ASCET V5.2

Schnelleinstieg

Copyright

Dokument EC010010 R5.2.2 DE

Inhalt

1 Einführung

ASCET bietet eine innovative Lösung für die funktionale Softwareentwicklung moderner, eingebetteter Softwaresysteme. ASCET unterstützt jeden Entwicklungsschritt mit einem neuen Ansatz zur Modellierung, Codegenerierung und Simulation, bessere Qualität, kürzere Innovationszyklen und Kostensenkungen sind die Folge.

Dieses Handbuch unterstützt den Leser beim Kennenlernen von ASCET, so dass schnell Ergebnisse erreicht werden. Es liefert eine schrittweise Einführung in das System; gleichzeitig können alle Informationen leicht nachgeschlagen werden.

1.1 Systeminformation

Die ASCET-Produktfamilie besteht aus mehreren Einzelprodukten, die Schnittstellen für Simulationsprozessoren, Softwarepakete Dritter und Fernzugriff auf ASCET bieten. Die folgenden Produkte sind für die aktuelle Version von ASCET erhältlich:

- *ASCET-MD* – Unterstützung für die Entwicklung und Simulation von Modellen.
- *ASCET-RP* – Unterstützung für Experimentaltargets, um Hardware in the Loop zu simulieren und schnell Anwendungsprototypen zu erstellen. Werkzeuge zum Durchführen von ETK-Bypass-Experimenten sind ebenfalls integriert. Mit ASCET-RP wird die Zusammenarbeit mit INTECRIO ermöglicht.
- *ASCET-SE* – Unterstützung für verschiedene Mikrocontroller-Targets. Erzeugung von optimiertem, ausführbarem Code für diverse Mikrcontroller und zwei Echtzeit-Betriebssysteme, einschließlich Betriebssystemkonfiguration und -einbindung.

Verschiedene Zusatzmodule sind gesondert erhältlich:

- *Konfigurationsverwaltung* – bietet eine Schnittstelle zu Werkzeugen für die Versions- und Konfigurationsverwaltung.
- *ASCET-MIP* – Dieses MATLAB-Integrationspaket beinhaltet zwei verschiedene Möglichkeiten, auf Matlab-Software zuzugreifen. Der erste Teil ist eine Schnittstelle, die es erlaubt, ASCET und MATLAB auf Simulationsebene zu verknüpfen. Der zweite Teil ist ein Modellkonverter, der das Lesen von Simulink-Modellen in ASCET ermöglicht.
- *ASCET-DIFF* – Ein Vergleichswerkzeug für ASCET-Modelle.

Unterschiedliche kundenspezifische Zusatzprodukte können in ASCET integriert werden. Näheres hierzu auf Anfrage.

1.2 Benutzerinformationen

1.2.1 Vorausgesetzte Kenntnisse

Dieses Handbuch richtet sich an ausgebildetes Fachpersonal im Entwicklungs- und Applikationsbereich von Kfz-Steuergeräten. Fachwissen in den Bereichen Mess- und Steuergerätetechnik wird vorausgesetzt.

ASCET-Benutzer sollten mit den Betriebssystemen Microsoft Windows 98, Windows NT 4.0, Windows 2000 oder Windows XP vertraut sein. Alle Benutzer sollten in der Lage sein, Menübefehle auszuführen, Schaltflächen zu aktivieren usw. Desweiteren sollte den Benutzern das Windows-Dateiablagesystem, insbesondere die Zusammenhänge von Dateien und Verzeichnissen, bekannt sein. Die grundlegenden Funktionen vom Windows-Datei- und Programm-Manager, bzw. Windows-Explorer, muss der Benutzer kennen und beherrschen. Weiterhin sollte der Anwender mit der Arbeitsweise „Drag & Drop" vertraut sein.

Alle Benutzer, die sich nicht mit den Grundtechniken von Microsoft Windows auskennen, sollten sich vor der Nutzung von ASCET damit vertraut machen. Für die Bedienung der Benutzeroberfläche Windows gelten die entsprechenden Handbücher der Microsoft Corporation.

Kenntnisse einer Programmiersprache, vorzugsweise ANSI-C oder Java können für fortgeschrittene Benutzer hilfreich sein.

1.2.2 Aufbau des Handbuchs

Das ASCET-Handbuch besteht aus drei Bänden:

1. Band „ASCET V5.2 – Schnelleinstieg" (dieser Band)

   In diesem Band finden Sie grundlegenden Informationen zur Arbeitsweise von ASCET.

2. Band „ASCET V5.2 – Benutzerhandbuch"

   In diesem Band ist die Bedienung von ASCET-MD beschrieben.

3. Band „ASCET V5.2 – Referenzhandbuch"

   In diesem Band befinden sich eine detaillierte Beschreibung der ASCET-Modelliersprache sovie zahlreiche Referenzinformationen.

Der Inhalt der einzelnen Bände wird im Folgenden genauer beschrieben.

Band „ASCET V5.2 – Schnelleinstieg"

Dieser Band besteht aus folgenden Kapiteln:

- **„Einführung“** (dieses Kapitel)

  Sie erhalten hier einen ersten Überblick über die Neuerungen der Version ASCET V5.2 sowie die Einsatzmöglichkeiten von ASCET. Des Weiteren finden Sie in diesem Kapitel allgemeine Informationen wie Benutzerhinweise und Systeminformationen.

- **„Programminstallation“**

  Dieses Kapitel richtet sich zum einen an alle Anwender, die ASCET auf einem PC oder im Netzwerk installieren, pflegen und deinstallieren, zum anderen an Systemadministratoren, die ASCET auf einem Fileserver für die Installation über das Netzwerk zur Verfügung stellen. Es werden wichtige Informationen über Lieferumfang, Hard- und Softwarevoraussetzungen für Einzelplatz- und Netzwerkinstallation und die Vorbereitung der Installation gegeben. Die Abfolge der Installation sowie die Deinstallation von ASCET wird beschrieben.

- **„Lizenzierung“**

  Dieses Kapitel enthält Informationen zur Lizensierung, z.B. zum Anfordern einer Lizenzdatei oder zum Ausleihen einer Lizenz.

- **„ASCET – Grundlagen“**

  Dieses Kapitel liefert einen Überblick über das ASCET-System, den Entwicklungsprozess, der durch das System unterstützt wird, und seinen Platz in der Kette von ETAS Tools.

  Dieses Kapitel sollte von allen neuen ASCET-Benutzern zuerst gelesen werden.

- **„Allgemeine Bedienmöglichkeiten von ASCET“**

  In diesem Kapitel erhalten Sie Informationen über die Fenster- und Menüstrukturen von ASCET sowie Bedienmöglichkeiten über Maus und Tastatur.

- **„Tutorial“**

  Das Tutorial richtet sich hauptsächlich an den ASCET-Neueinsteiger. Anhand von Beispielen erlernen Sie die Bedienung von ASCET. Der gesamte Lerninhalt wird dabei in einzelne kurze, aufeinander aufbauende Lernabschnitte unterteilt. Bevor Sie mit dem Tutorial arbeiten, sollten Sie das Kapitel „ASCET – Grundlagen“ auf Seite 51 durchgearbeitet haben.

Hinweis

Um eine noch gründlichere Einarbeitung in ASCET zu bieten, besonders wenn Sie in möglichst kurzer Zeit ein umfassendes Wissen über das Arbeiten mit ASCET erlangen wollen, bietet ETAS effiziente Schulungen an.

- **„Glossar“**

  In diesem Kapitel werden alle im Handbuch vorkommenden Abkürzungen und Fachbegriffe erklärt. Die Begriffe sind in alphabethischer Reihenfolge aufgelistet.

- **„Referenzlisten“**

  In diesem Kapitel erhalten Sie Informationen zur Fehlerbehandlung, Angaben zur Verzeichnisstruktur und der benötigten Referenzdateien. Des Weiteren finden Sie hier eine Liste sämtlicher Tastaturbefehle, die nach Arbeitsfenstern sortiert ist.

Band „ASCET V5.2 – Benutzerhandbuch“

Dieser Band beschreibt umfassend alle Komponenten des ASCET-Systems und gibt detaillierte Anweisungen zu deren Verwendung. Bevor Sie sich mit diesem Teil beschäftigen, sollten Sie sich mit der ASCET-Software vertraut gemacht haben, d. h. das Kapitel „ASCET – Grundlagen“ und das „Tutorial“ aus dem ASCET-Schnelleinstieg durchgearbeitet haben.

Die Beschreibung des ASCET-Systems ist so aufgebaut, dass sie die chronologische Reihenfolge des Entwicklungsprozesses eines Embedded Control Systems in ASCET widerspiegelt.

Dieser Band besteht aus folgenden Kapiteln:

- **„Einführung“**

  Erläuterungen zum typischen Arbeitsablauf

- **„Der Komponentenmanager“**

  Oberfläche, Bedienelemente und Menübefehle kennenlernen, ASCET individuell anpassen, Datenmanagement kennenlernen.

  Dieses Kapitel ist für alle Benutzer von ASCET von Bedeutung.

- **„Einfügen benutzerdefinierter Funktionen“**

  Dieses Kapitel beschreibt, wie Anwender eigene Menüfunktionen in verschiedenen ASCET-Fenstern definieren können.

- **„Spezifikation von Komponenten und Projekten“**

  Dieses Kapitel beschreibt das Arbeiten mit den verschiedenen Spezifikationseditoren, und wie die Attribute der verwendeten Elemente bearbeitet werden können.

- **„Signale und Symbole“**

  Dieses Kapitel beschreibt, wie Signale und Symbole in der Datenbank verwaltet und integriert werden.

- **„Experimentieren“**

  Dieses Kapitel beschreibt das Arbeiten mit der Experimentierumgebung im Offline-Experiment und gibt einen Überblick über die verschiedenen Mess- und Verstellfenster.

  Online-Experimente und Experimemntieren mit INCA oder INTECRIO sind im ASCET-RP-Handbuch beschrieben.

- **„Automatische Dokumentation“**

  Dieses Kapitel beschreibt die automatische Erstellung von Dokumentation zu ASCET-Komponenten und Projekten.

Band „ASCET V5.2 – Referenzhandbuch“

Der erste Teil dieses Bandes, „Die Modelliersprache“, ist ein umfassendes Nachschlagewerk zu den verschiedenen Möglichkeiten, eingebettete Softwaresysteme in ASCET zu beschreiben. Es ist ratsam, vor dem Lesen eines der Kapitel dieses Teils zunächst das „Tutorial“ aus dem ASCET-Schnelleinstieg durchzuarbeiten.

Dieser Teil besteht aus folgenden Kapiteln:

- **„Projekte“**

  Die Spezifikation eines eingebetteten Steuerungssystems wird Projekt genannt. Die Struktur eines Projekts wird hier beschrieben.

- **„Komponenten“**

  Komponenten sind die Bausteine eines eingebetteten Systems. Die verschiedenen Arten von Komponenten werden in diesem Kapitel vorgestellt.

- **„Typen und Elemente“**

  Dieses Kapitel beschreibt die Arten von Variablen und Datentypen, die von ASCET unterstützt werden.

- **„Daten und Implementierungen“**

  Jede Variable in ASCET besitzt Daten und eine Implementierung, die hier erörtert werden.

- **„Spezifikation in ESDL“**

  Dieses Kapitel behandelt das Spezifizieren von Komponenten in ESDL, der textuellen Modell-Beschreibungssprache von ASCET.

- **„Spezifikation mit Blockdiagrammen“**

  In diesem Kapitel wird die Spezifikation von Komponenten in Form von Blockdiagrammen beschrieben.

- **„Spezifikation in C“**

  Komponenten können auch in C spezifiziert werden, was hier diskutiert wird.

- **„Zeitkontinuierliche Systeme“**

  Ein Überblick über zeitkontinuierliche Systeme in ASCET, die verwendet werden, um technische Prozesse mathematisch zu modellieren.

- **„Zeitkontinuierliche Basisblöcke“**

  Basisblöcke beschreiben individuelle Komponenten eines zeitkontinuierlichen Systems.

- **„Zeitkontinuierliche Strukturblöcke und grafische Hierarchien“**

  Strukturblöcke integrieren die Basisblöcke in vollständige Modelle.

- **„Projekte und hybride Projekte“**

  In kombinierten Projekten können zeitkontinuierliche Systeme neben eingebetteten Systemspezifikationen laufen.

Der zweite Teil, „Referenzlisten“, beschreibt die Systembibliothek von ASCET und gibt andere Referenzinformationen.

Dieser Teil besteht aus folgenden Kapiteln:

- **„ASCET-Systembibliothek“**

  Dieses Kapitel gibt eine genaue Beschreibung jeder Komponente der Systembibliothek.

- **„Fehlersuche und Störungsbeseitigung“**

  Hier werden häufige Benutzerfehler und bekannte Probleme aufgelistet sowie Lösungsvorschläge unterbreitet.

- **„Meldungen bei der Codegenerierung“**

  Dieses Kapitel beinhaltet die Fehlermeldungen, die während der Codegenerierung auftreten können, und Ratschläge, wie das Softwaremodell anzupassen ist, um solche Fehler zu vermeiden.

1.2.3 Umgang mit dem Handbuch

Dokumentationsvereinbarungen

Alle vom Anwender auszuführenden Tätigkeiten werden in einem aufgabenorientierten Format dargestellt, wie das folgende Beispiel veranschaulicht. Eine *Aufgabe* in diesem Handbuch ist eine Folge von Aktionen, die ausgeführt werden müssen, um ein bestimmtes Ziel zu erreichen. Der Titel einer Aufgabenstellung gibt für gewöhnlich das Ergebnis der Aktionen an, z. B. „Eine neue

Komponente erstellen", oder „Ein Element umbenennen". Aufgabenstellungen enthalten oft Darstellungen des entsprechenden ASCET-Fensters, auf das sich die Aufgabe bezieht.

Erreichen eines Ziels:

---

eventuelle Vorabinformationen...

- Führen Sie Aktion 1 aus.

  Erklärungen werden unter einer Aktion gegeben.
- Führen Sie Aktion 2 aus.

  eventuelle Erklärungen zu Schritt 2...
- Führen Sie Aktion 3 aus.

  eventuelle Erklärungen zu Schritt 3...

eventuelle abschließende Bemerkungen...

konkretes Beispiel:

Definieren eines Signals:

Ein Signal kann nur definiert werden, wenn das Experiment beendet ist. Gehen Sie wie folgt vor:

- Wählen Sie im Datengenerator **Signal** → **Define Signal**.

  Das Dialogfenster „Select a Signal Item" wird geöffnet.
- Wählen Sie in der Fläche „Items" ein Signal.
- Klicken Sie auf **OK**.

Damit wird das Signal festgelegt. Sie können nur einen Signaleintrag pro Experiment verwenden.

Typografische Konventionen

Folgenden typografischen Konventionen werden verwendet:

Wählen Sie **Datei** → **Öffnen**.	Menübefehle werden fett/blau dargestellt.
Klicken Sie **OK**.	Schaltflächen werden fett/blau dargestellt.
Drücken Sie <ENTER>.	Tastaturbefehle werden in spitzen Klammern, in Kapitälchen dargestellt.
Das Dialogfenster „Datei öffnen" öffnet sich.	Namen von Programmfenstern, Dialogfenstern, Feldern u.ä. werden in Anführungszeichen gesetzt.

Wählen Sie die Datei `setup.exe` aus.	Text in Auswahllisten auf dem Bildschirm, Programmcode, sowie Pfad- und Dateinamen werden in der Schriftart `Courier` dargestellt.
Eine Konvertierung zwischen den Datentypen logisch und arithmetisch ist *nicht* möglich.	Inhaltliche Hervorhebungen und neu eingeführte Begriffe werden *kursiv* gesetzt
Die OSEK-Gruppe (siehe http://www.osekvdx.org/) hat einige Standards entwickelt.	Links zu Internet-Dokumenten werden in blauer Schrift und unterstrichen dargestellt.

Wichtige Hinweise für den Anwender werden so dargestellt:

Hinweis

Wichtiger Hinweis für den Anwender.

2 Programminstallation

Das Kapitel „Programminstallation“ richtet sich zum einen an alle Anwender, die ASCET auf einem PC oder im Netzwerk installieren, pflegen und deinstallieren, zum anderen an Systemadministratoren, die ASCET auf einem Datei-Server für die Installation über das Netzwerk zur Verfügung stellen. Es werden wichtige Informationen über Lieferumfang, Hard- und Softwarevoraussetzungen für Einzelplatz- und Netzwerkinstallation und die Vorbereitung der Installation gegeben. Die Abfolge der Installation sowie die Deinstallation von ASCET wird beschrieben.

2.1 Vorbereitung

Überprüfen Sie den Lieferumfang auf Vollständigkeit und Ihren Rechner auf Übereinstimmung mit den Systemvoraussetzungen. Je nach verwendetem Betriebssystem und Netzwerkanbindung müssen Sie sicherstellen, dass Sie die erforderlichen Benutzerrechte haben.

2.1.1 Lieferumfang

Zum Lieferumfang von ASCET gehören:

- ASCET CD-ROM
  - ASCET-Programmdateien
  - ASCET-Handbücher und ETAS Hardware-Dokumentation im PDF-Format (AcrobatReader)
  - Handbuch „FLEXnet Licensing End User Guide“ im PDF-Format
  - AcrobatReader-Programmdateien
- Handbuch „ASCET Getting Started“ (nur ASCET-MD)

2.1.2 Systemvoraussetzungen

Folgende Systemvorausetzungen sind erforderlich:

- 1 GHz Pentium-PC (empfohlen: 2 GHz Pentium IV)
- WINDOWS® 2000, WINDOWS® XP
- 512 MB RAM; 512 MB RAM empfohlen
- Festplatte mit mindestens 1 GB freiem Speicherplatz (Platz für Programmdaten nicht mitgerechnet; empfohlen: > 1 GB)
- CD-ROM-Laufwerk für die Installation
- VGA-Grafikkarte mit VGA-Bildschirm und einer Auflösung von mindestens 800 x 600 bei 256 Farben

2.1.3 Benutzerrechte für Installation und Betrieb

Notwendige Benutzerrechte bei der Installation:

Um ASCET auf dem PC zu installieren, benötigen Sie die Benutzerrechte eines Administrators. Wenden Sie sich gegebenenfalls an Ihren Systemadministrator.

Notwendige Benutzerrechte beim Betrieb (unter WIN 2000/XP):

Um mit ASCET unter WIN 2000/XP zu arbeiten, muss jeder Anwender das Recht „Anheben der Zeitplanungspriorität" vom Administrator erhalten. Einstellbar ist dies über die lokalen Sicherheitsrichtlinien (WIN 2000).

Hinweis

Um die im Folgenden beschriebenen Einstellungen vorzunehmen, benötigen Sie Administratorrechte.

Empfehlung: Weisen Sie das Recht „Anheben der Zeitplanungspriorität" der lokalen Gruppe „Benutzer" zu. Gehen Sie dazu wie folgt vor:

WIN 2000 – Benutzerrecht „Anheben der Zeitplanungspriorität" zuweisen:

- Wählen Sie über das Windows-Startmenü **Einstellungen** → **Systemsteuerung** → **Verwaltung** → **Lokale Sicherheitsrichtlinie**.
- Aktivieren Sie unter **Lokale Richtlinien** → **Zuweisen von Benutzerrechten** den Eintrag `Anheben der Zeitplanungspriorität` mit Doppelklick.
- Klicken Sie auf die Schaltfläche **Hinzufügen.**
- Wählen Sie den lokalen Rechner aus.
- Weisen Sie das Recht „Anheben der Zeitplanungspriorität" mit einem Doppelklick der Benutzergruppe `Benutzer` zu.
- Bestätigen Sie mit der Schaltfläche **OK**.
- Schließen Sie das Fenster „Lokale Sicherheitsrichtlinie" mit **OK**.
- Beenden Sie die lokalen Sicherheitseinstellungen.

WIN XP – Benutzerrecht „Anheben der Zeitplanungspriorität“ zuweisen:

- Wählen Sie über das Windows-Startmenü **Einstellungen** → **Systemsteuerung** → **Verwaltung** → **Lokale Sicherheitsrichtlinie**.
- Aktivieren Sie unter **Lokale Richtlinien** → **Zuweisen von Benutzerrechten** den Eintrag `Anheben der Zeitplanungspriorität` mit Doppelklick.

  Das Fenster „Eigenschaften von Anheben der Zeitplanungspriorität“ wird angezeigt.
- Klicken Sie auf die Schaltfläche **Benutzer oder Gruppe hinzufügen**.

  Das Fenster „Benutzer oder Gruppen wählen“ wird angezeigt.
- Klicken Sie auf die Schaltfläche **Pfade**.

  Das Fenster „Pfad“ wird angezeigt.
- Wählen Sie den lokalen Rechner aus und klicken Sie **OK**, um das Fenster zu schließen.
- Klicken Sie **Erweitert**, um die Suchfunktion im Fenster „Benutzer oder Gruppen wählen“ einzu-blenden.
- Klicken Sie **Jetzt suchen**, um die Liste der Benutzer für Ihren Rechner anzuzeigen.
- Wählen Sie in der Spalte „Name (RDN)“ den Benutzer oder die Gruppe aus, der Sie das Recht „Anheben der Zeitplanungspriorität“ zuweisen wollen.
- Bestätigen Sie mit der Schaltfläche **OK**.
- Schließen Sie das Fenster „Benutzer oder Gruppen wählen“ mit **OK**.
- Schließen Sie das Fenster „Eigenschaften von Anheben der Zeitplanungspriorität“ mit **OK**.
- Beenden Sie die lokalen Sicherheitseinstellungen.

2.2 Installation

Um ASCET-MD, ASCET-RP oder ASCET-SE (siehe Kapitel 4.4 auf Seite 85) zu installieren, müssen Sie in jedem Fall erst das ASCET-Grundsystem installieren. In diesem Kapitel wird die Installation des Grundsystems und ASCET-MD beschrieben; die Installationen von ASCET-RP und ASCET-SE sind in den entsprechenden Handbüchern beschrieben.

Die Installation ist unabhängig davon, ob Sie ASCET von CD-ROM oder dem Netzlaufwerk installieren.

Besonderheiten, die bei der Installation zu beachten sind (z.B. „Installationsvorgang abbrechen“ oder „Überschreiben einer bestehenden Programmversion“), werden im Kapitel 2.2.2 auf Seite 26 beschrieben.

2.2.1 Erstinstallation

ASCET-Grundsystem

ASCET Installation starten:

- Wählen Sie im Startmenü den Eintrag **Ausführen**.
- Geben Sie in der Befehlszeile den entsprechenden Pfad zu der Installationsdatei ein (bei CD-Installation z.B.: `d:\ASCET5.2\ascet.exe`).
- Bestätigen Sie mit **OK**.

  Das Installationsprogramm wird gestartet.

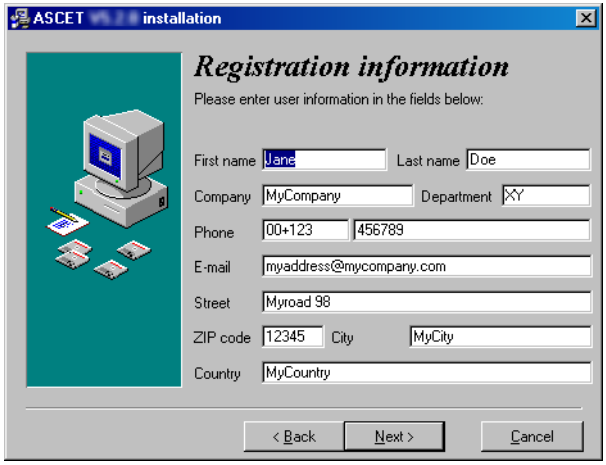
Anweisungen beachten:

- Akzeptieren Sie im Fenster „EULA“ die Lizenzvereinbarungen mit der Option **Accept**.
- Klicken Sie **OK**.
- Folgen Sie den Anweisungen am Bildschirm.

  Mit der Schaltfläche **Next** übernehmen Sie Ihre Eingaben und gehen ins nächste Installationsfenster, mit **Back** gelangen Sie ins vorige Fenster und mit **Cancel** brechen Sie die Installation ab.

ASCET-Registrierung durchführen:

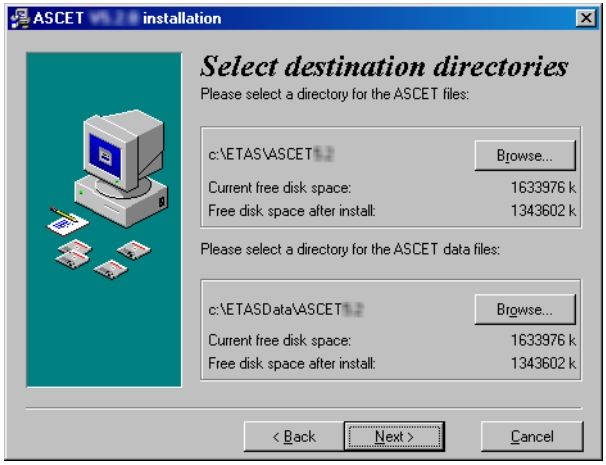
- Geben Sie in dem erscheinenden Registrierungsfenster Ihre persönlichen Angaben ein.



- Klicken Sie auf **Next**.

ASCET-Pfadangaben festlegen:

Nach erfolgter Registrierung werden Sie aufgefordert, Zielverzeichnisse für die Daten anzugeben. Dies geschieht in zwei unterschiedlichen Fenstern:



Programm*dateien* und Programm*daten* werden in unterschiedlichen Verzeichnissen abgelegt. Bei einer späteren Deinstallation bzw. einem Update können so nur die Programmdateien gelöscht bzw. überschrieben werden. Die Programmdaten stehen Ihnen weiterhin zur Verfügung. Zu den Programmdaten gehören u.a.:

- Datenbanken
- Anwenderprofile

Hinweis

Sie können ASCET in ein Verzeichnis installieren, dessen Pfad Leerzeichen enthält. Vergewissern Sie sich ***vorher****, ob auch alle externen Werkzeuge, die mit ASCET verwendet werden, einen solchen Pfad mit Leerzeichen ebenfalls unterstützen.*

- Wenn Sie die voreingestellten Verzeichnisse ändern wollen, klicken Sie auf die Schaltfläche **Browse**.
- Wählen Sie in dem Dialogfenster das gewünschte Verzeichnis aus.

  Wenn Sie ein Verzeichnis angeben, welches nicht existiert, wird es von der Installationsroutine automatisch angelegt.
- Klicken Sie auf **Next**.

  Das Fenster „Select global directories" erscheint. Hier können Sie die Verzeichnisse für Log-Dateien und temporäre Dateien angeben. Diese Verzeichnisse werden von allen ETAS-Produkten genutzt.
- Verwenden Sie die **Browse**-Schaltflächen, um die Pfadeinstellungen entsprechend Ihren Wünschen anzupassen.
- Klicken Sie auf **Next**.

Einstellungen für gemeinsam genutzte Dateien:

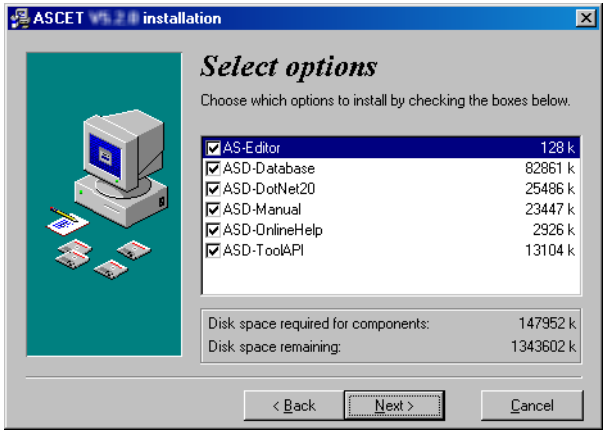
Im Fenster „Select Handling of ETAS-Shared modules“ legen Sie fest, wie die Module installiert werden, die grundsätzlich von allen ETAS-Programmen verwendet werden.

- Aktivieren Sie die Option **share modules between products**, wenn die Module gemeinsam genutzt werden sollen.

  Diese Einstellung ist sinnvoll, wenn Sie mehrere ETAS Produkte gleichzeitig nutzen wollen.
- Aktivieren Sie die Option **use local copies for each product**, wenn ASCET mit eigenen Kopien der Module installiert werden soll, wenn z.B. Anpassungen an den Komponenten geplant sind.
- Klicken Sie auf die Schaltfläche **Browse**, wenn Sie den Pfad für die Module bearbeiten wollen.
- Klicken Sie auf **Next**.

ASCET-Funktionsumfang festlegen:

Im Fenster „Select options" legen Sie den Funktionsumfang von ASCET fest.



- Markieren Sie die Optionen, die Sie installieren wollen.

  Folgende Optionen können ausgewählt werden:

  – *AS-Editor* – der Editor für arithmetische Dienste.

  – *ASD-Database* – die ETAS-Systembibliothek und die Tutoriumsdatenbank werden in das Exportverzeichnis Ihrer ASCET-Installation geschrieben.

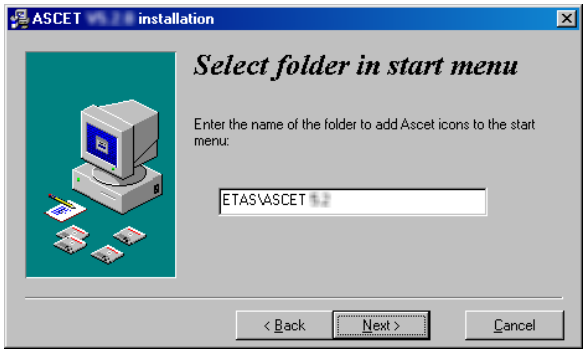
  – *ASD-Manual* – die ASCET-Dokumentation wird in Form von PDF-Dateien im Verzeichnis `ETAS\ETASManuals` abgelegt.

  – *ASD-OnlineHelp* – die ASCET-Onlinehilfe wird unter `ETAS\ASCET5.2\Help` abgelegt.

  – *ASD-ToolAPI* – Automation Interface für ASCET.

- Klicken Sie auf **Next**.

ASCET-Ordner im Startmenü festlegen:



- Übernehmen Sie die voreingestellte Ordnerbezeichnung

oder

- geben Sie einen anderen Ordnernamen ein.
- Klicken Sie auf **Next**.

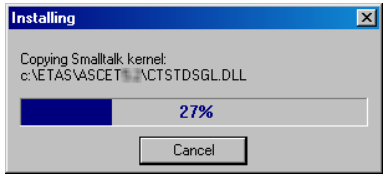
ASCET installieren:

Hinweis

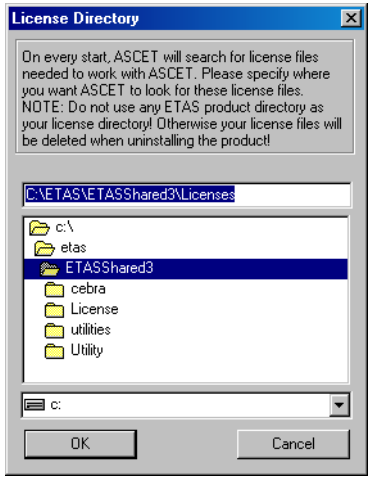
Mit dem nächsten Schritt starten Sie die Installation.

- Klicken Sie im Fenster „Ready to install“ auf **Next**, um die Installation zu starten.

  Die Programmdateien werden kopiert. An einem Balkendiagramm können Sie den Kopierfortschritt ablesen.



Nachdem alle Dateien kopiert sind, öffnet sich das Dateiauswahlfenster „License Directory“.



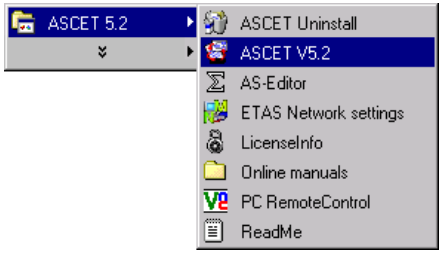
Verzeichnis für die Lizenzdatei angeben:

- Geben Sie im Fenster „License Directory“ das Verzeichnis ein, in dem die Lizenzdatei abgelegt werden soll.

  Jedes Mal, wenn ASCET eine Lizenz anfragt, wird dieses Verzeichnis durchsucht.
- Klicken Sie auf **OK**, um Ihre Auswahl zu bestätigen.

  Die Installation wird fertiggestellt und schließlich wird das Fenster „Installation completed“ angezeigt.
- Klicken Sie im Fenster „Installation completed“ auf **Finish**, um die Installation abzuschließen.

Sie finden im Startmenü den von Ihnen angegebenen Ordnernamen mit folgenden Einträgen:



- **ASCET Uninstall**

  Die Deinstallationsroutine wird gestartet (siehe Kapitel 2.4).

- **ASCET V5.2**

  Das Programm ASCET wird gestartet.

- **AS-Editor**

  Der AS-Editor (siehe Kapitel 4.14 im ASCET-Benutzerhandbuch) wird aufgerufen.

- **ETAS Network settings**

  Startet den Assistenten für die Konfiguration des ETAS-Netzwerks.

- **LicenseInfo**

  Das Fenster "Obtain License Info" wird gestartet (s. Kapitel 3).

- **Online manuals**

  Haben Sie bei der Installation die Online-Handbücher mitinstalliert, können Sie hier das Verzeichnis `ETAS\ETASManuals` öffnen. In diesem ordner sind, in verschiedenen Unterverzeichnissen, die Handbücher abgelegt.

- **PC RemoteControl**

  Ermöglicht die Konfiguration der Remote-Schnittstelle.

- **ReadMe**

  Hier erhalten Sie aktuelle Information zu ASCET V5.2.

ASCET-MD

Nachdem Sie das Grundsystem installiert haben, können Sie ASCET-MD installieren.

ASCET-MD installieren:

- Wählen Sie im Startmenü den Eintrag **Ausführen**.
- Geben Sie in der Befehlszeile den entsprechenden Pfad zu der Installationsdatei ein (bei CD-Installation z.B.: `d:\ASCET-MD V5.2\ASCET-MD.exe`).
- Bestätigen Sie mit **OK**.

  Das Installationsprogramm wird gestartet. Da das ASCET-Grundsysrem bereits installiert ist, brauchen Sie keine Angaben über Pfade, Funktionsumfang etc. zu machen.
- Klicken Sie auf **Next**, um ins nächste Installationsfenster zu gehen.
- Klicken Sie auf **Back**, um ins vorige Installationsfenster zu gehen.
- Klicken Sie auf **Cancel**, um die die Installation abzubrechen.

2.2.2 Besondere Installationsschritte und Dialoge

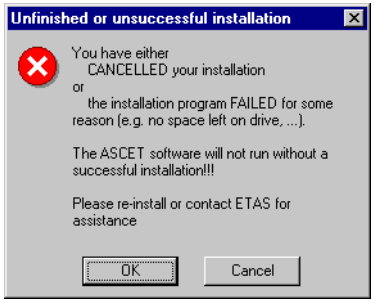
Installation abbrechen:

Zu jedem Zeitpunkt der Installation haben Sie die Möglichkeit, die Installation vorzeitig abzubrechen. Gehen Sie wie folgt vor:

- Klicken Sie in dem jeweiligen Dialogfenster auf die Schaltfläche **Cancel**.

- Wenn Sie auf **Resume** klicken, gelangen Sie wieder in das vorherige Dialogfenster.

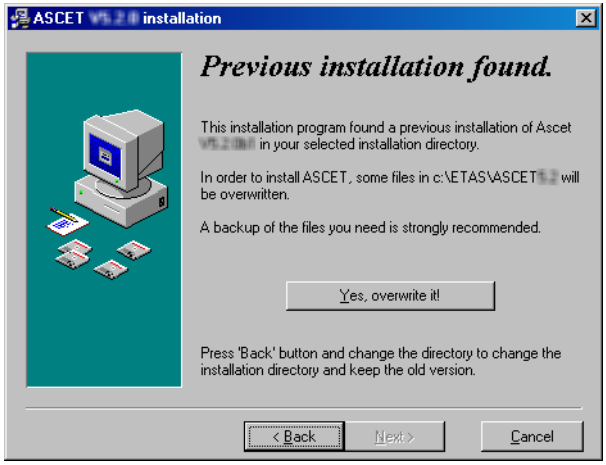
- Klicken Sie auf die Schaltfläche **Exit Setup**.



- Bestätigen Sie den Installationsabbruch mit **OK**.

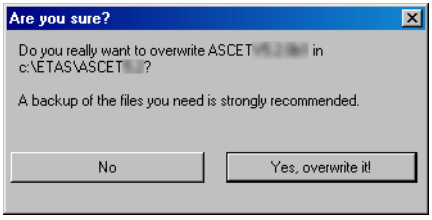
Bestehende Programmversion überschreiben:

Wenn eine ältere Version der zu installierenden Software auf dem Zielrechner gefunden wird oder eine völlig andere Software im ausgewählten Installationsverzeichnis vorhanden ist, so erscheint ein entsprechendes Dialogfenster. Der folgende Beispieldialog würde bei einer Installation der Version 5.2.0 erscheinen, wenn auf dem Zielrechner bereits die (Beta-)Version 5.2.0b1 installiert ist.



- Lesen Sie sich die Informationen genau durch.

  Sie werden darauf hingewiesen, dass eine frühere Installation gefunden wurde, und dass beim Installieren Dateien überschrieben werden.
- Wählen Sie zur Auswahl eines anderen Verzeichnisses die Schaltfläche **Back**.
- Wenn Sie die bestehenden Dateien überschreiben wollen, klicken Sie auf die Schaltfläche **Yes, overwrite it!**.

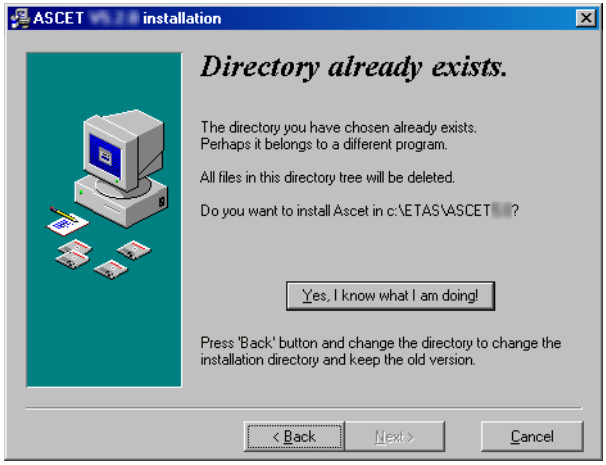


- Bestätigen Sie auch die Sicherheitsabfrage mit **Yes, overwrite it!**.

  Wenn Sie hier die Schaltfläche **No** verwenden, gelangen Sie wieder in das vorherige Fenster.

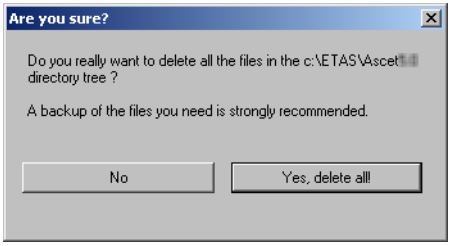
Existierende Verzeichnisse überschreiben:

Wenn die bei der Installation angegebenen Verzeichnisse bereits existieren, jedoch keine vollständige ASCET Installation auf dem Zielrechner gefunden wird, erscheint folgendes Dialogfenster. Dieser Fall tritt auf, wenn z.B. zuvor eine Installation abgebrochen wurde.



- Lesen Sie sich die Informationen genau durch.

  Sie werden darauf hingewiesen, dass das von Ihnen gewählte Verzeichnis bereits existiert.
- Wählen Sie zur Auswahl eines anderen Verzeichnisses die Schaltfläche **Back**.
- Wenn Sie die bestehenden Verzeichnisse überschreiben wollen, klicken Sie auf die Schaltfläche **Yes, I know what I am doing!**.



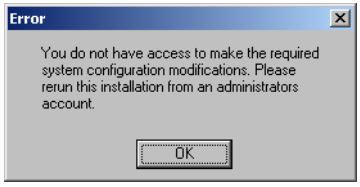
- Bestätigen Sie auch die Sicherheitsabfrage mit **Yes, delete all!**.

  ASCET wird installiert; bestehende Dateien werden gelöscht.

  Wenn Sie hier die Schaltfläche **No** verwenden, gelangen Sie wieder in das vorherige Fenster.

Installation ohne Administratorrechte:

Dieses Hinweisfenster erscheint, wenn der Benutzername, mit dem Sie sich unter Windows angemeldet haben, keine Administratorrechte besitzt. Die Installation wird nach Bestätigung der Meldung abgebrochen, da Administratorrechte Voraussetzung für die ASCET Installation sind.



- Bestätigen Sie mit **OK**.

  Die Installation wird abgebrochen.
- Wenden Sie sich an Ihren Systemadministrator.
- Führen Sie nach Erhalt der erforderlichen Rechte die Installation erneut aus.

2.3 Netzwerkinstallation

Neben der Installation von der CD können Sie ASCET auch von einem Netzlaufwerk aus auf dem PC installieren.

Bei der Netzwerkinstallation haben Sie den Vorteil, dass Sie schon vor der eigentlichen Installation auf dem Arbeitsplatzrechner eine Datenanpassung vornehmen können (siehe Abschnitt 2.3.2).

2.3.1 Daten im Netz bereitstellen

Hierzu müssen Sie die Installationsdateien von der CD auf das gewünschte Netzlaufwerk kopieren.

Daten auf dem Netzwerk-Server bereitstellen:

- Erstellen Sie ein Quellverzeichnis auf dem gewünschten Netzlaufwerk.
- Kopieren Sie sämtliche Daten von der CD in das Quellverzeichnis.

Protokollierung der Installation

Die Netzwerkinstallation eines Benutzers wird in einer Datei auf dem Netzwerk protokolliert. Deshalb benötigen *alle* Benutzer Schreibrecht in dem `x:\user`-Verzeichnis bzw. in dem in der `install.ini` angegebenen Verzeichnis für die Registrierung.

2.3.2 Netzwerkinstallation anpassen

Sie haben die Möglichkeit, eine individuelle Anpassung von ASCET (Veränderung bestimmter Default-Einstellungen) vorzunehmen und zwar schon bevor der Anwender ASCET auf seinem Arbeitsplatzrechner installiert.

Bei der Netzwerkinstallation haben Sie folgende Anpassungsmöglichkeiten:

- Sie können die Installationsdialoge anpassen (Verändern der Default-Einstellungen z.B. für Verzeichnisse usw.).
- Sie können die ASCET Installation vollautomatisch und ohne weitere Benutzereingaben unsichtbar im Hintergrund durchführen.
- Sie können die mitgelieferten Dateien im Produktdatenverzeichnis (Default-Einstellung `[Laufwerk]:\ETASdata\ASCET5.2\...`) mit Ihren individuell angepassten Dateien überschreiben und/oder zusätzlich in die bestehenden Verzeichnisse Dateien einfügen.

Installationsdialoge anpassen

Bei der Netzwerkinstallation in großen Firmen ist es sinnvoll, dass gewisse Default-Einstellungen bei der Installation auf firmeninterne Wünsche angepasst werden können. Dies ist durch das Anpassen der Konfigurationsdatei `install.ini` möglich. Die Datei befindet sich im Installationsverzeichnis.

Wie Sie die Default-Einstellungen verändern, wird Ihnen anhand eines Beispiels aufgezeigt.

Konfigurationsdatei anpassen:

- Öffnen Sie die Datei `install.ini` mit einem Text-Editor.

  Ein typischer Eintrag in dieser INI-Datei ist z.B.:

```
;Sets the main directory of ASCET
```

```
;MainDir=c:\etas\Ascet
```

- Zum Verändern der Default-Einstellung entfernen Sie das „**;**“ (Kommentar) in der Zeile mit dem Schlüsselwort `MainDir`.
- Verändern Sie den Pfad in z.B.: `H:\programs\etas\Ascet5.2`.

  Der Eintrag sollte jetzt folgendermaßen aussehen:

```
;Sets the main directory of ASCET
MainDir=H:\programs\etas\Ascet5.2
```

- Verändern Sie auf diese Weise alle gewünschten Einträge in der `install.ini`.
- Speichern Sie Ihre Veränderungen und schließen Sie den Editor.

Wenn Sie jetzt die Installation mit `Ascet.exe` starten, werden in den Dialogboxen die eben vorgenommenen Einstellungen als Default-Werte angezeigt.

Hinweis

Die hier veränderten Pfadeinstellungen wirken sich auf die im folgenden beschriebenen Funktionen aus.

Installation automatisieren

Mit dem Aufruf von `Ascet.exe /s` haben Sie die Möglichkeit, die ASCET Installation vollautomatisch ohne weitere Benutzereingaben unsichtbar im Hintergrund durchzuführen. Zur Auswahl kommen die jeweiligen Standard-Voreinstellungen. Diese Einstellungen können in der Datei `install.ini` konfiguriert werden (siehe „Installationsdialoge anpassen“ auf Seite 31).

Wenn Sie als Systemadministrator eine Batchdatei mit dem Inhalt `ascet.exe /s` erstellen und die entsprechenden Einstellungen in der Datei `install.ini` vornehmen, können die Anwender mit dem Aufruf der Batchdatei die Installation ohne weitere Programmrückfragen komplett selbständig ablaufen lassen.

Da bei dieser Art der Installation keinerlei Dialogfenster erscheinen, sollten Sie ggf. einen Mechanismus bereitstellen, der den Anwender über die Fertigstellung der Installation informiert.

Bei der hier beschriebenen Anpassungsmöglichkeit können Sie die Installationsroutine soweit beeinflussen, dass bestimmte Standard-Dateien während der Installation mit den von Ihnen angepassten Dateien überschrieben bzw. andere Dateien in die Installation miteinbezogen werden.

Sie können so von Ihnen angepasste Datenbanken, Anwenderprofile und Fenstervorlagen in die Installationsroutine integrieren.

Der Mechanismus ist relativ einfach. Hierzu legen Sie in dem Installationsverzeichnis ein Verzeichnis mit dem Namen `InstData\...` an und legen dort die von Ihnen angepassten Dateien inklusive der richtigen Verzeichnisstruktur ab.

Zur Erstellung der individuell angepassten Dateien installieren Sie ASCET auf einem Testrechner und erstellen daraus Ihre angepassten Dateien.

Nach der Standardinstallation von ASCET sind in dem Verzeichnis `ETASData\Ascet5.2\...` unterschiedliche Verzeichnisse mit Dateien, die die ASCET Standardeinstellungen bestimmen und von Ihnen angepasst werden können. Dazu gehören die Verzeichnisse:

- `Database\DB\`
  Das Verzeichnis `DB` beinhaltet die Default-Datenbank. Hier können Sie beispielsweise eine weitere Demo-Datenbank unter `Database\DemoDB\` anlegen.
- `User\[Benutzername]`, abhängig von der Windows-Anmeldung
  Im Verzeichnis `[Benutzername]` befindet sich das Default-Anwenderprofil. Sämtliche einstellbaren Optionen sind hier abgelegt.

Daten für die Netzwerkinstallation anpassen:

- Installieren Sie ASCET auf Ihrem PC.
- Starten Sie ASCET.
- Verändern Sie das Anwenderprofil.
- Verändern Sie die Datenbank bzw. fügen Sie eine neue hinzu.
- Beenden Sie ASCET.

Sie haben nun die gewünschten Anpassungen vorgenommen und möchten diese Dateien in die Installationsroutine integrieren. Sie haben dabei zwei Möglichkeiten:

- Sie können gleichnamige Dateien durch die von Ihnen angepassten überschreiben. Hierzu müssen Sie einen Ordner mit dem Namen `InstData\overwrite\` im Installationsverzeichnis anlegen.

- Sie können von Ihnen angepasste und umbenannte Dateien hinzufügen. Gleichnamige Dateien werden nicht überschrieben. Hierzu müssen Sie einen Ordner mit dem Namen `InstData\add-only\` im Installationsverzeichnis anlegen.

Um nun die von Ihnen angepassten Dateien in die Installationsroutine aufzunehmen, müssen Sie auch die übergeordneten Verzeichnisse mitkopieren. Hierbei gilt, dass die Verzeichnisebene `ETASData\ASCET5.2\` die gleiche ist, wie `InstData\overwrite\` bzw. `InstData\add-only\`.

Hier zwei Beispiele:

```
InstData\overwrite\user\userDef.ini
InstData\add-only\database\additionalDB\
```

Verändertes Anwenderprofil integrieren:

- Kopieren Sie die von Ihnen angepasste Datei `ETASData\ASCET5.2\user\[Benutzername]\Ascetsd.ini` in das Verzeichnis `InstData\overwrite\user\`.
- Benennen Sie die Datei `ascetsd.ini` in `userDef.ini` um.

  Dadurch wird nach der Installation für jeden neuen Benutzer diese Initialisierungsdatei verwendet.

Veränderte Datenbank integrieren:

- Kopieren Sie die von Ihnen angepasste Datenbank, also das Verzeichnis `\Database\DB\`, in das Verzeichnis `InstData\add-only\...`
- Benennen Sie das Verzeichnis `DB` nach Ihren Wünschen um, da die Datenbank sonst nicht kopiert wird.

  Selbstverständlich könnten Sie auch die Datenbank `db` durch Verwendung des Verzeichnisses `InstData\overwrite\` überschreiben.

Wenn die Installationsroutine mit `ascet.exe` gestartet wird, werden die Standard-Dateien durch die von Ihnen angepassten ersetzt bzw. die neuen Dateien in die entsprechenden Verzeichnisse hinzugefügt.

2.3.3 ASCET vom Netzlaufwerk installieren

Die Installation vom Netzlaufwerk erfolgt wie unter „Installation“ auf Seite 18 beschrieben. Sie müssen lediglich wissen, auf welchem Laufwerk und in welchem Quellverzeichnis die Installationsdateien liegen.

Hinweis

Um ASCET vom Netzlaufwerk zu installieren, benötigen Sie Schreibzugriff auf das Protokoll-Verzeichnis im Netzlaufwerk (siehe Kapitel 2.3.1).

2.4 Deinstallation

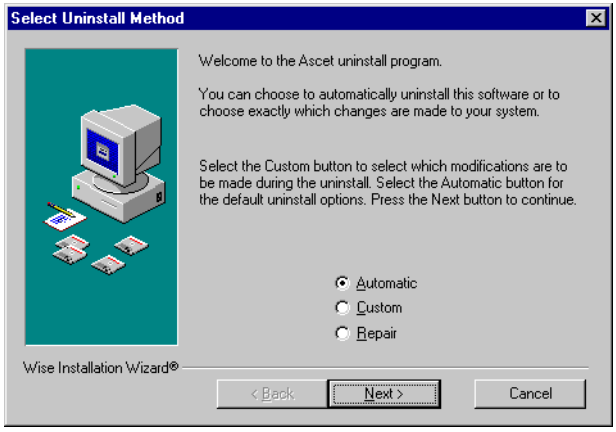
Wenn Sie ASCET deinstallieren, werden *alle* installierten Zusatzprogramme automatisch deinstalliert. Sie können nur die gesamte ASCET-Produktfamilie deinstallieren, nicht ASCET-MD, ASCET-SE oder ASCET-RP separat.

2.4.1 Automatische Deinstallation

ASCET deinstallieren (automatisch):

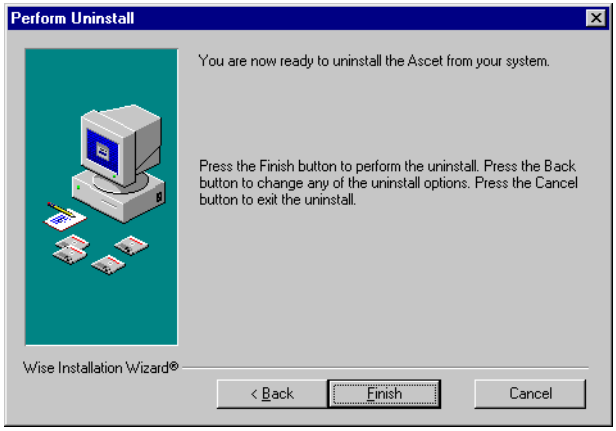
- Wählen Sie in der Programmgruppe im Startmenü den Eintrag **ASCET Uninstall**.

  Folgendes Fenster erscheint:

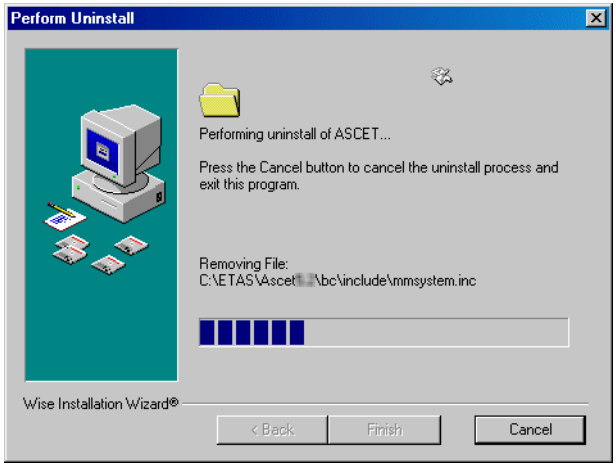


- Wählen Sie **Automatic**.

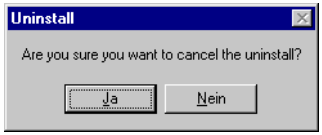
- Klicken Sie auf die Schaltfläche **Next**.



- Klicken Sie auf die Schaltfläche **Finish**, um die Deinstallation zu starten.



Sie haben die Möglichket, die Deinstallation abzubrechen. Wenn Sie auf die Schaltfläche **Cancel** klicken, erscheint folgendes Fenster:



Hinweis

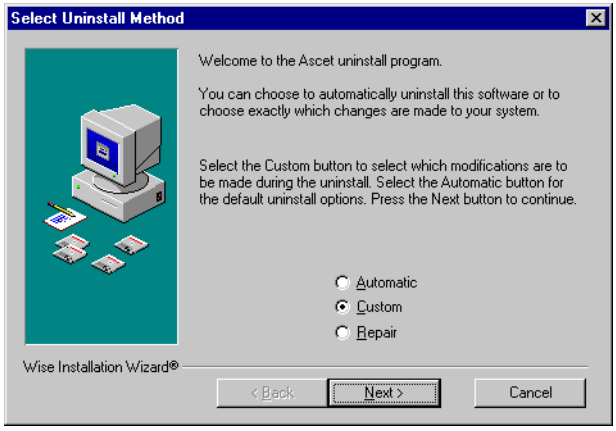
Sollten bereits Daten gelöscht sein, ist zum Arbeiten mit ASCET eine Neuinstallation erforderlich.

2.4.2 Benutzerdefinierte Deinstallation

ASCET deinstallieren (benutzerdefiniert):

- Wählen Sie in der Programmgruppe im Startmenü den Eintrag **ASCET Uninstall**.

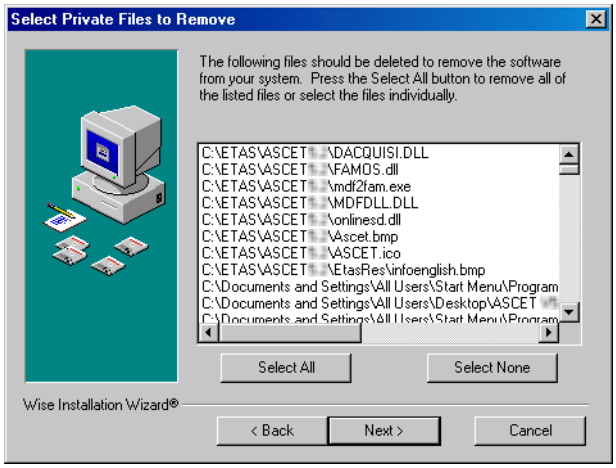
  Folgendes Fenster erscheint:



- Wählen Sie **Custom**.

- Klicken Sie auf die Schaltfläche **Next**.

  Das Fenster „Select Private Files to Remove“ öffnet sich.



- Wählen Sie im Fenster „Select Private Files to Remove“ die zu entfernenden Dateien.
- Klicken Sie auf die Schaltfläche **Next**.
- Wählen Sie im Fenster „Select Directories to Remove“ die zu entfernenden Verzeichnisse.
- Klicken Sie auf die Schaltfläche **Next**.
- Wählen Sie im Fenster „Select INI Files to Remove“ die zu entfernenden `*.ini`-Dateien.
- Klicken Sie auf die Schaltfläche **Next**.
- Wählen Sie im Fenster „Select INI Items to Edit“ die `*.ini`-Objekte, die bearbeitet werden sollen.
- Klicken Sie auf die Schaltfläche **Next**.
- Wählen Sie im Fenster „Select Registry Keys to Remove“ die zu entfernenden Registrierungsschlüssel.
- Klicken Sie auf die Schaltfläche **Next**.

- Wählen Sie im Fenster „Select Registry Trees to Remove“ die zu entfernenden Registrierungsordner.
- Klicken Sie auf die Schaltfläche **Next**.
- Wählen Sie im Fenster „Select Registry Keys to Edit“ die Registrierungsschlüssel, die bearbeitet werden sollen.
- Klicken Sie auf die Schaltfläche **Next**.
- Wählen Sie im Fenster „Select Sub-Systems to Remove“ die zu entfernenden Teilsysteme.
- Klicken Sie auf die Schaltfläche **Next**.
- Klicken Sie im Fenster „Perform Uninstall“ auf die Schaltfläche **Finish**.

  Die Deinstallation wird durchgeführt.

Wie bei der automatischen Deinstallation haben Sie die Möglichkeit, die Deinstallation mit der Schaltfläche **Cancel** abzubrechen.

