Rapid-Control-Prototyping und Hardware-in-the-Loop Simulationen mit xPC Target and xPCLabDesk/LabVIEW

N. Bajcinca
Institut für Robotik und Mechatronik,
DLR, Oberpfaffenhofen

Kurzfassung

xPC Target bietet eine flexible und generische Umgebung für Rapid-Control-Prototyping (RCP) und Hardware-in-the-Loop (HIL) Simulationen. xPCLabDesk ist ein in LabVIEW entwickeltes Mensch-Maschine-Werkzeug zur Gestaltung von benutzer-freundlichen GUIs für die Interaktion mit xPC-Echtzeitanwendungen. xPCLabDesk bietet eine einfache und übersichtliche Parameteranpassung, Signal- und Parametererfassung, Datenaufzeichnung und Automatisierung verschiedener Testabläufe. Somit stehen dem Entwurf einer Echtzeitanwendung einerseits die Matlab/Simulink Entwicklungs-Werkzeuge und andererseits die LabVIEW Umgebung zum Aufbau einer entsprechenden Mensch-Maschine-Schnittstelle zur Verfügung. Die Architektur vom xPCLabDesk bietet eine generische Basis, die auch für andere Echtzeitplatformen einfach erweitert werden kann.

1. Einleitung

Der Entwurfsprozess einer bestimmten Funktion im regelungstechnischen Sinne erfolgt grundsätzlich in zwei Entwicklungsphasen. Mit der Rapid Control Prototyping (RCP) Phase werden Methoden bzw. Werkzeuge bezeichnet, die zum schnellen und effizienten Bearbeiten von regelungstechnischen Problemstellungen dienen. Das beinhaltet (1) die theoretischen Methoden der Regelungstechnik von der Modellbildung und der Systemanalyse bis zum Entwurf eines Regelungskonzeptes, (2) den Test und die Voroptimierung der Regelung in der Simulation, (3) die automatische Code-Erzeugung für einen Echtzeitrechner innerhalb einer realen Versuchsumgebung und (4) die Verifikation und Optimierung der Regelung mit dem zu regelnden Objekt auf dem Versuchsstand. Das Ergebnis ist dann ein fertiger Regler-Prototyp der auf einem Steuergerät implementiert und optimiert werden muss.
Ein zuverlässiger und kosteneffizienter Test von Steuergeräten ist nur in einem geschlossenen Regelkreis möglich, wobei dem Steuergerät eine möglichst realitätsnahe Simulationsumgebung bereitgestellt wird. Das Steuergerät soll also identische Signale „sehen" mit denen es in einem echten Fahrzeugeinsatz interagiert. Diese Wechselwirkung zwischen dem Steuergerät und der Simulationsumgebung stellt die zweite Entwicklungsphase dar und wird als Hardware-in-the-Loop (HIL)-Simulation bezeichnet. Ein Vorteil dieses Ansatzes liegt vor allem an den umfangreichen Test- und Optimierungsmöglichkeiten des zu entwerfenden Systems ohne direkte Präsenz der echten physikalischen Regelstrecke. Die in der Realität nicht oder nur schwer messbaren Zustände der Regelstrecke sind am HIL-Simulator einfach zugänglich. Ein weiterer Vorteil ist die Möglichkeit, Steuergeräte in automatisierten Testprozeduren zu testen bzw. optimieren.

xPC Target ist eine sehr leistungsfähige Host-Target Simulationsumgebung, die die Ausführung von RCP/HIL Echtzeitsimulationen der in MATLAB/Simulink entwickelten Anwendungen auf einer kostengünstigen PC-Hardware ermöglicht. Somit wird die zunehmende Leistungsfähigkeit der millionenfach eingesetzten Prozessoren in PCs und gleichzeitig die extrem verbreitete PC-basierte Messtechnik optimal ausgenutzt, und somit dem Benutzer eine hohe Flexibilität in der Hardwarebeschaffung und -aktualisierung anbietet.
In diesem Beitrag wird ein neues MMI Softwarewerkzeug für xPC Target, das *xPCLabDesk* (**xPC** Target **Lab**VIEW Control **Desk**), vorgestellt, dass in LabVIEW programmiert wurde. Hierfür wird eine von *TheMathWorks* bereitgestellte C-API Bibliothek von über 90 Funktionen verwendet. *xPCLabDesk* stellt dem Benutzer (bereits) zwei Werkzeuge zur Verfügung, das *Developer xPCLabDesk* und das *Run-Time xPCLabDesk*. Das *Developer xPCLabDesk* läuft auf dem LabVIEW Kernel und wird beim Entwurf der MMI Schnittstelle benötigt. *Run-Time xPCLabDesk* hingegen, braucht nur das Run-Time Engine von LabVIEW, der seitens National Instruments kostenlos zur Verfügung gestellt wird. Somit können fertige *xPCLabDesk* Anwendungen auf Systeme ohne LabVIEW portiert und benutzt werden. Durch das xPCLabDesk verfügt der Benutzer, beim Aufbau einer xPC Target MMI Schnittstelle, über das ganze Spektrum der LabVIEW Programmierwerkzeuge. Darüberhinaus kann man xPC Target mit weiteren LabVIEW- kompatibelen Werkzeugen wie DIAdem, TestStand, usw. integrieren.

