Simulation, Parametrierung und Test von automotive Object-Code mit Matlab®/Simulink®

Thomas Zurawka

Abstract

The need to shorten the development time, reduce costs and increase the quality of the ECU software has driven companies to adopt virtual, PC-based simulation techniques rather than rely on expensive in-vehicle and dynamometer set-ups.
A major challenge is here the support of ECU object code in simulation applications based on Matlab®/Simulink®.
The new tool INCODIO overcomes these deficiencies.
INCODIO® supports simulation, calibration and test of automotive object-Code with Matlab/Simulink.

Kurzfassung

Um die Entwicklungszeiten weiter zu verkürzen, die Kosten zu reduzieren und die Qualität der ECU-Software zu erhöhen, haben viele Firmen in den letzten Jahren virtuelle, PC-basierte Simulationstechniken eingeführt.
Eine große Herausforderung ist dabei die Unterstützung von *ECU Object-Code* innerhalb von SIL-Anwendungen mit Matlab®/Simulink®.
Das Werkzeug INCODIO® schafft hier Abhilfe.
INCODIO® unterstützt die Simulation, Parametrierung und den Test von automotive Object-Code mit Matlab®/Simulink®. Dies ermöglicht SIL-Anwendungen für unterschiedliche Zwecke bei gleichzeitiger Gewährleistung des Know-How Schutzes.

1 Einführung und Motivation

1.1 Übersicht

Fahrzeuge sind heute mit einer Vielzahl von elektronischen Steuergeräten (engl. *E*lectronic *C*ontrol *U*nits - ECU) ausgestattet. Diese ECUs enthalten eine große und stetig zunehmende Menge an *Software*.

Die Entwicklung hochqualitativer ECU Software durch hunderte von Software-Entwicklern an vielen weltweiten Standorten ist eine große Herausforderung, welche durch exzellente Prozesse, Methoden und Tools unterstützt werden muss [1].

Die grundlegende Software-Architektur einer ECU ist in Bild 1-1 dargestellt. Die ECU-Software enthält hierbei mindestens 2 Ebenen: Anwendungs-Software und Basis-Software [1]. Jede Ebene enthält Software-Komponenten (*SWK*). Eine Software-Komponente besteht aus Code und Daten.