Hinweis

Sollten bereits Daten gelöscht sein, ist zum Arbeiten mit ASCET eine Neuinstallation erforderlich.

3 Lizenzierung

Produkte und Zusatzprodukte der ASCET-Produktfamilie unterliegen der Lizenzverwaltung. Um nach der Installation mit einem ASCET-Produkt zu arbeiten, benötigen Sie eine Lizenzdatei für Ihren Rechner. Ohne diese Datei können ASCET-Produkte installiert, aber nicht verwendet werden.

3.1 Lizenzen anfordern

Sie bekommen die Lizenzdatei bei ETAS.

Lizenzdatei anfordern:

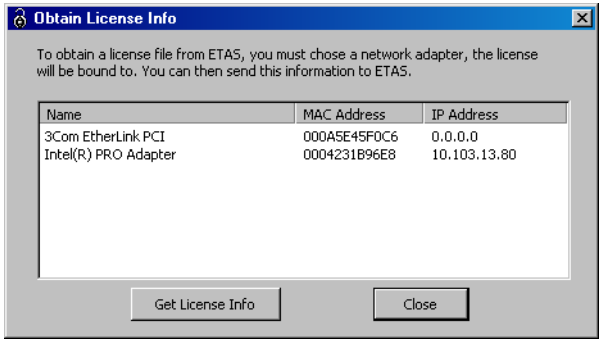
Die Lizenzdatei ist an den Benutzernamen und den Rechner gebunden. Um die Information für die Erstellung der Lizenzdatei zu ermitteln, verwenden Sie das Werkzeug **LicenseInfo** in der ASCET-Programmgruppe.

Hinweis

Wenn Sie die Lizenzdatei anfordern wollen, bevor Sie ein ASCET-Produkt oder -Zusatzprodukt installieren, können Sie das Programm `License-Info.exe` *von der CD-ROM des Produkts starten.*

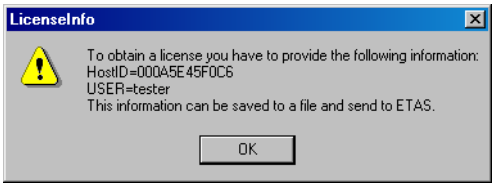
- Wählen Sie im Windows-Startmenü den ASCET-Ordner der aktuellen ASCET-Version.
- Wählen Sie im ASCET-Ordner **LicenseInfo**.

  Das Fenster „Obtain License Info" öffnet sich. Es enthält die MAC- und IP-Adressen für jede Netzwerkkarte Ihres Rechners.



- Wählen Sie einen der angegebenen Ethernet-Adapter und klicken Sie auf **Get License Info**.

  Die Informationen über Ihren Rechner, die ETAS für die Lizenzdatei benötigt, werden zusammengestellt und im Fenster „License info" angezeigt.



- Klicken Sie auf **OK**, um die Information in einer Textdatei zu speichern.
- Geben Sie im Dateiauswahlfenster Pfad und Namen für die Textdatei an.
- Klicken Sie auf **Speichern**.

  Die Datei wird angelegt und in einem Texteditor geöffnet. Neben Ihrer Host-ID und Ihrem Benutzernamen enthält die Datei Eingabezeilen für Ihre E-Mail-Adresse, Lizenznummer und weitere Angaben, dazu die ETAS-Kontaktadressen.
- Schließen Sie das Fenster „Obtain License Info".
- Schicken Sie die ausgefüllte Textdatei an ETAS.

  Die Lizenznummern entnehmen Sie Ihren Lizenzverträgen. Für jedes ASCET-Produkt, das Sie benutzen wollen, z.B. ASCET-MD oder ASCET-MIP, müssen Sie eine eigene vervollständigte Datei schicken.

  Sie bekommen von ETAS eine Lizenzdatei mit den erforderlichen Lizenzschlüsseln für Ihren Rechner.

Hinweis

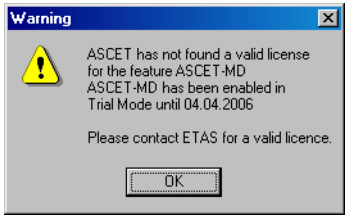
*Die Lizenzdatei darf **nicht** bearbeitet werden, sonst wird sie ungültig.*

- Legen Sie die Lizenzdatei in dem dafür vorgesehenen Lizenzverzeichnis ab (s. Seite 24).

  Standardmäßig ist dies das Verzeichnis `ETAS\ETASShared3\Licenses`. Es ist jedoch möglich, das Verzeichnis für die Lizenzdatei zu ändern, indem die Umgebungsvariable `ETAS_LICENSE_FILE` angepasst wird.

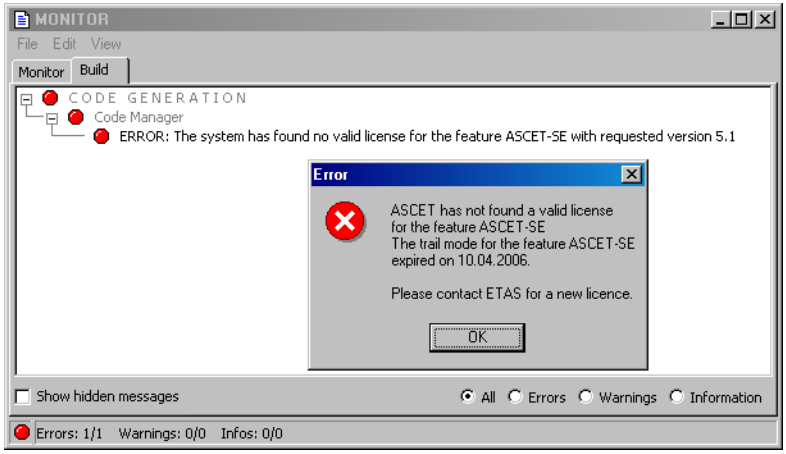
  In diesem Lizenzverzeichnis wird die Lizenzdatei beim Programmstart automatisch gefunden.

Wenn im Lizenzverzeichnis keine gültige Lizenzdatei gefunden wird, wird ein *Testmodus* für ASCET-MD, ASCET-RP oder ASCET-SE aktiviert. Das ist ein begrenzter Zeitraum, in dem das Produkt voll funktionsfähig ist, wo aber regelmäßig nach einer Lizenzdatei gesucht wird. Wenn die Suche fehlschlägt, wird eine Warnung ausgegeben.



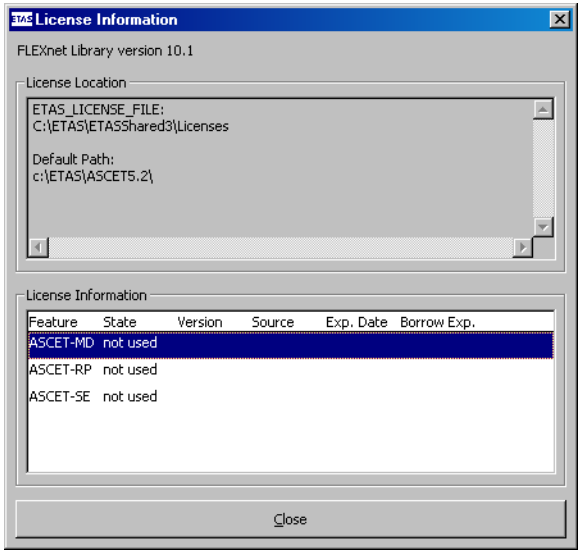
Zusatzprodukte wie ASCET-MIP oder INTECRIO-ASC haben *keinen* Testmodus. Ohne eine gültige Lizenzdatei funktionieren sie nicht.

Wenn der Testmodus abgelaufen ist, wird statt der Warnung eine Fehlermeldung ausgegeben; das Produkt funktioniert nicht, solange keine gültige Lizenzdatei gefunden wird.



3.2 Status der Lizensierung

Sie können jederzeit den aktuellen Status der Lizensierung anzeigen. Die Menüfunktion **Help → License Info** öffnet das Fenster „License Information".



Der obere Teil dieses Fensters enthält Informationen zum Ort der Lizenzen. Der untere Teil enthält eine Tabelle mit dem aktuellen Status der Lizensierung.

Spalte	**Beschreibung**
`Feature`	Installierte ASCET-Produkte oder -Zusatzprodukte.
`State`	Status der Lizensierung. Mögliche Werte: `not used` – für dieses Produkt/Zusatzprodukt wurde keine Lizenz angefordert. `licensed` – lizenziertes Produkt/Zusatzprodukt `grace mode` – Produkt im Testmodus `unlicensed` – Testmodus ist abgelaufen
`Version`	Versionsnummer des Produkts/Zusatzprodukts gemäß Lizenzdatei.

Spalte	Beschreibung
`Source`	Quelle der Lizenzdatei; mögliche Werte sind `local license` oder der Name eines Servers (oder ein leeres Feld, wenn keine Lizenz gefunden wurde).
`Exp. Date`	Ablaufdatum der Lizenz oder des Testmodus.
`Borrow Exp.`	Ablaufdatum der Ausleihe (siehe Kapitel 3.3).

Tab. 3-1 Beschreibung der Tabelle „License Information"

3.3 Lizenzen ausleihen

Wenn Sie einen Lizenzserver verwenden, aber eine lokale Lizenz benötigen (weil Sie z.B. ein Notebook im Fahrzeug verwenden wollen), können Sie die Lizenz vom Server für einen begrenzten Zeitraum *ausleihen*. (Sie brauchen die Lizenz nicht ausleihen, solange Sie mit dem Server verbunden sind.)

Die Ausleihdauer ändern:

Die voreingestellte Ausleihdauer ist 60 Tage; Sie können diesen Wert im ASCET-Optionsfenster, Register „General Settings", ändern.

- Wählen Sie im Komponentenmanager **Tools** → **Help**.

  Das Fenster „Options" öffnet sich.
- Öffnen Sie im Fenster „Options" den Knoten „Licensing".
- Geben Sie im Feld „Borrow license for days" die Ausleihdauer ein.
- Schließen Sie das Fenster „Options".

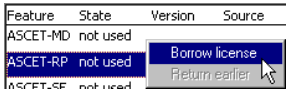
  Das nächste Mal, wenn Sie eine Lizenz ausleihen, wird das Ablaufdatum anhand dieses Wertes bestimmt.

  Existierende Ablaufdaten für Ausleihen werden nicht geändert.

Eine Lizenz ausleihen:

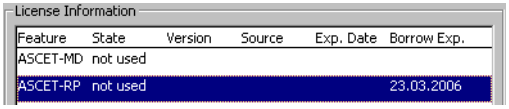
- Wählen Sie im Komponentenmanager **Help** → **License Info**, um das Fenster „License Information" zu öffnen.

  Sie können für alle Produkte/Zusatzprodukte im Zustand `not used` Lizenzen ausleihen.



- Klicken Sie mit der rechten Maustaste im Fenster „License Information" auf das Produkt/Zusatzprodukt, dessen Lizenz Sie ausleihen wollen, und wählen Sie **Borrow license** aus dem Kontextmenü.

  Damit haben SIe die Lizenz ausgeliehen. Das Ablaufdatum wird in die Tabelle „License Information" eingetragen.



  Die volle Funktionalität des gewählten Produkts/Zusatzprodukts ist nun lokal auf Ihrem Rechner vorhanden.

Solange Sie die Lizenz ausgeliehen haben, kann niemand anders sie verwenden. Um Engpässe zu vermeiden, können Sie die Lizenz auch vor Ablauf der Ausleihdauer zurückgeben.

Eine Lizenz zurückgeben (Normalfall):

Für die vorzeitige Rückgabe einer ausgeliehenen Lizenz muss Ihr PC mit dem Netzwerk verbunden sein.

- Öffnen Sie das Fenster „License Information".
- Klicken Sie mit der rechten Maustaste auf das Produkt/Zusatzprodukt, dessen Lizenz Sie zurückgeben wollen, und wählen Sie **Return earlier** aus dem Kontextmenü.

  Die Lizenz ist auf dem Server wieder verfügbar.

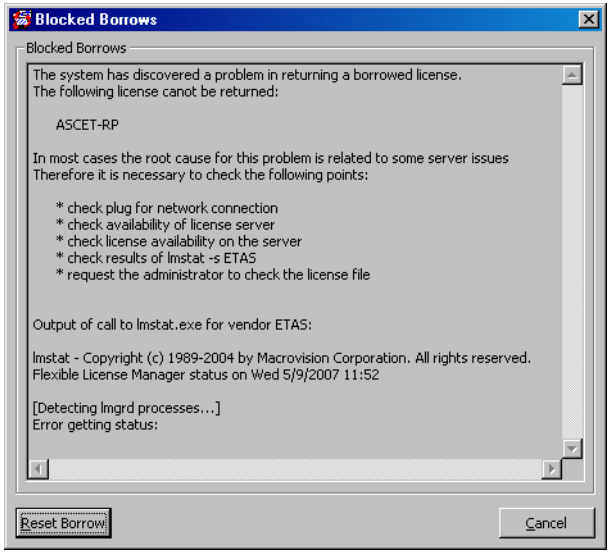
Eine Lizenz zurückgeben (Fehlerfall):

Wenn Sie eine Server-Lizenz ausgeliehen haben, sind sowohl auf dem Server als auch lokal auf Ihrem PC entsprechende Informationen abgelegt. Wenn die Information auf dem Server verlorengeht (das kann z.B. bei einem fehlerhaften oder komplett fehlgeschlagenen Neustart des Servers passieren), funktioniert die Rückgabe mit **Return earlier** nicht. Sie können den Befehl aber verwenden, um die lokalen Informationen auf Ihrem PC zu löschen.

- Öffnen Sie das Fenster „License Information".

- Klicken Sie mit der rechten Maustaste auf das Produkt/Zusatzprodukt, dessen Lizenz Sie zurückgeben wollen, und wählen Sie **Return earlier** aus dem Kontextmenü.

  Das Fenster „Blocked Borrows“ öffnet sich. Es enthält eine Liste von Punkten, die Sie überprüfen sollen.



- Prüfen Sie die genannten Punkte:
  - Ist Ihr Rechner am Netz angeschlossen?
  - Ist der Lizenzserver erreichbar?
  - Ist die Lizenz auf dem Server verfügbar?
  - Was für ein Ergebnis liefert `lmstat -s ETAS`?
  - Lassen Sie Ihren Administrator die Lizenzdatei auf dem Server prüfen.

- Nur wenn die auf dem Server gespeicherte Information über die von Ihnen ausgeliehene Lizenz tatsächlich verloren ist, klicken Sie auf **Reset Borrow**.

Hinweis

Wenn Sie die lokale Information zur ausgeliehenen Lizenz löschen, die Server-Information aber noch vorhanden ist, ist die Lizenz bis zum regulären Ablauf der Ausleihdauer ***nicht verfügbar****.*

Die lokal auf Ihrem Rechner gespeicherte Information über die ausgeliehene Lizenz wird entfernt.

4 ASCET – Grundlagen

ASCET ist ein qualitativ hochwertiges Werkzeug für schnelle Entwicklung, das die Wiederverwertung existierender Komponenten sicherstellt. Seine einzigartige grafische Entwicklungsumgebung erlaubt targetunabhängige funktionelle Spezifikationen. Steuersoftware für eingebettete Systeme kann automatisch aus Diagrammen generiert werden. Targetidentisches Prototyping erlaubt frühe Tests im Entwicklungszyklus. Die ETAS Toolkette macht Ihre Investition sicher und profitabel.

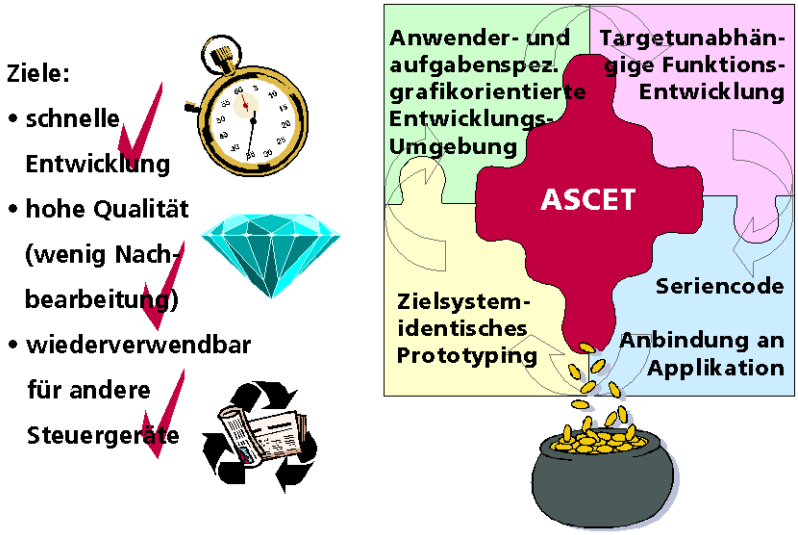


Abb. 4-1 Vorteile der Entwicklungsumgebung ASCET

4.1 Effizienzsteigerung in der Steuergeräteentwicklung

Elektronische Steuergeräte werden seit ca. 20 Jahren in verstärktem Maß zur Regelung im Kraftfahrzeug und anderen Branchen eingesetzt. Inzwischen sind die Anforderungen an Funktionalität, Geschwindigkeit und Vernetzung enorm gewachsen. Technologie und Methodik haben sich stark gewandelt. Heute ist die Elektronik ein Schlüsselfaktor für den Erfolg neuer Fahrzeugmodelle. Entwicklungskosten und vor allem Entwicklungsgeschwindigkeit gewinnen zunehmend an Bedeutung. Modernste Techniken schaffen hier Wettbewerbsvorteile. ETAS treibt diesen Bereich voran.

4.1.1 Moderne Embedded Control Systeme: technische Aufgaben

Die Integration vieler hochwertiger Funktionen auf einem Prozessorchip ist ein charakteristisches Merkmal heutiger Steuergerätetechnologie. Insbesondere sind die in der Praxis verwendeten Mikrocontroller mit vielfältigen I/O- und Kommunikationskanälen ausgestattet. Da auf diese Weise die Außenwelt direkt in den Prozessor eingebettet ist, spricht man hier von eingebetteten Systemen und bzgl. der entsprechenden Regelungstechnik von Embedded Control.

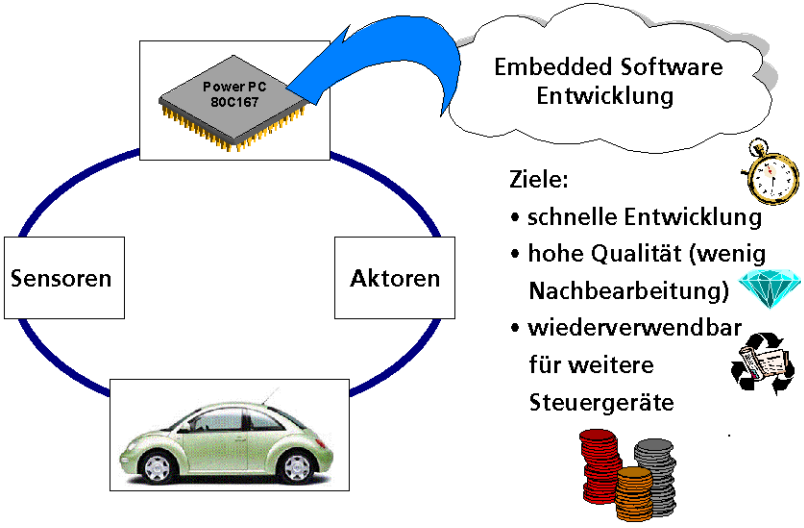


Abb. 4-2 Prinzip Embedded Control System

Embedded Control Systeme realisieren komplexe Regelkreise. Informationen über den aktuellen Zustand der Regelstrecke werden über Sensoren genau erfasst und mit früheren Daten gemeinsam verarbeitet. Ergebnis ist die Ausgabe von Steuerinformationen über die entsprechenden Aktoren. Die Berücksichtigung der Kopplung unterschiedlicher Systemaspekte wird immer wichtiger. Nur durch die Vernetzung der entsprechenden Steuergeräte lässt

sich heute das Gesamtsystem beherrschen, können aktuelle und zukünftige Umweltanforderungen an Sicherheit, Sauberkeit und Sparsamkeit umgesetzt werden.

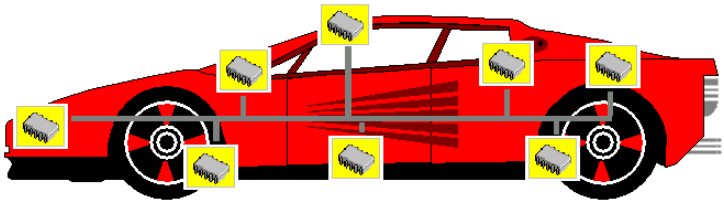


Abb. 4-3 Embedded Control Systeme in der Realität

Die Elektronik im Kraftfahrzeug hat bereits fast alle Bereiche des Systems erfasst. Motor- und Getriebesteuerung standen historisch am Anfang dieser Entwicklung. Bremseingriff und Fahrwerksregelung müssen inzwischen ebenfalls zu den Erfolgspositionen elektronischer Steuergeräte gerechnet werden. Deutlich sichtbar hat die Elektronik auch den Fahrzeuginnenraum erfasst (Airbag, Cockpit, Navigationssysteme etc.).

Embedded Control Systeme

Beim Entwurf von Embedded Control Systemen sind viele Details zu beachten. Meist müssen sehr viele Signale in die Regelung einbezogen werden. Darüber hinaus sind die rein physikalischen Zusammenhänge zwischen den Signalen äußerst komplex. Will der Ingenieur hier schnell zu Ergebnissen kommen, braucht er Methoden und Werkzeuge, die ihn von den zusätzlichen Schwierigkeiten bei der Implementierung der Regelung auf Mikroprozessoren entlasten. Durch die Abstraktion der Aufgabe und die weitgehende Verlagerung auf die physikalische Ebene muss es möglich sein, die Komplexität auf ein erträgliches Maß zu reduzieren, den Entwurf „einfach“ zu gestalten.

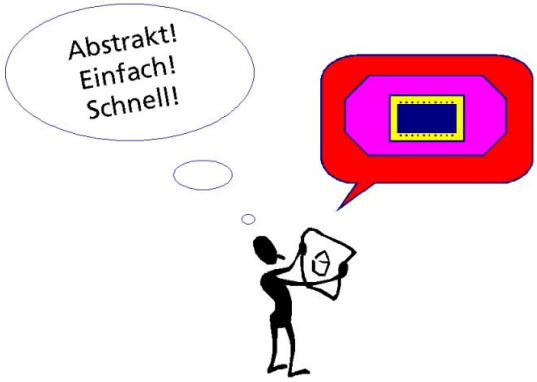


Abb. 4-4 Vision für Embedded Control Systeme

Die Vision für Embedded Control Entwicklungssysteme ist daher geprägt durch die Forderungen nach Abstraktion, Einfachheit und Schnelligkeit. ASCET erfüllt dies. Mit ASCET kann sich der Ingenieur voll auf die Physik des Embedded Control Systems konzentrieren, die Regelung abstrakt und einfach entwerfen, um anschließend automatisch Code für den Serieneinsatz zu erzeugen. Damit wird eine beispiellose Schnelligkeit im Entwurf erzielt.

Die Forderungen nach Abstraktion, Einfachheit und Schnelligkeit sind in den letzten Jahren in allen Teilen der Softwareentwicklung herausgestellt worden. Dabei sind vor allem visuelle Methoden für die Analyse- und Designphase entstanden. Diese verschiedenen Techniken sind nun in einer universellen Sprache UML (Universal Modeling Language) zusammengefasst. Auf dieser Ebene werden beispielsweise Entwurfsmuster (sogenannte Pattern) definiert. Sie übersetzen die Erfahrung der Ingenieure in die bislang abstrakteste Form. Allerdings bieten diese Ansätze derzeit keine Unterstützung für die Beschreibung von Echtzeitverhalten, und sie entsprechen in der Transparenz nicht unserem regelungstechnischen Ansatz. ASCET bietet all dies: eine angemessene Abstraktion, die klare Unterstützung von Echtzeitanforderungen sowie eine visuelle Beschreibung als regelungstechnische Blockschaltbilder und Zustandsautomaten. Und dort, wo es sich anbietet, können Spezifikationen von UML nach ASCET und zurück übertragen werden. Die **Offenheit** von ASCET eröffnet hier neue Perspektiven.

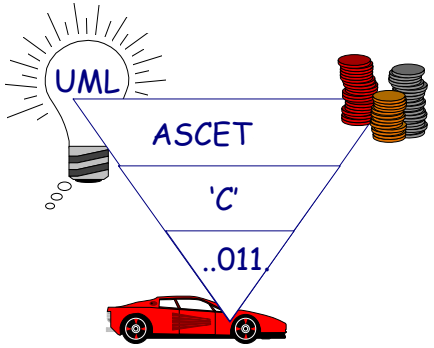


Abb. 4-5 Moderne Softwareentwicklung

Mit ASCET werden die Spezifikationen per Knopfdruck in C-Code übersetzt, der dann mit den jeweils targetspezifischen Werkzeugen (Compiler, Linker, Debugger) in ausführbaren Binärcode übersetzt und auf das Target geladen wird. Mit ASCET werden Entwicklungszeiten wirksam verkürzt und damit Kosten gespart. Darüber hinaus können bestehende Quellen (z.B. C-Code) wiederverwertet werden, so dass die Migration von einer reinen C-Entwicklung zu ASCET sehr gut unterstützt wird. Dabei wird die Abstraktion der Echtzeitanforderungen durch das **OSEK**-kompatible Betriebsystem ERCOS^{EK} voll erfüllt.

Mit der Erzeugung des ausführbaren Programms ist die Entwicklung von Embedded Control Systemen aber noch lange nicht beendet. In der Applikations- und Test- bzw. Erprobungsphase werden umfangreiche Messungen durchgeführt und Parameter optimiert. Hier müssen sich Entwicklungssysteme als anwendungsorientiert erweisen. Die eindeutige Forderung nach einer Durchgängigkeit im Entwicklungsprozess äußert sich dabei in der Unterstützung der entsprechenden Schnittstellen und Formate. ETAS erfüllt diese Wünsche mit ASCET und INCA-PC sowie LabCar voll und ganz. Eine aufeinander abgestimmte Palette von Produkten mit **standardisierten Schnittstellen** bietet dem Ingenieur einen beachtlichen Komfort, der es ihm gestattet, sein Hauptanliegen - die Entwicklung des Embedded Control Systems - in den Vordergrund zu rücken. Eine begleitende durchgängige Dokumentation ist hierbei ebenso selbstverständlich, wie die Offenheit der Werkzeuge für die Einbindung in eine projektübergreifende Verwaltung (Konfigurationsmanagement).

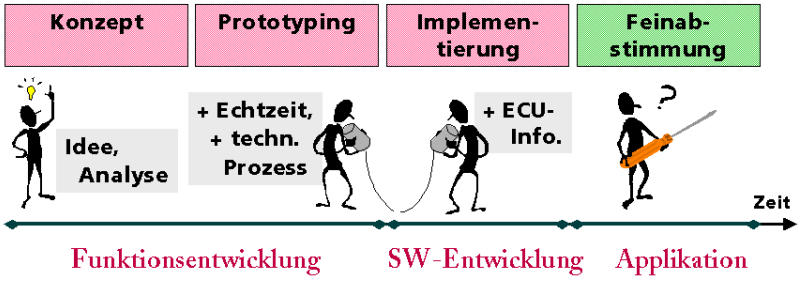


Abb. 4-6 Durchgängiger Prozess - Offene Schnittstellen

Ohne Durchgängigkeit im Prozess kommt es zu aufwendigen Konvertierungen von Daten. Mißverständnisse, Lücken in der Spezifikation oder gar Fehler sind die Folge. Darüber hinaus entstehen derartige Probleme bei jeder Änderung neu. In Summe entstehen auf diese Weise ungeheure Kosten, die durch eine durchgängige Werkzeugkette bereits im Ansatz vermieden werden. ASCET ist durchgängig und offen. Der Einsatz von ASCET spart Kosten.

Schwerpunkt Automobilindustrie

ASCET wurde schwerpunktmäßig für den Einsatz im Automobilbereich entwickelt. Daraus resultieren einige Anforderungen, die in dieser Form in anderen Branchen bislang nicht erkennbar sind. Andererseits sind viele allgemeine Konzepte in unsere Werkzeugkette eingeflossen, die auch für andere Entwicklungsaufgaben vorteilhaft genutzt werden können.

Charakteristisch für automobile Embedded Control Systeme sind die Forderungen nach einfachen regelungstechnischen Konzepten wie z.B. Kennfeldern. Hier äußert sich die Begrenztheit von Speicher und Performanz (Rechenzeit) besonders deutlich. Nur durch vereinfachte Modelle der idealen Regler können

die gewünschten sehr guten Resultate in Überwachung und Steuerung physikalischer Prozesse erzielt werden. Hinzu kommen eine Vielzahl an Forderungen, die jede für sich auch in anderen Anwendungsgebieten vorkommen, die in der Verknüpfung aber einzigartig sind: Qualität ist an dieser Stelle kein Allgemeinplatz, sondern allein unter Betrachtung der zu liefernden Stückzahlen des Endprodukts eine kritische Größe für Embedded Control Systeme. Dazu kommt der Sicherheitsaspekt. Viele der im Kraftfahrzeug realisierten Regelungen betreffen sicherheitsrelevante oder sogar sicherheitskritische Bereiche. Bremsen, Motorsteuerung aber auch Fensterheber (Gefahr des Einquetschens) sind charakteristische Beispiele. Das Thema Vernetzung wurde in diesem Zusammenhang bereits genannt.

Aber auch die zur Entwicklung verwendete Hardware muss besonderen Anforderungen genügen. Simulationssysteme, Mess- und Verstellsysteme wie auch Testsysteme müssen den rauen Gegebenheiten bei Sommer- und Wintererprobungen genügen. Sie müssen eine gute elektromagnetische Verträglichkeit aufweisen, und sie sollen selbstverständlich höchsten Leistungsansprüchen genügen.

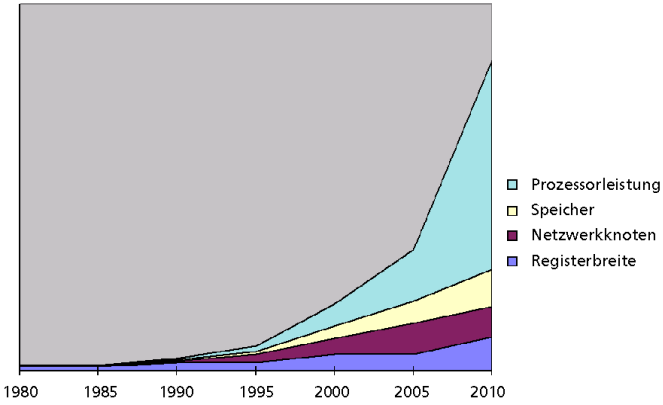


Abb. 4-7 Anforderungen in der Automobilindustrie

Die Innovationszyklen in der Automobilindustrie sind hauptsächlich durch die entsprechenden Erprobungen bestimmt. Die Produkte selbst haben aber eine deutlich längere Lebensdauer und sie werden in diesem Zeitraum auch gepflegt. Verbesserungen in der Mikroprozessortechnologie werden daher mit langfristigem Blickwinkel betrachtet oder müssen ohne Einfluss auf die Entwicklungsaktivtäten adaptierbar sein (z.B. verbesserte Speichertechnologie). Es wird daher von einem relativ groben Modell ausgegangen, in dem z.B. die Registerbreite, als ein Maß für die Prozessorleistung, sich nur alle fünf Jahre verdoppelt. Dabei ist zu berücksichtigen, dass es starke Überlappungen gibt, d.h. wenn die ersten 32 Bit Prozessoren eingesetzt werden, gibt es noch viele

Projekte, die mit 16-Bit Technik und einige, die in 8-Bit durchgeführt werden. Entsprechendes gilt z.B. für den anstehenden Innovationsschritt von reiner Integer-Arithmetik zu Floating Point.

4.1.2 Entwicklungsprozesse: wirtschaftliche Herausforderungen

Entwicklungsprozesse sind immer zu einem Teil firmenspezifisch. Es gibt aber eine Menge allgemeiner Anforderungen, die ihren Niederschlag nicht zuletzt in den unterstützenden Werkzeugen finden müssen.

In allen modernen Entwicklungen spielt das strategische Quadrat von Zeit, Kosten, Qualität und Flexibilität die entscheidende Rolle. Entwicklungsprozesse müssen sich an diesen Faktoren messen bzw. ausrichten. Die durchgängige Werkzeugkette leistet hier Wesentliches. Das geschieht aber nur, wenn auch jedes Glied in der Kette den entsprechenden Beitrag liefert. ASCET und die übrigen ETAS-Werkzeuge bieten diese Leistungen. Sie helfen Zeit zu sparen durch ihre aufeinander abgestimmten Schnittstellen und die einzigartige Ausrichtung auf die technischen Anforderungen von Embedded Control Systemen. Dies ist gleichzeitig auch ein wichtiger Beitrag zur Kostenvermeidung. Darüber hinaus werden Kosten durch die übersichtliche regelungstechnisch motivierte Modellierung gespart. Zusammen mit einem frühzeitigen Prototyping werden so überflüssige Rekursionen und Fehlentwicklungen vermieden. Dies verbessert zugleich die Qualität der Entwicklungsergebnisse. Zugleich steigt die Qualität durch die automatische Codegenerierung von ASCET; die sonst übliche Implementierungslücke ist geschlossen. Spezifikation und Produkt stimmen 100% überein. Aufwendige Nacharbeit entfällt. Zusammen mit dem einzigartigen Datenbankansatz von ASCET steigt auch die Flexibilität in der Entwicklung. Module können beliebig kombiniert werden, die notwendigen Optimierungen und Anpassungen übernimmt die automatische Codegenerierung. Zusätzlich bietet ASCET einfach anzupassende Schnittstellen, die es erlauben, praktisch jeden Entwicklungsprozess flexibel zu unterstützen.

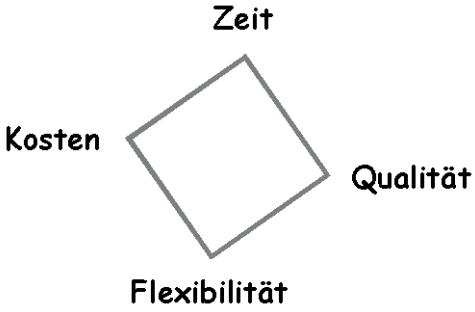


Abb. 4-8 Strategisches Quadrat - Anforderungen an die Produktentwicklung

Die besondere wirtschaftliche Herausforderung in der Automobilindustrie ist die Beherrschung von Großserien. Dies hat vor allem Folgen für die Fertigungsprozesse. Aus Entwicklungssicht ergeben sich zusätzlich weitere Aspekte, wie z.B. die Zusammenarbeit mehrerer Zulieferer an einem Projekt. Damit ergeben sich Anforderungen an die Werkzeugkette, die unter dem Begriff Teamunterstützung zusammengefasst werden. Es muss möglich sein, Daten gemeinsam zu verwalten und auch über große Entfernungen sicher und nachvollziehbar auszutauschen. Dabei darf das Thema Know-how Schutz nicht vernachlässigt werden. ASCET bietet hierzu die richtigen Konzepte und die notwendigen Schnittstellen zu den entsprechenden Werkzeugen.

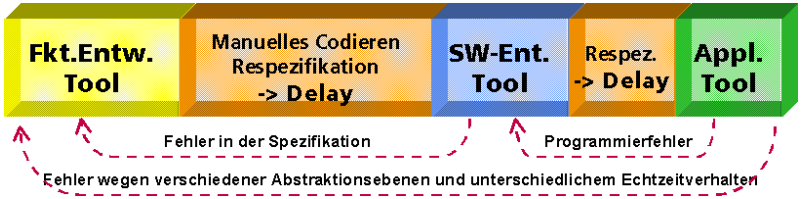


Abb. 4-9 Herkömmlicher Entwicklungsprozess

In engen Märkten ist der Kosten- und Innovationsdruck gleichermaßen permanent. Hier gilt es Synergien über Projekte hinweg zu erzielen. Dies kann nur durch ein übergreifend modulares Vorgehen (Gleichteilekonzept) geschehen. Für die Entwicklung von Embedded Control Systemen bedeutet dies die Einführung möglichst standardisierter Schnittstellen auf beliebigen Ebenen (Hardware, Betriebssystem, Protokolle, Funktionen). Auf diese Weise wird die notwendige Verkürzung der Entwicklungszyklen erreicht. Standards sind Programm bei ETAS: Unsere Werkzeuge unterstützen oder setzen die Standards im Bereich Embedded Control Systeme (z.B. ASAM, ASAP, CAN, MSR, NEXUS, OSEK, VME). Bei neu aufkommenden Standardisierungsforderungen arbeiten wir aktiv mit.

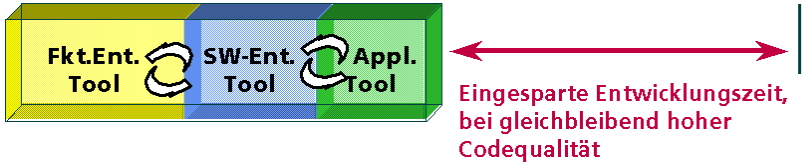


Abb. 4-10 Optimierter Entwicklungsprozess mit ASCET

Die heutigen wirtschaftlichen Anforderungen kann nur erfüllen, wer sowohl Effizienz als auch Effektivität in der Entwicklung realisiert. Das Richtige richtig zu tun, ist Leitlinie im Kleinen wie im Großen. Jeder Tag mit ASCET bringt effektiv mehr Effizienz in die Entwicklung moderner Embedded Control Systeme. Durch die beschriebenen Vereinfachungen im Entwicklungsalltag wer-

den die Potentiale freigesetzt, die nötig sind, damit sich der Ingenieur auf seine ureigensten Aufgabe konzentrieren kann – **die Erfindung und Umsetzung von neuen innovativen Regelungen und Steuerungen.**

4.1.3 Innovative Technologien - technologische Visionen

Seit mehreren Jahrzehnten wird Software für Embedded Control Systeme produziert. An den verwendeten Programmiersprachen und Werkzeugen hat sich über die Jahre einiges geändert. Die grundsätzliche Methodik ist jedoch lange unverändert geblieben. Nur Experten für Mikrocontroller konnten bislang die komplexe Programmierung dieser Systeme durchführen. Die Möglichkeiten zur Fehlersuche mittels eines Debuggers waren bzw. sind immer noch sehr aufwendig und zum Teil auch nur eingeschränkt möglich (Echtzeitproblematik). Funktionsentwickler haben so kaum eine Chance, direkt zu Ergebnissen zu kommen. Inzwischen hat ETAS diesen Bereich revolutioniert: Mit der Spitzentechnologie von ASCET ist es erstmals möglich, einen Regelungsentwurf sofort am Zielsystem zu verifizieren.

Zukunftsorientierung und Innovationskraft entsprechen unserem Charakter. Sie sind Triebkraft für Fortschritt und Verbesserungen. Unsere Produkte entsprechen diesem Leitbild. Wir setzen damit Standards. Dabei orientieren wir uns vollständig an den Wünschen unserer Kunden und optimieren unsere Produkte entsprechend. Entwickeln Sie mit uns die Zukunft der Embedded Control Systeme, damit der Fortschritt Sie nicht überholt.

Codegenerierung

Die Zeit der Bitkodierung ist vorbei. Auch Assembler trifft man immer seltener als ernsthafte Implementierungssprache für Embedded Control Systeme an. An vielen Orten wird heute in C entwickelt. Diese sogenannte Hochsprache markiert den Endpunkt einer Entwicklung, die stets auf das Zielsystem zentriert war. Damit können keine plattformübergreifenden Regelungskonzepte realisiert werden. Die Wiederverwertung von physikalisch identischen Ansätzen wird so zum Horrorszenario. Man beginnt das Gleiche stets von vorn. Sisyphus lässt grüßen.

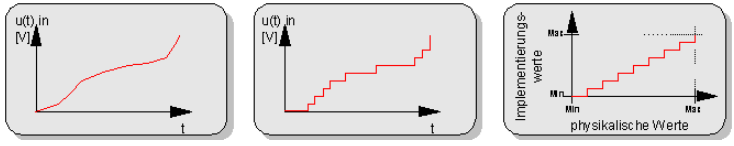


Abb. 4-11 Von physikalischen Größen zu Implementierungen

ASCET bietet hier etwas Unvergleichliches. Erstmals ist es möglich, direkt und automatisch vom grafischen regelungstechnischen Blockschaltbild zur Software auf dem Mikrocontroller zu kommen. Die ASCET Codegenerierung ist

einmalig, effizient und einfach einzusetzen. Dabei war die Aufgabe für die Entwickler bei ETAS alles andere als trivial. Die Schwierigkeit liegt zunächst in der richtigen Abstraktion der Anforderungen: Mikrocontroller arbeiten heute zumeist mit Integer-Arithmetik, sie haben begrenzten Speicher und unterschiedlichste Hardwarearchitekturen. Durch die Verfügbarkeit von ASCET-SE für eine Vielzahl von Targets ist es gelungen, diese Unterschiede zu kapseln und damit zu einer einheitlichen Schnittstelle für den Anwender zu kommen. Das Resultat: Der Wechsel von einem Mikrocontroller zu dem eines anderen Herstellers oder zu einem moderneren ist problemlos möglich. Wiederverwertung im Großen ist damit nicht mehr nur ein Schlagwort sondern praktische Realität des Ingenieuralltags. Und damit das so bleibt, setzen wir alle wichtigen neuen Prozessorentwicklungen in entsprechende ASCET-SE Portierungen um.

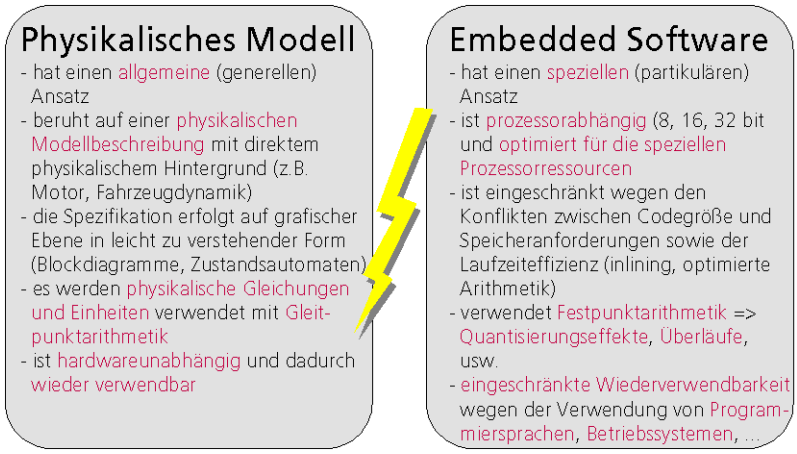


Abb. 4-12 Konträre Anforderungen von physikalischer Modellierung und Implementierung eingebetteter Software

ASCET-SE erzeugt C-Code. Es verwendet die üblichen Werkzeuge zur Mikrocontroller-Programmierung (Compiler, Linker, Locator und Debugger). Hier stellen sich neue Herausforderungen insbesondere in der Konfiguration und Handhabung dieser Werkzeuge. Hier wurden ebenfalls durch ASCET Meilensteine gesetzt. Der Fortschritt ist an dieser Stelle aber noch nicht beendet. Durch innovative Techniken wird zum Beispiel das Thema Debugging mit dem der Applikation elektronischer Steuergeräte (Messen und Verstellen) verbunden. Damit ist es möglich, die komplette Kontrolle über das Echtzeitverhalten auf dem Zielsystem zu erlangen. Erstmals kann auf der Ebene der physikalischen Regelungstechnik direkt am Zielsystem gearbeitet werden. Der Vorteil liegt auf der Hand: das permanente gedankliche Übersetzen von der Ebene der Regelung auf die Bitebene des Prozessors entfällt.

Simulationstechnik hat sich inzwischen als fester Bestandteil der Entwicklung komplexer Regelsysteme etabliert. Auf diese Weise werden neue Funktionen frühzeitig abgesichert. Meist wird offline zeitschrittbasiert simuliert. Der Einfluss der Echtzeitanforderungen auf die Regelung bleibt dabei unberücksichtigt. Entsprechend müssen auch Sensoren und Aktoren sowie die entsprechenden dahinter liegenden physikalischen Systeme, die sogenannten Streckenmodelle, für die Simulation nachgebildet werden. Es handelt sich dabei also um eine reine Softwarelösung. Daher die Bezeichnung Software in the Loop (SIL). Damit kann nur der erste Schritt in der Entwicklung innovativer Regelungen bewältigt werden, nämlich die Konzeption der Regelung. Bei der ereignisorientierten Online-Simulation hingegen werden Sensoren und Aktoren und die reale Strecke in Echtzeit einbezogen. Das Betriebssystem wird bereits an dieser Stelle erstmals integriert. So entstehen aus einer Simulation mit Hardware in the Loop (HIL) die ersten Softwareprototypen. Der Vorteil hierbei liegt in der realistischeren Nachbildung der regelungstechnischen Anforderungen. Neue Regelungsstrategien können so in der Zielumgebung praxisnah erprobt und angepasst werden. Dabei lassen sich zwei Fälle unterscheiden: In einer frühen Phase des Prototyping wird der Regelalgorithmus noch auf Basis physikalischer Größen behandelt. Quantisierung und Überläufe bleiben dabei weitgehend unberücksichtigt. Sie müssen nur an den Systemgrenzen (Sensoren, Aktoren) berücksichtigt werden. In der anschließenden Phase fließen diese Implementierungsaspekte mit ein. Anschließend ist es ein kleiner Schritt vom Prototypen zu Produkt, der nur noch in der Änderung der Rechnerhardware besteht. Auf diese Weise wird der Weg von der Idee bis zum Produkt schrittweise durchgeführt, so dass der Entwickler zu jedem Zeitpunkt die volle Kontrolle über das komplexe Gesamtsystem besitzt und sich jeweils nur um einen Aspekt der Gesamtaufgabe kümmern muss.

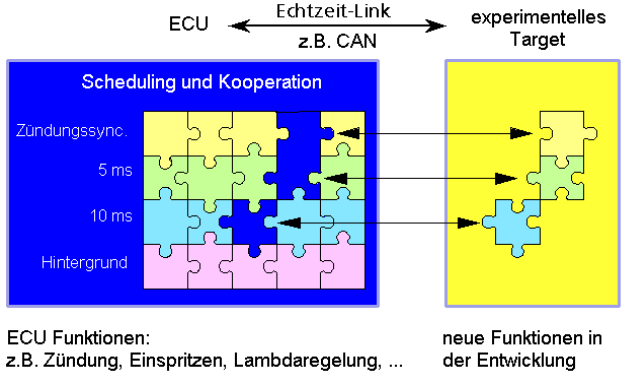


Abb. 4-13 Bypass Technik - Grundidee

Prototyping kann auch mit Hilfe eines bereits vorliegenden Steuergeräts durchgeführt werden. Dabei muss unterschieden werden zwischen der Situation, in der nur ein Teil der Gesamtregelung simuliert wird und der Rest auf einem Seriensteuergerät läuft (Bypass) und dem Fall, bei dem nur die Signalkonditionierung eines Seriensteuergeräts verwendet wird (Fullpass). Darüber hinaus besteht die Möglichkeit, ein bestehendes, aber zum Zielsystem unterschiedliches Steuergerät komplett mit einem neuen Programm zu versorgen (Entwicklungssteuergerät). All diesen Verfahren ist gemein, dass es sich um ein targetidentisches Prototyping handelt. Schon während des Prototyping kann damit das Funktionsverhalten in der erforderlichen Genauigkeit studiert werden. Die Einflüsse von Quantisierung, Überläufen und Echtzeiteffekten werden so frühzeitig beherrschbar. Voraussetzung für diese Methodik ist Äquivalenz der Simulation mit der Ausführung der Software auf dem Zielsystem d.h. dem Seriensteuergerät. ASCET bietet diese besondere Voraussetzung. Bei ASCET basiert sowohl Online-Simulation als auch Seriencode auf unserem Echtzeitbetriebssystem ERCOS^{EK} und auch die Quantisierung wird in der Online-Simulation genauso behandelt, wie im Seriensteuergerät.

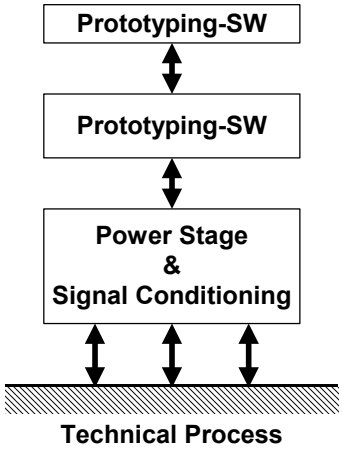


Abb. 4-14 Prinzip Prototyping

Ein weiterer wichtiger Aspekt beim Thema Prototyping ist die Behandlung der Umgebung. Wie bereits oben erwähnt ist es sehr nützlich, auch die hinter den Sensoren und Aktoren liegenden physikalischen Systeme nachbilden zu können. Hier handelt es sich im Gegensatz zur diskreten Modellierung der Steuergerätefunktionen um zeitkontinuierliche Systeme. Diese werden in der Regel durch Differentialgleichungssysteme dargestellt. Für die Echtzeitbehandlung stehen dabei allerdings nur besonders einfache Lösungsalgorithmen zur Wahl, da beispielsweise eine Schrittweitensteuerung für derartige Anwendungen eher nicht geeignet ist. ASCET bietet diese Methoden in einem Werkzeug, d.h.

mit identischen Schnittstellen und Bedienabläufen, also aus einem Guss. Damit können Streckenmodelle homogen in die Regelungsentwicklung integriert werden, wenn z.B. nicht alle physikalischen Komponenten gleichzeitig für das Prototyping zur Verfügung stehen. Im Extremfall kann so ein ganzes Fahrzeug in die Online-Simulation integriert werden (LabCar). Die Integration kontinuierlicher Systeme in ASCET schafft Vorteile in der täglichen Arbeit. Das Bereitstellen und Pflegen spezieller Schnittstellen bzw. das jeweils notwendige Umdenken in eine andere Werkzeugumgebung entfällt so.

4.2 Durchgängige Unterstützung für Embedded Control Systeme

Ein wesentlicher Faktor für die Effizienzsteigerung liegt in der Durchgängigkeit der Toolkette. Ein Bruch in diesem Umfeld führt unweigerlich zu Störungen im Entwicklungsablauf, zu kostentreibenden Rekursionen oder zu manueller Nachbearbeitung. Dies kann durch die entsprechenden Schnittstellen und Formate vermieden werden. Dabei sind insbesondere unter dem Gesichtspunkt der Migration zu neuen Systemen Standards gefragt, d.h. die Schnittstellen und Formate müssen von möglichst vielen Werkzeugherstellern unterstützt werden. ETAS ist hier aktiv am Setzen und Verbreiten von Standards beteiligt und bietet in allen Werkzeugen die nötigen Schnittstellen und Formate. Allerdings ist ETAS zugleich der einzige Anbieter, der dem Kunden die komplette Werkzeugkette aus einer Hand liefern kann. Durchgängigkeit ist also für uns nicht nur Schlagwort, sondern bewährte Praxis.

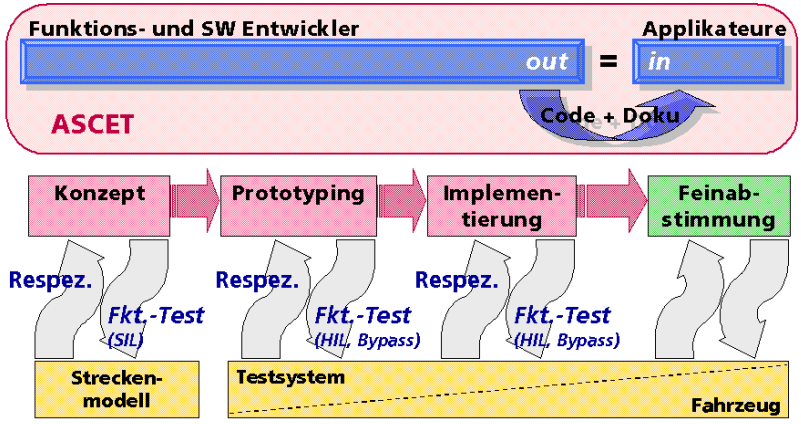


Abb. 4-15 Durchgängiger Entwicklungsprozess

4.2.1 Einstiegstechnologie Bypass

Durch die eindeutige, ausführbare Spezifikation von Funktionen im Bypass werden neue Techniken und Algorithmen effizient eingeführt. Da man den Großteil der Steuer- und Regelfunktionen aus dem jeweiligen Serienstand verwendet, besteht immer ein fester Stand, auf dem man aufbauen kann. Daher eignet sich dieses Vorgehen auch hervorragend für die Einführung innovativer Modellierungs- und Simulationstechniken wie ASCET. Umgekehrt ist ASCET auch besonders auf diese Situation vorbereitet. Mit dem Zusatzprodukt ASCET-RP steht eine Arbeitsumgebung zur Verfügung, die den Einstieg leicht macht und schnell zu verwertbaren, aussagekräftigen Resultaten führt.

Voraussetzung für die Anbindung des Bypasses ist der sogenannte Freischnitt der entsprechenden Funktion im Steuergerät, d.h. in der Seriensoftware bzw. seriennahen Software. Dieser muss vom jeweiligen Hersteller der Software bereitgestellt werden. Im Freischnitt kann der übliche Funktionsaufruf auf die Funktionsanforderung auf dem Bypassrechner umgebogen werden. Dazu wird ein Softwareschalter gesetzt. Falls der Bypass diese Anforderung nicht hinreichend schnell beantwortet (Sicherheit), wird auf ein Notprogramm geschaltet oder es werden die Werte aus der parallel berechneten Serienfunktion verwendet. Häufig werden solche Freischnitte für die Zusammenarbeit zwischen Zulieferern und Herstellern bereitgestellt.

Manchmal ist es nützlich, zusätzlich auf neue bzw. andere Daten der bestehenden Seriensoftware zuzugreifen. Dazu müssen dann nur die zugehörigen Adressen im Steuergerät bekannt sein, die üblicherweise auch für Applikationszwecke in Form einer Steuergerätebeschreibungsdatei (ASAM-MCD-2MC) bereitgestellt werden. Es ist ebenfalls möglich, gleichzeitig mit einem Applikationssystem am Seriensteuergerät Daten zu messen und verstellen, falls dies aufgrund der Wechselwirkung der neuen Funktion mit dem alten Stand nötig sein sollte.

In ASCET sind die nötigen Voraussetzungen zur Anbindung von Serienständen, die Schnittstellen zum Steuergerät, und auch die zusätzlich häufig benötigten Schnittstellen zu Aktoren und Sensoren vorhanden. Die gemeinsame Anwendung mit unserem Applikationssystem INCA-PC ist erprobt und hat sich bewährt.

4.2.2 Prototyping

Ein zweites Einsatzgebiet von ASCET ist der Bereich des Prototyping. Hier geht es in erster Linie darum, neue Ideen von Anfang an neu zu entwickeln. In diesem Fall kann häufig nicht auf einen Serienstand aufgesetzt werden. Prototyping erstreckt sich von der einfachen funktionalen physikalischen Simulation über das Studium von Quantisierungseffekten und Rechenzeitabschätzungen bis hin zur kompletten Umsetzung auf einem Steuergerät. Und dabei kann es

auf jeder Ebene wichtig bzw. nötig sein, die Umgebung des Steuergeräts in Form der Sensoren und Aktoren vollständig einzubeziehen. Letzteres geschieht entweder in Form einer Streckensimulation oder durch Einbeziehen realer Hardware im geschlossenen Regelkreis. Auf diese Weise kann die Machbarkeit einer regelungstechnischen Idee abgesichert werden, und es wird ein besseres Verständnis der Anforderungen erzielt.

ASCET stellt für diese Anwendungen die nötige Rechenleistung in Form universeller Prozessorboards zur Verfügung und liefert auf Basis unseres VME-Bus Systems ES1000 und entsprechender Einschubkarten auch die Schnittstellen zur Außenwelt.

An dieser Stelle aber endet die Arbeit häufig gar nicht. Durch die Öffnung unserer Simulationsumgebung für Applikationssysteme wie z.B. INCA-PC gelingt es, auch kleine Flotten mit diesen Prototypen auszurüsten und kostengünstige Lösungen für den gesamten Entwicklungsprozess bis zur Vorserienreife einfach darzustellen. Unsere einmalige Codegenerierungstechnologie geht noch weiter: durch Portierung weniger Implementierungsinformationen auf ein Mikroprozessorzielsystem und die automatische Codegenerierung ist man sehr schnell serienreif. Falls dies unter harten Bedingungen vorab getestet werden soll, kann das ETAS Entwicklungssteuergerät ES400 mit ASCET eingesetzt werden.

4.2.3 Automatische Codegenerierung

Die Abbildung eines funktionalen Modells auf ausführbaren Programmcode ist die wesentliche Herausforderung in der Entwicklung von Embedded Control Systemen. Die Ziele auf der Ebene physikalischer Modellierung und auf der Ebene des Steuergeräts könnten kaum unterschiedlicher sein. Von der physikalischen Modellierung wird erwartet, sie sei grafisch, hardwareunabhängig, wiederverwertbar und unterstütze physikalische Datentypen (Gleitkommaarithmetik). Dabei sollte die physikalische Modellierung durch die Verwendung von Blockdiagrammen und Zustandsautomaten leicht verständlich sein, damit eine entsprechende Dokumentation während der Applikationsphase als Referenz eingesetzt werden kann. Auf der Ebene des Steuergeräts, d.h. der Embedded Software Implementierung sieht das ganz anders aus. Hier wird eine speziell auf den vorliegenden Mikroprozessor und die entsprechende Speicherausstattung hin optimierte Codierung erwartet. Codegröße und Laufzeit stehen im Vordergrund. Die Datentypen sind in der Regel ganzzahlig (Festpunktarithmetik).

