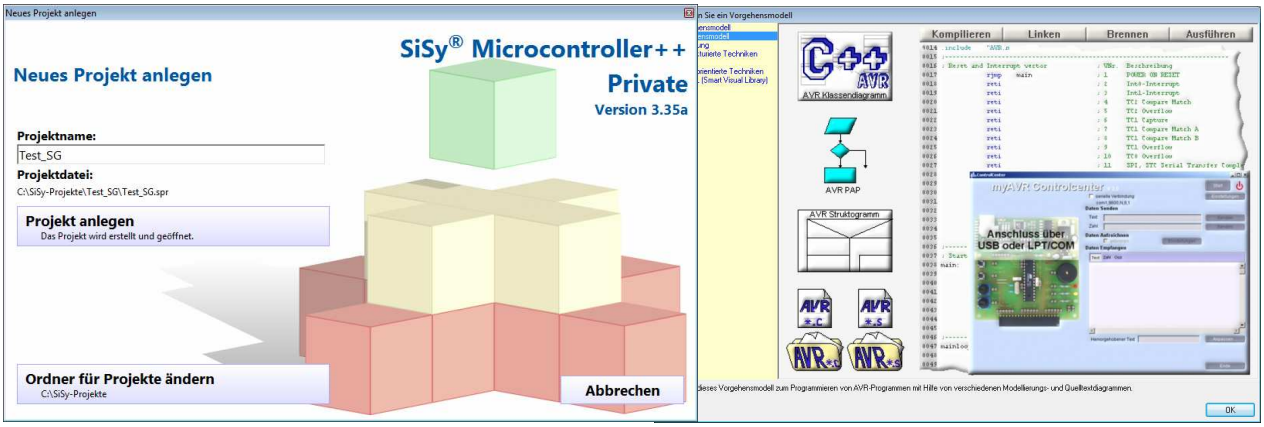
7.2.1 Zielstellung

Es soll eine Mikrocontrollerlösung entwickelt werden, bei der auf Tastendruck eine LED eingeschaltet wird.

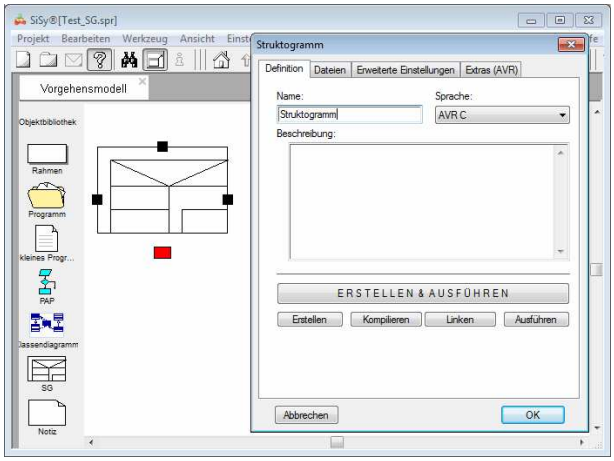
Schaltung:
Port B.0 = Taster
Port B.1 = LED

7.2.2 Vorbereitung

Das Struktogramm kann je SiSy-Projekt ein Hauptprogramm verwalten. Es ist also nötig, für jedes neue Struktogramm-Projekt auch ein neues SiSy-Projekt zu erzeugen. Starten Sie ggf. SiSy.
Erstellen Sie ein neues SiSy-Projekt mit dem Namen „Test_SG“. Wählen Sie das Vorgehensmodell „AVR-Vorgehensmodell“.

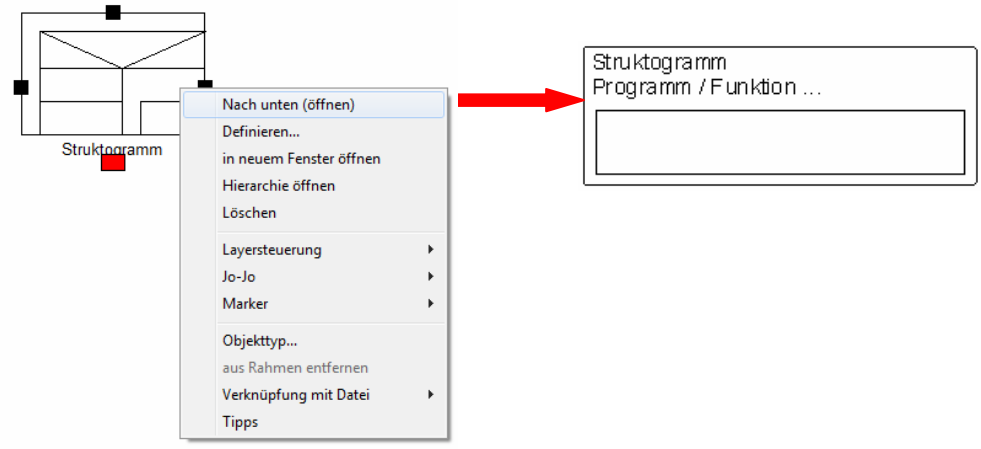


Legen Sie im weiteren Verlauf die AVR-Grundeinstellungen für das Projekt fest. Arbeiten Sie ohne Diagrammvorlage weiter.
Ziehen Sie ein Objekt vom Typ „SG“ (Struktogramm) per Drag & Drop in das Diagrammfenster. Legen Sie den Namen für das Struktogramm fest und die Sprache „AVR C“.



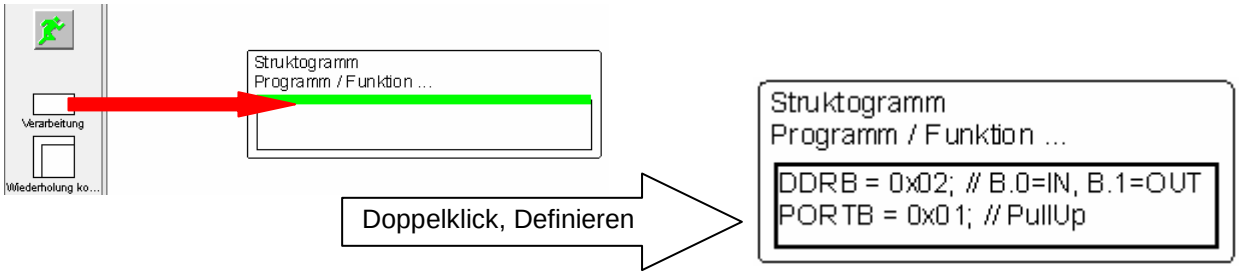
7.2.3 Struktogramm entwickeln

Für die Entwicklung des Struktogramms, muss das Struktogrammfenster geöffnet werden; rechte Maustaste -> Kontextmenü -> Nach unten (öffnen).

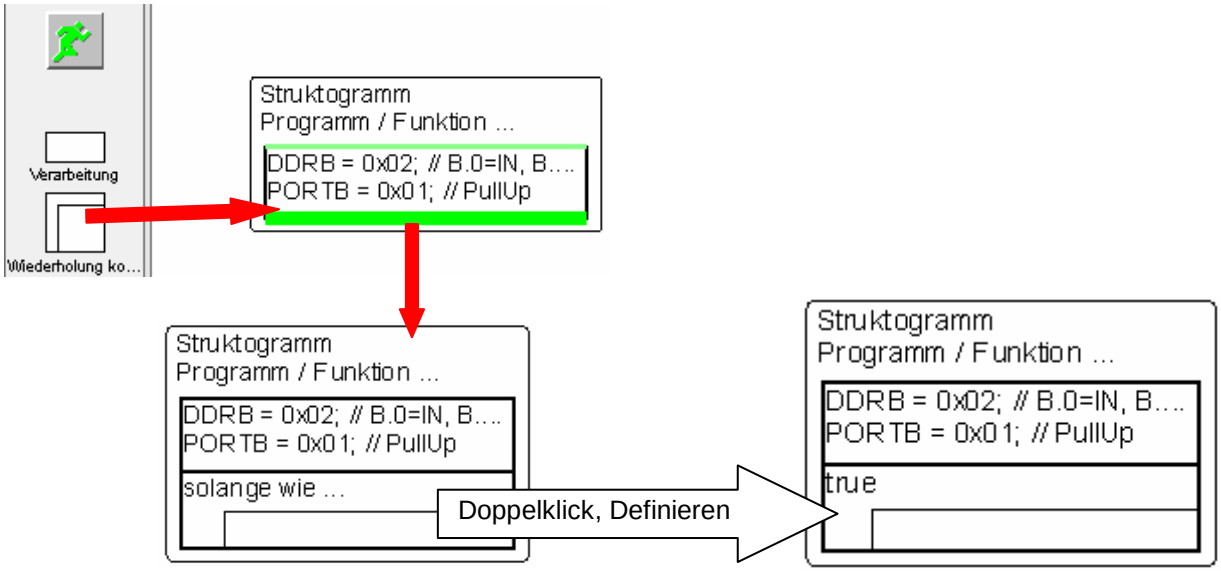


Beim Einfügen einzelner Struktogramm-Elemente in das Struktogramm sind die möglichen Positionen grün unterlegt. Bewegen Sie den Mauszeiger an die gewünschte Stelle, die grüne Linie wird zur Kontrolle breiter.
Führen Sie folgende Arbeitsschritte aus, um den Algorithmus für die oben genannte Aufgabe zu entwerfen:

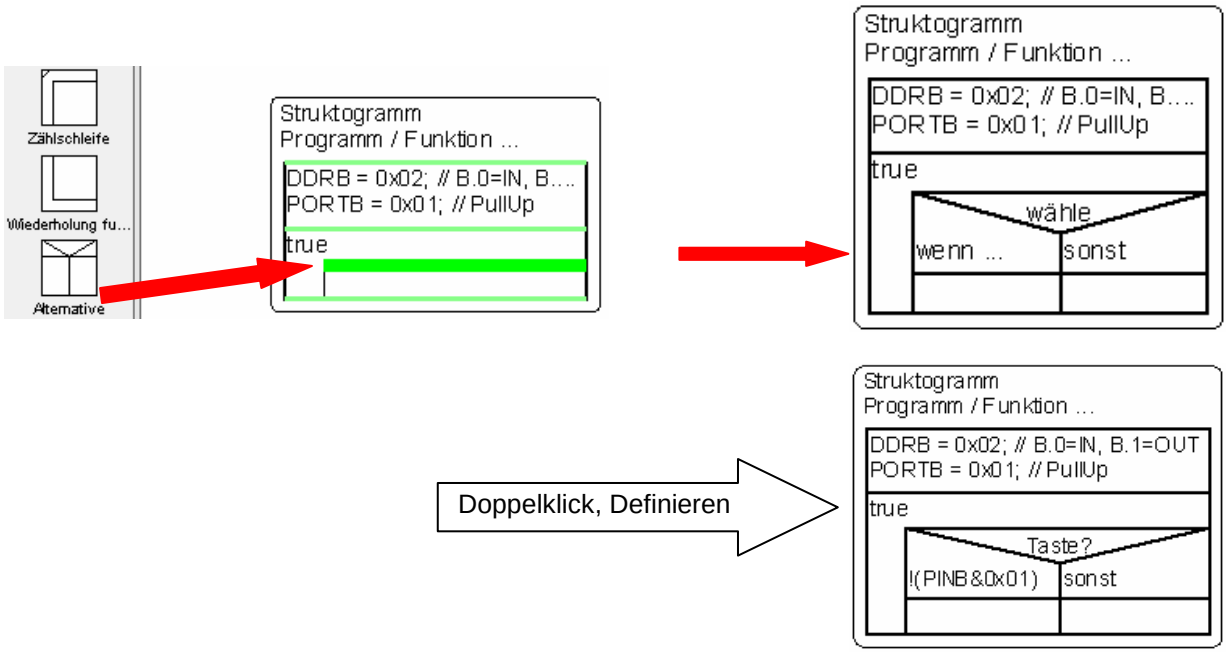
1. Konfigurieren Sie Port B 0 als Eingang und Port B 1 als Ausgang. Ziehen Sie dafür eine „Verarbeitungsroutine“ aus der Objektbibliothek in das Struktogramm.



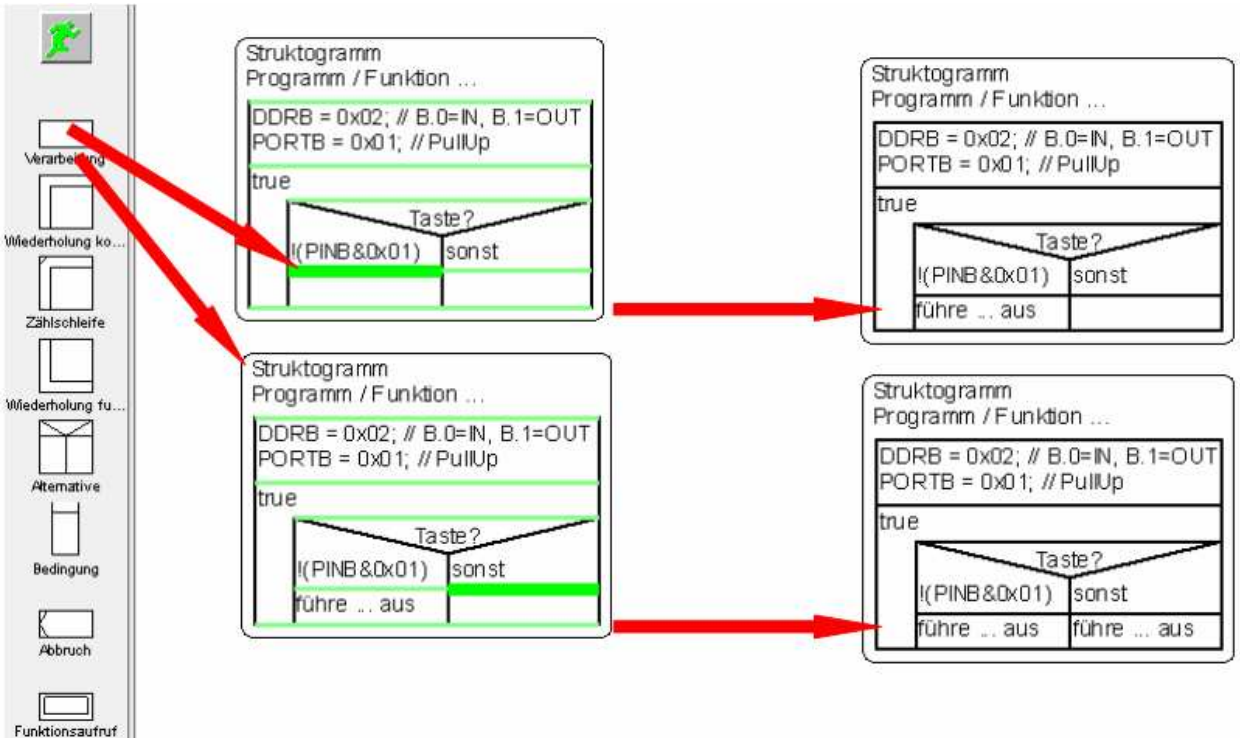
2. Fügen Sie eine „WHILE-Schleife“ ein.



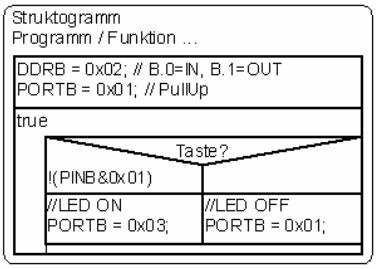
3. Ergänzen Sie die `Mainloop` durch eine „Alternative“ per Drag & Drop. Achten Sie darauf, dass die Alternative innerhalb der `Mainloop` liegt. Geben Sie den Titel und die Bedingung für die Alternative ein.



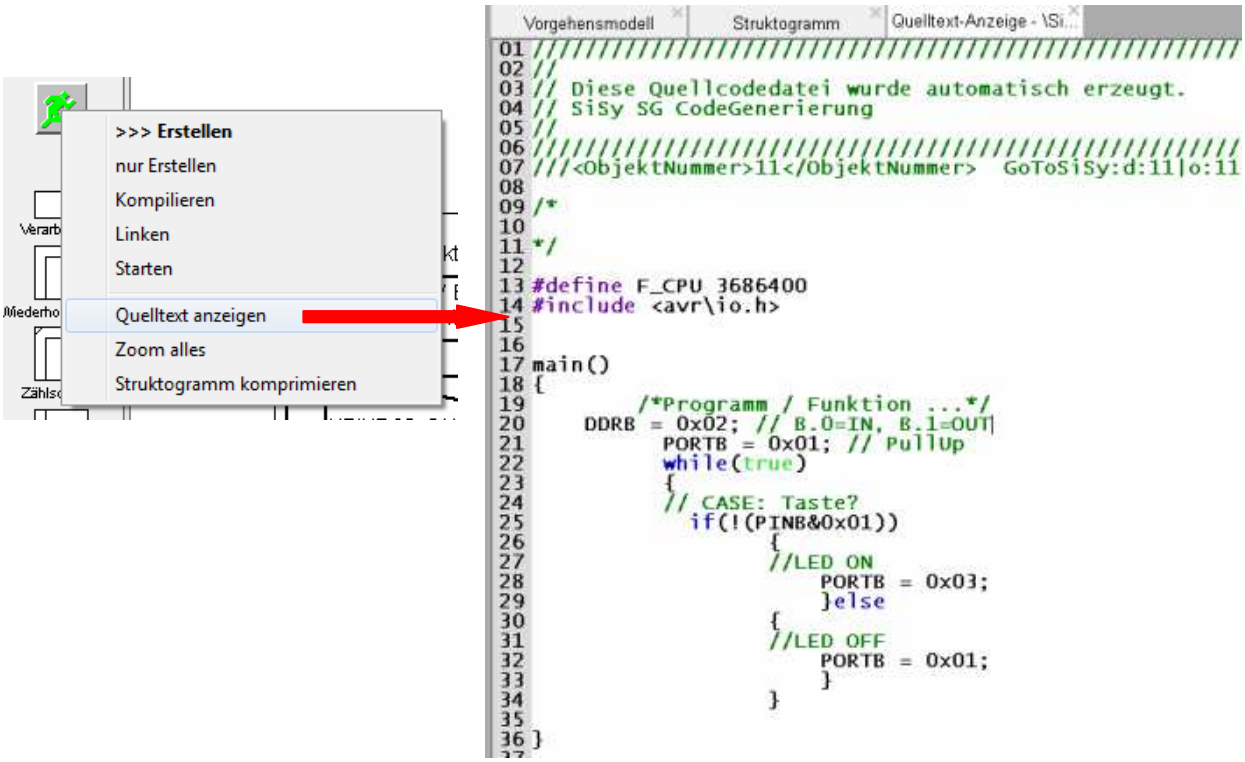
4. Als nächstes sind die Aktionen (`DO`-Elemente) in der Alternative zu ergänzen. Ziehen Sie dafür erneut jeweils eine „Verarbeitungsroutine“ in die Alternative des Struktogramms.



5. Tragen Sie den C-Quellcode zum Ein- und Ausschalten der LED in die `DO`-Elemente ein. Selektieren Sie dazu das betreffende `DO`-Element per Doppelklick. Mit ESC verlassen Sie den Editiermodus.

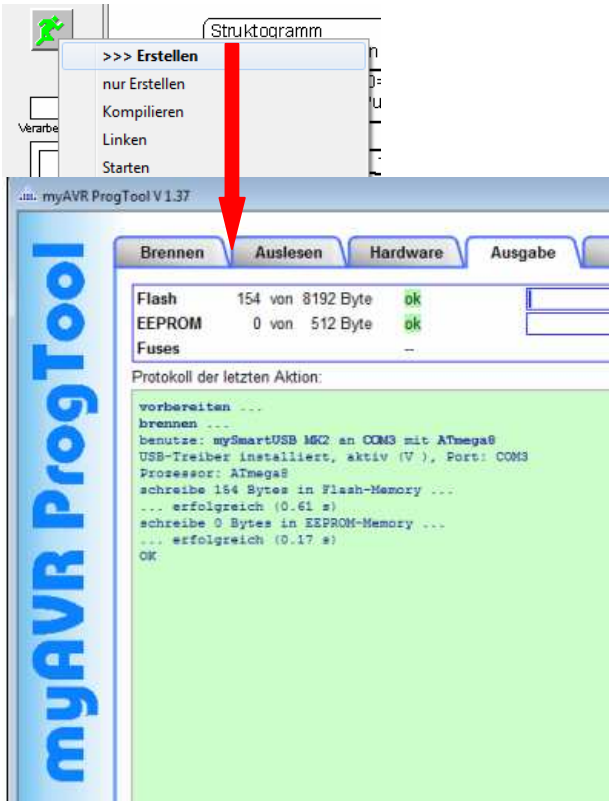


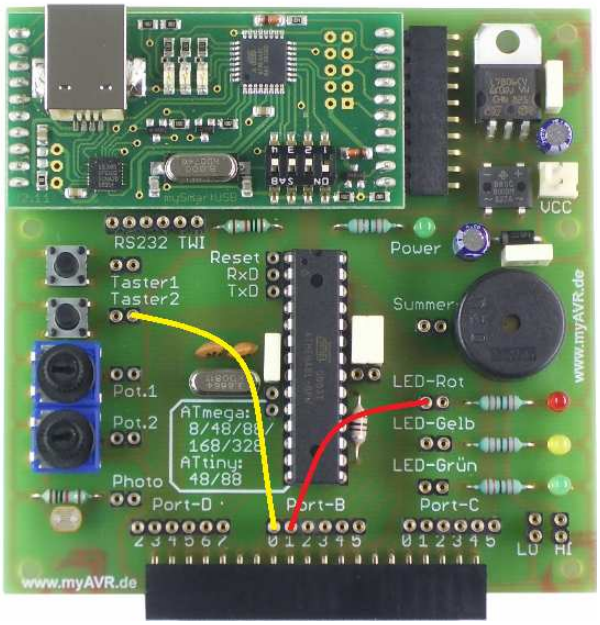
6. Generieren Sie den Quellcode für dieses Struktogramm und lassen Sie sich den Quellcode anzeigen.



7.2.4 Programmtest

Verbinden Sie das Board mit dem PC und aktivieren Sie aus dem Aktionsmenü den Befehl „>>> Erstellen“, damit brennen Sie das Programm auf den Controller. Es öffnet das myAVR ProgTool und zeigt das Protokoll an. Für den Test des Programms stecken Sie auf dem Board die Verbindungen.

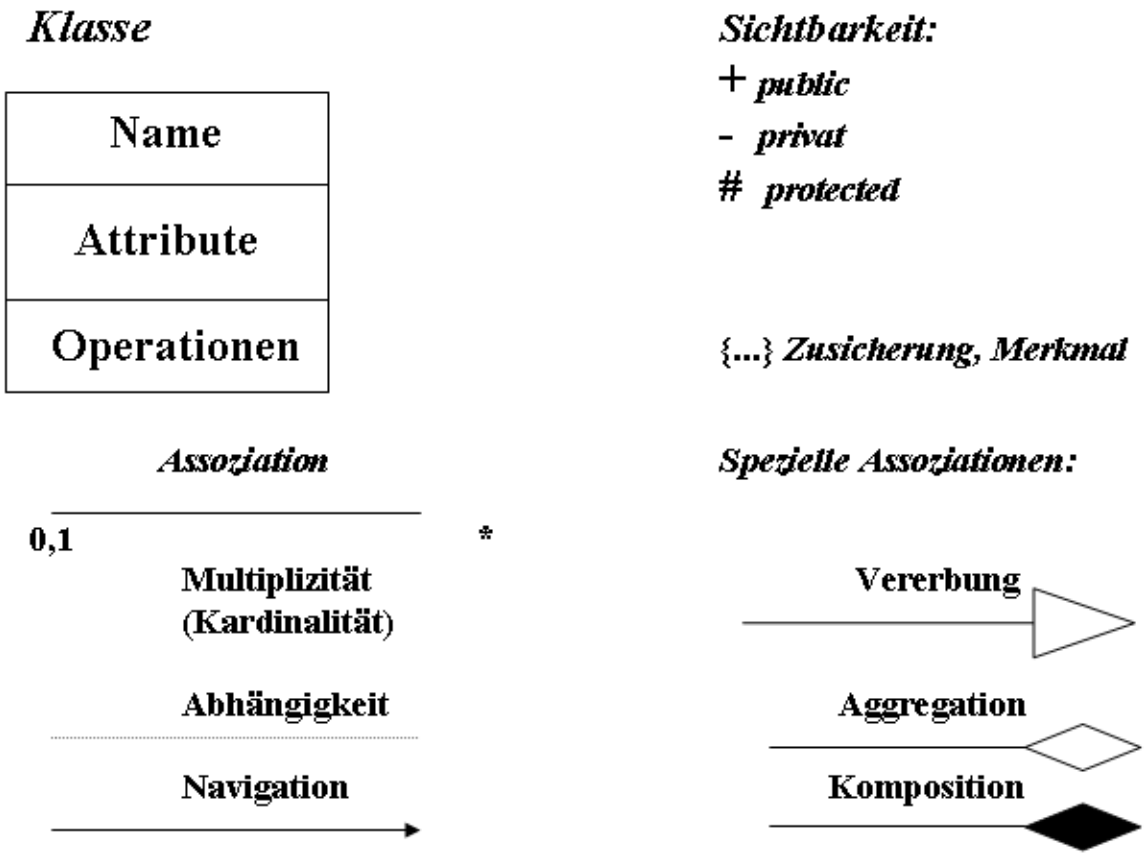




8 Entwicklung von Klassendiagrammen

8.1 Einleitung

Mit objektorientierten Programmiersprachen hat der Entwickler mächtige Sprachmittel, um komplexe Systeme realisieren zu können. C++ ist eine weit verbreitete objektorientierte Standardsprache. Als Visualisierungsmittel objektorientierter Programme gilt die international standardisierte Beschreibungssprache UML (Unified Modeling Language).
SiSy bietet dem Entwickler das UML Klassendiagramm mit Codegenerierung für C++, AVR C++ und ARM C++. Der folgende Abschnitt beschreibt die Handhabung des Klassendiagramms in SiSy. Die Abbildung zeigt Ihnen eine Kurzübersicht der Modellierungselemente des UML Klassendiagramms.



Kurzübersicht Elemente des UML Klassendiagramms

Schreibweise von Attributen:
Attribute beginnen mit einem Kleinbuchstaben.

```
Sichtbarkeit name : Typ = Initialwert {Merkmal}
# temperatur : uint8_t = 25
```

Schreibweise von Operationen:
Operationen beginnen mit einem Kleinbuchstaben.

```
Sichtbarkeit name (Parameter:Typ = Standardwert, ...) : Rückgabetyp {Merkmal}
+ setTemperatur ( temp : integer = 25 ) : bool
```

8.2 Vorgehensweise für PC- Programme mit SVL

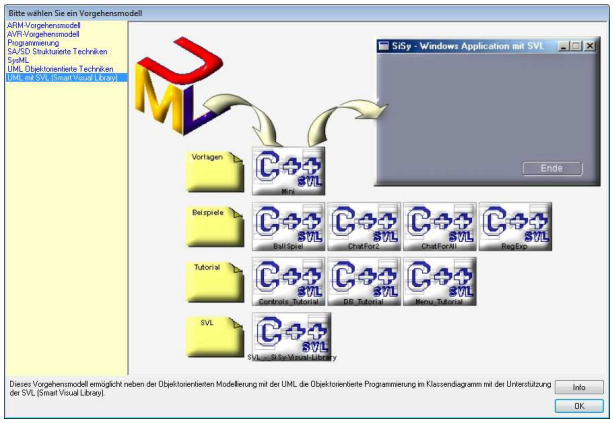
Hinweis:
Für dieses Beispiel benötigen Sie eine Ausgabe mit dem Add-On SVL

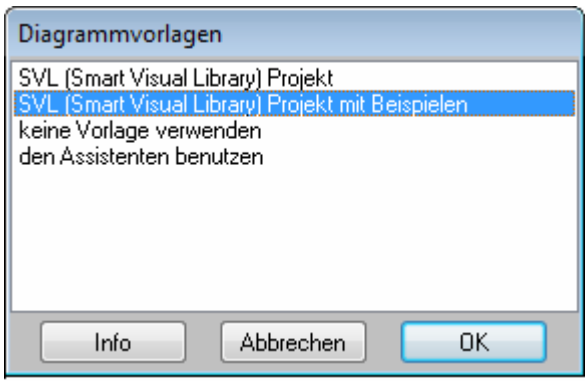
8.2.1 Zielstellung

Die Funktion des nachfolgenden Programms ist es, auf dem Bildschirm ein Fenster mit zwei Schaltflächen zu erstellen, welche das Fenster schließen bzw. eine Message Box erzeugen.

8.2.2 Vorbereitung

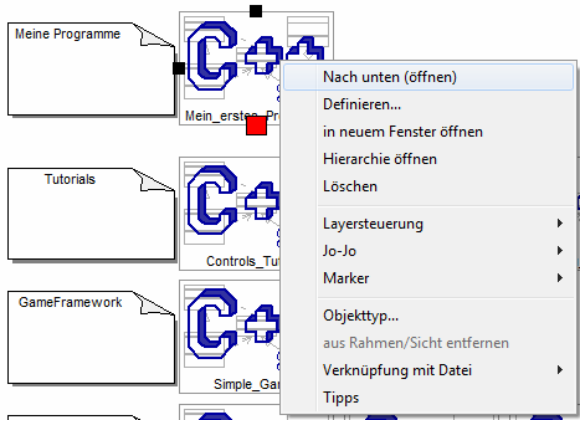
Starten Sie SiSy und wählen Sie „neues Projekt erstellen“. Vergeben Sie den Projektnamen „SVL_Button“, bestätigen Sie mit „Projekt anlegen“. Wählen Sie das Vorgehensmodell „UML mit SVL (Smart Visual Library) Projekt mit Beispielen“ und laden als Diagrammvorlage „Smart Visual Library (SVL)“, welches neben Beispielen und Tutorials auch ein Grundgerüst für SVL-Anwendungen beinhaltet.

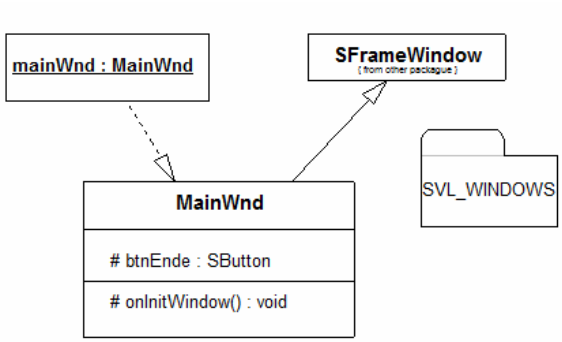




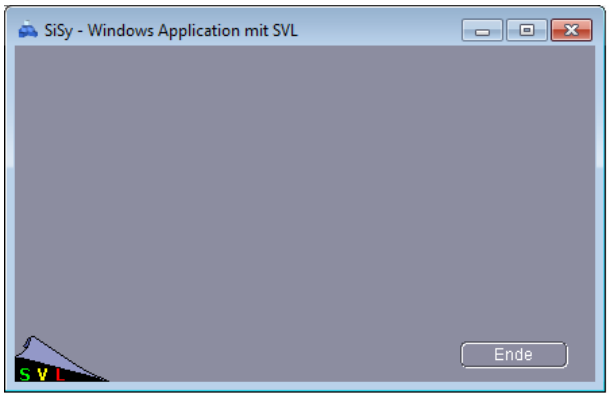
8.2.3 Grundgerüst für Fenster auswählen

Nach dem Import der Diagrammvorlage steht Ihnen eine Auswahl an Tutorials und Beispielen zur Verfügung. Wählen Sie neben der Notiz „Meine Programme“ das Klassendiagramm „Mein_erstes_Programm“ aus und öffnen Sie dieses (rechte Maustaste → „Nach unten (öffnen)“). Sie sehen das verfeinerte Klassendiagramm, welches das Grundgerüst für Ihr Programm ist.





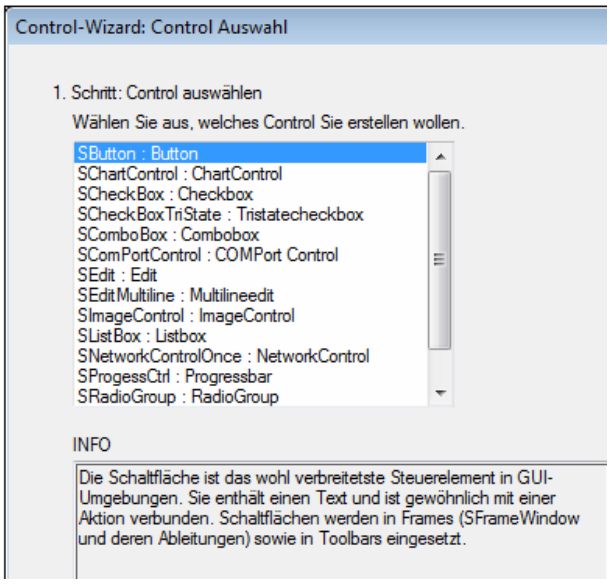
Mit der Aktion „Erstellen & Ausführen“, welche Sie über das Aktionsmenü erreichen, wird der dazugehörige Quellcode generiert, das Programm kompiliert und ausgeführt. Als Ergebnis erhalten Sie ein Fenster, welches bereits eine Schaltfläche für das Schließen des Fensters besitzt.



Diesem Fenster kann man nun beliebig viele Controls, d.h. Elemente (Schaltflächen, Checkbox, Textfeld etc.), hinzufügen und diesen einzelnen Controls entsprechende Funktionen zuweisen.
Dies erreicht man über eine direkte Eingabe des Quelltextes oder über den integrierten Control-Wizard, welcher das schnelle Einfügen und Konfigurieren der Controls ermöglicht und Bestandteil sowie Arbeitsmittel in der Abarbeitung dieses Beispiels ist.

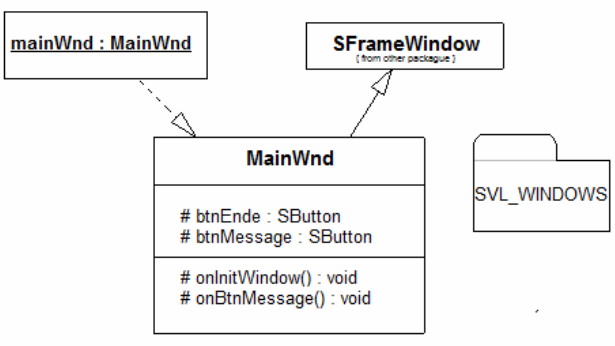
8.2.4 Schaltfläche mit Hilfe des Control-Wizards erstellen

Für den 2. Teil der Zielstellung fügen Sie dem Fenster eine weitere Schaltfläche hinzu. Dazu ziehen Sie aus der Objektbibliothek ein Objekt vom Typ „Attribut“ per Drag & Drop in die Klasse „MainWnd“. Daraufhin öffnet sich automatisch der Control-Wizard, welcher das Anlegen von Controls und Funktionen erleichtert, indem er Schritt für Schritt alle notwendigen Anweisungen ausführt.



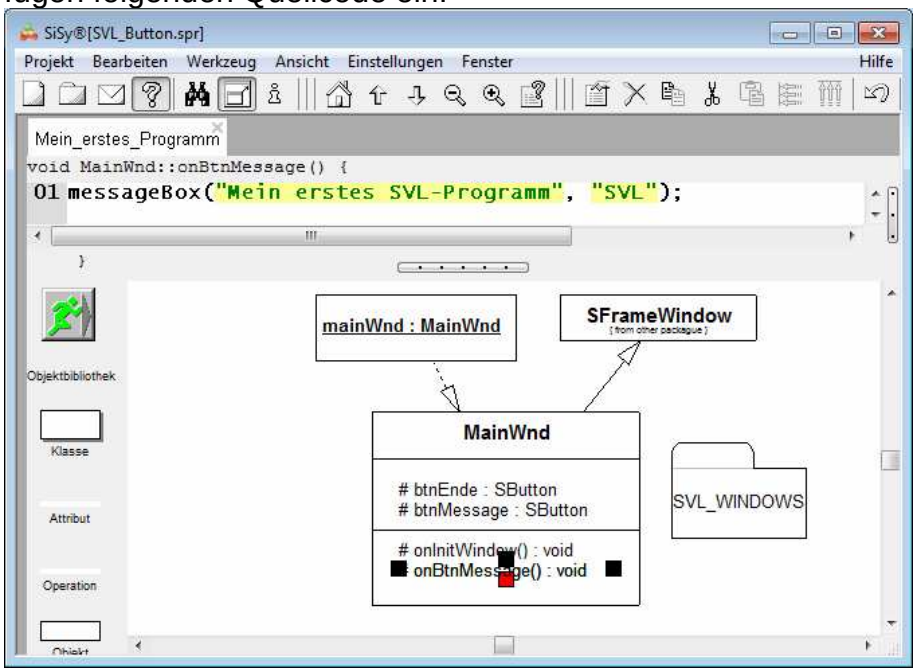
Wählen Sie im ersten Schritt
„SButton : SButton“
und klicken Sie auf „Weiter“ um eine Schaltfläche zu erzeugen.

Als nächstes geben Sie als Beschriftung des Elements den Namen „Message“ ein. Der Variablenname wird dabei automatisch vergeben, hier „`btnMessage`“. Klicken Sie nun „Weiter“ um zum nächsten Schritt zu gelangen, welcher sich mit der Positionierung der Schaltfläche beschäftigt. Diesen und den Folgeschritt (Tooltipp) überspringen Sie, indem Sie dreimal auf „Weiter“ und dann auf „Fertig stellen“ klicken. Die Schaltfläche und die Funktion `OnButtonMessage()` wird nun erstellt und der entsprechende Quellcode, welcher zur Generierung notwendig ist, wird automatisch erzeugt.



8.2.5 Quellcode hinzufügen

Damit die neu erzeugte Schaltfläche per Mausklick eine MessageBox aufruft, müssen Sie der Funktion `OnBtnMessage()` noch den entsprechenden Quellcode hinzufügen.
Dazu wählen Sie in der Klasse „MainWnd“ die Funktion `OnBtnMessage()` aus und fügen folgenden Quellcode ein:

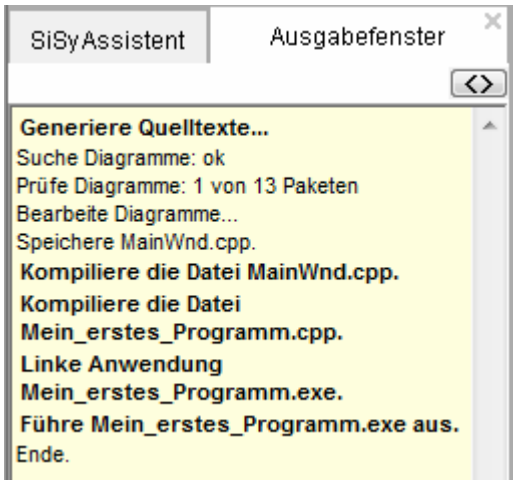


messageBox("Mein erstes SVL-Programm","SVL");

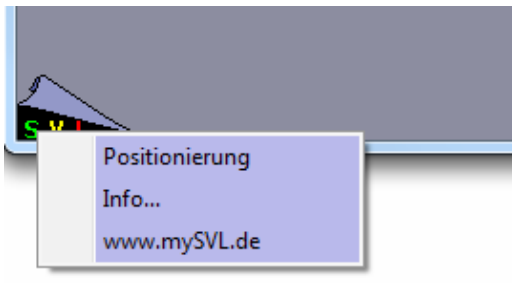
8.2.6 Kompilieren und Linken des fertigen Programms

Klicken Sie auf das Aktionsmenü und wählen Sie „>>> Erstellen & Ausführen“ um das Projekt zu kompilieren und zu linken. Das Programm wird nach erfolgreichem Abschließen des Vorgangs automatisch gestartet.

Sollten Sie keine Fehler im Ausgabefenster erhalten haben, dann können Sie nun durch einen Klick auf die Schaltfläche „Message“ eine MessageBox mit der Nachricht „Mein erstes SVL-Programm“ erzeugen.



Für eine andere Positionierung der Schaltfläche klicken Sie in der linken unteren Ecke auf das SVL-Logo und wählen „Positionierung“. Nun können Sie per Drag & Drop Verfahren die einzelnen Elemente des Fensters neu positionieren.



8.3 Vorgehensweise für AVR Programme

Hinweis:
Dieses Beispiel kann mit der Ausgabe SiSy Microcontroller ++ und SiSy AVR erstellt werden.