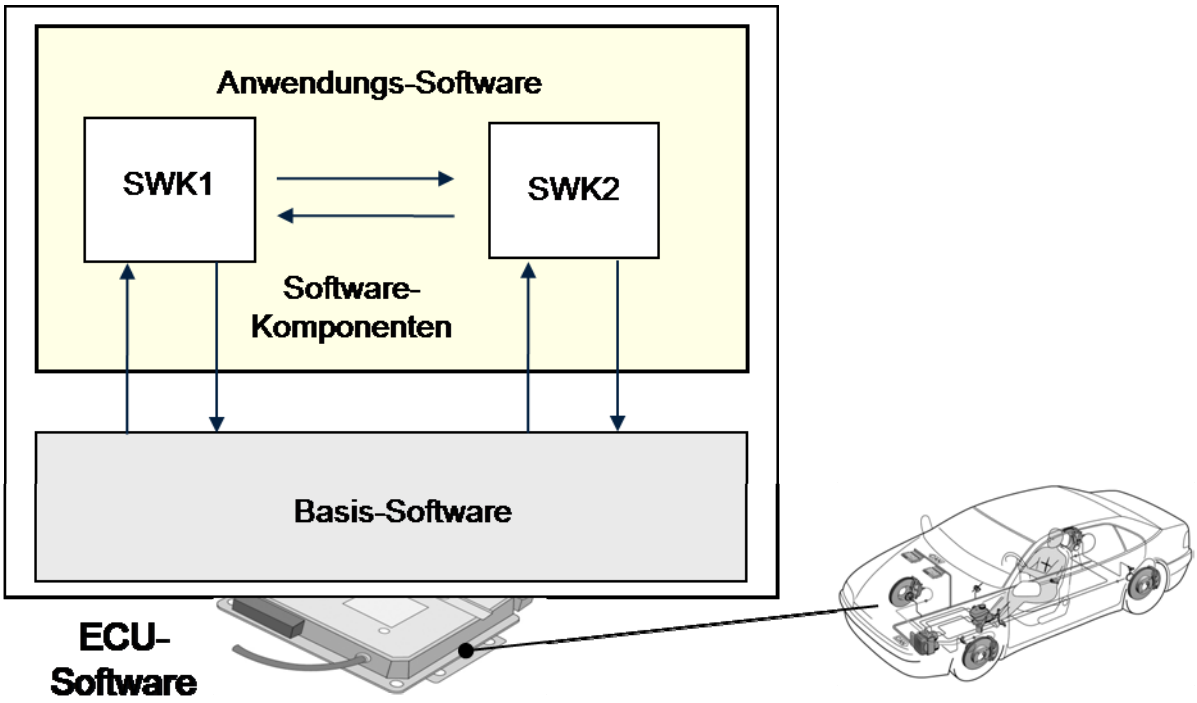

Bild 1-1: Architektur der ECU-Software

Ein weiterer wichtiger Aspekt ist, dass die einzelnen Software-Komponenten zunehmend von Fahrzeugherstellern und deren Zulieferern gemeinsam entwickelt werden. Aus *Gründen des Know-How Schutzes*, wird dabei meist nur der *ECU Object-Code* zwischen den Partnern ausgetauscht.

Die ECU-Software wird dabei in mehreren Phasen innerhalb des V-Zyklus entwickelt [1], siehe Bild 1-2.

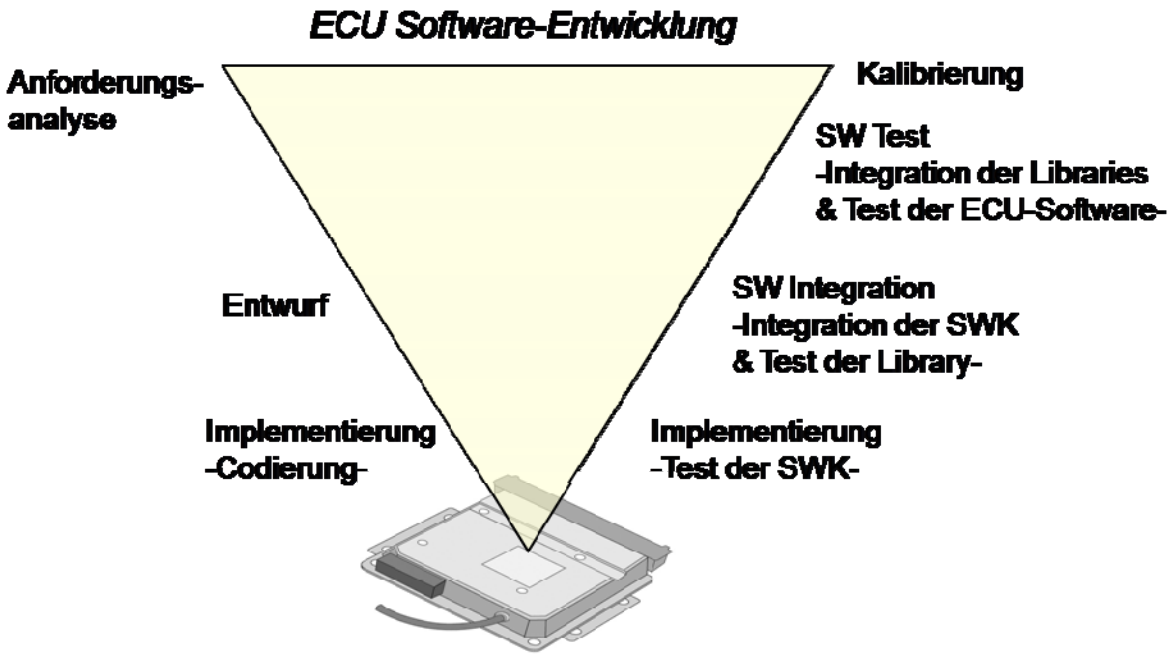

Bild 1-2: Entwicklung der ECU-Software (“V-Zyklus”)

Bei der Entwicklung der Software-Komponenten werden heute die folgenden Methoden eingesetzt:

- *Modellbasierte Entwicklung* mit automatischer Codegenerierung, z.B. mit Matlab/Simulink, Targetlink, ASCET etc.
- *Entwicklung* von Software-Komponenten mit der Programmiersprache *C/C++* [1].