Will man die fehlerträchtige, aufwendige Handcodierung vermeiden, so gelingt die Verbindung der beiden Welten nur durch eine automatische Codegenerierung. Diese basiert auf einer Aufteilung der Spezifikation in einen Anteil physikalischer Modellierung und die zugehörigen Implementierungsinformationen. Dabei bietet ASCET eine schrittweise Migration von reinen C-

Implementierungen zu den neuen physikalischen Beschreibungsmitteln. Darüber hinaus besteht die Möglichkeit der Wiederverwertung von generiertem C-Code in anderen Entwicklungsprojekten.

Dieser Ansatz setzt sich durchgängig in der Experimentierumgebung fort, so dass beispielsweise bereits früh Quantisierungseffekte oder Laufzeitprobleme untersucht werden können. Da sowohl Steuergeräteprogramm als auch Experimentierumgebung auf dem gleichen Mechanismus beruhen, spricht man hier besser vom Ausführen der Spezifikation auf unterschiedlichen Zielsystemen als von Simulation. Diesen Gedanken konsequent weitergedacht, landet man bei der Frage, ob man dann nicht auch auf dem Experimentiersystem einfach applizieren kann. Diese Frage kann für die ETAS-Werkzeuge (Hardware und Software) mit einem klaren Ja beantwortet werden. Die Entwicklung ist damit von Anfang bis Ende voll durchgängig.

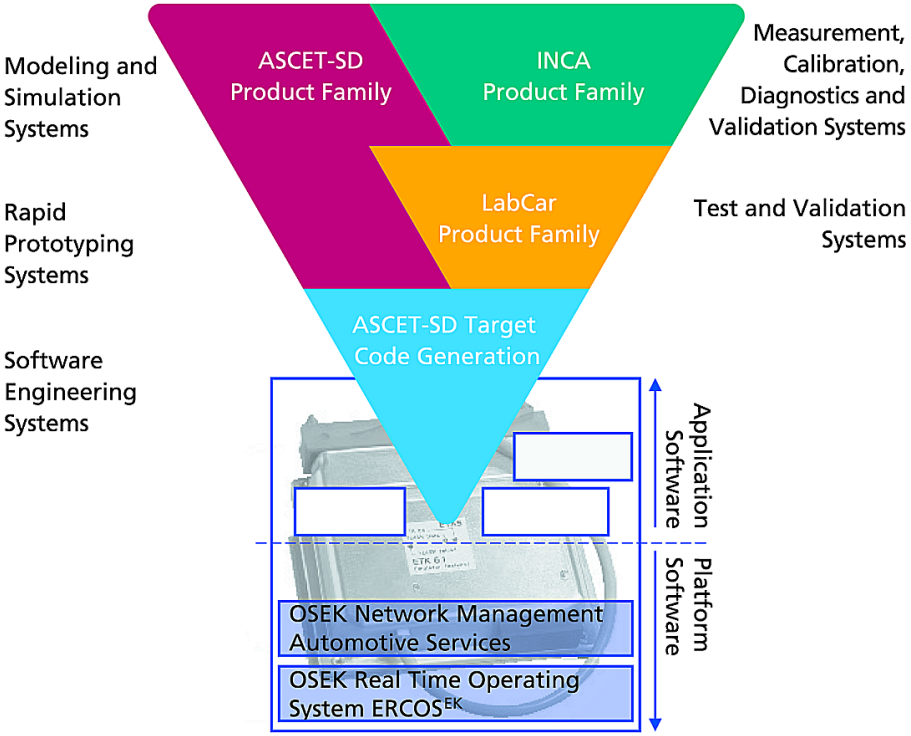


Abb. 4-16 Werkzeugeinsatz in der Entwicklung von Embedded Control Systemen

4.2.4 Weitere Einsatzoptionen der ETAS-Entwicklungswerkzeuge

Die einmalige ASCET Codegenerierung für Seriensteuergeräte kann auf zwei unterschiedliche Arten eingesetzt werden, nämlich im Sinne eines zusätzlichen Programmierers und als Integrationswerkzeug. Erstere kann wiederum sehr gut als Einstiegsszenario in die neue Technologie verstanden werden, insbesondere bei Großprojekten. ASCET als Integrationswerkzeug kann in allen Projekten sehr vorteilhaft eingesetzt werden. Bei kleineren Projekten kann auf diese Weise auch ein Neuling sehr schnell zu guten Ergebnissen kommen.

ASCET als zusätzlicher Programmierer

Beim Einsatz von ASCET in diesem Anwendungsszenario wird ein herkömmlicher Arbeitsplatz (Spezifikation und Handcodierung) durch ASCET ersetzt. Die Ergebnisse dieser Aktivitäten fließen, wie vorher auch, in Form von C-Code-Modulen in die Gesamtentwicklung ein. Zugleich bietet sich die Möglichkeit bestehenden Code schrittweise in ASCET umzusetzen (Reengineering). Zu jedem Zeitpunkt liegt ein lauffähiger Code vor.

Die Umsetzung der üblichen Infrastruktur, wie z.B. Generierung der Steuergerätebeschreibungsdatei (ASAM-MCD-2MC) für die Applikation, Verwaltung von Datenständen, muss in diesem Fall wie zuvor auch im wesentlichen von Hand erfolgen. ASCET generiert allerdings die nötigen Informationen dazu.

Die Zusammenarbeit im Team wird von ASCET voll unterstützt. Es liegt eine Schnittstelle zur Anbindung von Konfigurationsmanagement (KM) Werkzeugen vor, die zur gemeinsamen Verwaltung von ASCET Spezifikationen nutzbar ist. Leistungsfähige Import/Export-Funktionen unterstützen Anwender, die ohne KM-Werkzeuge arbeiten.

ASCET als Integrationswerkzeug

Die komplette Abstützung der Entwicklung auf ASCET hat den Vorteil, dass alle Komponenten optimal aufeinander abgestimmt sind. Die Zusatzaufwendungen für die Konvertierung zwischen unterschiedlichen Werkzeugen bzw. für die Integration unterschiedlicher Quellen entfällt. Aus einer Umgebung heraus entstehen sowohl Spezifikationen als auch Implementierungen und Testfälle für ein Steuergerät. Die Verwaltung dieser unterschiedlichen Informationen ist damit ebenfalls vereinheitlicht.

In dieser Integration von vorliegenden ASCET Modulen geht es vor allem um die Konfiguration des Betriebssystems. Dabei wird festgelegt, in welcher Form die einzelnen Funktionen aufgerufen werden und wie diese Informationen austauschen. Man unterscheidet zwischen zyklischen Tasks, die in einem periodisch wiederkehrenden Zeitraster abgearbeitet werden und Eventtasks, die auf ein bestimmtes Ereignis (Interrupt) hin gestartet werden. Die zugehörigen Parameter können in ASCET einfach eingestellt werden. Zusätzlich besteht die

Möglichkeit, an dieser Stelle Monitoring-Informationen zu generieren, um weitere Untersuchungen der Regelungsalgorithmen (z.B. Laufzeitkonsistenz) durchzuführen.

Betriebssystem und Komponenten

Die Einplanung der verschiedenen Tasks (Scheduling) basiert auf den vom Entwickler vergebenen Prioritäten. Das ETAS Betriebssystem ERCOS^{EK} unterstützt an dieser Stelle sowohl kooperatives als auch präemptives Scheduling. Hauptaufgabe des Betriebssystems ist die konsistente Datenübergabe zwischen den Prozessen. Dies geschieht durch Messages. Bei Unterbrechungen (Interrupts) wird der Kontext der laufenden Task gerettet und später - nach der Ausnahmebehandlung - wieder restauriert.

Treiber für HW-Komponenten gehören nur teilweise zum Betriebssystem, da die Peripherie eines Mikroprozessors je nach Anwendung stark unterschiedlich sein kann. Die sogenannte Hardware-Kapsel greift teilweise direkt auf die entsprechende Hardware zu und teilweise über das Betriebssystem. Die Abstraktion dieser Funktionalität in Form einer Dienstschnittstelle lässt sich hingegen sehr wohl relativ allgemein halten und nach Peripherietypen kategorisieren (z.B. AD/DA-Wandler, PWM, CAN). Oberhalb dieser Ebene von Betriebssystem und HW-Kapsel liegen Protokolle für Applikation und Messen, Diagnose und die Bypass- Unterstützung. Diese werden als Automotive Services zusammengefasst. Sie bilden gemeinsam mit HW-Kapsel und Betriebssystem die Basis für die Anwendungssoftware des Steuergeräts. Dieses modulare Modell der Steuergerätesoftware hat einige Vorteile: die Elemente können einfach geprüft und ausgetauscht werden. Häufig arbeiten tatsächlich unterschiedliche Entwicklergruppen an diesen Teilen. Darüber hinaus unterstützt diese Aufteilung die schnelle Portierung eines lauffähigen Stands auf ein neues Steuergerät.

ETAS bietet neben dem Betriebssystem ERCOS^{EK} auch die entsprechende HW-Kapsel und Automotive Services an. Auf der Basis dieses Standard-Core fällt der Einstieg in die Entwicklung der Steuer- und Regelalgorithmen leichter, da man sich nur sehr wenig um die niederen Ebenen der Software kümmern muss.

4.2.5 Schnittstellen und Standards in der Toolkette

Standards haben weltweit eine große Bedeutung in der Industrie. Globales Handeln ist ohne Standards nicht denkbar. Standards sichern die Kommunikation sowohl auf funktional technischer als auch auf menschlicher Ebene. Standards sind ein aktiver Beitrag zum Investitionsschutz. Standards bieten auch die Flexibilität, Entwicklungsergebnisse oder Werkzeuge von unterschiedlicher Herkunft beliebig auszutauschen. Standards bieten Mechanismen, um Vorhandenes einfach und kostengünstig zu erweitern.

ETAS hat das Thema Standards zum Kernthema erklärt. Unser Auftrag lautet, Standards zu setzen und zu unterstützen. Wir tun dies durch die Mitarbeit in den entsprechenden Gremien und in der Entwicklung unserer Produkte. Auf diesem Weg sind bereits wichtige Meilensteine erreicht worden.

Unsere Rapid Prototyping Hardware basiert auf dem VME-Industriestandard für Bussysteme. Damit sind unsere Systeme durch neue Einschubkarten einfach erweiterbar. Die Schnittstellen zum Steuergerät sind in der Automobilindustrie inzwischen ebenfalls standardisiert. Auf Basis des ASAM-MCD-Standards ist die Verbindung zwischen Steuergerät und Applikationssystem festgelegt. Über ASAM-MCD-1b-Treiber wird im Applikations- oder Entwicklungssystem die Hardware angesprochen. Die ASAM-MCD-2MC-Beschreibungsdatei enthält die dazu nötigen Informationen über die Adressen und Umrechnungsformeln der Steuergerätesoftware. Über ASAM-MCD-3MC ist die Ankopplung von Prüfständen eindeutig definiert.

Für die Themen Dokumentation und Datenaustausch auf Ebene der Entwicklungswerkzeuge ist ein automobiler Standard im MSR-Konsortium entstanden. Damit ist jederzeit eine konsistente Entwicklung an unterschiedlichen Standorten, von unterschiedlichen Partnern, mit unterschiedlichen Werkzeugen gewährleistet.

Auf der Ebene des Betriebssystems etabliert sich gerade der OSEK-Standard für Kraftfahrzeugsteuergeräte. Auch hier arbeitet ETAS aktiv mit. Neben dem Kern des Betriebssystems wird an Schnittstellen für Kommunikation und Netzwerke gearbeitet, also den Themen HW-Kapsel und Automotive Services.

ETAS bietet auf Wunsch Schnittstellen zu anderen Werkzeugen, die noch nicht den Rang eines Standards haben, aber die entsprechenden Ziele unterstützen, wie z.B. den Daten- oder Modellaustausch mit anderen Entwicklungswerkzeugen (z.B. MATLAB Integration Package). Damit wird der Wechsel zu ASCET besonders vereinfacht, ohne dass dabei wichtige und teure Arbeitsergebnisse verloren gehen.

4.3 Entwicklungsumgebung ASCET im Einsatz

Die Entwicklungsumgebung ASCET für elektronische Steuergeräte ist die Lösung für alle oben angesprochenen Anforderungen. Mit der innovativen Technologie von ASCET gelingt es, die Faktoren Zeit, Kosten, Qualität und Flexibilität im strategischen Quadrat günstig zu beeinflussen. Die Struktur von ASCET unterstützt dies in wirksamer Weise. Im Grundumfang von ASCET sind vielfältige Spezifikationsmöglichkeiten enthalten. Blockdiagramme, Zustandsautomaten, Textspezifikationen sowie C-Einbindung geben dem Ingenieur jeweils die passende Beschreibungsmöglichkeit für die Steuer- und Regelalgorithmen. Selbst die Konfiguration des Betriebssystems erfolgt grafisch und ist damit besonders schnell und einfach erledigt. Auch die Regelstrecke kann

dabei mit modelliert und in die Entwicklung einbezogen werden. Durch eine durchgängige Datenbank wird die projektübergreifende Wiederverwertung unterstützt. Schwerpunkte in den folgenden Abschnitten sind vor allem die technischen Eigenschaften der Entwicklungsumgebung ASCET.

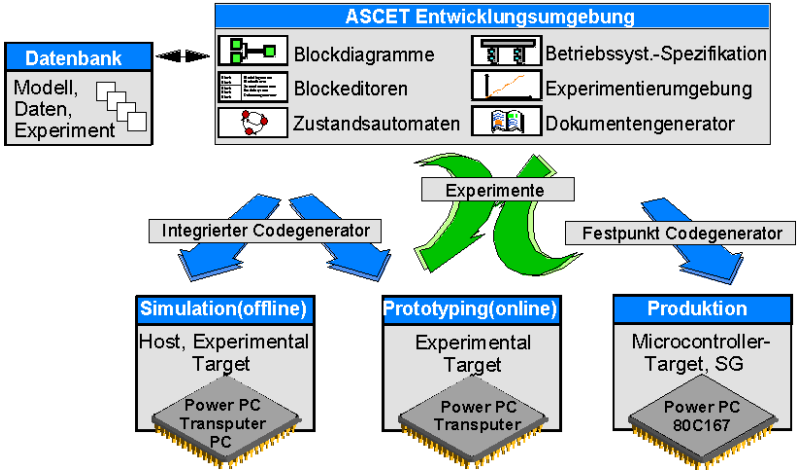


Abb. 4-17 Struktur der Entwicklungsumgebung ASCET

Unterhalb der Spezifikationsebene findet sich die Codegenerierung, die als Varianten für unterschiedliche Zielsysteme vorliegt. Sie basiert auf den Implementierungsinformationen des Anwenders. Durch diese Trennung von Physik und Implementierung wird die direkte Wiederverwertung der Regelungen erst möglich. Die Codegenerierung unterstützt dabei die folgenden Szenarien:

- Physik-Experiment
- Quantisierungs-Experiment
- Implementierungs-Experiment
- Controller-Implementierung

Die Codegenerierung betrifft sowohl die Arithmetik, die Speicherbehandlung als auch die Tasks und Prozesse des Betriebssystems. Plattformabhängigkeiten und projektspezifische Anpassungen sind in sinnvoller Weise gekapselt. Daher handelt es sich nicht allein um eine reine Codegenerierung, sondern allgemeiner um Integrationspakete, die alle Eigenschaften zur Anbindung der jeweiligen Zielsysteme beinhalten.

Eine leistungsfähige Experimentierumgebung gestattet den unmittelbaren Zugriff auf alle Daten der ablauffähigen Spezifikation bzw. des Steuergeräteprogramms. Selbst während der Echtzeit-Ausführung des Programms können Daten grafisch manipuliert und simultan aufgezeichnet werden. So besitzt der Anwender stets die volle Kontrolle über das Programm.

Durchdachte Systeme in Hard- und Software bieten eine große Flexibilität im Prototyping mit ASCET. Vorliegende Sensoren und Aktoren werden im geschlossenen Regelkreis eingebunden und gestatten so eine schrittweise Entwicklung vom Prototyp zum Produkt.

Bypass-Techniken unterstützen dieses Vorgehen und bieten einen kostengünstigen Einstieg in diese innovative Technologie. Die beiden gängigsten Hardware-Schnittstellen hierzu (ETK und CAN) werden durchgängig unterstützt.

Offene Schnittstellen bilden zusätzliche Pluspunkte für unsere Kunden. Wiederverwertung und Investitionsschutz sind dadurch praktische Realität.

4.3.1 Physikalische Spezifikation von Regelungssystemen

Die Spezifikation von Steuer- und Regelalgorithmen muss sich an der Sichtweise der Ingenieure orientieren. Hier wird auf die bewährten grafischen Darstellungsmethoden von Blockdiagrammen und Zustandsautomaten zurückgegriffen. Allerdings ist es in manchen Fällen viel einfacher, einen mathematischen Ausdruck direkt zu formulieren, als ein Blockdiagramm dafür zu erstellen. Daher unterstützt ASCET auch die Textspezifikation in einer JAVA konformen Syntax. Nachfolgend werden diese Methoden detaillierter erläutert. Dabei wird auch auf die Konfiguration des Betriebssystems und die Modellierung der Regelstrecke näher betrachtet.

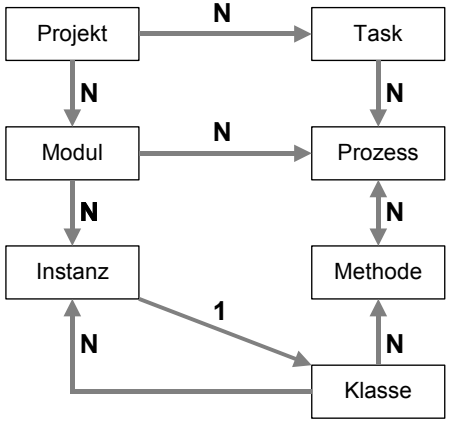


Abb. 4-18 Zusammenhang der ASCET Spezifikationselemente

Die Spezifikation in ASCET besteht aus verschiedenen abhängigen Elementen. Die ausführbare Spezifikation bzw. die Konfiguration des Betriebssystems wird stets in einem Projekt verwaltet (z.T. zur Vereinfachung implizit d.h. mit einem nicht sichtbaren Default-Projekt). Projekte bestehen aus jeweils einer Anzahl Modulen und Tasks. Diese Elemente wiederum enthalten jeweils eine Menge an Prozessen, die in den Modulen spezifiziert sind und in den Tasks zur Ausführung eingeplant werden. Module kapseln als Objekte ev. mehrere Instanzen von Klassen. Module kommen dabei jeweils nicht mehrfach vor. Zu jeder Instanz gehört dabei genau eine Klasse, welche die zugehörigen Methoden enthält. Eine Klasse kann im Gegensatz zum Modul zu mehreren Instanzen führen. Die Methoden unterscheiden sich von den Prozessen dadurch, dass sie Argumente und Rückgabewerte tragen. Die Interprozesskommunikation wird durch Messages abgebildet.

Klassen wie auch Module können durch Blockdiagramme oder durch Textspezifikationen realisiert sein. Zusätzlich werden C-Code Module und Klassen angeboten. Zustandsautomaten werden hier wie Klassen angesehen.

Klassen werden als die Hauptvertreter wiederverwertbarer Elemente angesehen. Entsprechend bietet ASCET eine umfangreiche Bibliothek von nützlichen Klassen aus dem Bereich der Regelungstechnik als auch dem allgemeinen Softwareengineering in der Datenbank an.

Alle o.g. Elemente von ASCET tragen Implementierungsinformationen. Auf diese Weise werden targetspezifische oder projektspezifische Varianten handhabbar.

Blockdiagramme

Die Blockbeschreibung der Regelungstechnik orientiert sich an den Schnittstellen der Regelstrecke. In ASCET wird dieser Ansatz ergänzt durch das Element der Kapselung, wie es aus objektorientierten Programmiersprachen bekannt ist. Die ASCET Blöcke stellen in diesem Sinn Objekte dar, die einerseits Informationen kapseln und miteinander über entsprechende Schnittstellen verbunden werden. Durch diese objektbasierte Darstellung wird die einfache und sichere Wiederverwertung gewährleistet. In den Blockdiagrammen findet man eine rein physikalische Darstellung der Steuer- und Regelalgorithmen. Die Implementierungsinformationen werden über entsprechende Editoren den verwendeten Elementen im Blockdiagramm einfach hinzugefügt.

In klassischen Blockdiagrammen ist die genaue Rechenreihenfolge nicht immer eindeutig bestimmt. Die Blockdiagramme in ASCET enthalten neben den typischen Elementen des Datenflusses (Variablen, Kenngrößen, arithmetische Operatoren) auch Kontrollflusselemente wie z.B. Verzweigungen. Der Kontrollfluss wird durch einen eigenen Linien- bzw. Verbindungstyp übersichtlich dargestellt. Darüber hinaus gestattet ASCET die Spezifikation der Abarbeitungsreihenfolge von Blockoperationen direkt durch Blockattribute in der Gra-

fik. Mittels Sequencing und Kontrollflusselementen können komplexe Algorithmen endlich auch grafisch exakt dargestellt und sehr einfach bearbeitet werden.

Durch ein anwendungsbezogenes Sichtenkonzept wird der Know-How-Schutz für das firmenübergreifende Arbeiten unterstützt. Der Anwender kann dazu in verschiedenen Sichten definieren, welche Blöcke in der Dokumentation gezeigt oder versteckt werden sollen. Darüber hinaus gestattet ASCET die Definition von passwortgeschützten Zugriffsrechten auf Klassen-, Modul- und Projektebene.

Zustandsautomaten

Die Anwendung von hierarchischen Zustandsautomaten ist in der Regelungstechnik weit verbreitet. Insbesondere für Systeme, in denen je nach Arbeitspunkt unterschiedliche Regelstrategien benötigt werden, bietet sich eine solche Modellierung an. Die Triggerbedingungen und Zustandsmethoden können dabei sowohl durch Blockdiagramme als auch textuell spezifiziert werden. Es kann also jeweils die günstigste Art der Darstellung gewählt werden.

Textspezifikation in ESDL

Im Gegensatz zur Entwicklung von C-Code ist die Textspezifikation von Steuer- und Regelalgorithmen in ESDL (Embedded Software Description Language) eine portable physikalische Beschreibung. Auch hier werden Instanzen in den jeweiligen Klassen bzw. Modulen gekapselt, die wahlweise mit targetspezifischen Implementierungsinformationen angereichert werden. Die Sprache ESDL orientiert sich an JAVA, da diese C ähnliche Sprache eine weite Verbreitung besitzt und leicht erlernt werden kann. Spezifikationen in ESDL, Zustandsautomaten und Blockdiagramme können beliebig gemischt werden, d.h. in einer Klasse können zum Beispiel einige Instanzen von Blockdiagrammen und andere von Textspezifikationen in ESDL enthalten sein. Im Gegensatz zu C wird in ESDL keine Pointerarithmetik benötigt, da alle Objekte direkt angesprochen werden können und keine dynamische Instanziierung stattfindet. Mit anderen Worten: In ASCET stehen alle Objekte zur Compilezeit fest. Das ermöglicht eine frühe Konsistenzprüfung der Steuergeräteprogramme und stellt sicher, dass der verfügbare Speicher tatsächlich ausreicht. Das gilt auch für den Umgang mit Arrays, Matrizen, Kennlinien und Kennfeldern. Diese Objekte besitzen als Klassen ein festgelegtes Protokoll für den Zugriff und benötigen daher auch keine Pointerarithmetik.

Einbindung von C

C-Code ist im Kontext von ASCET immer auf der Implementierungsebene zu sehen. Er ist daher zwangsläufig targetspezifisch und wird auch so verwaltet.

Die Einbindung von C-Quellen kann auf zweierlei Weise erfolgen, nämlich als interner oder externer C-Code. Beim internen C-Code werden die Quellen wie diejenigen entsprechender ESDL-Klassen in der Datenbank von ASCET verwaltet. Externer C-Code hingegen wird auf dem Dateisystem des Anwenders abgelegt und steht damit direkt für andere Anwendungen zur Verfügung (s.o. „ASCET als zusätzlicher Programmierer"). Dabei kann auch Binärcode eingebunden werden, wenn die C-Quellen nicht vorliegen.

Der Einsatz von C-Code-Blöcken ist vor allem zweckmäßig, wenn es sich um spezielle targetspezifische Treiber handelt. Die Schnittstelle ist bidirektional. Aus beliebigen Klassen können Methoden von C-Code-Blöcken aufgerufen werden. Zusätzlich besteht im C-Code auch eine Zugriffsmöglichkeit auf die übrigen physikalisch spezifizierten Objekte. Es ist klar, dass die Schnittstelle eine besondere Beachtung verdient, da es hier auf Implementierungsdetails ankommt.

Konfiguration des Betriebssystems

Die Schnittstelle zum Betriebssystem liegt im sogenannten Projekt (s.o.). Für die Tasks, die im Scheduling einzuplanen sind, werden Prioritäten vergeben. Zusätzlich enthält jede Task weitere Attribute, die zur Planung der Abarbeitungsreihenfolge wichtig sind, wie z.B. ob sie kooperativ oder präemptiv sein soll, ob sie zyklisch, auf ein externes Ereignis hin oder nur inital gestartet werden muss. Auf dieser Basis erfolgt dann die Konfiguration des Betriebssystems. Weitere Informationen müssen dabei berücksichtigt werden. Die Prozesse, die in den Tasks aufgerufen werden, kommunizieren über Messages. Messages aus unterschiedlichen Modulen werden über Namensgleichheit gebunden. Bei

der Codegenerierung werden diese Informationen auf Konsistenz geprüft (Verwendung der Messages an mehreren Stellen, Umgang mit globalen Variablen etc.).

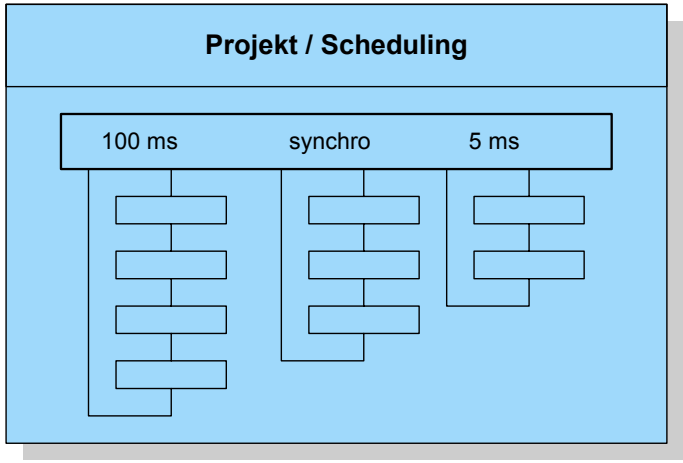


Abb. 4-19 Prozesse und Tasks im Projekt

Die Konfiguration des Betriebssystems erfolgt visuell auf Basis eines einfachen Editors. Dadurch hat man stets einen guten Überblick über das Gesamtsystem. Veränderungen im Scheduling können sehr einfach und schnell vorgenommen werden. Dies wird nicht zuletzt dadurch unterstützt, dass das ETAS-Betriebssystem ERCOS^{EK} als vorübersetzte Bibliothek vorliegt. Bei Änderungen an der Konfiguration des Betriebssystems entsteht also bis auf den Linkprozess kaum Aufwand.

Modellierung der Regelstrecke

Die Spezifikation von komplexen Steuer- und Regelalgorithmen ist ohne eine Modellierung der zugehörigen Strecke heute nicht mehr denkbar. ASCET bietet hier den besonderen Vorteil, dass Regelung und Streckenmodelle in einem System entwickelt werden können. Auf diese Weise entsteht kein Zusatzaufwand durch Konvertierungen oder Simulatorkopplungen. In kleinen Projekten kommen Strecke und Regelung häufig aus einer Hand, d.h. sie werden von einem Ingenieur allein entwickelt.

ASCET besitzt aber auch die Offenheit andere Systeme anzubinden. So wurde beispielsweise eine Kopplung zu Matlab geschaffen, die es gestattet, Simulink-Modelle mit ASCET Spezifikationen gemeinsam zu simulieren.

Insgesamt kommt man mit dieser Technik einen großen Schritt in Richtung Effizienz und Effektivität weiter. Das gilt vor allem, wenn das entsprechende Laborauto in Form von Modellen und Hardware vorliegt. Dann kann sehr viel Zeit gespart werden. Sicherheitskritische Untersuchungen können kostengünstig und gefahrlos am Schreibtisch vorangetrieben werden. Erst zur Erprobung muss dann ins Fahrzeug gestiegen werden. ETAS bietet diese Technologie für viele Systeme bereits heute an.

4.3.2 Implementierung und Codegenerierung

Automatische Codegenerierung für Seriensteuergeräte ist der Schlüssel für eine effiziente Entwicklungsmethodik. ASCET hat hier den Maßstab gesetzt. Dabei geht es nicht allein um die besondere Herausforderung der Festpunktarithmetik. Auch für Gleitkommaprozessoren muss die Schnittstelle zum Betriebssystem konfiguriert werden, muss die Speicherverwaltung optimal gestaltet sein und muss die jeweilige Hardware-Kapsel eingebunden werden. Im übrigen wird auch bei Gleitkommasystemen zwischen genauen und doppelt genauen Daten unterschieden. Das ist zumindest in der Simulation wichtig.

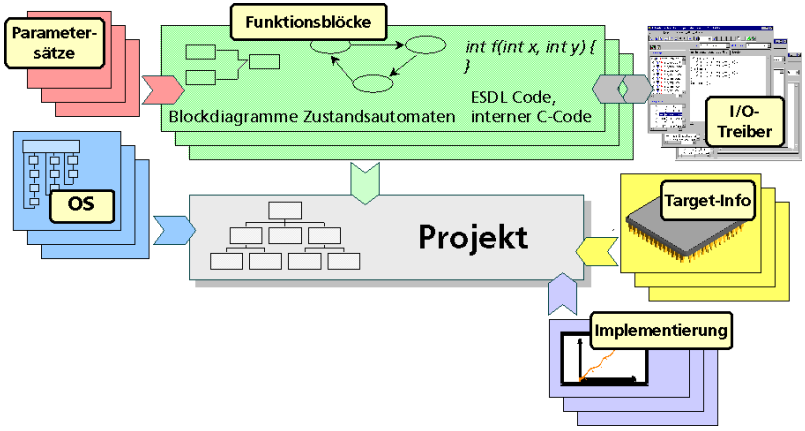


Abb. 4-20 Prinzip der Automatischen Codegenerierung in ASCET: die Inputs

In ASCET besteht die Implementierung aus den jeweiligen Datentypen, Wertebereichen und Speicherablageinformationen. Umrechnungsformeln bilden die Verknüpfung zwischen dieser Implementierungsebene und der physikalischen

Beschreibung von Daten. Für Funktionen wird neben der Speicherablageinformation noch festgelegt, ob die Funktion inline berechnet werden soll, oder ob z.B. bei Kennlinienauswertungen eine Dienstroutine verwendet werden soll.

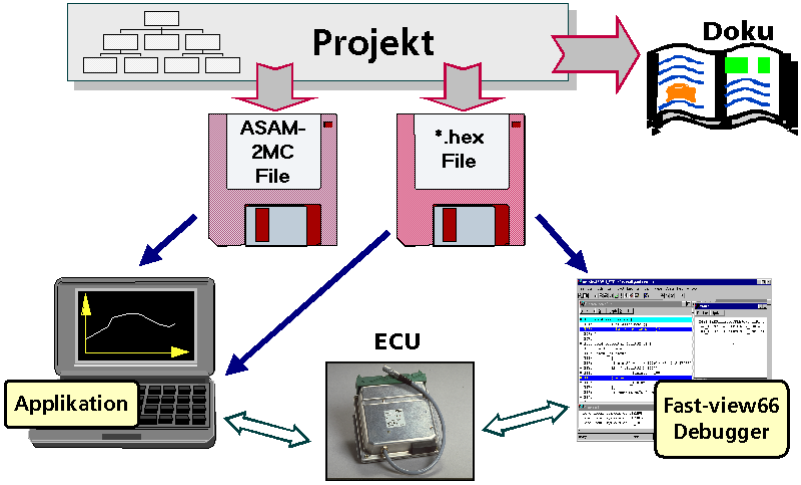


Abb. 4-21 Prinzip der Automatischen Codegenerierung in ASCET: die Outputs

Automatische Codegenerierung ist aber keine Einbahnstraße. Parallel zum erzeugten C-Code wird eine ASAM-MCD-2MC-Beschreibung benötigt, welche die nötigen Adressinformationen für die Applikations- und Messsysteme bereithält. Dazu muss das entsprechende MAP-File nach der Erzeugung des Programms zurückgelesen und interpretiert werden. ASCET-SE leistet das. Überdies kann man sogar von einer ASAM-MCD-2MC-Datei ausgehen, diese in ASCET einlesen und daraus eine erstes Datenmodell in ASCET aufbauen (Reengineering eines bestehenden Programms).

Algorithmen

Die automatische Umsetzung von Blockdiagrammen, Textspezifikationen und Zustandsautomaten in targetspezifischen C-Code läuft über eine gemeinsame Zwischenschicht. Dabei werden die Implementierungsinformationen benutzt, um die Abbildung der Algorithmen auf das Zielsystem optimal zu gestalten. Neben der Beherrschung komplexer Logik stellt insbesondere die Arithmetik eine besondere Herausforderung dar. Das gilt umso mehr, wenn es sich um die Integer-Codegenerierung handelt. Selbst die einfache Zuweisung

```
a = b
```

zweier Variablen ist eine nicht triviale Operation für die Codegenerierung, wenn die Implementierungen unterschiedlich sind. Es seien a und b durch die folgenden Formeln jeweils als vorzeichenlose 8 Bit-Größen (Wertebereich von 0 bis 255) implementiert:

```
a = 2 * a_impl, b = 3 * b_impl .
```

Dann ergibt einfaches Einsetzen:

```
a_impl = 3 * b_impl / 2 .
```

Hier muss nun schon auf die Reihenfolge der Operationen geachtet werden, um die Anforderung an maximale Genauigkeit zu berücksichtigen. Wenn man z.B. erst die Division durchführte, würden aufgrund der Integerrechnung die unterschiedlichen Umrechnungsformeln unwirksam und die Ergebnisse immer um 50% falsch sein.

```
a_impl = (3 / 2)* b_impl = b_impl.
```

Es muss auch auf das Thema Überlauf geachtet werden. Führt man nämlich die Multiplikation mit 3 zuerst aus, ergibt sich ein Überlauf, sobald b_impl größer als 255 / 3 = 85 ist. Ebenso muss im Allgemeinen aufgepasst werden, ob es zu Unterläufen bzw. Rundungsungenauigkeiten kommt. Falls nämlich zuerst durch 2 geteilt wird, entspricht dies einer Rechtsschiebeoperation, d.h. das letzte Bit verfällt. Es kann also nicht unterschieden werden, ob b_impl den Wert 1 oder 0 hatte. In beiden Fällen ist das Ergebnis für a_impl und damit auch für a der Wert 0.

Tatsächlich macht die Zuweisung a = b nur Sinn, wenn die physikalischen Wertebereiche gleich sind (hier maximal 0 bis 510). Damit kann b_impl maximal den Wert 510 / 3 = 170 annehmen. Ein Überlauf kann hier also vorkommen und muss auf jeden Fall vermieden werden. Man könnte nun auf den Gedanken kommen, in der Codegenerierung eine Fallunterscheidung zu treffen, nämlich für Werte von b_impl bis 170 die Multiplikation zuerst durchzuführen und für Werte von b_impl größer als 170 zuerst zu dividieren. Diese führt allerdings zu einem entsprechenden Mehrbedarf an Code. Es muss also hier der geringe Fehler in der Genauigkeit von maximal 1.5 im gesamten Wertebereich akzeptiert werden.

Es ist klar, dass die Situation selbst bei gewöhnlichen arithmetischen Operationen mit wenigen Operanden noch durchaus schwieriger wird, von komplexen Verknüpfungen bzw. Ausdrücken ganz zu schweigen. Die automatische Codegenerierung von ASCET entlastet den Anwender weitgehend von derartigen Problemen.

Speicherbehandlung

In der Steuergerätearchitektur sind Speicherbereiche definiert, die unterschiedlichen Zwecken dienen. Einige Bereiche sind für das Programm vorbehalten, andere dienen zur Aufnahme applizierbarer Daten. Bereits physikalisch muss zwischen ROM, RAM und Flash unterschieden werden. Bei manchen Steuergeräten werden spezielle Bereiche für die Ablage von Bits vorbehalten. Auch ist zu beachten, dass in manchen Fällen Werte nur auf geraden Adressen abgelegt werden dürfen. Einige Speicherbereiche sind zudem als Eingänge bzw. Ausgänge der Prozessorperipherie reserviert. In ASCET werden diese Informationen durchgängig berücksichtigt. Die Steuergerätearchitektur steht dem Anwender zur Beschreibung der Implementierung zur Verfügung, und sie kann an die jeweiligen physikalischen Gegebenheiten angepasst werden.

Alle Standarddaten (wie zum Beispiel Meß- und Verstellgrößen) erhalten automatisch einen Default-Speicherbereich zugewiesen. Die automatische Codegenerierung überführt diese Implementierungen in die entsprechenden Pragma-Anweisungen im C-Code. Damit ist der Anwender wiederum von einer störenden und fehlerträchtigen Verwaltungsarbeit entlastet.

Konfiguration des Betriebssystems

Die Konfiguration des Betriebssystems mit ASCET gestaltet sich für den Anwender ebenfalls sehr einfach. Geführt durch ein grafisches Bediensystem werden die nötigen Informationen im ASCET Projekt spezifiziert. Der entsprechende C-Code wird wiederum automatisch erzeugt.

Die Hauptaufgabe der automatischen Codegenerierung stellt sich hier in der Optimierung der Kommunikation unterschiedlicher Prozesse über Messages. Insbesondere bei Unterbrechungen müssen im Allgemeinen alle Messages gerettet werden, um nach der Unterbrechung mit konsistenten Daten weiterarbeiten zu können. Auf diese Weise entsteht aber ein gewaltiger Aufwand, der zum größten Teil überflüssig ist. Es müssen nämlich nur diejenigen Daten gerettet werden, die bei einer Unterbrechung überhaupt inkonsistent werden können. Hier ergibt sich ein hohes Optimierungspotential, das in der automatischen Codegenerierung ausgenutzt wird.

Für den Anwender ergibt sich somit eine große Arbeitsersparnis, da die Auswertung der komplexen Unterbrechungsmöglichkeiten automatisch analysiert und in der Codegenerierung berücksichtigt werden. Fehler durch das Vergessen einer entsprechenden Rettungsoperation sind damit ausgeschlossen. Vor allem während der Entwicklungsphasen, wenn sich auch die Struktur der Prozesse und Tasks noch ändert, spürt man diese Entlastung von aufwendigen Pflegeoperationen deutlich. ASCET bietet hier sowohl Komfort als auch Sicherheit.

Plattformabhängigkeit und projektspezifische Anpassung

Die Leistungsfähigkeit von ASCET und der automatischen Codegenerierung zeigt sich nicht nur in den erfolgreichen Produktentwicklungen, die unsere Kunden bereits durchgeführt haben, sondern auch in dem Potential, zukünftige Anforderungen problemlos bewältigen zu können. Der Schlüssel hierzu ist die offene, anpassungsfähige Schnittstelle dieser Technologie. Diese erweiterte Einsatzfähigkeit ist inzwischen an mehreren Beispielen nachgewiesen. So wurde auf dieser Basis sowohl die Anbindung fremder Betriebssysteme realisiert als auch projektspezifische Anforderungen (z.B. in Form von Namensbildungsregeln oder der Adaption an kundenspezifische Entwicklungsprozesse) erfolgreich umgesetzt.

Die Aufteilung der Entwicklung in die physikalische Modellierung und die Spezifikation der Implementierung stellt das Fundament für weitgehende Anpassungsmöglichkeiten dar. Damit werden die plattformspezifischen Eigenschaften an einer zentralen Stelle (der Implementierung) gekapselt und können so in einem Schritt ausgetauscht werden. Hinzu kommen weitgehende Konfigurationsmöglichkeiten der automatischen Codegenerierung selbst. Aufbauend auf einer zentralen Zwischenschicht in ASCET werden in ASCET-SE sogenannte Codeproduktionsregeln verwendet, die auf spezielle Bedürfnisse hin veränderbar sind. Damit ergeben sich ungeahnte Eingriffsmöglichkeiten in der automatischen Codegenerierung von ASCET, falls das im Kontext der Entwicklungsaktivitäten notwendig bzw. nützlich ist.

4.3.3 Prototyping mit ASCET

Mit ASCET entstehen erste Ergebnisse sehr zügig, und diese lassen sich auch sehr schnell in Produkte umsetzen. Für den Erfolg des Prototyping ist die Experimentierumgebung neben der unterstützten Hardware der entscheidender Faktor. Dies wird nachfolgend näher beleuchtet.

Experimentierumgebung für höchste Ansprüche

Die Experimentierumgebung in ASCET zeichnet sich dadurch aus, dass verschiedene Untersuchungen an den Entwicklungsobjekten durchgeführt werden können, ohne diese selbst zu verändern. Es liegt damit eine universelle Arbeitsumgebung vor, in der einfach und schnell Stimuli definiert und erzeugt werden, Daten aufgezeichnet und vermessen werden, Kenngrößen verändert werden. Die Simulation kann zu beliebigen Zeitpunkten gestartet und angehalten werden. Einmal erarbeitete Einstellungen und optimierte Datensätze sowie Fensterkonfigurationen lassen sich zur Wiederverwendung als sogenanntes Experiment separat abspeichern und damit längerfristig wiederverwenden. Mit jedem Modell können mehrere Experimente für unterschiedliche Zwecke verwaltet werden. Zum Messen und Verstellen liegen eine Vielzahl an leistungsfähigen grafischen und textuellen Anzeigen und Editoren vor, die fle-

xibel zu kombinieren sind. Die Experimentierumgebung von ASCET bietet diese überragende Funktionalität identisch sowohl offline als auch im Echtzeitexperiment online.

Nach der ersten Analyse der Steuer- und Regelalgorithmen als physikalisches Experiment unterstützt ASCET auch die Untersuchung der Quantisierung und von Implementierungen in der gleichen Arbeitsumgebung. Auf diese Weise ergibt sich die schrittweise Verfeinerung der Entwicklung bis hin zum Produkt. Besondere Unterstützung wird darüber hinaus durch das Bereitstellen spezieller Monitorgrößen in der Experimentierumgebung geboten.

Bei der ersten Untersuchung von neuen Modulen oder Klassen erlaubt die Experimentierumgebung das schrittweise Einbeziehen einzelner Prozesse und Methoden in die Simulation. Diese dynamische Konfiguration des Betriebssystems direkt in der Experimentierumgebung erlaubt eine flexible und schnelle Arbeitsweise und führt damit sehr schnell zu aussagekräftigen Resultaten.

Die in der Experimentierumgebung von ASCET verwendeten Anzeige- und Verstellelemente entsprechen denjenigen unseres Applikations- und Messsystems INCA-PC. Damit ist die Durchgängigkeit der Entwicklungsumgebung auch auf Ebene des Bediensystems von der Idee bis zum Produkt gegeben.

Simulationstechnik: Hardware und Software

Prototyping lebt vor allem von der unterstützten Hardware. Wichtigster Bestandteil ist dabei die Ausführungsplattform für die Simulation. Hier bietet ETAS ein skalierbares Konzept. Zur Basisuntersuchung neuer Steuer- und Regelalgorithmen dient ein Hochleistungsrechenknoten (Power-PC). Diese Plattform hat sich bewährt und bietet vielfältige Erweiterungsmöglichkeiten.

Das Betriebssystem auf der Simulationsplattform ist dasselbe wie in der Serie: ERCOS^{EK}. Damit ist das Prototyping besonders realistisch und es können alle nötigen Untersuchungen im Echtzeitkontext durchgeführt werden.

Abb. 4-22 Universal VME-Bus System ES1000

Als Einsteckkarte in unserem ES1000 VME-Bussystem können verschiedene andere Karten mit in die Simulation einbezogen werden. Damit wird der wesentliche Industriestandard in diesem Bereich unterstützt. Die Möglichkeiten reichen von der Verwendung bereits vorliegender marktgängiger Karten zur Einbindung von externen Sensoren und Aktoren bis hin zum Einsatz spezieller projektspezifischer Signalkonditionierungen, die auf Anforderung aufgebaut werden.

Anbindung vorliegender Sensoren und Aktoren im geschlossenen Regelkreis

ETAS bietet für alle gängigen Schnittstellen zu Sensoren und Aktoren die entsprechende Hardware- und Softwarelösung. VME-Buskarten liegen vor für

- AD/DA-Wandlung
- PWM-Signalaustausch
- CAN-Schnittstelle
- LIN-Schnittstelle
- Digital I/O
- FPGA

Darüber hinaus werden spezielle Lösungen für Temperaturmessung und die Erfassung von Abgasdaten angeboten, die sich leicht in diese Architektur integrieren.

Die Schnittstelle zum PC ist über Ethernet realisiert. Es kann also auch mit Laptops mobil gearbeitet werden. Auch ältere Hardwaresysteme mit paralleler Schnittstelle (Druckerport) werden noch unterstützt. Über ein effizientes Protokoll können Daten zwischen Experimentierhardware und Experimentierumgebung in ASCET online ausgetauscht werden. Die Ausführung des geschlossenen Regelkreises auf Hardware-Ebene wird dadurch nicht gestört. Arbeiten mit ASCET führt direkt von der Idee zu realistischen Prototypen.

4.3.4 Bypass

Für den Einsatz unserer Bypass-Technologie stehen zwei physikalische Schnittstellen allgemein zur Verfügung: Speicheremulation über ETK und die CAN-Schnittstelle. Andere Hardwareschnittstellen können auf Anforderung kundenspezifisch realisiert werden. Alle Bypass-Techniken gehen von einem lauffähigen Steuergerät aus, für das eine Funktion verändert oder zusätzlich entwickelt werden soll. Zum Aufsetzen der Bypass-Anwendung muss dementsprechend ein sogenannter Freischnitt dieser Funktion vorliegen. In dem Freischnitt ist das Scheduling der Funktion bereits vorgeleistet. Es muss also vorab bekannt sein, in welchem Zeitraster die Funktion abgearbeitet werden muss. Je nach Situation sind auch die notwendigen Sicherheitsmaßnahmen zu treffen für den Fall, dass es zu Kommunikationsfehlern kommt. Wenn also beispielsweise eine Timeout in der CAN-Kommunikation auftritt, kann ein Notprogramm gestartet werden, oder es wird mit Defaultdaten weiter gerechnet.

Die Bypass-Technologie unterstützt vor allem Entwicklungspartnerschaften, in denen bestimmte Funktionen aus Gründen des Know-How-Schutzes nicht offengelegt werden sollen. Da nur eine spezifische Schnittstelle freigelegt wird, kann man sich andere aufwendigere Maßnahmen sparen. Außerdem liegt stets ein Großteil des Gesamtsystems stabil vor, und man kann sich auf die Entwicklung einer innovativen Funktion voll konzentrieren. So entsteht eine kostengünstige Lösung mit großem Nutzen.

ETK-Schnittstelle – Speicheremulation

Mit dem Emulatortastkopf, kurz ETK, bietet ETAS eine einzigartige Schnittstelle zum Steuergerät. Basis ist die vollständige oder teilweise Emulation des Steuergerätespeichers durch ein DPRAM, auf das einerseits das Steuergerät direkt zugreift und das andererseits transparent vom PC aus bedient und ausgelesen wird. Dies erfordert eine Änderung des Steuergeräts, da die Speicheremulation nur über sehr kurze Entfernungen zwischen ETK und Prozessor realisiert werden kann. ETKs sind heute bereits sehr klein und können auch in Hybridtechnik dargestellt werden.

Für unser VME-System ES1000 bieten wir entsprechende Einschubkarten an, mit denen die Verbindung zum ETK hergestellt wird. Da ETKs für sehr viele unterschiedliche Prozessoren angeboten bzw. angepasst werden, ergibt sich hier eine universelle Lösung für die Anwendung.

CAN-Schnittstelle

In Fällen, in denen ein ETK nicht eingesetzt werden kann oder soll, bietet sich die Verwendung einer CAN-Schnittstelle an. Dies erfordert keine Änderung der Steuergeräte-Hardware und ist damit nahezu immer möglich. Allerdings werden bei diesem Ansatz die Leistungsdaten des ETK nicht erreicht. Auch muss eine ausreichende Anzahl freier Botschaften verfügbar sein. Unter diesen Einschränkungen ist der CAN-Bypass mit ASCET dennoch in vielen Fällen erste Wahl, da er neben den genannten Eigenschaften besonders kostengünstig ist.

4.3.5 Wiederverwertung und offene Schnittstellen

Das Thema Wiederverwertung ist schon seit Jahren in aller Munde. Schlagworte wie Modularisierung, Objektorientierung und Schnittstellenkompatibilität beherrschen die Diskussionen. Bei ASCET sind Wiederverwertung und offene Schnittstellen praktische Realität. Dies ist das Ergebnis einer sorgfältigen Analyse der Entwicklungsschritte und Entwicklungsergebnisse in realen Projekten und der zielgerichteten Umsetzung dieser Beobachtungen in nützliche Eigenschaften unserer Werkzeuge.

Auf diese Weise ist eine Bibliothek an ausgezeichneten Blöcken für die Spezifikation von Steuer- und Regelalgorithmen entstanden, die sich in der Praxis bewährt hat und nun den Standard für unsere Wettbewerber darstellt.

Wiederverwertbare Experimente sind mit dieser Bibliothek verknüpft. Sie gestatten ein reproduzierbares und nachvollziehbares Verhalten bei alten und neuen Entwicklungsaufgaben. Unsere Schnittstellen zu Werkzeugen für Applikation, Erprobung und Test gestatten dabei die Einbeziehung von allen relevanten Arbeitsschritten in die Entwicklung des Gesamtsystems. Dabei werden nicht nur Schnittstellen zu anderen ETAS-Werkzeugen angeboten sondern Schnittstellenstandards wie ASAP und MSR verwendet, die allgemein zugänglich sind.

Wiederverwertung durch Datenbankunterstützung

Die ASCET Datenbanktechnik ist die konzeptionelle Basis der Wiederverwertung. Sie wurde von Anfang an im Kern von ASCET realisiert und stellt somit das Rückgrat des Werkzeugs dar. Die Vorteile für den Anwender sind evident: In einer eigenen Arbeitsumgebung sind alle Daten sicher abgespeichert und vor unbeabsichtigter oder unerlaubter Manipulation oder sogar Vernichtung geschützt. Schließlich handelt es sich dabei um wertvolle Arbeitsergebnisse, die diesen besonderen Schutz erfordern. In der Verwaltung dieser Daten bietet

die Datenbank von ASCET darüber hinaus alle notwendigen Mechanismen für ein effizientes tägliches Arbeiten sowohl für Einzelpersonen als auch für das Arbeiten im Team.

Programm- und Datenhaltung mit Konfigurationsmanagement-Tools

Wenn Varianten von Entwicklungssträngen beherrscht werden müssen oder große Teams an komplexen Systemen gemeinsam arbeiten, wird häufig der Einsatz eines käuflichen Konfigurationsmanagement-Tools favorisiert. In diesen Werkzeugen werden dann mitunter auch andere Firmendaten verwaltet, so dass die Werkzeugauswahl vom Kunden vorgegeben wird. Für eine Anbindung eines beliebigen Konfigurations-Management-Tools liefert ASCET die nötigen Schnittstellen. Bei dieser Lösung arbeitet ASCET als Client an dem Konfigurationsmanagement-Tool. Verwendet man hingegen ASCET als Server, so steht die komplette Datenbankfunktionalität von ASCET für eine detaillierte Anpassung an die Verwaltungsmechanismen des Konfigurationsmanagement-Tools zur Verfügung. Dazu müssen lediglich die entsprechenden Schnittstellen in JAVA umgesetzt werden. ASCET steht also in jedem Fall bereit für den Einsatz zur Realisierung innovativer und komplexer Steuer- und Regelaufgaben.

Wenn Sie den Einsatz eines Konfigurationsmanagement-Tools wünschen, setzen Sie sich wegen einer auf Ihre speziellen Anforderungen abgestimmten Lösung bitte mit ETAS in Verbindung.

4.4 Aufbau der ASCET Software

Die ASCET Softwarefamilie ist in vier Produkte gegliedert, die die verschiedenen Arbeitsphasen im Arbeitsprozess des Anwenders unterstützen.

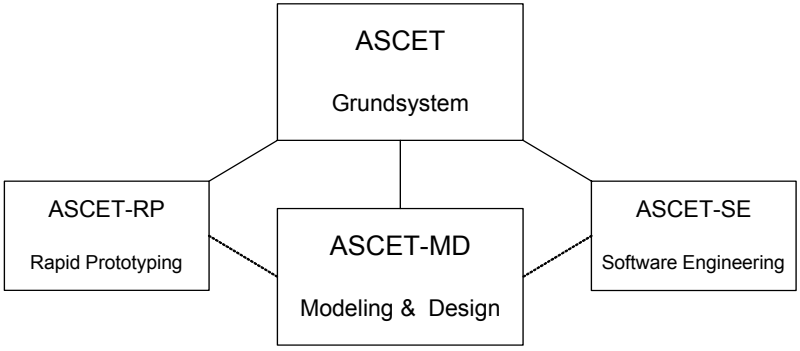


Abb. 4-23 Die modulare Struktur von ASCET

Im Weiteren wird der Funktionsumfang der einzelnen Produkte kurz beschrieben.

ASCET Grundsystem

Das ASCET Grundsystem ist die Basis für die anderen Produkte der ASCET Familie, die nur in Verbindung mit dem Grundsystem installiert werden können.

ASCET Modeling & Design

ASCET-MD ermöglicht die Spezifikation von Modellen als Blockdiagramme, in ESDL oder der Programmiersprache C. Wie aus früheren ASCET-SD Versionen bekannt, können hier Modelle spezifiziert und verwaltet sowie im Offline-Experiment simuliert werden.

ASCET-MD kann in derselben Version nur einmal auf demselben PC installiert werden.

ASCET Rapid Prototyping

ASCET-RP bietet den vollen zum Rapid Prototyping erforderlichen Funktionsumfang. Dieser ist in einem gesonderten Handbuch beschrieben.

In ASCET-RP können Sie Komponenten betrachten, das Erzeugen oder Bearbeiten von Modellen oder Modellelementen ist jedoch nur in Verbindung mit ASCET-MD möglich.

ASCET-RP kann in derselben Version nur einmal auf demselben PC installiert werden. Die gleichzeitige Installation von ASCET-RP, ASCET-MD und ASCET-SE für einen oder mehrere Microcontroller ist möglich.

ASCET Software Engineering

ASCET-SE bietet den vollen für die Generierung von Steuergerätecode erforderlichen Funktionsumfang. Dieser ist in einem gesonderten Handbuch beschrieben.

In ASCET-SE können Sie Komponenten betrachten, das Erzeugen oder Bearbeiten von Modellen oder Modellelementen ist jedoch nur in Verbindung mit ASCET-MD möglich.

ASCET-SE ist in Portierungen für verschiedene Mikrocontroller erhältlich. Anders als bei den anderen Produkten der ASCET Familie kann ASCET-SE für verschiedene Targets gleichzeitig auf demselben PC installiert werden. Die gleichzeitige Installation von ASCET-SE für einen oder mehrere Mikrocontroller, ASCET-RP und ASCET-MD ist ebenfalls möglich.

5 Allgemeine Bedienmöglichkeiten von ASCET

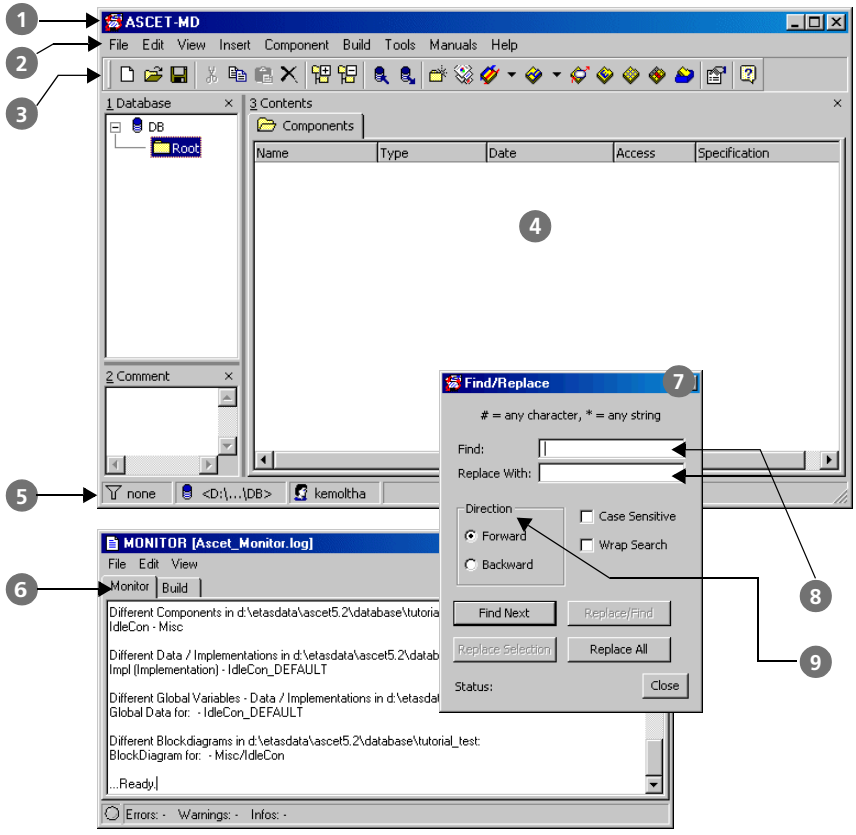
In diesem Teil erhalten Sie Informationen über die Fenster- und Menüstrukturen, Bedienmöglichkeiten über Maus und Tastatur sowie Angaben zu Hilfefunktionen.