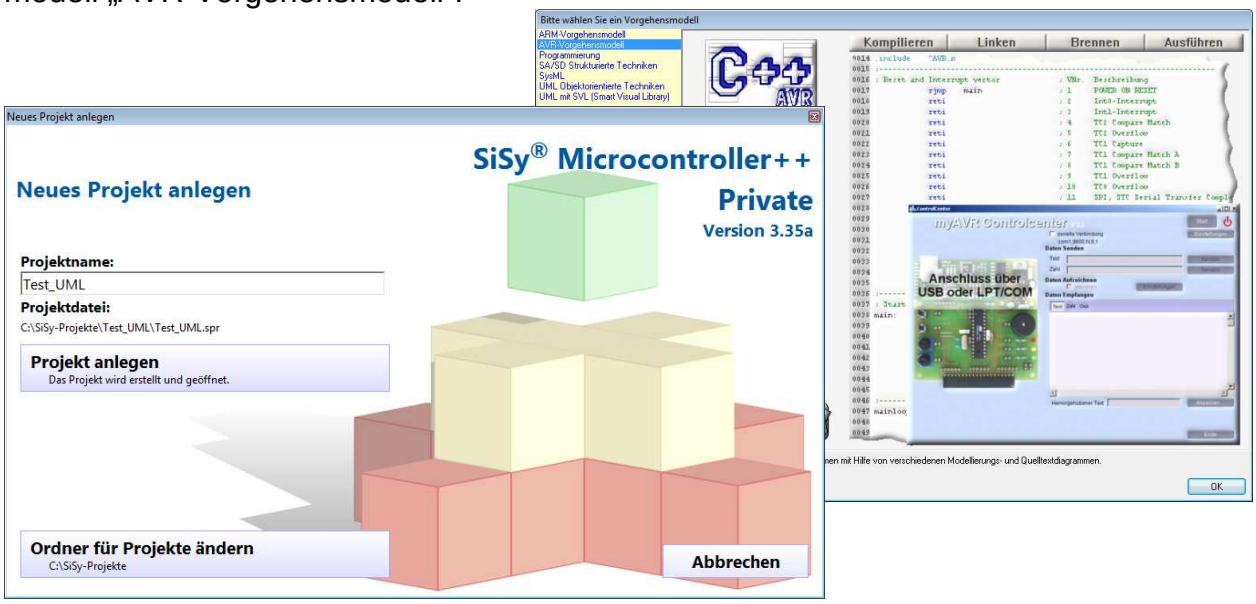
8.3.1 Zielstellung

Es soll eine Mikrocontrollerlösung entwickelt werden, bei der auf Tastendruck eine LED eingeschaltet wird. Die Realisierung dieser Aufgabe soll mit dem Klassendiagramm in SiSy erfolgen.

Schaltung:
Port B.0 = Taster
Port B.1 = LED

8.3.2 Vorbereitung

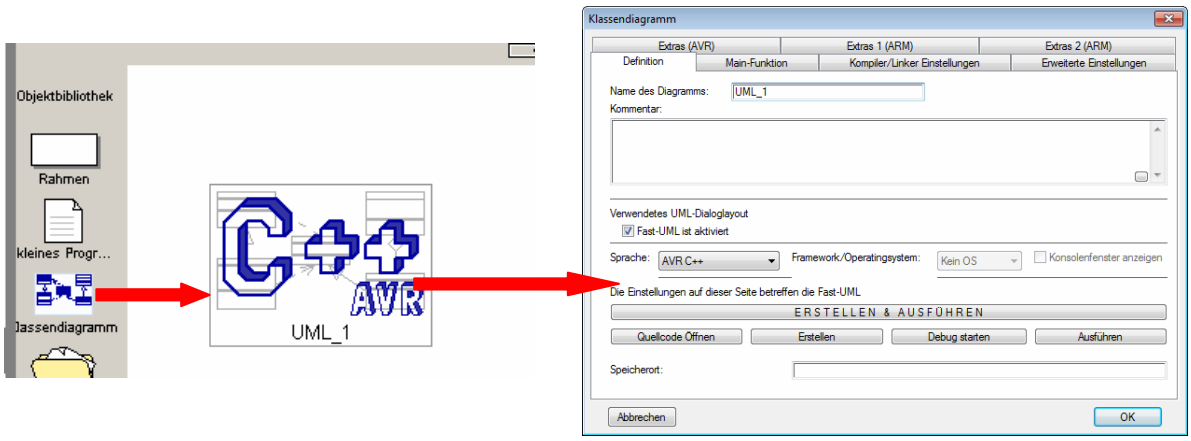
Starten Sie SiSy und wählen Sie „neues Projekt erstellen“. Vergeben Sie den Projektnamen „Test_UML“, bestätigen Sie mit „Projekt anlegen“. Wählen Sie das Vorgehensmodell „AVR-Vorgehensmodell“.



Legen Sie im weiteren Verlauf die AVR-Grundeinstellungen für das Projekt fest. Danach erscheint ein Dialogfenster mit möglichen Diagrammvorlagen, verwenden Sie keine Vorlage.

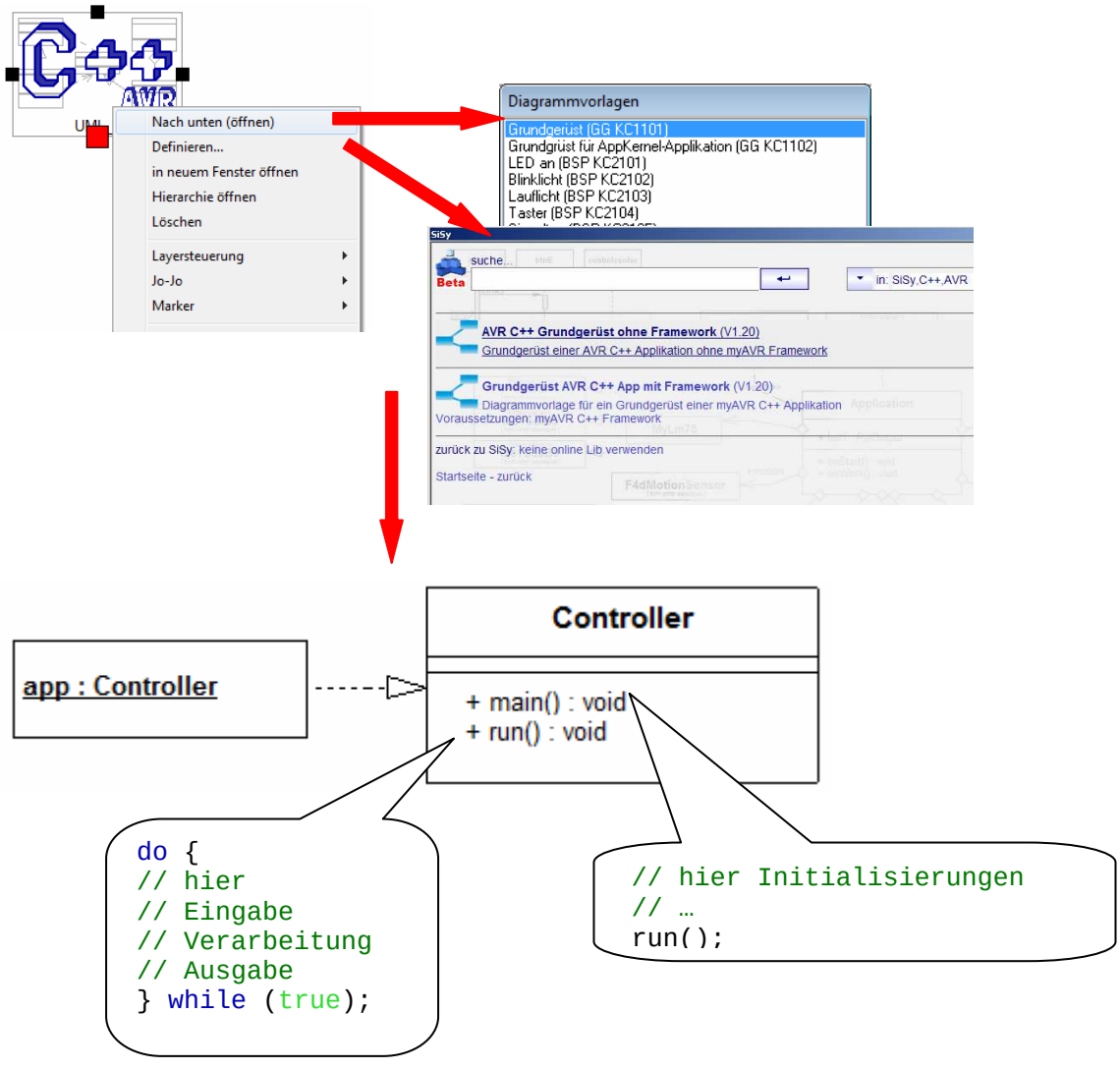
Hinweis:
Wenn Ihre online-Verbindung zum Internet aktiv ist, erhalten Sie an Stelle des Dialogfensters für die Auswahl der Vorlagen das Dialogfenster von LibStore.
Hier können Sie die Auswahl treffen für „myAVR C++ Framework“ und den Download ausführen. Nach dem Import der Dateien steht Ihnen im Diagrammfenster das Objekt „myAVR_Library“ mit zahlreichen Beispielen zur Verfügung.

Ziehen Sie ein Objekt vom Typ „Klassendiagramm“ per Drag & Drop in das Diagrammfenster. Legen Sie in dem sich öffnenden Dialogfenster einen Namen für das Klassendiagramm fest und die Sprache „AVR C++“.
Prüfen Sie unter „Extras (AVR)“ die Hardware-Einstellungen.



8.3.3 Grundstruktur laden

Zur Entwicklung des Klassenmodells öffnen Sie das Klassendiagramm (auf dem Objekt rechte Maustaste -> Kontextmenü -> Nach unten). Laden Sie die Diagrammvorlage „Grundgerüst (GGKC1101)“ bzw. aus LibStore „AVR C++ Grundgerüst ohne Framework“.



Hinweis:
Die Struktur einer objektorientierten Mikrocontrollerlösung in SiSy erfordert im Klassenmodell eine Applikationsklasse (Hauptklasse), die sich dadurch auszeichnet, dass diese über eine Methode (Operation) mit dem Namen `main` verfügt. Der Codegenerator erzeugt eine Instanz diese Klasse und ruft die Main-Methode auf.

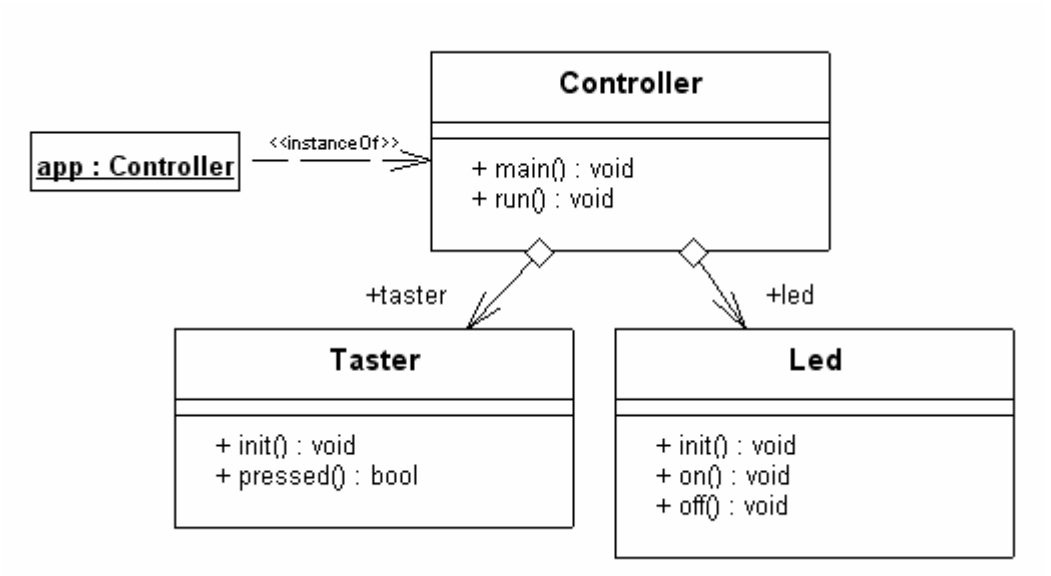
Beispiel:
```
#define cpp_Test_UML
#define F_CPU 3686400
#include <avr\io.h>

main (void)
{
    Controller MyApp;
    MyApp.main();
}
```

8.3.4 Systemstruktur entwerfen

Die Systemstruktur einer objektorientierten Anwendung bildet die Objekte und deren Beziehungen im Programm ab, welche im realen System als physische Elemente vorhanden sind. Als Bauplan der Objekte dienen Klassendeklarationen, welche die Eigenschaften (Attribute) und das Verhalten (Methoden/Operationen) der Objekte beschreiben. Das Klassendiagramm beschreibt also die Struktur der Klassen (Baupläne der Objekte) und die Beziehungen zwischen den Klassen. In unserer Aufgabenstellung finden wir die Objekte des Systems als **Substantive**, deren Beziehungen und Verhalten als **Verbalphrasen**.

*Es soll eine Mikro**controller**lösung entwickelt werden, bei der durch **drücken** eines **Tasters** eine **LED** **eingeschaltet** wird. Dabei soll der Taster an Port B 0 und die LED an Port B 1 **angeschlossen** werden*

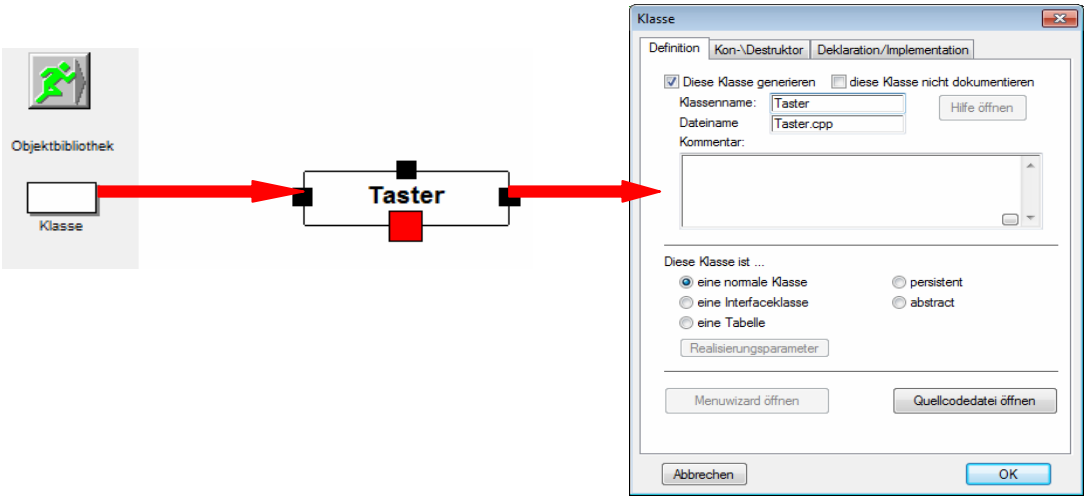
Daraus lässt sich folgende Klassenstruktur ableiten:



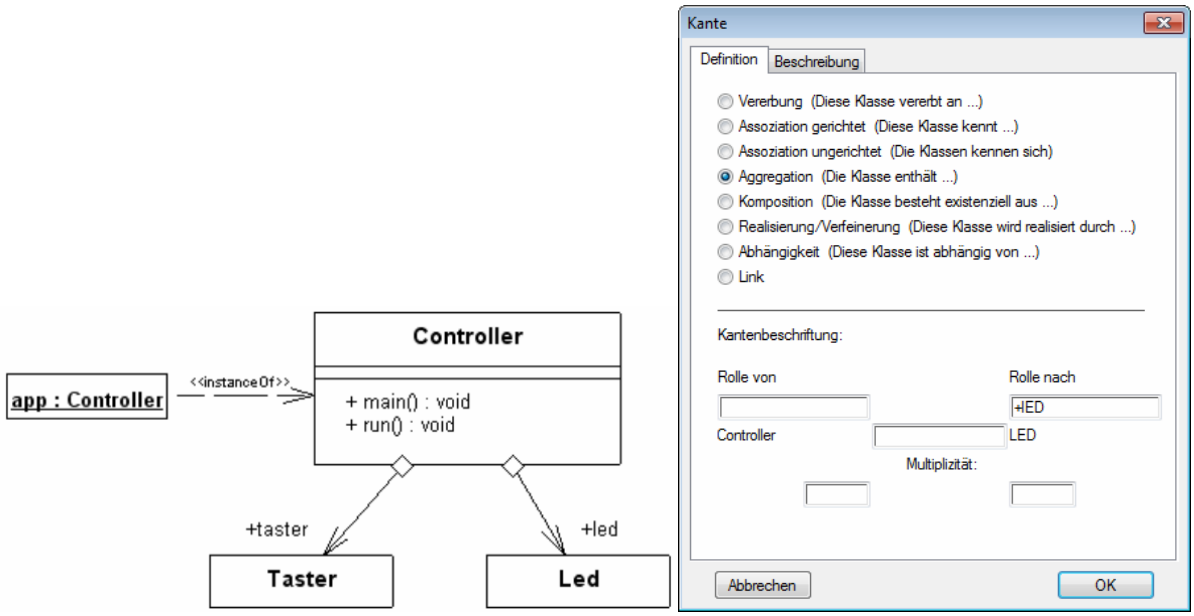
Klassenstruktur, Systementwurf mit dem UML Klassendiagramm

Zum Erstellen dieses Klassenmodells sind folgende Arbeitsschritte nötig:

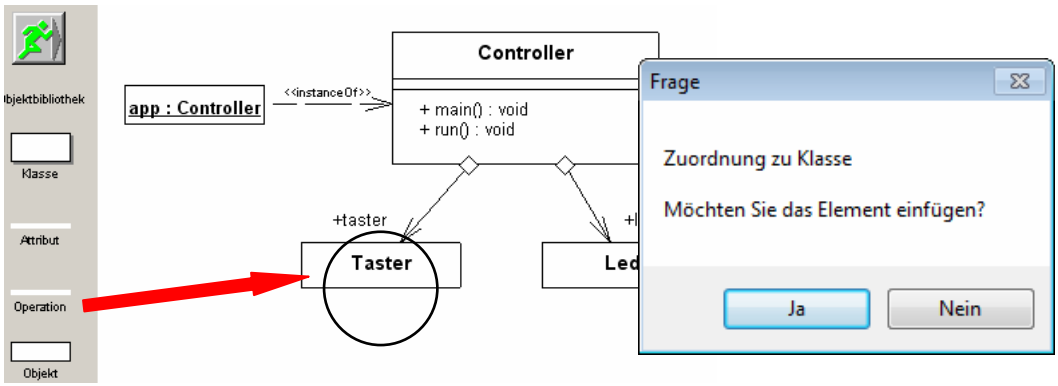
1. **Klassen einfügen und definieren**, ziehen Sie dazu das Objekt vom Typ „Klasse“ per Drag & Drop aus der Objektbibliothek in das Diagramm. Definieren Sie den Namen der Klasse und setzen die Option „diese Klasse generieren“.



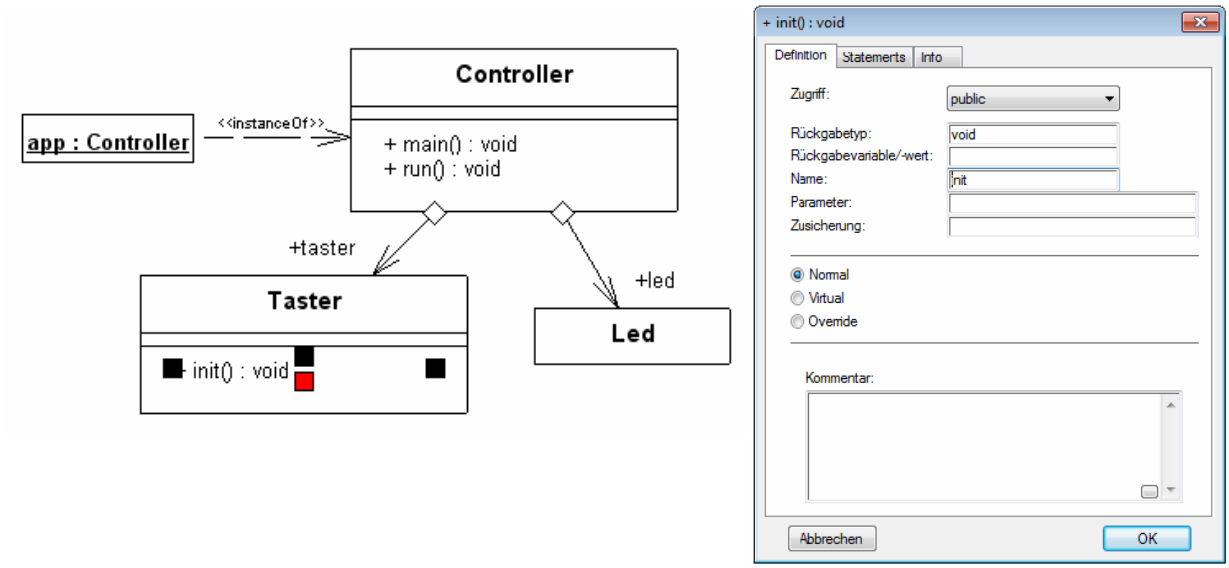
2. **Klassen verbinden**, selektieren Sie die Klasse, von der aus eine Verbindung gezogen werden soll. Ziehen Sie per Drag & Drop ausgehend vom roten Verteiler eine Verbindung auf die gewünschte Klasse. Wählen Sie den Verbindungstyp, zum Beispiel „Aggregation“ und beschriften Sie die Verbindung.



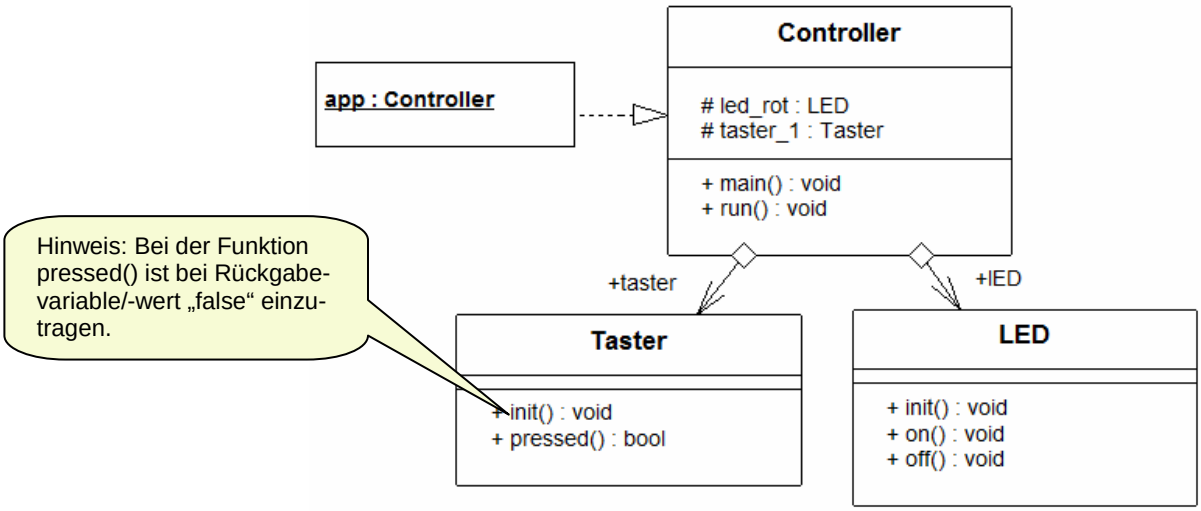
3. **Operationen und Attribute einfügen und definieren**, ziehen Sie dazu ein Objekt vom Typ „Operation“ oder „Attribut“ aus der Objektbibliothek auf die Klasse, in die das Attribut oder die Operation eingefügt werden soll. Bestätigen Sie die Sicherheitsabfrage zum Einfügen des Elementes.



Definieren Sie Zugriff (Sichtbarkeit), Name, Typ und Parameter der Operation bzw. Zugriff (Sichtbarkeit), Name, Typ und Initialwert des Attributes.



Vervollständigen Sie das Klassenmodell entsprechend der Abbildung.

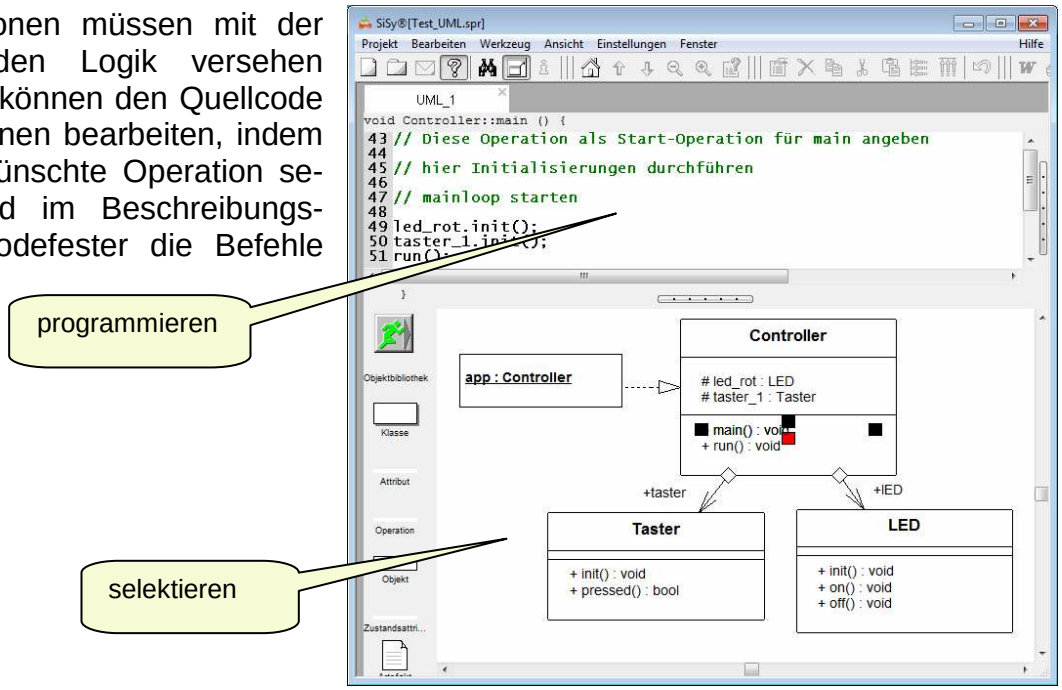


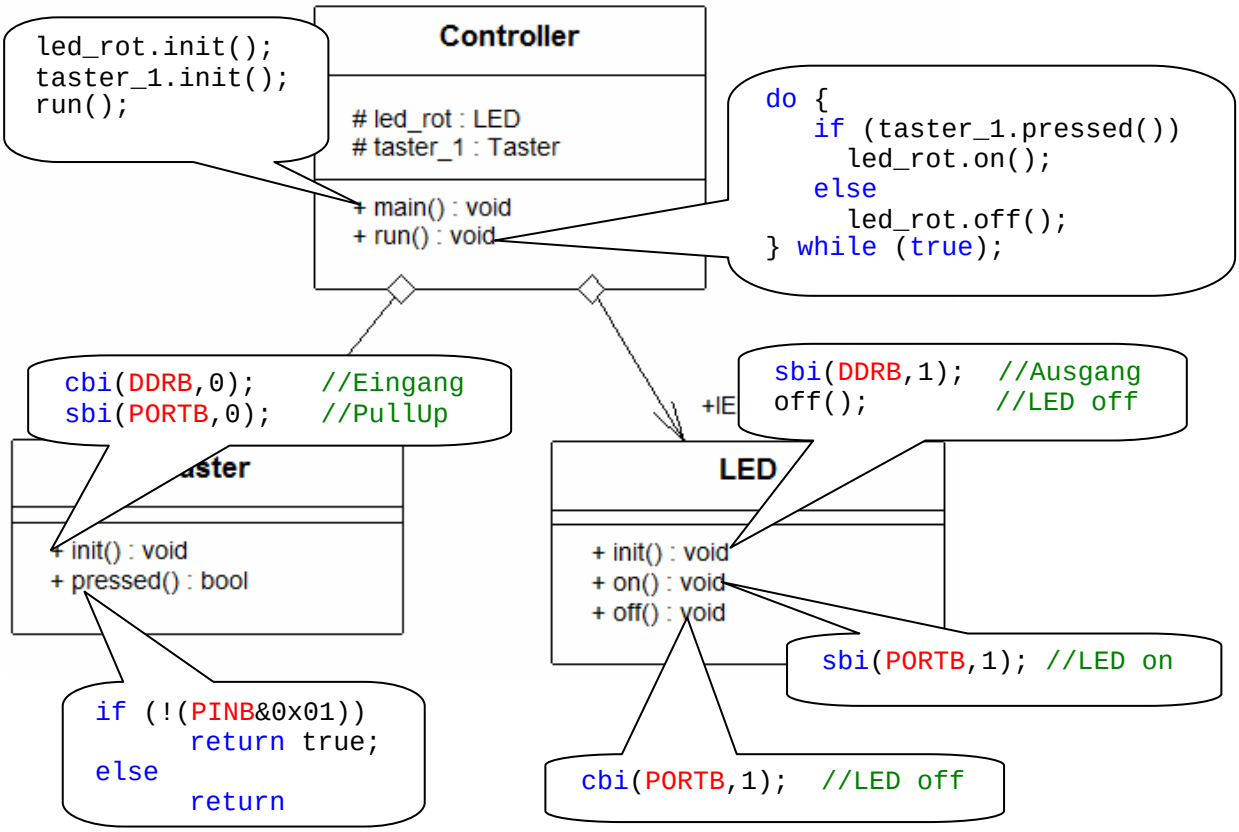
endgültiger Systementwurf mit dem UML Klassendiagramm

Hinweis:
SiSy kennt generischen Strukturen, Templates. *Es sind Vorlagen, die zum Übersetzungszeitpunkt etwas Konkretes erstellen. Die Generierung der konkreten Klassen übernimmt dabei der Codegenerator.*
Die Templates finden sich im Paket ***Tempos*** *und deren Unterordner. Tempos kann mit konventionellen Klassen durchaus gemischt eingesetzt werden. Die gewünschten Tempos-Pakete finden Sie über den Navigator (rechte Maustaste/UML-Pakete).*

8.3.5 Systemverhalten programmieren

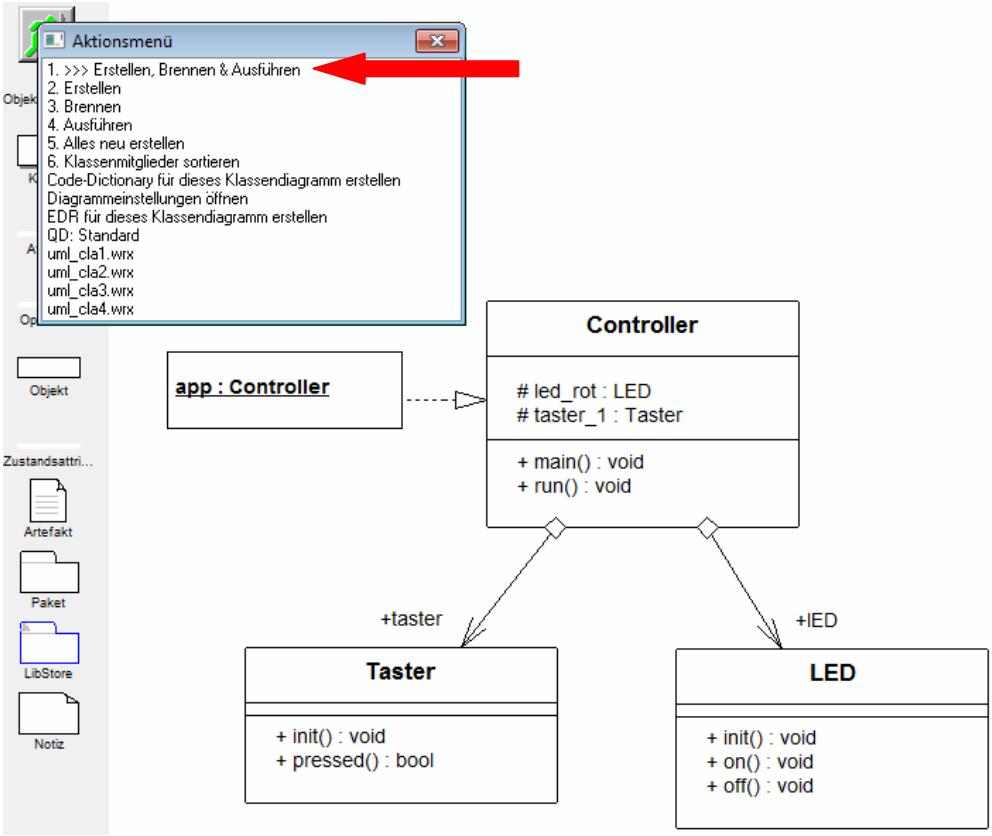
Die Operationen müssen mit der entsprechenden Logik versehen werden. Sie können den Quellcode der Operationen bearbeiten, indem Sie die gewünschte Operation selektieren und im Beschreibungs- bzw. Quellcodefester die Befehle eingeben.



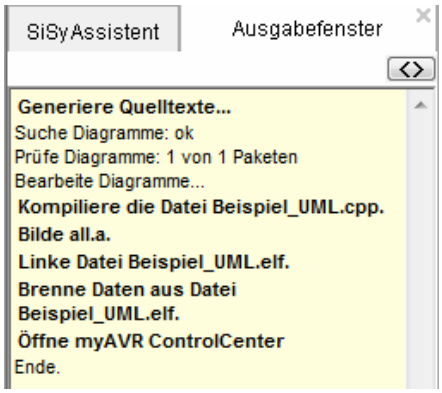


8.3.6 Übersetzen, Brennen und Testen

Die Codegenerierung aus dem UML Klassendiagramm, das Kompilieren, Linken und Brennen kann über das Aktionsmenü gestartet werden.



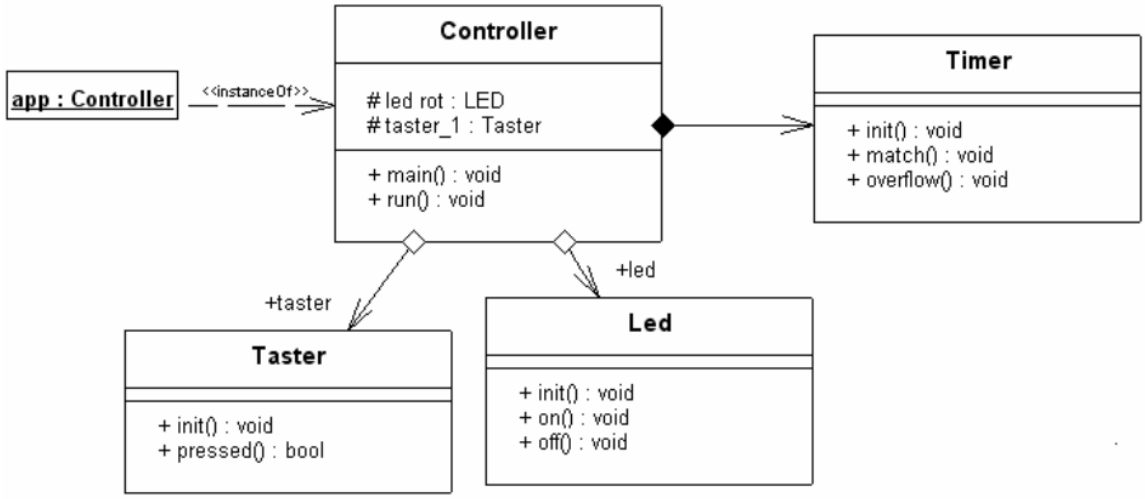
Sie erhalten im Ausgabefenster ein Protokoll der ausgeführten Aktionen.



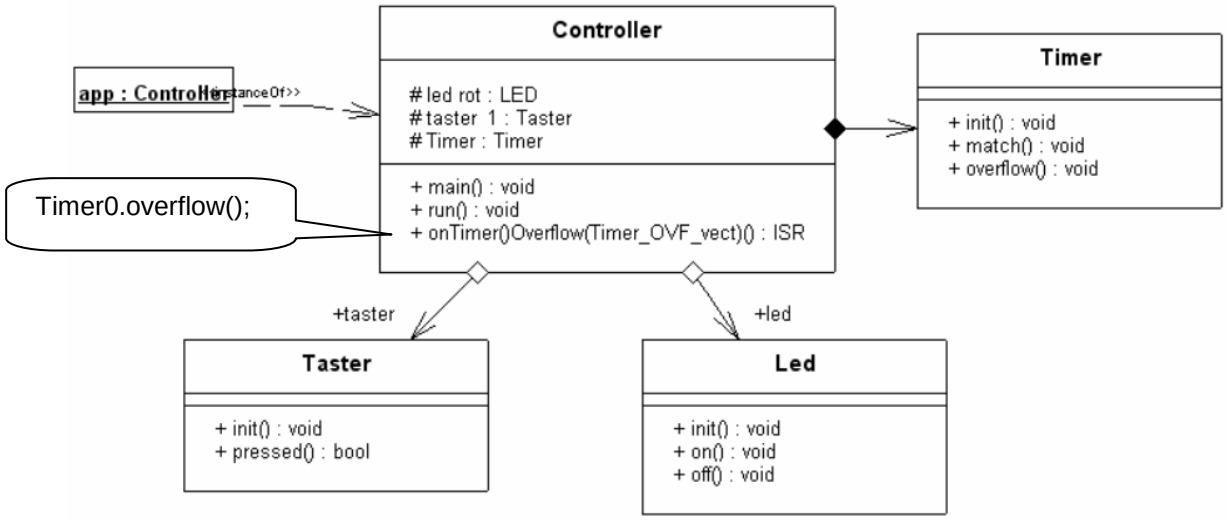
8.3.7 Interrupt-Service-Routinen (ISR) im Klassendiagramm

Interrupt-Service-Routinen (im weiteren ISR) werden in AVR C++ durch das Schlüsselwort ISR gekennzeichnet. Sie bilden eigenständige Funktionen. Die Besonderheit bei der Realisierung einer ISR liegt darin, dass es sich hier um ein C-Makro handelt und nicht um eine echte Funktion. Diese können also keine Methode einer Klasse sein. Um eine ISR im Klassendiagramm zu realisieren gehen Sie wie folgt vor (Erweiterung des Beispiels um die Klasse `Timer` und der `ISR Timer0 Overflow`):

1. Die Klasse für die interruptfähige Komponente modellieren und die Methode einfügen, welche beim Interrupt ausgeführt werden soll. In unserem Beispiel ist es die Klasse `Timer` mit der Methode `overflow`. Die Initialisierung des Timers muss entsprechend erfolgen.



2. Ergänzen Sie das Klassendiagramm weiter entsprechend der Abbildung.



Beispiel für Ergänzung der main:

```
#include    <io.h>

// Instanz der interruptfähigen Klasse anlegen
Timer Timer0;

// Interuptmakro mit Aufruf der betreffenden Methode
ISR (TIMER0_OVF_vect)
{
      Timer0.overflow();
}
```

In der Interrupt-Service-Routine rufen Sie nur die Methode der betreffenden Klasse auf.

8.4 Vorgehensweise für ARM Programme

Hinweis:
Dieses Beispiel kann mit der Ausgabe SiSy Microcontroller ++ und SiSy ARM erstellt werden.

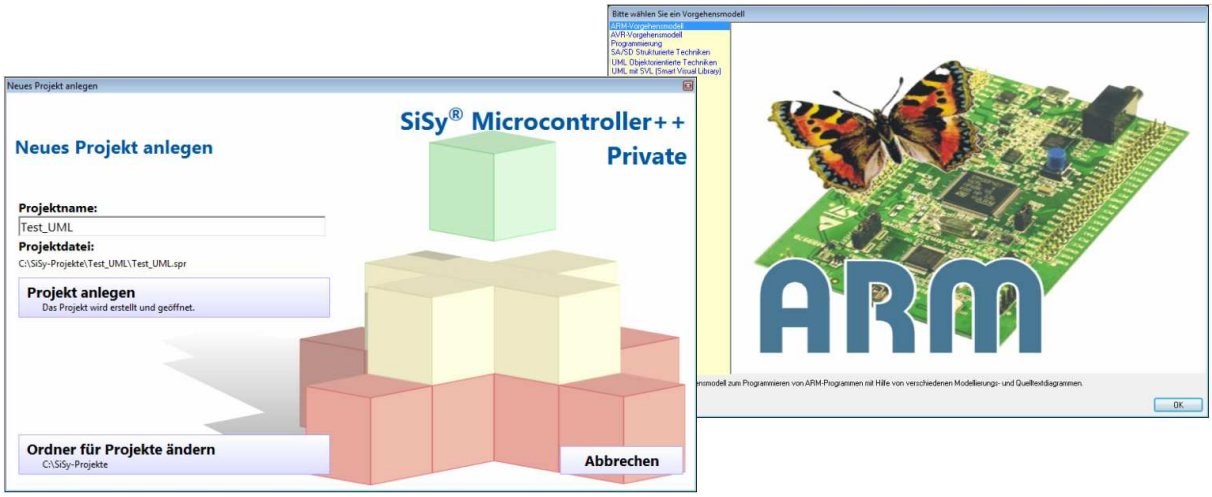
8.4.1 Zielstellung

Es soll eine Mikrocontrollerlösung mit einem Klassendiagramm entwickelt werden, bei der auf Tastendruck eine LED eingeschaltet wird. Die Referenzhardware ist ein STM32F4-Discovery.

Schaltung:
Port GPIOA.0 = Taster
Port GPIOD.13 = LED.