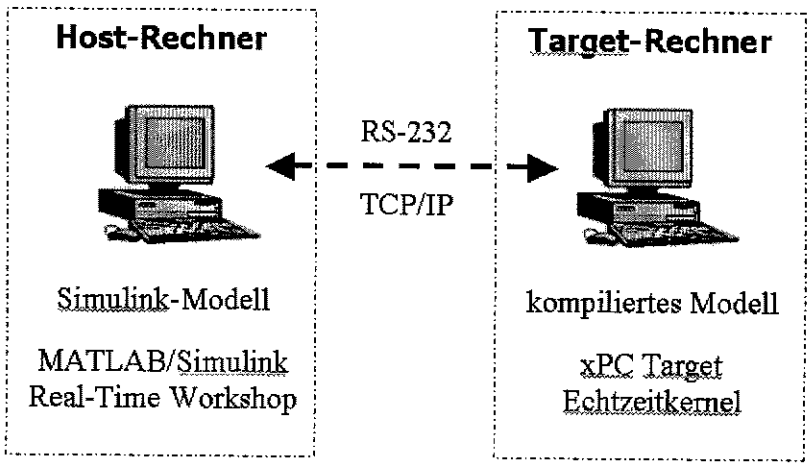

Abb 1: xPC Host-Target Umgebung

2. Echtzeitsimulation mit xPC Target

xPC Target ist eine Host-Target-Echtzeitumgebung von der Firma *The MathWorks*, die es ermöglicht, Simulink-Modelle mit physischen Systemen zu verbinden und sie auf PC-kompatibler Hardware in Echtzeit auszuführen. Die Host-Target-Umgebung besteht aus zwei physisch getrennten Rechnern, die entweder mittels serieller Schnittstelle (RS-232/V24) oder über Ethernet (TCP/IP-Protokoll) miteinander kommunizieren. Auf dem Target-Rechner wird der xPC-Echtzeitkernel geladen, und dadurch das sonst übliche Taskmanagement der Rechenzeiten umgangen. Das xPC Realtime Kernel ist ein Bestandteil von *xPC Target* und wird vom Matlab im Host auf einer Bootdiskette erzeugt. Dies ermöglicht die Berechnung eines in C-Code kompilierten Modells in Echtzeit und gewährleistet hohe Systemleistungen. Abbildung 1. zeigt den schematischen Aufbau einer „PC-in-the-Loop" unter Verwendung von xPC Target. Dabei kann der Host PC ein Desktop-PC oder Notebook sein und als Target können Desktop PCs, Industrie PCs, PC/104, CompactPCI und Single-Board-Computer, zB die xPCTargetBox , eingesetzt werden. xPC Target unterstützt eine große Anzahl unterschiedlichster Hersteller, worin A/D, D/A, Digitales I/O, inkrementale Enkoder, CAN-Bus, Watchdog, RS232, usw. enthalten sind.
Nach dem Aufbau des Modells im Simulink wird per Knopfdruck über Real-Time-Workshop erzeugte C-Code übersetzt und in einem speziellen DLM Format gelinkt und anschließend auf den mit dem xPC

Realtime-Kernel gebootenen Target-PC geladen. Anschließend können beispielsweise vom Host aus die Parameter des Modells geändert bzw. gelesen werden (Parameteranpassung und -erfassung) oder die Signale an der Host- bzw. Targetseite angezeigt (Signalerfassung) bzw. gespeichert (Datenaufzeichnung) werden.