Sie sollten dieses Kapitel unbedingt lesen, da einige Bedienmöglichkeiten nur an dieser Stelle beschrieben werden. Alle hier angesprochenen Techniken sind zwar Windows-Standards, könnten jedoch dem weniger erfahrenen Windows-Anwender unbekannt sein. Sie werden deshalb hier einmal zentral beschrieben.

Bei der Entwicklung von ASCET wurde großer Wert auf eine einfache Bedienung über die Tastatur gelegt. Besonderheiten und Abweichungen zu Windows-Vereinbarungen bei der Tastaturbedienung finden Sie im Kapitel „Bedienung über Tastatur“ auf Seite 235.

5.1 Aufbau der Fenster

Die ASCET-Fensterelemente



- Titelzeile (1)
- Menüzeile (2)
- Schaltflächenleiste (3)
- Fensterbereich (4)
- Fußzeile (5)
- Register (6)
- Dialogfenster (7)
- Felder (8)
- Feldüberschrift(9)

5.2 Schaltflächenleisten

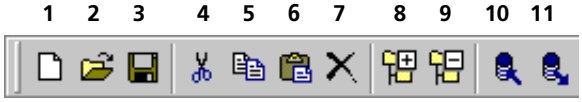
Die meistbenötigten Befehle stehen alternativ auch als Schaltflächen zur Verfügung. Ein Befehl kann so auf einen einfachen Mausklick hin ausgeführt werden.

Hinweis

Alle Befehle, die mit den einzelnen Schaltflächen gestartet werden können, befinden sich ebenfalls in entsprechenden Menüs.

Alle Schaltflächen auf den Schaltflächenleisten sind maussensitiv. Wenn Sie den Mauszeiger auf eine Schaltfläche ziehen und ihn eine Sekunde lang nicht bewegen, erscheint in unmittelbarer Nähe der ausgewählten Schaltfläche ein Textfeld, in dem die Funktion der Schaltfläche angezeigt wird.

5.2.1 Schaltflächen im Komponentenmanager



1. New (erzeugt eine neue Datenbank)
2. Open (öffnet eine Datenbank)
3. Save (sichert die Datenbank)
4. Cut (schneidet das ausgewählte Element aus)
5. Copy (kopiert das ausgewählte Element)
6. Paste (fügt das Element aus der Zwischenablage ein)
7. Delete (löscht das ausgewählte Element)
8. Expand all (klappt die Baumstruktur in der Liste „1 Items“ weitestmöglich auf)
9. Collapse all (klappt die Baumstruktur in der Liste „1 Items“ vollständig zu)
10. Import
11. Export

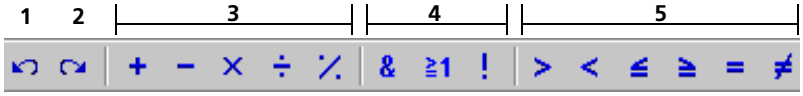
12. Insert Folder (fügt einen Ordner ein)
13. Insert Project (fügt ein Projekt ein)
14. Insert Module - <Type> (fügt ein Modul ein)
15. Insert Class - <Type> (fügt eine Klasse ein)

    Mit den Pfeilen (14a und 15a) kann der Typ des Objekts gewählt werden.
16. Insert State Machine (fügt einen Zustandsautomaten ein)
17. Insert Enumeration (fügt eine Enumeration ein)
18. Insert Boolean Table (fügt eine Boolesche Tabelle ein)
19. Insert Conditional Table (fügt eine Bedingungstabelle ein)
20. Insert Container (fügt einen Container ein)
21. Options (öffnet das Optionsfenster)
22. ? (öffnet das Fenster „About ASCET“ mit Informationen zu den installierten Produkten der ASCET-Produktfamilie)

23. Search - <*criterion*> (durchsucht die Datenbank nach einem Siuchstring, unter einem einstellbaren Suchkriterium)

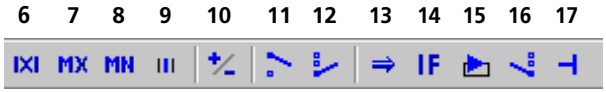
    Mit dem Pfeil 24a kann das Suchkriterium gewählt werden.
24. Eingabefeld für den Suchstring

5.2.2 Schaltflächenleisten im Blockdiagrammeditor



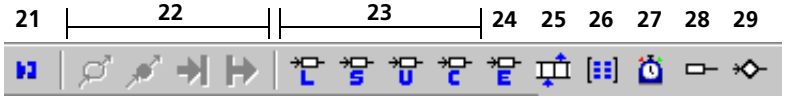
1. Undo
2. Redo

3. arithmetische Operatoren (Addition, Subtraction, Multiplication, Division, Modulo)
4. logische Operatoren (And, Or, Not)
5. Vergleichsoperatoren (Greater, Less, Less or Equal, Greater or Equal, Equal, Not Equal)



6. Abs (liefert den Absolutbetrag des Eingangswertes)
7. Max (liefert den größten Eingangswert)
8. Min (liefert den kleinsten Eingangswert)
9. Between (prüft, ob der Eingangswert zwischen den Begrenzungen liegt)
10. Negation (Vorzeichenumkehr)
11. MUX (Multiplex-Operator)
12. Case (Falloperator)
13. If-Then
14. If-Then-Else
15. While (Schleife)
16. Switch
17. Break (Unterbrechungsanweisung; legt die Rückkehr von einer Methode/einem Prozess fest)

18. Kombikästchen für die Auswahl der Anzahl Operatoreingänge
19. Kombikästchen für die Auswahl der Ansicht
20. Kombikästchen für den Zoomfaktor



21. Connect (schaltet den Verbindungsmodus ein)

22. Elemente im Zustandsautomaten (State, Junction, Input, Output); nur verfügbar, wenn Sie einen Zustandsautomaten bearbeiten
23. Variablen (Logic, Signed Discrete, Unsigned Discrete, Continuous)
24. Enumeration
25. Array
26. Matrix
27. Systemparameter `dT`

> **Hinweis**
>
> *Der Name* `dT` *ist für den Systemparameter reserviert. Sie können kein anderes Element mit dem Namen* `dT` *erzeugen; da Groß- und Kleinschreibung nicht unterschieden wird, sind auch die Namen* `DT`*,* `dt` *und* `Dt` *reserviert.*

28. Continuous Parameter
29. Implementation Cast

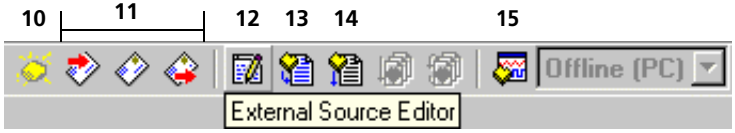
30. Kennlinien und -felder (Distribution, One D Table Parameter, Two D Table Parameter)
31. Kombikästchen zur Auswahl des Kennlinien-/feldtyps
32. Ressource
33. Messages (Receive, Send Receive, Send; nur für Module verfügbar)
34. Literale (String, True, False, 0.0, 1.0)
35. Self (Referenz auf das aktuelle Objekt selbst)

36. Hierarchy (fügt einen Hierarchieblock ein)
37. Comment (fügt einen Kommentar in den Zeichenbereich ein)
38. Generate Code (Code erzeugen)
39. Open Experiment for selected Experiment Target (generiert Code und öffnet das Offline-Experiment)

Nicht nummerierte Schaltflächenelemente sind im Blockdiagrammeditor immer deaktiviert.

5.2.3 Schaltflächenleisten in C-Code- und ESDL-Editor

Die Schaltflächenelemente (1) bis (9) sind sowohl im C-Code-Editor als auch im ESDL-Editor vorhanden; Schaltfläche (7) ist im C-Code-Editor allerdings deaktiviert. Sie entsprechen den Schaltflächenelementen (23) bis (31) im Blockdiagrammeditor (s. Seite 92).

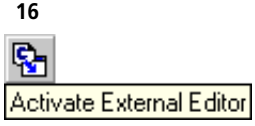


Die Schaltflächen (10), (11), (13) und (15) sind ebenfalls in beiden Editoren vorhanden. Sie entsprechen den Schaltflächen (32), (33), (38) und (39) im Blockdiagrammeditor (s. Seite 92).

12. External Source Editor (nur im C-Code-Editor; öffnet den Editor für externe Quellen)

14. Compile Generated Code (*nur* im C-Code-Editor aktiviert)

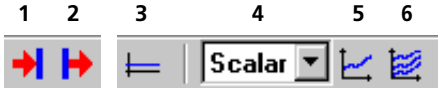
Nicht nummerierte Schaltflächenelemente sind im C-Code- und ESDL-Editor immer deaktiviert.



16. Activate External Editor (in der Fußzeile des jeweiligen Editors; aktiviert die Möglichkeit, den Code der Komponente in einem beliebigen Texteditor außerhalb von ASCET zu bearbeiten)

5.2.4 Schaltflächenleisten in den Editoren für CT-Blöcke

CT-Blöcke können in C-Code, ESDL oder als Blockdiagramm spezifiziert werden. Folgende Schaltflächenelemente sind in allen drei Editoren vorhanden:



1. Input
2. Output
3. Constant
4. Kombikästchen für die Auswahl des Parametertyps
5. One D Table Parameter (Kennlinie)
6. Two D Table Parameter (Kennfeld)

Die Schaltflächenelemente (7) und (9) entsprechen den Elementen (38) und (39) im Blockdiagrammeditor (s. Seite 92).

8. Compile Generated Code (nur im C-Code-Editor für CT-Blöcke aktiviert)

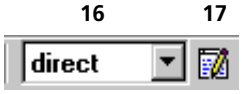
Nicht nummerierte Schaltflächenelemente sind in den Editoren für CT-Blöcke immer deaktiviert.

Die folgenden Schaltflächen sind nur im ESDL- oder C-Code-Editor für CT-Blöcke vorhanden:

10. Discrete State (diskreter Zustand)
11. Continuous State (kontinuierlicher Zustand)
12. Steplocal Variable
13. Parameter (Typ wird mit Kombikästchen (4) festgelegt)
14. Dependent Parameter (abhängiger Parameter; Typ wird mit Kombikästchen (4) festgelegt)

15. Activate External Editor (in der Fußzeile des Editors; aktiviert die Möglichkeit, Code in einem externen Texteditor zu bearbeiten)

Die folgenden Schaltflächenelemente sind nur im C-Code-Editor für CT-Blöcke vorhanden:



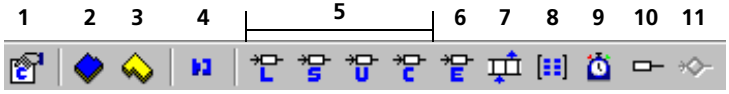
16. Kombikästchen zur Auswahl von direktem bzw. nicht-direktem Durchgriff
17. External Source Editor (öffnet den Editor für externe Quellen)

Die folgenden Schaltflächenelemente sind nur im Blockdiagrammeditor für CT-Blöcke vorhanden:

18. arithmetische Operatoren (Addition, Subtraction, Multiplication, Division)
19. Kombikästchen für die Auswahl der Anzahl Operatoreingänge
20. Kombikästchen für die Auswahl der Ansicht
21. Kombikästchen für den Zoomfaktor

22. Connect (schaltet den Verbindungsmodus ein)
23. Global Parameter (Typ wird mit Kombikästchen (4) festgelegt)
24. Hierarchy
25. Comment (fügt einen Kommentar in den Zeichenbereich ein)

5.2.5 Schaltflächenelemente im Projekteditor



1. Specify Code Generation Settings (öffnet das Fenster „Settings“ mit den Einstellungen für die Codegenerierung)

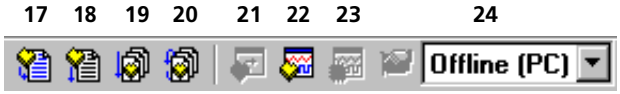
2. Edit Data (öffnet den datreneditor für das Projekt)
3. Edit Implementation (öffnet den Implementierungseditor für das Projekt)

Die Schaltflächenelemente (4) bis (11) entsprechen den Schaltflächenelementen (21) sowie (23) bis (29) im Blockdiagrammeditor (s. Seite 92).

Die Schaltflächenelemente (12), (13), (15) und (16) entsprechen den Schaltflächenelementen (30), (31), (36) und (37) im Blockdiagrammeditor (s. Seite 92).

14. Send Receive Message

Nicht nummerierte Schaltflächen sind im Projekteditor immer deaktiviert.



Die Schaltfläche (17) entspricht der Schaltfläche (38) im Blockdiagrammeditor (s. Seite 92).

18. Compile Generated Code
19. Build Executable Code (generiert ausführbaren Code)
20. Rebuild Executable Code (komplette Neugenerierung des ausführbaren Codes)
21. Transfer Project to selected Experiment Target (transferiert das Projekt zu dem im Kombikästchen (25) ausgewählten Experiment)

    Die Schaltfläche ist nur aktiviert, wenn in den Targetoptionen das Target `ES1130` oder `ES1135` und unter (25) der Eintrag `INCA` oder `INTECRIO` gewählt wurde.
22. Open Experiment for selected Experiment Target (generiert Code und öffnet das unter (25) gewählte Experiment)

    Die Schaltfläche ist *nicht* aktiviert, wenn im Kombikästchen (25) der Eintrag `INCA` oder `INTECRIO` gewählt wurde.
23. Reconnect to Experiment of selected Experiment Target (stellt die Verbindung zu dem auf dem im Kombikästchen (25) gewählten Target laufenden Experiment wieder her)

    Die Schaltfläche ist *nicht* aktiviert, wenn in den Targetoptionen das Target `PC` oder im Kombikästchen (25) der Eintrag `INCA` gewählt wurde.

24. Experiment Target (Auswahl des Experiments)

    Die verfügbare Auswahl hängt davon ab, welches Target Sie in den Targetoptionen ausgewählt haben und welche anderen Programme (z.B. INCA oder INTECRIO) auf Ihrem Rechner installiert sind.

25. Frischt die Anzeige im Register „Formulas“ auf.
26. Hinzufügen fehlender Formeln (Add missing Formulas)

    Findet undefinierte Formeln in der aktiven Implementierung und fügt für jeden gefundenen Formelnamen eine lineare Formel mit dem Verhalten der Identität hinzu.

5.2.6 Schaltflächenelemente im Offline-Experiment

1. Exit to Component (beendet das Experiment und ruft den Komponenteneditor auf; das Aussehen der Schaltfläche ist abhängig von der Komponente, mit der experimentiert wird)
2. Load Environment (lädt eine Experimentierumgebung, d.h. vordefinierte Mess- und Verstellfenster mit zugewiesenen Größen)
3. Save Environment (speichert die aktuelle Experimentierumgebung)
4. Save Environment As (speichert die aktuelle Experimentierumgebung unter einem beliebigen Namen)
5. Stop Offline Experiment (beendet das Experiment)
6. Start Offline Experiment (startet das Experiment)
7. Pause Offline Experiment (unterbricht das Experiment; es kann mit den Schaltflächen (6) oder (8) fortgesetzt werden)
8. Step Offline Experiment (schrittweise Abarbeitung des Experiments)
9. Eingabefeld für die Schrittweite

10. Open CT Solver (öffnet ein Fenster, in dem Sie die Integrationsmethode konfigurieren können)

    Diese Schaltfläche ist *nur* vorhanden, wenn Sie mit einem CT-Block oder einem hybriden Projekt experimentieren.

11. Open Event Generator

    Diese Schaltfläche ist *nicht* vorhanden, wenn Sie mit einem CT-Block experimentieren.

12. Open Event Tracer (öffnet das Fenster „Event Tracer" zur Erstellung von Datenprotokollen)
13. Open Data Generator
14. Open Data Logger
15. Update Dependent Parameters (aktualisiert die Werte abhängiger Parameter)
16. Expand / Collapse Window (blendet die Komponentenanzeige ein/aus)
17. Always on top (hält das Fenster „Physical Experiment" immer im Vordergrund)
18. Navigate down to child component (zeigt die ausgewählte aufgenommene Komponente)
19. Navigate up to parent component (zeigt die übergeordnete Komponente)

5.3 Bedienung per Tastatur

Bei der Entwicklung von ASCET wurde großer Wert auf eine einfache Bedienung gelegt. Einzeltasten haben dabei Vorrang vor den Funktionstasten <F1> bis <F12>, die wiederum kombinierten Tastenfunktionen mit <CTRL> (auch <STRG>) und <ALT> bevorzugt wurden. Eine vollständige Übersicht der aktuell benutzbaren Tastaturbefehle steht Ihnen jederzeit mit <CTRL> + <F1> zur Verfügung.

5.3.1 Allgemeine Tastaturbedienung

In dieser Tabelle finden Sie die wichtigsten Tasten und Tastenkombinationen für die Bedienung von ASCET. Eine vollständige Liste aller Tastaturbefehle finden Sie im Kapitel „Bedienung über Tastatur“ auf Seite 235.

Taste	**Funktion**
<F2>	In den Editiermodus wechseln (z.B. bei Namen)
<SHIFT> + <F10>	Kontextmenü für ausgewähltes Element öffnen (rechte Maustaste)
<ALT>	Hauptmenü aktivieren
<ALT> + <F4>	Aktuelles Arbeitsfenster schließen; im Komponentenmanager: ASCET beenden
<ALT> + <F6>	Zwischen geöffneten ASCET Fenstern wechseln
<ALT> + <LEER>	Systemmenü für Anwendungsprogrammfenster öffnen
<ALT> + <TAB>	Zwischen geöffneten Anwendungsprogrammen wechseln
<CTRL> + <F1>	Tastaturbelegung anzeigen
<CTRL> + <A>	Alle Objekte (z.B. in einer Liste) auswählen
<CTRL> + <C>	Kopieren in Zwischenablage
<CTRL> + <V>	Einfügen aus Zwischenablage
<CTRL> + <X>	Ausschneiden und einfügen in Zwischenablage
<CTRL> + <Y>	Letzte Aktion erneut ausführen (nur Editoren)
<CTRL> + <Z>	Letzte Aktion rückgängig machen (nur Editoren)
<PFEIL UNTEN> (↓), <PFEIL OBEN> (↑), <PFEIL LINKS> (←), <PFEIL RECHTS> (→)	Tabellenelement oder Listenelement mit Pfeiltasten anfahren, mit <PFEIL RECHTS> auch Verzeichnis aufklappen, mit <PFEIL LINKS> auch Verzeichnis zuklappen,
<EINGABE>	Eingabe bestätigen und Eingabemodus beenden; Verzweigungen öffnen oder schließen
<ENTF>	Ausgewählten Eintrag löschen
<ESC>	Eingabe abbrechen, Änderungen verwerfen
<LEER>	Tabellen- oder Listenelement selektieren oder aktive Auswahl aufheben
<TAB>	Auswahl (Fokus) auf das nächste Element (Option) eines Fensters setzen (<SHIFT> + <TAB>: andere Richtung)

5.3.2 Tastaturbedienung nach Windows-Vereinbarungen

Für die allgemeine Bedienung von ASCET, wie z.B. das Navigieren in Menüs oder das Aktivieren eines bestimmten Fensters, gelten die WINDOWS®-Vereinbarungen.

Ein im Menü unterstrichener Buchstabe gemeinsam mit der <ALT>-Taste gedrückt, ruft den entsprechenden Befehl auf. Einen hier untergeordneten Menübefehl können Sie aktivieren, indem Sie den unterstrichenen Buchstaben gemeinsam mit der <SHIFT>-Taste drücken.

Um also z.B. im Komponentenmanager das Menü **File** mit einem Tastaturbefehl aufzurufen, drücken Sie die Tastenkombination <ALT> + <F>.

Innerhalb der Arbeitsfenster erfolgt das Weiterschalten zum nächsten Fensterelement bzw. zum nächsten Listenfeld mit der <TAB>-Taste (in der Reihenfolge von links oben nach rechts unten). Alternativ kann mit der <ALT>-Taste und dem unterstrichenen Zeichen (Buchstabe oder Zahl) des Feld- oder Listentitels zum entsprechenden Listenfeld weitergeschaltet werden.

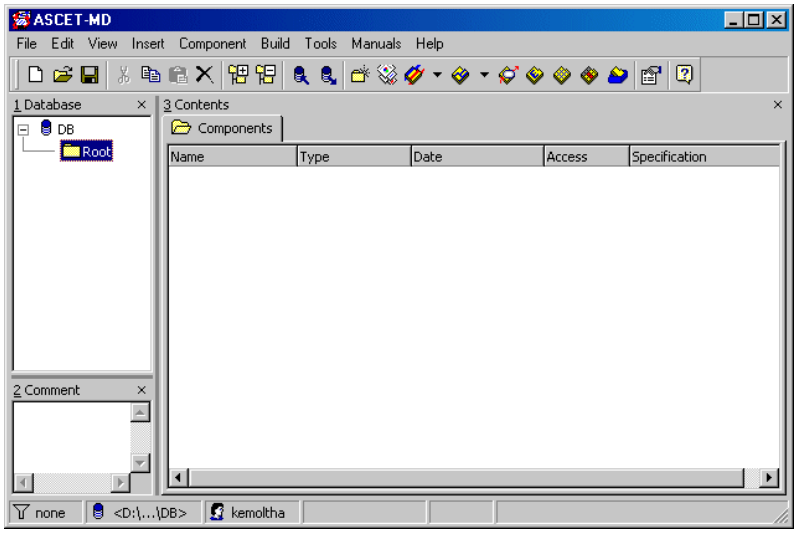
Die Pfeiltasten erlauben in Listenfeldern das Springen zum nächsten Listeneintrag. Die Mehrfachauswahl von Einträgen ist dabei über gleichzeitiges Drücken der <SHIFT>-Taste möglich.

Das Wechseln zwischen Fenstern unterschiedlicher Anwendungen erfolgt entsprechend den WINDOWS®-Vereinbarungen mit der Tastenkombination <ALT> + <TAB>. Hierbei werden alle Subsysteme von ASCET, wie z.B. die Komponenteneditoren, die Experimentierumgebung oder auch Implementierungs- oder Dateneditor, als eigenständige Anwendung behandelt.

Wird <TAB> innerhalb des Fensters für andere Funktionen benötigt, wie z.B. beim Editieren von Text, erfolgt das Weiterschalten zum nächsten Fensterelement mit der Tastenkombination <CTRL> + <TAB>.

Das Wechseln zwischen einzelnen Registern in einem Fenster oder Feld, z.B. im Komponentenmanager im Feld „3 Contents", erfolgt konform mit den MS-WINDOWS®-Vereinbarungen mit der Tastenkombination <CTRL> + <TAB>.

Innerhalb eines Fensters kann mit der unterstrichenen Zahl des Feld- oder Listentitels und Drücken der <ALT>-Taste zum entsprechenden Listenfeld weitergeschaltet werden. So würde z.B. <ALT> + <2> das Textfeld „2 Comment“ aktivieren.



5.4 Bedienung per Maus

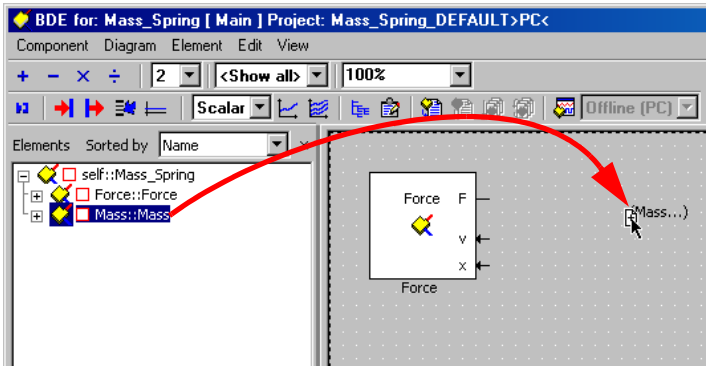
In Büro oder Labor können Sie zur komfortableren Bedienung von ASCET die Maus einsetzen. Die Bedienung erfolgt entsprechend den WINDOWS®-Vereinbarungen.

Mehrfachauswahl von Einträgen ist über Drücken der <SHIFT>- bzw. der <CTRL>-Taste möglich.

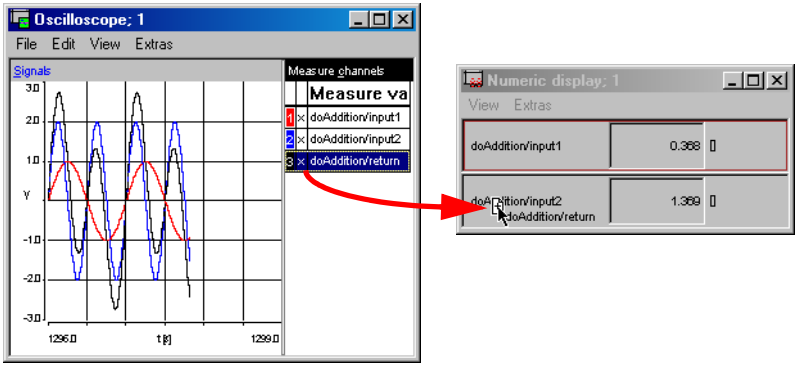
Über das Anklicken der Fensterelemente mit der rechten Maustaste werden dem jeweiligen Element entsprechende Kontextmenüs aufgerufen.

5.4.1 Drag & Drop

Beim Zeichnen von Blockdiagrammen oder dem Einrichten von Mess- und Verstellfenstern kann mit Drag & Drop (Anklicken des Elements mit der linken Maustaste, Maustaste gedrückt halten und das Element mit Maus in das Zielfenster/-feld verschieben) gearbeitet werden.



Ebenso einfach kann eine Größe von einem Fenster in ein anderes kopiert werden. Unmögliche Befehle, wie etwa das Kopieren einer Verstellgröße in ein Messfenster, werden dabei ignoriert.



5.5 Hierarchiebäume

Informationen werden in ASCET oft hierarchisch gegliedert als Baumstruktur angezeigt, wie zum Beispiel der Inhalt einer Datenbank. Um alle Verzweigungen und den ganzen Inhalt einer solchen Baumstruktur zu sehen, ist es not-

wendig, die Verzweigungen zu öffnen oder zu schließen. ASCET bietet Ihnen die Möglichkeit, automatisch mehrere Verzweigungen oder gezielt einzelne Teilbäume zu öffnen.

Mehrere Verzweigungen automatisch öffnen:

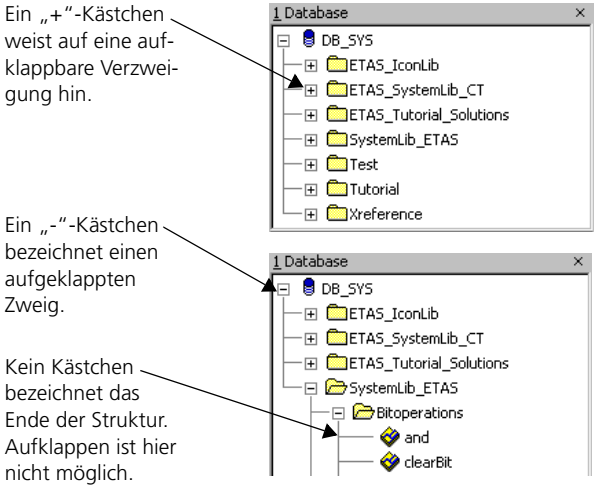
- Um alle Verzweigungen auf einmal zu öffnen oder zu schließen, stehen Ihnen die Menübefehle **View → Expand All** und **View → Collapse All** bzw. die Schaltflächen in der Schaltflächenleiste zur Verfügung (siehe „Schaltflächen im Komponentenmanager“ auf Seite 89).

Einzelne Teilbäume öffnen:

- Klicken Sie zum Öffnen der Verzweigung mit der Maus auf das „+“-Kästchen neben dem entsprechenden Eintrag bzw. verwenden Sie die Taste <+>. Ein zweiter Klick auf dasselbe Kästchen bzw. die Taste <-> schließt den Zweig wieder.

Oder

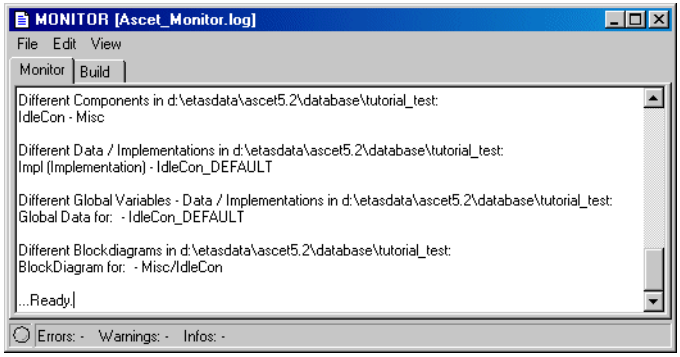
- Fahren Sie den Eintrag mit der <PFEIL UNTEN / OBEN>-Taste an und verwenden Sie dann zum Aufklappen der Verzweigung die <PFEIL RECHTS>-Taste. Zum Schließen benutzen Sie die <PFEIL LINKS>-Taste.



5.6 Unterstützende Funktionen

5.6.1 Monitorfenster

Im Monitorfenster (siehe Kapitel 2.4 im ASCET-Benutzerhandbuch) wird protokolliert, welche Arbeitsschritte von ASCET durchgeführt werden. Alle Vorgänge, auch aufgetretene Fehler und Hinweise, werden aufgezeichnet. Sobald ein Ereignis protokolliert wird, erscheint das Monitorfenster im Vordergrund.



Neben der Informationsanzeige bietet Ihnen das Monitorfenster zusätzlich noch die Funktionalität eines Editors.

- Das Anzeigefeld im Register „Monitor“ des Monitorfensters ist frei editierbar. So können eigene Anmerkungen und Kommentare zu den ASCET-Meldungen hinzugefügt werden.
- Die ASCET-Meldungen können zusammen mit Ihren Kommentaren als Textdatei abgespeichert werden.
- Andere, bereits abgespeicherte ASCET-Textdateien können geladen und so bestimmte Arbeitsschritte miteinander verglichen werden.

5.6.2 Tastaturbelegung

Eine Übersicht der aktuell benutzbaren Tastaturbefehle steht Ihnen jederzeit mit <CTRL> + <F1> zur Verfügung.

5.6.3 Handbuch und Onlinehilfe

Wenn bei der Installation nichts anderes angegeben wurde, liegt das komplette ASCET-Handbuch in elektronischer Form vor und kann jederzeit am Bildschirm geöffnet werden, z.B. über die Funktionen im Menü **Manuals** des Komponentenmanagers. Die einzelnen Bände sind im Ordner `Etas\ETAS-Manuals\ASCET5.2` unter den Namen **`ASCET V5.2 Schnellein-`**

stieg.pdf, **ASCET V5.2 Handbuch.pdf**, **ASCET V5.2 Referenz.pdf** abgelegt. Gedruckte Exemplare der Handbücher können Sie hier bestellen:

http://www.etasgroup.com/order_manual/ascet

Über den Index, Volltextsuche und Hypertextlinks sind relevante Stellen schnell und komfortabel zu finden.

Die Online-Hilfe kann mit der <F1>-Taste aufgerufen werden. Sie ist im Ordner `ETAS\ASCET5.2\Help` abgelegt.

6 Tutorial

Das Tutorial richtet sich hauptsächlich an den ASCET-Neueinsteiger. Anhand von Beispielen erlernen Sie die Bedienung von ASCET. Der gesamte Lerninhalt wird dabei in einzelne kurze, aufeinander aufbauende Lernabschnitte unterteilt. Bevor Sie mit dem Tutorial arbeiten, sollten Sie das Kapitel „ASCET – Grundlagen“ auf Seite 51 durchgearbeitet haben.

6.1 Ein einfaches Blockdiagramm

In ASCET verwenden Sie Komponenten wie z. B. Klassen und Module als Hauptkomponenten Ihrer Anwendungen. Sie können entweder vordefinierte Komponenten nutzen, die zusammen mit ASCET geliefert werden oder schon früher entwickelt wurden, oder sich Ihre eigenen Komponenten erstellen, und genau das werden Sie im folgenden Tutorium tun.

In ASCET werden Komponenten gewöhnlich grafisch vorgegeben. Nach Vorgabe aller Komponenten werden diese zu einem Projekt zusammengestellt, das die Grundlage eines ASCET Softwaresystems darstellt. Ein Softwaresystem besteht aus einem C-Code, der aus der grafischen Modellbeschreibung generiert wurde und der auf einem Mikrocontroller oder auf einem experimentellen Zielrechner ausgeführt werden kann.

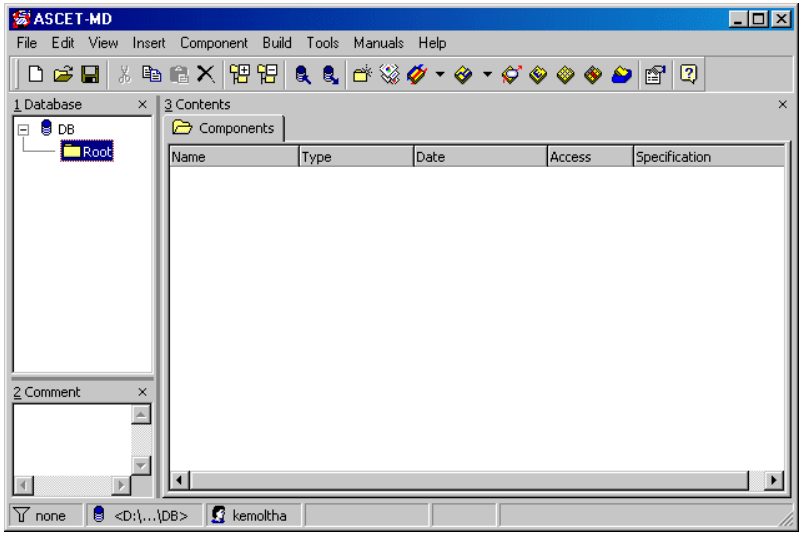
6.1.1 Vorbereitende Schritte

Ehe Sie beginnen können, müssen Sie eine Datenbank öffnen, in der Sie arbeiten möchten. Alle Komponenten des vorliegenden Tutoriums werden in dieser Datenbank abgelegt, und folglich müssen Sie dies lediglich ein einziges Mal tun.

Alle Komponenten und Projekte für das vorliegende Tutorium befinden sich in einem Ordner mit der Bezeichnung `ETAS_Tutorial_Solutions` in der Datenbank `tutorial`. Folglich brauchen Sie nicht alle hier beschriebenen Komponenten selbst zu spezifizieren.

Es ist jedoch ratsam, dass Sie zumindest die Komponenten der Lektionen 1, 3 und 4 spezifizieren, damit Sie sich im Umgang mit ASCET etwas Praxis aneignen.

Beim Start von ASCET wird der Komponentenmanager geöffnet. Dabei wird die Datenbank geladen, die zuletzt geöffnet war.



Es empfiehlt sich der besseren Übersicht wegen, für das Tutorium eine eigene Datenbank zu verwenden.

Anlegen einer neuen Datenbank:

- Wählen Sie im Komponentenmanager den Befehl **File** → **New Database**

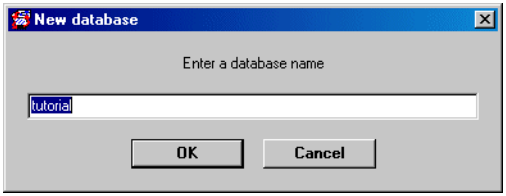
oder

- klicken Sie auf die Schaltfläche **New**

oder

- drücken Sie <CTRL> + <N>.

  Das Fenster „New database“ erscheint:



- Tragen Sie den Namen `tutorial` ein.
- Klicken Sie auf **OK**.

  Die neue, bis auf den Datenbanknamen und den Ordner `Root` noch leere Datenbank wird geöffnet.

Öffnen einer Datenbank:

Wenn die Datenbank `tutorial` schon existiert, gehen Sie wie folgt vor:

- Wählen Sie im Komponentenmanager **File** → **Open Database**

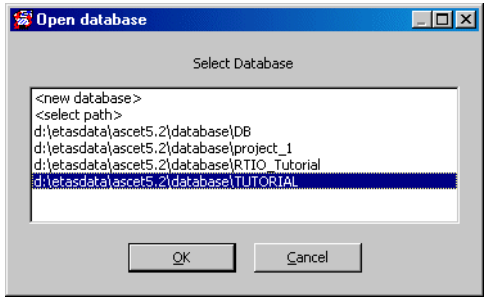
oder

- klicken Sie auf die Schaltfläche **Open**

oder

- drücken Sie <CTRL> + <O>.

  Das Dialogfeld „Open Database“ öffnet sich. Es enthält eine Übersicht der Datenbanken im aktuellen Datenbankpfad.



- Ist die Datenbank `tutorial` in der Liste enthalten, wählen Sie deren Namen und klicken Sie auf **OK**.

  Durch den Komponentenmanager wird der Inhalt der Datenbank `tutorial` zur Anzeige gebracht.
- Ist die Datenbank `tutorial` an anderer Stelle abgelegt, wählen Sie den Eintrag `<select path>` und klicken Sie auf **OK**.

- Wählen Sie anschließend im Fenster „Select database path“ die Datenbank aus und klicken Sie auf **OK**.

Der erste Schritt zum Erstellen Ihrer eigenen Komponenten besteht im Erstellen eines Hauptordners mit dem Namen `Tutorial` sowie eines Unterordners mit dem Namen `LessonN` für jede Lektion.

Erstellen neuer Ordner:

- Wählen Sie im Feld „1 Database“ den Namen der Datenbank.
- Wählen Sie **Insert** → **Folder**

oder

- klicken Sie auf die Schaltfläche **Insert Folder**

oder

- drücken Sie sie Taste <INSERT>.

  Im Feld „1 Database“ erscheint ein neuer Hauptordner mit der Bezeichnung `Root`.
- Bearbeiten Sie den Namen des Hauptordners und nennen Sie ihn `Tutorial`. Sie können den hervorgehobenen Namen überschreiben und dann <ENTER> drücken.
- Wählen Sie den Ordner `Tutorial` aus.
- Wählen Sie erneut **Insert** → **Folder**.

  Im Feld „1 Database“ wird ein neuer Ordner mit dem Namen `Folder` erzeugt.
- Bearbeiten Sie den Namen des neuen Ordners und bezeichnen Sie ihn mit `Lesson1`.

Alle von Ihnen im Rahmen dieses Tutoriums erstellten Komponenten werden im Ordner `LessonN` abgelegt. Sie sollten für jede Lektion einen neuen Ordner erstellen. Jede Datenbank hat zumindest einen Hauptordner, und dieser kann eine beliebige Anzahl Unterordner haben.

Hinweis

In ASCET 5.2 müssen alle Ordner- und Komponentenbezeichnungen sowie die Bezeichnungen von Variablen und Methoden, die sie enthalten, dem ANSI C-Standard entsprechen.

Sie können dann fortfahren, indem Sie Ihre erste Komponente im Ordner `Lesson1` erstellen.

Erstellen einer Komponente:

- Klicken Sie im Feld „1 Database“ auf den Ordner `Lesson1`.
- Wählen Sie **Insert** → **Class** → **Block Diagram**.

  Im Feld „1 Database“ erscheint unter dem Ordner `Lesson1` eine neue Komponente mit der Bezeichnung `Class_Block_Diagram`. Diese Komponente ist von der Art *class*, die in ASCET häufig zur Anwendung kommt.
- Ändern Sie die Komponentenbenennung in `Addition`.

6.1.2 Spezifizieren einer Klasse

Nach dem Erstellen der neuen Komponente im Ordner `Tutorial\Lesson1` können Sie deren Funktionalität spezifizieren. Zuerst definieren Sie die Schnittstelle für die Komponente, d.h. ihre Methoden, Argumente und Rückgabewerte. Dann zeichnen Sie ein Blockdiagramm, in dem vorgegeben wird, was die Komponente bewirkt.

Spezifizieren der Funktionalität einer Komponente:

- Wählen Sie im Feld „1 Database“ die Komponente `Addition`.
- Um die Komponente zu öffnen, wählen Sie **Component** → **Edit**

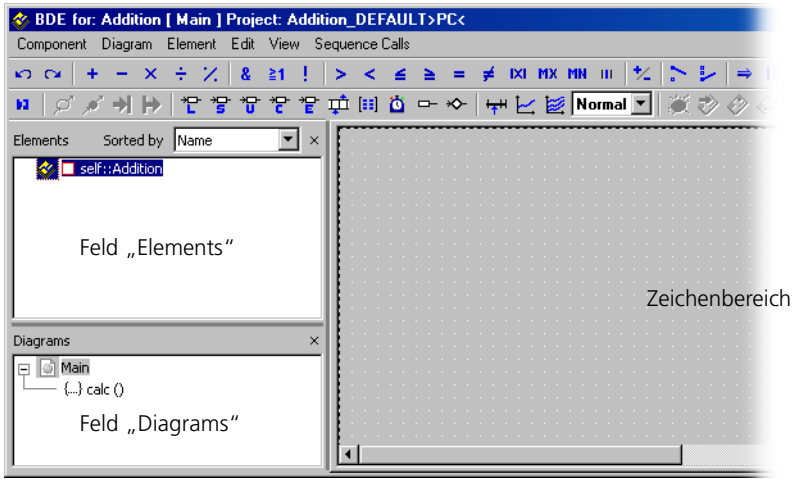
oder

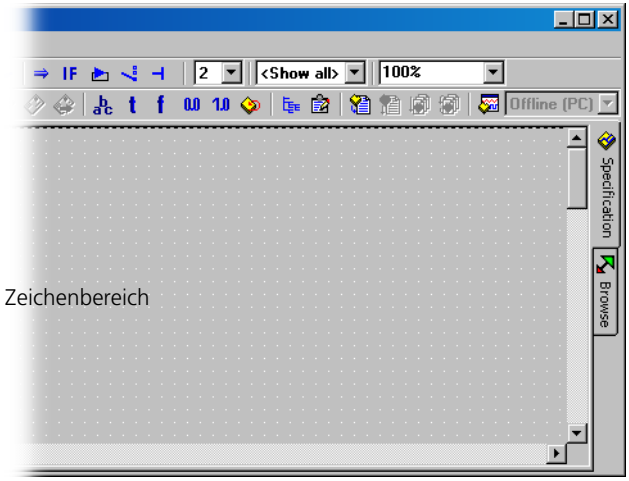
- doppelklicken Sie auf die Komponente

oder

- drücken Sie <RETURN>.

  Der Blockdiagrammeditor wird geöffnet. Dies ist das Hauptfenster, in dem die Funktionalität von Komponenten vorgegeben werden kann.





- Wählen Sie im Feld „Diagrams“ die Methode `calc`. Diese Methode wird standardmäßig erstellt.
- Wählen Sie **Diagram** → **Rename**

oder

- drücken Sie <F2>.

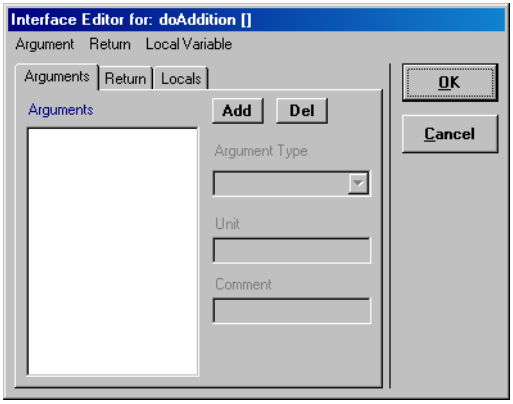
  Der Name der Methode `calc` wird hervorgehoben.

- Ändern Sie den Namen der Methode in `doAddition`.
- Wählen Sie **Diagram** → **Edit**

oder

- doppelklicken Sie auf die Methodenbenennung.

  Der Schnittstelleneditor für die Methode wird geöffnet.



Jede Klasse benötigt zumindest eine Methode. Methoden in ASCET ähneln Methoden bei der objektorientierten Programmierung oder Funktionen in prozeduralen Programmiersprachen. Eine Methode kann mehrere Argumente und einen Rückgabewert haben (diese sind alle optional). Argumente werden verwendet, um Daten an eine Komponente zu übertragen. Rückgabewerte dienen zum Zurückgeben der in einer Komponente berechneten Ergebnisse nach „außen".

Zum Spezifizieren der Schnittstelle der Methode können Sie unter Verwendung des Schnittstelleneditors zwei Argumente des Typs `continuous` sowie einen Rückgabewert hinzufügen.

Spezifizieren der Schnittstelle einer Methode:

- Wählen Sie im Schnittstelleneditor **Argument** → **Add**.

  Im Feld „Arguments" erscheint ein neues Argument mit der Bezeichnung `arg`.

- Ändern Sie den Namen des Arguments in `input1`.
- Fügen Sie ein weiteres Argument mit der Benennung `input2` hinzu.

  Der Datentyp der Argumente wird standardmäßig auf *continuous* (oder kurz *cont*) eingestellt, und genau das ist für das Beispiel beabsichtigt.
- Wechseln Sie in das Register „Return“ des Schnittstelleneditors.
- Aktivieren Sie de Option **Return Value**.

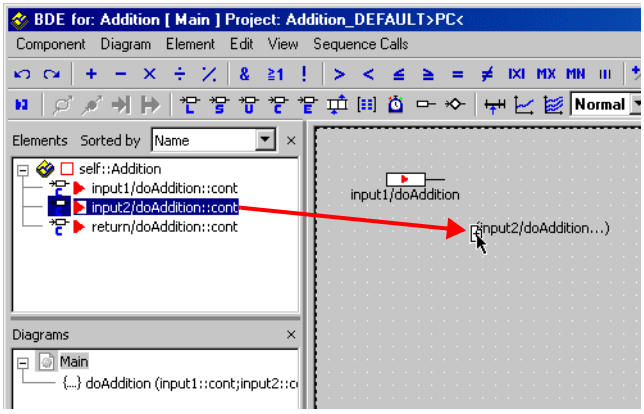
  Die Art des Rückgabewertes wird standardmäßig ebenfalls auf *cont* gesetzt.
- Zum Schließen des Schnittstelleneditors klicken Sie auf **OK**.

Die Bezeichnungen der Argumente und des Rückgabewertes der Methode `doAddition` erscheinen im Feld „Elements“ links vom Blockdiagrammeditor. Sie können nun die Funktionalität der Komponente durch Zeichnen eines Blockdiagramms spezifizieren.

Spezifizieren der Funktionalität der Komponente `Addition`:

- Ziehen Sie das erste Argument vom Feld „Elements“ in den Zeichenbereich des Blockdiagrammeditors.

  Das Symbol für das Argument erscheint im Zeichenbereich.



- Als nächstes fügen Sie das andere Argument und den Rückgabewert unter Verwendung der gleichen Drag-and-Drop-Funktion hinzu.

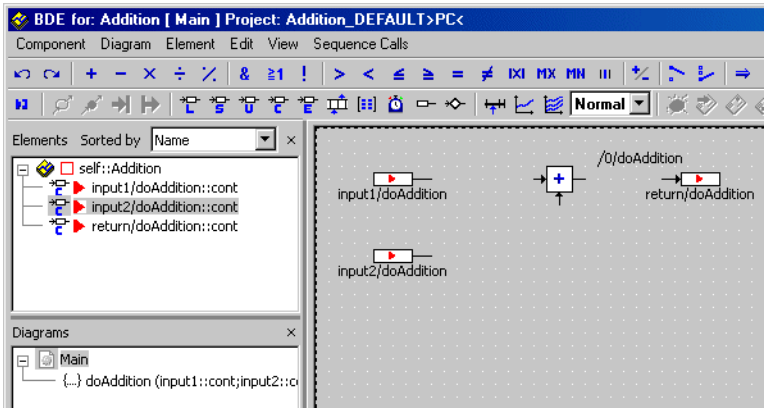
- Klicken Sie auf die Schaltfläche **Addition**.

  Der Mauszeiger wird mit einem Additionsoperator geladen.

- Klicken Sie in den Zeichenbereich zwischen die Symbole für das Argument und den Rückgabewert.

  Es erscheint ein Additionssymbol. Es verfügt standardmäßig über zwei (durch Pfeile angedeutete) Eingabepins und einen Ausgabepin. Der Ausgabepin ist rechts angeordnet.

Nun können Sie die Elemente und den Operator durch Ziehen an die entsprechende Stelle im Zeichenbereich anordnen. Als nächstes müssen Sie die Elemente zwecks Vorgabe des Informationsflusses miteinander verbinden.



Verbinden der Blockdiagrammelemente:

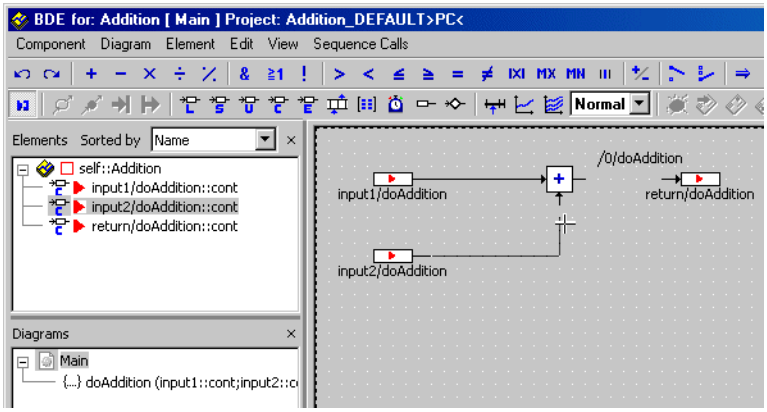
- Klicken Sie auf die Schaltfläche **Connect**.
- Sie können auch mit der rechten Maustaste in den Zeichenbereich (jedoch nicht auf eines der Elemente) klicken.

  Der Cursor ändert innerhalb des Zeichenbereichs seine Form zu einem Fadenkreuz.
- Klicken Sie auf den Ausgabepin des ersten Argumentsymbols, um eine Verbindung zu beginnen.

  Wenn Sie nun den Mauszeiger bewegen, wird eine Linie gezeichnet. Bei jedem Klicken in den Zeichenbereich bleibt die Linie bis zum betreffenden Punkt fixiert. Auf diese Weise können Sie den Verlauf der Verbindungslinie festlegen.

- Klicken sie auf den linken Eingang des Additionszeichens.

  Nun wird das Argumentsymbol mit dem Eingang des Additionszeichens verbunden.

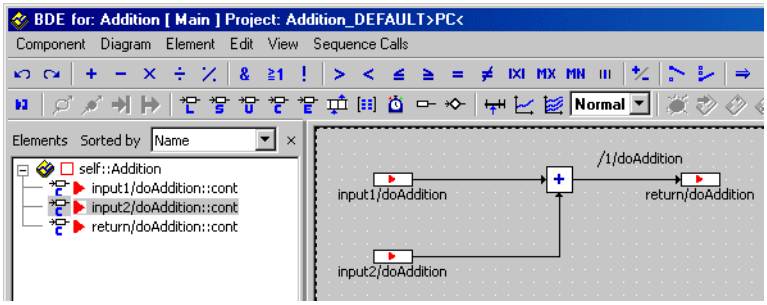


- Verbinden Sie das zweite Argumentsymbol mit dem anderen Eingang des Additionszeichens.
- Verbinden Sie das Rückgabewertsymbol mit dem Ausgang des Additionszeichens.

  Die Verbindung zwischen dem Additionsoperator und dem Rückgabewert erscheint als grüne Linie, wodurch angezeigt werden soll, dass die Reihenfolge dieser Operation festzulegen ist.

- Wählen Sie **Sequence Calls** → **Sequencing – Ignore Info**, um die Reihenfolge für die Addition automatisch festzulegen.

  Die Verbindung zwischen dem Additionsoperator und dem Rückgabewert erscheint als schwarze Linie.

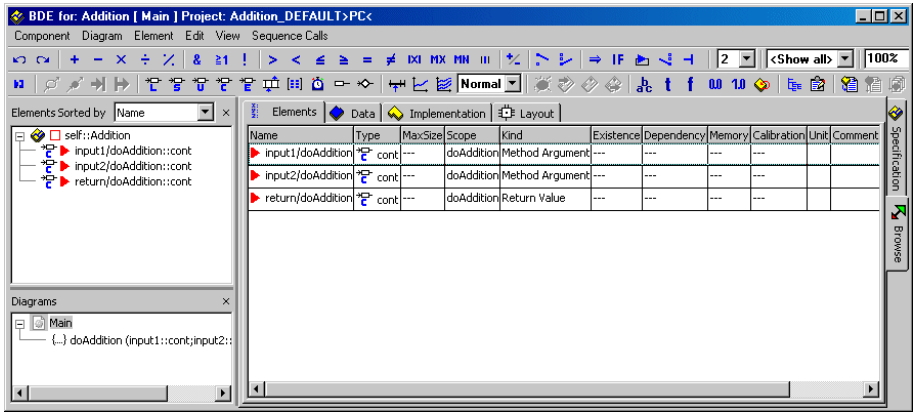


Damit ist die Spezifikation der Komponente abgeschlossen. Der letzte Schritt beim Bearbeiten Ihrer Komponente besteht in der Spezifikation des Layouts, d.h. die Art der Darstellung bei Verwendung innerhalb anderer Komponenten.

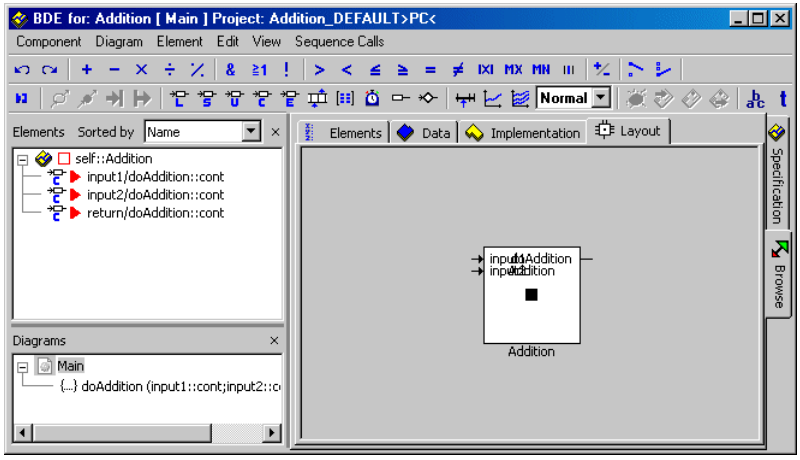
Bearbeiten des Layouts einer Komponente:

Zum Bearbeiten des Layouts stehen zwei Alternativen zu Verfügung:

- Wechseln Sie über die Lasche **Browse** in die Darstellung „Information/Browse“.

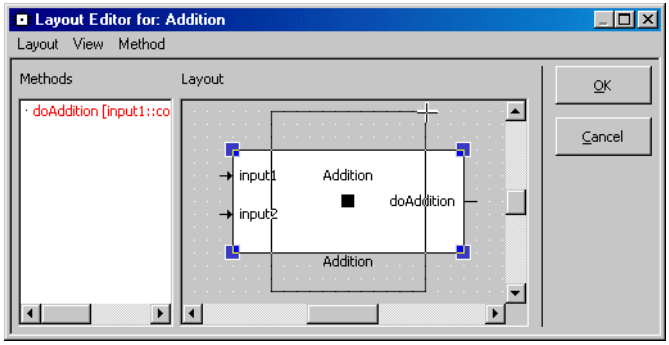


- Öffnen Sie mit einem Doppelklick innerhalb des Registers „Layout“ den Layout-Editor.



- Alternativ wählen Sie die Menüfunktion **Component** → **Edit Layout**.

  Der Layout-Editor öffnet sich.



- Um die Größe des Blocks zu ändern, klicken Sie darauf und ziehen dann die Anfasser bis zur gewünschten Größe.
- Zum Erstellen einer symmetrischen Darstellung ziehen Sie an den Pins der Argumente und des Rückgabewertes.
- Klicken Sie auf **OK**.

Nach Fertigstellung Ihrer Komponente wird als letzter Schritt dieser Lektion die Komponente in der Datenbank abgespeichert.

Abspeichern der Komponente `Addition`:

- Wählen Sie **Diagram** → **Store to Cache** und schließen Sie den Blockdiagrammeditor.
- Wählen Sie im Komponentenmanager **File** → **Save Database**

oder

- klicken Sie auf die Schaltfläche **Save**.

  Ihre Arbeit wird nur dann auf Festplatte gespeichert, wenn Sie diese Operation ausführen.
  Mit **Store to Cache** werden die Änderungen nur im Cachespeicher abgelegt. Folglich empfiehlt sich während des Fortschreitens der Arbeit ein häufiges Klicken auf **Save**.

  Durch Aktivieren der entsprechenden Benutzeroptionen (siehe Abschnitt „Grundlegende Optionen“ im ASCET-Benutzerhandbuch) können Sie die von Ihnen vorgenommenen Änderungen automatisch speichern.