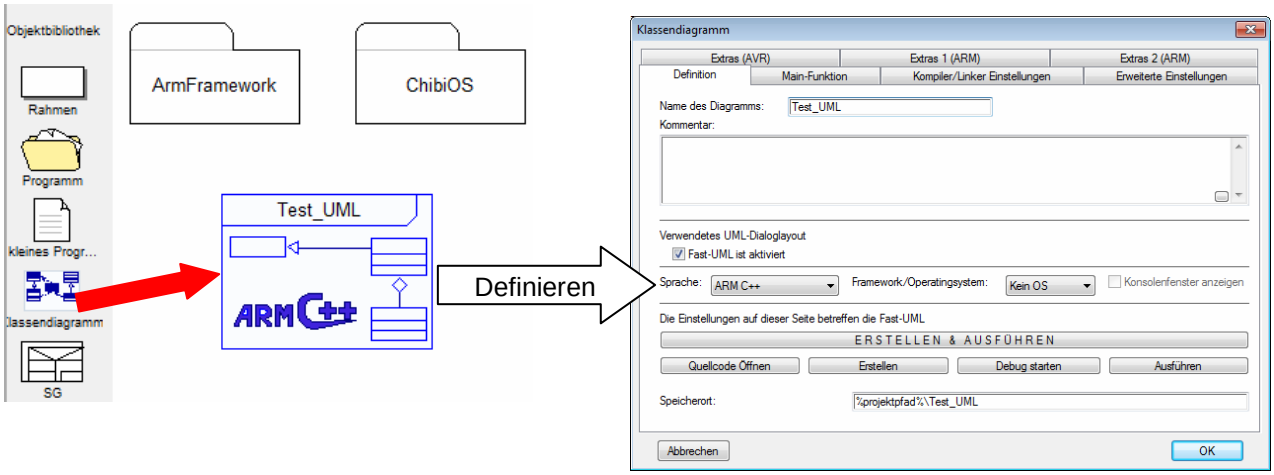
8.4.2 Vorbereitung

Starten Sie SiSy und wählen Sie „neues Projekt erstellen“. Vergeben Sie den Projektnamen „Test_UML“, bestätigen Sie mit „Projekt anlegen“. Wählen Sie das Vorgehensmodell „ARM-Vorgehensmodell“.



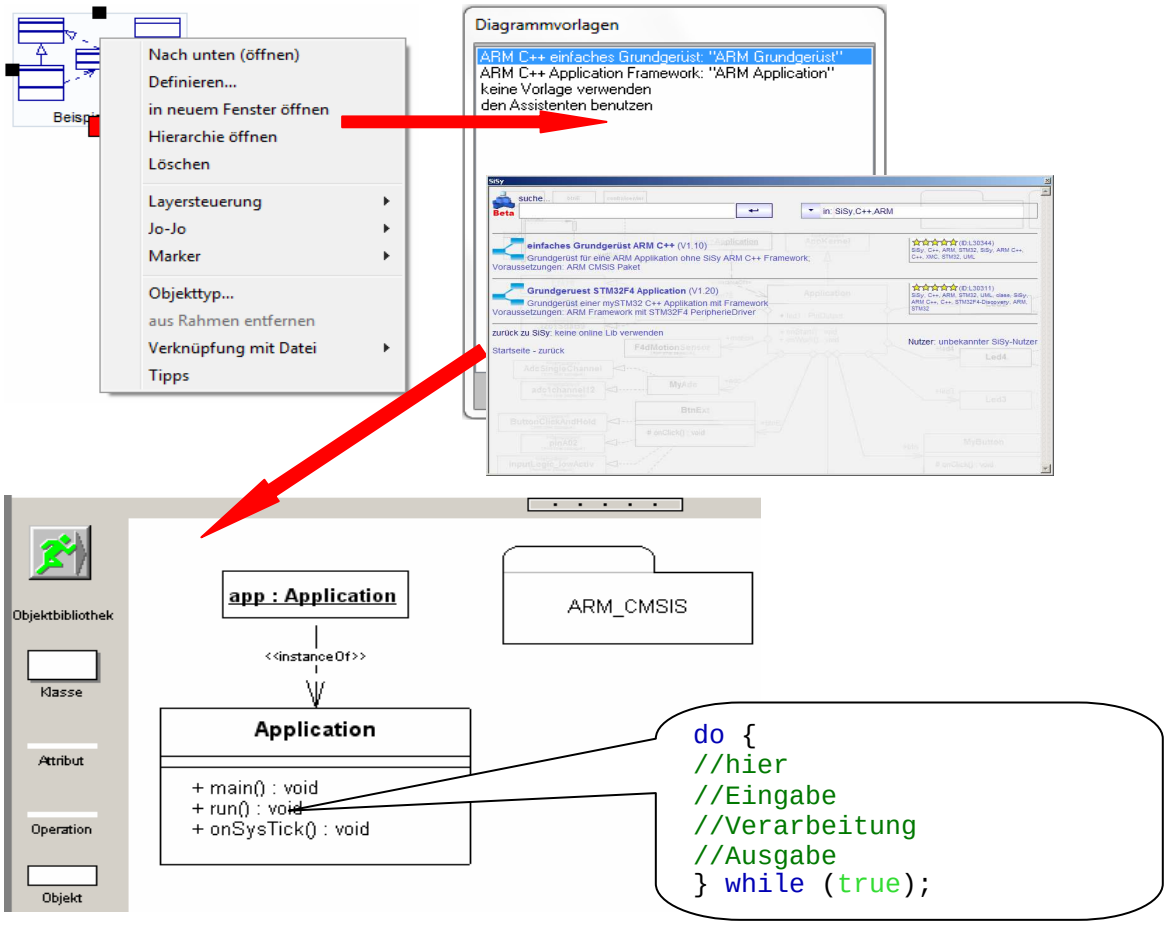
Von den möglichen Diagrammvorlagen wählen Sie „ARM C++ Framework“ aus. Wenn Ihre online-Verbindung zum Internet aktiv ist, erhalten Sie an Stelle des Dialogfensters für die Auswahl der Vorlagen das Dialogfenster von LibStore.
Hier können Sie die Auswahl treffen für „ARM Framework“ oder „ARM Framework mit Beispielen“.

Ziehen Sie ein Objekt vom Typ „Klassendiagramm“ per Drag & Drop in das Diagrammfenster. Vergeben Sie für das Klassendiagramm einen Namen und wählen Sie die Sprache „ARM C++“ aus. Auf der Registerkarte „Options (ARM)“ kontrollieren Sie die voreingestellte Hardware bzw. wählen Ihre Hardware aus und laden die Vorlage.



8.4.3 Grundstruktur laden

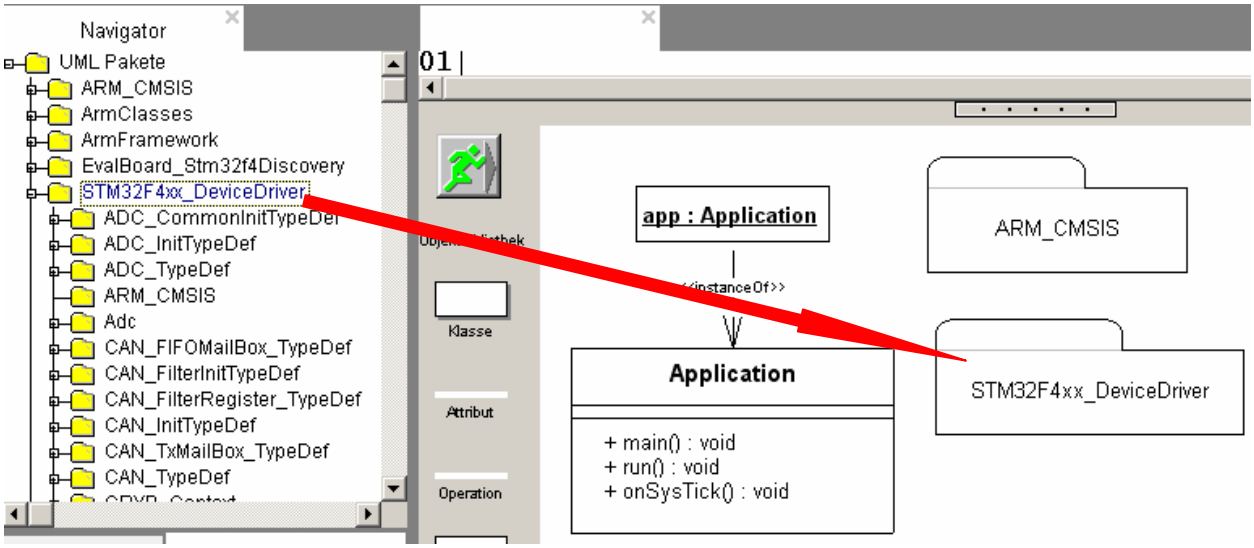
Zur Entwicklung des Klassenmodells muss das Klassendiagramm geöffnet werden. Wählen Sie dazu auf dem Objekt rechte Maustaste -> Kontextmenü -> Nach unten. Es öffnet das Dialogfenster mit Vorlagen oder LibStore.



Hinweis:
Die Struktur einer objektorientierten Mikrocontrollerlösung in SiSy erfordert im Klassenmodell eine Applikationsklasse (Hauptklasse), die sich dadurch auszeichnet, dass diese über eine Methode (Operation) mit dem Namen ***main*** *verfügt. Der Codegenerator erzeugt eine Instanz dieser Klasse und ruft die Main-Methode auf.*

```
Systick::config(SystemCoreClock/100);
// Led initialisieren
Rcc rcc;
rcc.ahb1PeriphClockCmd(RCC_AHB1Periph_GPIOD,ENABLE);
led.initStruct.GPIO_Pin |= 1<<12;
led.initStruct.GPIO_Mode = GPIO_Mode_OUT;
led.initStruct.GPIO_Speed = GPIO_Speed_50MHz;
led.initStruct.GPIO_OType = GPIO_OType_PP;
led.initStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
led.init(GPIOD);
// Mainloop
this->run();
```

Da für dieses Beispiel die Hardware STM32F4 verwendet wird, sollte das Treiberpaket mit in das Klassendiagramm gezogen werden. Klicken Sie mit der rechten Maustaste auf den „Navigator“ und wählen Sie die „UML-Pakete“ aus. Danach ziehen Sie das Paket in das Diagramm.

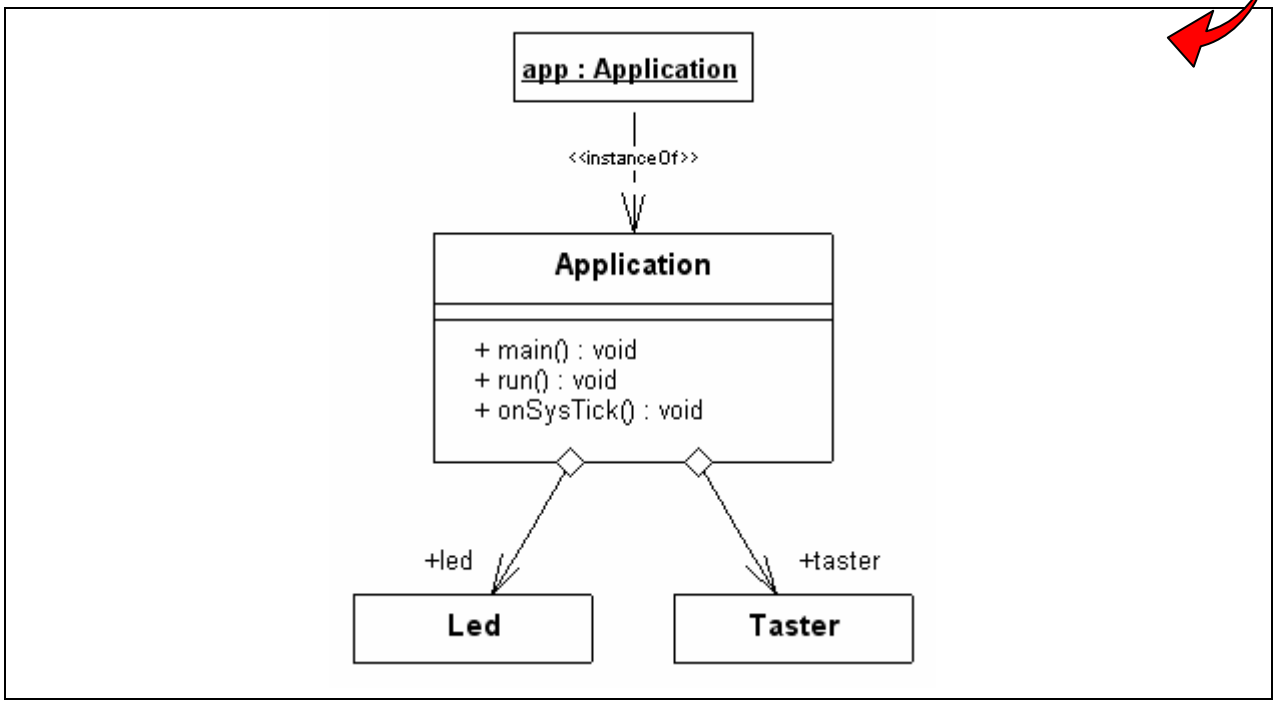


8.4.4 Systemstruktur entwerfen

Die Systemstruktur einer objektorientierten Anwendung bildet die Objekte und deren Beziehungen im Programm ab, welche im realen System als physische Elemente vorhanden sind. Als Bauplan der Objekte dienen Klassendeklarationen, welche die Eigenschaften (Attribute) und das Verhalten (Methoden/Operationen) der Objekte beschreiben. Das Klassendiagramm beschreibt also die Struktur der Klassen (Baupläne der Objekte) und die Beziehungen zwischen den Klassen. In unserer Aufgabenstellung finden wir die Objekte des Systems als **Substantive,** deren Beziehungen und Verhalten als **Verbalphrasen**.

*Es soll eine Mikro**controller**lösung entwickelt werden, bei der durch **drücken** eines **Tasters** eine **LED** **eingeschaltet** wird. Dabei soll der Taster an Port GPIOA.0 und die LED an Port GPIOD.13 **angeschlossen** werden*

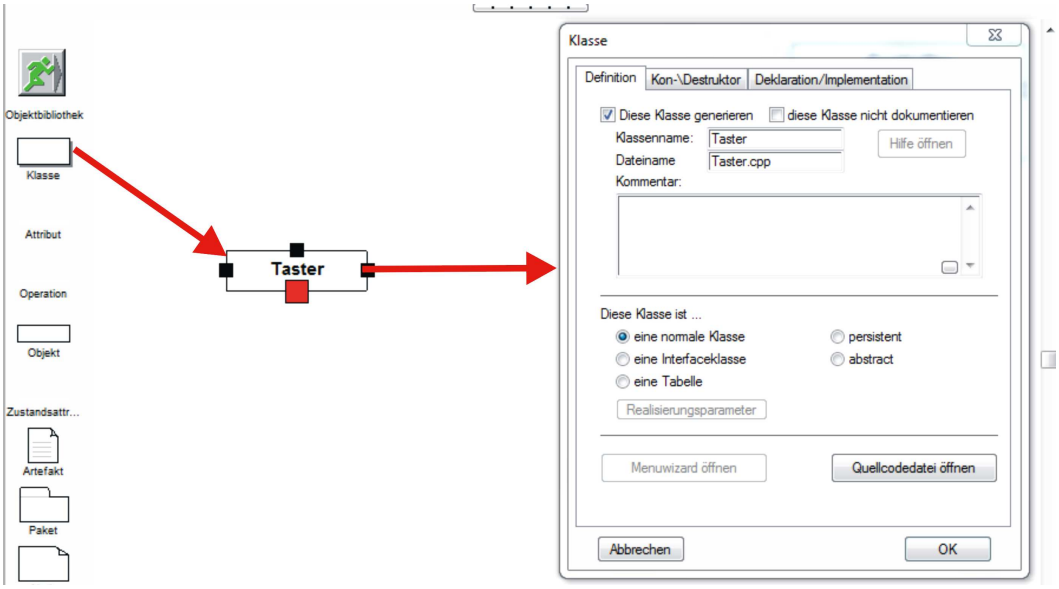
Daraus lässt sich folgende Klassenstruktur ableiten:



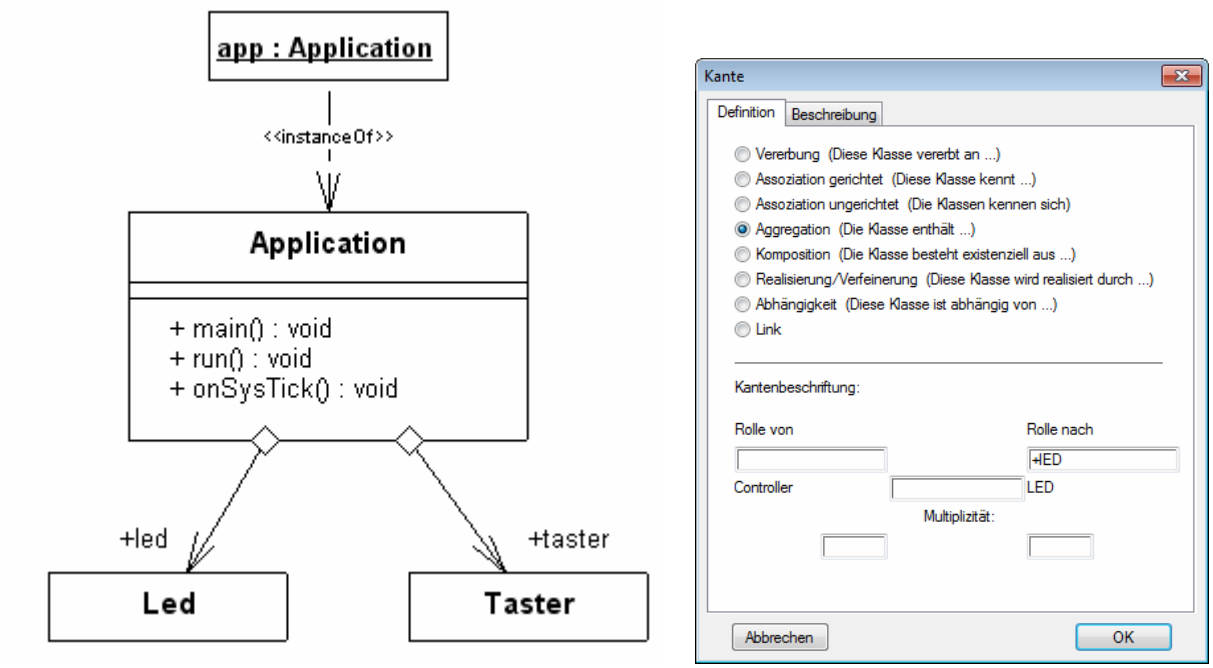
Klassenstruktur, Systementwurf mit dem UML Klassendiagramm

Zum Erstellen dieses Klassenmodells sind folgende Arbeitsschritte nötig:

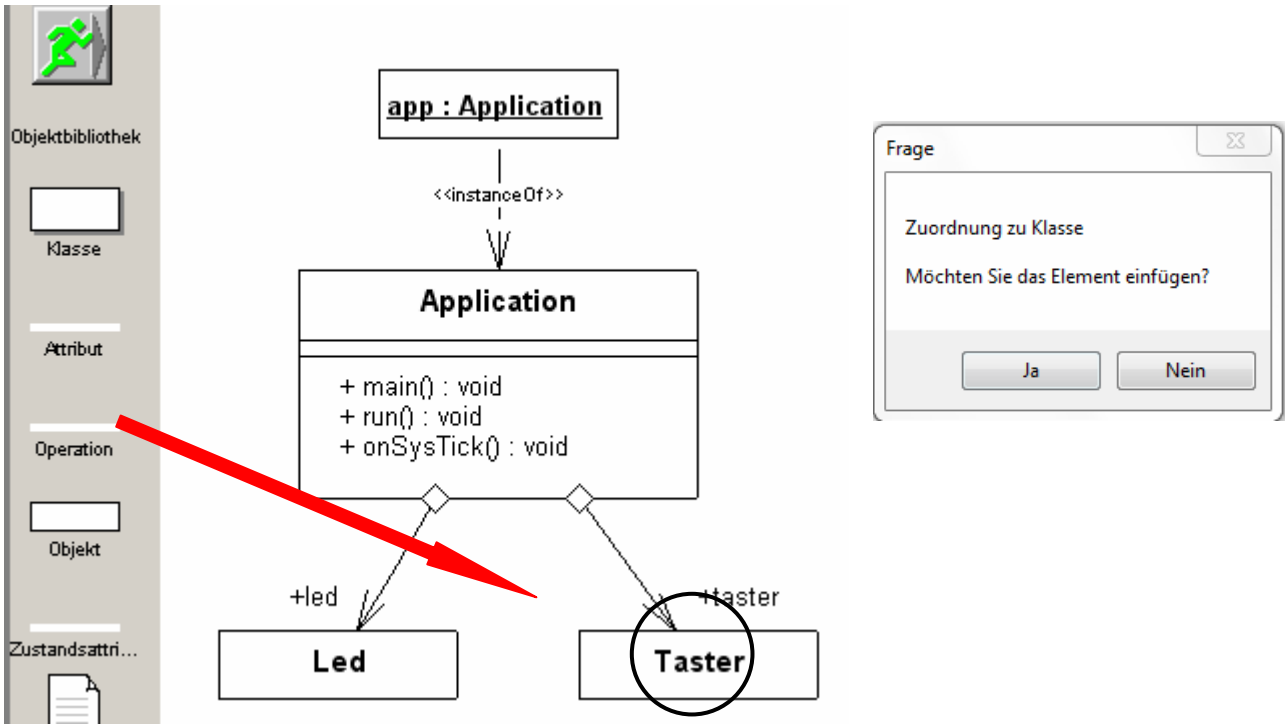
1. **Klassen einfügen und definieren**, ziehen Sie dazu ein Objekt vom Typ „Klasse“ per Drag & Drop aus der Objektbibliothek in das Diagramm. Definieren Sie den Namen der Klasse und setzen die Option „diese Klasse generieren“. Führen Sie diese Aktion für „Taster“ und „Led“ aus.



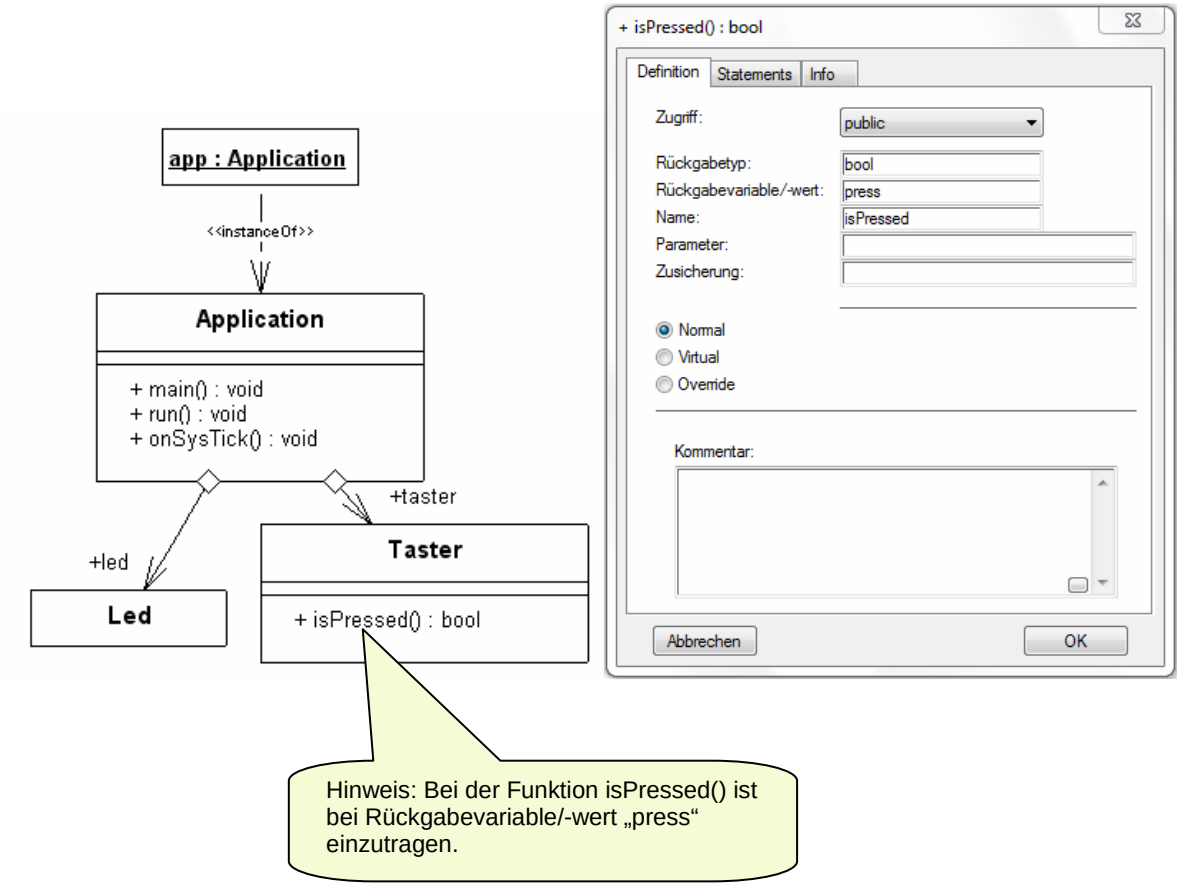
2. **Klassen verbinden**, selektieren Sie die Klasse, von der aus eine Verbindung gezogen werden soll. Ziehen Sie per Drag & Drop ausgehend vom roten Verteiler eine Verbindung auf die gewünschte Klasse. Wählen Sie den Verbindungstyp, zum Beispiel „Aggregation“ und beschriften Sie die Verbindung.



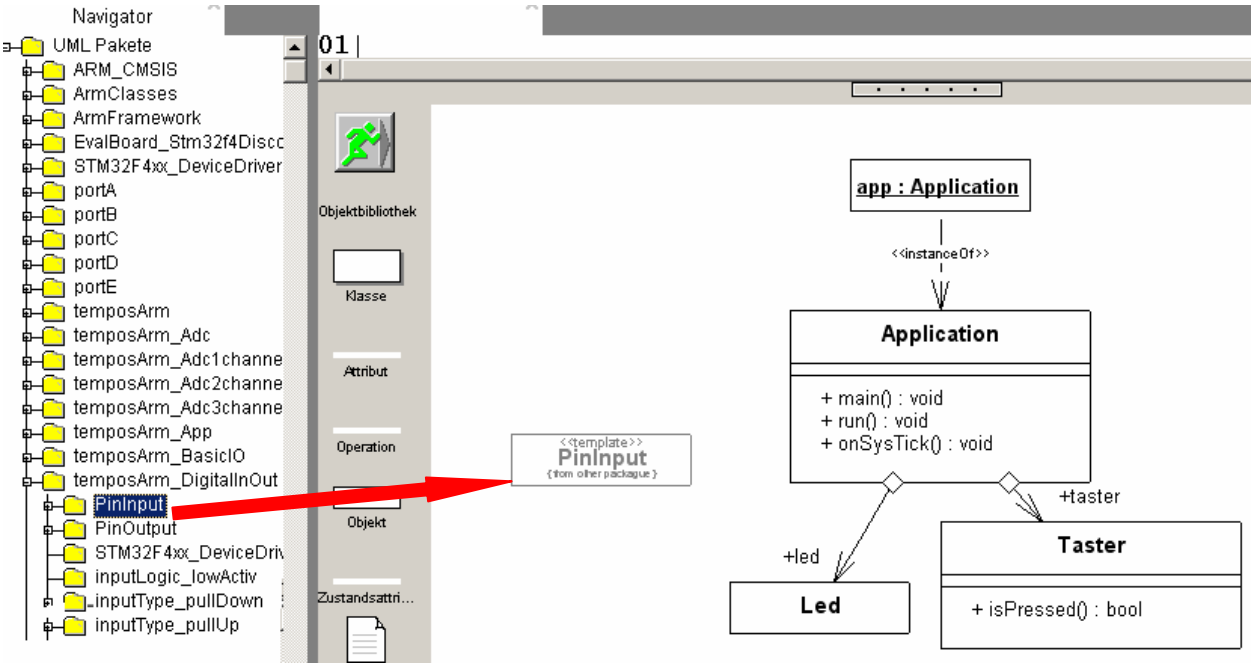
3. **Operationen und Attribute einfügen und definieren**, ziehen Sie dazu ein Objekt vom Typ „Operation“ oder „Attribut“ aus der Objektbibliothek auf die Klasse, in die das Attribut oder die Operation eingefügt werden soll. Bestätigen Sie die Sicherheitsabfrage zum Einfügen des Elementes.



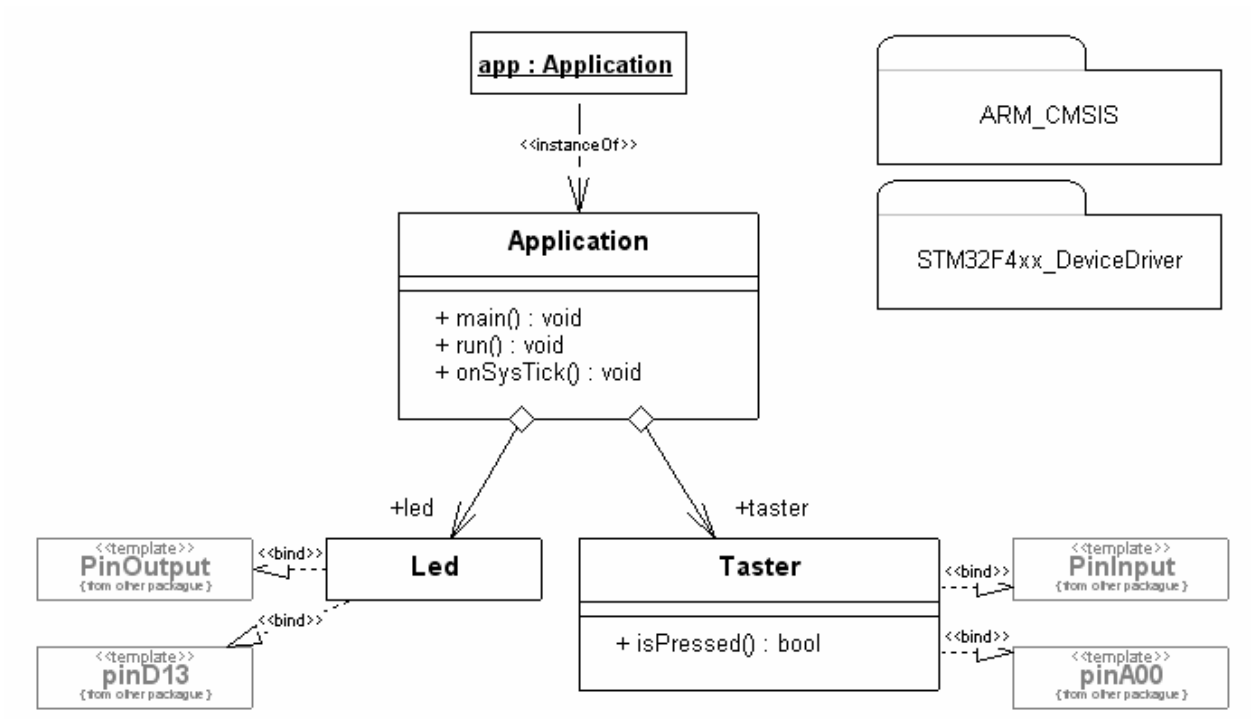
Definieren Sie Zugriff (Sichtbarkeit), Name, Typ und Parameter der Operation bzw. Zugriff (Sichtbarkeit), Name, Typ und Initialwert des Attributes.



4. **Templates einfügen**, klicken Sie mit der rechten Maustaste auf den „Navigator“ und wählen Sie die „UML-Pakete“ aus. Danach folgen Sie der Hierarchie und ziehen das gewünschte Template in das Diagramm. Zuletzt verbinden Sie das Template mit einer Klasse.



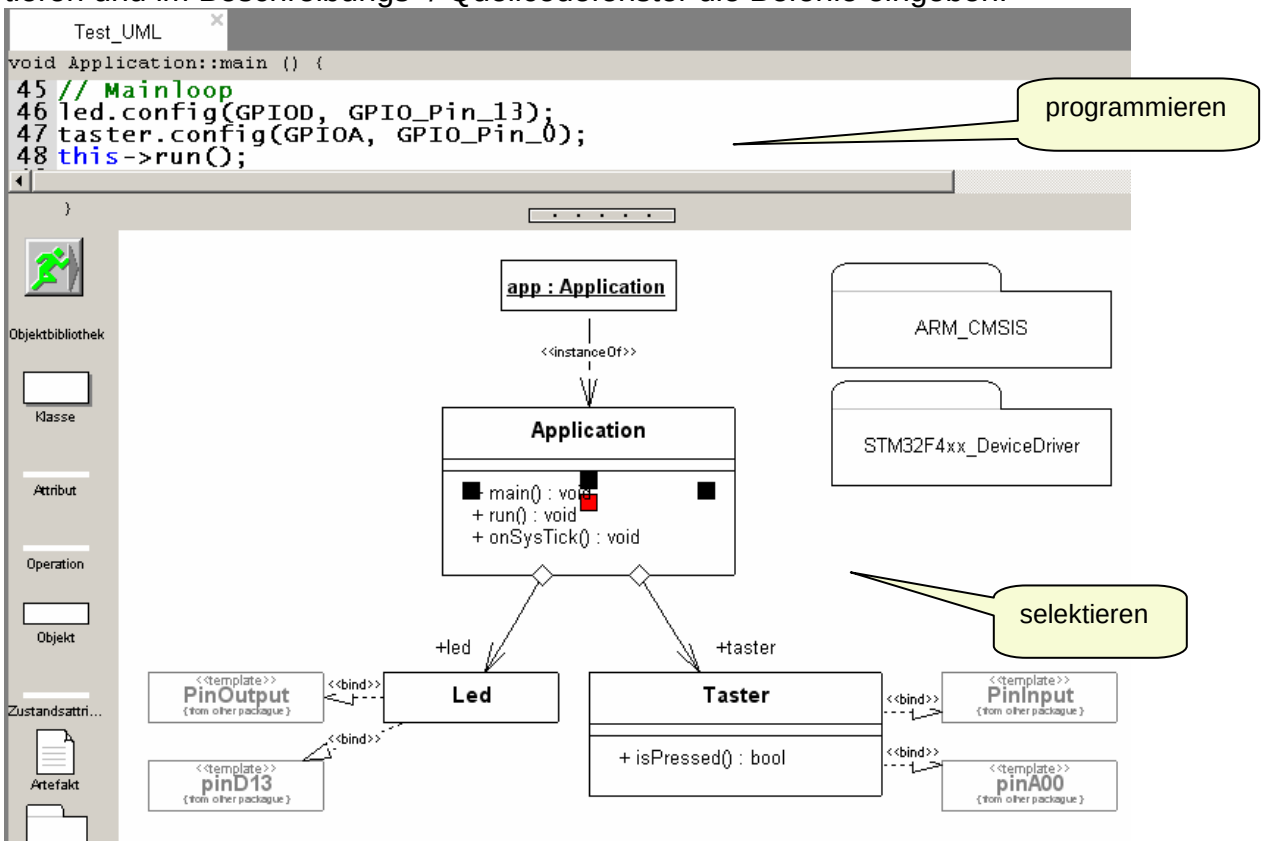
Vervollständigen Sie das Klassenmodell entsprechend der Abbildung.



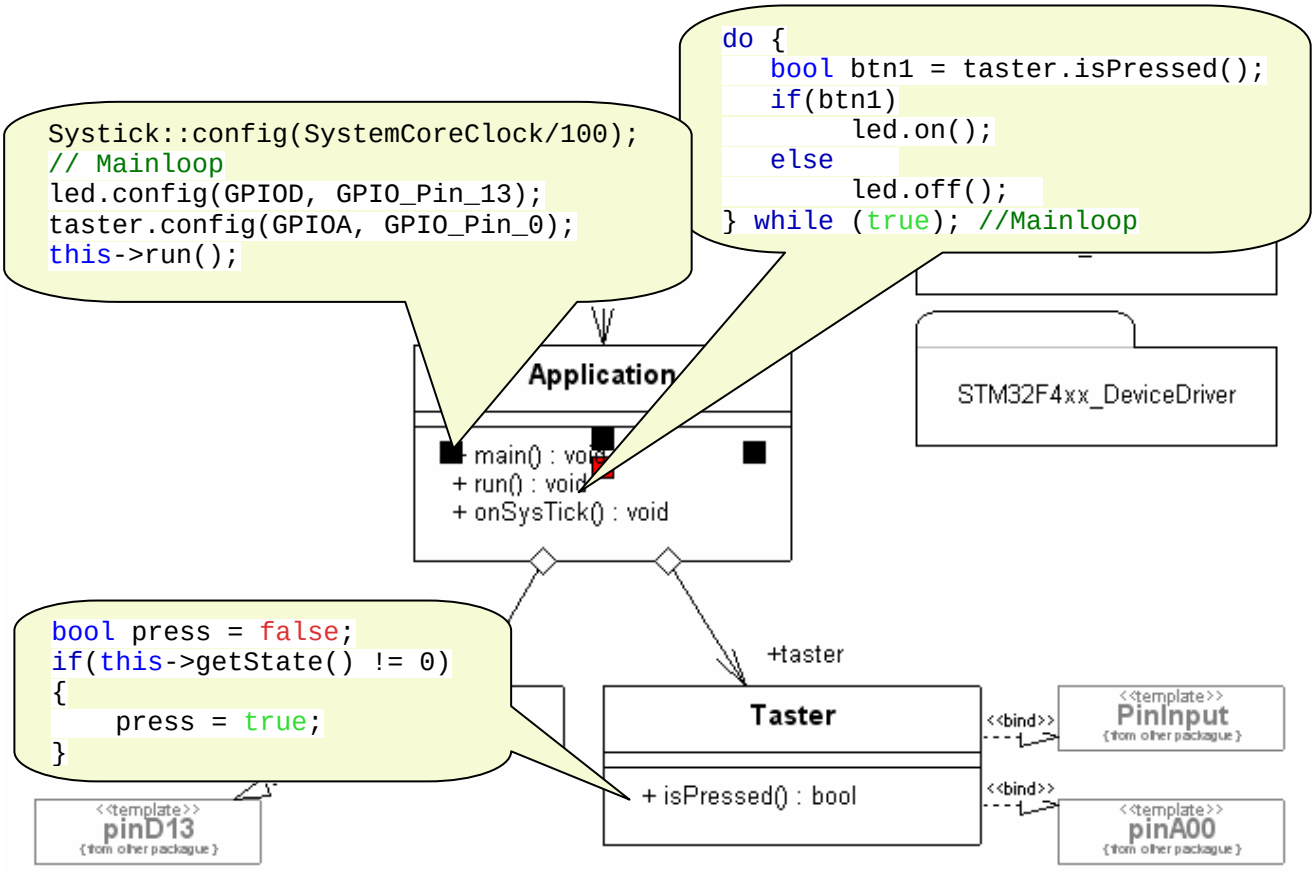
endgültiger Systementwurf mit dem UML Klassendiagramm

8.4.5 Systemverhalten programmieren

Die Operationen müssen mit der entsprechenden Logik versehen werden. Sie können den Quellcode der Operationen bearbeiten, indem Sie die gewünschte Operation selektieren und im Beschreibungs- / Quellcodefenster die Befehle eingeben.

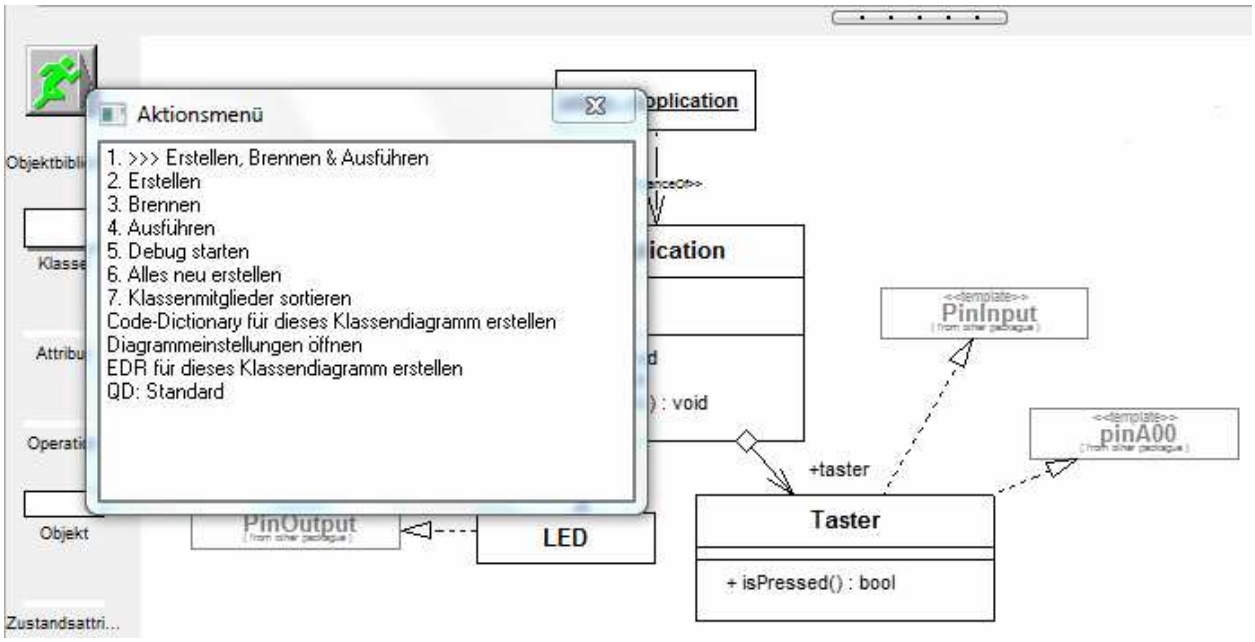


Überprüfen Sie ggf. die Realisierungsparameter.