Diese Aktivitäten werden typischerweise in 3 unterschiedlichen Umgebungen durchgeführt:

- *Virtuelle* Umgebung, d.h. die Entwicklung erfolgt nur auf dem Computer
- *Labor*, d.h. die Entwicklung erfolgt mit einem Computer und zusätzlicher Hardware. Die Hardware kann ein simuliertes Laborauto oder eine Fahrzeugkomponente eines Prüfstands sein.
- *Fahrzeug*, d.h. die Entwicklung erfolgt im Fahrzeug selbst.

Noch heute wird der überwiegende Teil der Arbeit im Fahrzeug und Labor ausgeführt. Um die Entwicklungszeiten weiter zu verkürzen, ist es zukünftig unabdingbar, Software zunehmend in einer virtuellen Umgebung zu entwickeln.

1.2 Motivation

Betrachtet man die einzelnen V-Zyklus Phasen genauer ergibt sich derzeit folgendes Bild bzgl. der eingesetzten Methoden in den Phasen “Software-Integration” und „Software-Test“:

- Einsatz von Hardware-in-the-Loop Tests (HIL) im Labor: Integration der ECU Hardware & ECU Software sowie Test der Serien-ECU (Software & Hardware) zusammen mit einem simulierten Fahrzeug.

- Einsatz von Software-in-the-Loop Tests (SIL) in einer virtuellen Umgebung: Test der Serien-ECU (nur Software!) zusammen mit simulierter ECU Hardware und einem simulierten Fahrzeug. Einzelne Komponenten der ECU-Software sind dabei nur als Object-Code verfügbar

HIL-Tests sind heute breit in der Industrie eingeführt, jedoch aufgrund des erforderlichen Hardware-Equipments wenig flexibel und sehr teuer.
SIL-Tests für die ECU Software bieten hier eine wesentlich bessere Lösung.

Warum sind SIL-Anwendungen so wichtig?
Die Einführung des neuen Standards AUTOSAR [3] unterstützt die klare Trennung zwischen ECU Hardware und ECU Software und damit auch ihren isolierten Test. HIL wird aller Voraussicht nach aus diesem Grund zukünftig primär zum Test von ECU Hardware eingesetzt werden.
Die SIL-Technologie bietet zudem die Möglichkeit der (automatischen) Kalibrierung von Parametern, Kennlinien und Kennfeldern in einer virtuellen Umgebung. Aktuell sind hierfür aber die Fahrzeugmodelle vielfach noch unzureichend.

Für die SIL-Technologie spielt das weit verbreitete Werkzeug Matlab®/Simulink® ebenfalls eine wichtige Rolle, da dieses Tool umfangreiche Analyse- und Optimierungsverfahren bietet, die z.B. für eine automatische Kalibrierung oder zur Simulation von Fahrzeugen zwecks Verbrauchsoptimierung sehr wichtig ist.

Ziel ist damit die Bereitstellung einer SIL-Technologie auf Basis von Matlab®/Simulink®, welche die Simulation von Object-Code und C-Code einer ECU-Software unterstützt.

1.3 Lösung

Da die Simulation, Parametrierung und Test von Object-Code mit Matlab/Simulink nicht unterstützt wird, wird meist ein äußerst aufwändiges Re-Engineering des Object-Codes durchgeführt, um z.B. SIL-Anwendungen mit Simulink zu ermöglichen.