Als optionale Zusatzübung können Sie jetzt dieselbe Funktionalität in *ESDL* modellieren (ESDL: *Embedded Software Description Language*). Wenn Sie diese Übung weiter verfolgen, lernen Sie die Umgang mit dem ESDL-Editor und die Handhabung des externen Quellcode-Editors.

Im ersten Schritt kopieren Sie die Schnittstelle des Moduls auf ein neues Modul vom Typ ESDL und benennen dieses um. Anschließend realisieren Sie die gewünschte Funktionalität entweder direkt im ESDL-Editor von ASCET oder Sie verwenden den externen Texteditor.

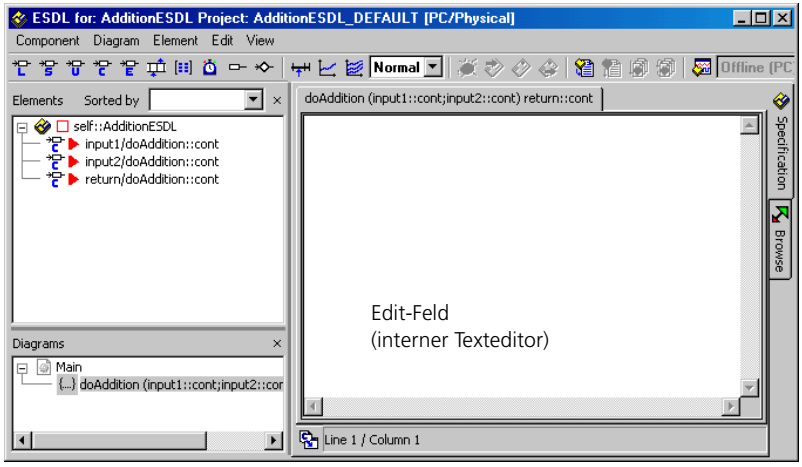
Kopieren und Spezifizieren der Komponenten(-Schnittstelle) `Addition`:

- Wählen Sie im Feld „1 Database“ des Komponentenmanagers die Komponente `Addition`.
- Wählen Sie **Component** → **Reproduce As** → **ESDL**.

  Eine Kopie der Komponente wird angelegt, sie hat den Namen `Addition1`.

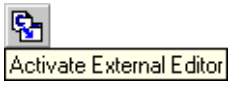
- Benennen Sie die neue Komponente um in `AdditionESDL`.
- Doppelklicken Sie im Feld „1 Database“ auf die neue Komponente `AdditionESDL`.

  Der ESDL-Editor wird für `AdditionESDL` geöffnet und stellt Ihnen Funktionalitäten zum Editieren zu Verfügung.



- Geben Sie jetzt in das Edit-Feld des internen Texteditors die Funktionalität ein:

  ```
  return input1 + input2;
  ```



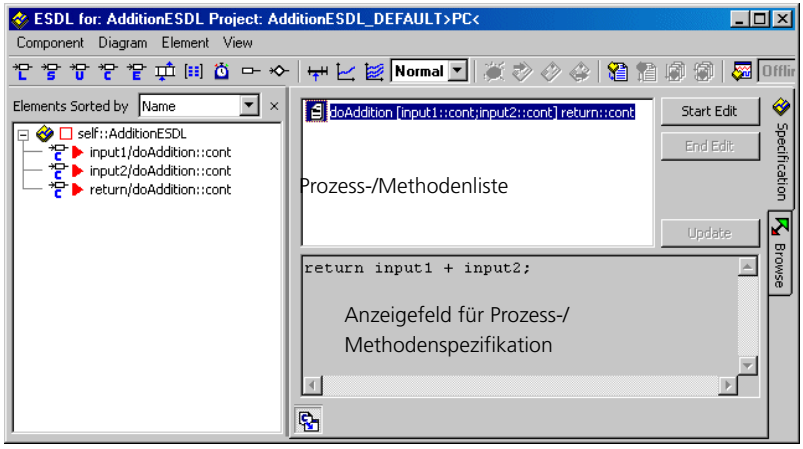
- Schalten Sie nun über die Schaltfläche **Activate External Editor** in den Modus des externen Editors um.

  Sie werden gefragt, ob Sie Ihre Änderungen speichern wollen.

- Bestätigen Sie mit **Yes**.

  Die Daten werden gespeichert, der ESDL-Editor wechselt in den Modus „Externer Editor“.

Der Editor stellt sich in diesem Modus verändert dar.



- Wählen Sie in der Prozess-/Methodenliste die zu spezifizierende Methode oder den Prozess.

  Jetzt erscheint im Anzeigefeld die vorher eingegebene Funktionalität, und die Schaltfläche **Start Edit** ist aktiviert.
- Aktivieren Sie mit **Start Edit** den externen Editor.

Hinweis

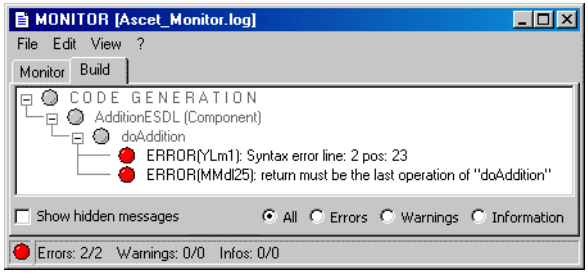
Beim Starten des externen Editors wird die in der Registrierungsdatenbank des Betriebssystem mit den Dateiendungen `*.c` *und* `*.h` *verbundene Applikation gestartet. Der Datentransfer erfolgt über temporäre Dateien, deshalb müssen Sie vor dem Schließen des Editors bzw. vor dem Übertragen in den ESDL-Editor die Dateisicherung anstoßen.*

Jetzt sind im ESDL-Editor die Schaltflächen **End Edit** und **Update** aktiviert, während **Start Edit** deaktiviert ist.

- Bearbeiten Sie die Funktionalität im externen Editor.
- Speichern Sie die Funktionalität im externen Editor.

- Klicken Sie im ESDL-Editor auf **Update**, um die Daten aus dem externen Editor zu übertragen.
- Beenden Sie anschließend mit **End Edit** die Verbindung mit dem externen Editor.

  Der externe Editor bleibt auch nach Klicken von **End Edit** geöffnet. Weitere Änderungen können aber nicht mehr in den ESDL-Editor übertragen werden.
- Klicken Sie erneut auf **Activate External Editor**, um den Modus des externen Editors zu beenden.
- Wählen Sie **Diagram** → **Analyze Diagram**, um den eingegebenen Code zu überprüfen. Fehler werden im ASCET Monitorfenster angezeigt.



6.1.3 Zusammenfassung

Nach Abschluss dieser Lektion sollten Sie in der Lage sein, die folgenden Aufgaben in ASCET auszuführen:

- Öffnen einer Datenbank
- Erstellen und Benennen eines Ordners
- Erstellen und Benennen einer Komponente
- Festlegen der Schnittstelle einer Methode
- Anordnen von Diagrammelementen im Zeichenbereich
- Verbinden von Diagrammelementen
- Bearbeiten des Layouts einer Komponente
- Umschalten zwischen der Spezifikations- und der Browse-Ansicht

- Abspeichern einer Komponente.
- Kopieren einer Komponenten-Schnittstelle.
- Benutzung des ESDL-Editors.
- Benutzung des externen Editors.

6.2 Experimentieren mit Komponenten

Nach dem Erstellen der Komponenten `Addition` oder `AdditionESDL` können Sie damit experimentieren. Durch Experimentieren können Sie sehen, wie die Komponente arbeitet, und zwar wie in einer realen Anwendung. Die Experimentierumgebung stellt verschiedene Werkzeuge bereit, mit denen die Werte von Eingaben, Ausgaben, Parametern und Variablen innerhalb einer Komponente dargestellt werden können.

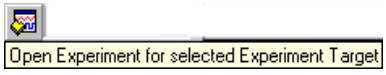
6.2.1 Aufrufen der Experimentierumgebung

Die Experimentierumgebung wird aus dem Blockdiagramm- oder ESDL-Editor aufgerufen. Dazu müssen Sie diesen zuerst mit der Komponente öffnen, mit der Sie experimentieren möchten.

Aufrufen der Experimentierumgebung:

- Öffnen Sie aus dem Komponentenmanager den Blockdiagrammeditor für Ihre `Addition` im Ordner `Tutorial\Lesson1`.
- Wählen Sie im Blockdiagrammeditor **Component → Open Experiment**

oder

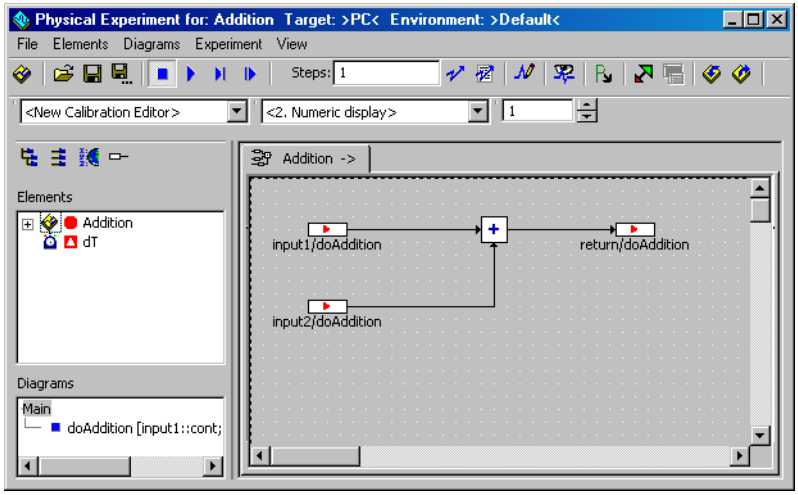


- klicken Sie auf die Schaltfläche **Open Experiment for selected Experiment Target**.

  Der Code für das Experiment wird erzeugt. ASCET analysiert das Modell in Ihrer Aufgabenstellung und generiert C-Code, der das Modell implementiert. Es ist möglich, einen speziellen Code für verschiedene Plattformen zu generieren.

  In diesem Beispiel werden einfach die standardmäßigen Einstellungen zum Generieren eines Codes für den PC verwendet.

  Nach dem Generieren und Kompilieren des Codes wird die Experimentierumgebung geöffnet.



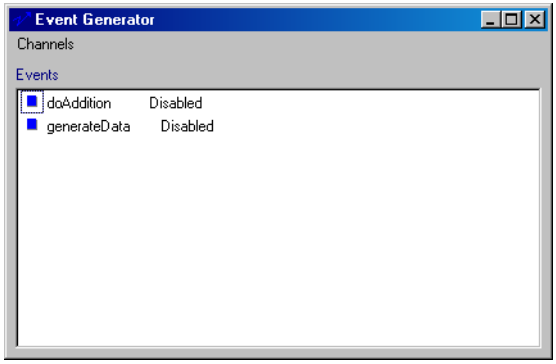
6.2.2 Einrichten der Experimentierumgebung

Ehe Sie mit dem Experimentieren beginnen können, müssen Sie die Umgebung einrichten, d.h. die Eingabewerte festlegen, die für das Experiment erzeugt werden, und bestimmen, wie Sie sich die Ergebnisse ansehen möchten. Dazu müssen Sie drei Schritte ausführen. Zuerst richten Sie den Ereignisgenerator, danach den Datengenerator und zum Schluss das Mess-System ein.

Einrichten des Ereignisgenerators:

- Klicken Sie auf die Schaltfläche **Open Event Generator**.

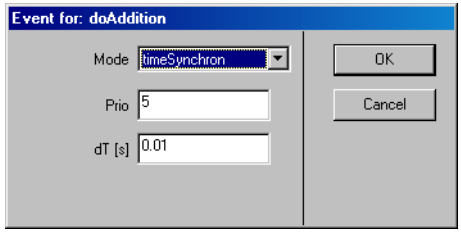
  Das Fenster „Event Generator“ wird geöffnet. Sie müssen für jede zu simulierende Methode ein Ereignis und zusätzlich ein Ereignis `generateData` erzeugen. Die Ereignisse simulieren das Scheduling, das vom Betriebssystem einer realen Anwendung vorgenommen wird.



- Wählen Sie das Ereignis `doAddtion`.
- Wählen Sie **Channels** → **Enable.**
- Wählen Sie erneut das Ereignis `doAddtion`.
- Wählen Sie **Channels** → **Edit**.

  Beide Funktionen sind auch im Kontextmenü in der Fläche „Events“ verfügbar.

  Das Dialogfenster „Event“ öffnet sich.

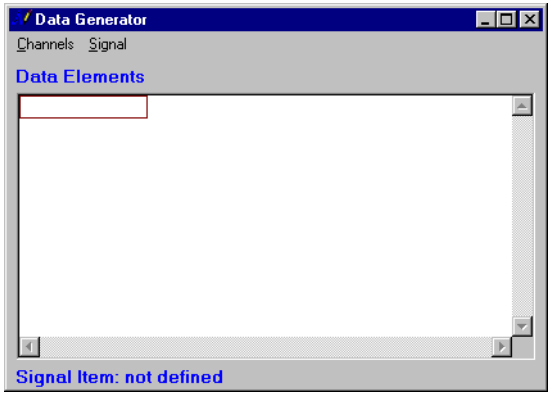


- Stellen Sie den `dT`-Wert auf 0.001.
- Klicken Sie auf **OK**.
- Wählen Sie im Ereignisgenerator das Ereignis `generateData` und stellen Sie seinen `dT`-Wert auf 0.001.
- Schließen Sie das Fenster des Ereignisgenerators.

Einrichten des Datengenerators:

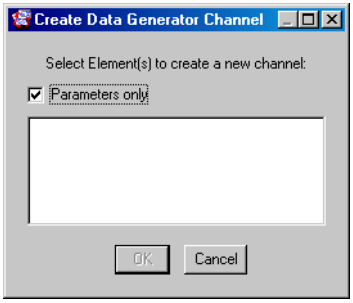
- Klicken Sie auf die Schaltfläche **Open Data Generator**.

  Es wird das Fenster „Data Generator“ geöffnet.



- Wählen Sie **Channels** → **Create**.

  Das Dialogfenster „Create Data Generator Channel“ öffnet sich.

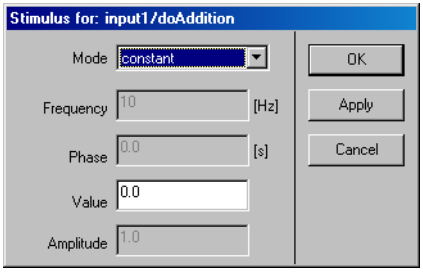


- Deaktivieren Sie bei Bedarf die Option **Parameters only**; dadurch werden sowohl Parameter als auch Variablen in der Liste angezeigt.
- Wählen Sie aus der Liste die Variablen `input1/doAddition` und `input2/doAddition` aus.

- Klicken Sie auf **OK**.

  Jetzt werden beide Eingaben im Feld „Data Elements“ des Datengenerator-Fensters aufgelistet.
- Wählen Sie im Feld „Data Elements“ `input1/doAddition` aus.
- Wählen Sie **Channels** → **Edit**.

  Es erscheint das Dialogfeld „Stimulus“.



- Stellen Sie die Werte wie folgt ein.

Mode:	sinus
Frequency:	1.0 Hz
Phase:	0.0 s
Offset:	0.0
Amplitude:	1.0

- Zum Schließen des Dialogfelds „Stimulus“ klicken Sie auf **OK**.
- Stellen Sie die Werte für `input2` wie folgt ein:

Mode:	sinus
Frequency:	2.0 Hz
Phase:	0.0 s
Offset:	0.0
Amplitude:	2.0

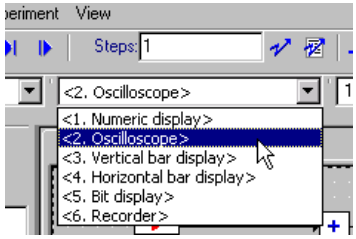
- Schließen Sie das Fenster „Data Generator“.

Mit diesen Einstellwerten erhalten Sie zwei Sinuswellen mit unterschiedlichen Frequenzen und unterschiedlichen Amplituden. Durch die Komponente `Addition` werden die beiden Wellen addiert und die sich daraus ergebende Kurve angezeigt.

Um die drei Kurven auf einem Oszilloskop abgebildet zu sehen, richten Sie nun ein Mess-System ein.

Einrichten des Mess-Systems:

- Wählen Sie im Fenster „Physical Experiment“ aus dem Kombikästchen „Measure View“ ein `<2. Oscilloscope>` als Datenanzeigeart.

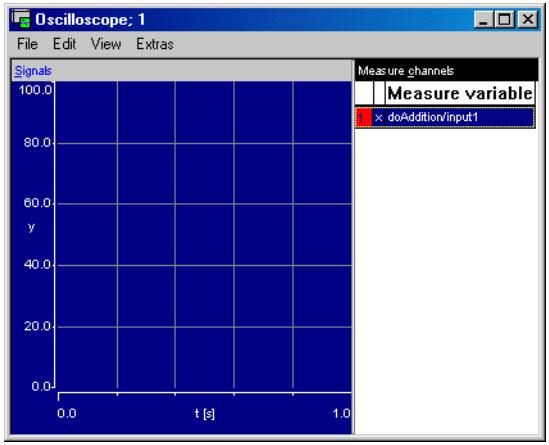


- Klicken Sie im Feld „Elements“ auf das Kästchen mit dem Pluszeichen neben `Addition`, um die Elementliste aufzuklappen.

  Die Elemente der Komponente `Addition` werden angezeigt.

- Wählen Sie `input1/doAddition`.
- Wählen Sie **Elements** → **Measure**.

  Ein Oszilloskopfenster mit `input1` als Messkanal öffnet sich. Das Kombifeld „Measure View“ der Experimentierumgebung wird aktualisiert, um den Titel des Messfensters anzuzeigen.



- Wählen Sie im Fenster „Physical Experiment“ in der Liste „Elements“ `doAddition/input2` aus.
- Wählen Sie **Elements** → **Measure**.

  `input2` wird als Messkanal zum Oszilloskop hinzugefügt.
- Fügen Sie `return/doAddition` zum Messfenster hinzu.
- Wählen Sie in der Experimentierumgebung **File** → **Save Environment**.

Damit ist die Experimentierumgebung eingerichtet, und Sie können mit dem Experiment beginnen. Da Sie das Experiment gespeichert haben, wird es beim nächsten Starten der Experimentierumgebung für diese Komponente automatisch erneut geladen.

6.2.3 Verwenden der Experimentierumgebung

Die Experimentierumgebung stellt eine Reihe von Tools bereit, mit denen Sie sich die Werte aller Variablen in Ihrer Komponente ansehen und auch bei laufendem Experiment die Einrichtung ändern können. Sie können auch die Darstellungsweise der Werte ändern und aus mehreren Darstellungsarten auswählen.

Beginn des Experiments:

- Klicken Sie im Fenster „Physical Experiment“ auf die Schaltfläche **Start Offline Experiment**.

  Das Experiment beginn zu laufen, und die Ergebnisse werden auf dem Oszilloskop abgebildet.

- Um das Experiment zu beenden, klicken Sie auf die Schaltfläche **Stop Offline Experiment**.

Sie werden nur einen kleinen Teil der Kurven auf dem Oszilloskop sehen. Um die Kurven auf dem Oszilloskop abzubilden, müssen Sie den Maßstab auf der Werteachse verändern.

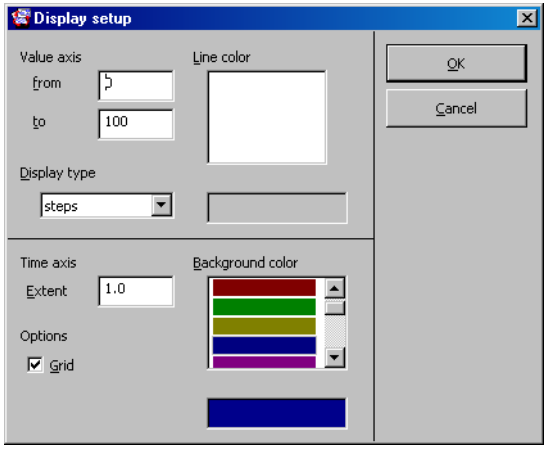
Verändern des Maßstabs auf dem Oszilloskop:

- Wählen Sie aus der Liste „Measure Channels“ im Oszilloskopfenster alle drei Kanäle aus.

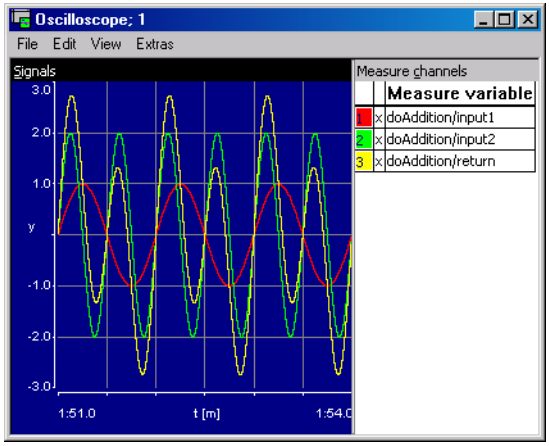
  Zur Mehrfachauswahl halten Sie <CTRL> gedrückt und klicken dabei auf einzelne Kanäle.

  Nun sind alle Datenelemente markiert, und somit betreffen die von Ihnen vorgenommenen Änderungen alle drei Kanäle.

- Wählen Sie **Extras** → **Setup**.
  Es erscheint das Dialogfeld „Display Setup“.



- Stellen Sie die „Value Axis“ auf einen Bereich von -3 bis 3 ein.
- Stellen Sie „Time Axis Extent“ auf 3.
- Wählen Sie im Feld „Background Color“ eine Hintergrundfarbe.
- Drücken Sie <ENTER>.



Das Oszilloskop zeigt nun die Ausgabe mit der entsprechenden Maßeinteilung auf der Werteachse an. Sie sehen die beiden eingegebenen Sinuswellen zusammen mit der sich aus deren Addition ergebenden Welle. Nun können Sie die Eingabewerte verstellen, um zu sehen, wie dadurch die Ausgabe beeinflusst wird.

Ändern der Eingabewerte beim Experimentieren:

- Wählen Sie im Fenster „Physical Experiment" **Experiment** → **Data Generator**, um das Fenster „Data Generator" zu öffnen.
- Wählen Sie im Datengenerator die zu ändernde Variable aus.
- Wählen Sie **Channels** → **Edit**.

  Es erscheint das Dialogfeld „Stimulus".
- Stellen Sie die zu ändernden Werte ein.
- Klicken Sie auf **Apply**.

Die Kurven auf dem Oszilloskop ändern sich entsprechend den neuen Einstellwerten. Sie können bei laufendem Experiment alle Einstellwerte der Experimentierumgebung verändern.

6.2.4 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, die folgenden Aufgaben in ASCET durchzuführen:

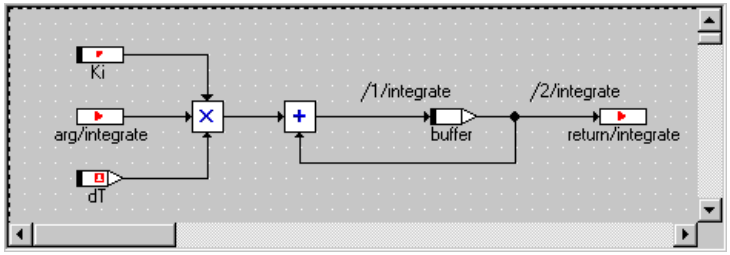
- Aufrufen der Experimentierumgebung
- Einrichten des Ereignisgenerators
- Einrichten des Datengenerators
- Einrichten des Mess-Systems
- Beginnen und Beenden des Experiments
- Speichern des Experiments
- Verändern der Ansteuerung (Stimulus) bei laufendem Experiment.

6.3 Spezifizieren einer wiederverwendbaren Komponente

In dieser Lektion erstellen Sie eine Klasse, die einen Integrator, ein bei Mikrocontrollersoftware häufig zum Einsatz kommendes standardmäßiges Funktionalitätselement, implementiert. Auch wenn es sich dabei um ein etwas komplizierteres Diagramm handelt, sind die Methoden zum Erstellen und Experimentieren die gleichen wie die schon bisher von Ihnen erlernten.

Im vorliegenden Beispiel spezifizieren Sie einen Integrator, der bei bekannter Zeit und Geschwindigkeit die zurückgelegte Strecke errechnet. Der Eingabewert wird in Meter pro Sekunde angegeben und nach jedem Intervall mit einem `dT` in Sekunden multipliziert. Der Wert für jede solche Zeitscheibe wird in einem Akkumulator aufaddiert. Der Akkumulator speichert die Strecke in Metern, die nach einer bestimmten Zeit zurückgelegt wurden.

In ASCET lässt sich ein Standardblock wie z. B. ein Akkumulator mit einem einfachen Diagramm realisieren.



6.3.1 Erstellen des Diagramms

Ehe Sie mit der Arbeit am Diagramm beginnen, müssen Sie die gleichen Schritte wie bei der Komponente `Addition` ausführen. Als erstes erstellen Sie einen neuen Ordner im Ordner `Tutorial`, dann fügen Sie eine neue Klasse hinzu. Abschließend können Sie die Schnittstelle der Methoden, dann das Blockdiagramm und das Layout spezifizieren.

Sie beginnen mit dem Erstellen des Ordners und der neuen Klasse.

Erstellen der Integratorklasse:

- Öffnen Sie im Komponentenmanager den Ordner `Tutorial`.
- Erstellen Sie einen neuen Ordner und benennen Sie diesen mit `Lesson3`.
- Im Ordner `Lesson3` erstellen Sie eine neue Klasse und benennen diese mit `Integrator`.

Festlegen der Schnittstelle des Integrators:

- Wählen Sie im Feld „1 Database“ das Element `Integrator`.
- Doppelklicken Sie auf das Element oder wählen Sie **Component** → **Edit Item**.

  Der Blockdiagrammeditor wird geöffnet.
- Benennen Sie im Feld „Diagrams“ die Methode `calc` in `integrate` um.
- Bearbeiten Sie die Methode `integrate`, indem Sie ein Argument (Typ `cont`) und einen Rückgabewert (Typ `cont`) hinzufügen.
- Klicken Sie auf **OK**.

  Der Schnittstelleneditor wird geschlossen, und Sie befinden sich wieder im Blockdiagrammeditor.
- Setzen Sie das Argument und den Rückgabewert von `integrate` in den Zeichenbereich.

Der Integrator verwendet zwei neue Typen, die wir bislang noch nicht eingesetzt haben: eine Variable und einen Parameter.

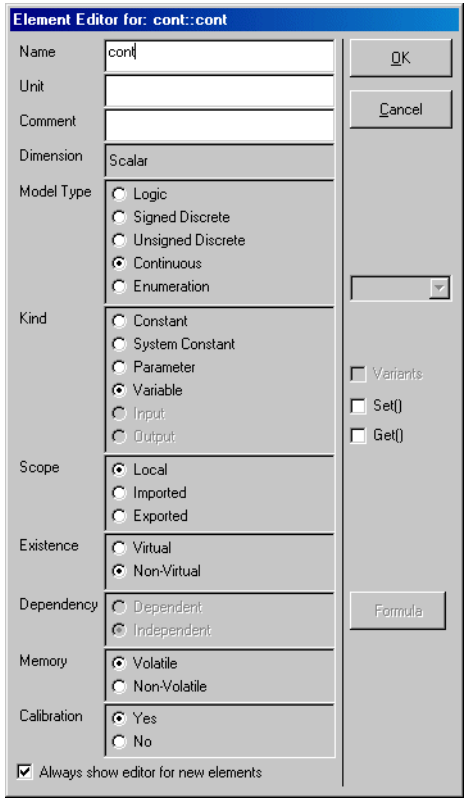
Variablen werden in der gleichen Weise verwendet wie in Programmiersprachen; Sie können darin Werte speichern und die Werte für weitere Berechnungen lesen. Im Gegensatz dazu können *Parameter* nur gelesen werden. Sie lassen sich nur von außen verändern; zwar können sie z. B. in der Experimentierumgebung verstellt werden, lassen sich aber durch eine Berechnung innerhalb der Komponente selbst nicht überschreiben.

Zusätzlich wollen wir in diesem Beispiel auch einen *abhängigen Parameter* spezifizieren, der allerdings für die Funktionalität des Integrators keine Rolle spielt. Ein *abhängiger Parameter* ist von einem oder mehreren Parametern abhängig, d.h. sein Wert berechnet sich aufgrund der Änderung eines anderen. Die Berechnung bzw. Abhängigkeit wird nur bei der Spezifizierung, Kalibrierung bzw. Applikation durchgeführt. Im Targetcode verhält sich ein abhängiger Parameter genauso wie ein normaler Parameter.

Erstellen einer Variablen:

- Klicken Sie auf die Schaltfläche **Continuous**.

  Der Elementeditor öffnet sich.



- Geben Sie im Feld „Name“ den Namen `buffer` ein.
- Klicken Sie auf **OK**.

  Die Variable heißt nun `buffer`. Der Cursor nimmt die Form eines Fadenkreuzes an. Er wird mit der kontinuierlichen Variablen geladen.

- Klicken Sie zum Anordnen der Variablen in den Zeichenbereich.

  Die Variable wird im Zeichenbereich abgelegt. Ihr Name wird in der Liste „Elements“ hervorgehoben.

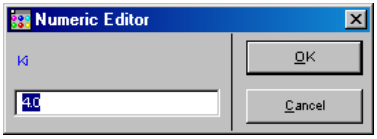
Wenn sich der Elementeditor nicht automatisch öffnet, legen Sie die Variable erst im Zeichenbereich ab. Anschließend doppelklicken Sie in der Liste „Elements“ auf die Variable, um den Elementeditor manuell zu öffnen. Nehmen Sie dann dort die nötigen Einstellungen vor und aktivieren Sie die Option **Always show dialog for new elements**. Wenn Sie dann das nächste Element erzeugen, öffnet sich der Elementeditor automatisch.

Erstellen eines Parameters:

- Klicken Sie auf die Schaltfläche **Continuous Parameter**.

  Der Elementeditor öffnet sich.
- Geben Sie im Feld „Name“ den Namen `Ki` ein.
- Klicken Sie auf **OK**.
- Klicken Sie zum Anordnen des Parameters in den Zeichenbereich.
- Klicken Sie in der Liste „Elements“ mit der rechten Maustaste auf den Parameter und wählen Sie aus dem Kontextmenü **Edit Data** aus.

  Es wird ein Datenkonfigurationsfenster (numerischer Editor) geöffnet.



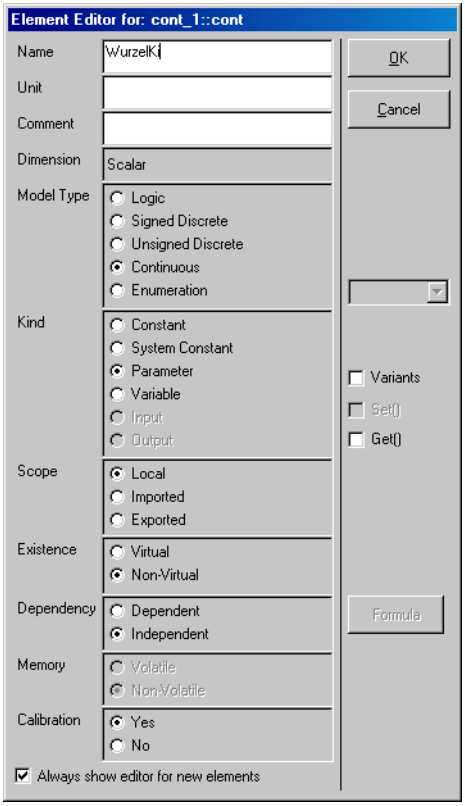
- Stellen Sie den Wert im Fenster auf `4.0`, indem Sie ihn in das Verstellfeld eingeben und <ENTER> drücken.

  Dieser Wert wird damit zum standardmäßigen Wert für den Parameter. Sie können allen Parametern oder Variablen in einem Diagramm standardmäßige Werte zuweisen.

Erstellen eines abhängigen Parameters:

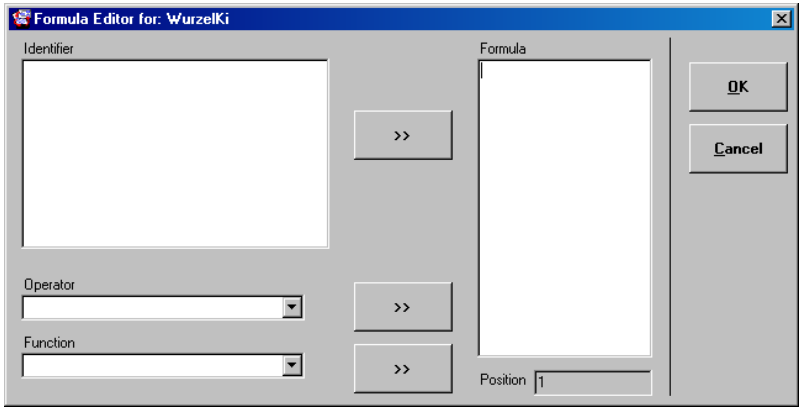
- Klicken Sie auf die Schaltfläche **Continuous Parameter**.

  Der Elementeditor öffnet sich.



- Nennen Sie den Parameter `WurzelKi`.
- Wählen Sie im Feld „Dependency“ die Option **Dependent**

- Öffnen Sie über die Schaltfläche **Formula** den Formeleditor



- Klicken Sie mit der rechten Maustaste in das Feld „Identifier“ und wählen Sie aus dem Kontextmenü **Add**.

  Ein formaler Parameter wird angelegt.
- Geben Sie im Feld „Identifier“ den Namen x für den neuen Parameter ein.
- Spezifizieren Sie im Feld „Formula“ die Berechnungsvorschrift.

  Bei komplexeren Formeln können im Kombifeld „Operator“ verschiedene Operatoren ausgewählt werden. Sie werden nacheinander mit der Schaltfläche **>>** neben dem Kombifeld „Operator“ in das Feld „Formula“ übernommen.

  Ebenso können im Kombifeld „Function“ verschiedene Funktionen ausgewählt und nacheinander mit der Schaltfläche **>>** neben dem Kombifeld „Function“ in das Feld „Formula“ übernommen werden.

- Wählen Sie für das vorliegende Beispiel die Berechnung der Wurzel des formalen Parameters:

  Formal Parameter: `x`
  Formula: `sqrt(x)`

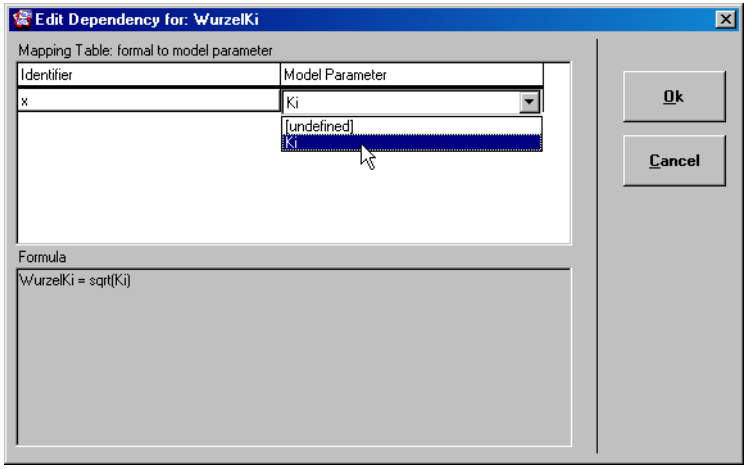
- Beenden Sie die Eingabe mit **OK** und schließen Sie auch den Elementeditor.

  Der Cursor nimmt die Form eines Fadenkreuzes an.

- Klicken Sie zum Anordnen des Parameters in den Zeichenbereich.

- Klicken Sie im Blockdiagrammeditor in der Liste „Elements“ mit der rechten Maustaste auf den Parameter `WurzelKi` und wählen Sie aus dem Kontextmenü **Edit Data**.

- Weisen Sie im Fenster „Dependency Editor“ dem formalen Parameter `x` einen Modellparameter aus dem Kombifeld zu (im Beispiel `Ki`).



- Schließen Sie die Eingabe mit **OK** ab.

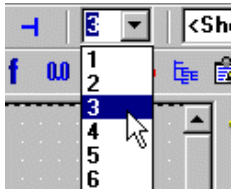
  Sie haben jetzt abhängig vom Parameter `Ki` einen Parameter spezifiziert, der bei der Kalibrierung automatisch auf Basis des Wertes des

Parameter `Ki` berechnet wird. Sie können später im Experiment die Abhängigkeit bzw. Berechnung überprüfen.

Nachdem Sie alle Elemente hinzugefügt haben, müssen Sie einen Integrator vorgeben. Beginnen Sie mit der Erstellung des Diagramms.

Erstellen des Diagramms:

- Setzen Sie im Kombifeld „Argument Size“ den aktuellen Wert auf 3, wodurch die Anzahl von Eingabewerten für das Multiplikationszeichen vorgegeben wird.



- Erstellen Sie ein Multiplikationszeichen und ordnen Sie dieses im Zeichenbereich an.

- Klicken Sie auf die Schaltfläche **dT**, um ein dT-Element zu erstellen.

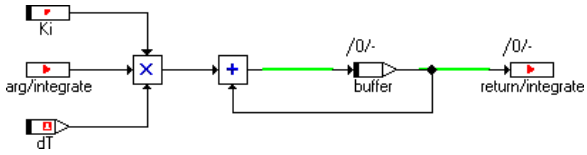
  Der Elementeditor öffnet sich. Sie haben keine Einstellmöglichkeiten.

- Klicken Sie auf **OK**.
- Ordnen Sie das dT-Element im Zeichenbereich an.
- Erstellen Sie ein Additionszeichen mit *zwei* Eingängen und ordnen Sie dieses im Zeichenbereich an.

  Dabei ist zu beachten, dass vor dem Erstellen des Zeichens die Argumentgröße auf 2 zurückgestellt werden muss.

- Verbinden Sie die Elemente gemäß der untenstehenden Darstellung miteinander.

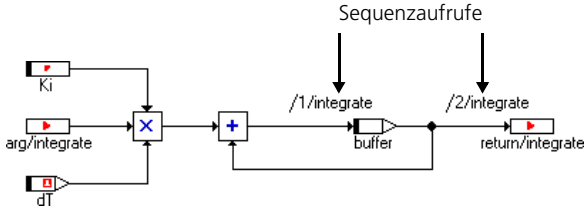
  Die Eingabelinien für den Puffer und den Rückgabewert erscheinen grün.



Nun sind alle Elemente des Diagramms an Ort und Stelle. Als nächstes müssen Sie die Rechenfolge durch Einstellen des Sequenzaufrufs festlegen.

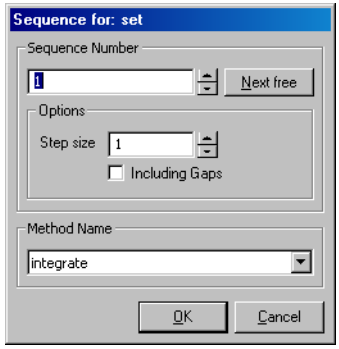
Zuweisen eines Wertes zu einem Sequenzaufruf:

- Klicken Sie mit der rechten Maustaste auf den Sequenzaufruf über der Variablen `buffer`.



- Wählen Sie **Edit** aus dem Kontextmenü.

  Der Sequenzeditor öffnet sich.



- Zum Übernehmen der standardmäßigen Einstellwerte klicken Sie auf **OK**.

  Die Zuweisung erfolgt in dem Algorithmus für Ihren Integrator zuerst.

Einstellen der Sequenznummer in einem Sequenzaufruf:

- Klicken Sie mit der rechten Maustaste auf den Sequenzaufruf über dem Rückgabewert für `integrate`.
- Wählen Sie **Edit** aus dem Kontextmenü.
- Setzen Sie im Sequenzeditor den Wert im Feld „Sequence Number“ auf 2.
- Klicken Sie auf **OK**.

  Der Rückgabewert wird erst nach Aktualisierung der Variablen `buffer` aktualisiert.

Einstellen des Layouts:

- Wählen Sie **Component** → **Edit Layout**.

  Der Layout-Editor wird geöffnet.
- Alternativ können Sie wiederum aus der Darstellung „Information/Browse“ durch Doppelklicken auf das Layout innerhalb des Registers „Layout“ in den Layout-Editor gelangen.
- Ziehen Sie das Argument von `integrate` auf die Mitte der linken Seite des Blockes.
- Ziehen Sie den Rückgabewert auf die Mitte der rechten Seite des Blockes.
- Klicken Sie auf **OK**.

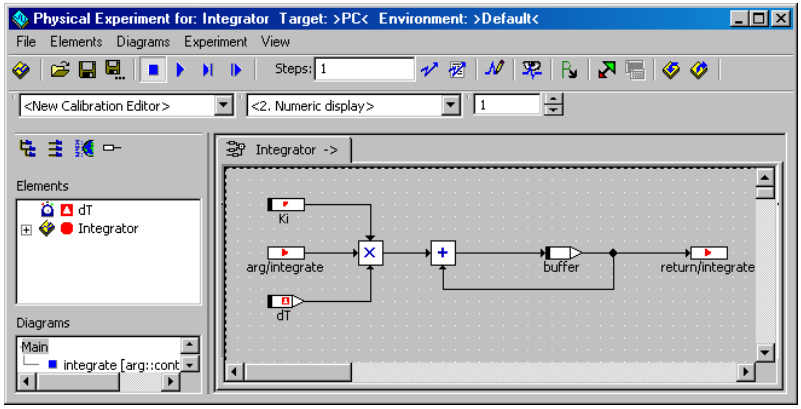
Damit ist das Diagramm für die Integratorklasse fertiggestellt. Sie sollten nun die Änderungen am Diagramm durch Anwahl von **Diagram** → **Store To Cache** speichern. Änderungen, die keine Auswirkungen auf das Diagramm selbst haben, werden automatisch gespeichert. Als nächstes speichern Sie die Änderungen an der Datenbank durch Anwahl von **File** → **Save Database** im Komponentenmanager.

6.3.2 Experimentieren mit dem Integrator

Sie müssen wiederum zuerst den Ereignisgenerator, danach den Datengenerator und zum Schluss das Mess-System einrichten.

Einrichten der Experimentierumgebung für den Integrator:

- Starten Sie die Experimentierumgebung mit **Component → Open Experiment.**



- Klicken Sie auf die Schaltfläche **Event Generator**.
- Aktivieren Sie die Ereignisse `integrate` und `generateData` unter Verwendung des standardmäßigen `dT`-Werts von 0.01.
- Schließen Sie das Fenster „Event Generator“.

- Klicken Sie auf die Schaltfläche **Data Generator**.
- Erstellen Sie einen Datenkanal für die Methode `integrate` durch Anwahl von **Channels → Create** und Anwahl des Arguments von `integrate`.
- Stellen Sie die Werte wie folgt ein:

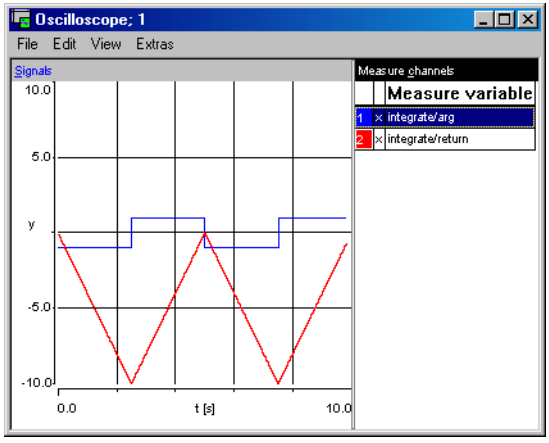
Mode:	pulse
Frequency:	0.2 Hz
Phase:	0.0 s

Offset: -1.0

Amplitude: 2.0

- Schließen Sie den Datengenerator.
- Öffnen sie ein Oszilloskopfenster mit den Werten von `arg` und `return` der Methode `integrate`.
- Stellen Sie die Werteachse auf einen Bereich von -10 bis 10 und den Zeitachsenbereich bis 10 Sekunden ein.
- Zum Starten des Experiments wählen Sie **Start Offline Experiment** an.

Der Ausgabewert der Methode `integrate` steigt bei positivem Argument und nimmt bei negativem Argument ab. Da der positive und negative Anteil der Eingabekurve gleich sind, bleibt die Ausgabe innerhalb stabiler Grenzen.



Rücksetzen eines Experiments:

Wenn Sie ein Experiment unterbrechen und dann wieder neu starten, werden die Werte aller Variablen gespeichert. Manchmal möchte man u. U. jedoch alle Variablen auf ihre ursprünglichen Werte zurücksetzen.

- Wählen Sie **Elements** → **Reinitialize** → **Variables** oder **Parameters** oder **Both**.

  Je nach Ihrer Wahl werden alle Variablen oder alle Parameter oder beide auf ihre ursprünglichen Werte zurückgesetzt.

Als nächstes sollten Sie mit verschiedenen Einstellungen experimentieren, um die Funktion des Integrators zu veranschaulichen. Sie können den Parameter `Ki` einstellen und die Eingabe verändern.

Experimentieren mit dem Integrator:

- Zum Erweitern des Elements `Integrator` klicken Sie im Feld „Elements“ auf das Pluszeichen neben dem Element `Integrator`.
- Wählen Sie den Parameter `Ki` an.
- Wählen Sie **Elements** → **Calibrate**.

  Es wird ein numerischer Editor für den Parameter geöffnet.
- Setzen Sie den Wert auf 5, indem Sie diesen eingeben und dann <ENTER> drücken.

  Die Ausgabekurve auf dem Oszilloskop wird steiler.
- Setzen Sie den Wert auf 3.

  Nun wird die Ausgabekurve wieder flacher.
- Setzen Sie den Parameter zurück auf 4 und schließen Sie den numerischen Editor.
- Öffnen Sie das Fensters „Data Generator“.
- Stellen Sie den Offset des Eingabeimpulses auf -0.5.
- Klicken Sie auf **OK**.

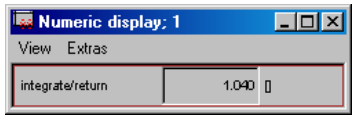
Jetzt ist der positive Anteil größer, und folglich wird der Ausgang zu steigen beginnen. An einer bestimmten Stelle wird er so gestiegen sein, dass er die Grenzen des Oszilloskops überschreitet. Sie können den Maßstab des Oszilloskops für jeden Wert einzeln einstellen, indem Sie nur den betreffenden Wert auswählen, wenn Sie Veränderungen vornehmen. Um sich den Ausgabewert anzusehen, können Sie auch ein numerisches Anzeigefenster öffnen.

Numerische Anzeige eines Wertes:

- Wählen Sie im Kombifeld „Measure View“ der Experimentierumgebung `<1.Numeric Display>`.
- Wählen Sie im Feld „Elements“ den Rückgabewert (`return`) der Methode `integrate`.

- Wählen Sie **Elements** → **Measure**.

  Ein numerisches Anzeigefenster zeigt den aktuellen Ergebniswert an.



- Zeigen Sie sich auch den abhängigen Parameter `WurzelKi` an.

- Experimentieren Sie damit, indem Sie `Ki` verändern und über die Schaltfläche **Update Dependent Parameters** die Aktualisierung anstoßen.

6.3.3 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, die folgenden Aufgaben in ASCET auszuführen:

- Erstellen eines Parameters
- Erstellung und Spezifikation eines abhängigen Parameters
- Erstellen einer Variablen
- Erstellen eines Operators mit mehreren Eingängen
- Einstellen der Sequenznummer eines Sequenzaufrufs
- Zuweisen eines standardmäßigen Wertes
- Kalibrieren eines Wertes während des Experimentierens
- Anzeigen von Werten in einem numerischen Anzeigefenster.

6.4 Ein praktisches Beispiel

In dieser Lektion werden Sie einen Regler erstellen, der auf einem etwas erweiterten standardmäßigen PI-Filter basiert. Der Regler wird zum Konstanthalten der Drehzahl eines im Leerlauf arbeitenden Fahrzeugmotors eingesetzt.

Beim Regeln der Leerlaufdrehzahl eines Motors müssen Sie darauf achten, dass die Istdrehzahl `n` möglichst dicht bei der Leerlaufnenndrehzahl `n_nominal` bleibt. Zum Ermitteln der zu regelnden Abweichung wird der Wert `n` von `n_nominal` subtrahiert.

Die Abweichung der Istdrehzahl bildet die Grundlage für die Berechnung des Luftnennwerts `air_nominal`, der seinerseits die Drosselklappenstellung bestimmt, d.h. die Menge der dem Motor zugeführten Luft.

6.4.1 Spezifikation des Reglers

Die Schritte bei der Erstellung des Diagramms für Ihren Regler sind die gleichen wie schon früher:

- Hinzufügen eines neuen Ordners und Erstellen der Komponente im Komponentenmanager,
- Definieren der Schnittstelle und Zeichnen des Blockdiagramms.

Der Hauptunterschied besteht darin, dass Sie den Regler als Modul implementieren werden. Module werden bei Projekten als die Komponenten der höchsten Ebene eingesetzt. Sie definieren die Prozesse, die ein Projekt darstellen.

Erstellen der Regler-Komponente:

- Im Komponentenmanager fügen Sie zum Ordner `Tutorial` einen neuen Unterordner hinzu und benennen diesen mit `Lesson4`.
- Rufen Sie den Ordner `Lesson4` auf und wählen Sie **Insert** → **Module** → **Block Diagram** zum Hinzufügen eines neuen Moduls.
- Benennen Sie das neue Modul mit `IdleCon` und öffnen Sie den Blockdiagrammeditor.
- Im Feld „Diagrams“ nehmen Sie eine Umbenennung des Diagramms `process` in `p_idle` vor.

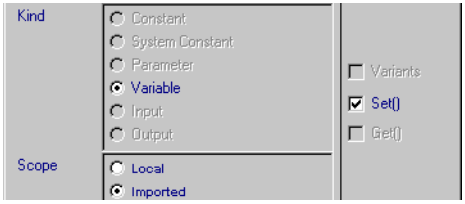
Die Funktionalität von Modulen wird in Prozessen vorgegeben, die den Methoden in Klassen entsprechen. Im Gegensatz zu Methoden verfügen Prozesse nicht über Argumente oder Ergebniswerte. Der Datenaustausch (Kommunikation) zwischen Prozessen basiert auf gerichteten Messages, die in ASCET als *Receive-Messages* (Eingänge) und *Send-Messages* (Ausgänge) bezeichnet werden.

In Ihrem Regler werden Sie eine Receive-Message zum Verarbeiten der Istdrehzahl `n` und eine Send-Message zum Verstellen der Drosselklappe in die Stellung `air_nominal` verwenden.

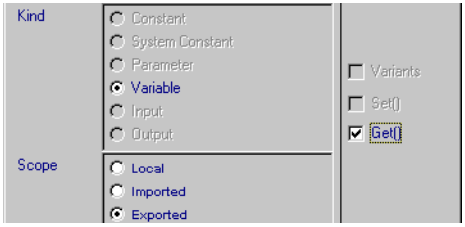
Spezifizieren der Schnittstelle des Reglers:

- Erstellen Sie eine Receive-Message, indem Sie auf die Schaltfläche **Receive Message** klicken und diese mit `n` benennen.

- Aktivieren Sie im Elementeditor für die Message `n` die Option **Set()**.



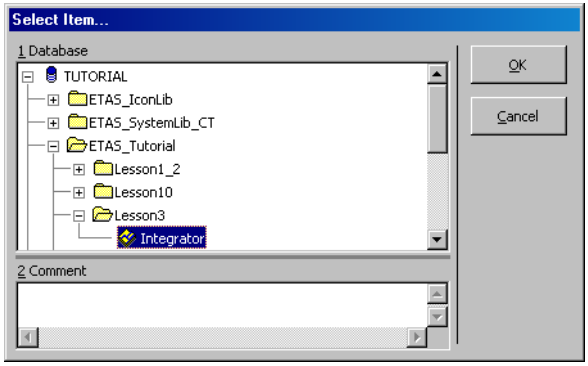
- Klicken Sie auf die Schaltfläche **Send Message** und dann in den Zeichenbereich, um eine Send-Message zu erstellen.
- Nehmen Sie eine Umbenennung in `air_nominal` vor.
- Aktivieren Sie im Elementeditor für die Message `air_nominal` die Option **Get()**.



Das Reglerelement benutzt den von Ihnen in Lektion 3 erstellten Integrator.

Hinzufügen des Integrators zum Regler:

- Wählen Sie **Element** → **Add Item** zum Öffnen des Dialogfelds „Select Item“.



- Wählen Sie im Feld „1 Database“ die Komponente `Integrator` aus dem Ordner `Tutorial\Lesson3` und klicken Sie zum Hinzufügen des Integrators auf **OK**.

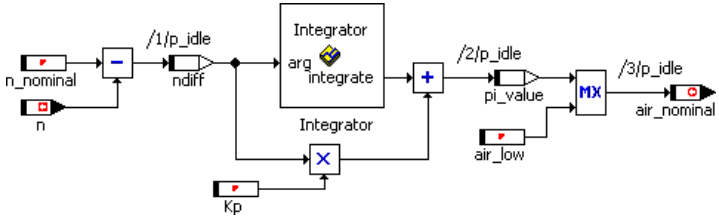
  Der Integrator wird in die Komponente `IdleCon` aufgenommen. Eine Komponente wird durch Referenzieren aufgenommen, d.h. wenn Sie die ursprüngliche Spezifikation des Integrators ändern, kommt diese Änderung auch in der aufgenommenen Komponente zum Ausdruck.

Neben den bisher hinzugefügten Elementen müssen Sie noch folgendes zu Ihrem Regler hinzufügen:

- zwei kontinuierliche Variablen mit der Benennung `ndiff` und `pi_value`
- drei kontinuierliche Parameter mit der Benennung `n_nominal`, `Kp` und `air_low`

Spezifizieren des Rests des Reglers:

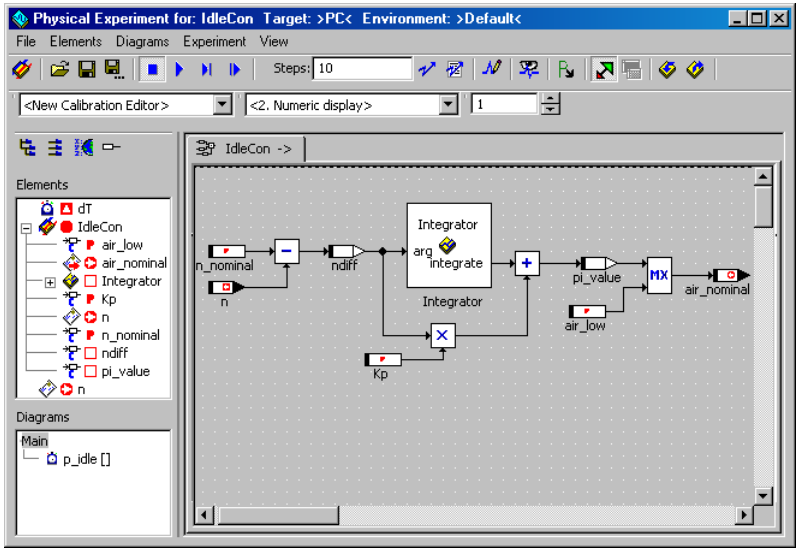
- Erstellen Sie die Operatoren und die anderen notwendigen Elemente und verbinden Sie diese dann wie im untenstehenden Blockdiagramm.



- Wählen Sie in der Liste „Elements“ den Parameter `n_nominal` und danach **Element** → **Edit Data**.
- Stellen Sie den Wert für `n_nominal` auf 900 ein.
- Stellen Sie den Wert für `Kp` auf 0.5 ein.
- Speichern Sie Ihre Spezifikation im Diagramm ab und übernehmen Sie die Änderungen in die Datenbank.

6.4.2 Experimentieren mit dem Regler

Das Experimentieren mit Modulen geht wie das Experimentieren mit anderen Komponenten vor sich. Zuerst werden der Daten- und der Ereignisgenerator eingerichtet und dann das Mess-System.



Einrichten der Experimentierumgebung:

- Wählen Sie **Component** → **Open Experiment** zum Starten der Experimentierumgebung.
- Öffnen Sie das Fenster „Event Generator“ und aktivieren Sie das Ereignis für den Prozess `p_idle` unter Verwendung des standardmäßigen Werts von 0.01 für `dT`.

  Ein Ereignis für einen Prozess funktioniert ebenso wie ein Ereignis für eine Methode.

- Öffnen sie das Fenster „Data Generator“ und richten Sie den Kanal für die Receive-Message `n` mit den folgenden Werten ein:

Mode:	pulse
Frequency:	1.0 Hz
Phase:	0.0
Offset:	800.0
Amplitude:	200.0

- Richten Sie ein Oszilloskop mit den Variablen `n_diff` und `air_nominal` ein.
- Setzen Sie im Oszilloskop die Werteachse auf -500 bis 500 und den Zeitachsenbereich auf 2.

- Klicken Sie auf die Schaltfläche **Save Environment**.

Das Experiment ist jetzt für die Darstellung der Beziehung zwischen der Drehzahlabweichung und der Drosselklappenstellung eingerichtet.

Experimentieren mit dem Regler:

- Starten Sie das Experiment durch Klicken auf die Schaltfläche **Start Offline Experiment**.
- Öffnen Sie einen numerischen Editor für die Variablen `Ki` und `Kp`. Damit können Sie die Werte `Ki` und `Kp` verstellen und die jeweilige Auswirkung auf die Ausgabe beobachten.