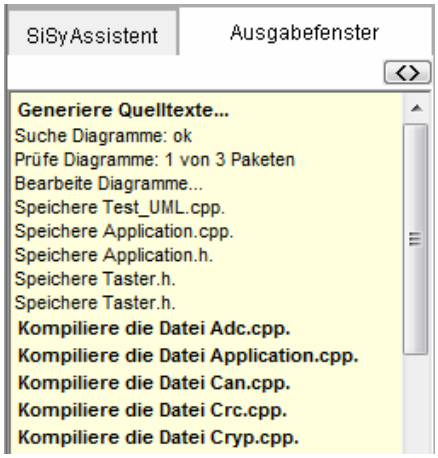


8.4.6 Übersetzen, Brennen und Testen

Die Codegenerierung aus dem UML Klassendiagramm, das Kompilieren, Linken und Brennen kann über das Aktionsmenü gestartet werden.
Verbinden Sie das Board mittels Programmierkabel mit dem PC:



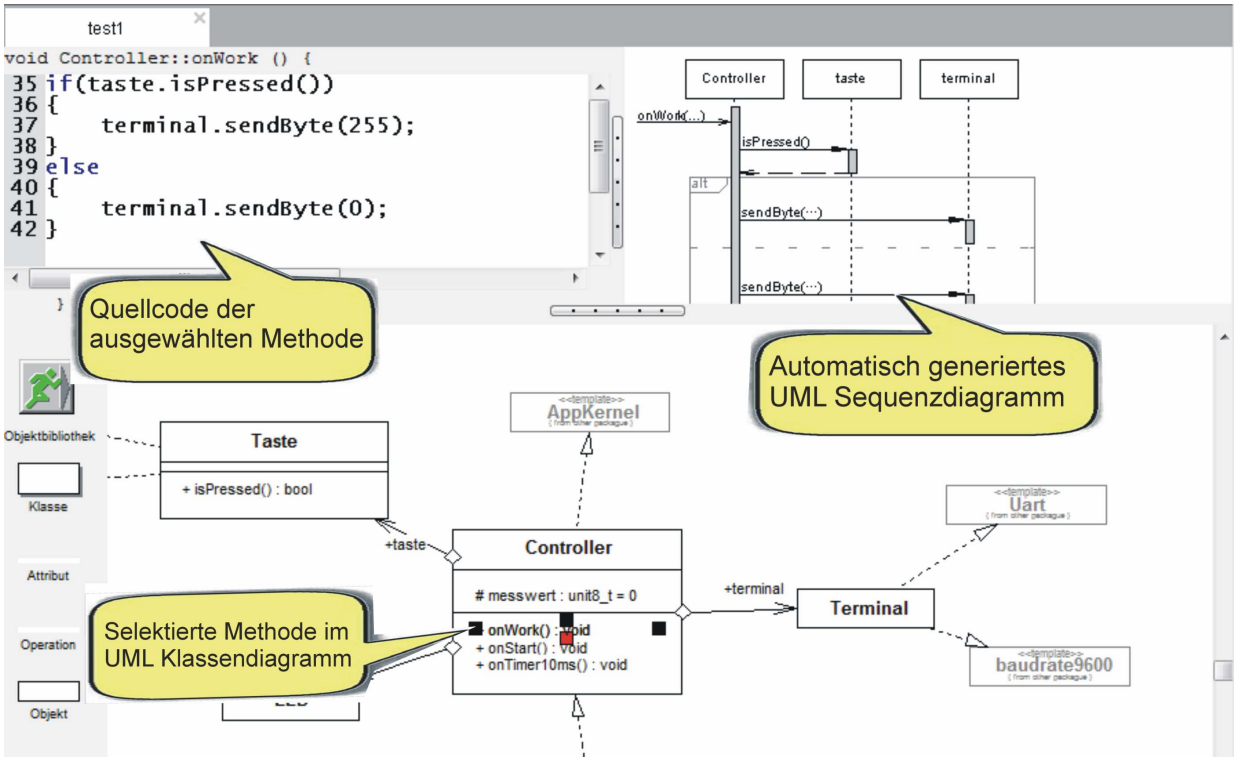
Sie erhalten im Ausgabefenster ein Protokoll der ausgeführten Aktionen.



8.5 Der Sequenzdiagrammgenerator

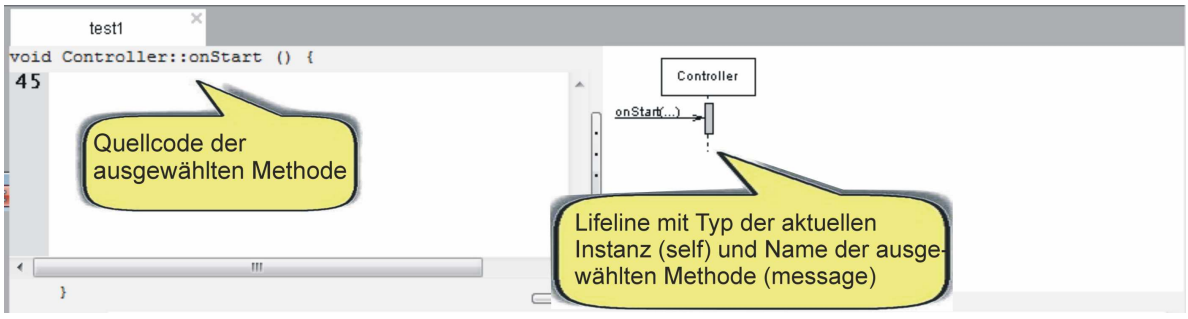
8.5.1 Einführung in das Sequenzdiagramm

Sequenzdiagramme sind eine wichtige Darstellungstechnik der UML für die Dokumentation des Systemverhaltens und der Interaktion von Objekten. SiSy verfügt über die Möglichkeit aus objektorientiertem Programmcode, wie zum Beispiel dem C++ Quellcode einer Klassenmethode, automatisch das entsprechende Sequenzdiagramm zu generieren. Dabei wird bereits während der Eingabe des Quellcodes simultan das Sequenzdiagramm erstellt. Der Entwickler ist damit in der Lage, die Reihenfolge der Nachrichten und die beteiligten Objekte mit deren Lebenszeit zu überblicken.

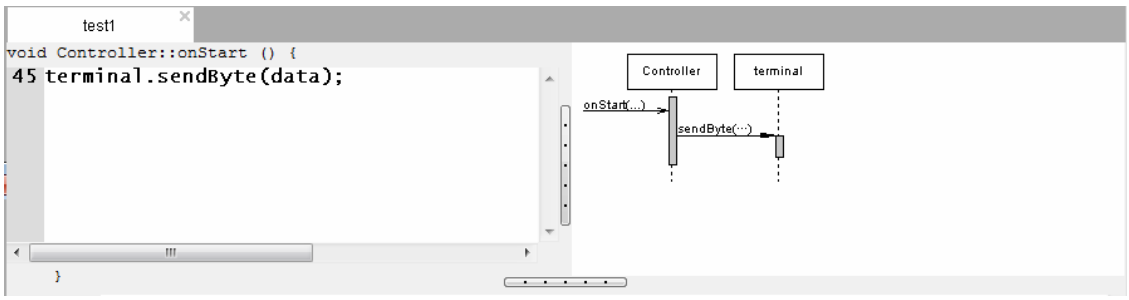


8.5.2 Sequenzen

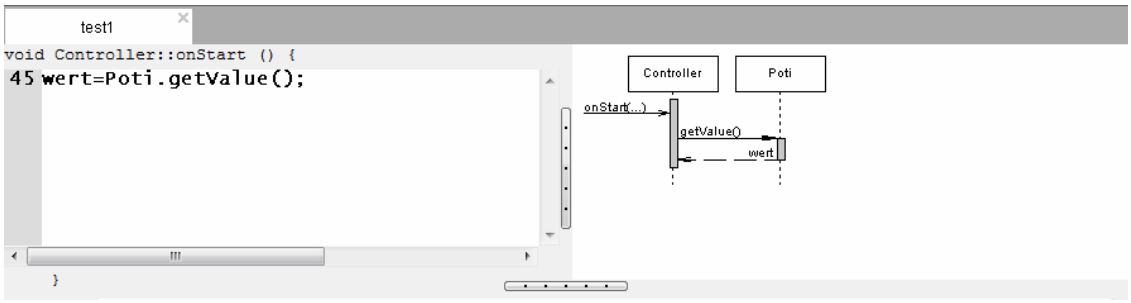
Eine leere Methode:



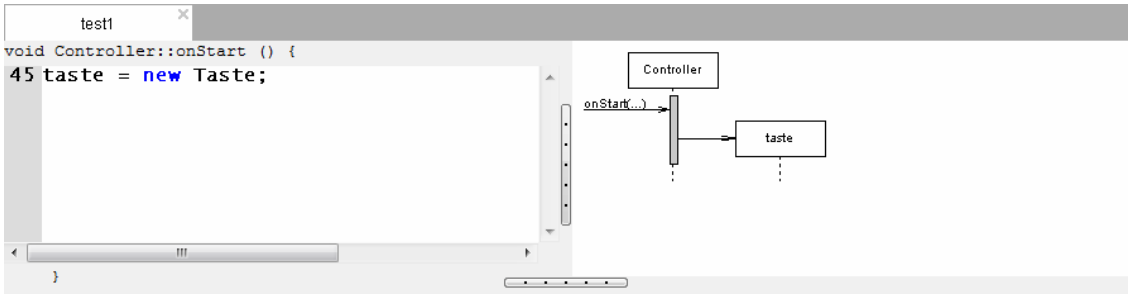
Eine einfache synchrone Nachricht: *Instanz.Nachricht (Parameter);*



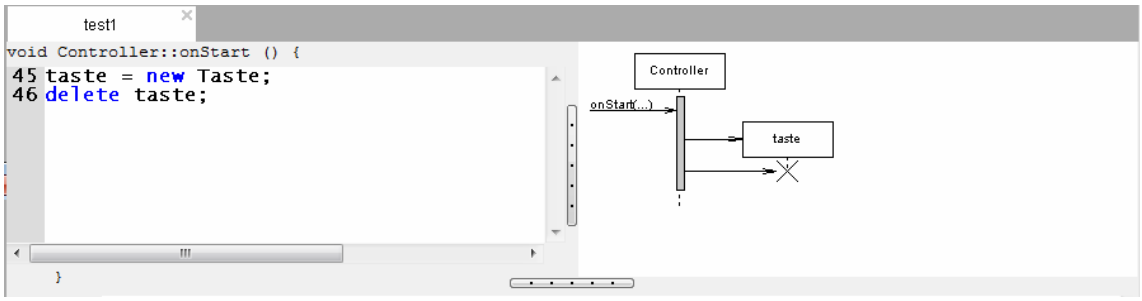
Eine Nachricht mit Antwort: *Variable = Instanz.Nachricht (Parameter);*



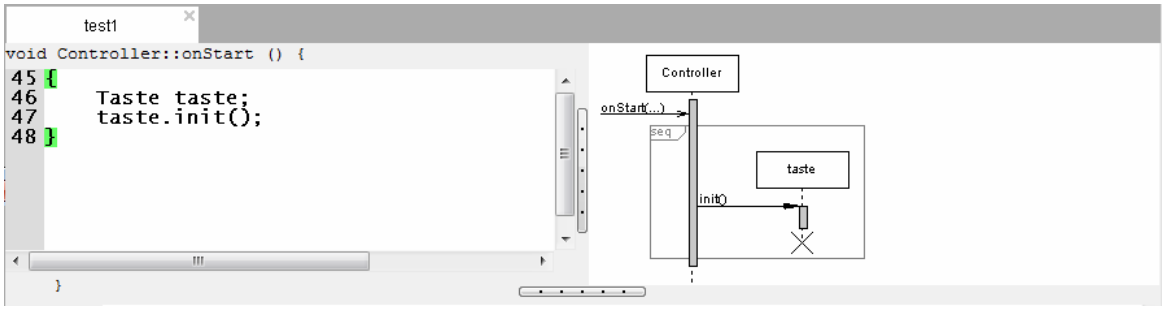
Explizites Erzeugen einer Instanz mit new: *Instanz = new Typ;*



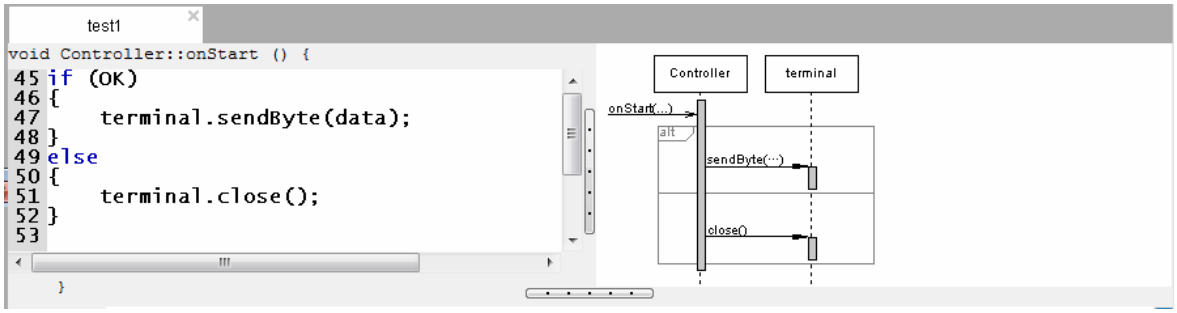
Explizites Zerstören einer Instanz mit delete: *delete Instanz;*



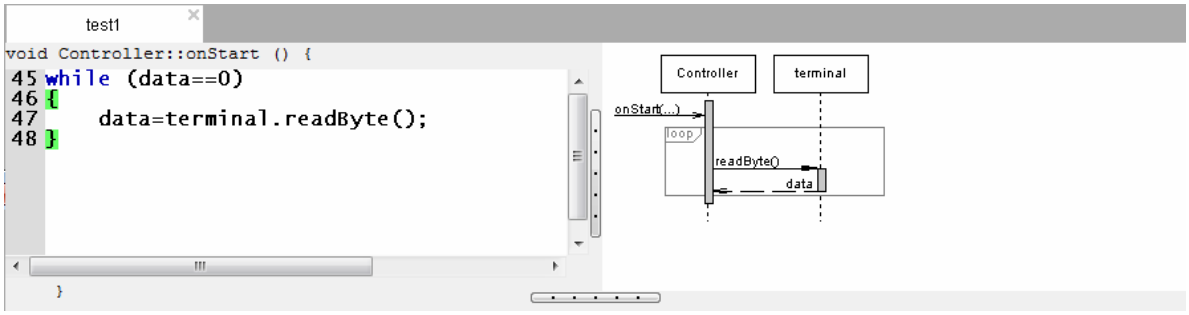
Eine lokale Instanz:



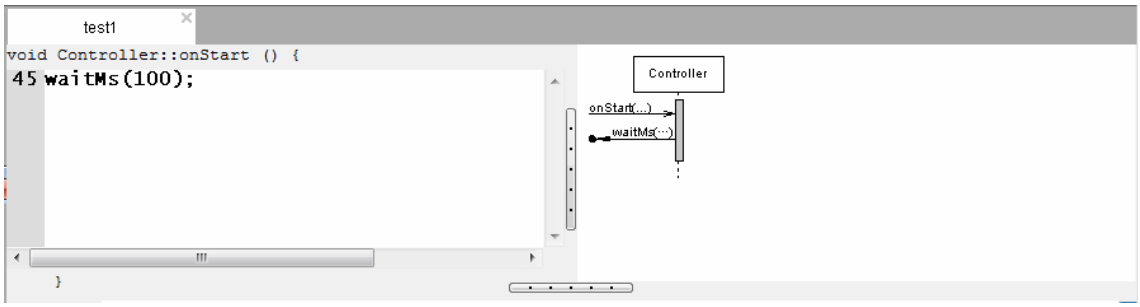
Eine Alternative:



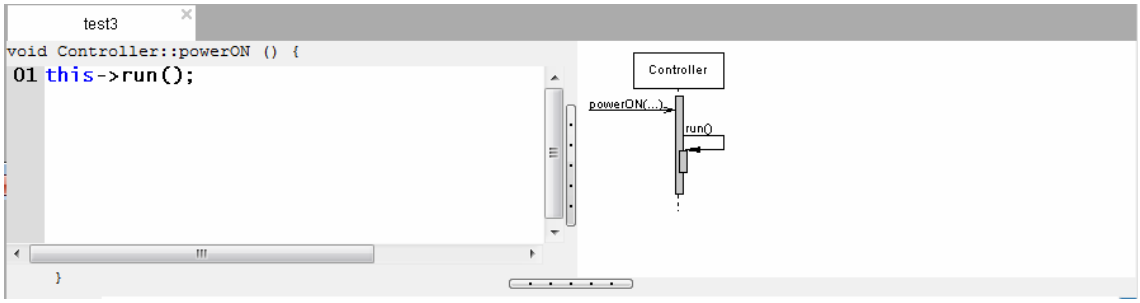
Eine Schleife:



Nachricht an unbekannten Empfänger:

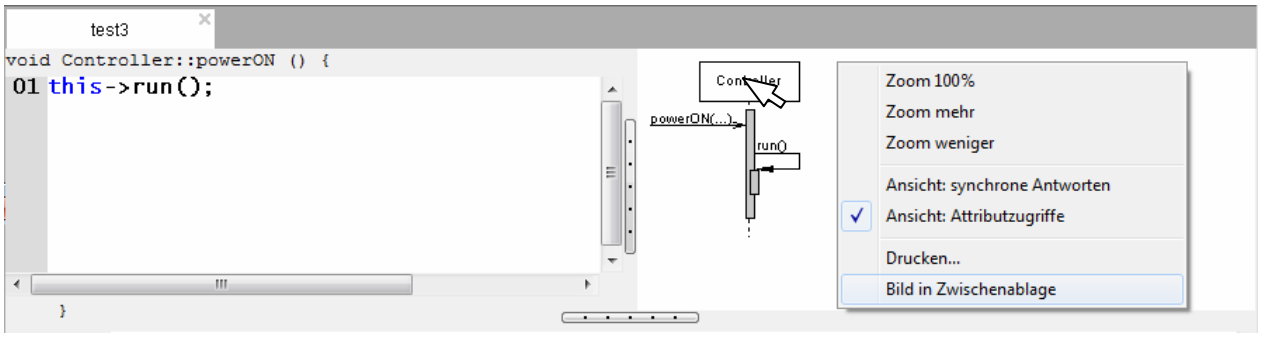


Nachricht an die aktuelle Instanz:



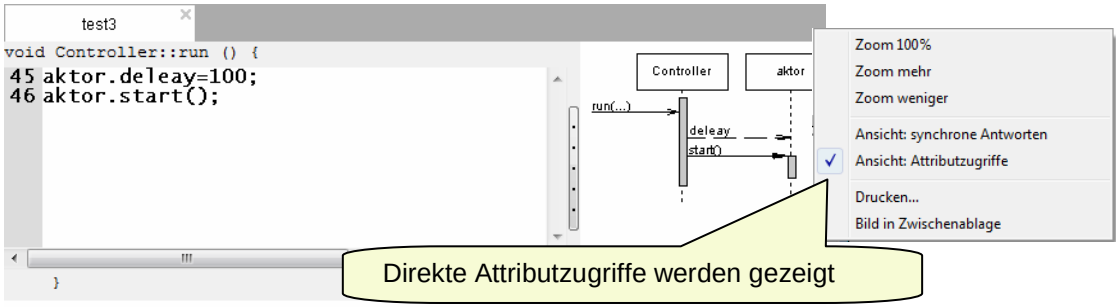
8.5.3 Sequenzen weiter verwenden

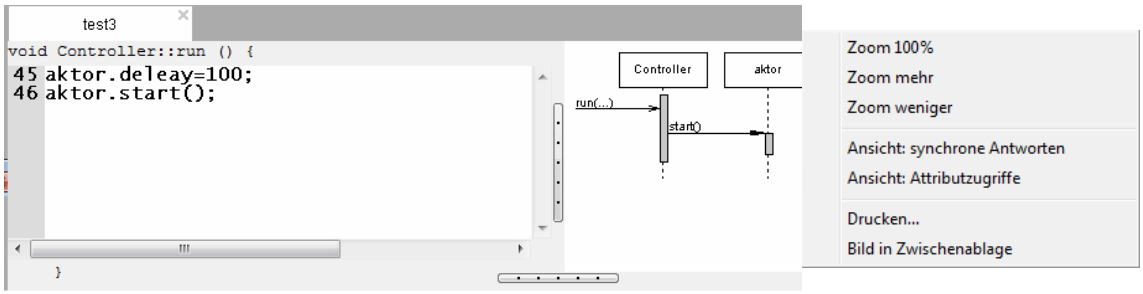
Sie können das automatisch erzeugte Sequenzdiagramm weiter verwenden, indem Sie es auf einen Drucker ausgeben oder die Darstellung in die Zwischenablage kopieren um diese zum Beispiel in eine Textverarbeitung einzufügen.

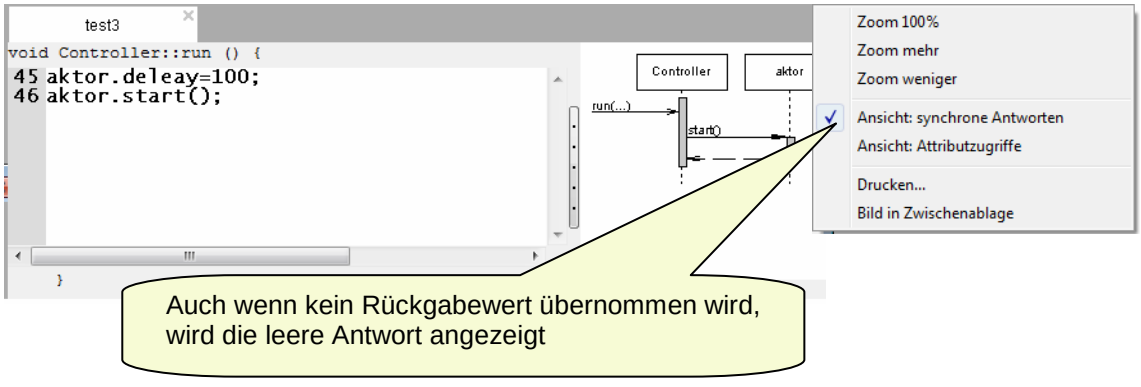


8.5.4 Besonderheiten des SiSy-Sequenzdiagramms

Für die direkten Zugriffe auf öffentliche Attribute einer Klasse trifft die UML derzeit keine verbindlichen Regeln. Es wird davon ausgegangen, dass Klassenattribute prinzipiell geschützt sind und nur über Methoden (Operationen) ein Zugriff erfolgen kann (Kapselung und Nachricht). Des Weiteren kann man davon ausgehen, dass auch eine leere (`void`) Antwort eine Antwort ist und angezeigt werden sollte. Dazu kann die Darstellung entsprechend über das Kontextmenü des Sequenzdiagramms angepasst werden.





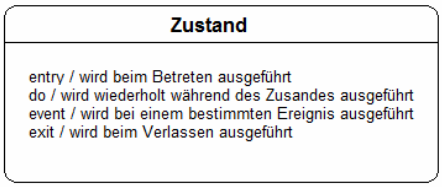


9 Programmieren mit dem UML Zustandsdiagramm

9.1 Einführung in die Zustandsmodellierung

Viele Problemstellungen in der Softwareentwicklung lassen sich als eine Folge von Zuständen und Zustandsübergängen (Zustandsautomat, state machine) verstehen und auch als solche implementieren. Dabei handelt es sich im Quellcode um oft recht aufwendige Fallunterscheidungen bzw. if/else Konstruktionen. SiSy ist mit dem entsprechenden Add-On in der Lage, aus UML-Zustandsdiagrammen den entsprechenden Quellcode automatisch zu generieren.

Notationsübersicht Zustandsdiagramm (Auszug)

Notation	Bezeichnung	Beschreibung
	Startknoten	Ein Startknoten aktiviert Abläufe. Startknoten besitzen nur ausgehende Kanten.
	Zustand	Zustände eines Objektes sind gekennzeichnet durch zustandsspezifische Aktivitäten beim Einnehmen, während und beim Verlassen des Zustandes (entry, do, exit)
	Zustands-Übergangs	Ein Zustandsübergang (transition) repräsentiert mindestens zwei Aktivitäten. Eine Aktivität vom Typ „exit“ beim alten und eine Aktivität vom Typ „entry“ beim neuen Zustand.
	Endknoten	Ein Endknoten beendet Abläufe und besitzt nur eingehende Kanten.

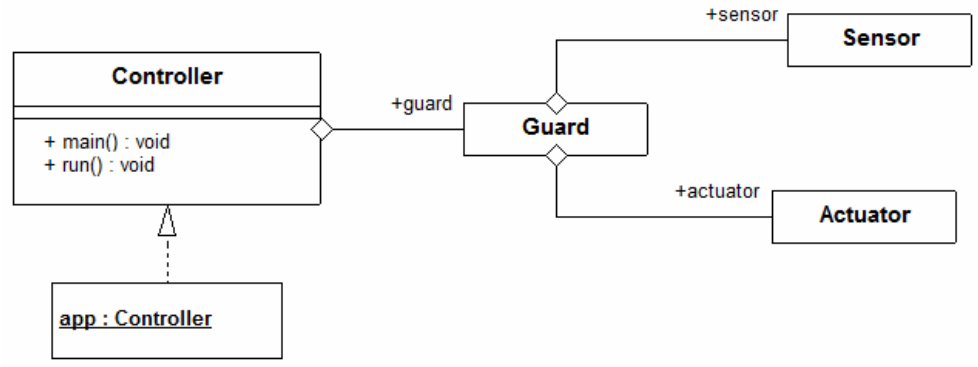
9.2 Erstellen von Zustandsdiagrammen

Hinweis:
Dieses Beispiel kann mit der Ausgabe SiSy Microcontroller ++ und SiSy AVR erstellt werden.

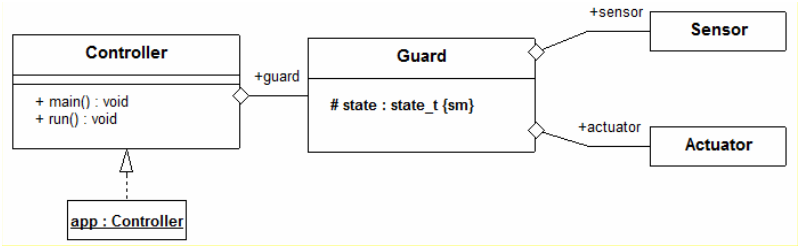
Das Zustandsdiagramm mit Quellcodegenerierung erhalten Sie über ein spezielles Attribut im Klassendiagramm der UML. Dieses Zustandsattribut kann in mit einem Zustandsdiagramm verfeinert werden. Dabei wird der Quellcode in spezielle Klassenmethoden generiert. Der Entwickler kann selbst Methoden definieren, die an der Verarbeitungslogik des Zustandsautomaten (state machine) beteiligt sind. Voraussetzung ist es, ein neues Projekt mit einem Klassendiagramm anzulegen.

9.2.1 Zielstellung

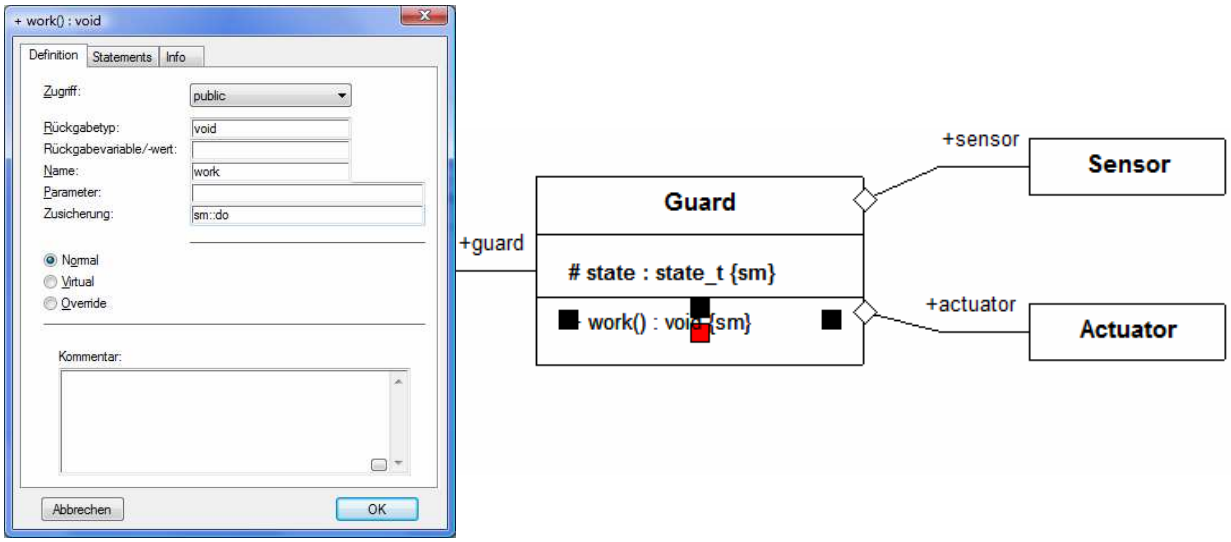
Die Aufgabe welche mit einem Zustandsdiagramm gelöst werden soll lautet wie folgt: Ein Controller steuert einen Wächter (guard) der einen Sensor überwacht und entsprechend einen Aktor (actuator) ansteuert. Als Referenzhardware dient ein myAVR Board.



Der Wächter (guard) ist ein Zustandsautomat (state machine). Somit ist der Klasse „Guard“ ein Zustandsattribut zuzuordnen. Dazu muss ein Objekt vom Typ „Zustands-attribut“ per Drag und Drop aus der Objektbibliothek in die betreffende Klasse gezogen werden. Die Zugehörigkeit zum Zustandsdiagramm wird mit der Zusicherung {sm} gekennzeichnet.

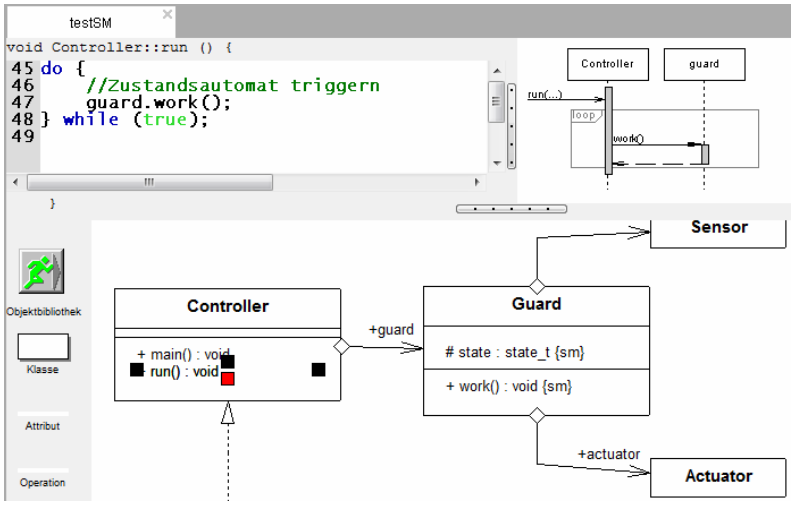


Für die Verarbeitungslogik des Zustandsautomaten soll eine eigene Methode angelegt werden, welche in der Hauptschleife der Mikrocontrolleranwendung (mainloop) zyklisch aufgerufen wird. Damit die Methode im Zustandsdiagramm verfügbar ist, muss diese mit einer entsprechenden Zusicherung versehen werden (z.B.: {sm::do}). Dazu muss aus der Objektbibliothek per Drag und Drop eine Operation (Methode) in die betreffende Klasse gezogen und wie folgend dargestellt definiert werden.

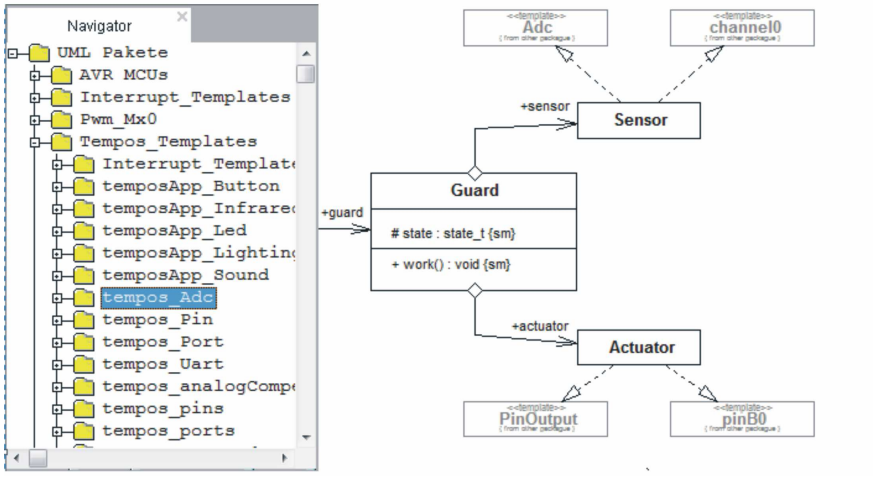


Diese Methode benötigt keine Statements. Der Codegenerator fügt beim Bilden der Anwendung hier Code entsprechend der Modellierung im Zustandsdiagramm ein. Es ist jedoch nötig, die Verarbeitungsfunktion kontinuierlich aufzurufen um die Zustandswechsel auszulösen (trigger). Das erfolgt in diesem Fall in der Methode *run*(). Es gibt in den SiSy-Klassenbibliotheken spezielle Klassen, die bereits einen Zustandsautomaten ab-

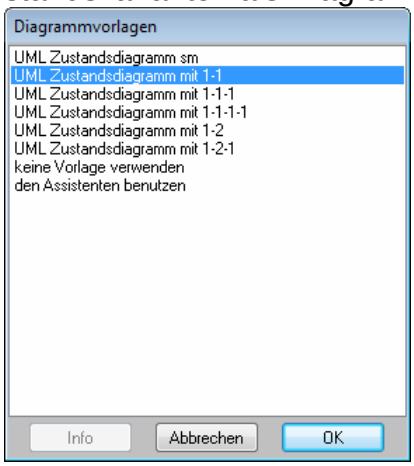
bilden. Wird von solchen Basisklassen abgeleitet, sind die Auslöser (trigger) für den Zustandsautomaten bereits implementiert.

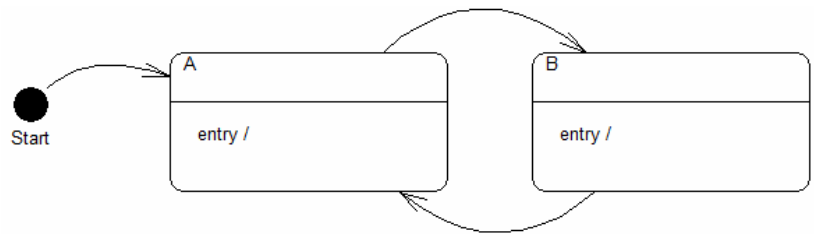


Für eine schnelle Realisierung der Klassen „Sensor“ und „Actuator“ wird im folgenden Beispiel das myAVR Programmiermodell *Tempos* benutzt. Dabei müssen den Klassen lediglich geeignete Templates (Muster) aus der Tempos-Bibliothek zugewiesen werden.



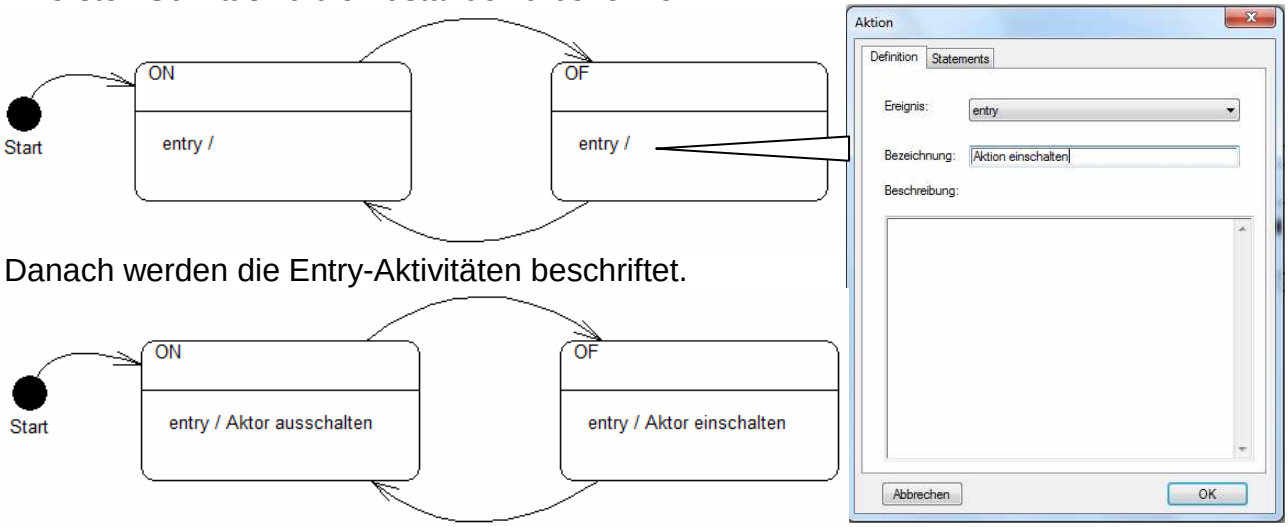
Die Modellierung der Zustände und Zustandswechsel erfolgt in Verfeinerung des Zustandsattributes. Dazu selektieren Sie das Zustandsattribut und wählen im Kontextmenü den Menüpunkt *Nach unten (öffnen)*. Dabei wird bereits eine Auswahl typischer Zustandsvarianten als Diagrammvorlage angeboten. Wählen Sie die Vorlage 1-1.





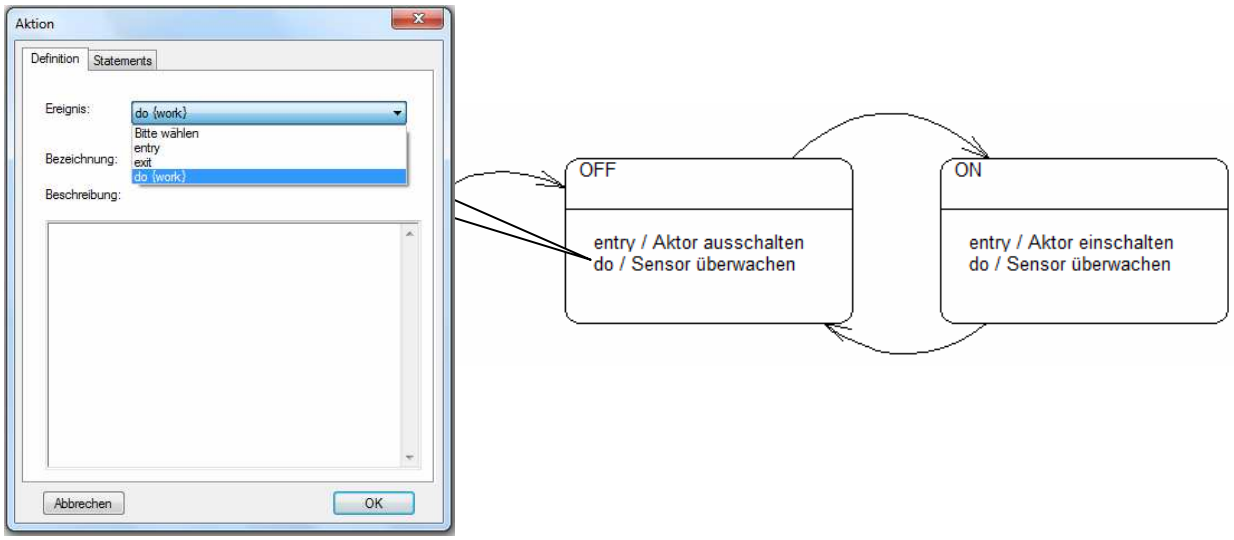
Die Zustände des Wächters sollen ON und OFF sein. Dabei wird nach dem Start der Zustand OFF eingenommen. Die Zustandswechsel erfolgen in Abhängigkeit der Sensorwerte. Wird der Zustand OFF eingenommen, muss der Aktor ausgeschaltet werden, wird der Zustand ON eingenommen, ist der Aktor einzuschalten. Während der Zustände ist der Sensor fortlaufend zu überwachen.

Im ersten Schritt sind die Zustände zu benennen.