Das Werkzeug INCODIO® schafft hier Abhilfe.
INCODIO® analysiert den prozessorspezifischen Object-Code sowie die Beschreibungsdateien und generiert automatisch eine Matlab/Simulink S-Function mit integriertem Prozessor-Simulator sowie Editoren zur Parametrierung der ECU-Software.

Anschließend erfolgt in üblicher Weise der Test und die automatische Kalibrierung der ECU-Software zusammen mit Matlab/Simulink Fahrzeugmodellen. Hierbei werden während der Laufzeit der Simulation permanent veränderte Kalibrierdaten auf die ECU-Software geladen und gleichzeitig alle relevanten Messgrößen aufgezeichnet.
Das nächste Kapitel beschreibt diese neue Technologie ausführlich.

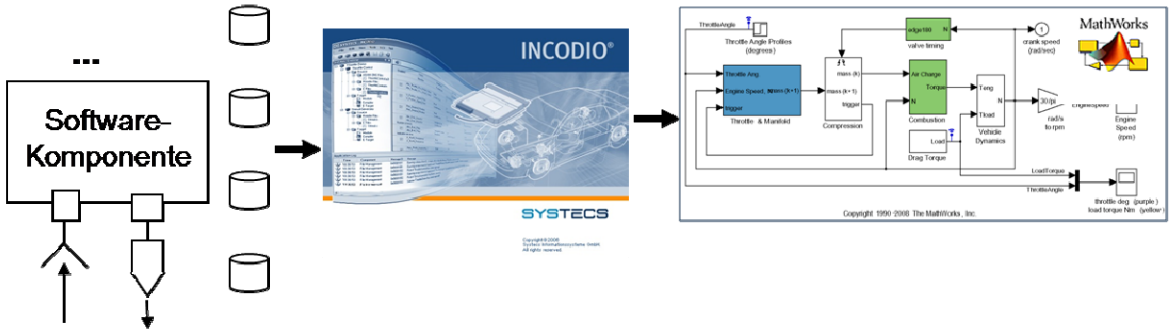

Bild 1-3: Integration von Object-Code in Matlab/Simulink mit INCODIO®

2 Simulation, Parametrierung & Test von OBJ-Code

Die Software für eine Serien-ECU wird für spezielle Mikrocontroller erstellt. Jede Software-Komponente wird dabei wie folgt beschrieben (Bild 2-1):

- *Object-Dateien .obj*, Header-Dateien .h *oder* C- und Header Dateien; teilweise ohne Datendefinition für Messgrößen, Parameter (einfach, 1-D, 2-D)
- Beschreibungsdatei für die Schnittstelle, d.h. Definition der Eingangs- und Ausgangsgrößen (AUTOSAR [3], MSR [4], ASAM-MCD [5])
- Beschreibungsdatei für das Echtzeitverhalten, d.h. Definition der Prozesse und Timing-Parameter (OSEK/VDX [6], MSR [4], AUTOSAR [3])
- Beschreibungsdatei für Mess- und Kalibriergrößen (MSR [4], ASAM-MCD [5])