  Ab und zu muss u. U. das Modell neu gestartet werden, um wieder zu vernünftigen Werten zu kommen.

6.4.3 Ein Projekt

Ein Projekt ist die Haupteinheit der ASCET-Software, die ein komplettes Softwaresystem darstellt. Dieses Softwaresystem lässt sich auf experimentellen oder auf Mikrocontroller-Targets in Echtzeit mit einem Online-Experiment ausführen. Individuelle Komponenten können nur in der Offline-Experimentierumgebung erprobt werden.

Jedes Experiment läuft im Rahmen eines Projekts. Wird Code für ein Projekt generiert, dann wird auch der Betriebssystem-Code erzeugt. Die Spezifikation des Betriebssystems muss auf einem ASCET-Softwaresystem im Echtzeitbetrieb

laufen. Läuft ein Softwaresystem im Echtzeitbetrieb, so spricht man vom *Online-Experimentieren*. Bislang haben wir nur *offline* experimentiert, d.h. nicht im Echtzeitbetrieb.

Hinweis

Alle ASCET-Experimente – sowohl Online als auch Offline – laufen im Rahmen eines Projekts ab. Dies ist bei Offline-Experimenten deutlich erkennbar, die ein (ansonsten unsichtbares) Standardprojekt nutzen. Das Erstellen und Einrichten eines Projekts für den ausdrücklichen Zweck der Spezifikation eines Betriebssystems ist nur für Online-Experimente erforderlich. Allerdings haben Sie auch die Möglichkeit, das Standardprojekt für Ihren speziellen Anwendungsfall zu konfigurieren.

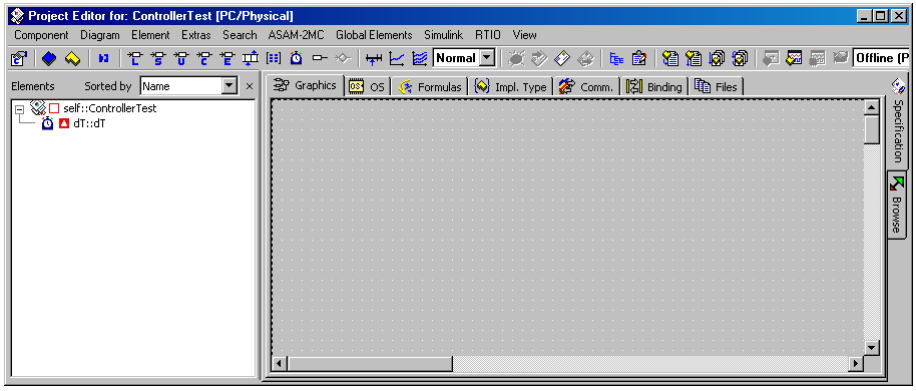
6.4.4 Einrichten des Projekts

Das Projekt wird im Komponentenmanager erstellt. Sie können es zum gleichen Ordner hinzufügen wie das Modul `IdleCon`.

Erstellen eines Projekts:

- Wählen Sie im Komponentenmanager **Insert** → **Project** oder klicken Sie auf **Insert Project**, um ein neues Projekt hinzuzufügen.
- Nennen Sie das Projekt `ControllerTest`.
- Wählen Sie **Component** → **Edit Item** oder doppelklicken Sie auf das Projekt.

  Der Projekteditor wird für das Projekt geöffnet.



Als nächster Schritt ist der Regler `IdleCon` zur Liste „Elements“ des Projekts hinzuzufügen.

Aufnehmen von Komponenten in ein Projekt:

- Wählen Sie im Projekteditor **Element** → **Add Item**, um das Dialogfeld „Select Item“ zu öffnen.
- Wählen Sie aus der Liste „1 Database“ die Komponente `IdleCon` aus dem Ordner `Tutorial\Lesson4`.
- Klicken Sie auf **OK** zum Hinzufügen der Komponente.

  Der Name der Komponente wird in der Liste „Elements“ des Projekteditors angezeigt.

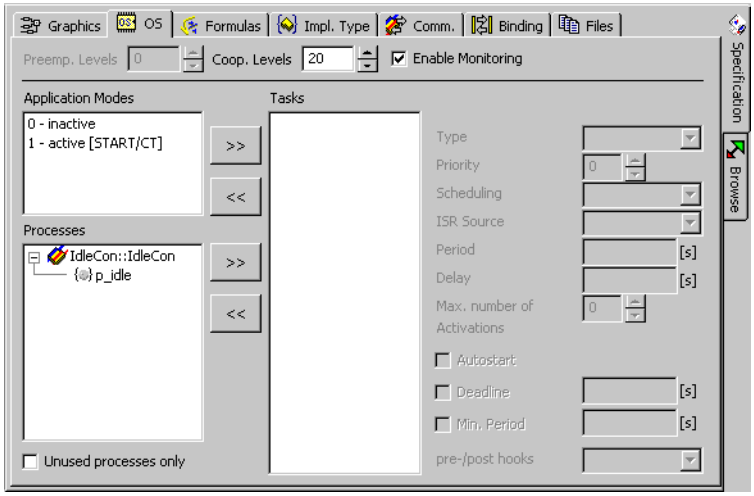
Komponenten werden durch Referenzieren aufgenommen, d.h. wenn Sie das Diagramm einer enthaltenen Komponente ändern, wird die betreffende Änderung auch im Projekt wirksam.

Das Betriebssystem plant die Tasks und Prozesse eines Projekts. Ehe Sie einen Code für das Projekt generieren können, müssen sie die notwendigen Tasks erstellen und diesen die Prozesse zuweisen.

Der Betriebssystem-Schedule wird im Register „OS“ des Projekteditors spezifiziert. Sie legen als nächstes für den Betriebssystem-Schedule fest, dass der Prozess `p_idle` alle 10 ms aktiviert werden soll.

Einrichten des Betriebssystem-Schedules für das Projekt:

- Klicken Sie auf das Register „OS“.



- Wählen Sie **Task** → **Add** zum Erstellen einer neuen Task.
- Benennen Sie diese mit `Task10ms`.

  Die erstellten Tasks sind standardmäßig Alarm-Tasks, d.h. sie werden in regelmäßigen Abständen durch das Betriebssystem aktiviert.
- Weisen Sie der Task im Feld „Period“ ein Intervall von 0.01 Sekunden zu.

  Das Intervall legt fest, wie häufig die Task aktiviert wird, im vorliegenden Fall alle 10 ms.

- Erweitern Sie in der Liste „Processes“ den Eintrag `IdleControl` durch Klick auf das daneben befindliche Pluszeichen.
- Wählen Sie den Prozess `p_idle` und wählen Sie **Process** → **Assign** an.

  Der Prozess wird der Task `Task10ms` zugewiesen. Er wird unterhalb des Tasknamens in der Liste „Tasks“ dargestellt.

In Projekten werden importierte und exportierte Elemente für die Kommunikation zwischen Prozessen verwendet. Dabei handelt es sich um globale Elemente, die den Send- und Receive-Messages in den Modulen entsprechen. Globale Elemente müssen im Projekt vereinbart und mit ihren entsprechenden Pendants in den im Projekt enthaltenen Modulen verknüpft werden.

Festlegen globaler Elemente:

- Wählen Sie im Projekteditor **Global Elements** → **Resolve Globals**.

  Die notwendigen globalen Elemente werden erstellt und automatisch mit ihren Pendants verknüpft. Elemente mit dem gleichen Namen werden automatisch miteinander verknüpft.

  Dabei ist zu beachten, dass Send-Messages standardmäßig im Modul definiert (exportiert) sind.

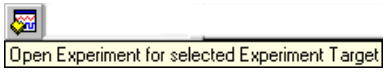
6.4.5 Experimentieren mit dem Projekt

Sie werden nun mit diesem Projekt ein Offline-Experiment durchführen. Offline-Experimente lassen sich auf dem PC ohne den Anschluss zusätzlicher Hardware durchführen. Projekte laufen standardmäßig auf dem PC. Somit brauchen Sie keine Einstellwerte zu verändern. Das Offline-Experimentieren mit Projekten funktioniert so wie das Offline-Experimentieren mit Komponenten.

Einrichten der Experimentierumgebung:

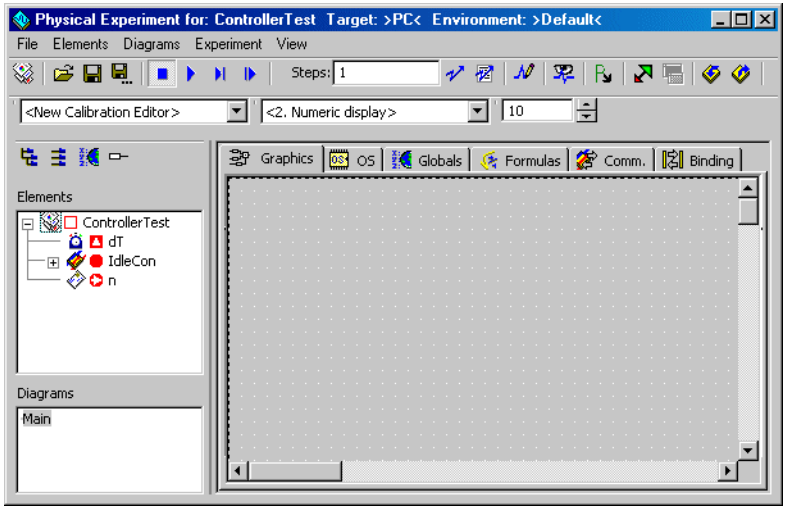
- Wählen Sie im Komponentenmanager **File** → **Save Database**.

  Es empfiehlt sich immer, Änderungen in die Datenbank zu übernehmen, ehe Sie die Experimentierumgebung starten.



- Klicken Sie auf die Schaltfläche **Open Experiment for selected Experiment Target**.

  Code für das Projekt wird generiert und die Offline-Experimentierumgebung geöffnet.



- Klicken Sie auf die Schaltfläche **Open Event Generator**.

  Im Ereignisgenerator sehen Sie ein Ereignis für jede Task, die Sie im Experiment verwenden möchten, und nicht - wie beim Experimentieren mit Komponenten - für jede Methode oder jeden Prozess.

- Aktivieren Sie die Task `generateData` im Ereignisgenerator und benutzen Sie den standardmäßigen `dT`-Wert von 0.01 s.

  Die Task `Task10ms` ist schon standardmäßig aktiviert, und beide Ereignisse haben jetzt 0.01 Sekunden als `dT`-Wert; folglich brauchen Sie keine weiteren Einstellungen vorzunehmen.

- Schließen Sie den Ereignisgenerator.

- Richten Sie den Datengenerator und das Mess-System mit den gleichen Werten wie im vorangegangenen Experiment ein (vgl. „Experimentieren mit dem Regler“ auf Seite 152).
- Speichern Sie die Umgebung durch Anwahl von **File** → **Save Environment** ab.

Durchführen des Experiments:

- Klicken Sie auf die Schaltfläche **Start Offline Experiment**.
- Verändern Sie die Parameter `Ki` und `Kp` wie im vorhergehenden Abschnitt, um die Auswirkungen der von Ihnen vorgenommenen Änderungen auf die Ausgabe beobachten zu können.

6.4.6 Zusammenfassung

Nach Abschluss dieser Lektion müssten sie in der Lage sein, die folgenden Aufgaben in ASCET auszuführen:

- Erstellen von Modulen
- Erstellen von Mitteilungen in Modulen
- Verwenden von Komponenten aus dem Komponentenmanager im Blockdiagramm
- Erstellen eines Projekts
- Aufnahme von Komponenten in Projekte
- Erstellen von Tasks und Zuweisen von Prozessen zu diesen Tasks
- Experimentieren mit Projekten

6.5 Erweitern des Projekts

In dieser Lektion werden Sie einige Verbesserungen vornehmen, um Ihren Regler realistischer zu machen. Sie werden einen Signalwandler erstellen, der Sensorwerte in Istwerte umwandelt. Viele Messfühler, wie sie z. B. in Fahrzeuganwendungen zum Einsatz kommen, liefern eine elektrische Spannung, die einem bestimmten Messwert entspricht, z. B. einer Temperatur, einer Stellung oder einer Drehzahl. Das Verhältnis zwischen Spannung und Messwert ist nicht immer linear. ASCET stellt Kennfelder zum wirksamen Modellieren eines solchen Verhaltens zur Verfügung.

6.5.1 Spezifizieren des Signalwandlers

Der erste Schritt beim Modellieren des Signalwandlers besteht im Erstellen eines Ordners sowie eines Moduls, das die Funktionalität vorgibt. Der Signalwandler verwendet zwei Kennlinien zum Abbilden seiner Eingabewerte im Vergleich zu den entsprechenden Ausgängen.

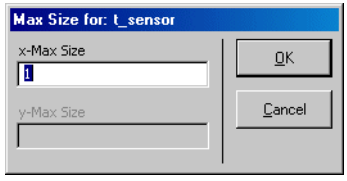
Erstellen des Moduls:

- Erstellen Sie im Komponentenmanager einen neuen Ordner `Tutorial\Lesson5`.
- Erstellen Sie ein neues Modul und benennen Sie dieses mit `SignalConv`.
- Wählen Sie **Component** → **Edit Item** oder doppelklicken Sie auf das Element, um den Blockdiagrammeditor zu öffnen.
- Wählen Sie im Blockdiagrammeditor **Diagram** → **Add Process**, um einen zweiten Prozess zu erzeugen.
- Benennen Sie die beiden Prozesse mit `n_sampling` und `t_sampling`.
- Erstellen Sie in der Liste „Elements“ zwei Receive-Messages `U_n` und `U_t` sowie zwei Send-Messages `t` und `n`.

- Erstellen Sie eine Kennlinie durch Klicken auf die Schaltfläche **One-D Table Parameter**.

  Der Elementeditor öffnet sich.
- Benennen Sie die Tabelle mit `t_sensor` und schließen Sie den Elementeditor.

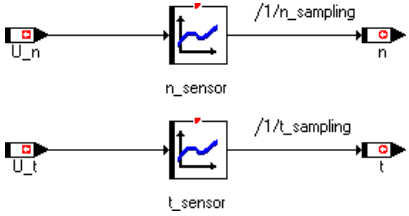
  Das Dialogfenster „Max Size“ wird geöffnet. Da Sie eine eindimensionale Kennlinie angelegt haben, ist das Feld „y-Max Size“ inaktiv.



- Geben Sie im Feld „x-Max Size“ den Wert 13 ein.

  Die Kennlinie kann nun maximal 13 Spalten umfassen.
  Da Sie eine eindimensionale Kennlinie erzeugt haben, ist das Feld „y-Max Size“ deaktiviert.
- Klicken Sie auf **OK**, um das Dialogfeld zu schließen.
- Klicken Sie dann in den Zeichenbereich, um die Tabelle dort anzuordnen.

  Die Tabelle wird zur Liste „Elements“ hinzugefügt.
- Erstellen Sie eine zweite Tabelle mit maximal 2 Spalten und benennen Sie sie mit `n_sensor`.
- Verbinden Sie die Elemente wie dargestellt und bearbeiten Sie die Sequenzierung, um die entsprechenden Prozesse zuzuweisen.



Im nächsten Schritt werden die Daten für die beiden Kennlinien bearbeitet. ASCET stellt einen Tabelleneditor zum Bearbeiten von Arrays, Matrizen und Kennfeldern bereit.

Bearbeiten der Tabellen:

- Klicken sie mit der rechten Maustaste auf die Tabelle `t_sensor` und wählen Sie aus dem Kontextmenü **Edit Data**.

  Der Tabelleneditor wird geöffnet.

- Stellen Sie die Größe der Tabelle wie folgt ein:

  Die Tabelle wird auf 13 Spalten erweitert, wobei alle z-Werte standardmäßig auf 0 gesetzt werden.

- Geben Sie die in der nachfolgenden Tabelle angegebenen Werte ein. Die obere Reihe entspricht der Reihe X, die untere der Reihe Z.

0.00	0.08	0.30	0.67	1.17	2.5	5.00	7.50	8.83	9.33	9.70	9.92	10.00
-40.0	-26.0	-13.0	0.0	13.0	40.0	80.0	120.0	146.0	160.0	173.0	186.0	200.0

  Sie sollten die Tabelle so bearbeiten, dass Sie zuerst die Messpunkte (X-Werte) von links nach rechts eingeben.

- Klicken sie auf einen X-Wert und geben Sie dann im Dialogfeld den neuen Wert ein.

  Der neue X-Wert muss innerhalb der von den Messpunkten links und rechts angegebenen Grenzen liegen.

- Geben Sie dann die Ausgabewerte ein, indem Sie auf einen Wert klicken und den markierten Wert überschreiben.

- Bearbeiten sie die zweite Tabelle in gleicher Weise unter Verwendung der folgen Werte:

  0.0 10.0

  0.0 6000.0

- Wählen Sie im Blockdiagrammeditor **Diagram** → **Store to Cache**.

- Klicken Sie im Komponentenmanager auf die Schaltfläche **Save** zum Abspeichern der von Ihnen vorgenommenen Änderungen.

Im vorliegenden Beispiel stellt die zweite Tabelle eine lineare Beziehung zwischen Eingabe und Ausgabe dar, folglich benötigt sie nur zwei Messpunkte. Dies funktioniert, da Sie als Interpolationsart zwischen Werten *linear* vorgegeben haben.

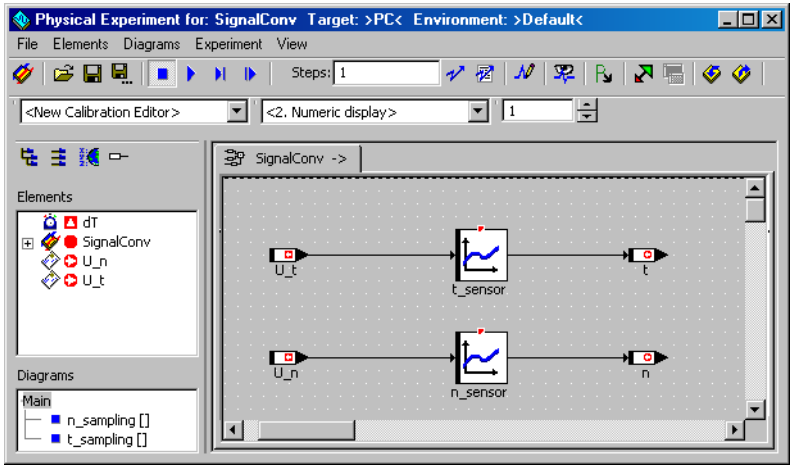
Bei der linearen oder Geradeninterpolation wird für einen Eingabewert zwischen zwei Messpunkten der Ausgabewert anhand einer Geraden ermittelt. Im vorliegenden Fall liefert eine Eingabe von 0 den Rückgabewert 0 und eine Eingabe von 10 den Wert 6000. Beträgt der Eingabewert 5, wird der Rückgabewert demgemäß zu 3000 interpoliert.

6.5.2 Experimentieren mit dem Signalwandler

Sie können nun mit der neuen Komponente experimentieren und das Verhalten der Tabellen beobachten. Da die beiden Tabellen über unterschiedliche Wertebereiche verfügen, werden Sie für jede Tabelle ein gesondertes Oszilloskopfenster einrichten.

Einrichten der Experimentierumgebung:

- Wählen Sie **Component** → **Open Experiment** zum Öffnen der Experimentierumgebung.



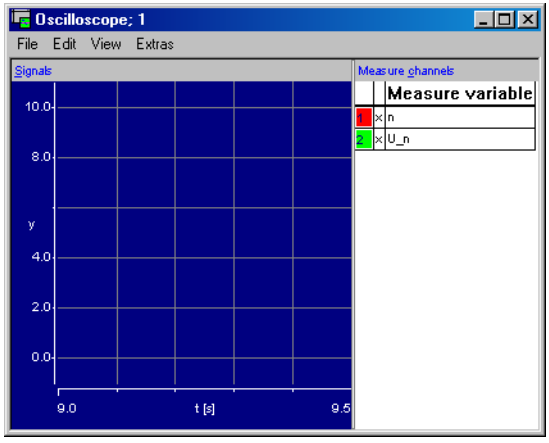
- Erstellen Sie ein Ereignis für jeden Prozess in der Komponente (`n_sampling`, `t_sampling`, `generateData`) und weisen Sie jedem Ereignis einen `dT`-Wert von 4 ms zu.

- Erstellen Sie im Datengenerator einen Kanal für die Message `U_n` und einen für `U_t` und richten Sie beide Kanäle mit den folgenden Werten ein:

Mode:	sinus
Frequency:	2.0 Hz
Phase:	0.0
Offset:	5.0
Amplitude:	5.0

- Erstellen Sie ein Oszilloskopfenster mit den Messages `n` und `U_n` und ein zweites Oszilloskop mit den Messages `t` und `U_t`.

  Vor dem Erstellen des zweiten Oszilloskops müssen Sie darauf achten, dass der Eintrag `<2. Oscilloscope>` im Kombifeld „Select Measure View“ aktiviert ist.



Die Auflösung der Messpunkte und ihrer entsprechenden Interpolationswerte ist so unterschiedlich, dass Sie jeden Kanal in den beiden Oszilloskopen individuell konfigurieren sollten, um dadurch die Darstellungsart des Verhaltens der beiden Tabellen zu optimieren.

Einrichten der Oszilloskope zum Messen:

- Aktivieren Sie das Oszilloskop für den Prozess `n_sampling` (Kanäle `U_n` und `n`).
- Wählen Sie in der Liste „Measure Channels“ die Message `n` und wählen Sie **Extras** → **Setup**.

  Es wird das Dialogfeld „Display Setup“ für die Message `n` angezeigt.
- Stellen Sie den Bereich der Werteachse auf 0 bis 6000 und die Zeitachse auf 0.5 ein.
- Öffnen Sie das Dialogfeld „Display Setup“ für die Message `U_n`.
- Stellen Sie seine Werteachse auf einen Bereich von -1 bis 11 ein.

  Die Zeitachse muss für alle Variablen in einem Oszilloskopfenster gleich sein, und somit brauchen Sie keine Änderung vornehmen.
- Aktivieren Sie das Oszilloskop für den Prozess `t_sampling` (Kanäle `U_t` and `t`) und richten Sie seine Kanäle wie folgt ein:

	`U_t`	`t`
Min	-1	-40
Max	11	200
Extent	0.5	0.5

- Speichern Sie die Experimentierumgebung durch Klicken auf die Schaltfläche **Save Environment** ab.

Sie sind jetzt zur Durchführung des Experimente bereit; dabei können Sie beobachten, wie Ihr Signalwandler arbeitet. Beobachten Sie die Unterschiede zwischen den beiden Wandlungsarten.

Durchführen des Experiments:

- Klicken Sie auf die Schaltfläche **Start Offline Experiment**.

  In der Tabelle `n_sensor` ändert sich einfach die Amplitude der Eingangssinuswelle. Der Eingang ist hier ein Spannungssignal im

Bereich 0 – 10 V; dieses wird auf die Drehzahl im Bereich von 0 –6000 U/min abgebildet. Die Tabelle `t_sensor` stellt keine lineare Beziehung zwischen der Eingangsspannungs und der Ausgangstemperatur dar. Sie entspricht dem charakteristischen Verhalten von Temperatursensoren, die gewöhnlich in der Fahrzeugindustrie zum Einsatz kommen.

- Ändern Sie die Datengenerator-Kanäle in unterschiedliche Wellenformen und beobachten Sie die Auswirkungen auf die beiden Ausgabekurven.

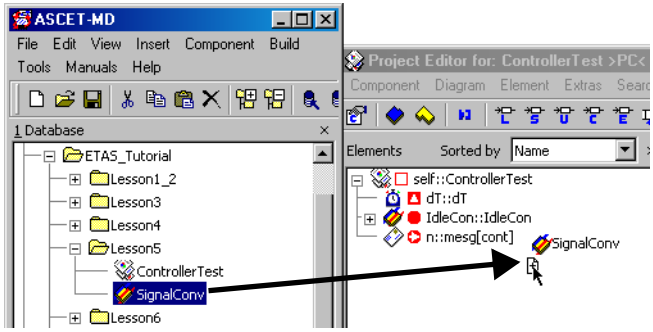
6.5.3 Einbau des Signalwandlers in das Projekt

Nach der Spezifizierung des Signalwandlers können Sie diesen in das von Ihnen in Lektion 4 erstellte Projekt mit einbauen. Das Ausgangssignal des Signalwandlers für die Drehzahl wird als Eingangssignal für den Regler des Motors benutzt.

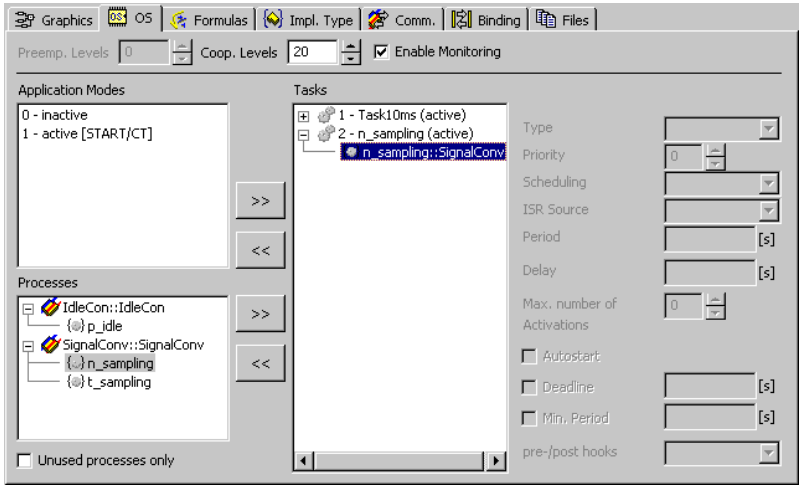
Zum Einbau des Signalwandlers in das Projekt werden Sie eine weitere Task für die neuen Prozesse im Betriebssystem-Schedule einrichten und die für die Kommunikation zwischen den Prozessen notwendigen globalen Elemente vereinbaren sowie die erforderlichen Zuweisungen vornehmen.

Hinzufügen des Signalwandlers zum Projekt:

- Öffnen Sie aus dem Komponentenmanager den Projekteditor für das Projekt `ControllerTest`.
- Ziehen Sie das Modul `SignalConv` aus dem Feld „1 Database“ des Komponentenmanagers auf die Liste „Elements“ des Projekts.



- Klicken Sie auf das Register „OS“ zum Aktivieren des Betriebssystemeditors.
- Erstellen Sie eine neue Task `n_sampling`.
- Stellen Sie als Periode für die neue Task 0.004 Sekunden ein.
- Weisen Sie den Prozess `n_sampling` der Task `n_sampling` zu.



Das Projekt verfügt jetzt über zwei Tasks. Die erste Task wird alle 10 ms aktiviert, und die zweite Task alle 4 ms. Alle einer bestimmten Task zugewiesenen Prozesse werden im angegebenen Intervall ausgeführt. In unserem Beispiel hat jede Task lediglich einen Prozess, aber es ist auch denkbar, dass eine beliebige Anzahl Prozesse pro Task zutrifft.

Der nächste Schritt zum Einbau des Signalwandlers ist die Lösung des Problems der Kommunikation zwischen den Modulen. Die Kommunikation zwischen den Prozessen funktioniert über globale Elemente. Alle im Rahmen eines Projekts einzusetzenden globalen Elemente müssen in den entsprechenden Modulen als Messages definiert werden.

Send-Messages sind standardmäßig in einem Modul definiert, während Receive-Messages gewöhnlich nur in ein Modul importiert werden, folglich müssen sie jetzt im Rahmen des Projekts definiert werden.

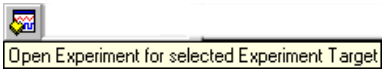
Einrichten der globalen Elemente:

- Wählen Sie **Global Elements → Resolve Globals** zum Aktivieren der automatischen Verknüpfungen.
- Wählen Sie **Global Elements → Delete Unused Globals** zum Entfernen der Verknüpfungen von der vorhergehenden Lektion.

Alle notwendigen globalen Elemente werden erstellt und automatisch mit den entsprechenden Elementen passenden Namens verknüpft. Zum Beispiel wird die globale Message `U_n` automatisch mit der Message `U_n` in `SignalConv` verknüpft.

Dabei ist zu beachten, dass ungenutzte globale Elemente gelöscht werden müssen, da die Message `n` in Lektion 4 im Rahmen des Projekts definiert wurde, während sie jetzt im Modul `SignalConv` definiert ist. Die Message `n` wir jetzt für die Kommunikation zwischen den Prozessen der Module eingesetzt.

Experimentieren mit dem Projekt:



- Klicken Sie auf die Schaltfläche **Open Experiment for selected Experiment Target** zum Aktivieren der Experimentierumgebung.
- Öffnen Sie den Ereignisgenerator und aktivieren Sie die Task `n_sampling`.
- Setzen Sie den `dT`-Wert für die Task auf 4 Millisekunden.

  Während des Offline-Experimentierens mit Projekten simuliert der Ereignisgenerator das Scheduling, das beim Online-Experimentieren durch das Betriebssystem erfolgt.
- Öffnen Sie den Datengenerator und löschen sie den bestehenden Kanal.
- Danach erstellen Sie einen neuen Kanal für die Message `U_n`.
- Richten Sie den Kanal `U_n` wie folgt ein:

Mode:	pulse
Frequency:	1.0 Hz

Phase:	0.0
Offset:	4/3
Amplitude:	1/3

- Aktivieren Sie nun U_n, die Ausgangsspannung des Sensors für die Drehzahl.

  Der Signalwandler wandelt den Spannungswert unter Verwendung der Kennlinie n_sensor in den Istwert für n um.

  Die oben angegebenen Werte erzeugen einen Ausgangsbereich für n, der an den Bereich aus dem vorhergehenden Experiment (ohne Signalverarbeitung) angepasst ist.

- Klicken Sie auf die Schaltfläche **Save Environment**.
- Starten Sie das Experiment.

Die Ausgabekurven müssten die gleichen wie im Beispiel ohne Signalverarbeitung sein. Der vom Datengenerator erzeugte Ansteuerimpuls (Stimulus) ist unterschiedlich, wird dann aber in der Tabelle so verarbeitet, dass er ebenso aussieht wie vorher.

6.5.4 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, die folgenden Aufgaben in ASCET auszuführen:

- Erstellen und Verwenden von Kennlinien
- Hinzufügen von Komponenten zu einem Projekt
- Definieren der Kommunikation zwischen unterschiedlichen Komponenten in einem Projekt.

6.6 Modellierung eines zeitkontinuierlichen Systems

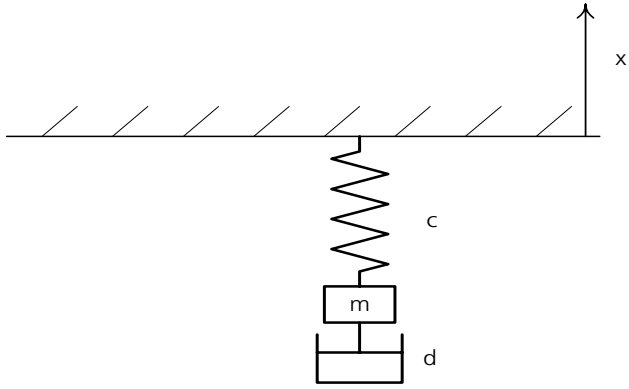
Damit physikalische, mechanische, elektrische und mechatronische Vorgänge, die häufig durch Differentialgleichungen beschrieben werden, realitätsnah modelliert werden können, sind zeitkontinuierliche Verfahren notwendig. Bevor ein solches Verfahren in das in den vorangegangenen Kapiteln erstellte Projekt integriert wird, beschreibt dieses Kapitel die Modellierung eines zeitkontinuierlichen Systems anhand eines detaillierten Beispiels.

ASCET unterstützt die Modellierung und Simulation solcher zeitkontinuierlichen Systeme mit sogenannten CT-Blöcken. CT steht für „Continuous Time“ und bezeichnet Elemente, die quasi zeitkontinuierlich modelliert bzw. berechnet werden. Dabei basiert die zeitkontinuierliche Modellierung in ASCET auf der Zustandsraum-Darstellungsform, der Standardbeschreibungsform beim Design eines zeitkontinuierlichen Systems. Anhand dieser Darstellung lassen sich CT-Basisblöcke mit nichtlinearen gewöhnlichen Differentialgleichungen erster Ordnung und nichtlinearen Ausgabegleichungen beschreiben. ASCET stellt mehrere echtzeitfähige Integrationsverfahren zur Verfügung, um diese Differentialgleichungen optimal zu lösen (Näheres dazu finden Sie im Kapitel 8.2 „Lösen von Differentialgleichungen - Integrationsalgorithmen“ des ASCET Referenzhandbuchs).

Die Vorgehensweise bei der Modellierung eines zeitkontinuierlichen Systems soll nun anhand eines Masse-Feder-Pendels mit Dämpfung im Schwerefeld der Erde dargestellt werden.

6.6.1 Bewegungsgleichung

Die in der Abbildung dargestellte Masse `m` unterliegt folgenden Kräften:



- Schwerkraft: $F_g = -mg$
  (g = Erdbeschleunigung)
- Federkraft: $F_F = -c\,(x + l_0)$
  (c = Federkonstante, l_0 = Länge der ungespannten Feder und x = Lage der Masse m)
- Dämpfungskraft $F_D = -d\,x'$
  (d = Dämpfungskonstante und x' = Geschwindigkeit der Masse)

Damit lautet die Bewegungsgleichung:

$mx'' = -mg + F$ oder $x'' = -g + F/m$ (mit $F = F_F + F_D$)

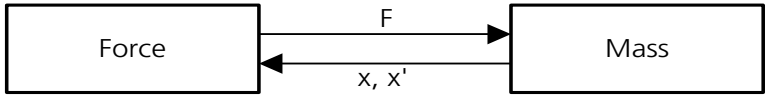
Die Zerlegung der einen Differentialgleichung 2. Ordnung in zwei Differentialgleichungen 1. Ordnung (`x = x, v = x'`) ergibt:

```
x' = v
v' = -g + F/m
```

Diese Differentialgleichungen werden in der nun folgenden Modellierung verwendet.

6.6.2 Modellierung

Der Einfachheit halber würde man das Modell des Masse-Feder-Pendels in einem einzelnen CT-Block realisieren. Zur Verdeutlichung der Eigenschaften „direkter Durchgang" bzw. „nichtdirekter Durchgang" und zur Demonstration, wie durch geschickte Vergabe dieser Eigenschaften eine algebraische Schleife vermieden werden kann, wird dieses Modell mit zwei Blöcken ausgeführt.



- Der Block `Force` berechnet die Federkraft `F` aus der Lage der Pendelmasse m und die Reibungskraft aus der Geschwindigkeit `x'.`
- Der Block `Mass` berechnet aus dieser Federkraft F die Beschleunigung `x''` aus der sich durch Integration die Geschwindigkeit `x'` und die Lage `x` ergeben.

Auf den ersten Blick stellt sich dieses System als algebraische Schleife dar - jeder Block erwartet eine Eingangsgröße vom jeweils anderen Block, um daraus wieder eine Ausgangsgröße zu berechnen.

Durch geschickte Vergabe der Eigenschaften *direkter Durchgang* bzw. *nichtdirekter Durchgang* kann diese algebraische Schleife aufgelöst werden:

- Im Block `Force` hängt die Ausgangsgröße `F` über die Gleichung

  $F = -c(x + l_0) - dx'$

  direkt von den Eingangsgrößen `x` und `x'` ab. Dieser Block ist also mit direktem Durchgang vereinbart.
- Im Block `Mass` hängen die Ausgangsgrößen `x` und `x'` dagegen nicht direkt von der Eingangsgröße `F` ab, sondern nur von internen Zustandsgrößen des Blocks. Für diese gibt es zumindest beim Startzeitpunkt Anfangswerte, aus denen die Ausgangsgrößen `x` und `x'` berechnet werden können, ohne dass die Eingangsgröße `F` bekannt ist.

Ansonsten erfolgt die Berechnung der Ausgangsgrößen über die folgenden Differentialgleichungen:

```
x' = v
v' = -g + F/m
```

Dieser Block ist also als nichtdirekter Durchgang vereinbart.

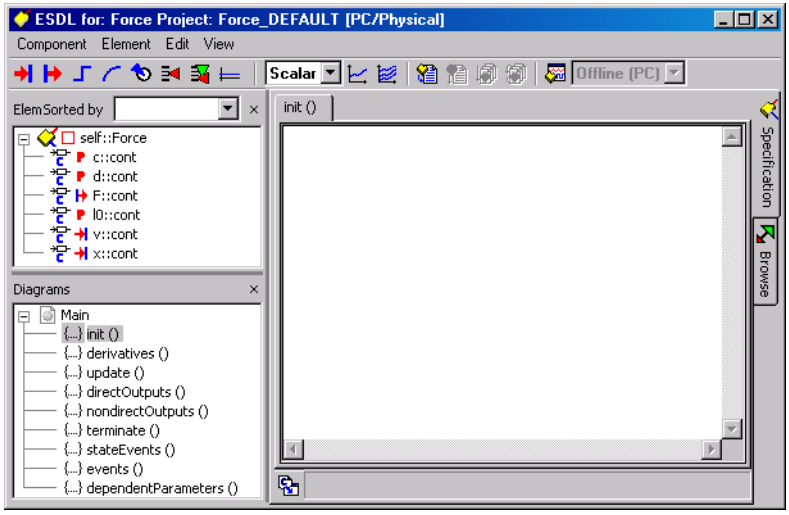
Erstellen des Modells

- Legen Sie im Komponentenmanager mit **Insert** → **Folder** einen Ordner an und benennen Sie ihn mit `Lesson6`.
- In diesem Ordner erstellen Sie mit **Insert** → **Continuous Time Block** → **ESDL** einen Block `Force` und einen Block `Mass`.
- Öffnen Sie den ESDL-Editor mit einem Doppelklick auf den Block `Force`.

- Legen Sie durch Klicken auf die Schaltfläche **Input** zwei Eingänge `x` und `v` an (vom Typ `continuous`).

- Legen Sie durch Klicken auf die Schaltfläche **Output** einen Ausgang `F` an (vom Typ `continuous`).

- Legen Sie durch Klicken auf die Schaltfläche **Parameter** die Konstanten `c` (Federkonstante), `d` (Dämpfungskonstante) und `l0` (Länge der ungespannten Feder) an.

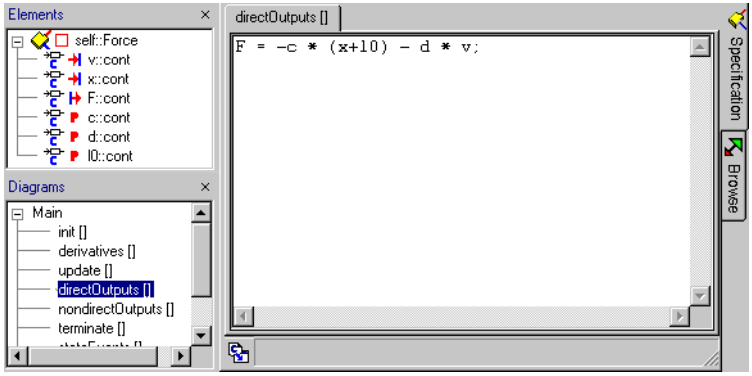


Die Methoden im Feld „Diagrams“ sind fest vorgegeben.

- Markieren Sie durch Mausklick nacheinander jeweils eine der Konstanten im Feld „Elements“.
- Ein Klick mit der rechte Maustaste auf die jeweils markierte Konstante öffnet ein Kontextmenü.
- Wählen Sie den Menüpunkt **Edit Data.**

  Es öffnet sich das Dialogfenster „Numeric Editor“.
- Weisen Sie darin den Konstanten sinnvolle Werte zu (z.B. 5.0 für die Federkonstante `c`, 1.0 für die Dämpfungskonstante `d` und 2.0 für die Länge der ungespannten Feder `l0`).

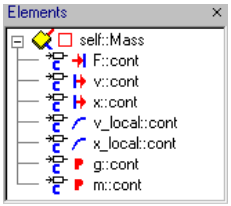
- Klicken Sie im Feld „Diagrams“ auf die Methode `directOutputs[]` und geben Sie im Edit-Feld die Formel an, mittels derer die Kraft berechnet wird:
  `F = -c * (x + l0) - d*v;`



- Klicken Sie auf die Schaltfläche **Generate Code**.

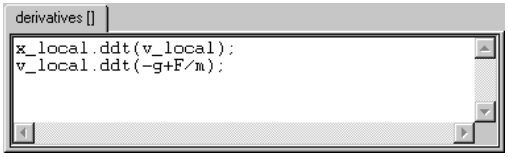
  Der CT-Block `Force` wird kompiliert.
- Öffnen Sie einen weiteren ESDL-Editor mit einem Doppelklick auf den Block `Mass`.
- Erstellen Sie wie oben einen Eingang `F`, zwei Ausgänge `x` und `v`, einen Parameter `m` (Masse) und eine Konstante `g` (Erdbeschleunigung).
- Weisen Sie wie oben beschrieben `g` und `m` Werte zu (9.81 für g und z.B 2.0 für die Masse `m`).

- Erstellen Sie durch Klicken auf die Schaltfläche **Continuous State** zwei Zustandsvariablen `x_local` und `v_local` für die interne Berechnung der Ausgänge.

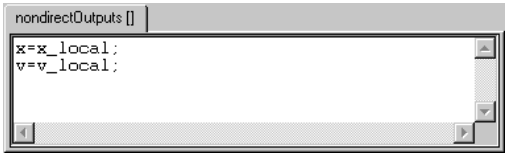


- Geben Sie bei der Methode `derivatives[]` die zur Berechnung notwendigen Differentialgleichungen an:

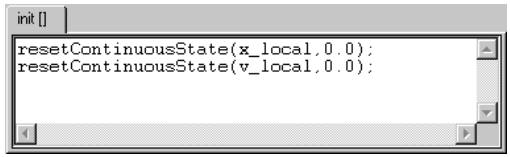
```
x_local.ddt(v_local);
v_local.ddt(-g + F/m);
```



- Übergeben Sie in `nondirectOutputs[]` die Zustandsvariablen `x_local` und `v_local` an die Ausgänge `x` und `v`.



- In der Methode `init[]` können Sie dem System mit der Funktion `resetContinuousState()` sinnvolle Anfangswerte für x und v übergeben.



- Klicken Sie auf die Schaltfläche **Generate Code**.

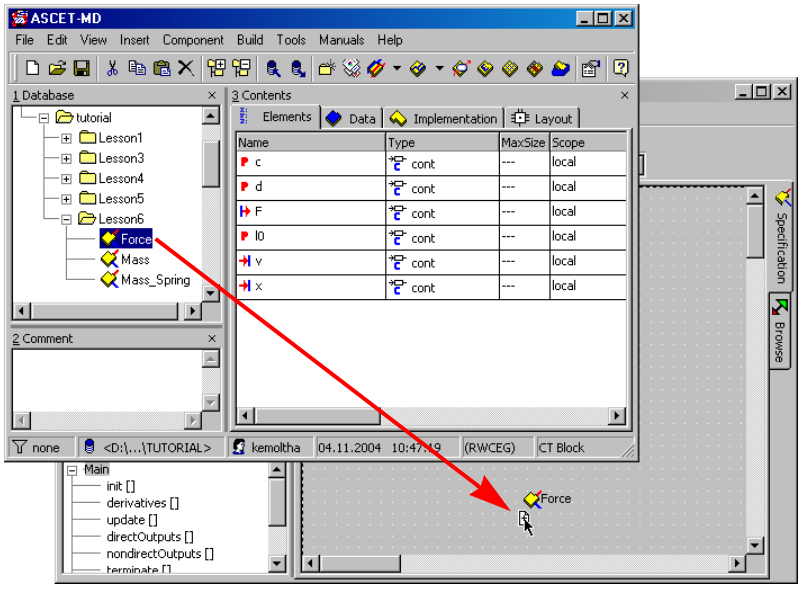
  Der CT-Block `Mass` wird kompiliert.

Die Zusammenführung der beiden CT-Basisblöcke zu einem CT-Strukturblock erfolgt nun im Blockdiagrammeditor (BDE).

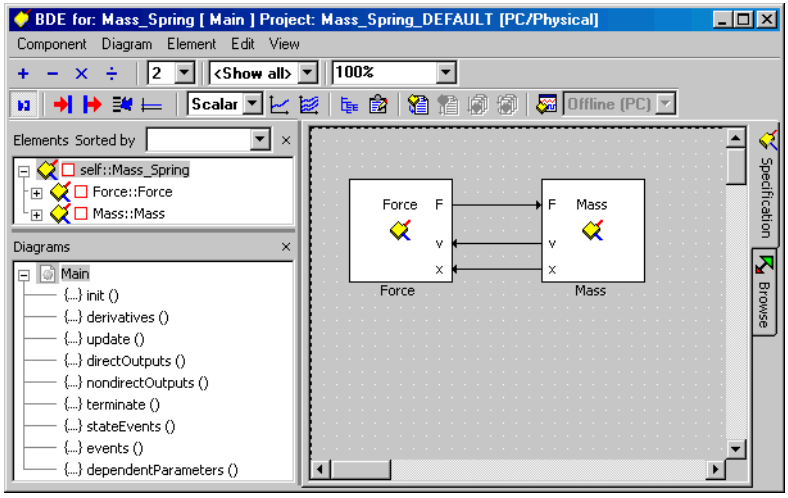
Zusammenfügen der beiden CT-Basisblöcke:

- Gehen Sie dazu in Ihr Verzeichnis im Datenbank Browser.
- Legen Sie mit **Insert** → **Continuous Time Block** → **Block Diagram** einen neuen Block `Mass_Spring` an.
- Öffnen Sie diesen neuen Block im Blockdiagrammeditor durch Doppelklick.

- Ziehen Sie im Komponentenmanager nacheinander die beiden Blöcke `Mass` und `Force` auf das Fenster des BDE und legen Sie beide dort ab.

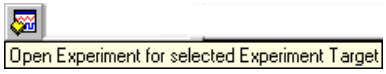


- Verbinden Sie die entsprechenden Ein- und Ausgänge miteinander.



Hinweis

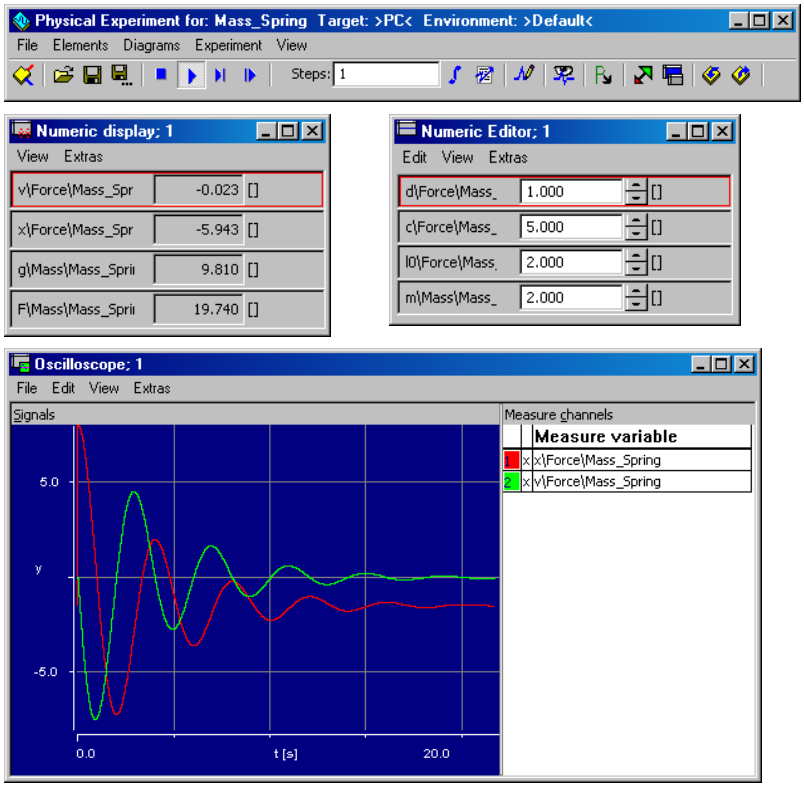
Durch Doppelklick auf einen der CT-Basisblöcke kann dieser bearbeitet werden. Beachten Sie dabei aber, dass sich Änderungen an diesen auf die gesamte Bibliothek auswirken, d.h. auf alle Strukturblöcke, in denen diese Basisblöcke verwendet werden.



- Klicken Sie auf die Schaltfläche **Open Experiment for selected Experiment Target**.

  Der CT-Block wird kompiliert und das Experiment gestartet.

- Erstellen Sie sich die gewünschte Experimentierumgebung mit numerischen Editoren für die Parameter und grafischen Anzeigen.



- Skalieren Sie die beiden Kanäle im Oszilloskop getrennt; x von -10 bis 0, v von -8 bis +8.

6.6.3 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, in ASCET die folgenden Aufgaben auszuführen:

- Erstellen eines Modells zur Simulation einer Strecke
- Erstellen von CT-Blöcken im ESDL-Editor mit direktem und nichtdirektem Durchgang
- Zusammensetzen von CT-Blöcken im Blockdiagrammeditor
- Durchführung des physikalischen Experiments

6.7 Ein Streckenmodell

Nachdem im letzten Kapitel die CT-Blöcke eingeführt wurden, werden Sie diese jetzt bei der Erprobung Ihres Reglers einsetzen. In ASCET können Sie ein Modell des zu regelnden technischen Prozesses entwickeln und dann mit einem Regelkreis experimentieren. Auf diese Weise lässt sich der Regler vor dem Einsatz in einem realen Fahrzeug gründlich erproben.

Im vorliegenden Beispiel ist der Motor der technische Prozess. Er stellt einen Wert `U_n` bereit, bei dem es sich um einen Sensormesswert der Drehzahl des Motors handelt. Dieser Wert wird vom Regler verarbeitet, der daraufhin einen Wert `air_nominal` bereitstellt. Der Reglerausgang bestimmt die Drosselklappenstellung des Motors und beeinflusst auf diese Weise die Drehzahl.

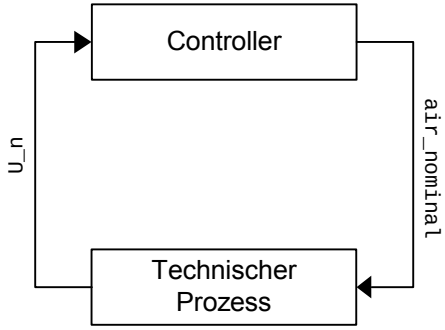


Abb. 6-1 Ein Experiment im Regelkreis

Für dieses Streckenmodell werden Sie einen CT-Block benutzen. Dieser Komponententyp ist für Streckenmodelle besonders geeignet. Das Modell basiert auf der folgenden Differentialgleichung, die ein PT2 - System modelliert:

$$T^2\ s'' + 2DTs' + s = Ku$$

Gln. 6-1 Ein PT2 - System

Den Parametern T, D und K sind entsprechende Werte zuzuweisen.

6.7.1 Spezifizieren des Streckenmodells

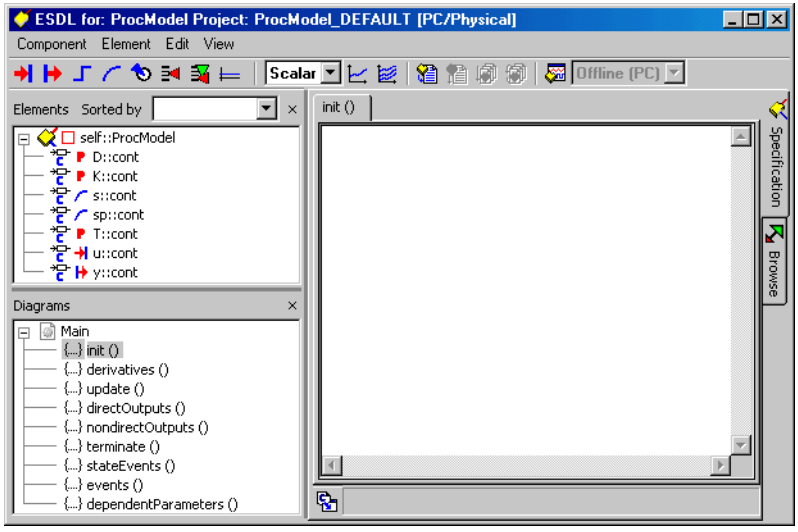
Das Erstellen zeitkontinuierlicher Komponenten unterscheidet sich vom Erstellen anderer Komponenten. Sie haben Eingänge und Ausgänge, die Argumenten und Rückgabewerten gleichzusetzen sind. Der Hauptunterschied besteht darin, dass ein CT-Block mehrere Ein- und Ausgänge haben kann, die nicht an eine bestimmte Methode gebunden sind. In jedem CT-Block gibt es einen festen Satz von Methoden, die vom Anwender nicht verändert werden können.

Für das vorliegende Beispiel werden Sie den ESDL-Code benutzen. Die Syntax des ESDL-Codes ist ähnlich der von C++ oder Java. Eine Methode eines Objekts wird aufgerufen mit dem Namen des Objekts, einem Punkt, dem Namen der Methode und den Argumenten in Klammern gefolgt von einem Semikolon. Die zum Ableiten verwendete Methode wird `ddt()` genannt. Beispielsweise ist die Gleichung $sp = \dot{s}$ gleich der ESDL-Anweisung `s.ddt(sp);`.

Erstellen einer kontinuierlichen Zeitkomponente:

- Erstellen Sie im Komponentenmanager den Ordner `Tutorial\Lesson7`.
- Zum Hinzufügen eines CT-Blocks wählen Sie **Insert** → **Continuous Time Block** → **ESDL**.
- Nennen Sie die neue Komponente `ProcModel`.
- Wählen Sie **Component** → **Edit Item** zum Öffnen des ESDL-Editors.

  Selbstverständlich können Sie auch den externen Texteditor verwenden. Die Handhabung ist im ersten Teil des Tutoriums beschrieben.



Zum Bearbeiten des Streckenmodells werden Sie zuerst die erforderlichen Elemente hinzufügen und dann die Methoden `derivatives` und `nondirectOutput` bearbeiten.

Bearbeiten des Streckenmodells:

- Erstellen Sie im ESDL-Editor mit der Schaltfläche **Continuous State** zwei kontinuierliche Zustände.
- Benennen Sie die Zustände mit `s` und `sp`.

- Erstellen Sie einen Eingang durch Klicken auf die Schaltfläche **Input**.
- Benennen Sie den Eingang mit `u`.

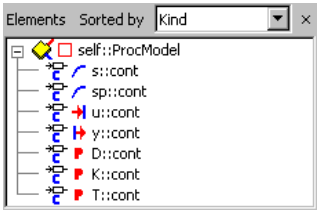
Erstellen Sie einen Ausgang durch Klicken auf die Schaltfläche **Output**.

- Benennen Sie den Ausgang mit `y`.

Beide Elemente sind vom Typ `cont`.

- Erstellen Sie einen Parameter durch Klicken auf die Schaltfläche **Parameter**.
- Benennen Sie den Parameter mit `D`.
- Erstellen Sie die Parameter `K` und `T`.

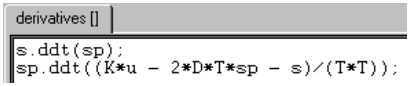
Die Liste „Elements“ für das Streckenmodell müsste wie folgt aussehen:



- Stellen Sie die Parameter wie folgt ein:

```
D = 0.4,
K = 0.002,
T = 0.05.
```

- Wählen Sie in der Liste „Diagrams“ die Methode `derivatives` und bearbeiten Sie den Code wie folgt:



  Die Abbildung stellt den internen Texteditor dar.

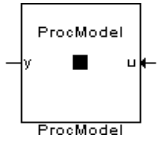
Hinweis

Angaben zum Lösen einer Differentialgleichung siehe Tab. 9-1.in Kapitel 9.1 des ASCET Referenzhandbuchs.

- Wählen Sie in der Liste „Elements“ die Methode `nondirectOutputs` und geben Sie den folgenden Text ein.

- Richten Sie das Layout im Layout-Editor ein.

  Dabei ist zu beachten, dass bei einem Streckenmodell vorzugsweise die Ausgaben links und die Eingaben rechts angeordnet werden.



- Wählen Sie **Edit** → **Save**.
- Klicken Sie im Komponentenmanager auf die Schaltfläche **Save** zum Abspeichern des Streckenmodells.

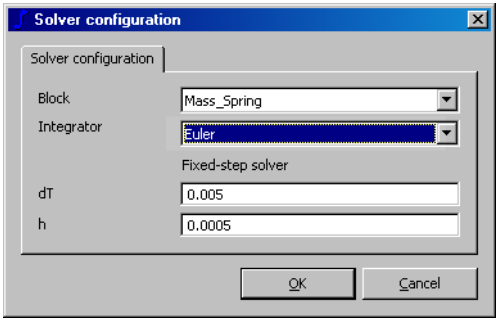
Sie können jetzt mit dem Experimentieren mit dem neuen Modell beginnen.

Experimentieren mit dem Modell:

- Wählen Sie im ESDL-Editor **Component** → **Open Experiment** zum Öffnen der Experimentierumgebung.

- Klicken Sie auf die Schaltfläche **Open CT Solver** zum Öffnen des Dialogfelds „Solver Configuration".