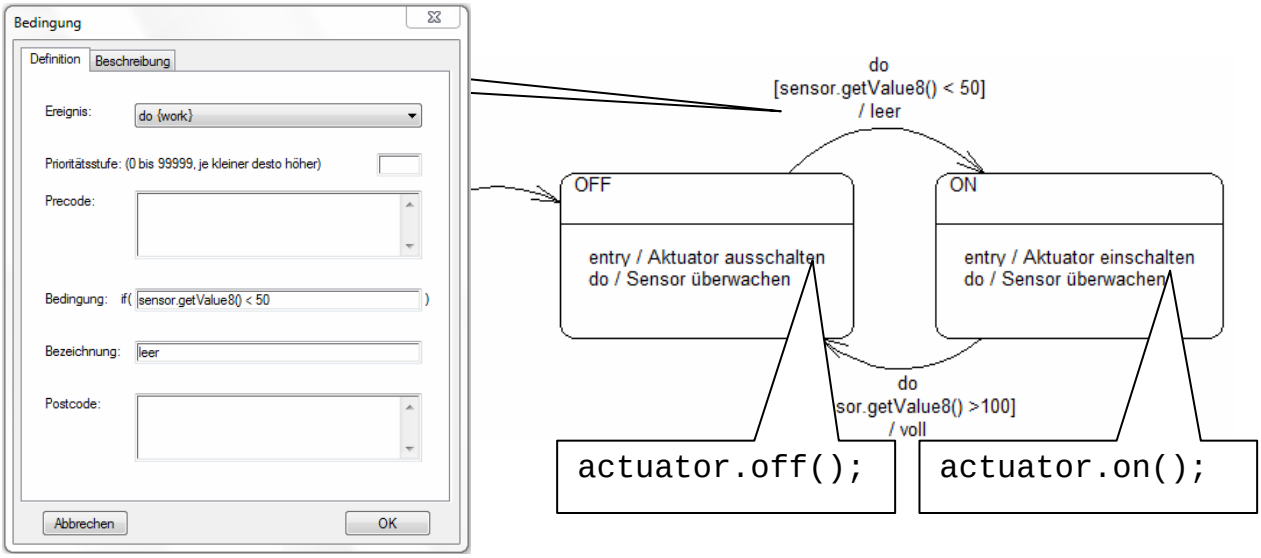


Danach werden die Entry-Aktivitäten beschriftet.

Um weitere Aktionen (`entry/do/exit`) einzufügen, müssen per Drag und Drop Objekte vom Typ „Aktion“ in die betreffenden Zustände gezogen werden.
Das Ereignis/Aktivitätstyp ist auszuwählen und die Aktivität zu beschriften.



Die Zustandsübergänge müssen definiert werden indem festgelegt wird, wann der Zustandswechsel erfolgen darf und unter welchen Bedingungen. Des Weiteren sollte der Zustandsübergang eine Bezeichnung erhalten.



Ist das Zustandsmodell erarbeitet, können die nötigen Statements für den Programmcode eingearbeitet werden. Dazu wird die betreffende Aktivität selektiert und im Quellcodeeditor die entsprechenden Befehle eingegeben. Danach kann im dazugehörigen Klassendiagramm (rechte Maustaste Menüpunkt *nach oben*) das Programm gebildet, übertragen und getestet werden.

10 Zusätzliche Werkzeuge

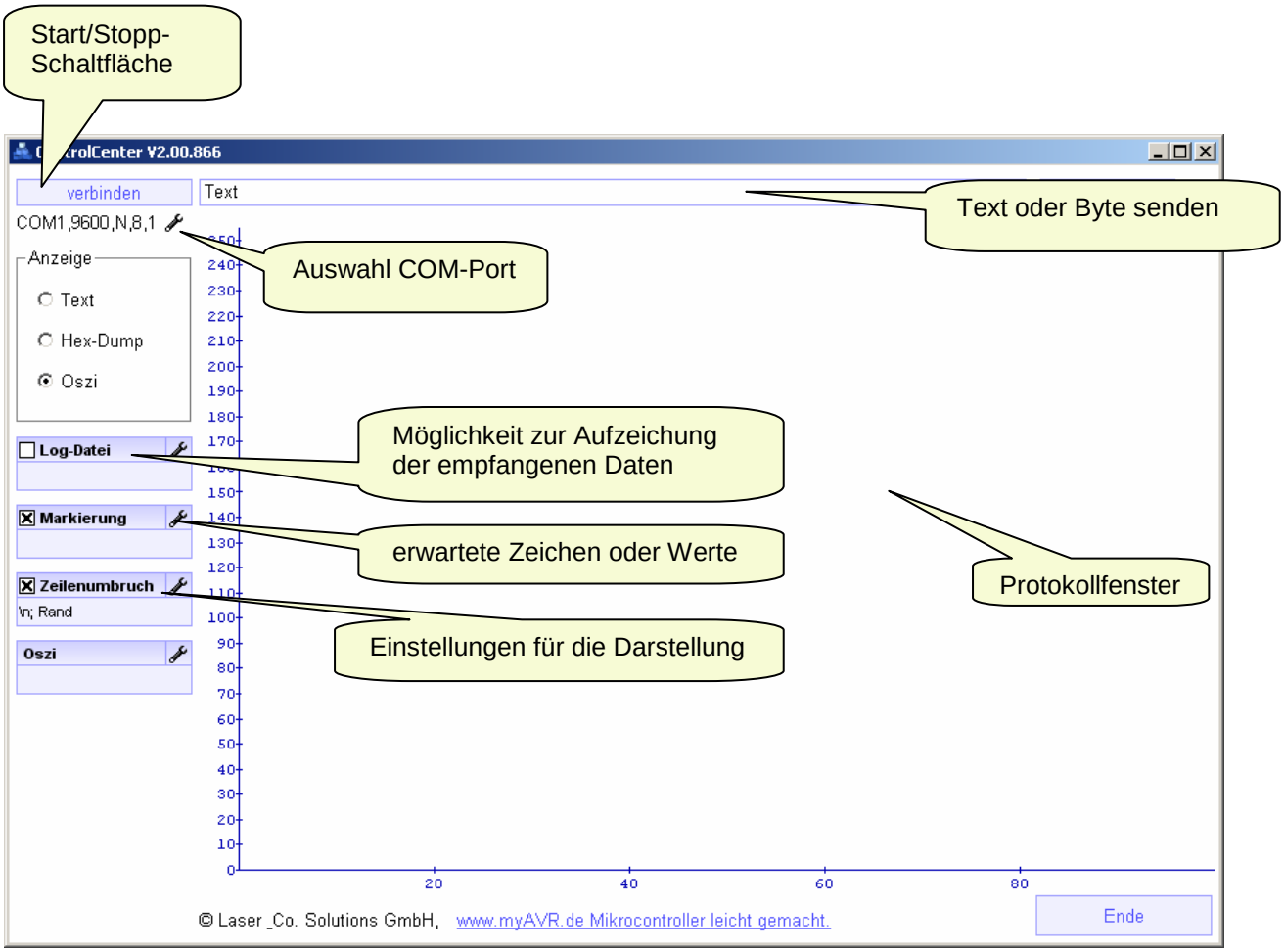
10.1 Einführung

Bestimmte Add-Ons installieren als zusätzliche Komponenten externe Programme. Diese sind nach der Installation über den Menüpunkt *Werkzeuge* erreichbar.

10.2 Das ControlCenter

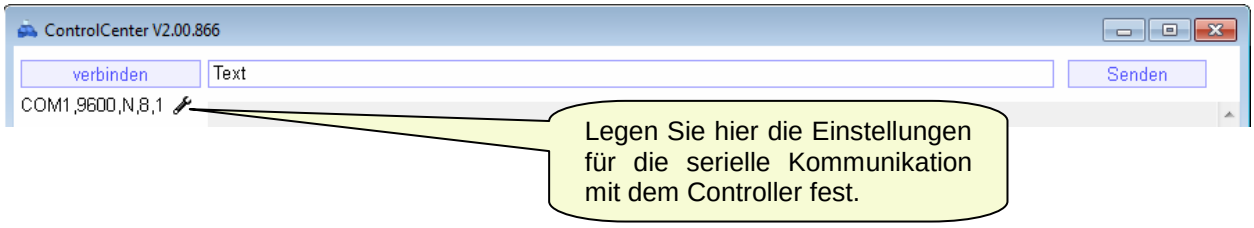
10.2.1 Einleitung

Das ControlCenter ist ein universelles Terminalprogramm zur Kommunikation mit Mikrocontrollerapplikationen, die über eine serielle Schnittstelle (UART) oder USB Anbindung mit virtuellem COM-Port zum PC verfügen. Es kann für Test- und Debug-Meldungen sowie Visualisierung und Protokollierung von Messdaten genutzt werden. Dazu bietet das ControlCenter umfangreiche Konfigurationsmöglichkeiten.



10.2.2 Besonderheiten für myAVR Systemboards

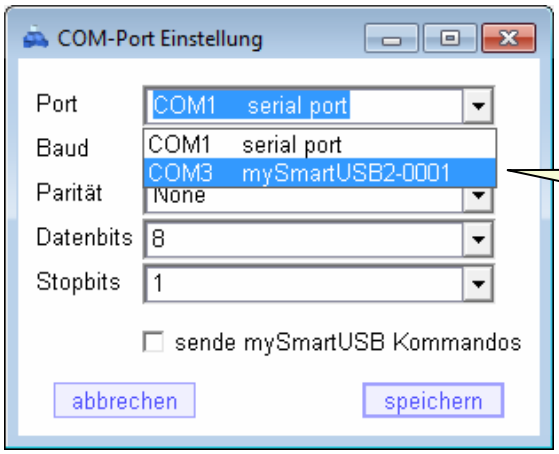
Bei einem myAVR Systemboard können Sie die Spannungsversorgung des Boards aus dem ControlCenter heraus steuern. Dazu wählen Sie die Einstellungen für die serielle Verbindung und Power-Control.



Einstellung zur seriellen Verbindung des Terminals, beim USB-Board die des virtuellen COM-Ports. (siehe Gerätemanager)

Power-Control
Nur beim Anschluss eines myAVR Boards MK2 bzw. mySmartControl MK2 Programmers verwenden.

Power-On / Off
Beim Verlassen des ControllCenters kann dem myAVR Board der Zustand Power-On oder OFF zugewiesen werden.



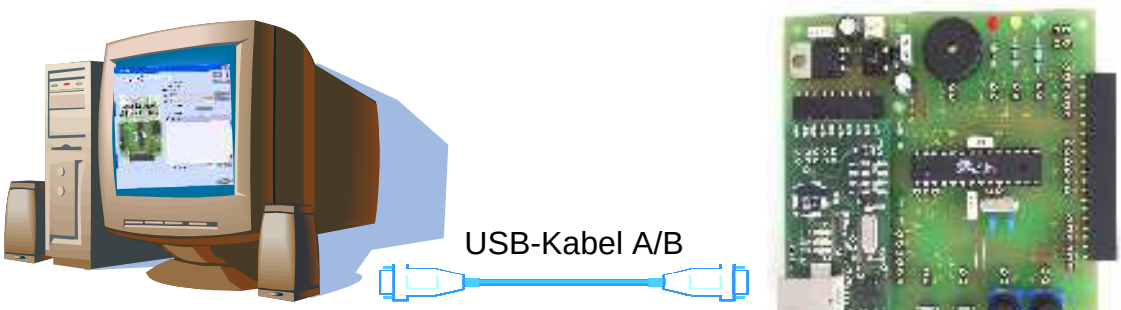
Der virtuelle COM-Port steht entsprechend Gerätemanager als Auswahl zur Verfügung.

Bei einer externen Spannungsversorgung, zum Beispiel durch einen 9V Block oder ein Netzteil, hat die Power-Control-Funktion keine Wirkung auf die Boardspannung. In diesem Fall bezieht sich die Wirkung ausschließlich auf die RESET-Leitung, um den Controller zu starten bzw. zu stoppen.

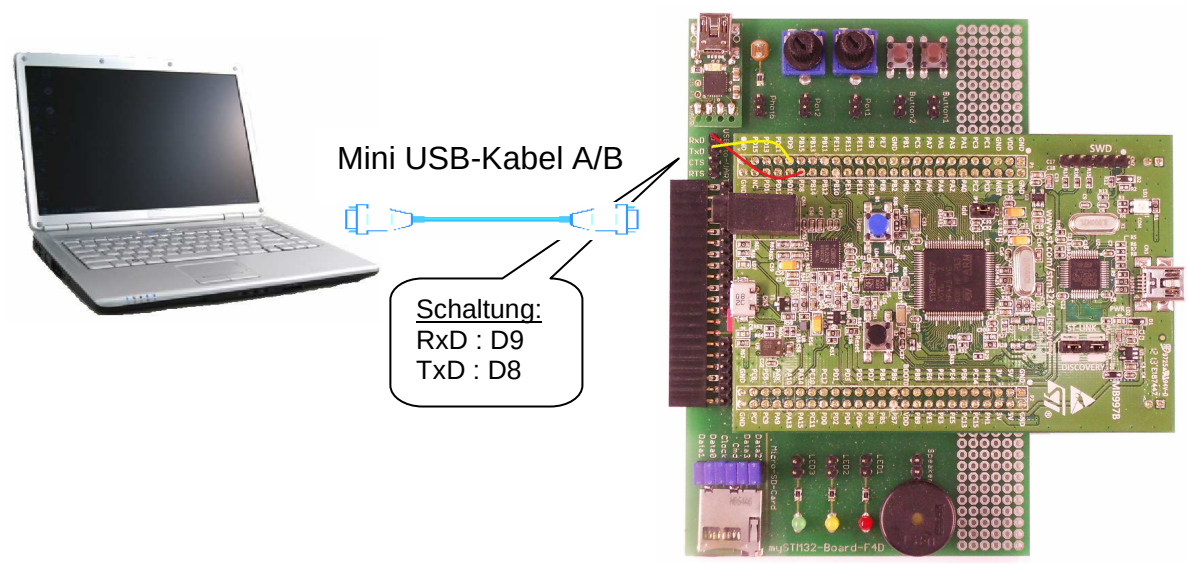
10.2.3 Kommunikation mit dem Controller

Grundlagen (USB-Variante)

Das myAVR Board MK2 verfügt über den USB Programmer mySmartUSB MK2. Dieser stellt gleichzeitig einen virtuellen COM-Port für die Kommunikation zur Verfügung.



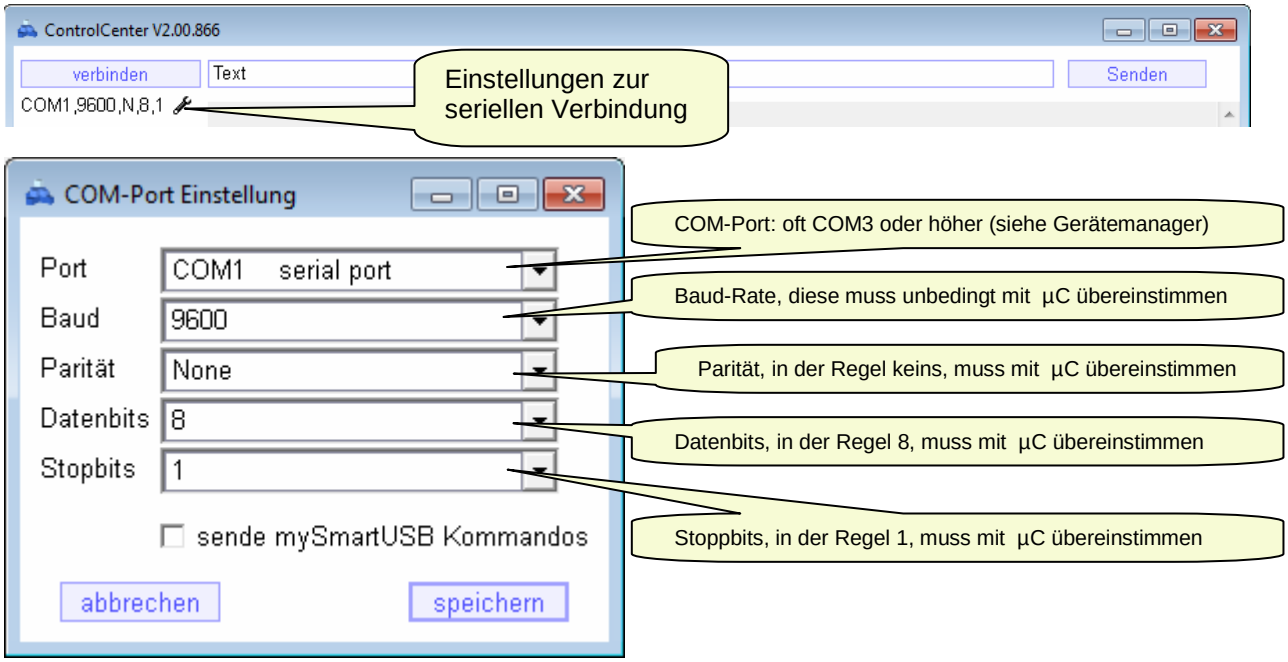
Beispiel mit myAVR Board MK2



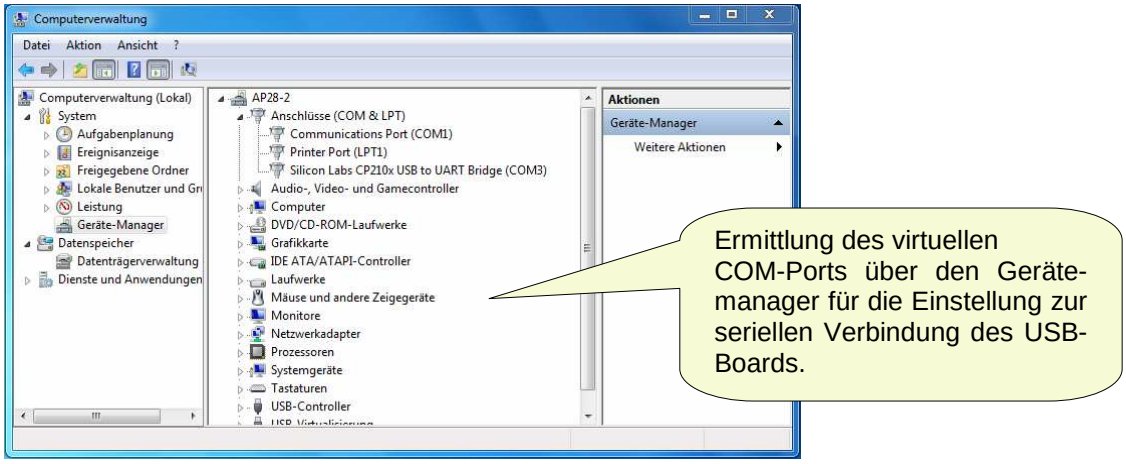
Beispiel mit STM32F4-Discovery und einem mySTM32-Board-F4D

Einstellungen für die seriellen Verbindung

Für eine erfolgreiche Kommunikation mit dem myAVR Board ist es wichtig, dass Sender und Empfänger von seriellen Daten die gleichen Parameter für die Datenübertragung konfiguriert haben. Auf der PC Seite werden die Kommunikationsparameter im Control-Center über die Schaltfläche für die Einstellungen der Verbindung konfiguriert.



Beachten Sie die jeweiligen Beschreibungen zum Controller-Board.



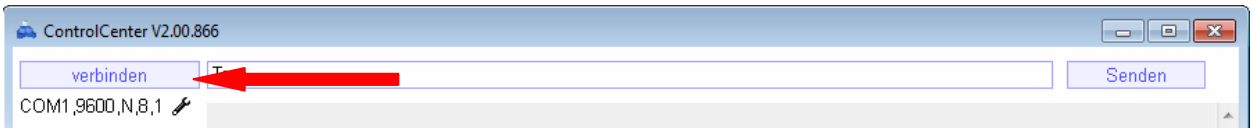
Im Mikrocontrollerprogramm sind die gleichen Parameter bei der Initialisierung der UART zu wählen. Beachten Sie die eingestellte Taktquelle (hier Quarz mit 3,6 MHz).

```
#define     F_CPU   3686400
#define     BAUD    9600
#include    <avr\io.h>
//---------------------------------------------------------------
// UART initialisieren
void initUART()
{
    sbi(UCSRB,3);   // TX aktiv
    sbi(UCSRB,4);   // RX aktivieren
    UBRRL=(uint8_t)(F_CPU/(BAUD*16L))-1;       // Baudrate festlegen
    UBRRH=(uint8_t)((F_CPU/(BAUD*16L))-1)>>8;  // Baudrate festlegen
}
```

Beispiel für die Konfiguration der UART des Mikrocontrollers (µC) mit 9600 Baud.

Daten empfangen vom Controller

Das ControlCenter empfängt Daten über den gewählten COM-Port und stellt diese im Protokollfenster dar. Damit können Statusmeldungen, Fehlermeldungen oder auch Messwerte erfasst werden. Die Voraussetzung ist, dass die serielle Verbindung hergestellt (USB Kabel), korrekt konfiguriert und aktiviert wurde. Die Kommunikation beginnt mit dem Betätigen der Schaltfläche „verbinden“, diese wird ersetzt mit der Schaltfläche „trennen“, womit die Kommunikation beendet werden kann.



Darstellung der empfangen Daten

Die empfangenen Daten werden fortlaufend im Protokollfenster dargestellt. Der Darstellungsmodus kann während der Kommunikation umgeschaltet werden.
Das ControlCenter bietet folgende Darstellungsmodi:

- Text
- Zahl
  - Dezimal
  - Hexadezimal
- Grafik (Oszi)

Die Daten des folgenden Programmbeispiels sollen als Testdaten dienen:

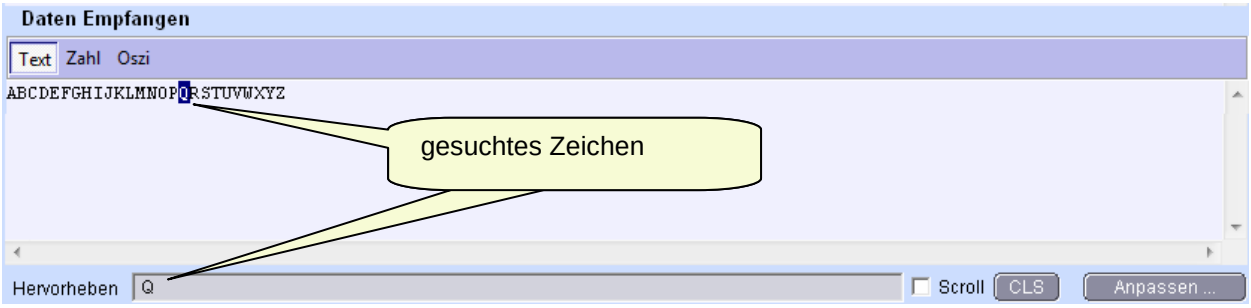
```
//-------------------------------------------------------
main()
{
    uint8_t a='A';                  // Zeichen
    initUART();                 // Initialisierungen
    while (true)                // Mainloop-Begin
    {
        putChar(a);                 // Zeichen senden
        a++;                        // nächstes Zeichen
        myWait_1000ms();            // Pause
    }                           // Mainloop-Ende
}
//-------------------------------------------------------
```

Der Textmodus

Der Textmodus dient zur Visualisierung alphanumerischer Werte im ASCII Format (Zeichenketten). Zahlen, die mit Zeichenketten gesendet werden, müssen vom Mikrocontrollerprogramm zuvor ins ASCII-Format gewandelt werden (siehe `itoa` und `sprintf`).

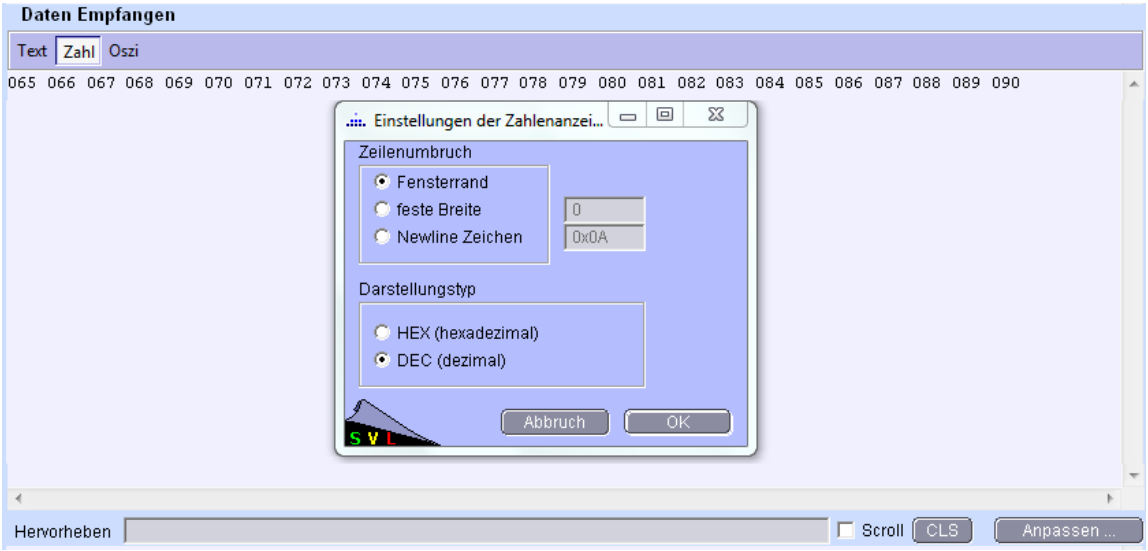
Über die Schaltfläche „Anpassen “ lässt sich das Protokollfenster weiter konfigurieren. Die

Zeilenbreite und das Zeichen für einen Zeilenumbruch lassen sich auswählen. Es können wichtige/gesuchte Textstellen im Protokollfenster hervorgehoben werden.

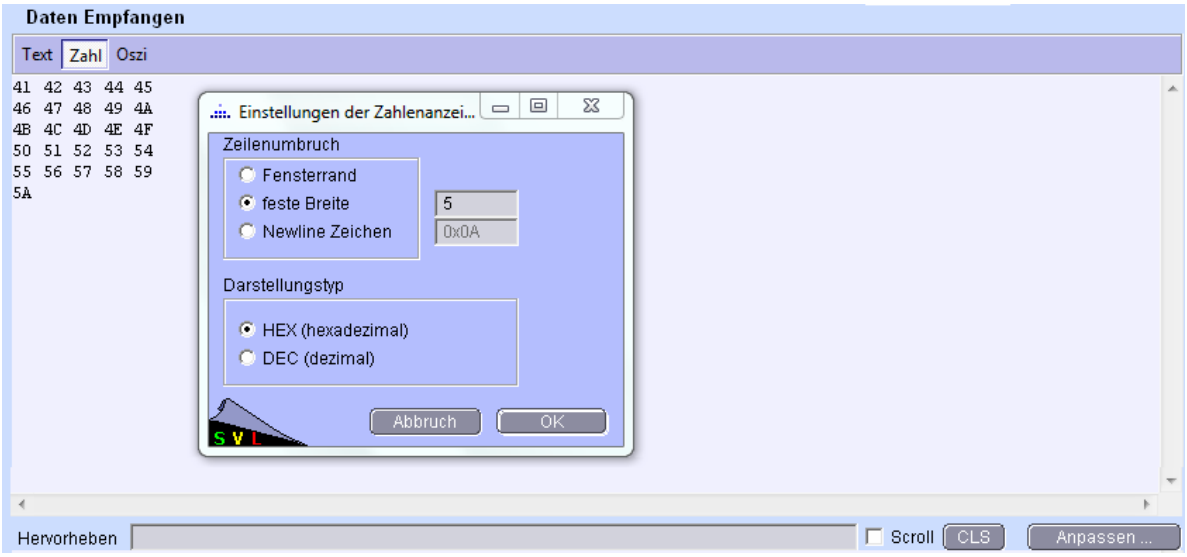


Der Zahlenmodus

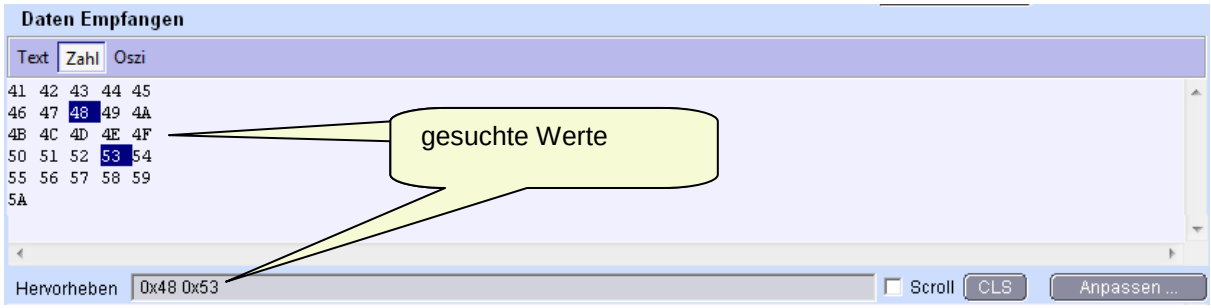
Der Zahlenmodus visualisiert die empfangenen Datenbytes (8 Bit, Werte von 0 bis 255) als Dezimal- oder Hexadezimalzahlen. Dezimalzahlen werden dreistellig mit führender Null dargestellt.



Die Anzeige kann über die Schaltfläche „Anpassen“ weiter konfiguriert werden. Dabei werden die Zeilengröße oder das Zeilenumbruchzeichen sowie das Format der Zahlendarstellung (Hexadezimal/Dezimal) ausgewählt.

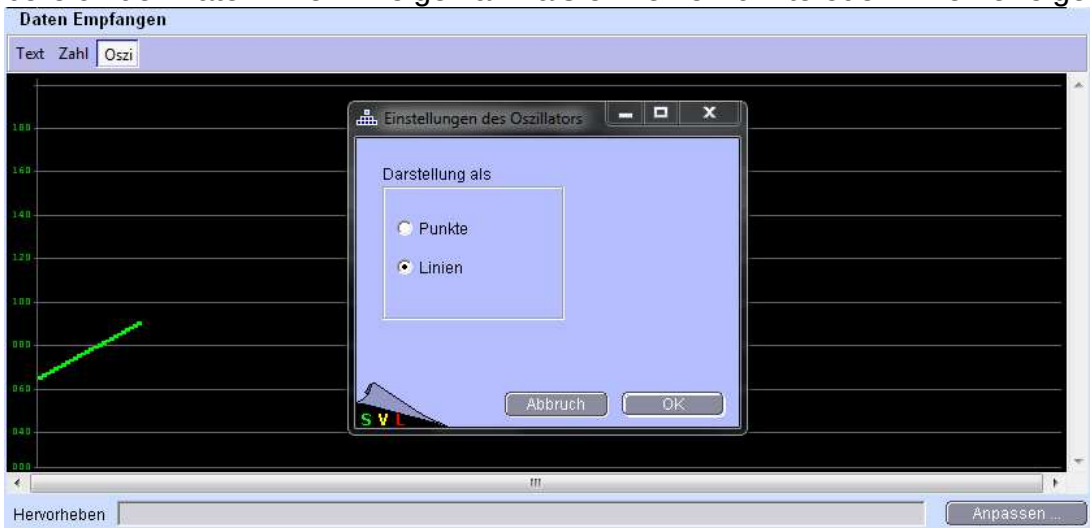


Es können wichtige/gesuchte Zahlenwerte im Protokollfenster hervorgehoben werden. Hexadezimale Zahlen sind durch den Präfix 0x zu kennzeichnen. Mehrere Werte können mit Leerzeichen getrennt angegeben werden.



Der Grafikmodus

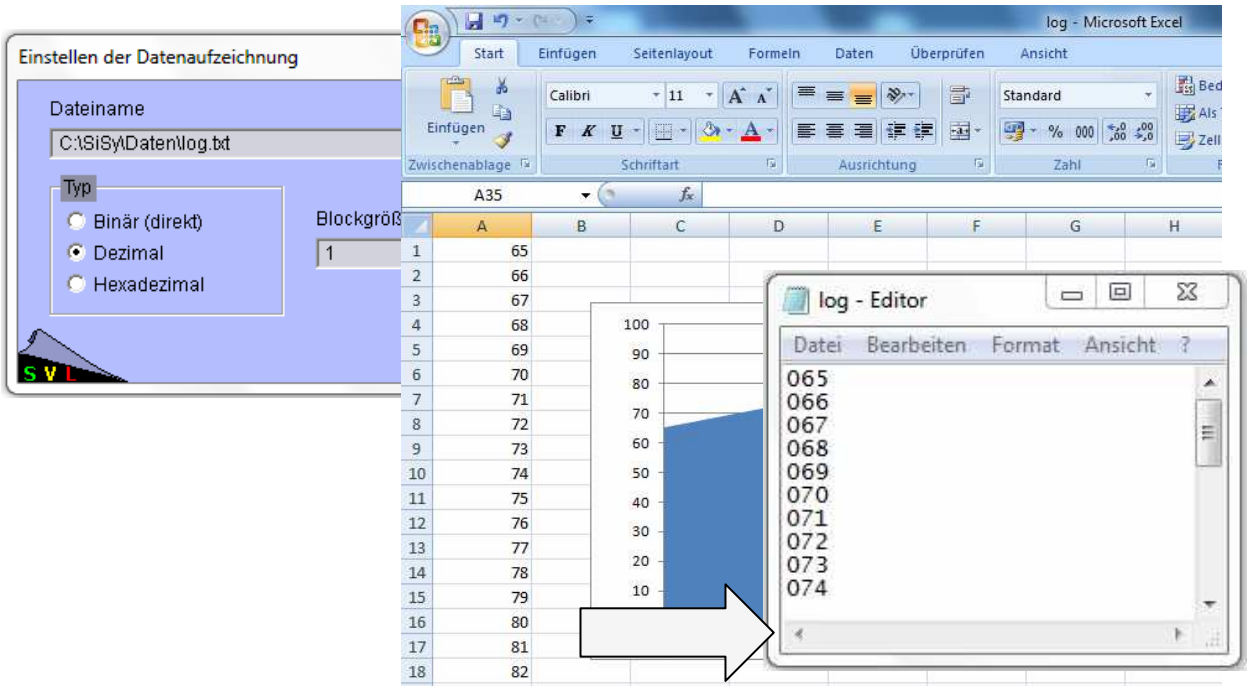
Messwerte können auch grafisch visualisiert werden. Dabei werden die Werte fortlaufend und byteweise (Wertebereich 0 bis 255) als Punkte in einem Koordinatensystem visualisiert. Die X-Achse repräsentiert den zeitlichen Verlauf, die Y-Achse den Wertebereich der Daten. Die Anzeige kann als einzelne Punkte oder Linien erfolgen.



10.2.4 Empfangene Daten speichern

Bei der Erfassung von Messdaten die vom Mikrocontroller an den PC gesendet werden ist es oft wichtig, diese in eine Datei zu speichern. Damit wird eine Weiterverarbeitung der Daten in ent-sprechenden Programmen (z.B.: EXCEL) möglich. Das ControlCenter ermöglicht es, Protokolldaten aufzuzeichnen (Log-Datei, Rekorderfunktion). Um diese Funktion zu aktivieren, wählen Sie die Option „Daten aufzeichnen“

Über die Schaltfläche „Einstellungen“ muss der Dateiname und Pfad der Log-Datei angegeben werden. Zusätzlich ist es möglich, das Format und die Blockgröße (Zeilenumbruch) festzulegen. Die Weiterverarbeitung der Daten erfolgt entsprechend der Möglichkeiten der Zielanwendung (z.B.: Öffnen, Importieren oder Zwischenablage).



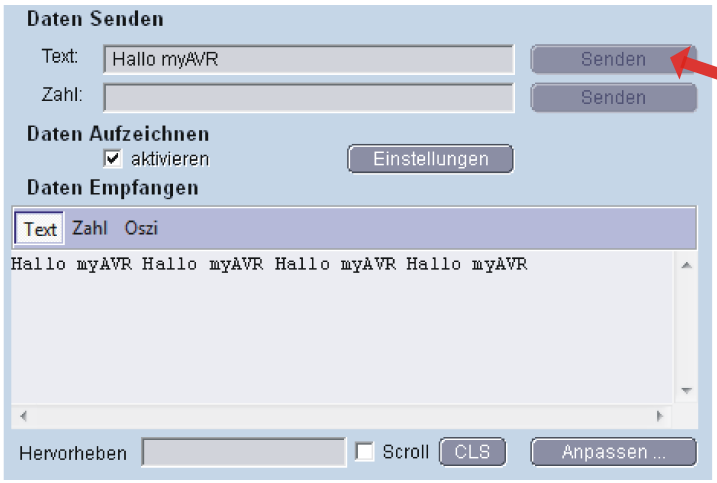
10.2.5 Daten an den Controller senden

Über das ControlCenter können Daten im Text- oder Zahlenformat an den Mikrocontroller gesendet werden. Das folgende Programm dient als Veranschaulichung dieser Funktion.

```
//------------------------------------------------------
main()
{
    uint8_t zeichen;
    initUART();                 // Initialisierungen
    while (true)                // Mainloop-Begin
    {
        zeichen=getChar();      // Zeichen abholen
        putChar(zeichen);       // Zeichen zurücksenden
    }                           // Mainloop-Ende
}
//------------------------------------------------------
```

Text senden

Es können einzelne Zeichen, aber auch Zeichenketten gesendet werden. Mit der zugehörigen Schaltfläche „Senden“ wird immer der gesamte Inhalt der Eingabezeile „Text“ gesendet.



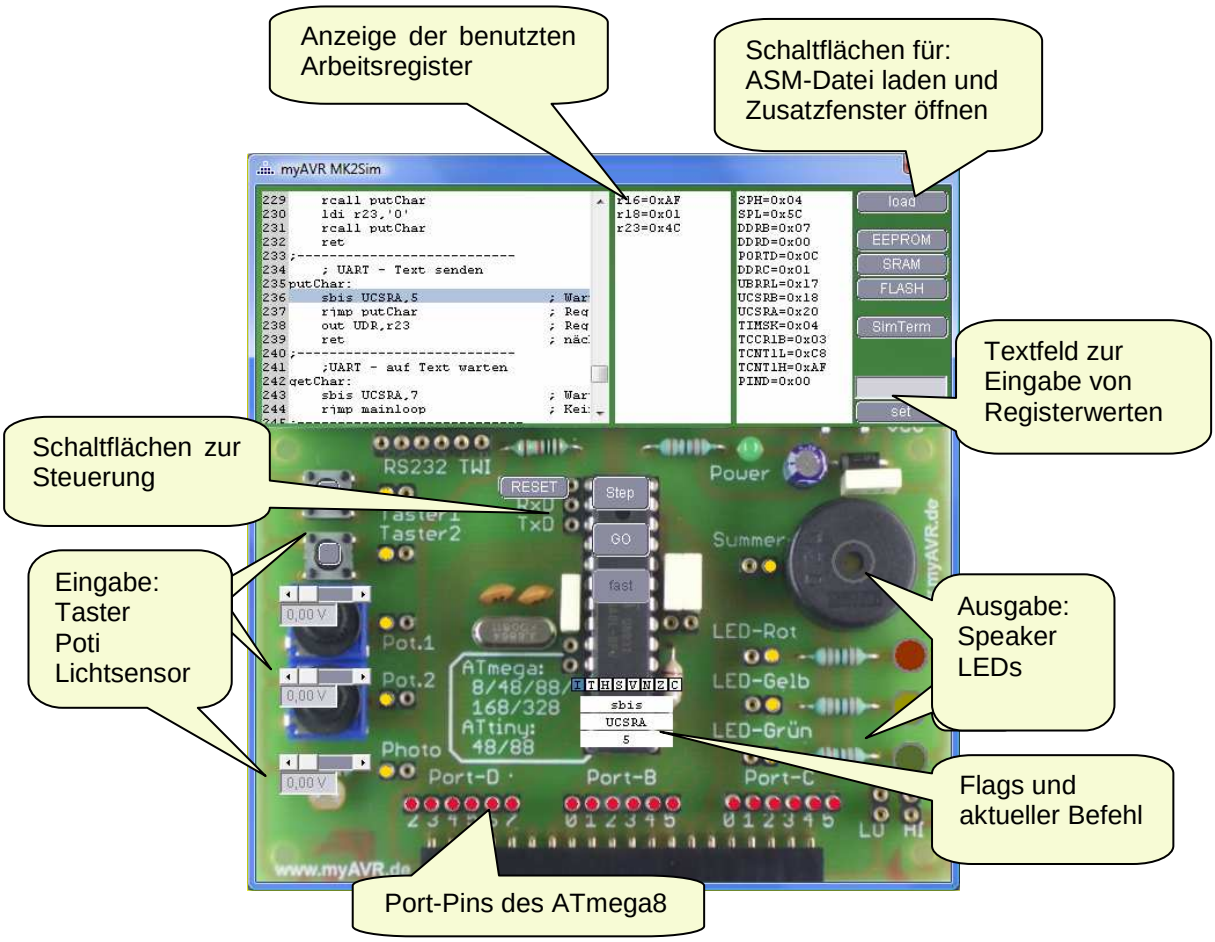
Zahlen senden

Zahlenwerte von 0 bis 255 (1 Byte) können einzeln oder als Zahlenfolge gesendet werden. Zwischen den Werten ist ein Leerzeichen als Trennzeichen einzufügen. Zahlen, die größer sind als 255, werden auf den niederwertigen Teil gekürzt. Es ist möglich, Zahlen im Hexadezimalformat zu schreiben (Präfix 0x). Mit der zugehörigen Schaltfläche „Senden“ wird immer der gesamte Inhalt der Eingabezeile „Zahl“ gesendet.