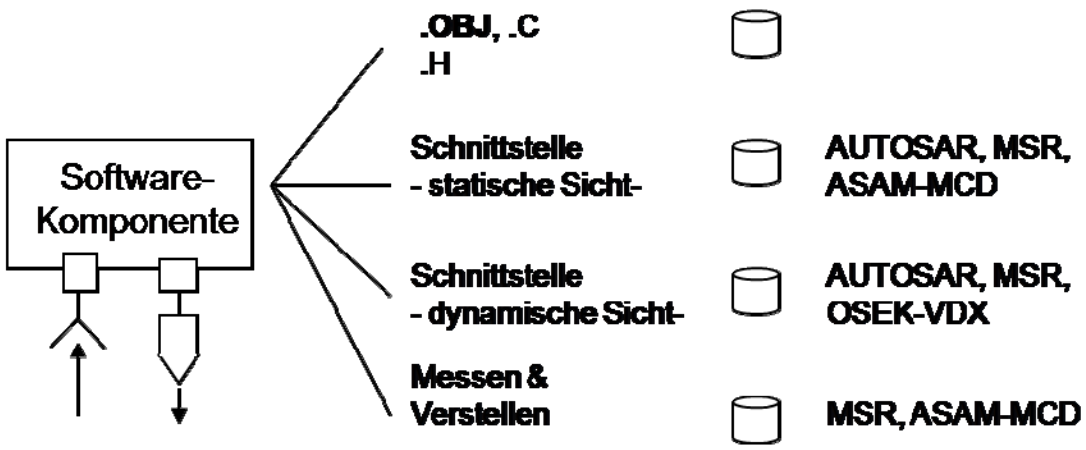

Bild 2-1: Beschreibung einer Software-Komponente

Diese originale ECU-Software kann nicht unmittelbar für virtuelle Anwendungen eingesetzt werden, da der C-Code bzw. Object-Code abhängt von:

- vom Mikrocontroller, z.B. von speziellen Bitoperationen
- von der ECU, z.B. durch spezielle Speicher-Layouts für Parameter und Konstanten
- vom Compiler, z.B. durch „pragma“-Anweisungen

Hinzu kommt, dass Simulationstools wie Matlab®/Simulink® die speziellen Ablageschemata für Kennlinien und Kennfelder nicht unterstützen, d.h. eine Kalibrierung bzw. Applikation dieser Größen ist nicht möglich.
Des weiteren sind unterschiedliche Namenskonventionen für Software-Komponenten und Variablen eine große Hürde bei der Simulation, insbesondere falls C- bzw. Object-Code von unterschiedlichen Firmen integriert werden muss.

Aus diesen Gründen muss der Object-Code oder auch der C-Code, falls dieser verfügbar ist, für Simulationstools wie Matlab®/Simulink® entsprechend aufbereitet werden.
Folgende Anforderungen existieren für eine benutzerfreundliche Integration der Software-Komponente:

- Prozesse, Schnittstellen, Echtzeitverhalten einer Software-Komponente müssen in Simulink, z.B. über eine S-Function, automatisch angelegt werden, um Inkonsistenzen zu vermeiden
- Kalibrierung bzw. Applikation des Object-Code muss unterstützt werden, d.h. eine reine Black-Box Integration reicht nicht aus