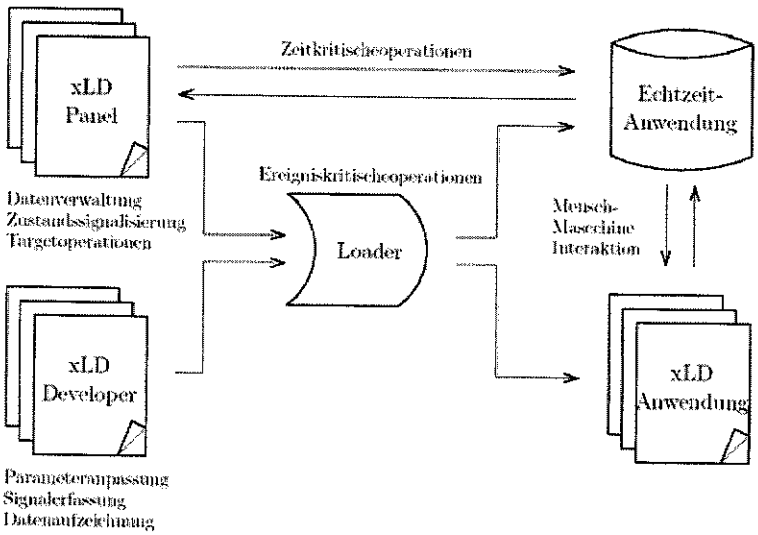

Abb 2: Architektur des xPCLabDesks

2. Die Architektur des xPCLabDesk

Die Hauptbauteile des xPCLabDesk sind (1) das *xLD Panel*, (2) der *xLD Developer* und (3) die *xLD Anwendung*. Eine wesentliche funktionelle Rolle spielen dabei noch (4) der *Loader* und (5) der *Zustandsbeobachter*. Die Zusammenhänge dieser Bausteine werden in *Abb. 1* dargestellt. Dabei sind zusätzlich die wesentlich beinhalteten Funktionen der Einzelkomponenten dargestellt.
Die xPCLabDesk Architektur wurde in der Weise festgelegt, dass das System *Panel* + *Developer* + *Loader* mehrere Prozesse (d.h. Echtzeitanwendungen) ansprechen kann. Dies ist insbesondere in einer HIL-Umgebung mit mehreren zusammenhängenden Steuergeräte vorteilhaft.
In der Datenkommunikation zwischen den xLD Bauteilen wird zwischen den (1) zeitkritischen und (2) ereigniskritischen Operationen unterschieden. Als zeitkritisch wird die Operation bezeichnet, die ohne Verzögerung und direkt in der Echtzeitanwendung am Target Computer ausgeführt werden muss. Eine solche typische Funktion ist beispielsweise die Anzeige der Ausführungszeit der Echtzeitanwendung oder das Starten bzw. Stoppen der Echtzeitsimulation. Eine ereigniskritische Operation wird durch ein Systemereignis bedingt. Deren Ausführung wird von dem sog. *Loader* Operationspuffer verwaltet, wobei die Ausführungskriterien nach einer bestimmten fehlertoleranten Hierarchie- und Prioritätsstruktur zugeordnet sind.
Das Monitoring des Systemszustands wird durch einen *Zustandsbeobachter* realisiert, der in einem bestimmten Zeittakt (333 ms) aktualisiert wird und falls nötig die Benachrichtigung von anderen Systembauteilen übernimmt. Zu den Aufgaben des *Zustandsbeobachters* zählen beispielsweise (1) Veränderungserkennung von Zustandsvariablen, (2) Signalisierung vom Aufzeichnungsmodus, (3) Überwachung und automatische Erkennung der Host-Target Verbindung ohne Simulationsunterbrechung, (5) Erkennung der auf dem Target laufenden Echtzeitanwendung, usw. Somit ist es möglich mit *xPCLabDesk* auch Standalone xPC Target Anwendungen (erzeugt mit *xPC Target Embedded Option*) zu bedienen. Schließlich finden die Initialisierungsroutinen von Parameter(n), Signale(n) und Graphen im *Loader* statt.