10.3 Der myAVR MK2 Simulator

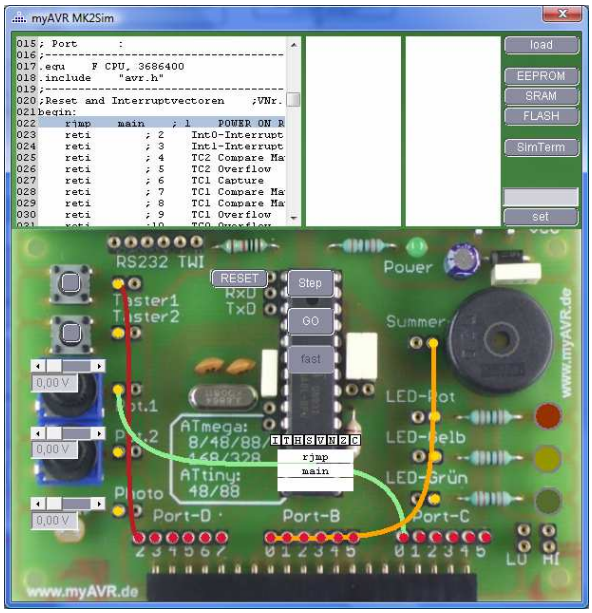
10.3.1 Einleitung

Mikrocontrolleranwendungen werden zur Laufzeit auf dem entsprechenden Zielsystem verarbeitet. Nicht jede Zielhardware verfügt über eine eigne Debughardware. Für das myAVR Board mit dem ATmega8 bietet der myAVR MK2 Simulator die Möglichkeit, Assemblerprogramme auf einer virtuellen Hardware zu simulieren. Der Simulator ist vor allem zu Lernzwecken und für Einsteiger konzipiert und verfügt nicht über alle Möglichkeiten der kompletten AVR-Controllerfamilie.



10.3.2 Simulator starten und konfigurieren

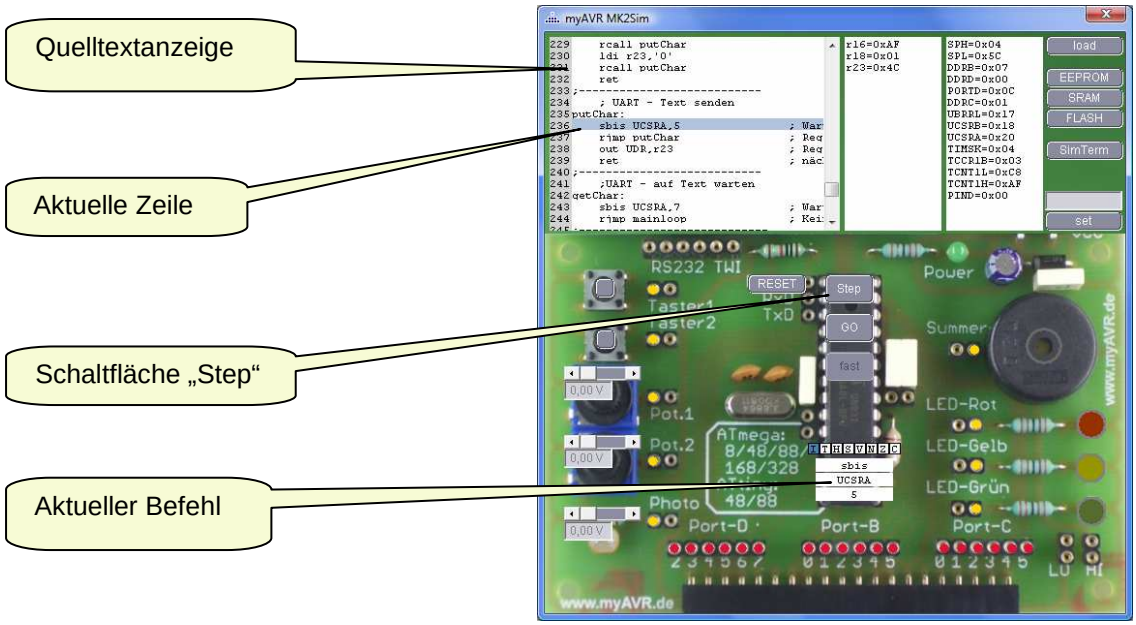
Der Simulator kann über das Menü *Werkzeuge* oder aus dem Diagrammfenster für die Programmierung von AVR Assemblerprogrammen gestartet werden. Wird der Simulator aus dem Menü *Werkzeuge* gestartet, muss die gewünschte Assemblerdatei vor der Simulation geladen werden. Als nächstes sind die Hardwareverbindungen zu Ein- und Ausgabegeräten herzustellen. Die Anschlusspunkte sind dazu jeweils nacheinander mit der Maus anzuklicken. Mit der rechten Maustaste kann Farbe und Form der virtuellen Patchkabel geändert werden.



10.3.3 Die Programmsimulation durchführen

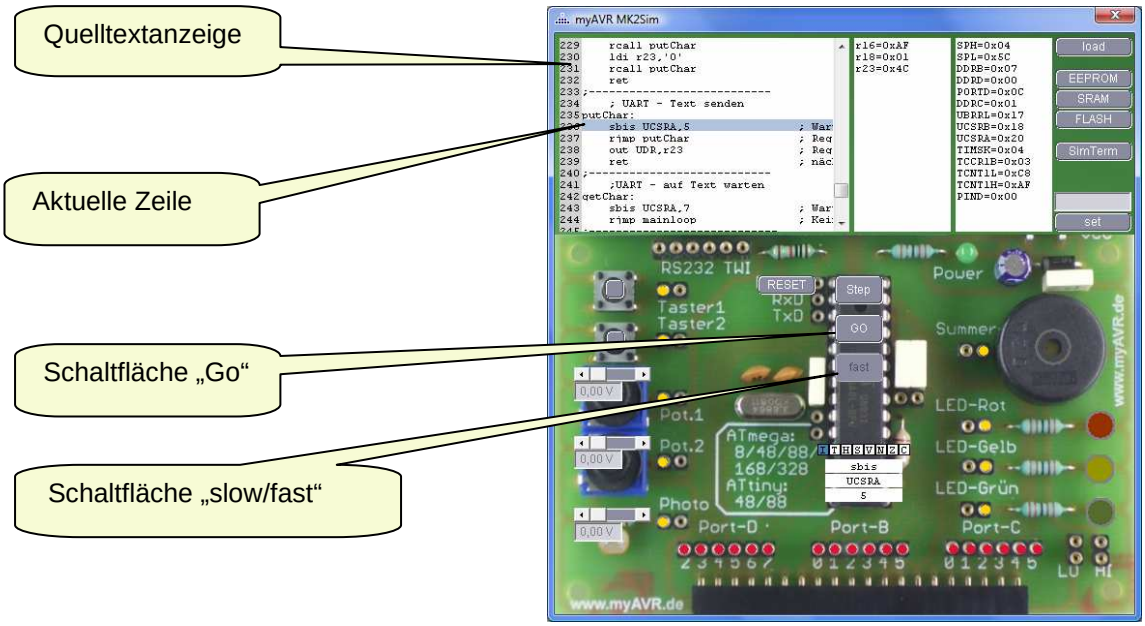
Einzelschritte

Die einfachste Form der Simulation ist der Einzelschrittbetrieb. Dabei wird Anweisung für Anweisung ausgeführt. Die Schaltfläche „Step“ simuliert dabei das Taktsignal des Controllers. Die Abarbeitung eines Befehls erfordert eine dem realen Befehl entsprechende Anzahl von Steps (Takten, vgl. Datenblatt des ATmega8). Die meisten Befehle des AVR RISC-Core benötigen einen Takt zur Abarbeitung.



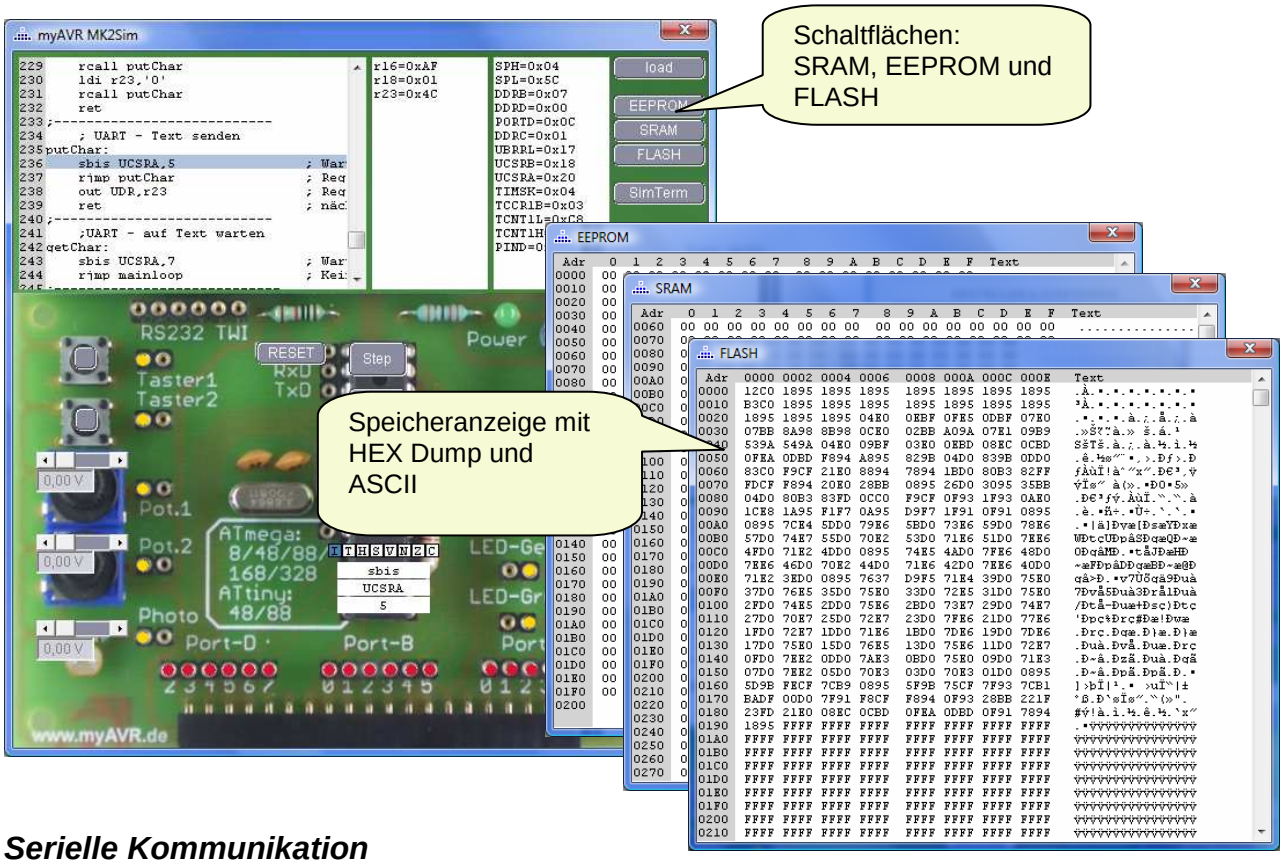
Langsame und schnelle Simulation

Die Schaltfläche „Go“ simuliert einen kontinuierlichen Takt. Es ist nicht mehr nötig, die Schaltfläche „Step“ zu betätigen. Mit der Schaltfläche „slow/fast“ kann die Simulationsgeschwindigkeit umgeschaltet werden. Die Quellcodeanzeige und die aktuelle Programmzeile werden während der schnellen und langsamen Simulation nachgeführt, jedoch kann das Auge nur der langsamen Simulation folgen.



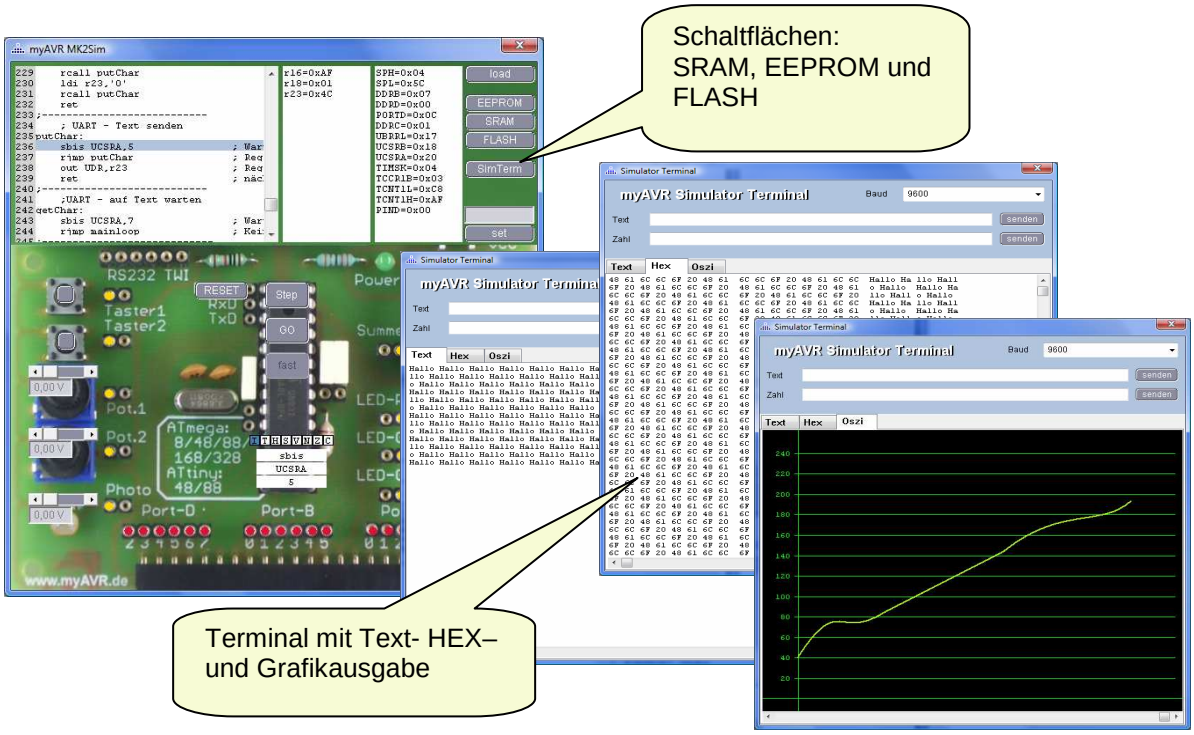
Speicherinhalte anzeigen

Es ist möglich im Schrittbetrieb oder während der automatischen Simulation die Speicherinhalte von FLASH, EEPROM und SRAM zu betrachten und Änderungen zu verfolgen. Die Speicheranzeigen öffnen Sie mit den entsprechenden Schaltflächen.



Serielle Kommunikation

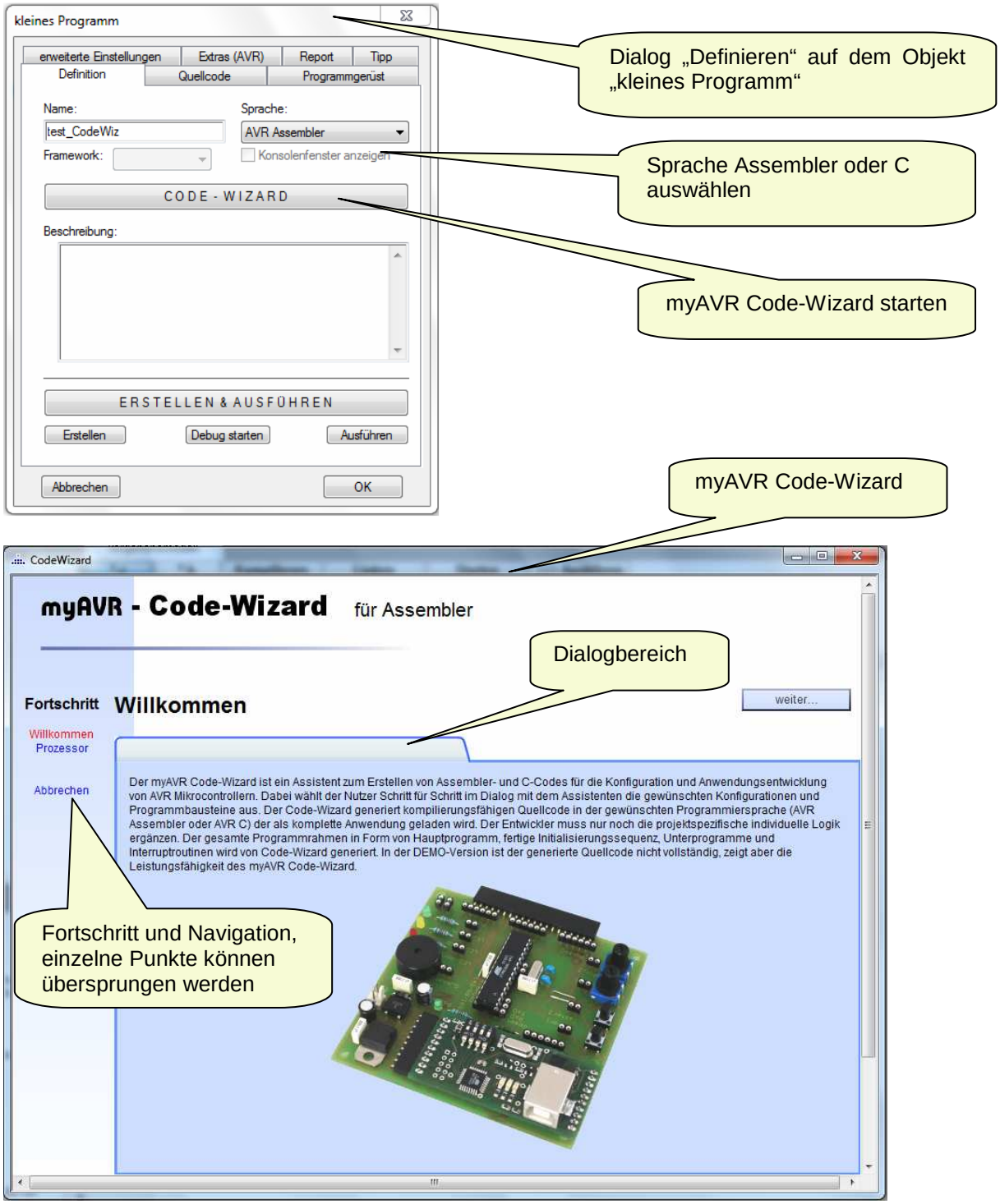
Ein wesentlicher Bestandteil zahlreicher Mikrocontrolleranwendungen ist die Kommunikation mit einem Personalcomputer. Der Simulator verfügt über ein Terminal zur seriellen Kommunikation über die UART des ATmega8 und ist dem ControlCenter nach empfunden. Sie können zwischen Text-, Hex- und Grafikanzeige wählen.



10.4 Der myAVR Code-Wizard

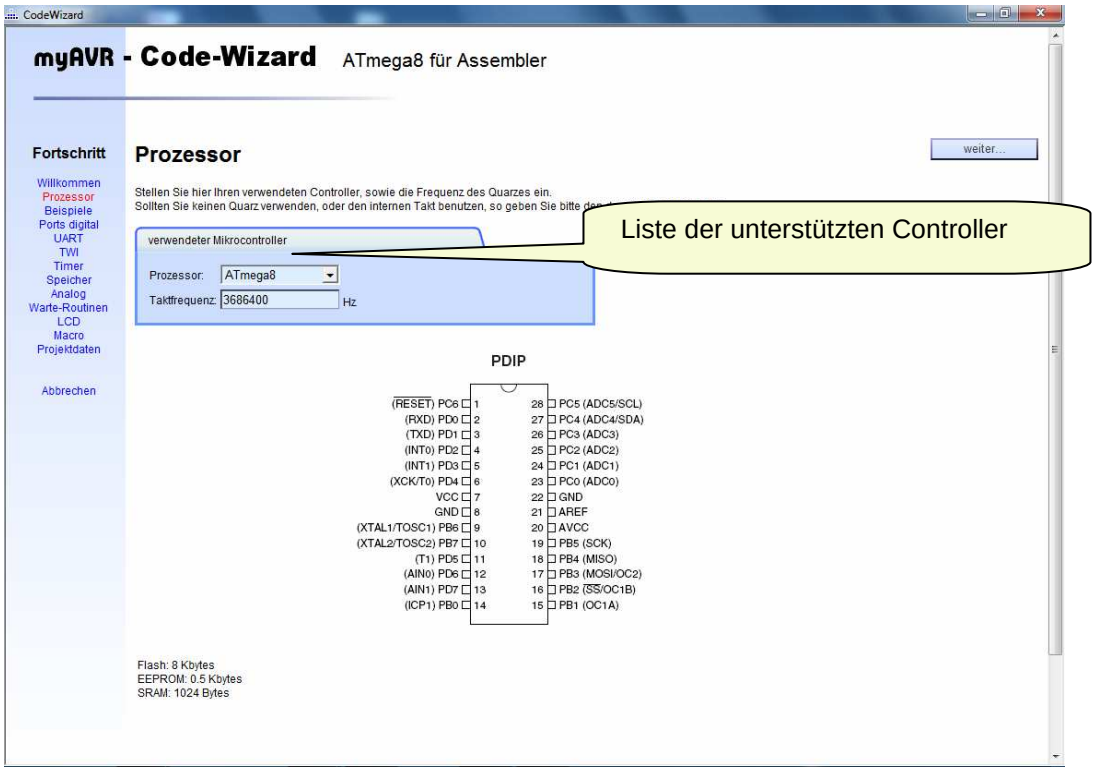
10.4.1 Einführung

Der myAVR Code-Wizard ist ein Assistent zum Erstellen von Assembler- und C-Codes für die Konfiguration und Anwendungsentwicklung von AVR Mikrocontrollern. Dabei wählt der Nutzer Schritt für Schritt im Dialog mit dem Assistenten die gewünschten Konfigurationen und Programmbausteine aus. Der Code-Wizard generiert kompilierungsfähigen Quellcode in der gewünschten Programmiersprache (AVR C oder AVR Assembler), der als komplette Anwendung geladen wird. Der Entwickler muss nur noch die projektspezifische individuelle Logik ergänzen. Der gesamte Programmrahmen in Form von Hauptprogramm, fertige Initialisierungssequenz, Unterprogramme und Interruptroutinen wird von Code-Wizard generiert.



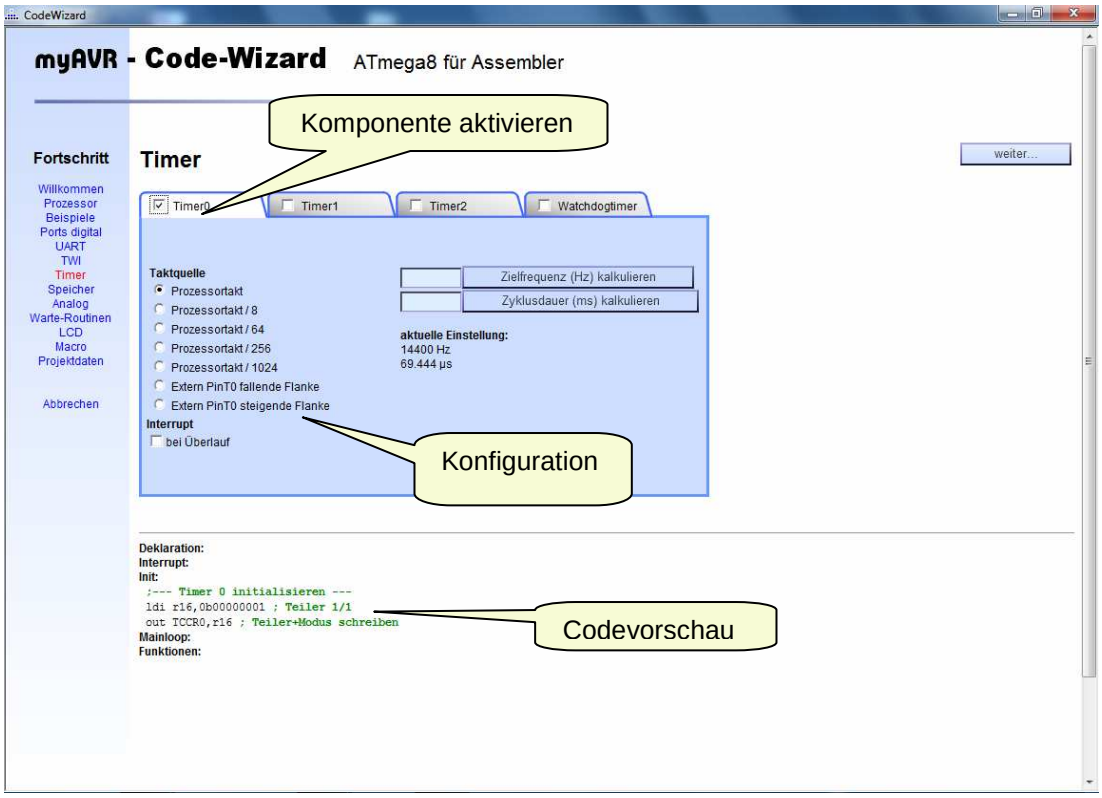
10.4.2 Grundeinstellungen

Die Zielsprache wurde vor dem Start des Code-Wizards ausgewählt. Für die Generierung von korrektem Quellcode ist es notwendig, den Controllertyp und die Taktgeschwindigkeit festzulegen.



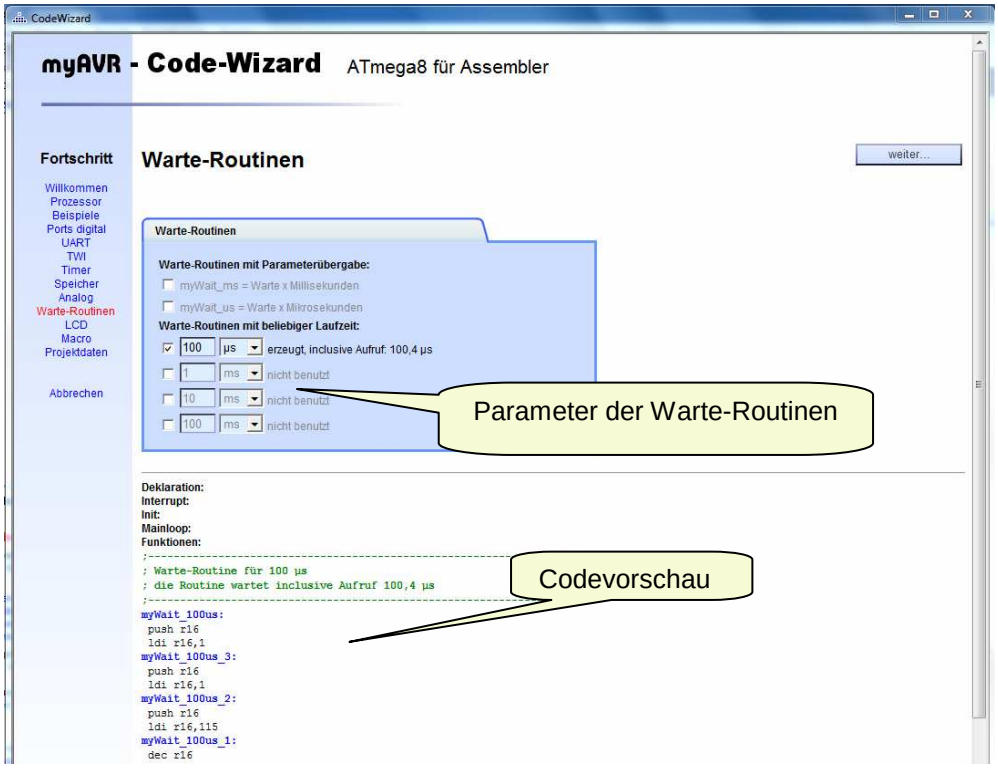
10.4.3 Geräteeinstellungen

Über den Dialogbereich können Schritt für Schritt die Komponenten des gewählten Controllertyps konfiguriert werden.



10.4.4 Unterprogramme

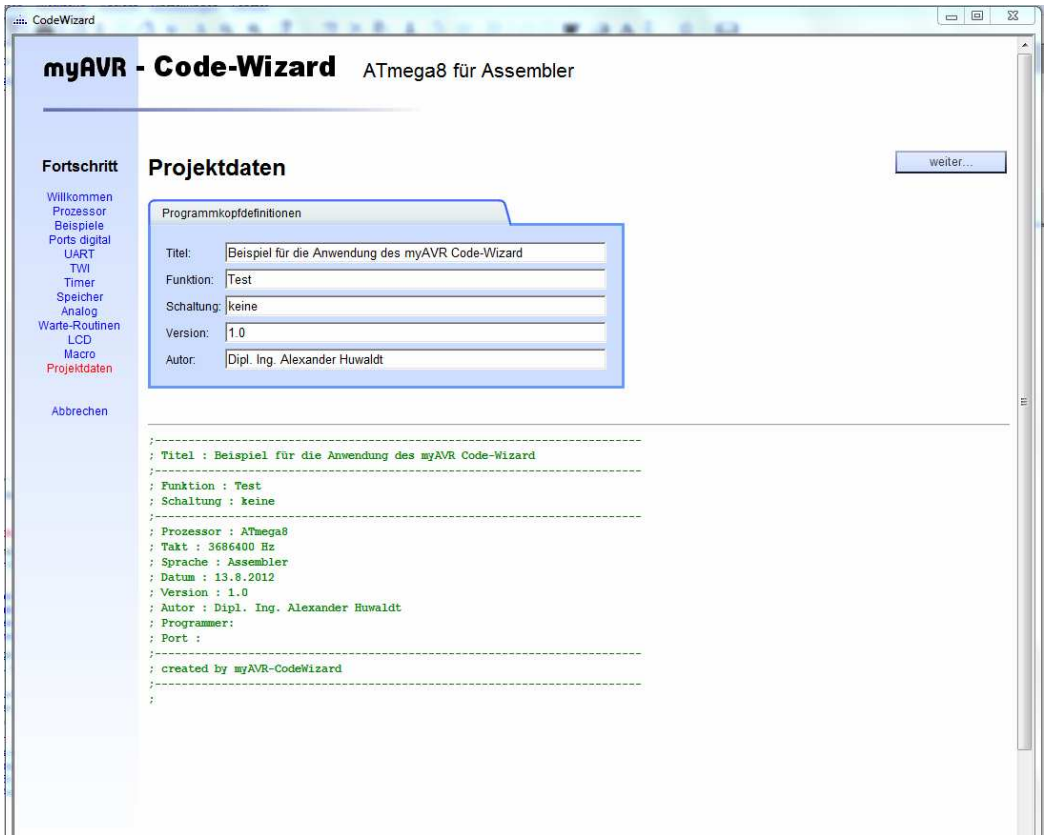
Neben der Möglichkeit die Hardwarekomponenten des gewünschten Controllers zu konfigurieren, bietet der Code-Wizard auch eine Reihe typischer Unterprogramme. So verfügt er über einen Warteroutinen-Rechner zum Generieren präziser Wartefunktionen.



.

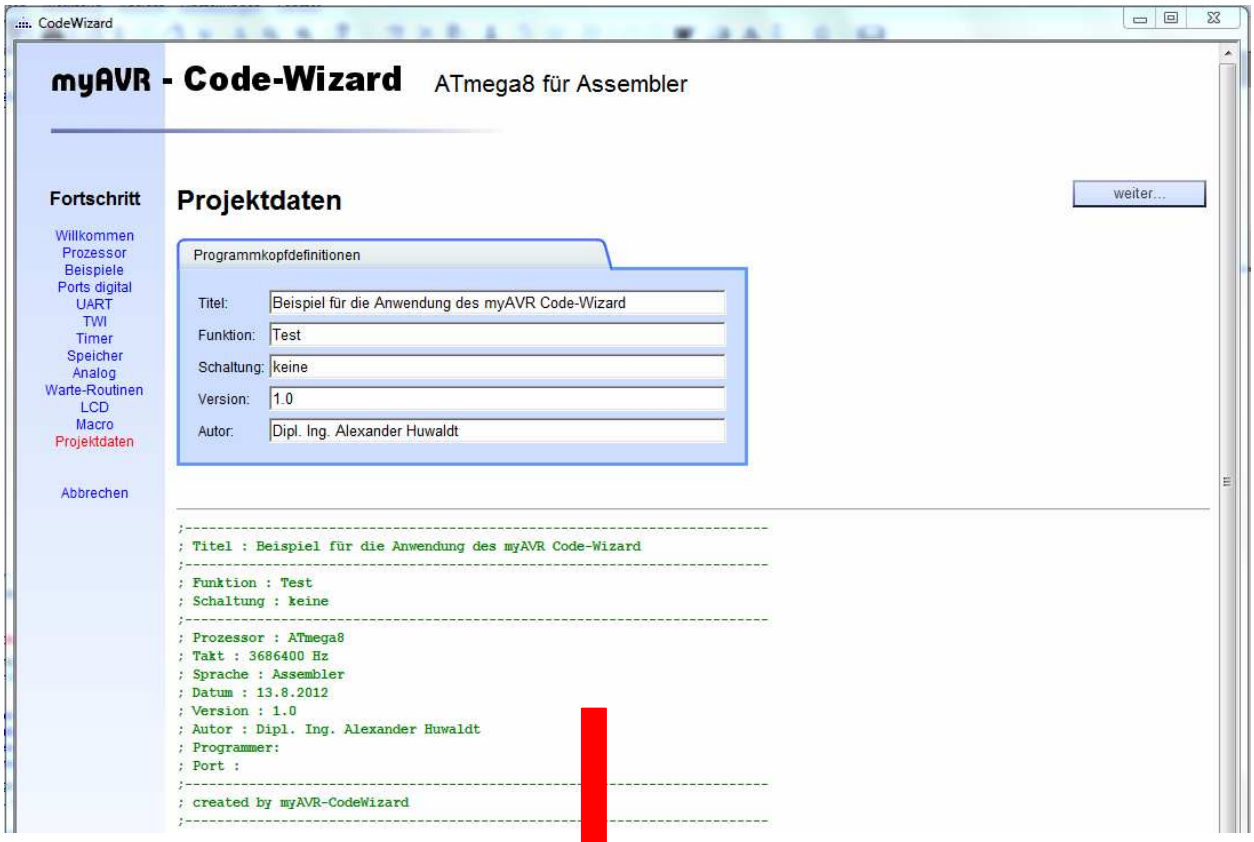
10.4.5 Projektdaten

Der letzte Punkt im Code-Wizard ist die Eingabe der Projektdaten. Aus diesen wird eine Programmkopfdokumentation generiert und dem kompletten Quellcode vorangestellt.



10.4.6 Codegenerierung

Der vollständige Quellcode wird zur Kontrolle dem Anwender angezeigt. Es können jetzt noch Änderungen vorgenommen werden, indem man die betreffenden Punkte im Navigationsbereich auswählt und die Parameter ändert. Mit Bestätigung des Quellcodes wird dieser als komplettes Programm eingefügt.



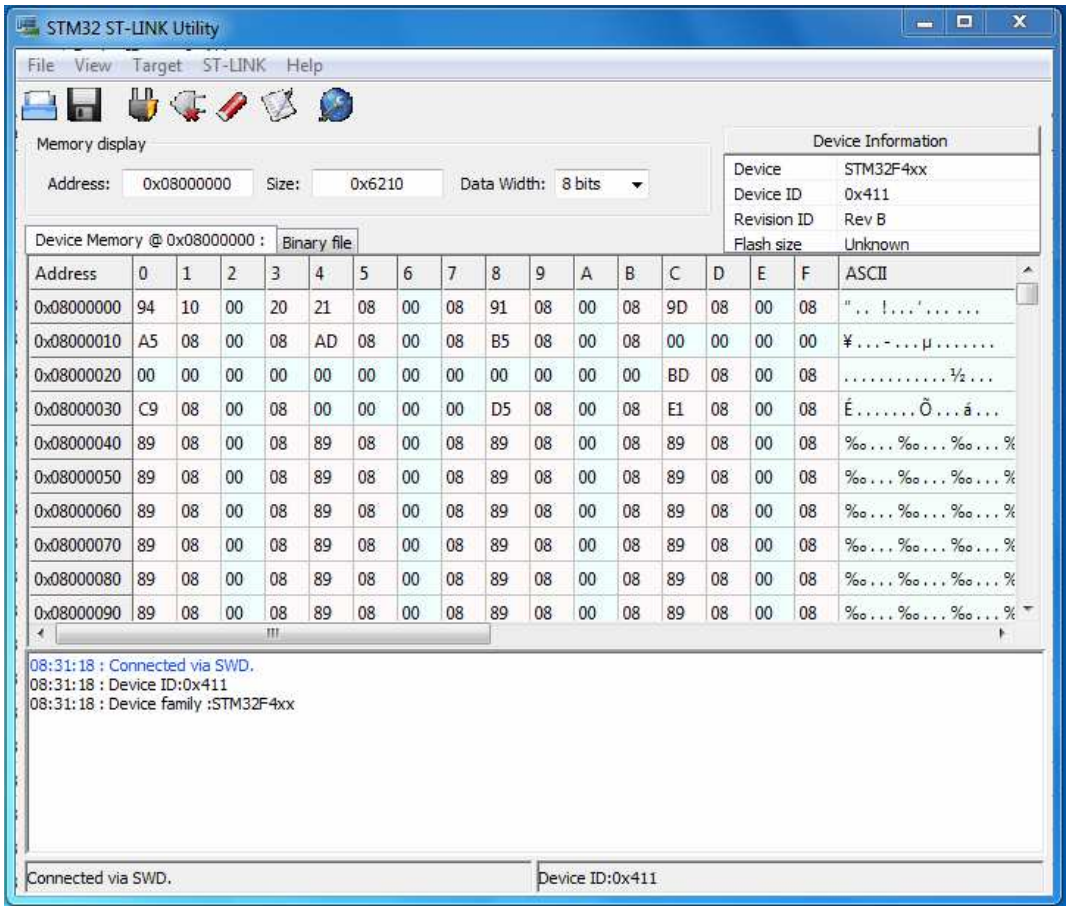

```
//-------------------------------------------------------------------------
// Titel : Test der seriellen Verbindung
//-------------------------------------------------------------------------
// Funktion : empfängt und sendet Daten per UART (9600,8,n,1)
// Schaltung : Nullmodemkabel anschließen
//-------------------------------------------------------------------------
// Prozessor : ATmega8
// Takt : 3686400 Hz
// Sprache : C
// Datum : 20.1.2011
// Version : 1.0
// Autor : Dipl. Ing. Päd. Huwaldt
// Programmer:
// Port :
//-------------------------------------------------------------------------
// created by myAVR-Code-Wizard
//-------------------------------------------------------------------------
#define   F_CPU   3686400
#include <avr\io.h>
#include <avr\interrupt.h>
//-------------------------------------------------------------------------
// Initialisierungen
//-------------------------------------------------------------------------
void init()
{
    // Ports initialisieren
    ...
```

10.5 Das STM32 ST-Link Utility

ST-Link ist ein Werkzeug zum Programmieren von ARM-Mikrocontrollern. Damit können Sie auch den Programmspeicher komfortabel programmieren.

Brennen

Der Inhalt der ausgewählten Dateien und des Speichers des ARM-Mikrocontrollers wird im mittleren Bereich des Fensters angezeigt.



Ausgabe

Die Ausgabe befindet sich im unteren Teil des ST-Link Fensters, die Aktionen werden im Ausgabefenster protokolliert. Es werden Informationen und Dauer der Aktion angezeigt. Zustand und Ergebnis werden durch Signalfarben und Ausgaben dargestellt.

Signalfarben
rote Schrift: keine Verbindung
blaue Schrift: Zustandsinformation
schwarze Schrift: verbundener Controller
grüne Schrift: erfolgreiches Brennen

Speicher anzeigen und ändern

Neben dem Geräte-Informationsbereich enthält das Hauptfenster 2 weitere Bereiche:

- Memory Display
- Memory Data

Memory Display:

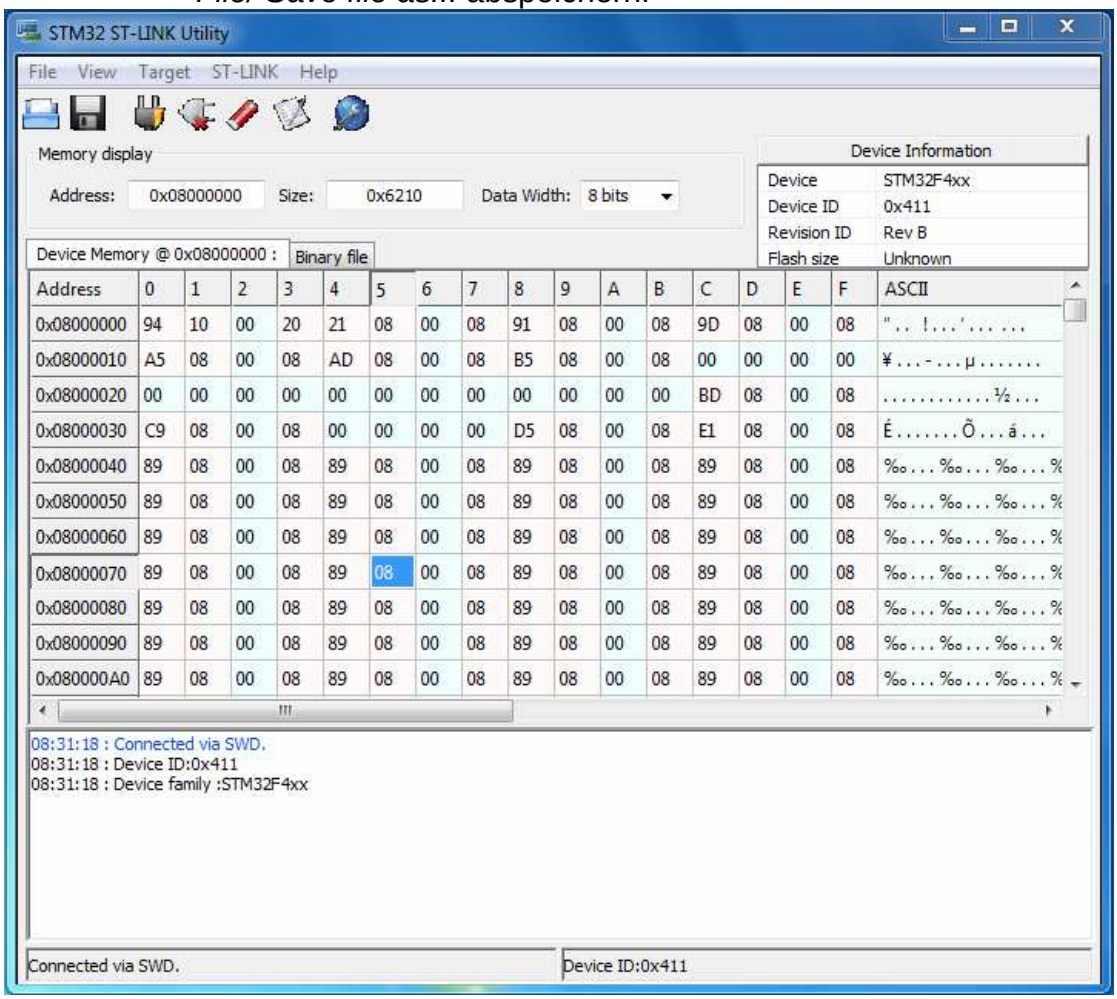
Dieser Bereich enthält drei Eingabefelder:

- *Address*: Startadresse des Speichers, von der Sie lesen wollen.
- *Size*: Anzahl der zu lesenden Daten in Byte (Hexadezimalformat).
- *Data Width*: Breite der angezeigten Daten (8-bit, 16-bit oder 32-bit).