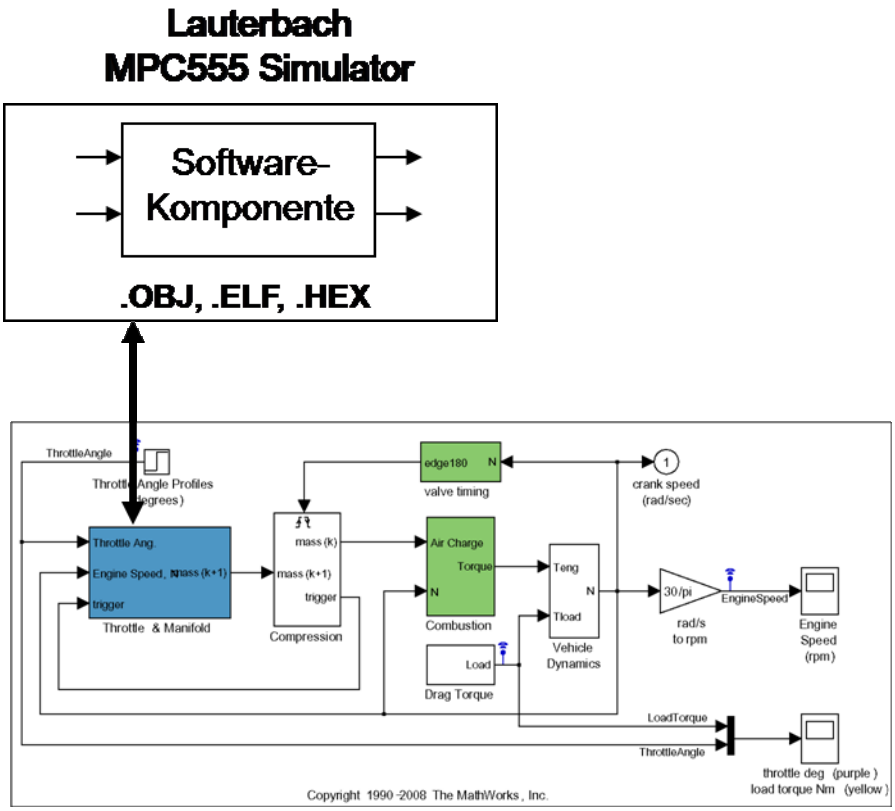

Bild 2-2: Object-Code Technologie im Detail

Als Prozessorsimulator wurde der Lauterbach-Simulator für den MPC555 eingesetzt (Bild 2-2). Dieser bietet eine offene Schnittstelle für die Fernsteuerung durch eine andere Anwendung. Die Schnittstelle ist adressbasiert und prozessorunabhängig.

Pro Simulationsschritt in Simulink werden folgende Aktivitäten innerhalb der S-Function durchgeführt:

- Schreiben der Eingänge in den Prozessorsimulator
    - Die Adressen sind aus der ASAM-MCD-2MC Beschreibung bekannt
-
- Schreiben der Zustandsspeicher (globale und statische Variablen) in den Prozessorsimulator
    - Die Adressen sind aus der ASAM-MCD-2MC Beschreibung bekannt

- Methode einer Software-Komponente im Prozessorsimulator ausführen
    - N Schritte von Startadresse bis zur Endadresse
    - Die Adressen müssen aus der .OBJ-Datei extrahiert werden

- Lesen der Ausgänge vom Prozessorsimulator
    - Die Adressen sind aus der ASAM-MCD-2MC Beschreibung bekannt

Im Rahmen der Initialisierung der S-Function werden einmalig alle Parameter der Software-Komponente, z.B. alle Kennlinien und Kennfelder, an den Prozessorsimulator übertragen. Die Startadressen und die Ablageschemata sind aus der ASAM-MCD-2MC Beschreibung bekannt.
Die S-Function sowie die Editoren für deren Parameter werden vollständig automatisch generiert.

Es ist wichtig, dass das Verhalten der simulierten Software-Komponente mit dem Verhalten der originalen ECU-Software des Fahrzeugs übereinstimmt. Um dies überprüfen zu können, müssen im Fahrzeugbetrieb aufgezeichnete Messdaten als Stimuli-und Referenzsignale der Software-Komponente in Matlab/Simulink verwendet werden können.
Mit Hilfe dieser Technik kann auch das Fahrzeugmodell auf hinreichende Übereinstimmung mit dem realen Fahrzeug überprüft werden.

INCODIO® unterstützt diese Anwendungsfälle durch automatische Generierung von Stimuli-Blöcken aus MDF-Dateien für Matlab®/Simulink®.

Die ECU-Software und das Fahrzeugmodell sind nun bereit für Simulation, Parametrierung und Test.

Der letzte Schritt ist die automatische Applikation bzw. Kalibrierung der ECU-Software in Matlab®/Simulink®. Während der Simulation wird die ECU-Software mit neuen Kalibrierdaten überschrieben. Die Daten müssen dabei in den Standardformaten MSR (CDF) oder DCM vorliegen.
Parallel hierzu werden alle relevanten Messgrößen in Matlab/Simulink im MDF-Format aufgezeichnet (Bild 2-3).