  Die Konfiguration wird wie folgt dargestellt:



- Klicken Sie auf **OK** zur Übernahme der standardmäßigen Konfiguration.
- Öffnen Sie den Datengenerator und erstellen Sie einen Kanal für die Eingabe u.
- Richten Sie den Kanal u mit den folgenden Werten ein:

Mode:	pulse
Frequency:	0.5 Hz
Phase:	0.0 s
Offset:	-0.5
Amplitude:	1.0

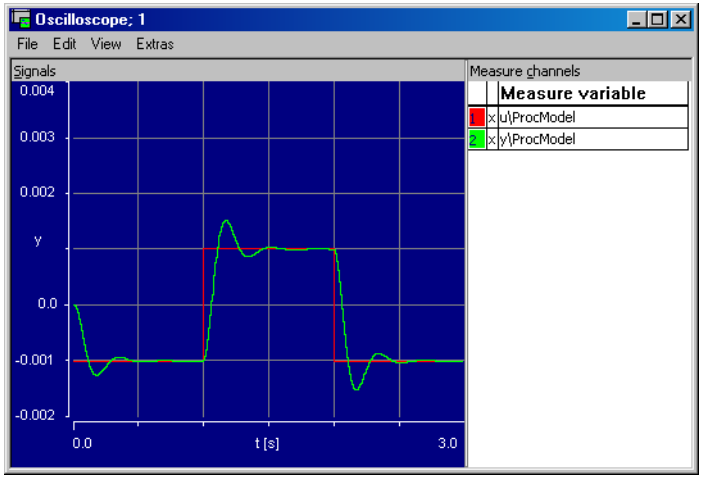
- Öffnen Sie ein Oszilloskopfenster mit den Kanälen u und y.

- Stellen Sie die Messkanäle für das Oszilloskop wie folgt ein:

	u	y
Min	-1	-0.002
Max	2	0.004
Extent	3.0	3.0

- Klicken Sie auf die Schaltfläche **Save Environment**.
- Beginnen Sie mit dem Experiment.

  Die Ausgabe müsste wie folgt aussehen:



6.7.2 Integration des Streckenmodells

Zum Erstellen eines Regelkreises werden wir nun das Streckenmodell in das früher von uns erstellte Reglerprojekt integrieren. Die dazu erforderlichen Schritte sind die gleichen wie zuvor: Aufnahme des Moduls, Einrichten des Betriebssystems und Verknüpfen der globalen Elemente.

Hinweis

Aus Gründen der Vereinfachung wird das Streckenmodell hier zum gleichen Projekt hinzugefügt. Dies ist häufig im Frühstadium der Erprobung der Simulation eines Regelkreises von Nutzen. Bei regulären Projekten wird das Streckenmodell über ein Netzwerk in einem anderen Projekt verteilt, da sie nicht Bestandteil des gleichen eingebetteten Systems sind.

Einfügen des Streckenmodells:

- Öffnen Sie aus dem Komponentenmanager heraus den Projekteditor für `Controller-Test`.
- Fügen Sie im Projekteditor die Komponente `ProcModel` zur Liste „Elements“ hinzu.

- Aktivieren Sie das Register „OS“ des Projekteditors, um das Scheduling für die CT-Tasks zu spezifizieren.
- Wählen Sie die Task `simulate_CT1` und setzen Sie den Wert im Feld „Period“ auf 0.01 s.

  Der Regler und das Streckenmodell laufen beide im gleichen Zeitintervall.

Die Verknüpfung zwischen CT-Blöcken und Modulen kann nicht automatisch hergestellt werden. Sie müssen explizit in einem Blockdiagramm miteinander verknüpft werden.

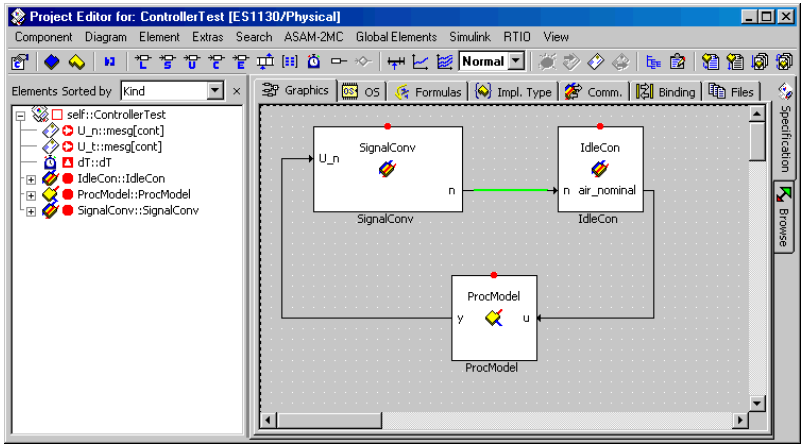
Einstellen der Verknüpfung zwischen Modulen und CT-Block:

- Klicken Sie auf das Register „Graphics“.
- Ziehen Sie aus der Liste „Elements“ die drei Komponenten und legen Sie diese im Zeichenbereich ab.

- Verbinden Sie die Messages der Module mit dem entsprechenden Eingang und Ausgang des CT-Blocks.

  Um das Beispiel aufzubauen, verbinden Sie den Ausgang `y` von `ProcModel` mit der globalen Message `U_n` und verbinden Sie den Eingang `u` von `ProcModel` mit der globalen Message `air_nominal`.

- Klicken Sie mit der rechten Maustaste auf jede Komponente und wählen Sie **Unconnected Ports** zum Entfernen dieser Ports aus dem Diagramm.



Die Verknüpfung der Messages zur Kommunikation zwischen Modulen wird automatisch hergestellt. Es werden Messages gleichen Namens miteinander verknüpft.

Somit ist das Projekt komplett, und es kann damit experimentiert werden. Sie werden jetzt online experimentieren, und dazu sind eine ASCET-RP-Installation und ein Echtzeittarget (z.B. ES1000) erforderlich. Wenn Sie darüber nicht verfügen, müssen Sie wie bisher offline experimentieren.

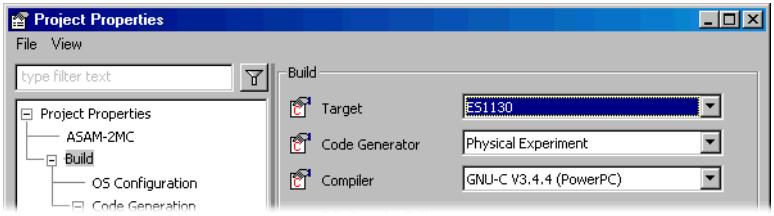
Hinweis

Wenn Sie weiterhin offline experimentieren, müssen Sie die globale Message U_n *aus dem Datengenerator entfernen.*

Einrichten des Projekts zum Online-Experimentieren:

- Klicken Sie auf die Schaltfläche **Project Properties**.
- Wählen Sie im Dialogfenster „Options“ im Knoten „Build“ ein Experimentaltarget (z.B. `ES1130`) und den Compiler `GNU-C * (Power-PC)` aus.

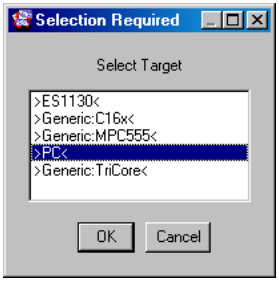
  Diese Optionen geben die Hardware und den entsprechenden Compiler zum Generieren des Codes an.



- Klicken Sie auf **OK**, um das Fenster zu schließen.

  Die Schaltflächen **Open Experiment for selected Experiment Target** und **Reconnect to Experiment of selected Experiment Target** sind nun verfügbar.

- Klicken sie auf das Register „OS“ zum Aktivieren des Betriebssystemeditors.
- Zum Kopieren des früher von Ihnen erstellten Schedules wählen Sie **Operating System** → **Copy From Target**



- Wählen Sie aus dem Dialogfeld „Selection Required“ den Eintrag **PC** und klicken sie auf **OK**.

  Jetzt hat das Projekt für das neue Target das gleiche Scheduling, das zuvor für die Offline-PC-Simulation vorgegeben wurde.

Es gibt mehrere Unterschiede zum Offline-Experiment. Es gibt im Online-Experiment keinen Ereignis- oder Datengenerator. Der Ereignisgenerator dient zur Simulation des Schedulings der Betriebssystemtasks, das für Online-Experimente generiert wird.

Im Online-Experiment werden der Experimentiercode und die Messungen separat gestartet, und sie haben separate Schaltflächen in der Symbolleiste. Dies ist erforderlich, weil die Messungen das Echtzeitverhalten des Experiments beeinflussen können, und folglich müssen sie u. U. manchmal deaktiviert werden.

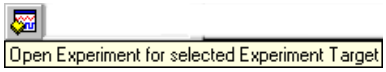
Online-Experimentieren mit dem Projekt:

- Wählen Sie aus dem Kombikästchen „Experiment Target“ den Eintrag `Online (RP)`.

  Der Eintrag `Offline (RP)` ist für Offline-Experimente auf dem Target gedacht.

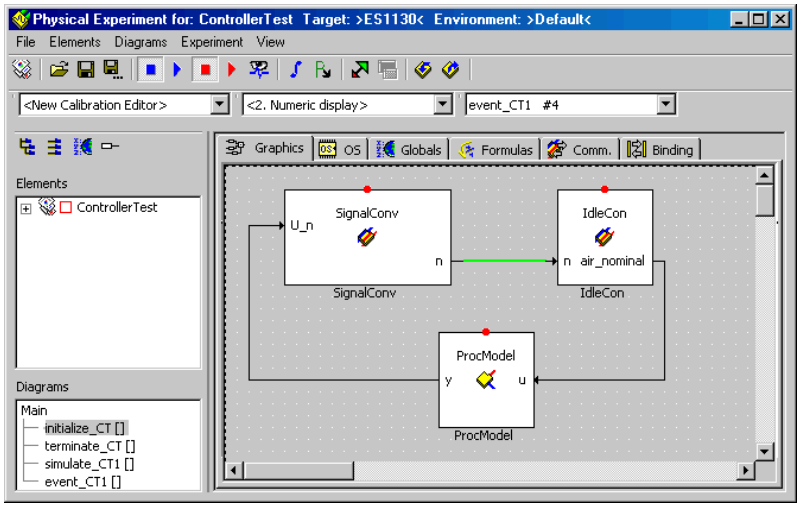
- Wählen Sie **Component** → **Open Experiment**

oder



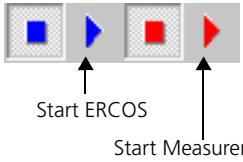
- klicken Sie auf die Schaltfläche **Open Experiment for selected Experiment Target**.

  Es wird der Code für das Experiment erzeugt, und das Experiment wird mit der gleichen Umgebung wie vorher definiert geöffnet.



Wenn Ihr Projekt mehrere Tasks enthält, werden Sie unter Umständen aufgefordert, für jeden Messwert eine Erfassungstask auszuwählen.

- Wählen Sie im Fenster „Selection Required“ die Task `#3 simulate_CT1` und klicken Sie **OK**.



- Nehmen Sie `n` und `n_nominal` in das vorhandene Oszilloskop mit auf und setzen Sie deren Wertebereich auf 0 bis 2000.
- Öffnen Sie numerische Editoren für die Variablen `n_nominal`, `Ki` und `Kp`.
- Klicken Sie nacheinander auf die Schaltflächen **Start ERCOS** und **Start Measurement**.

  Das Experiment wird gestartet, und die Ergebnisse werden auf dem Oszilloskop angezeigt. Der Wert für `n` müsste sich schnell `n_nominal` nähern und dort bleiben.
- Ändern Sie `n_nominal` im numerischen Editor.

  Der Wert `n` müsste sich entsprechend allen Änderungen von `n_nominal` ändern.
- Sie können das Verhalten des Regelkreises durch Verstellen der Parameter `Ki` und `Kp` optimieren.

6.7.3 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, die folgenden Aufgaben in ASCET auszuführen:

- Erstellen und Spezifizieren von CT-Blöcken
- Experimentieren mit CT-Blöcken
- Einbau von CT-Blöcken in ein Projekt
- Erstellen von Verknüpfungen von Variablen
- Umschalten zwischen verschiedenen Targets
- Online-Experimentieren mit einem Projekt

6.8 Zustandsautomaten

Zustandsautomaten sind von Nutzen zum Modellieren von Systemen, die sich zwischen einer begrenzen Anzahl verschiedener Zustände bewegen. ASCET stellt ein leistungsfähiges Mittel zum Spezifizieren von Komponenten als Zustandsautomaten dar. In dieser Lektion werden wir einen einfachen Zustandsautomaten spezifizieren und erproben, der eine temperaturabhängige Änderung der Nenndrehzahl eines sich im Leerlauf befindlichen Motors

implementiert. Dieser Zustandsautomat wird danach in unser Projekt integriert. Im nächsten Kapitel werden dann hierarchische Zustandsautomaten aufgebaut.

Bei einem kalten Motor ist eine höhere Leerlaufdrehzahl erforderlich, damit der Motor nicht „ausgeht". Nach dem Warmlaufen des Motors kann die Leerlaufdrehzahl zwecks Kraftstoffeinsparung verringert werden. Somit hat unser Zustandsautomat zwei Zustände: kalter Motor und warmer Motor. Dies stellt eine Zweiphasenregelung dar.

6.8.1 Spezifizieren des Zustandsautomaten

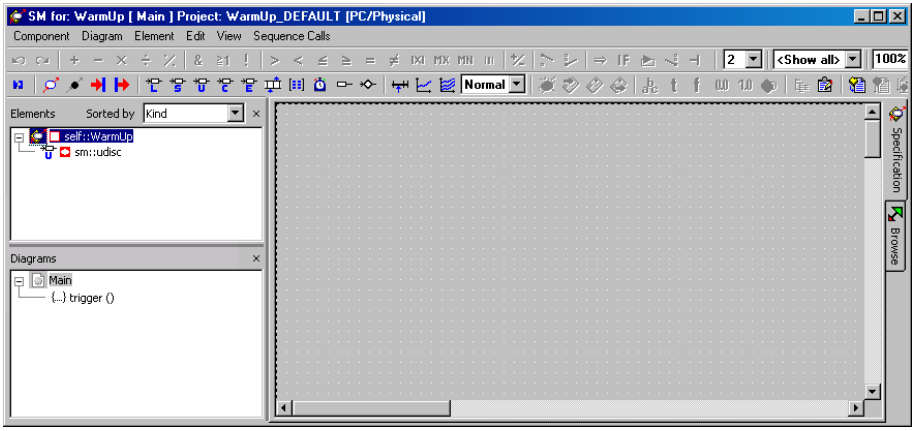
Ein Zustandsautomat besteht aus dem eigentlichen Zustandsdiagramm sowie einer Reihe von Spezifikationen von Aktionen und Bedingungen. Die Aktionen und Bedingungen lassen sich unter Verwendung von Blockdiagrammen oder ESDL-Code spezifizieren. Sie legen fest, was in den verschiedenen Zuständen sowie während der Übergänge zwischen Zuständen geschieht.

Die Diagramme werden im Blockdiagrammeditor spezifiziert. Eine weitere Möglichkeit besteht darin, einen ESDL-Code direkt in einen Texteditor zu schreiben, der für jeden Zustand und jeden Übergang geöffnet werden kann (d.h. *ohne* Öffnen des ESDL-Editors). Zustandsautomaten haben Eingänge und Ausgänge für den Datenaustausch mit anderen Komponenten.

Erstellen eines Zustandsautomaten:

- Erstellen Sie im Komponentenmanager den Ordner `Tutorial\Lesson8`.
- Wählen Sie **Insert** → **State Machine** oder klicken Sie auf die Schaltfläche **State Machine**, um einen neuen Zustandsautomaten zu erstellen.
- Benennen Sie diesen mit `WarmUp`.

- Doppelklicken Sie in der Liste „1 Database“ auf den Namen des Zustandsautomaten, um den Zustandsautomaten-Editor zu öffnen.



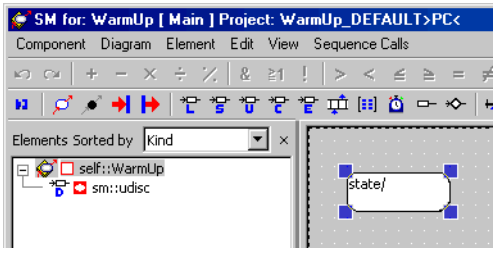
Zum Erstellen eines Zustandsautomaten spezifizieren Sie zuerst das Zustandsdiagramm und definieren danach die verschiedenen Aktionen und Bedingungen im Zusammenhang mit Zuständen und Zustandsübergängen.

Der Zustandsautomat, der Ihren Motor regelt, verfügt über zwei Zustände: einen für den kalten Motor und einen für den warmen Motor.

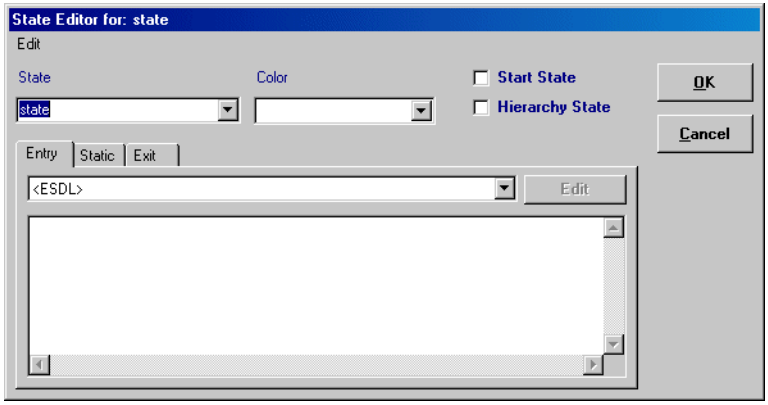
Spezifizieren des Zustandsdiagramms:

- Klicken Sie auf die Schaltfläche **State**, um den Cursor mit einem Zustand zu laden.
- Klicken Sie an der Stelle in den Zeichenbereich, wo Sie den Zustand anordnen möchten.

  An die angeklickte Stelle wird ein Zustandssymbol gezeichnet.



- Erstellen Sie ein zweites Zustandssymbol und platzieren Sie dieses unterhalb des ersten Zustandssymbols im Zeichenbereich.
- Klicken sie mit der rechten Maustaste auf das zuerst erstellte Zustandssymbol (das obere) und wählen Sie zum Öffnen des Zustandseditors **Edit State** aus dem Kontextmenü.



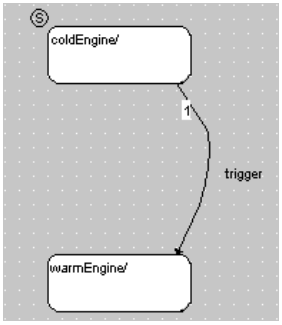
- Geben Sie im Feld „State“ den Namen `coldEngine` ein.
- Aktivieren Sie die Option **Start State**, um den Zustand des Automaten beim ersten Start zu ermitteln.

  Jeder Zustandsautomat muss über einen Startzustand verfügen.
- Klicken sie auf **OK** zum Schließen des Zustandseditors.

  Der Name wird im Zustandssymbol angezeigt.
- Benennen Sie das zweite Zustandssymbol mit `warmEngine`.
- Klicken Sie mit der rechten Maustaste in den Zeichenbereich außerhalb eines Symbols zum Aktivieren des Verknüpfungsmodus.

- Klicken Sie in die rechte Hälfte des Zustandssymbols `coldEngine`, um eine Verknüpfung aufzubauen; danach klicken Sie in die rechte Hälfte des Zustandssymbols `warmEngine`, um die beiden Zustände zu verknüpfen.

  Zwischen den beiden Zustandssymbolen wird eine Linie gezogen. Sie besitzt an einem Ende einen Pfeil, der vom oberen zum unteren Symbol zeigt. Die Linien stellen mögliche Übergänge zwischen Zuständen dar.



- Erstellen Sie einen weiteren Übergang von der linken Hälfte des unteren zur linken Hälfte des oberen Symbols.
- Wählen Sie **Diagram** → **Store to Cache**, um das Diagramm zu sichern.
- Wählen Sie im Komponentenmanager **File** → **Save Database**, um die Änderungen in die Datenbank zu übernehmen.

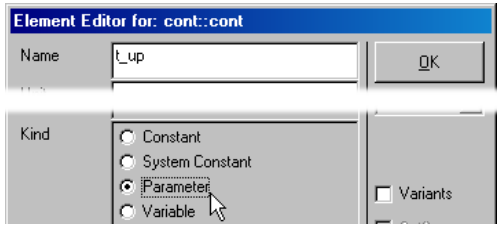
Der nächste Schritt beim Bauen des Zustandsautomaten ist das Spezifizieren seiner Schnittstelle. Sie benötigen eine Eingabe für den Temperaturwert und eine Ausgabe für die Drehzahl. Darüber hinaus sind Parameter erforderlich, mit denen der Temperaturoberwert und -unterwert sowie die Drehzahl spezifiziert werden.

Spezifizieren der Schnittstelle des Zustandsautomaten:

- Erstellen Sie einen Eingang durch Klick auf die Schaltfläche **Input**.
- Benennen Sie den Eingang mit `t`.

- Erstellen Sie einen Ausgang durch Klick auf die Schaltfläche **Output**.
- Benennen Sie den Ausgang mit `n_nominal`.

- Legen Sie mit der Schaltfläche **Continuous** eine Variable an.
- Geben Sie der Größe im Elementeditor den Namen `t_up` und aktivieren Sie die Option **Parameter**.



Damit ist die ursprünglich angelegte Variable in einen Parameter umgewandelt.

- Erstellen Sie auf die gleiche Weise drei weitere Parameter.
- Benennen Sie die Parameter und setzen Sie deren Werte wie folgt:

  `t_up` = 70

  `t_down` = 60

  `n_cold` = 900

  `n_warm` = 600

Danach können Sie die Aktionen und Bedingungen für die Zustände sowie für die Übergänge zwischen Zuständen spezifizieren. Für jeden Zustand können Sie drei Aktionen spezifizieren:

- Die Eintrittsaktion wird bei jedem Eintreten des Zustands ausgeführt.
- Die Austrittsaktion wird bei jedem Verlassen des Zustands ausgeführt.
- Die statische Aktion wird ausgeführt, während sich der Automat im stationären Zustand befindet.

Analog können für jeden Übergang ein Triggerereignis, eine Bedingung, eine Priorität und eine Aktion spezifiziert werden. Der Name des Triggers und der Bedingung erscheinen neben dem Übergang. Beim Erstellen des Zustandsautomaten wird automatisch *ein* Trigger erzeugt.

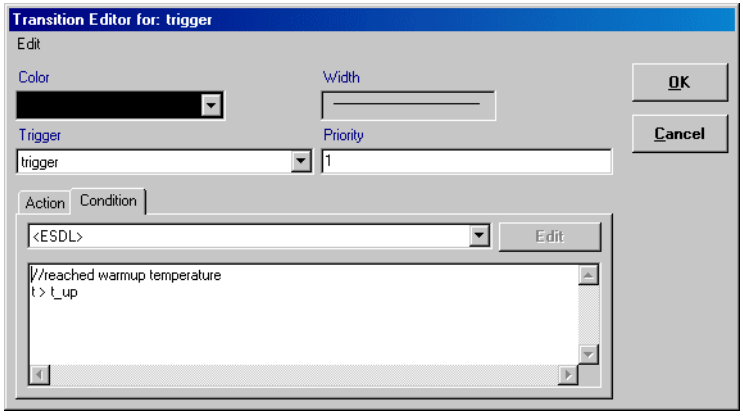
Die Aktionen und Bedingungen werden in gewöhnlichen Diagrammen oder im ESDL-Code spezifiziert. Im vorliegenden Beispiel werden Sie ESDL-Code verwenden.

Spezifizieren der Trigger-Aktionen und -Bedingungen:

- Klicken Sie mit der rechten Maustaste auf den Übergang vom Zustand `coldEngine` in den Zustand `warmEngine`.
- Wählen Sie im Kontextmenü **Edit Transition** zum Öffnen des Übergangseditors.

  Die Bedingung für einen Übergang von kalt zu warm besteht darin, dass der Temperaturistwert `t` größer als `t_up` ist.
- Wählen Sie aus dem Kombifeld im Register „Condition“ den Eintrag `<ESDL>`.

  Beachten Sie, dass Sie die voreingestellte Auswahl dieses Kombifelds in den Optionen beeinflussen können.
- Geben Sie den untenstehenden Code in das Codefeld der Bedingung ein:



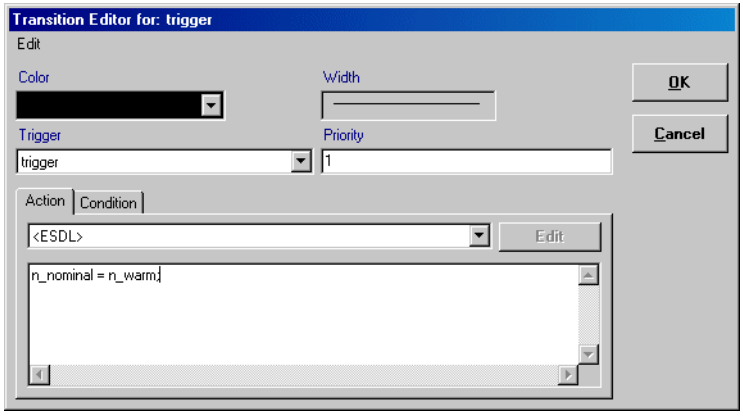
Hinweis

Im Übergangseditor wird die Bedingung nicht mit einem Semikolon abgeschlossen. Dies gilt auch für regulären ESDL-Code, wo Bedingungen in Klammern erscheinen.

Wird die Bedingung mit `true` beurteilt, wird die Leerlaufdrehzahl des Motors auf `n_warm` gestellt.

Beachten Sie, dass innerhalb der Zustandsautomaten-Darstellung dieser Code mit dargestellt wird. Im Beispiel wird mit Hilfe der Kommentarzeichen ein Alias-Name für die Übergangsbedingung erzeugt und im Diagramm dargestellt.

- Wählen Sie auch für die Aktion `<ESDL>` und geben Sie den folgenden Code ein:



- Klicken Sie auf **OK**, um den Übergangseditor zu schließen.
- Schauen Sie sich die Darstellung des Diagramms an. Sie sehen, dass die Bedingung und die Aktion aus der Zustands-Darstellung ersichtlich ist.
- Öffnen Sie einen weiteren Editor für den Übergang von `warmEngine` zu `coldEngine`.
- Wählen Sie `<ESDL>` für die Bedingung und geben Sie den folgenden Code ein:

```
t < t_down
```

Beachten Sie, dass diesmal der komplette Code im Diagramm dargestellt wird, da kein Alias (durch Kommentar) vergeben wurde.

- Wählen Sie auch für die Aktion <ESDL> und geben Sie den folgenden Code ein:

```
n_nominal = n_cold;
```

- Schließen Sie den Editor und wählen Sie **Diagram** → **Store to Cache**.

Anstatt die Aktionen und Bedingungen in ESDL zu spezifizieren, können Sie diese auch im BDE realisieren. Dazu legen Sie als erstes ein eigenes Diagramm für die Aktionen und Bedingungen an.

Diagramm für Aktionen/Bedingungen angelen:

- Wählen Sie im Zustandsautomaten-Editor **Diagram** → **Add Diagram** → **Actions/Conditions BDE**.

  In der Fläche „Diagrams“ wird ein Diagramm namens `ActionCondition_BDE` angelegt.

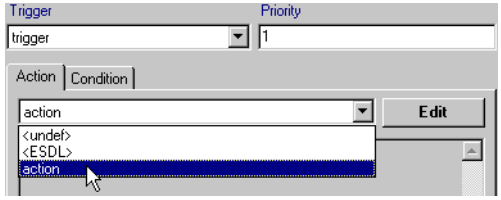
- Übernehmen Sie den Standardnamen.

Jetzt können Sie Bedingungen und Aktionen hinzufügen und spezifizieren.

Aktion/Bedingung als Blockdiagramm spezifizieren:

- Klicken Sie im Feld „Diagrams“ auf das Diagramm `ActionCondition`.
- Legen Sie mit **Diagram** → **Add Action** oder **Diagram** → **Add Condition** neue Aktionen und Bedingungen an.

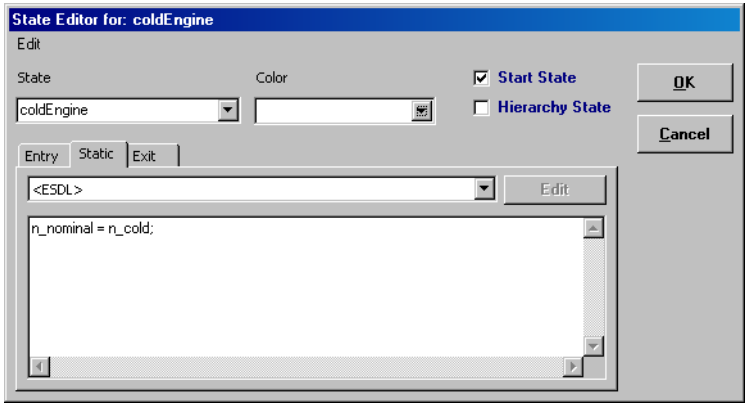
  Diese Aktionen oder Bedingungen können Sie dann im Dialogfenster „Transition Editor“ in den Kombikästchen der Register anwählen. Über die Schaltfläche **Edit** gelangen Sie dann direkt in die grafische Spezifikation im BDE.



Jetzt fehlt noch der Anfangswert für den Ausgang `n_nominal`. Dieser kann nicht wie die Werte der Parameter gesetzt werden; Sie müssen stattdessen eine Aktion für den Startzustand `coldEngine` spezifizieren. Da beim erstmaligen Aufruf des Zustandsautomaten die Eintrittsaktion des Startzustands *nicht* ausgeführt wird, müssen Sie den Anfangswert in der statischen Aktion spezifizieren.

Spezifizieren einer Aktion:

- Klicken Sie mit der rechten Maustaste auf den Startzustand `coldEngine`.
- Wählen Sie im Kontextmenü **Edit State** zum Öffnen des Zustandseditors.



- Wählen Sie im Register „Static“ aus dem Kombifeld `<ESDL>`, um die Eintrittsaktion zu spezifizieren.

  Beachten Sie, dass Sie die voreingestellte Auswahl dieses Kombifelds über das Register „Defaults“ im Optionsfenster des Komponentenmanagers beeinflussen können.
- Geben Sie `n_nominal = n_cold;` in das Codefeld ein, um den Anfangswert von `n_nominal` auf 900 zu setzen.
- Schließen Sie den Zustandseditor durch Klicken auf **OK**.

Damit ist die Spezifikation Ihres Zustandsautomaten abgeschlossen. Ehe Sie damit experimentieren können, müssen Sie verstehen, wie er arbeitet.

6.8.2 Wie ein Zustandsautomat arbeitet

Während aus der grafischen Spezifikation gewöhnlich leicht zu verstehen ist, was eine Standardkomponente macht, ist die Funktion eines Zustandsautomaten u. U. nicht sofort verständlich. In diesem Abschnitt werden die Prinzipien von Zustandsautomaten unter Verwendung des Beispiels aus dem vorhergehenden Abschnitt erläutert. Eine detaillierte Beschreibung der Zustandsautomaten und ihrer Funktionsweise finden Sie im ASCET-Referenzhandbuch, Kapitel 2.5 „Zustandsautomaten".

Jeder Zustand eines Zustandsautomaten verfügt über einen Namen, eine Eintrittsaktion, eine statische Aktion und eine Austrittsaktion. Er hat Übergänge zu und von anderen Zuständen. Jeder Übergang hat eine Priorität, einen Trigger, eine Aktion und eine Bedingung. Alle Aktionen sind optional.

In jedem Zustandsautomaten muss es einen Startzustand geben. Ein Zustandsautomat befindet sich beim ersten Aufrufen im Startzustand. Er prüft dann die Bedingungen aller von diesem wegführenden Übergänge. In unserem Beispiel liegt nur ein solcher Übergang mit der Bedingung `t > t_up` vor. Diese Bedingung untersucht, ob der Eingangswert den Wert des Parameters `t_up` überschreitet. Trifft dies zu, ist die Bedingung wahr und der Übergang findet statt.

Die Parameter `t_up` und `t_down` bestimmen die Temperatur, die der Motor erreichen muss, ehe die Nenndrehzahl geändert werden kann. In unserem Beispiel kann die Drehzahl bei Anstieg der Motortemperatur auf über 70 Grad auf 600 U/min reduziert werden. Fällt die Motortemperatur dann auf unter 60 Grad, muss die Nenndrehzahl wieder auf 900 U/min gestellt werden.

Bei jedem stattfindenden Übergang wird die spezifizierte Übergangsaktion ausgeführt. In unserem Beispiel stellt die Übergangsaktion `n_nominal = n_warm`, die beim Übergang vom Zustand `coldEngine` zu `warmEngine` ausgeführt wird, die Variable `n_nominal` auf 600. Im umgekehrten Fall stellt die Übergangsaktion `n_nominal = n_cold` die Variable auf 900. Wenn ein Übergang stattfindet, führt der Zustandsautomat auch die Austrittsaktion des Zustands, den er verlässt, und die Eintrittsaktion des Zustands, in den er eintritt, aus. In unserem Beispiel sind diese leer, d.h. es geschieht nichts.

Nach Eintritt des Zustandsautomaten in den zweiten Zustand verbleibt er in diesem, bis die Bedingung im Übergang vom zweiten zum ersten Zustand erfüllt ist. Während sich der Zustandsautomat in einem Zustand befindet, wird bei jedem Triggern des Zustandsautomaten die statische Aktion ausgeführt. Das Triggern ist immer ein Ereignis von außen, das *einen* Durchlauf durch den Zustandsautomaten auslöst.

Ein Durchlauf durch einen Zustandsautomaten besteht aus einem ersten Prüfen aller Bedingungen der vom aktuellen Zustand wegführenden Übergänge. Die Übergänge und ihre Bedingungen werden in der Reihenfolge ihrer Prioritä-

ten geprüft. Ist eine Bedingung wahr, wird der entsprechende Übergang durchgeführt, und es werden die Austritts-, die Übergangs- und die Eintrittsaktion ausgeführt. Stellt sich die erste Bedingung als wahr heraus, werden andere vom gleichen Zustand wegführende Übergänge geringerer Priorität nicht geprüft. Ist keine Bedingung wahr, verbleibt der Automat im aktuellen Zustand und führt die statische Aktion einmal pro Durchgang aus.

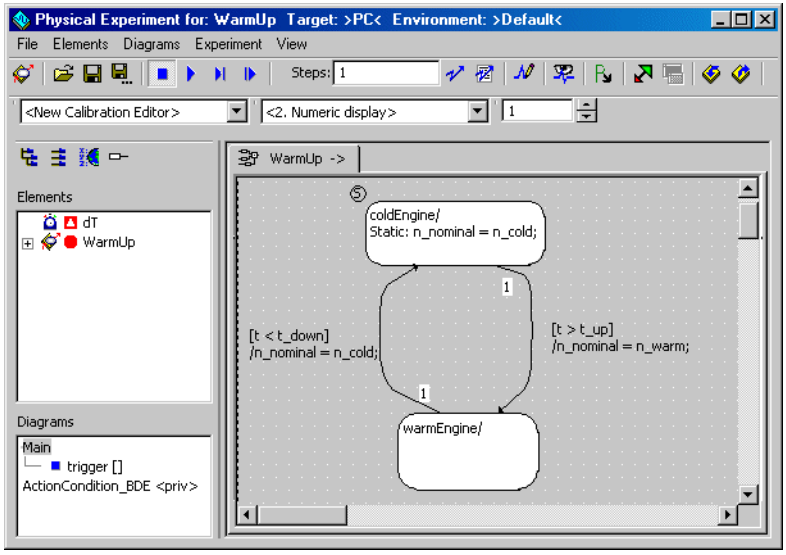
Ist die Bedingung im zweiten Übergang unseres Zustandsautomaten wahr, fällt also der Eingangswert unter die Schwelle ab, geht der Zustandsautomat in den ersten Zustand zurück. Der Automat bleibt dann in diesem Zustand (und unternimmt gar nichts, weil keine statische Aktion stattfindet), bis der Eingangswert wieder die Schwelle überschreitet.

6.8.3 Experimentieren mit dem Zustandsautomaten

Die Experimentierumgebung funktioniert für Zustandsautomaten genauso wie für andere Typen von Komponenten. Ein zusätzliches Merkmal beim Experimentieren mit Zustandsautomaten ist die Animation von Zuständen, d.h. der aktuelle Zustand wird bei laufendem Experiment im Zustandsautomatendiagramm hervorgehoben.

Experimentieren mit einem Zustandsautomaten:

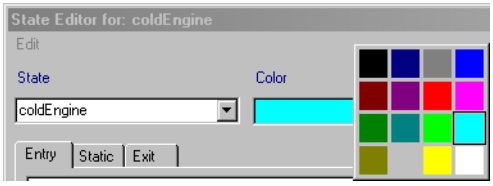
- Wählen sie im Zustandsautomaten-Editor **Component** → **Open Experiment** zum Öffnen der Experimentierumgebung.



- Klicken Sie mit der rechten Maustaste auf einen Zustand und wählen Sie **Animate States** aus dem Kontextmenü.
- Aktivieren Sie das Ereignis `trigger`.
- Erzeugen Sie im Datengenerator einen Kanal für die Variable `t`.
- Weisen Sie dem Kanal eine Sinuswelle mit der Frequenz 1 Hz, dem Offset 70 und der Amplitude 20 zu.
- Öffnen Sie ein Oszilloskopfenster für `t` und `n_nominal`.

- Klicken Sie auf die Schaltfläche **Start Offline Experiment** zum Experimentieren mit dem Zustandsautomaten.
- Verändern Sie die Farben der einzelnen Zustände zur besseren Übersichtlichkeit.

- Dazu verlassen Sie mit der Schaltfläche **Exit to Component** die Experimentierumgebung und rufen den Zustandseditor auf.
- Im Kombikästchen „Color“ des Zustandseditors wählen Sie die Farbe.



- Starten Sie das Experiment erneut.

Der Wert von `n_nominal` ändert sich je nachdem, ob die Sinuswelle die entsprechende Temperaturschwelle über- oder unterschreitet. Sie können die Schwelle unter Verwendung des Verstellsystems ändern und die Auswirkungen unterschiedlicher Werte auf den Ausgang beobachten. Darüber hinaus wird im Zustandsdiagramm der aktuelle Zustand hervorgehoben.

6.8.4 Integrieren des Zustandsautomaten in den Regler

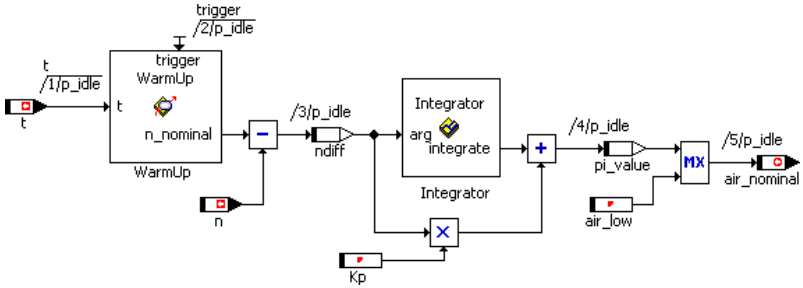
Ein Zustandsautomat lässt sich wie alle anderen Komponenten in ASCET als Baustein innerhalb einer anderen Komponente gleich welchen Typs einsetzen. Sie können nun den Zustandsautomaten in Ihr Reglermodul integrieren und die Drehzahl an die Motortemperatur anpassen.

Integrieren des Zustandsautomaten:

- Öffnen sie aus dem Komponentenmanager das Modul `Lesson4\IdleCon` in einem Blockdiagrammeditor.
- Entfernen Sie den Parameter `n_nominal` aus dem Diagramm und dann auch aus der Liste „Elements“.

  Sie werden den Parameter im Blockdiagramm durch den Zustandsautomaten ersetzen.
- Wählen Sie **Element** → **Add Item** und fügen Sie den Zustandsautomaten zur Liste „Elements“ des Reglers hinzu.
- Erstellen Sie eine Receive-Message und benennen Sie diese mit `t`.
- Verbinden Sie den Ausgang der Komponente `WarmUp` anstelle der gelöschten Variablen mit dem Subtraktionsoperator und verbinden Sie den Eingang von `WarmUp` mit der Receive-Message `t`.

- Passen Sie das Diagramm entsprechend der Darstellung an. Dabei ist darauf zu achten, dass die zeitliche Abfolge im Diagramm so eingestellt wird, dass alle Punkte in der richtigen Reihenfolge enthalten sind.



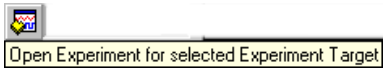
- Speichern Sie das Diagramm ab und klicken Sie im Komponentenmanager auf die Schaltfläche **Save**.

Damit der veränderte Regler mit unserem Projekt arbeiten kann, müssen wir am Projekt einige Anpassungen vornehmen. An dieser Stelle werden wir auch den Temperaturfühler integrieren, der bislang ungenutzt war.

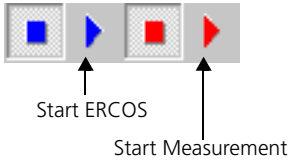
Modifizieren des Projekts:

- Öffnen Sie den Projekteditor für das Projekt `ControllerTest`.

- Wechseln Sie in das Register „OS“.
- Weisen Sie den Prozess `t_sampling` der Task `Task10ms` zu.
- Verwenden Sie den Befehl **Task** → **Move Up,** damit der Prozess `t_sampling` in dieser Task an erster Stelle steht.



- Klicken Sie auf die Schaltfläche **Open Experiment for selected Experiment Target**.
- Öffnen Sie einen zusätzlichen numerischen Editor für den Wert `U_t`.
- Fügen Sie die Variable `t` zum Oszilloskop hinzu.



- Klicken Sie auf die Schaltfläche **Start ERCOS**.

- Klicken Sie auf die Schaltfläche **Start Measurement**.
- Verstellen Sie den Wert `U_t` und beobachten Sie die Auswirkungen.

  Wenn der Wert von `t` die Schwelle von 70 Grad überschreitet, schaltet der Zustandsautomat den Nennwert für `n` auf den unteren Wert von 600 um. Fällt die Temperatur wieder unter 60 Grad ab (wird durch Verstellen von `U_t` simuliert), steigt der Nennwert für `n` wieder auf den ursprünglichen Wert von 900.

6.8.5 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, die folgenden Aufgaben in ASCET auszuführen:

- Erstellen eines Zustandsdiagramms
- Erstellen und Zuweisen von Bedingungen, Aktionen und Triggern
- Experimentieren mit Zustandsautomaten
- Integrieren von Zustandsautomaten in andere Komponenten.

6.9 Hierarchische Zustandsautomaten

Nachdem Sie in der vorherigen Lektion die Arbeitsweise von Zustandsautomaten kennengelernt haben, soll nun ein etwas komplexeres System realisiert werden. Der Schwerpunkt dieser Einheit liegt auf hierarischen Zustandsautomaten. Zusätzlich lernen Sie auch den Umgang mit den von ASCET zur Verfügung gestellten Systembibliotheken bzw. Komponenten wie z.B. Timern.

ASCET ermöglicht die Strukturierung von Zustandsautomaten durch geschlossene und offene Hierarchien. Bei geschlossenen Hierarchien verbirgt sich die interne Funktionalität, bei offenen Hierarchien sind die Subzustände auch grafisch dargestellt.

Es soll eine Ampelsteuerung aufgebaut werden, die über parametrierbare Zeiten die einzelnen Phasen einer Ampel durchläuft. Zusätzlich soll die Ampel einen Fehlerzustand haben, in welchem ein Blinken der Ampel erfolgt.

6.9.1 Spezifizieren des Zustandsautomaten

Zuerst werden Sie die benötigten Bibliotheken importieren und die Vorbereitungen für diese Aufgabe treffen.

Importieren der Systembibliothek `SystemLibETAS.exp`:

- Klicken Sie im Komponentenmanager auf **Import**.

  Das Fenster „Select Import File“ öffnet sich.

- Wählen Sie im Feld „Import File“ mit Hilfe der Schaltfläche aus dem Export-Verzeichnis Ihrer ASCET-Installation (z.B. `C:\etas\ASCET5.2\export`) die Datei `SystemLibETAS.exp`.

  Jetzt ist die Schaltfläche **OK** verfügbar.

- Klicken Sie auf **OK**, um den Import zu starten.

  Das Fenster „Import“ öffnet sich. Alle in der Datei enthaltenen Objekte sind ausgewählt.

- Bestätigen Sie den Import aller Dateien mit **OK**.

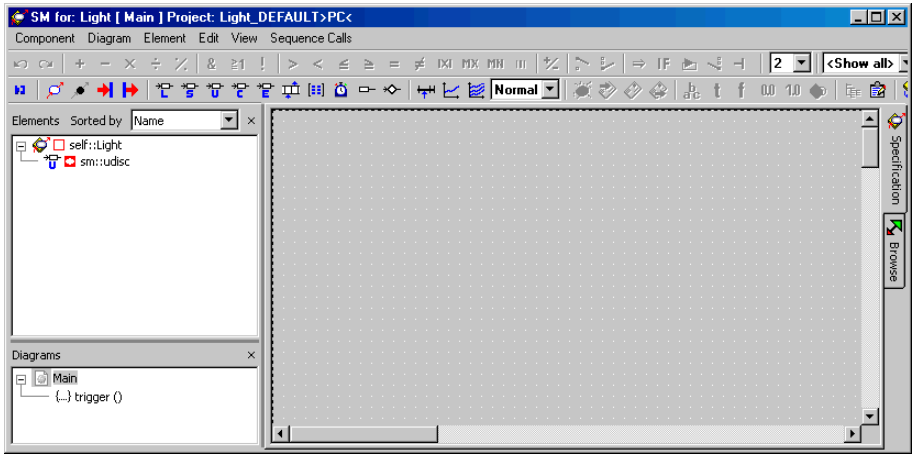
  Die Dateien werden importiert. Dies kann einige Minuten dauern. Nach Abschluss des Importvorgangs werden die importierten Dateien im Fenster „Imported Items“ aufgelistet.

Im zweiten Schritt werden Sie die beiden möglichen Hauptzustände der Ampel spezifizieren (`NormalMode` und `ErrorMode`).

Erstellen des Zustandsautomaten:

- Erstellen Sie im Komponentenmanager den Ordner `Tutorial\Lesson9`.
- Wählen Sie **Insert** → **State Machine** zum Erstellen eines neuen Zustandsautomaten und benennen Sie diesen mit `Light`.

- Wählen Sie **Component** → **Edit Item** zum Öffnen des Zustandsautomaten-Editors.



Sie können mit der Spezifikation des Zustandsautomaten, der Ihre Ampel steuert, beginnen.

- Legen Sie die beiden Zustände `ErrorMode` und `NormalMode` an.

Als nächstes fügen Sie aus der System-Bibliothek einen Timer in Ihr Projekt ein.

Einfügen des Timer-Objekts:

- Wählen Sie **Element** → **Add Item**.
- Wählen Sie im Dialog „Select Item“ aus der Bibliothek `SystemLib_ETAS` im Ordner `Counter_Timer` das Timer-Objekt `Timer`.
- Bestätigen Sie Ihre Auswahl durch **Ok**.

  In der Liste „Elements“ Ihres Zustandsautomaten ist nun ein Objekt `Timer` eingefügt worden.

Spezifizieren des Zustandsdiagramms:

- Spezifizieren Sie die benötigten Datenelemente wie folgt:
  - Einen Eingang `error` vom Typ `Logic`,

  - drei Ausgänge (`yellow`, `green`, `red`) vom Typ `Logic`, die die Ampelfarben symbolisieren,
  - vier kontinuierliche Parameter (`BlinkTime`, `YellowTime`, `GreenTime`, `RedTime`) für die verschiedenen Zeiten der Ampel.

  Damit Sie den Umgang mit abhängigen Parametern noch intensiver üben, werden Sie die Parameter derart konfigurieren, dass nur die Grünphase vorgegeben wird und abhängig davon die übrigen Parameter mit Werten belegt werden:

  ```
  RedTime = 2 * GreenTime
  YellowTime = GreenTime/3
  BlinkTime = YellowTime/10
  ```

- Spezifizieren Sie nun die Berechnungen und Abhängigkeiten der einzelnen Parameter.
- Aktivieren Sie dazu im Elementeditor für die Parameter `RedTime`, `YellowTime` und `BlinkTime` das Kontrollkästchen **Dependent** unter „Dependency“.

  Der Elementeditor wird über Doppelklick auf das Element oder über das Kontextmenü **Edit** gestartet.
- Klicken Sie auf die Schaltfläche **Formula**, um den Formeleditor zu öffnen.
- Geben Sie für jeden der abhängigen Parameter im Formeleditor die Berechnung vor, indem Sie erst einen formalen Parameter `x` anlegen und dann im Formelfeld die Berechnung hinterlegen.

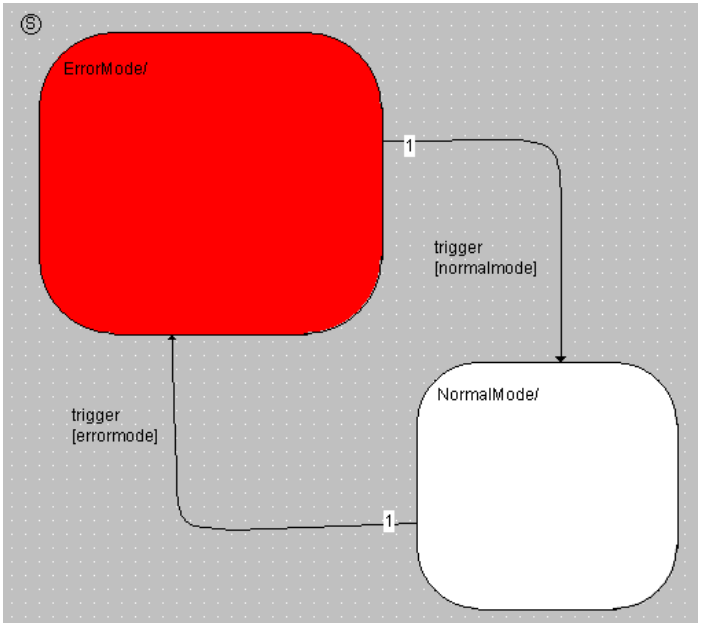
  ```
  Redtime : 2*x
  YellowTime : x/3
  BlinkTime : x/10
  ```

- Schließen Sie den Formeleditor und den Elementeditor.

- Öffnen Sie über das Kontextmenü **Edit Data** das Fenster „Edit Dependency“.
- Weisen Sie nun für jeden der abhängigen Parameter im Fenster „Edit Dependency“ dem formalen Parameter `x` den entsprechenden Modell-Parameter zu.

  ```
  RedTime : x = GreenTime
  YellowTime : x = GreenTime
  BlinkTime : x = YellowTime
  ```
- Belegen Sie die Datenelemente mit sinnvollen Werten vor (z.B. `GreenTime = 5`).
- Öffnen Sie den Zustandseditor für den Zustand `ErrorMode`.

  Den Zustandseditor öffnen Sie entweder über einen Doppelklick auf einen Zustand oder über das Kontextmenü **Edit State**.
- Definieren Sie diesen Zustand als Startzustand und färben Sie ihn rot ein.
- Vergrößern Sie beide Zustände, damit die Hierarchien eingefügt werden können.
- Legen Sie die Übergänge zwischen den beiden Zuständen an.
- Öffnen Sie den Übergangseditor über das Kontextmenü **Edit Transition** oder über einen Doppelklick auf die grafische Repräsentation.
- Spezifizieren Sie die Übergänge zwischen den beiden Zuständen, indem Sie im Übergangseditor die Bedingungen eingeben.
- Geben Sie die Bedingungen in ESDL so ein, dass der Normalzustand `NormalMode` aktiviert wird, wenn am Eingang `error` der Wert `false` anliegt (also kein Fehler aufgetreten ist), und `ErrorMode` aktiviert wird, wenn ein Fehler anliegt.



- Wählen Sie **Diagram** → **Store to Cache.**
- Speichern Sie Ihre Arbeit im Komponentenmanager mit **File** → **Save Database** ab.
- Experimentieren Sie eventuell mit den Hauptzuständen.

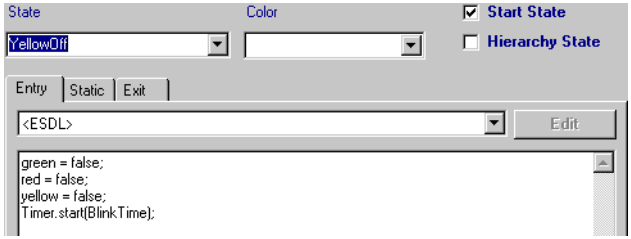
Der nächste Schritt beim Aufbau der Ampelsteuerung ist das Spezifizieren der Subzustände. Zuerst spezifizieren Sie das Verhalten im Fehlerfall `ErrorMode`. In diesem Zustand soll ein gelbes Blinklicht ausgegeben werden. Dazu führen Sie zwei Subzustände `YellowOff` und `YellowOn` ein, zwischen denen timergesteuert hin und her geschaltet wird. Im Zustand `YellowOn` soll der Ausgang `yellow` auf `true` gesetzt werden, der Zustand `YellowOff` setzt diesen zurück auf `false`.

Spezifizieren der Subzustände des Fehlerfalls:

- Erstellen Sie die Zustände `YellowOff` und `YellowOn` und platzieren Sie diese innerhalb des Zustandes `ErrorMode`.
- Definieren Sie `YellowOff` als Startzustand und färben Sie `YellowOn` gelb ein.

- Definieren Sie das Verhalten des Zustands `YellowOff` im Zustandseditor.

  Rufen Sie dazu den Zustandseditor entweder über das Kontextmenü **Edit State** oder durch einen Doppelklick auf den Zustand auf.
- Beschreiben Sie das Verhalten wie folgt:
  Für die Eintrittsaktion wählen Sie im Kombikästchen des Registers „Entry“ `ESDL` und geben den folgenden Code ein:



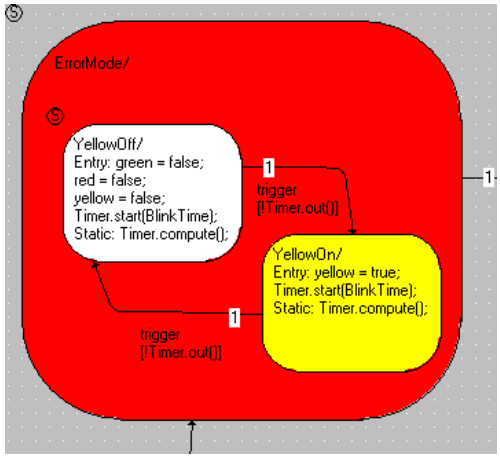
- Für die statische Aktion geben Sie im Register „Static“ folgenden Code ein:

  ```
  Timer.compute();
  ```
- Definieren und beschreiben Sie nun den Zustand `YellowOn` auf dieselbe Weise.

  Eintrittsaktion:
  ```
  yellow = true;
  Timer.start(BlinkTime);
  ```
  Statische Aktion:
  ```
  Timer.compute();
  ```

- Definieren Sie nun die Übergänge zwischen den beiden Subzuständen.

  Die Bedingung für einen Zustandsübergang ist, dass der Timer abgelaufen ist (`Timer.out() == false`).



Im Zustand `ErrorMode` wird somit der Zustand `YellowOff` gestartet. Hierbei wird neben dem Ausschalten aller Farbsignale auch der Timer mit der parametrierbaren Blinkzeit innerhalb der Eintrittsaktion gestartet. Die statische Aktion des Zustands `YellowOff` ruft jedesmal die Timerfunktion `compute()` auf, welche den Timerzähler dekrementiert. Ist dieser Zähler 0, dann liefert die Funktion `out()` des Timers den Rückgabewert `false`, und damit ist die Übergangsbedingung erfüllt. Der Zustand `YellowOn` verhält sich entsprechend; nur wird hier in der Eintrittsaktion das Farbsignal `Yellow` eingeschaltet.

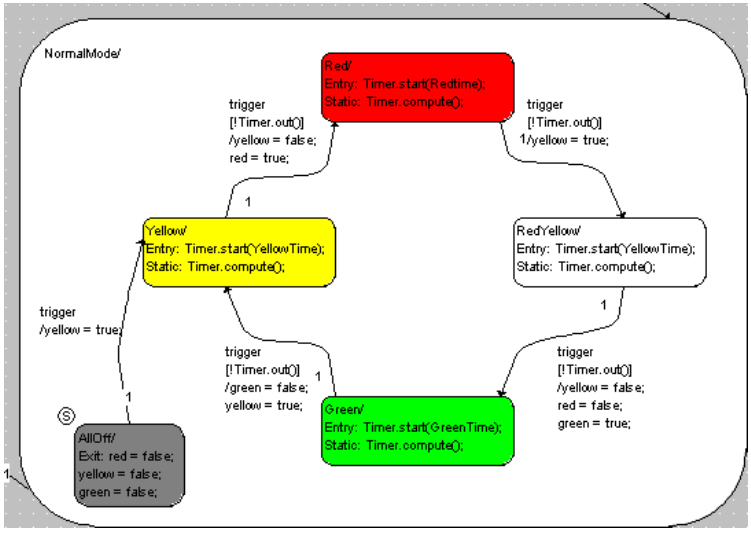
Im nächsten Schritt spezifizieren Sie das Verhalten des Normalbetriebs. Legen Sie hierzu als Startzustand `AllOff` an und platzieren diesen innerhalb des Zustands `NormalMode`. Setzen Sie in der Austrittsaktion alle Farbsignale auf einen definierten Zustand. Überlegen Sie sich jetzt ein geeignetes Verhalten für die Ampelsteuerung.