Memory Data:

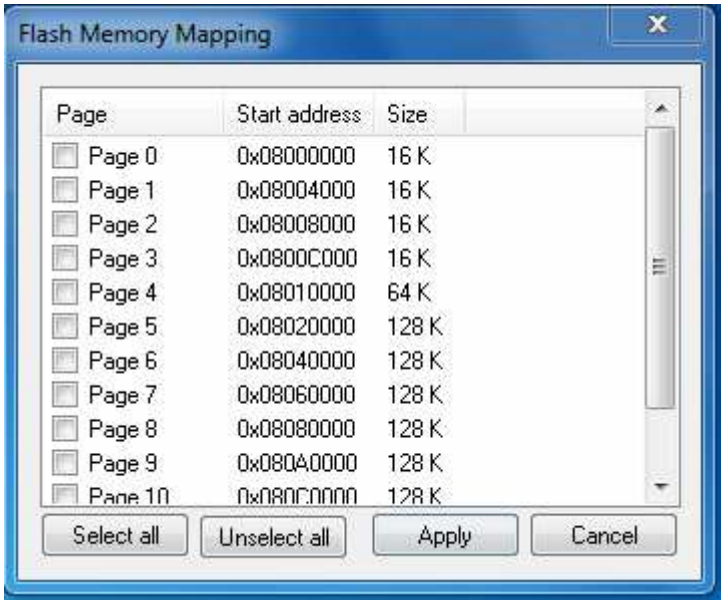
In diesem Bereich werden die Daten aus einer binären Datei oder aus dem Speicher eines angeschlossenen Gerätes ausgelesen. Sie können den Inhalt der Datei vor dem Brennen ändern.

- Um den Inhalt der binären Datei anzeigen zu lassen, klicken Sie auf *File/Open file ...*
- Um den Speicherinhalt eines verbundenen Geräts auszulesen und anzeigen zu lassen, geben Sie die Startadresse, Datengröße und die Datenbreite in die Felder ein und bestätigen dann mit „Enter“.
- Nach dem Lesen von Daten können Sie auch jeden Wert verändern, wenn Sie auf den Wert doppelklicken. Die betroffene Zelle wird markiert.
- Sie können auch den Speicherinhalt in eine binäre Datei über das Menü *File/ Save file as...* abspeichern.

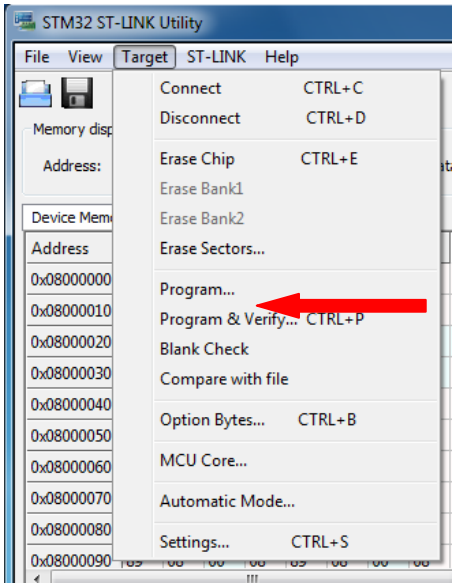


***Flash-Speicher löschen**:*
Es gibt zwei Vorgehensweisen den Flash-Speicher zu löschen:

1. Gesamten Flash löschen:
   - Löschen Sie alle Flash-Speichersektoren des angeschlossenen Gerätes. Das wird über das Menü *Target/Erase Chip* ausgeführt.
2. Sektorweise den Flash löschen:
   - Zur Auswahl der Sektoren klicken Sie im Menü *Target* auf *Erase Sectors…*. Das "Flash Memory Mapping" Fenster zeigt die ausgewählten Sektoren des Flashs an.
     - „Select all" markiert alle Sektoren
     - „Deselect all" macht die Markierung rückgängig
     - „Cancel" bricht den Löschvorgang ab
     - „Apply" löscht alle markierten Sektoren

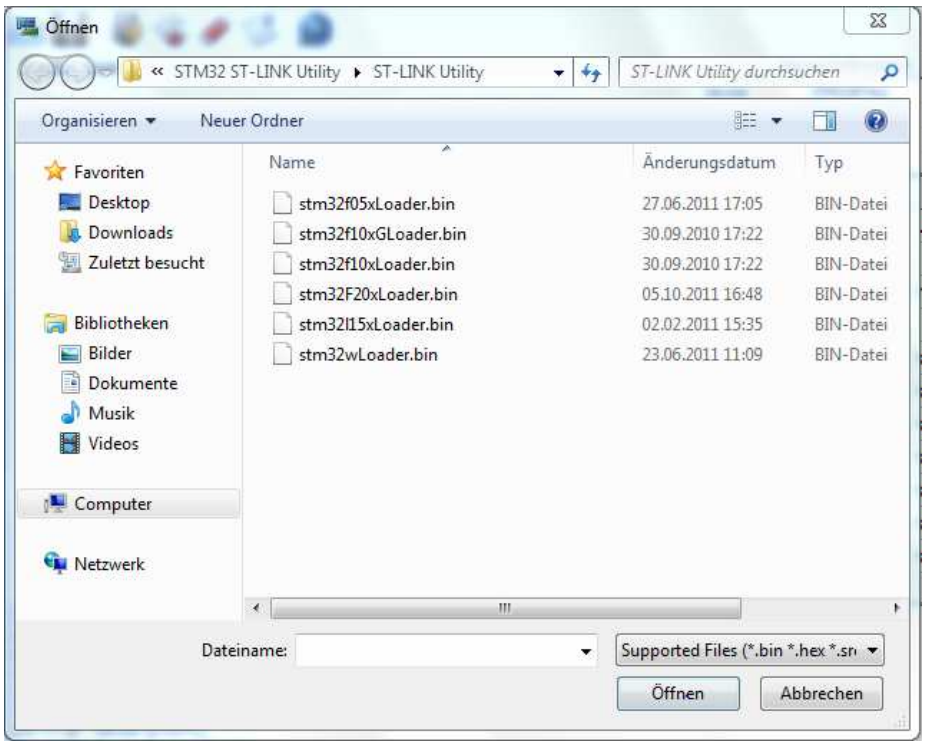


***Programm brennen**:*
Das STM32 ST-Link kann *.bin, *.hex, oder *.srec Dateien in den Flash oder RAM-Speicher brennen. Um das zu tun, führen Sie diese Schritte aus:



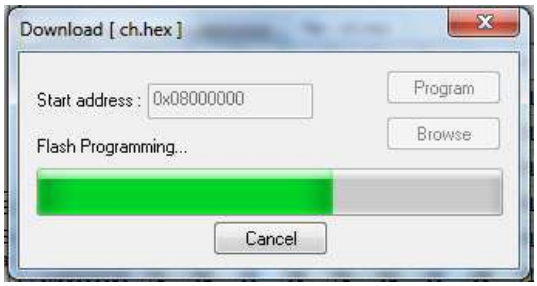
1. Klicken Sie auf *Target/Programm*... (oder *Target/Programm & verify…*, wenn nach dem Brennen die Daten überprüft werden sollen).

2. Es öffnet sich das „Open file" Dialogfenster, in diesem können Sie die Datei auswählen und mit der Schaltfläche „Open" die Datei öffnen.



3. In dem „Device programming“ Dialogfenster, geben Sie die Startadresse ein, ab welcher das Programm gebrannt werden soll, es kann eine Flash oder RAM-Adresse sein

4. Zum Schluss klicken Sie auf die Schaltfläche „Program“ und das Programm wird gebrannt. Wenn Sie *Target/Programm & verify…* ausgewählt wurde, wird das Programm nach dem Brennen überprüft.

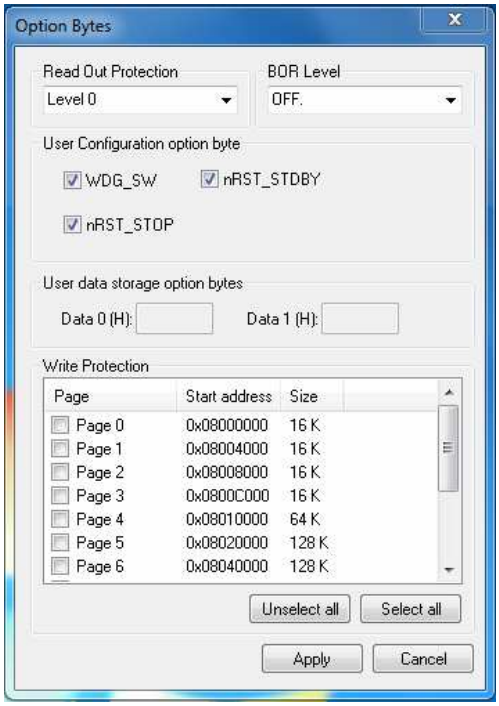


Hinweise:

1. *Die STM32F2 und STM32F4 Reihe haben unterschiedliche Programmierverfahren, die von der Spannungsversorgung der MCU abhängt. ST-Link verwendet die standardmäßige MCU-Spannungsversorgung. Unter dem Menü Target/Settings… kann die Spannungsversorgung geändert werden.*
2. *Wenn das Gerät schreibgeschützt gelesen wird, dann wird der Schreibschutz deaktiviert. Wenn einige Flash-Sektoren schreibgeschützt sind, dann wird der Schreibschutz während der Programmierung deaktiviert und nach der Programmierung wieder aktiviert.*

Option-Bytes-Konfiguration

Das STM32 ST-Link kann alle Option-Bytes über die „Option Bytes“ Dialogbox ändern, das über *Target/Option Bytes…* geöffnet werden kann.



Der „Option Bytes“ Dialogbox enthält die folgenden Einstellungsmöglichkeiten:

1. *„Read Out Protection“:* Verändert den Leseschutz des Flash-Speichers. Für STM32F2, STM32F4 und STM32L1 Geräte sind folgende Schutzniveaus verfügbar:
   - Level 0: kein Leseschutz
   - Level 1: Flash Leseschutz aktiviert
   - Level 2: Flash Leseschutz und alle Debug-Eigenschaften deaktiviert

   Für die anderen Geräte kann der Leseschutz nur aktiviert oder deaktiviert werden.

2. *„BOR Level“* (Brown Out Reset-Pegel): Diese Liste enthält unterschiedliche Spannungseinstellungen, beim Unterschreiten der Spannungsgrenze wird das Gerät in den Reset-Modus gesetzt.
   Diese Auswahl ist nur für das angeschlossene STM32 L1 Gerät gültig.
   Für Geräte mit niedriger Spannungsversorgung kann aus 5 VBOR Einstellungen ausgewählt werden:
   - BOR LEVEL 1:
     Reset-Schwelle im Spannungsbereich von 1,69 bis 1,8 V
   - BOR LEVEL 2:
     Reset-Schwelle im Spannungsbereich von 1,94 bis 2,1 V
   - BOR LEVEL 3:
     Reset-Schwelle im Spannungsbereich von 2,3 bis 2,49 V
   - BOR LEVEL 4:
     Reset-Schwelle im Spannungsbereich von 2,54 bis 2,74 V
   - BOR LEVEL 5:
     Reset-Schwelle im Spannungsbereich von 2,77 bis 3,0 V

Für STM32F2 und STM32F4 Geräte können 4 programmierbare VBOR Einstellungen ausgewählt werden:

- BOR LEVEL 3:
  Spannungsversorgung im Bereich von 2,70 bis 3,60 V
- BOR LEVEL 2:
  Spannungsversorgung im Bereich von 2,40 bis 2,70 V
- BOR LEVEL 1:
  Spannungsversorgung im Bereich von 2,10 bis 2,40 V
- BOR off:
  Spannungsversorgung im Bereich von 1,62 bis 2,10 V

3. „*User Configuration option byte“:*
   - WDG_SW: Wenn aktiviert, muss der Watchdog durch die Software aktiviert werden, sonst wird er automatisch beim Einschalten aktiviert.
   - nRST_STOP: Wenn nicht aktiviert, wird der Reset im Standby-Modus generiert. Wenn aktiviert, wird kein Reset generiert beim Eintreten in den Standby-Modus.
   - nRST_STDBY: Wenn nicht aktiviert, wird ein Reset erzeugt beim Betreten des Stop-Modus (alle Uhren sind gestoppt). Wenn aktiviert, wird kein Reset generiert beim Betreten des Stop-Modus.
   - BFB2: Wenn nicht aktiviert und die Boot-Pins gesetzt sind, um das Gerät beim Start vom Anwender-Flash zu booten, startet das Gerät von der „Flash bank“ 2, sonst von der „Flash bank“ 1. Diese Option ist nur aktiviert, wenn das Gerät zwei Flash-Banks besitzt.

4. „*User data storage option bytes*“: Enthält 2 Byte für den “User storage”. Diese Bytes sind nicht verfügbar auf STM32F2, STM32F4 und STM32L1 Geräten

5. „*Write Protection*“: Die Flash-Sektoren sind geräteabhängig gruppiert. Hier kann der Schreibschutz für die einzelnen Sektoren aktiviert werden.

10.6 Das myAVR ProgTool

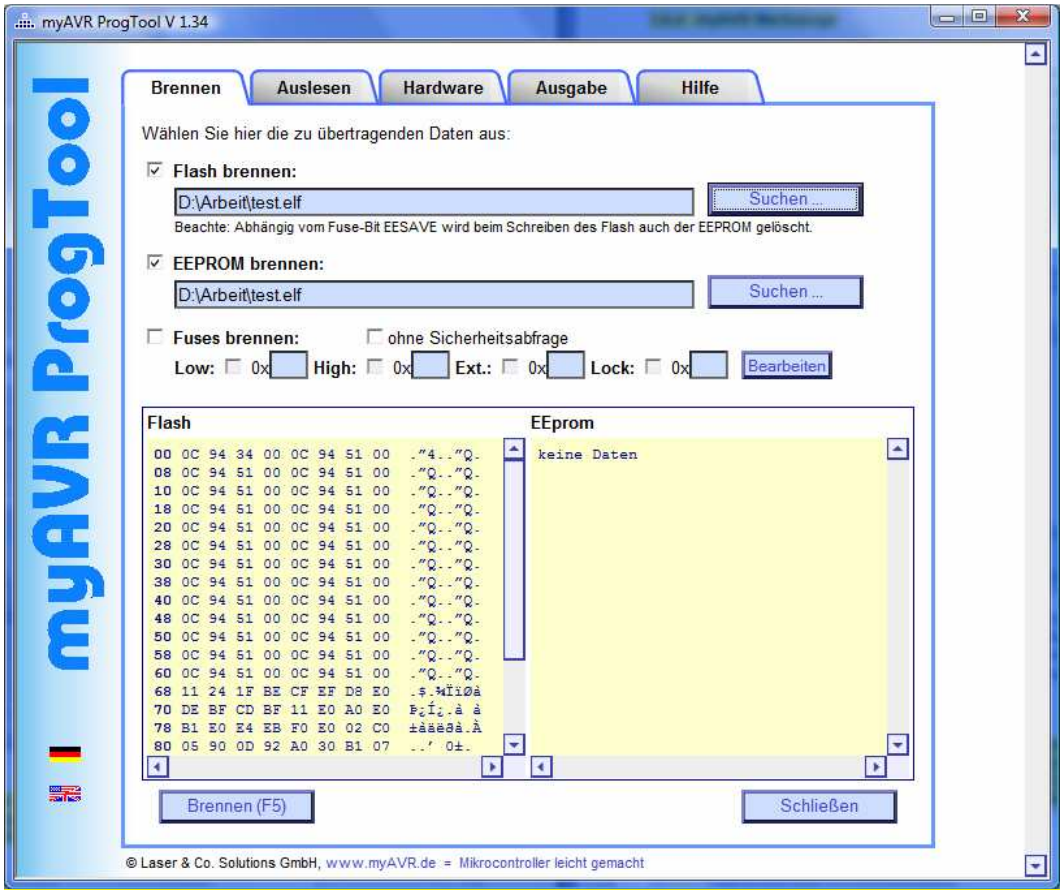
Hinweis:
Das myAVR ProgTool ist nur in SiSy Ausgaben mit dem Add-On AVR verfügbar.

10.6.1 Übersicht zum myAVR Progtool

Das myAVR ProgTool ist ein Werkzeug zum Programmieren von AVR-Mikrocontrollern. Sie können den Programmspeicher, den EEPROM und die Fuse-/Lock-Bits der unterstützten AVR-Mikrocontroller komfortabel programmieren.

Brennen

Der Inhalt der ausgewählten Dateien wird im unteren Bereich des Fensters angezeigt. Die Inhalte der Eingabefelder bleiben nach dem Schließen erhalten und sind nach erneutem Öffnen des Tools wieder verfügbar.



Signalfarben:
rötlich gefülltes Datei-Eingabefeld: Datei existiert nicht
bläulich gefülltes Datei-Eingabefeld: Datei existiert
graue Schrift: Element inaktiv

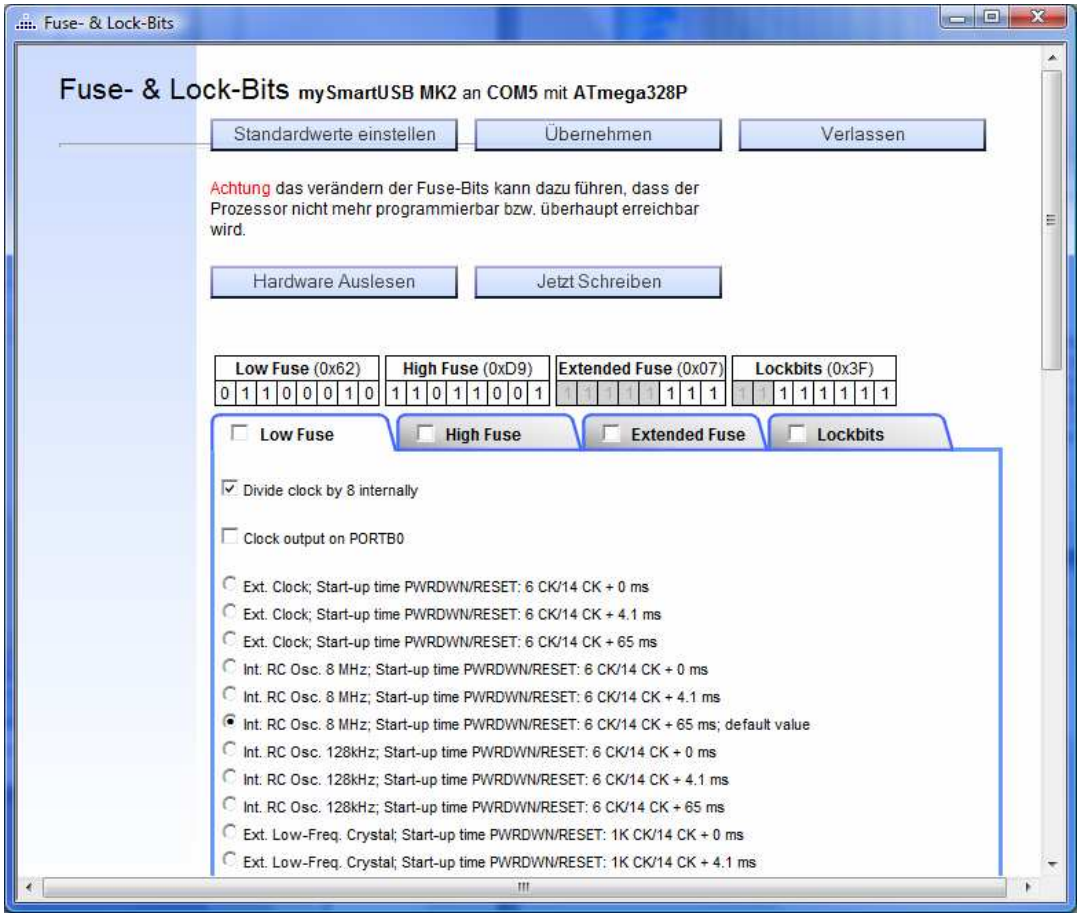
Flash brennen

Aktivieren Sie zunächst die Auswahl "Flash brennen". Wählen Sie anschließend per Schaltfläche "Suchen..." die gewünschte Datei (*.elf, *.hex, *.raw, *.bin) aus. Der Dateiname kann ebenso per Hand in das Eingabefeld eingegeben werden.
Bei Auswahl einer *.elf-Datei, wie sie zum Beispiel mit WinAVR erstellt wird, erfolgt eine automatische Konvertierung in das Intel-HEX-Format. Sind in der ELF-Datei die Sektionen ".eeprom", ".fuses" bzw. ".lock" enthalten, werden diese extrahiert und als Werte für EEPROM und die Fuses verwendet.

EEPROM brennen
Aktivieren Sie zunächst die Auswahl "EEPROM brennen". Wählen Sie anschließend per Schaltfläche "Suchen..." die gewünschte Datei (*.eep, *.elf, *hex, *.raw, *.bin, *.txt) aus. Der Dateiname kann ebenso per Hand in das Eingabefeld eingegeben werden.

Fuses brennen
Aktivieren Sie zunächst die Auswahl "Fuses brennen" (auf der Seite "Brennen") sowie die gewünschten Fuses (Low-, High- und Extended Fuse sowie Lock-Bits). Danach öffnen Sie die Fuse- und Lock-Bit Konfiguration über “Bearbeiten„.Tragen Sie die hexadezimalen Werte entsprechend der Datenblätter des zu programmierenden Mikrocontrollers ein. Sie können die Werte auch über die Schaltfläche "Bearbeiten" einstellen.

Fuse- und Lock-Bits
Sie ereichen die Fuse- und Lock-Bit Konfiguration über "Bearbeiten" auf der Seite "Brennen".
Sie können die Fuse- und Lock-Bits in einer Listenansicht nach Ihren Bedürfnissen konfigurieren. Dazu muss vorher im Reiter "Hardware" der korrekte Controller ausgewählt sein. Wollen Sie die aktuellen Werte auslesen oder verändern, müssen Sie auch den richtigen Programmer einstellen und anschließen.
Beachten Sie, dass bei fehlerhaften Einstellungen der Fuse- und Lock-Bits der Mikrocontroller unter Umständen nicht mehr programmierbar oder gar erreichbar sein kann.



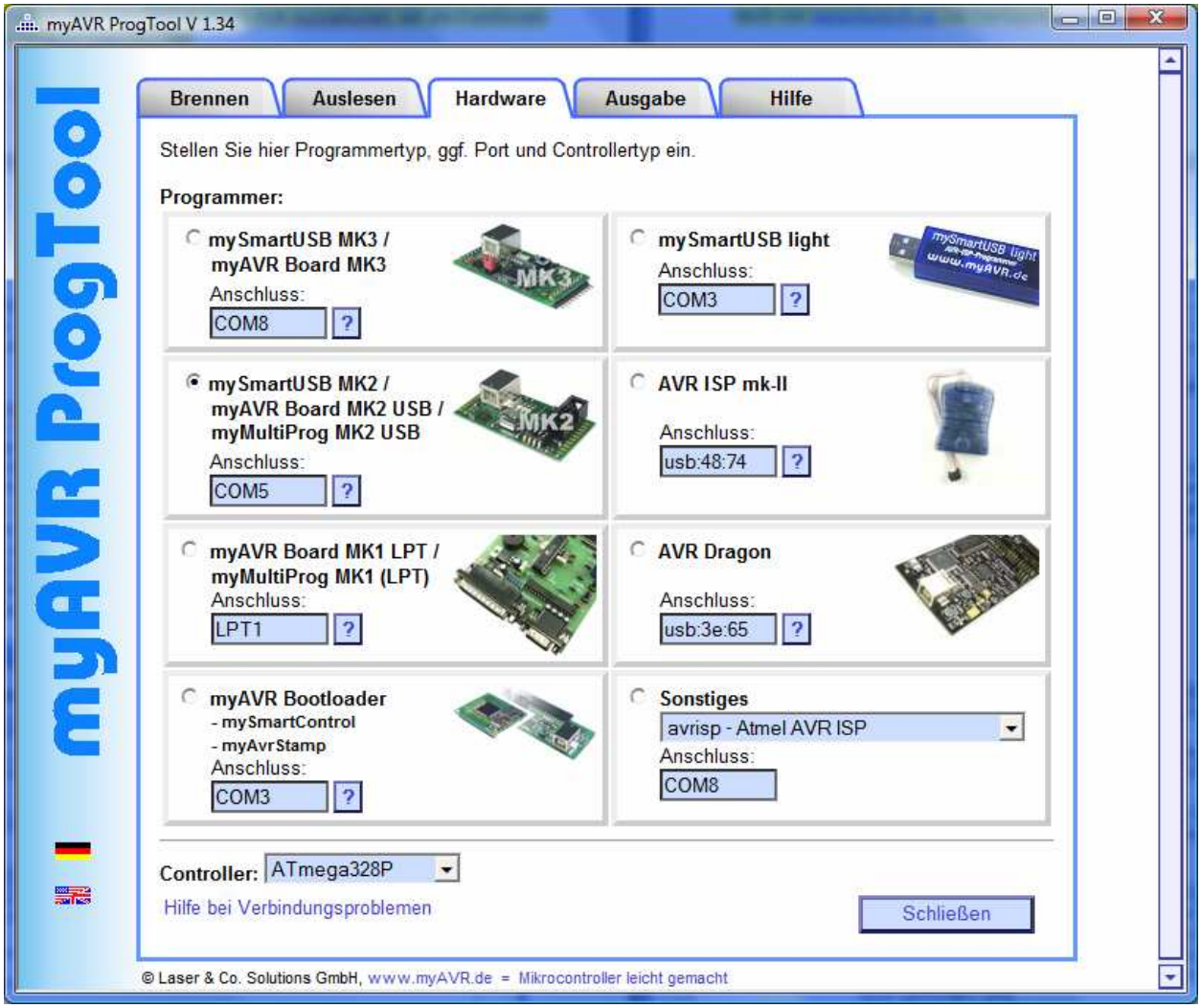
Weitere Informationen zu den Fuse- und Lock-Bit finden Sie im Kapitel 10.6.2

Auslesen eines Controllers

Zum Auslesen muss die richtige Hardware (Controller und Programmer) eingestellt sein. Die aktuellen Einstellungen sehen Sie im Kopfteil der Seite.
Lesen Sie die Daten des Flash- oder EEPROM-Speichers durch Betätigen der jeweiligen Schaltfläche aus. Dieser Vorgang kann je nach Speichergröße länger dauern. Es wird der komplette Speicher ausgelesen, haben Sie also etwas Geduld.
Nach erfolgreichem Auslesen werden die gelesenen Daten dargestellt. Nicht belegte Speicherbereiche am Ende der Daten werden vor der Darstellung entfernt.
Sie können die ausgelesenen Daten in eine Datei speichern. Die Speicherung erfolgt im Intel-HEX-Format.

Hardware einstellen

Wählen Sie vor dem Brennen immer die korrekte Programmierhardware und den zu programmierenden Controllertyp aus. Die Einstellungen werden beim Schließen gespeichert und sind nach erneutem Öffnen wieder verfügbar. Mit "Test" bzw. "?" kann der aktuelle Controller und der verwendete Port ermittelt werden.



Signalfarben:
gelb: Test läuft
grün: Test war erfolgreich
rot: Test ist fehlgeschlagen

Besonderheiten:

- mySmartUSB, myAVR Board, myMultiProg : → MK2, MK3, light /
  mySmartControl MK2: → 8K, 16K, 32K /
  mySmartControl MK3
  - USB-Treiber muss installiert sein (CP210X)
  - mySmartUSB muss angeschlossen sein
  - virtueller COM-Port (COMx) muss aktiv sein
  - Port-Einstellungen sind nicht notwendig, da der Programmer automatisch gefunden wird
  - Ziel-Hardware wird vom mySmartUSB mit Spannung versorgt
  - Über "?" kann der verwendete Controller ermittelt werden

- JTAG ICE mk-II:
  - LIB-USB-Treiber muss installiert sein
  - JTAG ICE mk-II muss angeschlossen sein
  - Spannungsversorgung der Ziel-Hardware muss eingeschaltet sein
  - korrekte USB-Port-Nummer muss im Feld "Anschluss:" eingetragen sein
  - Anschluss und Controller können per "?" ermittelt und überprüft werden

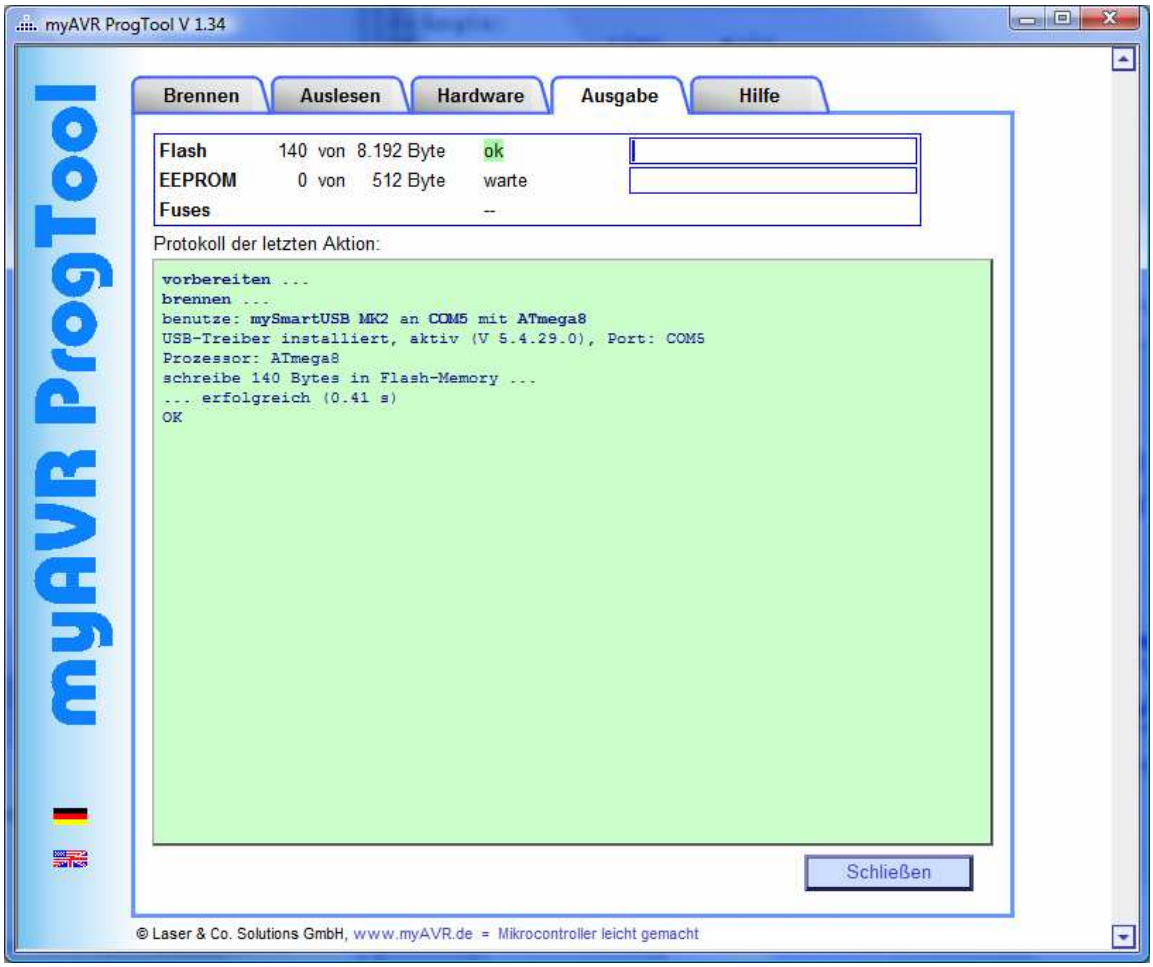
- STK 500:
  - Board muss an einem COM-Port angeschlossen sein
  - externe Spannungsversorgung muss angeschlossen sein
  - Board muss eingeschaltet sein (Power)
  - ISP-Pin muss verbunden sein
  - korrekter COM-Port muss im Feld "Anschluss:" eingetragen sein
  - Anschluss wird nicht automatisch ermittelt

- AVR ISP mk-II:
  - LIB-USB-Treiber muss installiert sein
  - AVR ISP mk-II muss angeschlossen sein
  - Spannungsversorgung der Ziel-Hardware muss eingeschaltet sein
  - korrekte USB-Port-Nummer muss im Feld "Anschluss:" eingetragen sein
  - Anschluss und Controller können per "?" ermittelt und überprüft werden

- AVR Dragon:
  - LIB-USB-Treiber muss installiert sein
  - AVR Dragon muss angeschlossen sein
  - Spannungsversorgung der Ziel-Hardware muss eingeschaltet sein
  - korrekte USB-Port-Nummer muss im Feld "Anschluss:" eingetragen sein
  - Anschluss und Controller können per "?" ermittelt und überprüft werden

Ausgabe

Die Aktionen werden im Ausgabefenster protokolliert. Neben anderen Informationen werden Anzahl der übertragenen Bytes und Dauer der Aktion angezeigt. Zustand und Ergebnis der Aktionen werden durch Signalfarben und Ausgaben angezeigt.



Signalfarben:
hellgrauer Hintergrund: keine Aktion, wartend
hellgelber Hintergrund: Aktion wird momentan ausgeführt
hellroter Hintergrund: Aktion wurde nicht erfolgreich ausgeführt, Fehler
hellgrüner Hintergrund: Aktion wurde erfolgreich beendet

Problembehandlung:

- Allgemein: Überprüfen Sie, ob alle Kabel richtig angeschlossen sind und überprüfen Sie, ob die Spannungsversorgung der Zielplattform ausreichend ist.
- Bei USB-Verbindungen trennen Sie diese kurz (ca. 20s) um eine Reinitialisierung des USB-Treibers durchzuführen.
- mySmartUSB funktioniert nicht:
  - USB-Treiber (CP210x) nicht installiert -> Treiber installieren
  - mySmartUSB im Datenmodus -> in Programmiermodus schalten (DIP)
- mySmartControl funktioniert nicht:
  - USB-Treiber (CP210x) nicht installiert -> Treiber installieren
  - kein Bootloader vorhanden -> Bootloader nachrüsten

10.6.2 Einstellungen Fuse- und Lock-Bits für AVR Produkte

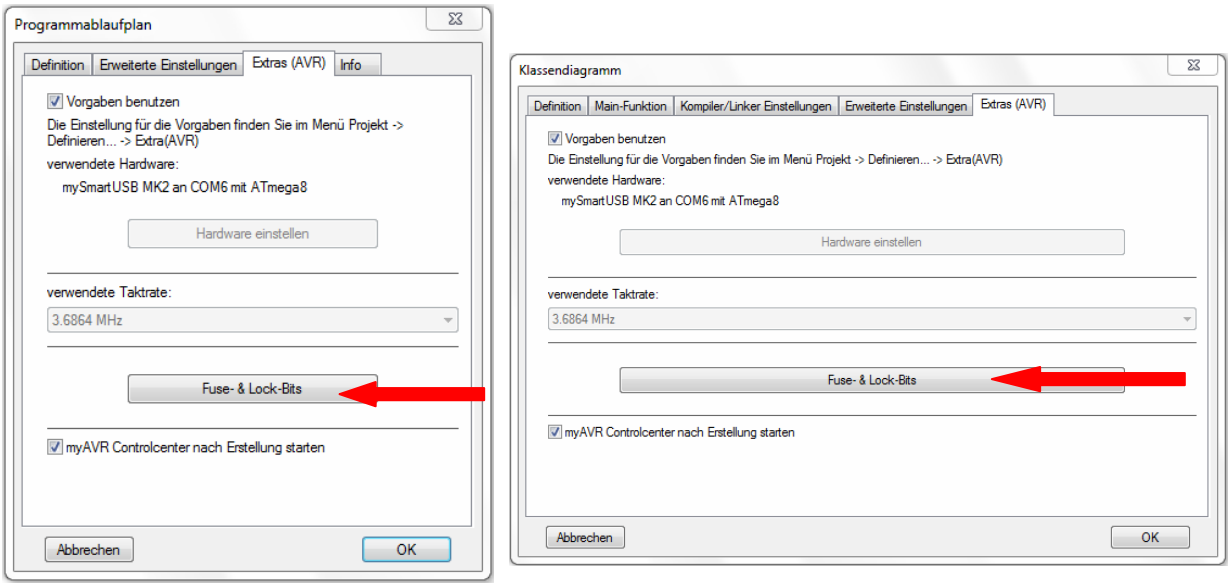
Einleitung

Fuse- und Lock-Bits (engl. fuse = Sicherung, engl. lock = Schloss) nennt man die Bits in bestimmten Registern des AVR zum Konfigurieren des Controllers. Die Fuse-Bits müssen über ein entsprechendes Interface (Software) eingestellt werden. Der normale Programmiermodus verändert die Fuse- und Lock-Bits nicht. Je nach Controllertyp sind unterschiedliche Fuse- und Lock-Bits verfügbar. Die verbindliche und exakte Beschreibung findet man im jeweiligen Datenblatt des Controllers. Das falsche Setzen der Fuse- und Lock-Bits zählt zu den häufigsten Problemen bei der Programmierung von AVR-Controllern, daher sollte hier mit Umsicht vorgegangen werden. Das Verändern der Fuse-Bits sollte man nicht als Anfänger vornehmen. Das geöffnete Datenblatt zur Überprüfung der Konfiguration ist das wichtigste Instrument um Fehler zu vermeiden.

Fuse- und Lock-Bits, Benutzeroberfläche in SiSy AVR

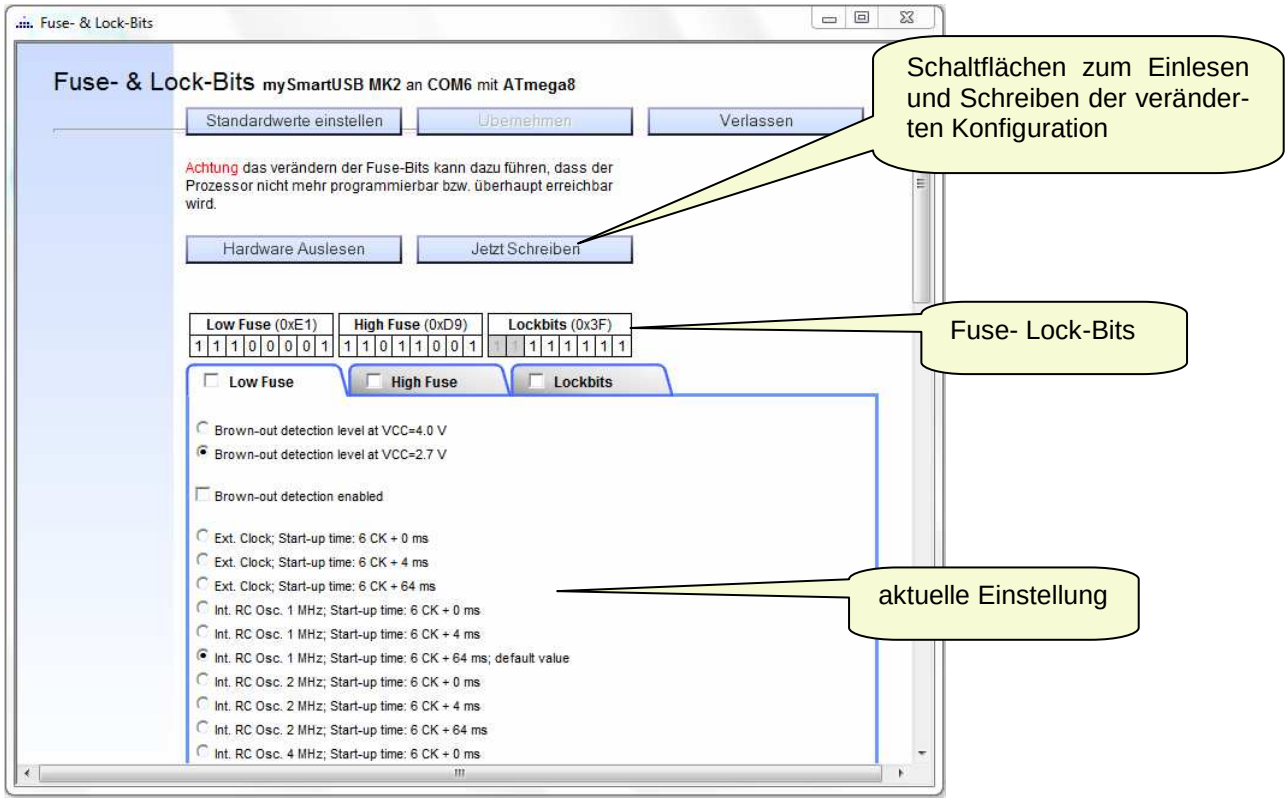
Sie erreichen die Benutzeroberfläche zum Auslesen, Verändern und Programmieren der Fuse- und Lock-Bits in SiSy wie folgt:

1. *Hauptmenü/Werkzeuge/myAVR ProgTool* -> Bearbeiten -> Fuse- und Lock-Bits
2. Auf Programm-Objekten wie „kleines Programm“, „Programm“, „PAP“ usw. : rechte Maustaste -> Kontextmenü -> Definieren … Dialogfeld Extras (AVR) -> Schaltfläche „Fuse- & Lock-Bits“.