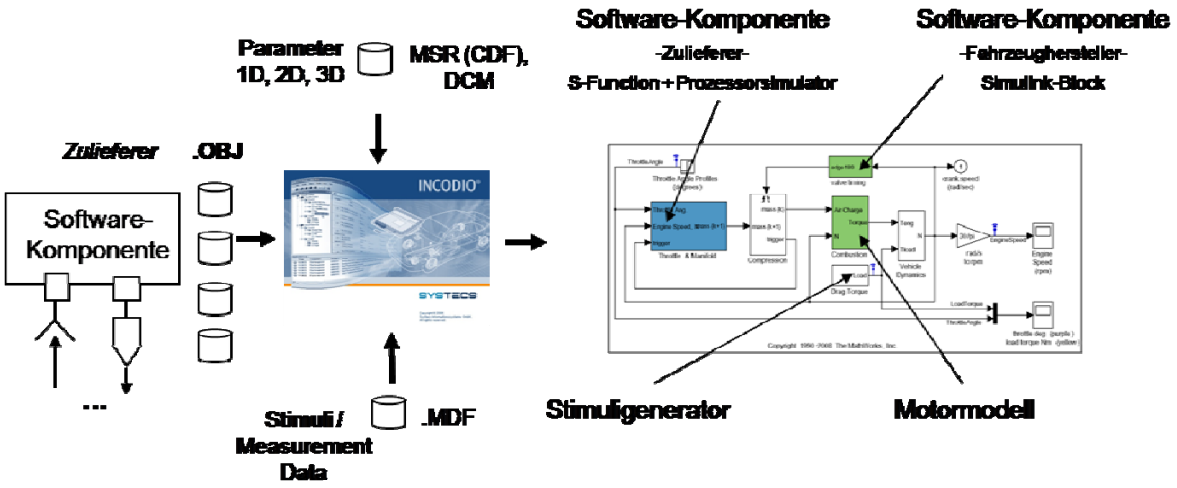

Bild 2-3: Simulation, Parametrierung und Test von Object-Code mit Matlab/Simulink

3 INCODIO® in Kurzform

INCODIO® unterstützt SIL-Anwendungen mit Matlab/Simulink mittels durchgängiger Integration von Embedded Object- und C-Code. Dieses Kapitel gibt eine kurze Einführung in das Tool, weitere Informationen findet man in [2].

Das Hauptfenster von INCODIO® ist sehr benutzerfreundlich und besteht aus 2 Teilen (Bild 3-1):

- Browser für Bibliotheken und Software-Komponenten
- Working Sheet

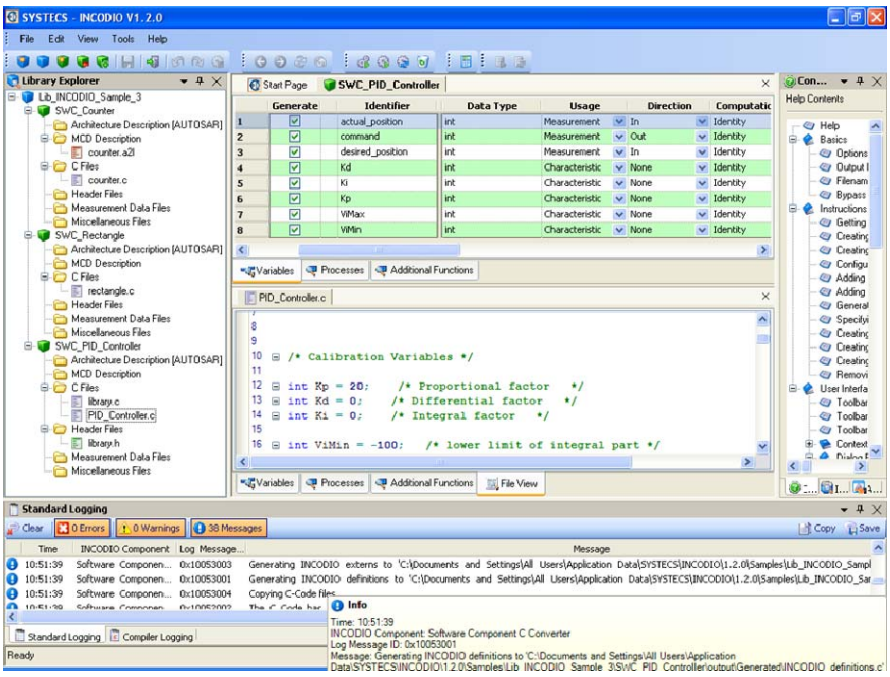

Bild 3-1: Benutzerschnittstelle von INCODIO®

Der Browser unterstützt primär die folgenden Anwendungsfälle für einzelne Software-Komponenten:

- Einlesen und Analyse des Object- -und C-Code
- Einlesen und Analyse der Beschreibungsdateien (Mess- und Kalibriergrößen, Echtzeitverhalten, Schnittstelle)
- Generierung der Simulink S-Function inklusive dazugehöriger Editoren für die Kalibriergrößen

- Einstellen der INCODIO® Parser und Generatoren, um verschiedene Compiler und Simulations-Tools zu unterstützen

Das Working Sheet beinhaltet einige Editoren für Software-Komponenten:

- Editoren für alle Dateien
- Editor für ASAM-MCD-2MC Information, d.h. für Umrechnungsformeln, Messgrößen etc., um diese in Simulink in physikalischen Größen anzeigen zu können
- Editoren für Schnittstellen und Echtzeitverhalten

Nach der Generierung der Simulink S-Function und der Konfigurations-Dateien für den Lauterbach Prozessor-Simulator kann die Software-Komponente in Matlab/Simulink simuliert, parametriert und getestet werden.

4 Zusammenfassung

Zum frühen Test und zur automatischen Kalibrierung einer ECU-Software, z.B. einer Motor- oder Getriebesteuerung, bietet die *Software-in-the-Loop (SIL) Technologie* erhebliche Vorteile.
Hierbei wird ein Teil der oder die gesamte Original-Software einer ECU zusammen mit zeitkontinuierlichen Modellen eines Fahrzeugs auf einem oder mehreren PCs z.B. mit Hilfe von Matlab/Simulink simuliert.

Die Original-Software einer ECU wird für spezielle Mikrocontroller geschrieben und umfasst neben C-Code oder Object-Code (compilierter C-Code), Beschreibungsdateien für das dynamische Verhalten einer Fahrzeugfunktion (MSR, OSEK/VDX) sowie für Mess– und Kalibriergrößen (ASAM-MCD-2MC).
Die Original-Software kann, aufgrund mikroprozessorspezifischer Befehle sowie spezieller Ablageschemata für die Kalibriergrößen, dabei nicht ohne weiteres für SIL-Anwendungen eingesetzt werden.

Eine große Herausforderung für Fahrzeughersteller ist weiterhin, dass meist nur der *ECU Object-Code der Zulieferer* für SIL-Anwendungen zur Verfügung steht.

Simulationstools wie Matlab/Simulink unterstützen diesen Anwendungsfall nicht, deshalb wird meist ein äußerst aufwändiges Re-Engineering des Object-Codes durchgeführt, um SIL-Anwendungen zu ermöglichen.

Das Werkzeug INCODIO® schafft hier Abhilfe.

INCODIO® analysiert den prozessorspezifischen Object-Code sowie die Beschreibungsdateien und generiert automatisch eine Matlab/Simulink S-Function mit integriertem Prozessor-Simulator sowie Editoren zur Parametrierung der ECU-Software.

Anschließend erfolgt in üblicher Weise der Test und die automatische Kalibrierung der ECU-Software zusammen mit Matlab/Simulink Fahrzeugmodellen. Hierbei werden während der Laufzeit der Simulation permanent veränderte Kalibrierdaten auf die ECU-Software geladen und gleichzeitig alle relevanten Messgrößen aufgezeichnet.

5 Literatur

[1] J. Schäuffele, T. Zurawka: Automotive Software Engineering, Vieweg 2003, SAE 2005, PHEI 2007, Star Japan 2008.

[2] SYSTECS (www.systecs.com): INCODIO® Manual, 2010.

[3] AUTOSAR (www.autosar.org): Standard & auxiliary specifications, 2010.

[4] MSR (www.msr-wg.de): Standard specifications, 2006.

[5] ASAM (www.asam.net): Standard specifications, 2006.

[6] OSEK/VDX (www.osek-vdx.org): Standard specifications, 2006.

[7] ETAS (www.etas.de): ASCET Manual, 2010.

[8] The Mathworks (www.mathworks.com): Matlab®/Simulink® Manual, 2010.