In diesem Beispiel sollen Sie die Aktivierung bzw. Deaktivierung der einzelnen Farbsignale nicht wie bisher in den Eintrittsaktionen der Zustände, sondern in den Übergangsaktionen beschreiben.

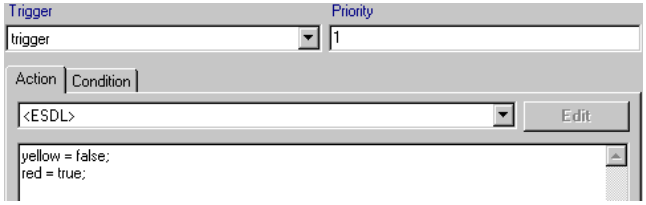
Spezifizieren der Subzustände des Normalbetriebs:

- Erstellen und platzieren Sie die Zustände `AllOff` (Startzustand), `Yellow`, `Red`, `RedYellow` und `Green`.
- Spezifizieren Sie das Verhalten der Zustände, indem Sie für jeden Zustand den entsprechenden Timer für die jeweilige Farbe starten (Eintrittsaktion) und in der statischen Aktion die Timer-Verarbeitung anstoßen (`Timer.compute()`).
- Definieren Sie die Übergänge der Zustände und beschreiben Sie das Verhalten der Zustände innerhalb der Übergangsaktionen.

  Der Übergang von Zustand `AllOff` nach `Yellow` soll generell erfolgen, alle anderen Übergänge sollen nach Ablaufen des jeweiligen Timers erfolgen.



- Geben Sie im Register „Action“ des Übergangseditors die jeweiligen Aktionen für die Farbsignale ein, z.B.



- Schließen Sie den Übergangseditor und wählen Sie **Diagram** → **Store to Cache**.

Damit ist die Spezifikation Ihrer Ampelsteuerung abgeschlossen. Ehe Sie damit experimentieren können, sollten Sie für die Parameter der unterschiedlichen Farb-Timer noch sinnvolle Werte eingeben.

6.9.2 Experimentieren mit dem hierarchischen Zustandsautomaten

Sie können mit hierarchischen Zustandsautomaten auf dieselbe Art und Weise experimentieren wie mit einfachen Zustandsautomaten. Vergessen Sie bitte nicht, die Animation im Experiment zu aktivieren.

Experimentieren mit dem Zustandsautomaten:

- Wählen sie im Zustandsautomaten-Editor **Component** → **Open Experiment** zum Öffnen der Experimentierumgebung**.**
- Klicken Sie mit der rechten Maustaste auf einen Zustand und wählen Sie **Animate States** aus dem Kontextmenü.
- Aktivieren Sie das Ereignis `trigger`.

- Klicken Sie auf die Schaltfläche **Start Offline Experiment** zum Experimentieren mit dem Zustandsautomaten.
- Experimentieren Sie mit dem Zustandsautomaten, indem Sie den Parameter `GreenTime` verändern und anschließend die abhängigen Parameter über **Update Dependent Parameter** aktualisieren.

- Legen Sie gelegentlich den Eingang `error` auf `true`.

6.9.3 Wie hierarchische Zustandsautomaten arbeiten

Hierarchische Zustandsautomaten arbeiten wie normale Zustandsautomaten auch. Im Prinzip stellen hierarchische Zustandsautomaten nur eine grafische Strukturierung des Gesamtverhaltens dar. Überlegen Sie sich oder realisieren Sie als Zusatzaufgabe, wie das beschriebene Verhalten ohne Hierarchien erreicht werden kann.

Das Beispiel Ampelsteuerung ist mit zwei hierarchischen Zuständen strukturiert. Über die logische Eingabevariable `error` wird zwischen den beiden Zuständen `ErrorMode` und `NormalMode` gewechselt. Innerhalb dieser Zustände ist das Sub-Verhalten definiert.

Betrachten Sie zum Verständnis die Abarbeitung im Hierarchiezustand `ErrorMode`. Bei jedem Aufruf des Triggers wird die Bedingung für den Übergang vom Hierarchiezustand `ErrorMode` in den Hierarchiezustand `NormalMode` geprüft (Bedingung: `!error`). Ist kein Übergang notwendig, werden die Übergänge vom Subzustand `YellowOff` nach `YellowOn` bzw. andersherum geprüft und die notwendigen Aktionen ausgeführt.

Wenn Sie nun den Fall `NormalMode` betrachten, dann bedeutet dies, dass bei jedem Triggeraufruf zunächst geprüft wird, ob am Eingang `error` der Wert `true` anliegt und damit ein Übergang in `ErrorMode` notwendig wird. Erst wenn das nicht der Fall ist, werden die Übergänge der Unterzustände (`AllOff`, `Yellow`, `Red`, `RedYellow`, `Green`) geprüft. Im Ampel-Beispiel wird also untersucht, ob ein Timer abgelaufen ist.

Zum Verständnis der Arbeitsweise können Sie sich den aus dem Zustandsdiagramm generierten Code direkt anzeigen.

Darstellung des generierten Codes

- Wählen Sie im Zustandsautomaten-Editor **Component** → **View Generated Code** zur Darstellung des generierten Codes.

  Der Code der Komponente wird in eine temporäre Datei geschrieben und dann mit einer in der Registrierungsdatenbank des Betriebssystems definierten Applikation geöffnet.

Hinweis

Zur Darstellung des generierten Codes wird in der Registrierungsdatenbank des Betriebssystems gesucht, welche Applikation mit Dateien des Typs `*.c` *und* `*.h` *verbunden ist. Abhängig von den registrierten Dateiendungen wird der entsprechende Editor gestartet.*

6.9.4 Zusammenfassung

Nach Abschluss dieser Lektion müssten Sie in der Lage sein, in ASCET die folgenden zusätzlichen Aufgaben auszuführen:

- Erstellen von hierarchischen Zustandsdiagrammen
- Beschreibung des Verhaltens sowohl in den Aktionen der Zustände als auch in den Übergangsaktionen
- Importieren von Modulen, Klassen oder Komponenten
- Importieren von Systemkomponenten aus den ASCET Bibliotheken
- Verwendung der Systemkomponente Timer
- Umgang mit abhängigen Parametern
- Darstellung des generierten Codes

7 Glossar

Im vorliegenden Glossar werden die in der Dokumentation von ASCET verwendeten technischen Begriffe und Abkürzungen erläutert. Viele Begriffe werden zwar auch in einem allgemeineren Sinn verwendet, aber hier wird die speziell auf ASCET zutreffende Bedeutung erläutert.

Die Begriffe sind in alphabethischer Reihenfolge aufgelistet.

7.1 Abkürzungen

ASAM-MCD

Arbeitskreis zur **S**tandardisierung von **A**utomations- und **M**esssystemen, mit den Arbeitsgruppen **M**essen, **C**alibrieren und **D**iagnose

ASCET

Entwicklungswerkzeug für Steuergerätesoftware

ASCET-MD

ASCET **M**odeling & **D**esign

ASCET-RP

ASCET **R**apid **P**rototyping

ASCET-SE

ASCET **S**oftware **E**ngineering

BDE

Block**d**iagramm**e**ditor

CPU

Central **P**rocessing **U**nit

ECU

Embedded **C**ontrol **U**nit (Steuergerät)

ERCOS^{EK}

OSEK-konformes Echtzeit-Betriebssystem von ETAS

ESDL

Embedded **S**oftware **D**escription **L**anguage

ETK

Emulator**t**ast**k**opf

FPU

Floating **P**oint **U**nit

HTML

Hyper**t**ext **M**arkup **L**anguage

INCA

Mess-, Applikations- und Diagnosessystem (**In**tegrated **C**alibration and **A**cquisition Systems)

INTECRIO

Eine neue ETAS-Produktfamilie. INTECRIO integriert Code aus verschiedenen Modellierungswerkzeugen in ein komplettes Rapid Prototyping-System, ermöglicht alle nötigen Konfigurationen, erlaubt die Erzeugung von ausführbarem Code und bietet eine Experimentierumgebung für die Durchführung des Rapid Prototyping-Experiments.

OS

Betriebssystem (**o**perating **s**ystem)

OSEK

Arbeitskreis **O**ffene **S**ysteme für die **E**lektronik im **K**raftfahrzeug

RAM

Random **A**ccess **M**emory (Schreib-/Lesespeicher)

ROM

Read-**O**nly **M**emory (Nur-Lesespeicher)

SG

Steuer**g**erät

UML

Unified **M**odeling **L**anguage

XML

E**x**tensible **M**arkup **L**anguage

7.2 Begriffe

Aktion

Eine Aktion ist Teil eines Zustandsautomaten und steht mit Zuständen oder Übergängen des Zustandsautomaten im Zusammenhang. Eine Aktion ist ein Teil der Funktionalität, dessen Ausführung durch den Zustandsautomaten ausgelöst wird.

Anwenderprofil

Satz von anwendereigenen Optionseinstellungen.

Application Mode

siehe Betriebsmodus

Argument

Ein Argument ist die Eingabe für eine Methode einer Klasse. Argumente können nur bei der Spezifikation jener Methode benutzt werden, zu der sie gehören, und nicht bei anderen Methoden der Klasse.

Arithmetische Dienste

Vom Benutzer bereitgestellte C-Funktionen, die verwendet werden, um elementare Operationen wie z.B. Additionen zu optimieren und mit speziellen Eigenschaften wie Wertebegrenzungen oder Überlaufsbehandlungen zu erweitern.

Array

Ein Array ist eine eindimensionale statische Liste von Elementen mit einem der skalaren Basistypen `continuous` oder `discrete`, indiziert durch den skalaren Basistypen `discrete`.

Art

Es gibt drei Arten (engl. *kinds*) von Elementen: Variable, Parameter und Konstanten. Variable dürfen gelesen und geschrieben werden. Parameter dürfen nur gelesen, aber während des Experimentierens kalibriert werden. Konstanten dürfen während des Experimentierens nur gelesen werden.

ASAM-MCD-2MC-Datei

Standard-Austauschformat für Programmbeschreibungen im ASCII-Format zur Beschreibung von Mess- und Verstellgrößen. Die Datei hat die Endung `*.a2l`.

Basismodelltypen

Basismodelltypen werden zum Modellieren physischen Verhaltens verwendet. Es gibt drei Typen: `continuous`, `discrete` und `logical`. Für sie sind Operationen wie z.B. Addition oder Vergleich definiert. Die Implementierung dient zum Umwandeln der Modelltypen in Implementierungstypen.

Bedingung

Eine Bedingung wird zur Beschreibung des Regelflusses in einem Zustandsautomaten verwendet. Sie liefert einen logischen Wert zurück, der bestimmt, ob ein Übergang von einem Zustand in einen anderen stattfindet.

Beschreibungsdatei

Enthält die physikalische Beschreibung der Kenn- und Messgrößen im Steuergerät (Namen, Adressen, Umrechnungsformeln, Funktionszuordnungen usw.).

Betriebsmodus

Ein Betriebsmodus ist Teil des Betriebssystems von ASCET; er beschreibt verschiedene mögliche Zustände eines Systems wie z.B. den Betriebsmodus EEPROM-Programmiermodus, Anlaufen oder Normalbetrieb.

Betriebssystem

Das Betriebssystem steuert den zeitlichen Ablauf der Ausführung/Aktivierung eines ASCET-Softwaresystems. Das Betriebssystem leistet auch Dienste für die Kommunikation (Messages) und den Zugang zu reservierten Teilen der Hardware (Ressourcen). Das Betriebssystem von ASCET basiert auf dem Echtzeit-Betriebssystem ERCOS^{EK}.

Blockdiagramm

Ein Blockdiagramm ist eine grafische Beschreibung für eine Komponente, in der die verschiedenen Elemente, Operatoren und Eingänge/Argumente sowie Ausgänge/Ergebniswerte durch Richtungslinien miteinander verbunden sind. Ein Blockdiagramm besteht aus mehreren Diagrammen. Die Beschreibung als Blockdiagramm ist eine physikalische Beschreibung im Gegensatz zur Darstellung mittels C-Code.

Bypass-Experiment

In einem Bypass-Experiment ist ASCET direkt mit einem Mikrocontroller verbunden, und Teile der Mikrocontroller-Software werden durch ASCET simuliert.

C-Code

C-Code ist eine implementierungsabhängige Beschreibung einer Komponente.

Code

Der ausführbare Code ist das „eigentliche" Programm, mit Ausnahme der Daten (enthält die eigentlichen Algorithmen). Der Code ist der durch die CPU ausführbare Teil des Programms.

Codegenerierung

Codegenerierung ist der erste Schritt bei der Umwandlung eines physischen Modells in einen ausführbaren Code. Das physikalische Modell wird in ANSI C-Code umgesetzt. Da der C-Code compilerabhängig (und damit targetabhängig) ist, wird für jedes Target ein unterschiedlicher Code erzeugt.

Container

Container dienen als Behälter für Projekte, Klassen und Module. Ihr Zweck ist es, Modelle und Datenbanken zu strukturieren und verschiedene Datenbankobjekte unter eine gemeinsame Versionskontrolle zu stellen.

Data-Logger

Mit einem Data-Logger lassen sich Messdaten aus einem Experiment erfassen und zur weiteren Untersuchung auf der Festplatte abspeichern.

Daten

Die Daten sind die verstellbaren Größen eines Programmes.

Datenbank

Alle vorgegebenen oder mit ASCET erzeugten Informationen werden in einer Datenbank abgelegt. Eine Datenbank besteht aus Ordnern.

Datengenerator

Der Datengenerator ist Teil der Experimentierumgebung. Er stimuliert Eingabeparameter oder Variablen des Modells während des Experimentierens.

Datensatz

Ein Datensatz enthält bzw. referenziert die Ausgangsdaten für alle Elemente einer Komponente oder eines Projekts.

Diagramm

Ein Diagramm wird für die grafische Darstellung von Komponenten als Blockdiagramme oder Zustandsautomaten verwendet.

Dimension

Die Dimension beschreibt die Größe von Grundelementen. Die Dimension kann ein Skalar (dimensionslos), ein Array (eindimensional) oder eine Kennlinie/-feld sein.

Distribution

Siehe Stützstellenverteilung.

Editor

Siehe Verstellfenster.

Element

Ein Element ist ein Teil einer Komponente, der Daten wie z. B. eine Variable, einen Parameter oder eine andere, eingesetzte Komponente liest oder schreibt.

Ereignis

Ein Ereignis ist ein (externer) Trigger, der eine Aktion des Betriebssystems wie z. B. eine Task auslöst.

Ereignisgenerator

Der Ereignisgenerator ist Teil der Experimentierumgebung. Er wird zur Beschreibung der Reihenfolge und des zeitlichen Ablaufs der Generierung von Ereignissen für die Aktivierung von Tasks (Methoden/Prozessen) bei einem Offline-Experiment eingesetzt.

Experiment

Ein Experiment definiert die Einstellungen in der Experimentierumgebung, die zum Prüfen der ordnungsgemäßen Funktion von Komponenten oder Projekten verwendet werden. Es enthält Angaben zu Größe, Anordnung und Inhalt der Mess- und Verstellfenster sowie die Einstellungen des Ereignisgenerators, des Datengenerators und des Data-Loggers. Ein Experiment kann entweder offline (nicht in Echtzeit) oder online (in Echtzeit) ablaufen und kann zur Steuerung eines technischen Prozesses in einer Bypass- oder einer Fullpass-Anwendung zum Einsatz kommen. In allen Fällen wird aus einer ASCET-Spezifikation generierter instrumentierter Code zur Ausführung des Experiments verwendet.

Experimentierumgebung

Hauptarbeitsumgebung, in der der Anwender seine Experimente durchführt.

Fensterelemente

Oberbegriff für Verstell- und Anzeige-Elemente.

Festkomma-Code

Aus der physikalischen Aufgabenstellung lässt sich ein Festkomma-Code generieren, der auf Prozessoren ohne Gleitpunkteinheit ausgeführt werden kann.

Formel

Eine Formel ist Teil einer Implementierung, die eine Umwandlung von Modelltypen zu Implementierungs- (Daten-)Typen beschreibt.

Fullpass-Experiment

In einem Fullpass-Experiment ist ASCET direkt mit einem experimentellen Mikrocontroller verbunden, und die gesamte Anwendung wird durch ASCET simuliert.

Geltungsbereich

Ein Element hat immer einen von zwei Geltungsbereichen: lokal (nur innerhalb einer Komponente sichtbar) oder global (im Rahmen eines Projekts definiert).

Größe

Oberbegriff für Verstellgrößen (Kenngrößen) und Messgrößen.

Gruppenkennlinien/-felder

Gruppenkennlinien/-felder sind Kennlinien/-felder, die die gleiche Verteilung von Achsenpunkten, jedoch verschiedene Anzeigewerte haben. Die Verteilungen der Achsenpunkte (Distribution oder Stützstellenverteilung) und der einzelnen Gruppenkennlinien/felder werden als gesonderte Elemente festgelegt.

HEX-Datei

Austauschformat eines Programmstandes in den Ausprägungen Intel-Hex- oder Motorola-S-Record-Datei.

Hierarchie

Ein Hierarchieblock wird zum Strukturieren der grafischen Spezifikation eines Blockdiagramms verwendet.

Implementation-Cast

Element, das dem Anwender die Möglichkeit gibt, Einfluss auf die Implementierung von Zwischenergebnissen in arithmetischen Rechenketten zu nehmen, ohne die physikalischen Repräsentationen der betroffenen Elemente zu beeinflussen.

Implementierung

Eine Implementierung beschreibt die Umsetzung der physikalischen Aufgabenstellung (des Modells) in ausführbarem Festkomma-Code. Eine Implementierung besteht aus einer (linearen) Umsetzungsformel und einem Begrenzungsintervall für die Modellwerte.

Implementierungsdatentypen

Implementierungsdatentypen sind die Datentypen der unterliegenden Programmiersprache C, z. B. `unsigned byte` (`uint8`), `signed word` (`sint16`), `float`.

Implementierungstypen

Templates für Implementierungen. Implementierungstypen beinhalten die wesentlichen Spezifikationen einer Implementierung; sie werden im Projekteditor definiert und können in den Implementierungseditoren einzelner Elemente zugewiesen werden.

Intel-Hex

Austauschformat für Programmstände

Kennfeld

3-dimensionale Verstellgröße

Kenngröße

Oberbegriff für Kennfeld, Kennlinie und Kennwert (siehe auch Parameter)

Kennlinie

2-dimensionale Verstellgröße

Kennwert

1-dimensionale Verstellgröße (Parameter)

Klasse

Eine Klasse ist ein Komponententyp in ASCET. Klassen in ASCET sind vergleichbar mit objektorientierte Klassen. Die Funktionalität einer Klasse wird durch Methoden beschrieben.

Komponente

Eine Komponente ist die Grundeinheit einer wiederverwendbaren Funktionalität in ASCET. Komponenten können als Klassen, Module oder Zustandsautomaten spezifiziert werden. Jede Komponente besteht aus mit Operatoren verknüpften Elementen, wodurch ihre Funktionalität gegeben ist.

Komponentenmanager

Arbeitsumgebung, in der der Anwender ASCET einrichten und die von ihm erzeugten und in der Datenbank gespeicherten Daten verwalten kann.

Konfigurationsdialog

Dialog, in dem die Konfiguration der einzelnen Mess- und Verstellfenster und der darin enthaltenen Größen erfolgt.

Konstante

Eine Konstante ist ein Element, das während der Ausführung eines ASCET-Modells nicht verändert werden kann.

L1

Das Message-Format für den Austausch von Daten zwischen dem Hostrechner und dem Zielrechner, auf dem das Experiment durchgeführt wird. Daten werden beispielsweise zur Anzeige von Werten in Messfenstern übertragen.

Layout

Eine Komponente verfügt über eine grafische Darstellung, in der Pins für Eingaben/Argumente, Ausgaben/Ergebniswerte und Methoden/Prozesse stehen. Darüber hinaus enthält das Layout ein Symbol, das die Komponente bei Einsatz in anderen Komponenten grafisch darstellt.

Literal

Ein Literal wird in der Beschreibung von Komponenten benutzt. Ein Literal enthält eine Zeichenkette, die als Wert interpretiert wird, z. B. als kontinuierlicher oder logischer Wert.

Message

Eine Message ist ein Echtzeitsprachkonstrukt von ASCET für den geschützten Datenaustausch zwischen gleichzeitig ablaufenden Prozessen.

Messen

Erfassen von Daten, die entweder angezeigt oder abgespeichert oder sowohl angezeigt als auch abgespeichert werden.

Messfenster

ASCET-Arbeitsfenster, in dem Messgrößen während einer Messung angezeigt werden.

Messgröße

Bezeichnung einer zu messenden Größe.

Messkanalparameter

Parameter, die für die einzelnen Kanäle eines Messmoduls eingestellt werden können.

Messung

Eine Messung ist die Darstellung von Werten (physikalisch/Implementierung) von Variablen/Parametern während eines Experiments. Die Werte lassen sich mit mehreren verschiedenen Messfenstern wie z.B. Oszilloskopen, numerischen Darstellungen usw. anzeigen.

Methode

Eine Methode ist Teil der Beschreibung der Funktionalität einer Klasse unter dem Aspekt der objektorientierten Programmierung. Eine Methode verfügt über Argumente und einen Rückgabewert.

Modelltyp

Jedes Element einer Komponentenspezifikation von ASCET ist entweder selbst eine Komponente oder ein Modelltyp. Im Gegensatz zu Implementierungstypen stellen Modelltypen physikalische Größen dar.

Modul

Ein Modul ist ein Komponententyp in ASCET. Er beschreibt Prozesse, die vom Betriebssystem aktiviert werden können. Ein Modul kann nicht als Teilkomponente in anderen Komponenten eingesetzt werden.

Monitor

Mit einem Monitor lässt sich der Datenwert eines Elements während eines Experiments in einem Diagramm darstellen.

Motorola-S-Record

Austauschformat für Programmstände

Ordner

Ein Ordner ist eine Verwaltungseinheit zum Strukturieren einer ASCET-Datenbank. Ein Ordner enthält Elemente beliebiger Typen.

Offline-Experiment

In einem Offline-Experiment kann der durch ASCET generierte Code auf dem PC oder auf einem experimentellen Target laufen, jedoch läuft er dabei nicht in Echtzeit. Schwerpunkt ist die Erprobung der Funktionsspezifikation.

Online-Experiment

In einem Online-Experiment lässt sich ein Projekt unter realistischeren Bedingungen erproben. Der Code läuft dabei auf einem experimentellen Target in Echtzeit; Schwerpunkt ist das Echtzeitverhalten des Steuersystems.

Oszilloskop

Ein Oszilloskop ist ein Typ von Messfenster, in dem im Verlauf von Experimenten Datenwerte grafisch dargestellt werden.

Parameter

Ein Parameter (Kennwert, -linie, -feld) ist ein Element, dessen Wert sich durch in einem ASCET-Modell vorgenommene Berechnungen nicht verändern lässt. Er kann jedoch im Verlauf eines Experiments verstellt werden.

Priorität

Jede Task verfügt über eine Priorität in Form einer Zahl. Je höher die Zahl, desto höher ist die Priorität. Durch die Priorität wird die Reihenfolge festgelegt, in der die Tasks ablaufen sollen.

Programm

Ein Programm besteht aus Code und Daten und wird als Einheit durch die CPU des Steuergerätes ausgeführt.

Projekt

Ein Projekt beschreibt ein gesamtes eingebettetes Softwaressystem. Es enthält Komponenten, die die Funktionalität definieren, eine Betriebssystem-Spezifikation sowie einen Bindemechanismus, der die Kommunikation festlegt.

Prozess

Ein Prozess ist eine zeitgleich ausführbare Funktionalität, die durch das Betriebssystem aktiviert wird. Prozesse werden in Modulen spezifiziert und verfügen über keinerlei Argumente/Eingaben oder Ergebniswerte/Ausgaben.

Ressource

Eine Ressource wird zum Modellieren von Teilen eines eingebetteten Systems verwendet, die nur sich gegenseitig ausschließend eingesetzt werden können, wie z. B. Zeitgeber. Soll auf ein solches Gerät zugegriffen werden, muss es zunächst reserviert werden; nach Beendigung seiner Tätigkeit wird es wieder freigegeben. Dies geschieht mit Hilfe von Ressourcen.

Scheduling

Das Scheduling ist das Zuweisen von Prozessen zu Tasks und die Aktivierung von Tasks durch das Betriebssystem.

Schnittstelle

Eine Schnittstelle einer Komponente beschreibt, wie der Datenaustausch mit anderen Komponenten erfolgt. Sie ist mit der `.h`-Datei in C vergleichbar.

Stützstellenverteilung

Eine Stützstellenverteilung (auch Distribution) enthält die Stützstellen für eine oder mehrere Gruppenkennlinien/-felder.

Symbol

Symbole können zur Veranschaulichung der Funktion von ASCET-Komponenten eingesetzt werden.

Target

Ein Target ist die Hardware, auf der ein Experiment läuft. Ein Target kann entweder ein experimentelles Target (PC, Transputer, PowerPC) oder ein Mikrocontroller-Target sein.

Task

Eine Task ist eine geordnete Sammlung von Prozessen, die durch das Betriebssystem aktiviert werden können. Attribute einer Task sind ihre Betriebsarten, ihr Aktivierungstrigger, ihre Priorität und der Modus ihres Schedulings. Bei Aktivierung werden die Prozesse der Task in der angegebenen Reihenfolge ausgeführt.

Trigger

Ein Trigger aktiviert die Ausführung einer Task (im Rahmen des Betriebssystems) oder eines Zustandsautomaten.

Typ

Variablen und Parameter sind vom Typ `cont` (continuous), `udisc` (unsigned discrete), `sdisc` (signed discrete) oder `log` (logic). `Cont` wird für physikalische Größen benutzt, die beliebige Werte annehmen können; `udisc` wird für positive Integerwerte verwendet, `sdisc` für negative; `log` wird für Boolesche Werte (`true` oder `false`) verwendet.

Übergang

Ein Übergang ist eine Verbindung zwischen Zuständen. Übergänge beschreiben mögliche Zustandsänderungen. Jeder Übergang ist einem Trigger des Zustandsautomaten zugeordnet, hat eine Priorität, eine Bedingung und eine Aktion.

Variable

Eine Variable ist ein Element, das während der Ausführung eines ASCET-Modells gelesen oder geschrieben werden kann. Der Wert einer Variablen lässt sich auch mit dem Verstellsystem ändern.

Verstellen

Verstellen ist das Manipulieren der Werte (physikalisch/Implementierung) von Elementen während der Ausführung eines ASCET-Modells (Experiment).

Verstellfenster

Arbeitsfenster, in dem Verstellgrößen verändert werden können.

Zustand

Ein Zustand ist ein Teil eines Zustandsautomaten. Ein Zustandsautomat befindet sich immer in einem seiner Zustände. Einer der Zustände ist als Anfangszustand gekennzeichnet, der den Ausgangszustand des Zustandsautomaten darstellt. Jeder Zustand ist durch Bögen mit anderen Zuständen verbunden. Ein Zustand verfügt über eine Eintrittsaktion (die nach Eintreten eines Zustands ausgeführt wird), eine statische Aktion (die ausgeführt wird, während der Zustand unverändert bleibt) sowie eine Austrittsaktion (die nach Verlassen des Zustands ausgeführt wird).

Zustandsautomat

Ein Zustandsautomat ist ein Komponententyp in ASCET. Das Verhalten wird durch einen Zustandsgrafen beschrieben, der aus durch Übergänge verbundenen Zuständen besteht.

8 Referenzlisten

Im Kapitel „Referenzlisten“ erhalten Sie Informationen zur Fehlerbehandlung, Angaben zur Verzeichnisstruktur und der benötigten Referenzdateien. Desweiteren finden Sie hier eine Liste sämtlicher Tastaturbefehle, die nach Arbeitsfenstern sortiert ist.

8.1 Fehlerbehandlung und Anwenderrückmeldung

Bei der Entwicklung von ASCET wurde größter Wert auf die Funktionssicherheit des Programmes gelegt. Sollte trotzdem ein Fehler auftreten, leiten Sie bitte folgende Informationen an uns weiter:

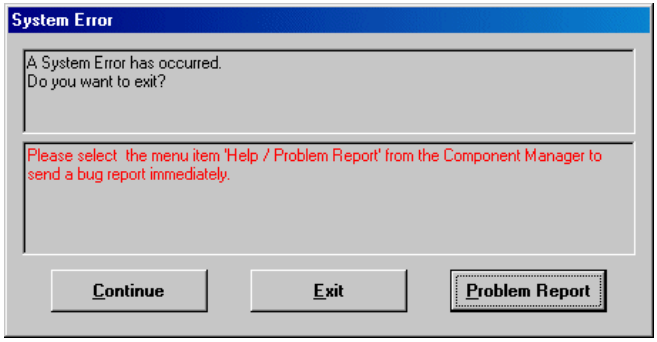
- Welchen Arbeitsschritt wollten Sie mit ASCET durchführen, als der Fehler auftrat?
- Welcher Fehler trat auf (Fehlfunktion, Systemfehler oder Systemabsturz)?
- Welches Modellelement oder Modell wurde zum Fehlerzeitpunkt bearbeitet?

Hinweis

Wichtig für die Pflege und Weiterentwicklung von ASCET ist, dass Sie alle Fehler, die bei einer Applikation aufgetreten sind, an ETAS weiterleiten. Verwenden Sie hierzu die Support-Funktion „Problem Report“.

Wenn Sie die Supportfunktion nutzen, packt ASCET den gesamten Inhalt des „log“-Verzeichnisses (alle `*.log`-Dateien) zusammen mit einer textlichen Beschreibung in ein Archiv und speichert es unter `...\ETAS\LogFiles` als Datei `EtasLogFiles00.zip`. Bei weiteren Archivdateien wird der Dateiname automatisch hochgezählt (bis max. `19`), so dass alte Archivdateien nicht sofort überschrieben werden.

Tritt ein kritischer Systemfehler auf, erscheint folgendes Fenster:



Verhalten im Fehlerfall:

1. Schaltfläche **Problem Report**
   - Klicken Sie auf die Schaltfläche **Problem Report**.

     Die Supportfunktion wird gestartet.
   - Beschreiben Sie den Fehler und leiten Sie die Information – zusammen mit dem Modell – an ETAS weiter.
2. Schaltfläche **Exit**
   - Klicken Sie auf die Schaltfläche **Exit**.

     ASCET wird beendet; alle nichtgespeicherten Änderungen gehen verloren.

     Eventuelle Abfragen, die Sie zum Speichern von Daten aufrufen, schließen Sie ohne zu speichern.
   - Starten Sie ASCET neu.
3. Schaltfläche **Continue**

Hinweis

*Verwenden Sie **Continue** nur, wenn Sie wichtige Konfigurationsdaten speichern müssen. Folgefehler bzw. falsche Konfigurationen sind nicht ausgeschlossen!*

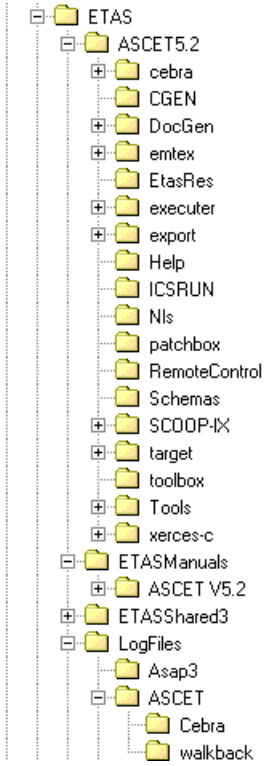
- Klicken Sie auf die Schaltfläche **Continue**.

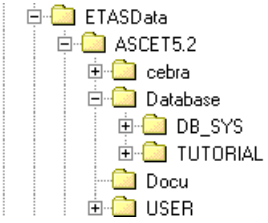
  Die Anwendung wird fortgesetzt; es erfolgt der Rücksprung ins Programm an die Stelle, bevor der Fehler aufgetreten ist.
- Sichern Sie die Daten.
- Beenden Sie ASCET.
- Starten Sie ASCET neu.

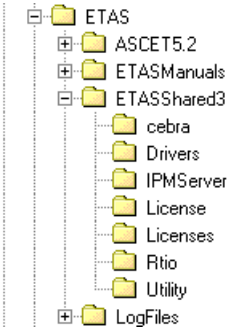
Grundsätzlich ist es ratsam, das Programm zu beenden (ohne Speichern) und neu zu starten. Dadurch wird das Risiko von eventuell auftretenden Folgefehlern ausgeschlossen.

8.2 ASCET-Verzeichnisse

Bei der Installation von ASCET (unveränderte Pfadeinstellungen) wird auf dem Installationslaufwerk diese Verzeichnisstruktur angelegt:







8.2.1 Standard-Speicherverzeichnisse

- Datenbank
  `ETASData\Ascet5.2\Database`
- Export
  `ETAS\Ascet5.2\Export`
- automatisch generierte Dokumentation
  `ETASData\Ascet5.2\Docu`

8.2.2 Standardverzeichnisse ändern

Über das Dialogfenster „Options“ können die Standard-Speicherverzeichnisse verändert werden. Gehen Sie dazu wie folgt vor:

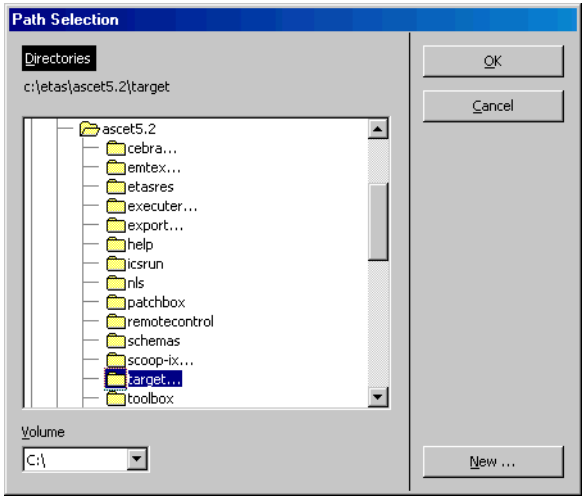
Standard-Speicherverzeichnisse ändern:

- Wählen Sie im Komponentenmanager den Menübefehl **Tools** → **Options**.

  Das ASCET-Optionsfenster öffnet sich. Im Knoten „Options“ können Sie den Datenbankpfad ändern, im Knoten „Export“ das Exportverzeichnis und im Knoten „Documentation“ das Dokumentationsverzeichnis.

- Klicken auf die Schaltfläche neben dem Pfad, den Sie ändern wollen.

  Das Fenster „Path Selection“ öffnet sich.



- Bestimmen Sie das Verzeichnis, das Sie für die gewählte Option als Voreinstellung festlegen wollen.
- Klicken Sie auf **OK**.

  Das gewählte Verzeichnis wird im Fenster „Options“ angezeigt.
- Wiederholen Sie die Schritte für jede Option, die Sie ändern wollen.
- Klicken Sie abschließend im Fenster „Options“ auf **OK**, um die Änderungen zu übernehmen, oder auf **Cancel**, um sie zu verwerfen.

8.3 Bedienung über Tastatur

8.3.1 Allgemeine Bedienfunktionen

Taste	Funktion
ALT + TAB	Ermöglicht das Wechseln zwischen geöffneten Anwendungsprogrammen
ALT + SPACE	Öffnet das Systemmenü für Anwendungsprogrammfenster
ALT + F4	Schließt das aktuelle Fenster
Cursor-Tasten (← ↑ → ↓)	Tabellenelement oder Listenelement anfahren, ↓ auch aktives Listenfeld öffnen
TAB	Setzt die Markierung (Fokus) auf das nächste Element (Option) eines Fensters (SHIFT + TAB andere Richtung)
DEL	Löschen eines markierten Eintrages
SPACE	Aktiviert in Tabellen den Eingabemodus, Tabellen- oder Listenelement wird selektiert bzw. deselektiert.
ESC	Schließt den Eingabemodus ohne Übernahme der Eingabe.
SHIFT	Aktiviert die Mehrfachauswahl, bei gedrückter Shift-Taste kann mit den Cursor-Tasten ein Tabellenbereich gewählt werden.
ENTER	Bestätigt die Eingabe und schließt den Eingabemodus, öffnet bzw. schließt Verzweigungen.
CTRL + A	Auswählen aller Objekte (z.B. in Liste)
CTRL + C	Kopieren in Zwischenablage (nicht für Messfenster)
CTRL + V	Einfügen aus Zwischenablage
CTRL + X	Ausschneiden und einfügen in Zwischenablage

Taste	Funktion
CTRL + Y	Letzte Aktion wird erneut ausgeführt (für Verstellfenster gilt hier aber CTRL + D)
CTRL + Z	Letzte Aktion wird rückgängig gemacht (für Verstellfenster gilt hier aber CTRL + U)
CTRL + F1	Anzeigen der wichtigsten Tastaturbefehle
F10	Hauptmenü aktivieren

Tab. 8-1 Allgemeingültige Tastaturbefehle (im Einzelfall kann ein Tastaturbefehl in einem speziellen Fenster eine andere Bedeutung haben).

8.3.2 Tastaturbefehle im Komponentenmanager

Im Komponentenmanager stehen folgende spezielle Tastaturbefehle zur Verfügung:

Taste	Funktion
CTRL + N	neue Datenbank erzeugen
CTRL + O	Datenbank öffnen
CTRL + S	Datenbank sichern
CTRL + E	Export-Funktion aktivieren
CTRL + M	Import-Funktion aktivieren
ALT + F4	Komponentenmanager schließen und ASCET beenden
F2	Markiertes Objekt umbenennen
CTRL + F	Sucht einer Zeichenkette in C-Code- oder ESDL-Komponenten.
CTRL + H	Ersetzt einer Zeichenkette in C-Code- oder ESDL-Komponenten.
CTRL + Q	Durchsucht die Datenbank unter verschiedenen Gesichtspunkten.
F5	Anzeige im Komponentenmanager aktualisieren
INSERT	Ordner einfügen / Objekt in Container einfügen
RETURN	Editor für den markierten Datenbankeintrag starten
ALT + 1	in das Feld „1 Database“ wechseln
ALT + 2	in das Feld „2 Comment“ wechseln
ALT + 3	in das Feld „3 Contents“ wechseln
ALT + F6	in das nächste Fenster wechseln
ESC	Ausschneiden (<CTRL> + <X>) abbrechen

8.3.3 Tastaturbefehle im Monitorfenster

Im ASCET Monitorfenster stehen folgende spezielle Tastaturbefehle zur Verfügung:

Taste	Funktion
CTRL + O	Log-Datei für das Register „Monitor“ öffnen
CTRL + S	Inhalt des Registers „Monitor“ in Datei sichern
CTRL + F	Text im Register „Monitor“ suchen/ersetzen
CTRL + R	Text im Register „Monitor“ löschen
CTRL + +	Monitorfenster vergrößern
CTRL + -	Monitorfenster verkleinern

8.3.4 Tastaturbefehle in den Editoren

Im Blockdiagrammeditor stehen folgende spezielle Tastaturbefehle zur Verfügung:

Taste	Funktion
F2	Markiertes Element umbenennen
CTRL + PFEIL RECHTS	nächsten Sequenzaufruf anzeigen
CTRL + PFEIL LINKS	vorigen Sequenzaufruf anzeigen

Im C-Code- und ESDL-Editor stehen folgende spezielle Tastaturbefehle zur Verfügung:

Taste	Funktion
CTRL + F	Suchen/Ersetzen
CTRL + S	Methode/Prozess speichern

Im AS-Editor stehen folgende spezielle Tastaturbefehle zur Verfügung:

Taste	Funktion
CTRL + N	neue Datei erzeugen
CTRL + O	Datei öffnen
CTRL + S	Datei sichern

Im Daten- und Implementierungseditor für Komponenten/Projekte stehen folgende spezielle Tastaturbefehle zur Verfügung:

Taste	Funktion
ALT + C, ALT + O	Editorfenster schließen
F2	Markierten Datensatz/Implementierung umbenennen

8.3.5 Tastaturbefehle in der Offline-Experimentierumgebung

In der Offline-Experimentierumgebung stehen folgende spezielle Taststurbefehle zur Verfügung:

Taste	Funktion
ALT + F4	Offline-Experimentierumgebung schließen
F10	Aktivieren des Hauptmenüs
CTRL + C	Element verstellen
CTRL + M	Element messen
CTRL + S	Element stimulieren
CTRL + L	Aktiviert die Aufzeichnung der gewählten Elemente im Data-Logger
CTRL + A	Aktiviert die Aufzeichnung aller Elemente im Data-Logger
CTRL + I	Implementierung eines Elements ansehen
CTRL + U	abhängige Parameter aktualisieren

8.3.6 Mess- und Verstellfenster allgemein

Die im folgenden aufgeführten Tastaturbefehle gelten für alle Mess- und Verstellfenster gleichermaßen. Spezifische Tastaturbefehle für einzelne Mess- und Verstellfenster finden Sie weiter unten.

Taste	Funktion
CTRL + H	Anzeige hexadezimaler Werte im aktiven Fenster
CTRL + I	Information für markierte Größe anzeigen
CTRL + P	Anzeige physikalischer Werte im aktiven Fenster
CTRL + S	Aufruf des Display-Setup für das aktive Fenster (ausgenommen 3D-Grafikeditor)
DEL	Größe aus aktivem Fenster löschen (ausgenommen grafische und numerische Tabelleneditoren)

Tab. 8-2 Tastaturbefehle in allen Mess- und Verstellfenstern

Verstellfenster

Folgende Tastaturbefehle stehen – zusätzlich zu denen in Tab. 8-2 – in allen Verstellfenstern zur Verfügung:

Taste	Funktion
CTRL + M	Inkrementiert markierte Werte (nicht 3D-Editor)
CTRL + N	Dekrementiert markierte Werte (nicht 3D-Editor)
CTRL + D	Letzte Aktion wird erneut ausgeführt
CTRL + U	Letzte Aktion wird rückgängig gemacht

Folgende Tastaturbefehle stehen nur im *numerischen Editor* zur Verfügung:

Taste	Funktion
CTRL + R	Anzeige binärer Werte
CTRL + Z	Anzeige dezimaler Werte
CTRL + BILD OBEN	Verschiebt die gewählte Größe im Fenster um eine Position nach unten
CTRL + BILD UNTEN	Verschiebt die gewählte Größe im Fenster um eine Position nach oben

Folgende Tastaturbefehle stehen nur im *Tabelleneditor* zur Verfügung:

Taste	Funktion
+	Offset zu markierten Werten addieren
*	Markierte Werte mit Faktor multiplizieren
=	Markierte Zellen mit Wert füllen
CTRL + J	Dekrement x-Achsenwert (nur Kennlinie/-feld)
CTRL + K	Inkrement x-Achsenwert (nur Kennlinie/-feld)
CTRL + R	Dekrement y-Achsenwert (nur Kennfeld)
CTRL + T	Inkrement y-Achsenwert (nur Kennfeld)
CTRL + X	x-Achsenpunkt einen konkreten Wert zuweisen (nur Kennlinie/-feld)
CTRL + Y	y-Achsenpunkt einen konkreten Wert zuweisen (nur Kennfeld)

Folgende Tastaturbefehle stehen nur im *1D-* oder *2D-Grafikeditor* zur Verfügung:

Taste	Funktion
X	wechseln in die xz-Darstellung (nur 2D-Kennfeldeditor)
Y	wechseln in die yz-Darstellung (nur 2D-Kennfeldeditor)
Z	Vertauschen von x (y)-Achse und z-Achse
CTRL + B	Ermöglicht die Auswahl mehrerer Werte auf der Kennlinie.

Folgende Tastaturbefehle stehen nur im *3D-Kennfeldeditor* zur Verfügung:

Taste	Funktion
PFEIL LINKS, PFEIL RECHTS	Drehung um die Z-Achse
PFEIL OBEN, PFEIL UNTEN	Drehung um die Bildhorizontale
<NUM 4>, <NUM 6>	Drehung um die Z-Achse
<NUM 8>, <NUM 2>	Drehung um die Bildhorizontale

Messfenster

Folgende Tastaturbefehle stehen – zusätzlich zu denen in Tab. 8-2 – in allen Messfenstern zur Verfügung:

Taste	Funktion
CTRL + C	Kopiert die Einstellungen des aktuellen Messfensters in die Zwischenablage
CTRL + W	Kopiert die Einstellungen aus der Zwischenablage in das aktuelle Messfenster

Folgende Tastaturbefehle stehen in der *numerischen Anzeige*, der *Bitanzeige* sowie der *horizontalen und vertikalen Balkenanzeige* zur Verfügung:

CTRL + BILD OBEN	Verschiebt die gewählte Größe im Fenster um eine Position nach unten (vertikale Balkenanzeige: nach links)
CTRL + BILD UNTEN	Verschiebt die gewählte Größe im Fenster um eine Position nach oben (vertikale Balkenanzeige: nach rechts)

Folgende Tastaturbefehle stehen nur in der *numerischen Anzeige* zur Verfügung:

Taste	Funktion
CTRL + Z	Anzeige dezimaler Werte im aktiven Fenster
CTRL + R	Anzeige binärer Werte im aktiven Fenster

Folgende Tastaturbefehle stehen nur im *Oszilloskop- und Recorderfenster* zur Verfügung:

Taste	Funktion
CTRL + A	Passt die Skalierung der Y-Achse für den gewählten Messkanal an.
CTRL + U	Macht die letzte Skalierung rückgängig.
CTRL + L	Messkanalliste wird ein- bzw. ausgeblendet.
CTRL + X	Markierte Größe wird ein bzw. ausgeblendet.
CTRL + V	Analysemodus wird ein- bzw. ausgeschaltet.
CTRL + G	Anzeigegitter wird ein- bzw. ausgeblendet (nur Oszilloskop)
T	Triggerereignis manuell auslösen.
BILD UNTEN	Wählt den untersten Kanal in der Liste „Measure channels“ oder „Bit channels“.

Taste	Funktion
BILD OBEN	Wählt den obersten Kanal in der Liste „Measure channels“ oder „Bit channels“.
PFEIL LINKS, PFEIL RECHTS	Ausgewählten Messcursor in Einzelschritten bewegen (nur im Analysemodus)
CTRL + PFEIL LINKS, CTRL + PFEIL RECHTS	Ausgewählten Messcursor mehrere Schritte auf einmal bewegen (nur im Analysemodus)

9 Windows XP-Firewall und ASCET

In Windows XP ist ein *Personal Firewall*-Programm im Lieferumfang enthalten und standardmäßìg aktiviert. Auf vielen anderen Systemen finden sich mittlerweile häufig entsprechende Programme von unabhängigen Anbietern wie Symantec, McAffee oder BlackIce.

Firewall-Programme können in ASCET-RP die Hardwarekommunikation über die Ethernetschnittstelle behindern. Dabei werden, obwohl die Schnittstelle richtig konfiguriert ist, beim automatischen Suchen nach Hardware angeschlossene Geräte nicht gefunden. Es besteht eine hohe Wahrscheinlichkeit, dass ein installiertes Firewall-Programm die Kommunikation behindert.

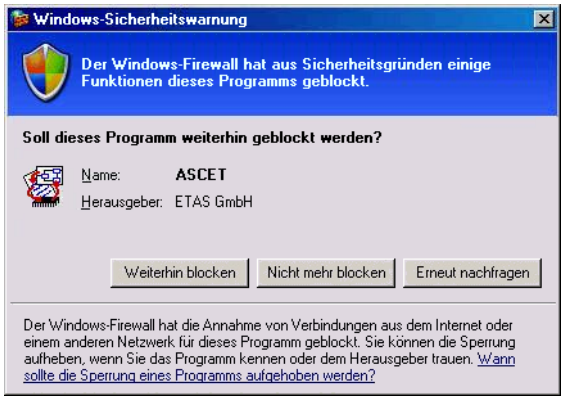
Dieses Kapitel hilft Ihnen, die Windows XP-Firewall zu knfigurieren, wenn unter Windows XP mit Service Pack 2 der Hardwarezugroff blockiert ist.

Die folgenden Aktionen in ETAS-Produkten können zu Problemen führen, wenn die Windows XP-Firewall nicht ordentlich parametrisiert ist:

- ASCET
  - ein Experiment öffnen
  - reconnecting to an experiment
- Hardware Service Pack
  - Hardwaresuche
  - Firmware-Update starten
- INCA
  - Hardwaresuche
  - Hardwarekonfigurationseditor öffnen
  - ein Experiment öffnen

9.1 Benutzer mit Administratorrechten

Wenn Sie auf Ihrem PC Administratorrechte haben, öffnet sich das folgende Dialogfenster, wenn die Firewall ein ETAS-Produkt blockiert.



Ein Produkt freischalten:

- Klicken Sie im Fenster „Windows-Sicherheitswarnung“ auf **Nicht mehr blocken**.

  Die Firewall blockiert das ETAS-Produkt (im Beispiel: ASCET) nicht mehr. Die Einstellung wird bei einem Neustart des Produkts oder des PC beibehalten.

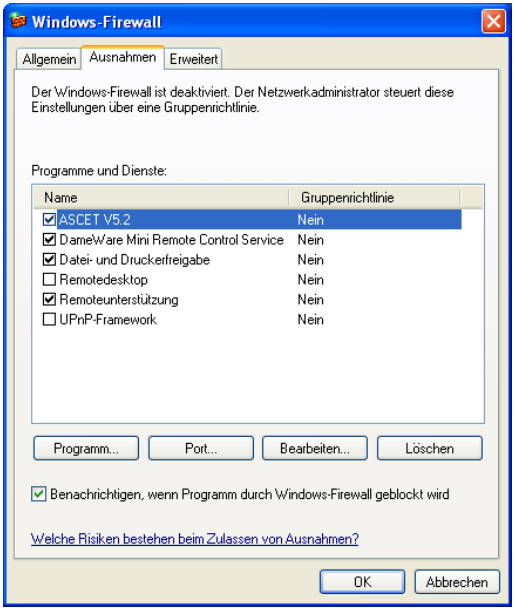
Anstatt auf das Fenster „Windows-Sicherheitswarnung“ zu warten, können Sie ETAS-Produkte vorab freischalten.

ETAS-Produkte in der Firewall-Steuerung freischalten:

- Wählen Sie im Windows-Startmenü **Einstellungen** → **Systemsteuerung**.

- In der Systemsteuerung doppelklicken Sie auf das Symbol **Windows-Firewall**, um das Fenster „Windows-Firewall“ zu öffnen.

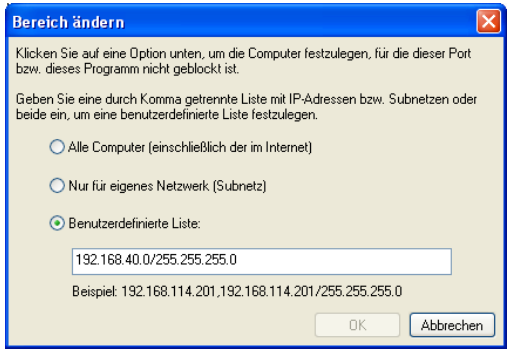
- Im Fenster „Windows-Firewall“ öffnen Sie das Register „Ausnahmen“.



  Dieses Register listet die Ausnahmen, die nicht durch die Firewall blockiert werden. Benutzen Sie die Schaltflächen **Programm** oder **Bearbeiten**, um neue Programme hinzuzufügen oder vorhandene zu bearbeiten.

- Stellen Sie sicher, dass die ETAS-Produkte und -Dienste, die Sie verwenden wollen, richtig konfigurierte Ausnahmen sind.

– Öffnen Sie das Fenster „Bereich ändern“.



– Stellen Sie sicher, dass wenigstens die IP-Adressen `192.168.40.xxx` freigeschaltet sind, um funktionierenden Zugriff auf ETAS-Hardware zu gewährleisten.

– Schließen Sie das Fenster „Bereich ändern“ mit **OK**.

- Schließen Sie das Fenster „Windows-Firewall“ mit **OK**.

  Die Firewall blockiert das ETAS-Produkt (im Beispiel: ASCET) nicht mehr. Die Einstellung wird bei einem Neustart des PC beibehalten.

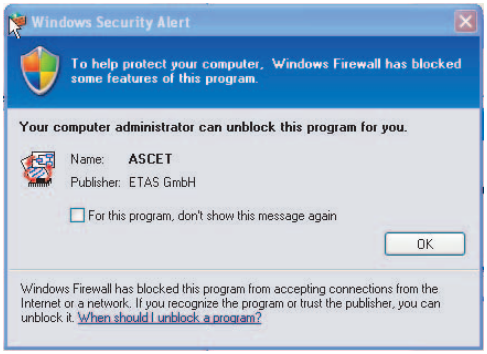
9.2 Benutzer ohne Administratorrechte

Dieses Kapitel richtet sich an Benutzer mit eingeschränkten Rechten, z.B. keine Änderungen am System, eingeschränkte Schreibrechte, lokaler Login.

Die Arbeit mit einem ETAS-Produkt erfordert die Rechte „Write“ und „Modify“ in den Verzeichnissen `ETAS`, `ETASData` und den temporären ETAS-Verzeichnissen. Andernfalls erscheint eine Fehlermeldung, wenn das Produkt (z.B. ASCET) gestartet und eine Datenbank geöffnet wird. Ein korrekter etrieb des Produkts ist nicht möglich, da die Datenbank-Datei sovie verschiedene `*.ini`-Dateien während der Arbeit geändert werden.

ASCET muss in jedem Fall von einbem Administrator installiert werden. Es wird empfohlen, dass der Administrator sicherstellt, dass das ETAS-Produkt oder die Prozesse nach der Installation zur Liste der gewählten Ausnahmen der Windows-Firewall hinzugefügt werden. Wenn das nicht passiert, geschieht folgendes:

- Das Fenster „Windows-Sicherheitswarnung“ öffnet sich, wenn eine der auf Seite 243 genannten Aktionen ausgeführt wird.



Ein Produkt freischalten (ohne Administratorrechte):

- Aktivieren Sie im Fenster „Windows-Sicherheitswarnung“ die Option.
- Schließen Sie das Fenster mit **OK**.

  Ein Administrator muss das Produkt (z.B. ASCET) im Register „Ausnahmen“ des Fensters „Windows-Firewall“ auswählen, um künftige Probleme beim Hardwarezugriff mit dem betreffenden Produkt zu vermeiden.

9.3 Support und Problembericht

Wenn Sie Fragen haben, wenden Sie sich an die ETAS-Hotline.

	Telefon	**E-Mail**
Europa (ohne Frankreich, Belgien, Luxemburg, Großbritannien)	+49–711–89661–311 (ASCET) +49–711–89661–315 (INCA)	ec.hotline@etas.de inca.hotline@etas.de
Frankreich, Belgien, Luxemburg	+33–1–5670–0235 (ASCET) +33–1–5670–0234 (INCA)	support.ascet@etas.fr support.inca@etas.fr
Großbritannien	+44–1283–546–512	support@etas-uk.net

	Telefon	E-Mail
Japan	+81–45–222–0951 (ASCET) +81–45–222–0950 (INCA)	ec.hotline@etas.co.jp inca.hotline@etas.co.jp
Korea	+82(2)5747–101 (ASCET) +82(2)5747–061 (INCA)	ec.hotline@etas.co.kr inca.hotline@etas.co.kr
USA	+1–888-ETASINC (382–7462)	support@etasinc.com

10 ETAS Kontaktinformation

ETAS Hauptsitz

ETAS GmbH

Borsigstraße 14	Telefon:	+49 711 89661-0
70469 Stuttgart	Telefax:	+49 711 89661-105
Germany	E-Mail:	sales@etas.de
	WWW:	www.etasgroup.com

Nordamerika

ETAS Inc.

3021 Miller Road	Telefon:	+1 888 ETAS INC
Ann Arbor, MI 48103	Telefax:	+1 734 997-9449
USA	E-Mail:	sales@etas.us
	WWW:	www.etasgroup.com

Japan

ETAS K.K.

Queen's Tower C-17F	Telefon:	+81 45 222-0900
2-3-5, Minatomira, Nishi-ku	Telefax:	+81 45 222-0956
Yokohama 220-6217	E-Mail:	sales@etas.co.jp
Japan	WWW:	www.etasgroup.com

Großbritannien

ETAS Engineering Tools Application and Services Ltd.

Studio 3, Waterside Court	Telefon:	+44 1283 54 65 12
3rd Avenue, Centrum 100	Telefax:	+44 1283 54 87 67
Burton-upon-Trent	E-Mail:	sales@etas-uk.net
Staffordshire DE14 2WQ	WWW:	www.etasgroup.com
Great Britain		

Frankreich

ETAS S.A.S.

1, place des Etats-Unis	Telefon:	+33 1 56 70 00 50
SILIC 307	Telefax:	+33 1 56 70 00 51
94588 Rungis Cedex	E-Mail:	sales@etas.fr
France	WWW:	www.etasgroup.com

Korea

ETAS Korea Co. Ltd.

4F, 705 Bldg. 70-5	Telefon:	+82 2 57 47-016
Yangjae-dong, Seocho-gu	Telefax:	+82 2 57 47-120
Seoul 137-889	E-Mail:	sales@etas.co.kr
Korea		

China

ETAS (Shanghai) Co., Ltd.

2404 Bank of China Tower	Phone:	+86 21 5037 2220
200 Yincheng Road Central	Fax:	+86 21 5037 2221
Shanghai 200120, P.R. China	E-mail:	sales.cn@etasgroup.com
	WWW:	www.etasgroup.com

Index