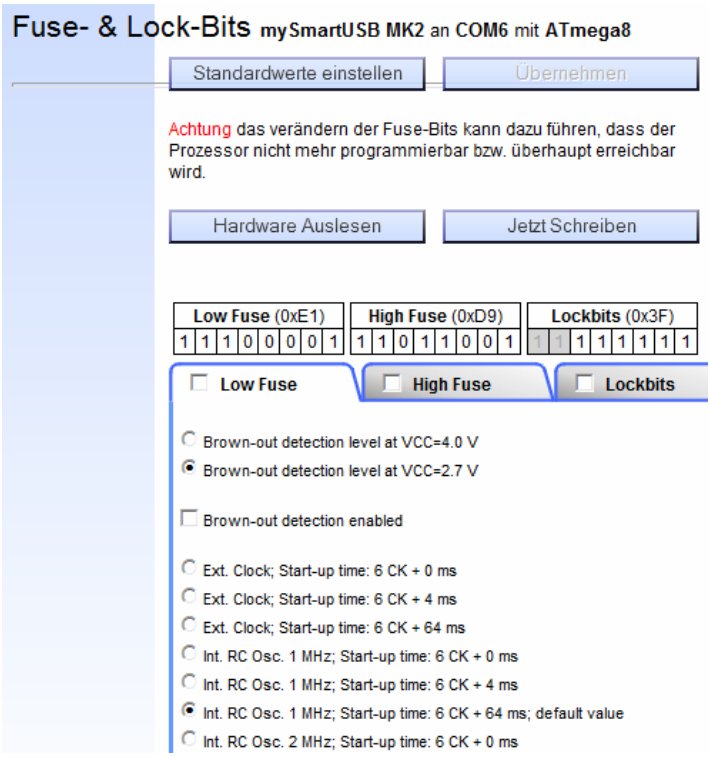


Beim Start der Fuse- und Lock-Bits-Benutzeroberfläche wird eine Verbindung zum Controller aufgebaut, der Controllertyp ermittelt, die Fuse- und Lock-Bit-Definitonen des Controllers geladen, die Fuse- und Lock-Bit Einstellungen des Controllers ausgelesen und angezeigt. Dieser Vorgang kann je nach Controllertyp und Verbindung einige Sekunden dauern. Die Verbindung zum Controller wird, solange diese Benutzeroberfläche offen ist, dauerhaft offen gehalten.

Die Benutzeroberfläche passt sich dem ermittelten Controller und den dazugehörigen Definitionsdaten automatisch an. Es werden immer nur die Optionen angezeigt, die zum ermittelten Controller gehören. Vergleichen Sie dazu immer das betreffende Datenblatt. Es ist nicht zulässig, den Programmer oder den Controller während der Sitzung zu entfernen oder zu wechseln. Dazu ist dieses Fenster zu schließen, danach der Controller oder Programmer zu wechseln und die Benutzeroberfläche erneut zu starten.



Fuse- und Lock-Bits des ATmega8



Fuse- und Lock-Bits des ATtiny15

Fuse- und Lock-Bits verändern

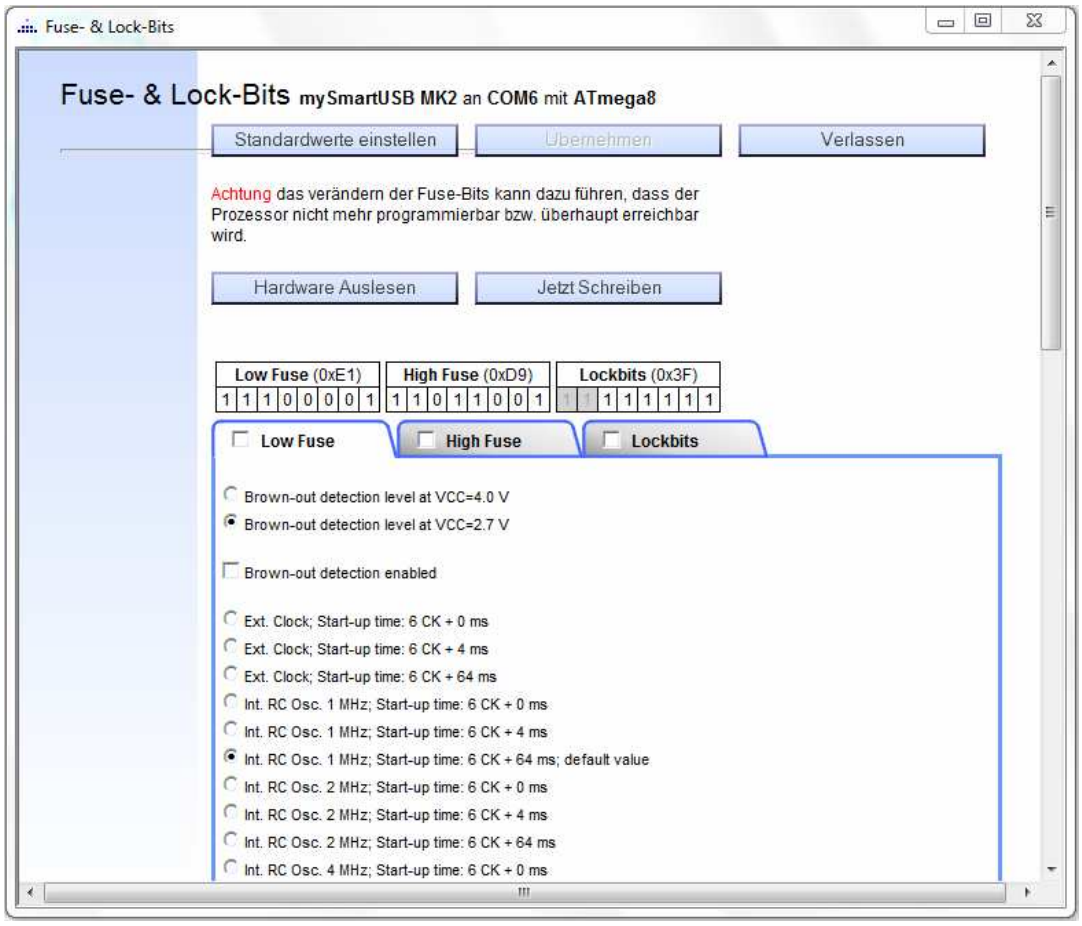
Zum Verändern der Fuse- und Lock-Bits sollte der entsprechende Abschnitt im Datenblatt des Controllers studiert werden. Über die Dialogfelder Low-, High- und Extended-Fuse- sowie Lock-Bits können die einzelnen Optionen bequem ausgewählt werden. Die Änderungen werden im Anzeigebereich für die Fuse- und Lock-Bits visualisiert. Erst mit dem Betätigen der Schaltfläche „Jetzt Schreiben“ werden die neuen Einstellungen an den Controller übertragen.

Beachte: Falsche Einstellungen der Fuse- oder Lock-Bits können dazu führen, dass der Controller in der aktuellen Hardware nicht mehr angesprochen werden kann. Häufige Fehleinstellungen durch unerfahrene Entwickler sind:

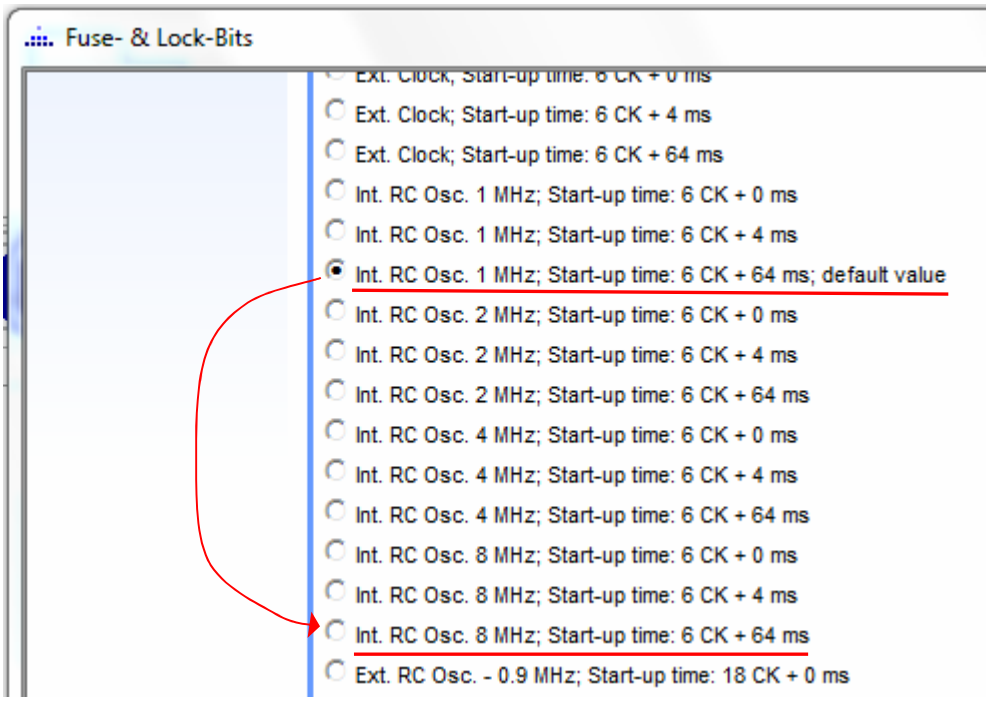
- **Reset disable ->** führt dazu, dass kein ISP mehr möglich ist
- **ISP enable ausgeschaltet ->** führt dazu, dass kein ISP mehr möglich ist
- **Taktquelle umgeschaltet ->** führt u.U. dazu, dass der Controller nicht arbeitet

Im Folgenden wird die Vorgehensweise beschrieben, wie die Taktquelle eines ATmega8 vom internen 1 MHz Oszillator auf intern 8 MHz umgeschaltet wird.

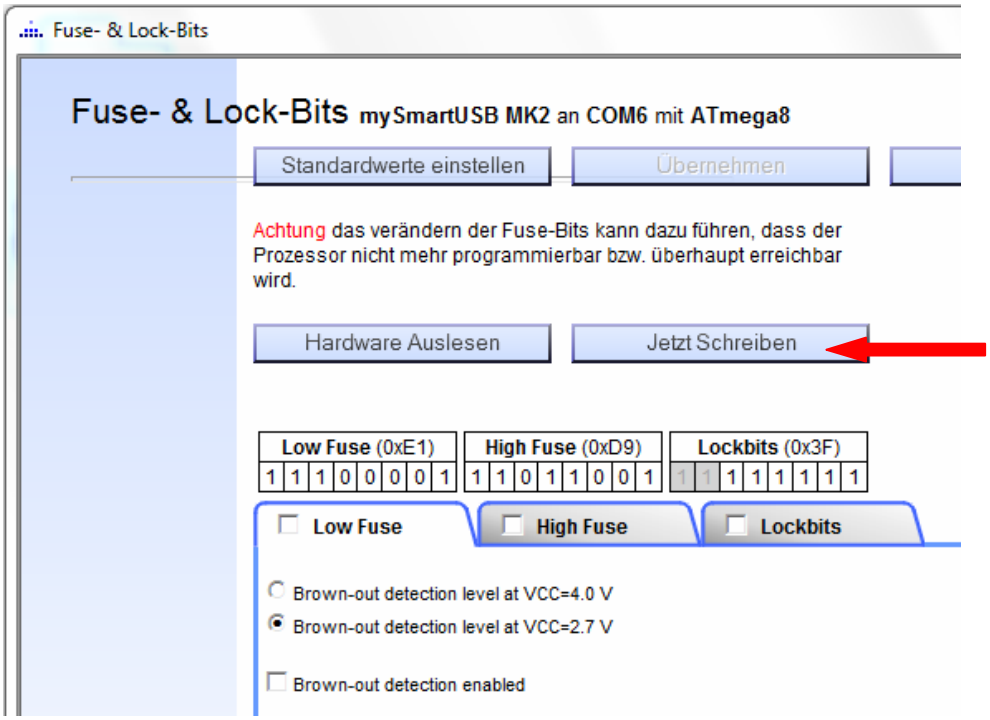
1. Das Board und den Programmer anschießen.
2. Benutzeroberfläche für das Verändern der Fuse- & Lock-Bits starten.



3. In der Liste der Optionen nach unten scrollen und **`Int. RC. Osc. 8MHz`** auswählen. Vergleichen Sie dazu die Beschreibung im Datenblatt des ATmega8.



4. Die Optionsseite wieder zurückscrollen und die veränderten Fuse-Bits überprüfen.



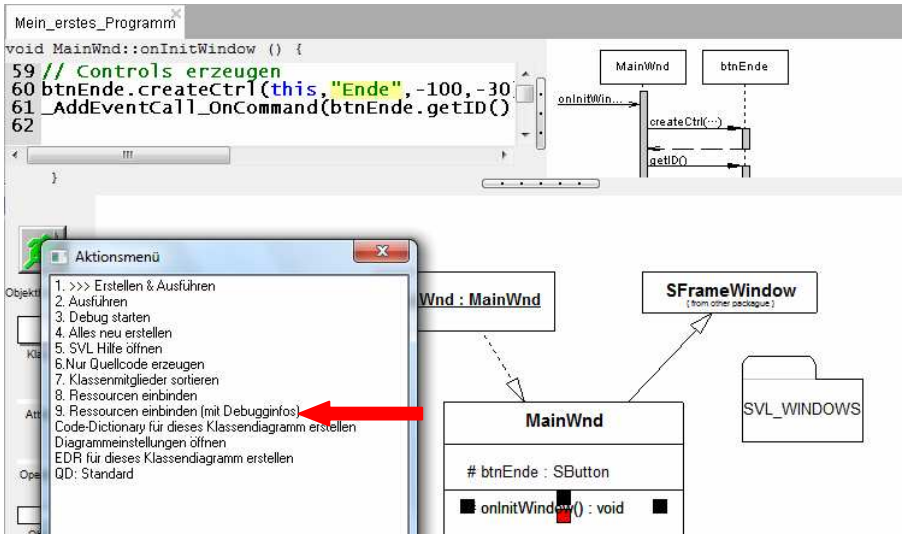
5. Die Schaltfläche „Jetzt Schreiben“ wählen und die Sicherheitsabfrage bestätigen. Danach sollten die Einstellungen überprüft werden durch das Betätigen der Schaltfläche „Hardware Auslesen“.

Über die Schaltfläche „Verlassen“ kann die Sitzung zum Verändern der Fuse- und Lock-Bits beendet werden. Die Verbindung zum Controller und Programmer wird dann geschlossen.

10.7 Der Debugger

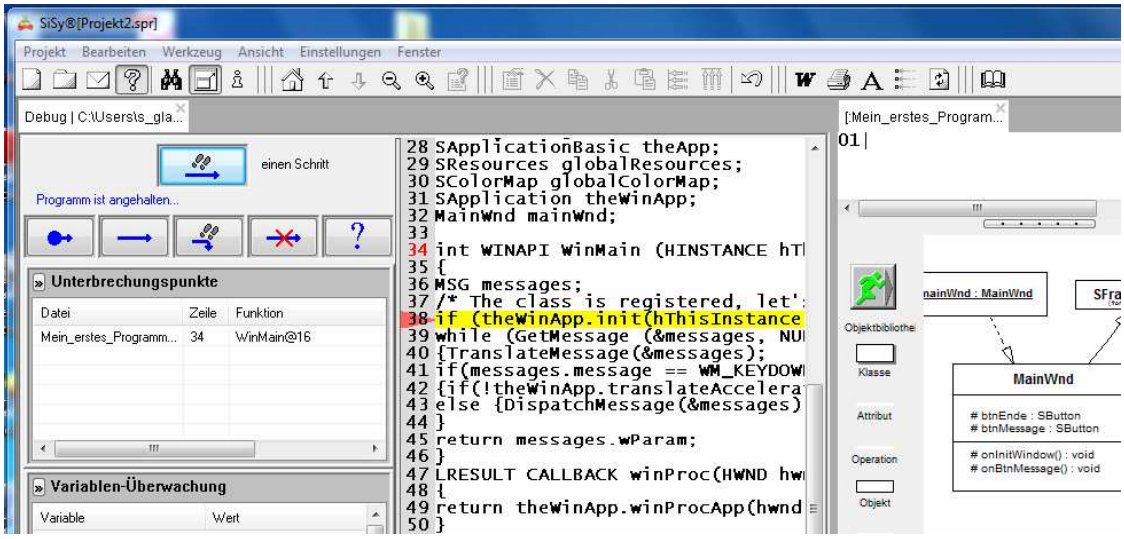
10.7.1 Debuggen von SVL Programmen

SiSy verfügt über ein Debug-Interface um C oder C++ Programme zu debuggen. Unter Debugging versteht man bestimmte Arbeitstechniken zur Fehlersuche in laufenden Programmen. Dazu gehören vor allem der Schrittbetrieb, Unterbrechungspunkte, Kontrollausgaben und Datenüberwachung.
Der integrierte Debugger ermöglicht es Ihnen, sowohl auf Quellcodeebene als auch auf Modellebene Fehler zu suchen. Das Starten des Debuggers erfolgt in der Regel über das Aktionsmenü des aktiven Diagramms.



Den Debugger starten

Der Debugger startet und wird mit dem dazugehörigen Quelltext angezeigt. Dabei wird in jedem Fall ein Unterbrechungspunkt beim Programmstart (main / WinMain) gesetzt und das Programm dort angehalten. Jetzt kann der Anwender weitere Unterbrechungspunkte hinzufügen. Die Position an der ein Programm angehalten wurde, ist im Quelltext als Zeile markiert. Wenn die entsprechende Position im Modell ermittelt werden konnte, wird das Modellfenster nachgeführt und das betreffende Element selektiert.



SiSy mit geöffneten Debugger im Klassendiagramm:

Unterbrechungspunkte können im Quellcodeeditor des Diagrammfensters hinzugefügt werden. Nutzen Sie dazu den Menüpunkt „Breakpoint umschalten“ im Kontextmenü des Editors.

Die Schaltflächen des Debuggers:

step (over)

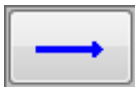
einen Schritt: dient dazu, das Programm Schritt für Schritt ablaufen zu lassen. Dies hilft vor allem bei der Fehlersuche, da hier Zeile für Zeile des Quellcodes entsprechend des Programmablaufs abgearbeitet wird. Funktionen werden dabei jedoch als ganze Einheit betrachtet.

step in to

einen Schritt hinein: verzweigt das Programm zusätzlich in die Einzelanweisungen von aufgerufenen Funktionen.

restart

Programm neu starten



go

Programm laufen lassen

end

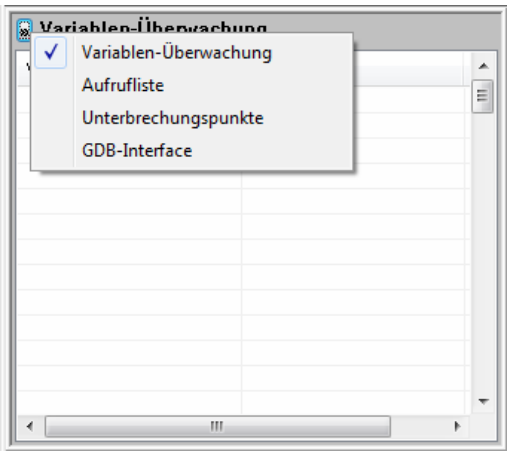
Debugger beenden; kann auch erfolgen, wenn das Programm nicht komplett abgearbeitet wurde.

help

Hilfe zu Debugger öffnen: hier werden weitere Hinweise zum Debugger gegeben.

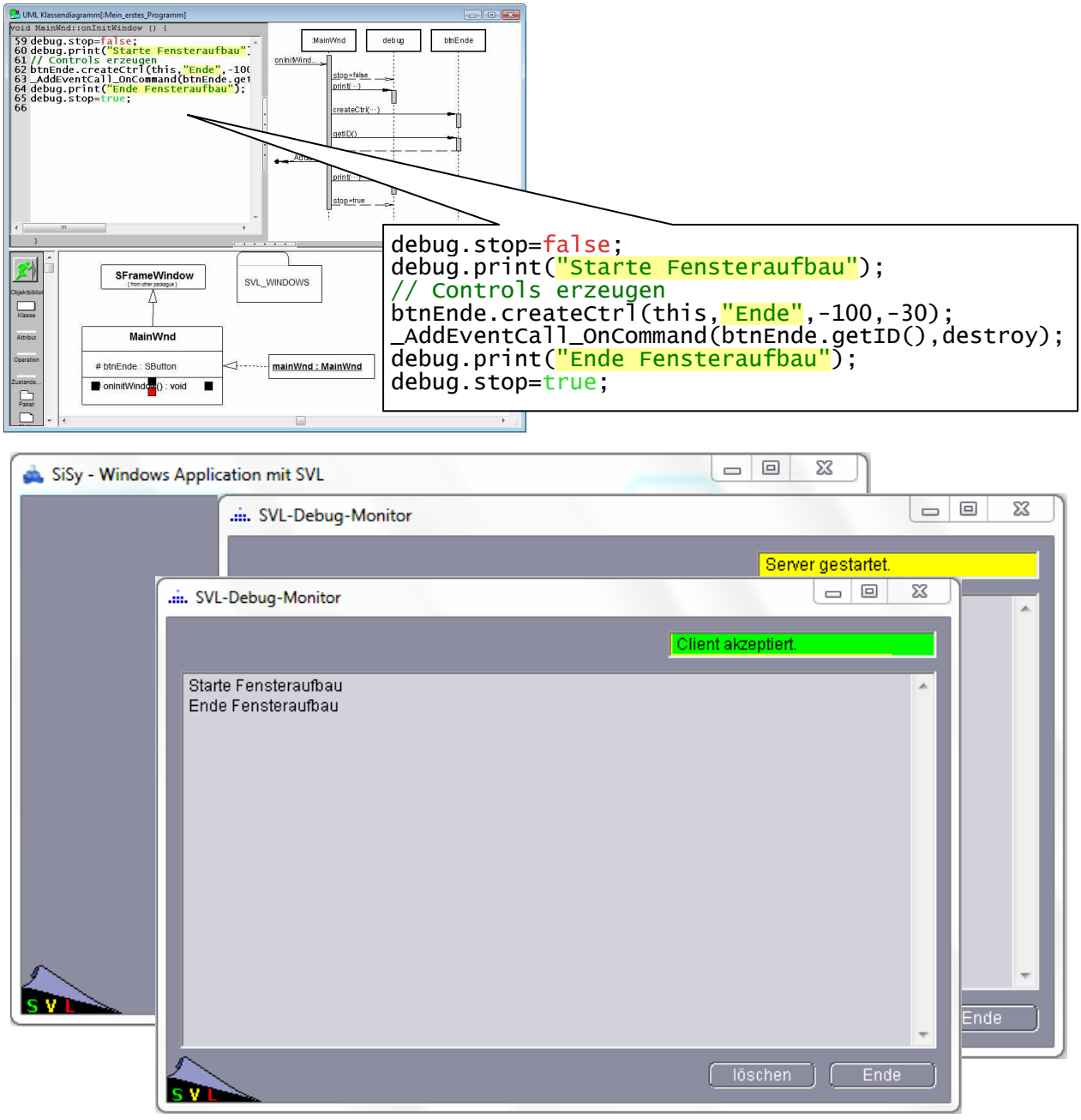
menu

Auswahl-Menü öffnen: ein Drop-Down Menü wird geöffnet, aus dem wie in der Abbildung dargestellt, verschiedene Debug-Ansichten gewählt werden können.



10.7.2 Der SVL-DebugMonitor

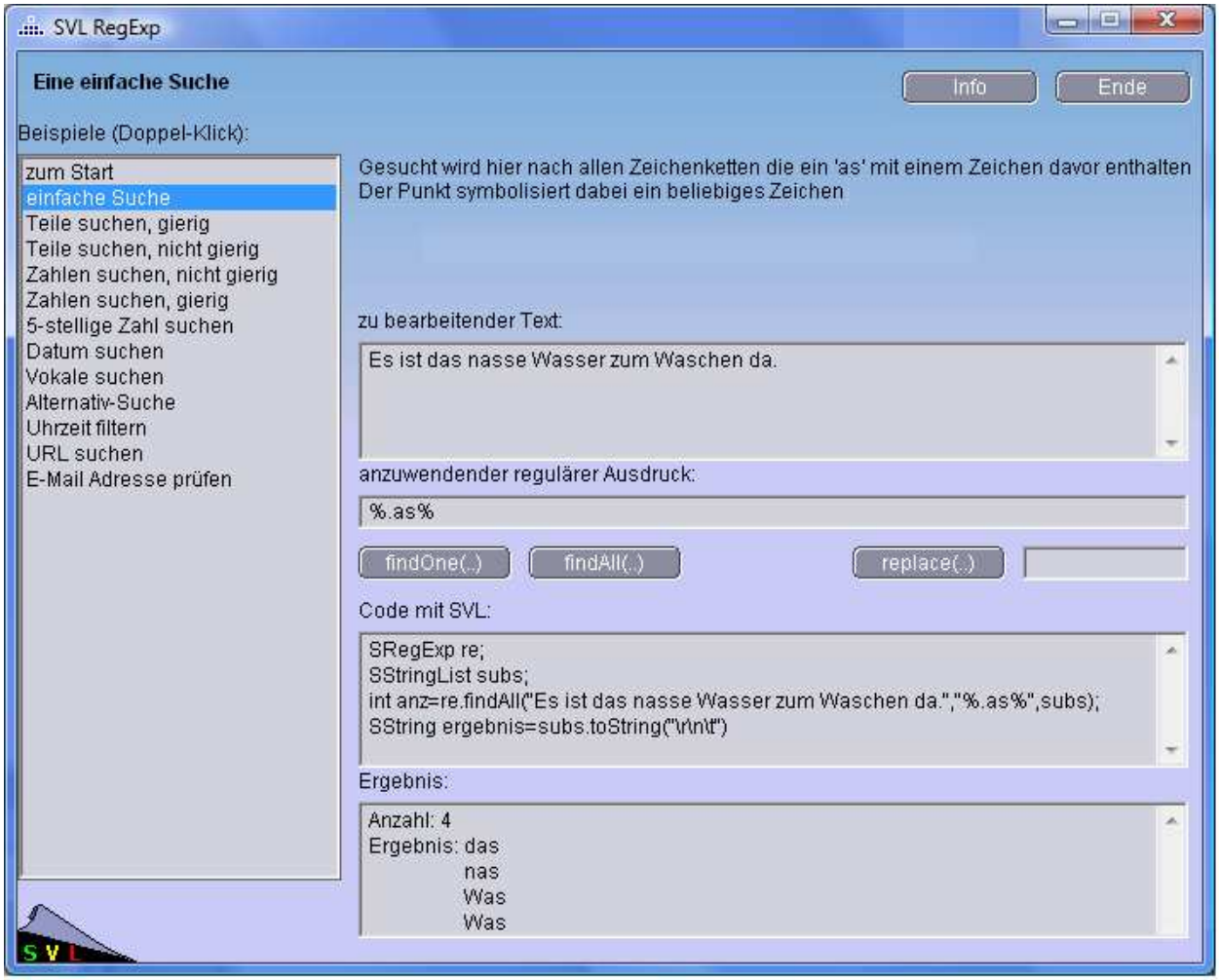
Für die Fehlersuche im laufenden Betrieb einer Windowsanwendung die auf der SVL-Klassenbibliothek basiert, steht Ihnen der SVL-Debugmonitor zur Verfügung. In der folgenden Darstellung sehen Sie eine typische Anwendung des Debugmonitors. Dabei fungiert der Debugmonitor als lokaler Server und die Anwendung als Client. Der Debugmonitor muss immer zuerst gestartet werden und danach erst die Anwendung. Sie können in der SVL Hilfe unter den Stichwort „SDebug“ detaillierte Informationen zur Nutzung der Debugschnittstelle abrufen.



<u>Hinweis:</u>
SVL Anwendungen können nicht mit jeder SiSy Ausgabe erstellt werden. Sie benötigen dazu die entsprechenden Add-Ons beziehungsweise die entsprechende Ausgabe von SiSy.
Bei der Nutzung des Debugmonitors kann es zu Warnhinweisen Ihrer Firewall kommen. Es ist für die korrekte Funktion des Debugmonitors notwendig, die Verbindung zwischen Anwendung und Debugmonitor nicht zu blockieren.

10.7.3 Das SVL-Werkzeug RegExp

Reguläre Ausdrücke sind Regeln für die Suche in Zeichenketten. Man definiert also mit dieser Syntax Suchmuster. Benutzen Sie dieses Werkzeug um die Syntax eines gewünschten regulären Ausdruckes zu testen. Sie erhalten zusätzlich den C++ Code für die Anwendung des getesteten regulären Ausdrucks mit der SVL. Weitere Informationen finden Sie in der SVL-Hilfe unter den Stichwort „SRegExp“.



Hinweis:

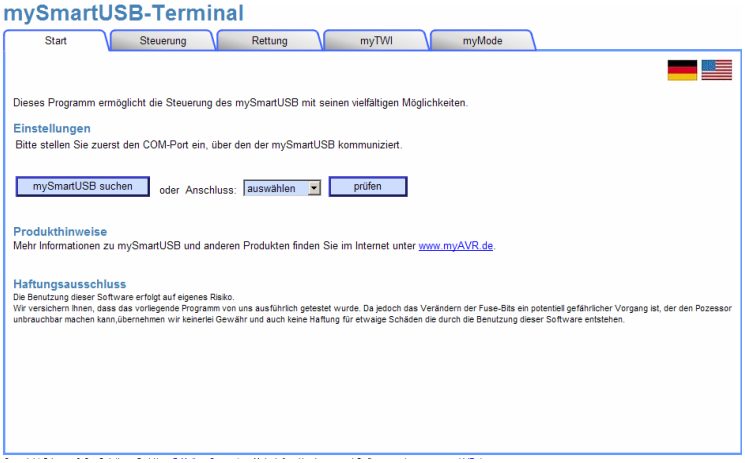
SVL Anwendungen können nicht mit jeder SiSy Ausgabe erstellt werden. Sie benötigen dazu die entsprechenden Add-Ons beziehungsweise die entsprechende Ausgabe von SiSy.

10.8 Weitere Werkzeuge

Entsprechend der installierten Add-Ons Ihrer SiSy Ausgabe können weitere Werkzeuge verfügbar sein.

mySmartUSB Terminal

Dieses Programm ermöglicht die Steuerung des mySmartUSB mit seinen vielfältigen Möglichkeiten. Es ist sehr gut geeignet, die einzelnen Schritte eines TWI- oder SPI-Protokolls auszuführen und die Abfolge der Aktionen und Rückmeldungen zu beobachten.



mySmartUSB-Light SupportBox

ein kleines Tool auf einer grafischen Oberfläche zum Schalten und Updaten des mySmartUSB light.

11 Informationen zu SiSy-Ausgaben

Die Laufzeitkomponenten von SiSy, das Metamodell, Add-Ons sowie spezifische Skripte, Dokumente und Daten sind lizenzpflichtige Produkte. Ausgenommen sind zusätzlich bzw. nachträglich installierte Komponenten Dritter, die von SiSy als Modellierungswerkzeug lediglich zum Beispiel über Skripte angesprochen werden oder in die sich SiSy über offen gelegte Schnittstellen eingebettet hat. Solche Produkte unterliegen den Lizenz und Nutzungsbedingungen des jeweiligen Herstellers bzw. Herausgebers.

Ausgaben

- SiSy Professional
  umfasst alle verfügbaren Add-Ons
- SiSy Business
  umfasst Add-Ons für Prozessmodellierung, Qualitäts- und Projektmanagement
- SiSy Developer
  umfasst umfangreiche Add-Ons zur Systementwicklung
- SiSy MC ++
  umfasst die Add-Ons AVR, ARM, ausgewählte Teile der UML, SVL und SysML
- SiSy AVR
  beinhaltet nur das Add-On AVR
- SiSy ARM
  umfasst die Add-Ons ARM, ausgewählte Teile der UML und SysML

Lizenzmodelle

- Evaluation License
  erlaubt die Nutzung auf einem Arbeitsplatzrechner für Evaluierungszwecke
  Lizenzlaufzeit 1 Jahr
- Education License
  erlaubt die Nutzung auf einem Arbeitsplatzrechner für Ausbildungszwecke
  Vorraussetzung: Bildungseinrichtung, Student, Schüler, keine kommerzielle Nutzung
- Private License
  erlaubt die Nutzung auf einem Arbeitsplatz für private oder hausinterne Zwecke
  Vorraussetzung: ausschließlich für private, nicht kommerzielle Nutzung
  in Unternehmen hausinterne Nutzung der mit SiSy erstellten Produkte
- Single License
  erlaubt die Nutzung auf einem Arbeitsplatz kommerzielle Zwecke oder
  in öffentlichen Verwaltungen
- Enterprise License
  erlaubt die Nutzung auf bis zu 20 Arbeitsplatzen für kommerzielle Zwecke oder
  in öffentlichen Verwaltungen

Anhang: Tastaturbelegung, allgemein

Die Tastenbelegung ist abhängig vom jeweiligen Diagramm und der verwendeten Ausgabe:

F1	Hilfe wird geöffnet
F2	Zoomen
F4	Textfeld / Infofeld vergrößern
F5	Farbe bei Rahmen wird geändert; Form am Anfang einer Verbindung ändert sich
F6	Bei Rahmen und Verbindungen ändert sich die Form
F7	Bei Rahmen und Verbindungen ändert sich der Linientyp
F8	Form am Ende einer Verbindung ändert sich
F9	Bewirkt, dass der Mittelpunkt einer Kante auf Null gesetzt wird
ESC	Im Diagramm: Nach oben
	Im Dialog: Bricht ihn ab ohne zu speichern
Tab	In der Reihenfolge, in der die Objekte erstellt wurden, werden sie markiert
Umschalttaste + Enter	Objektbeschreibung wird geöffnet
Leertaste	Objektbeschreibung wird geöffnet
Alt + Enter	Dialog Definieren
Strg + Enter	Diagramm nach unten
Strg + `R	Report für selektiertes Objekt
Strg + `A`	Executebefehl ausführen (nur in bestimmten Ausgaben)
Strg + `D`	Diagrammreport wird aufgerufen
Strg + `I`	Import von Diagrammen
Strg + `T`	Tokensteuerung starten/beenden
Strg + `X`	Export von Diagrammen
Strg + `+`	Selektiertes Objekt wird vergrößert (nur in bestimmten Ausgaben)
Strg + `-`	Selektiertes Objekt wird verkleinert (nur in bestimmten Ausgaben)
Strg + `*`	Ursprüngliche Objektgröße wird wiederhergestellt

Strg + Maustaste	Selektiertes Objekt wird innerhalb des Diagramms kopiert
Strg + Cursortasten:	
- Cursor nach links	Selektiertes Objekt wird in X-Richtung verkleinert
- Cursor nach rechts	Selektiertes Objekt wird in X-Richtung vergrößert
- Cursor nach oben	Selektiertes Objekt wird in Y-Richtung vergrößert
- Cursor nach unten	Selektiertes Objekt wird in Y-Richtung verkleinert
Enter	Editormodus zum Definieren der Objekte
Entf	Löschen
Cursortasten	Selektiertes Objekt wird verschoben (in Verbindung mit der Umschalttaste sind größere Schritte möglich)
`+`	Diagramm vergrößern
`-`	Diagramm verkleinern
`*`	Einpassen des Diagramms

Anhang: Mausoperationen

Die Maus hat in SiSy eine Anzahl von nützlichen Funktionen, welche die Arbeit in Projekten erleichtern.

Selektion	*Klick auf Objekt* Objekt ist markiert und kann separat weiterbearbeitet werden.
Selektion aufheben	*Klick auf Fensterhintergrund* Aufhebung der Objektmarkierung.
Mehrfachselektion	*Umschalttaste + Klick auf Objekt* Selektion/Markierung von mehreren Objekten zur Weiterbearbeitung. *Markise* Mit *Shift* und *gedrückter linker Maustaste* auf Fensterhintergrund und Ziehen eines Rechtecks über zu markierende Objekte.
Verschieben	*Drag & Drop im Diagramm* Objekt mit linker Maustaste anfassen und verschieben. Objekte werden am Raster verschoben. *Umschalttaste + Drag & Drop im Diagramm* Verschieben von Objekten ohne Raster.
Fensterinhalt schieben	*Linke und rechte Maustaste drücken + Verschieben der Maus im Diagramm* Der komplette Diagramminhalt wird geschoben.
Objekt kopieren	*STRG + Drag & Drop* Maustaste gedrückt halten und Mauszeiger vom Objekt auf den Fensterhintergrund führen. Eine Originalkopie des Objektes wird im aktuellen oder in einem anderen Diagramm erzeugt.
Referenz erzeugen	*Drag & Drop aus Navigator* Ziehen des gewünschten Objektes aus dem Navigator in das Diagramm. Es wird eine Referenz des gewählten Objektes erzeugt. *Drag & Drop aus Objektbibliothek* Rot beschriftete Objekte können nur als Referenz erzeugt werden. Eine Liste zur Auswahl des gewünschten Typs erscheint. *Strg + Drag & Drop aus Objektbibliothek* Eine Liste zur Auswahl der gewünschten Referenz des Originalobjektes erscheint.

	Drag & Drop aus anderem Diagramm Ziehen des gewünschten Objektes aus dem Quelldiagramm in das Zieldiagramm. Es wird eine Referenz des gewählten Objektes erzeugt.
Objekt anlegen	*Drag & Drop aus Objektbibliothek* Ein Objekt aus der Objektbibliothek wird im Diagramm angelegt und steht zur Verfeinerung bereit.
Objekt anhängen	*Drag & Drop Verteiler auf Fensterhintergrund* Durch Ziehen einer Kante vom Verteiler auf den Fensterhintergrund wird ein neues Objekt erzeugt. Nach Auswahl des Objekttyps sind die Objekte miteinander verbunden.
Objekte verbinden	*Drag & Drop Verteiler zu Objekt* Klick auf den Verteiler des zu verbindenden Objektes. Bei gedrückter linker Maustaste auf das gewählte Objekt ziehen. *Verbindung aus Objektbibliothek (in der UML)* Hierbei wird erst die gewünschte Verbindung in der Objektbibliothek angeklickt und danach die beiden zu verbindenden Objekte im Diagramm nacheinander.
Verbindung anordnen	*Drag & Drop Mittelpunkt* Beliebige Gestaltung der Verbindung durch Ziehen mit der Maus.
Verbindung ändern	*Drag & Drop Anfangs-/Endpunkt einer Kante* Für die Verbindung wird ein neues Zielobjekt gewählt.
Objekt definieren	*Doppelklick auf Objekt* Durch Doppelklick auf Objekte öffnet sich das Kontextmenü. Bei Abschalten des Menüs unter *Einstellungen*/*Menü bei Doppelklick* erscheint eine Zeile zur Namensgebung. Mit ESC wird die Eingabe bestätigt. *Doppelklick auf Verteiler* Es wird der Definieren-Dialog aufgerufen, in dem das Objekt benannt und beschrieben werden kann.
Kontextmenü öffnen	*Klick mit rechter Maustaste auf Objekt*
Fenster neu zeichnen	*Doppelklick auf Fensterhintergrund* *Hinweis***:** Doppelklick mit linker Maustaste wirkt wie Enter.
Fenster aktualisieren	*Strg + Doppelklick auf Fensterhintergrund* Die vom Programm ausgeführten aber noch nicht sichtbargemachten Befehle werden im Fenster erstellt. Das Fenster wird aktualisiert.