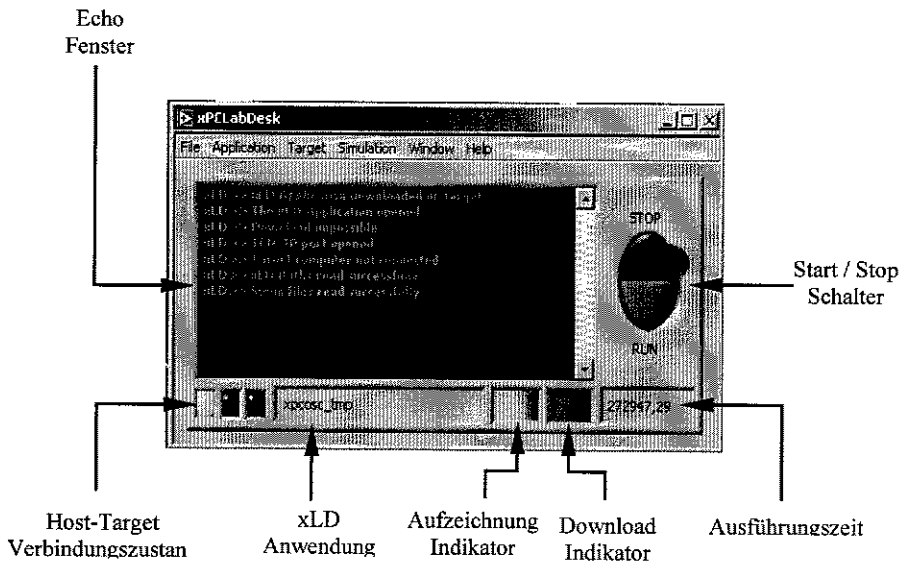

Abb 3: Das xLD Panel

3. xLD Panel

Das xLD Panel entspricht der obersten Steuerungsebene der in xLD implementierten Funktionalitäten (*Abb.2*). Darin sind drei Gruppen der Funktionen enthalten: (1) Echtzeitanwendungsoperationen, (2) Zustandssignalisierung, und (3) Targetoperationen.
In der ersten Gruppe sind folgende Funktionen enthalten: (1) Erstellung einer neuen Anwendung, (2) Dateiverwaltung (Öffnen, Speichern, Schliessen), (3) Laden bzw. Entladen der Anwendung auf dem Target, (4) Starten bzw. Stoppen der Simulation, (5) Aktualisierung der Änderungen im Simulinkmodell, (6) Ändern von der Simulationsabtastzeit und –Stoppzeit, usw.
Die Zustandssignalisierung beinhaltet (1) Erfolgs- und Fehlerbenachrichtigung des Benutzers von xPC Realtime Kernel bzw. von *xPCLabDesk* Routinen über ein Echo-Fenster, (2) Signalisierung der Host-Target Verbindung, (3) Simulationsausführungszeit, (4) Signalisierung der laufenden Echtzeitanwendung usw. (siehe *Abb.2*).
Schließlich beinhaltet xLD Panel Targetoperationen wie (1) Verbinden bzw. Trennen vom Target Rechner, (2) Pingfunktion, (3) Neustart des Target Rechners, usw.

4. xLD Developer

Der *xLD Developer* unterstützt den Benutzer bei der Erstellung und Anpassung von Anwendungen. Bei der Übersetzung eines Simulink-Modells in C-Code mit dem Real-Time Workshop werden unter anderem zwei Dateien erzeugt, die in diesem Modell enthaltenen Parameter bzw. Signale beinhalten. Mit Hilfe dieser Dateien erstellt der *xLD Developer* eine Baumstruktur, aus der die in der Anwendung benötigten Parameter und Signale für die Parameteranpassung, Signalerfassung und Datenaufzeichnung mittels eines *Browsers* ausgewählt werden können (siehe *Abb.4*).

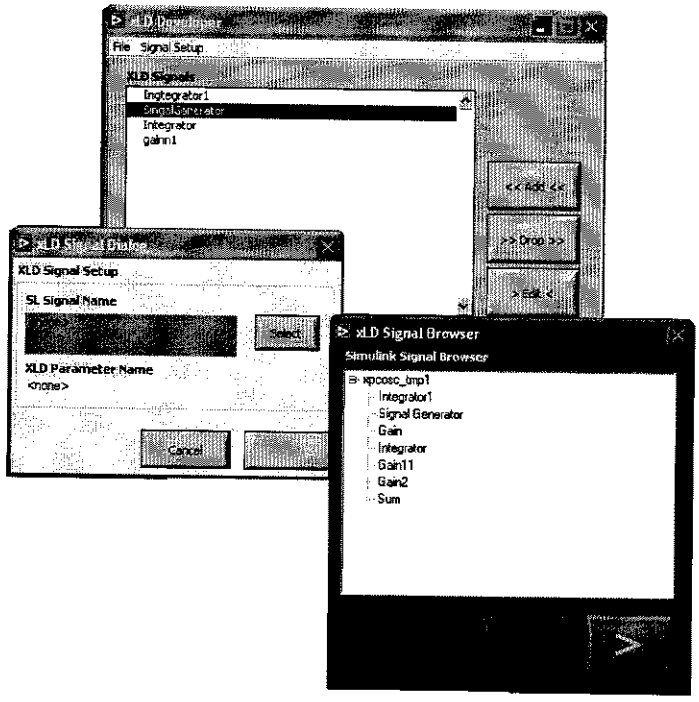

Abb 4: Der Signal-Browser

Zusätzlich kann der Benutzer im *xLD Developer* verschiedene *Scopes* definieren, um Signale zu visualisieren. Neben der Auswahl der gewünschten Signale werden auch Scope-Einstellungen bezüglich Trigger und Anzeige vorgenommen, *Abb.5*. Diese Scope-Einstellungen können auch online während der laufenden Simulation getätigt werden. Damit können sowohl die *Target-Scopes* als auch die *Host-Scopes* unterstützt werden. Die *Target Scopes* laufen auf dem Target Rechner und können an einem an den Target Rechner angeschlossenen Monitor visualisiert werden. Intuitive Mechanismen, wie automatische Erkennung von den in Simulink definierten *Target Scopes* und die darin enthaltenen Signale werden zusätzlich realisiert.

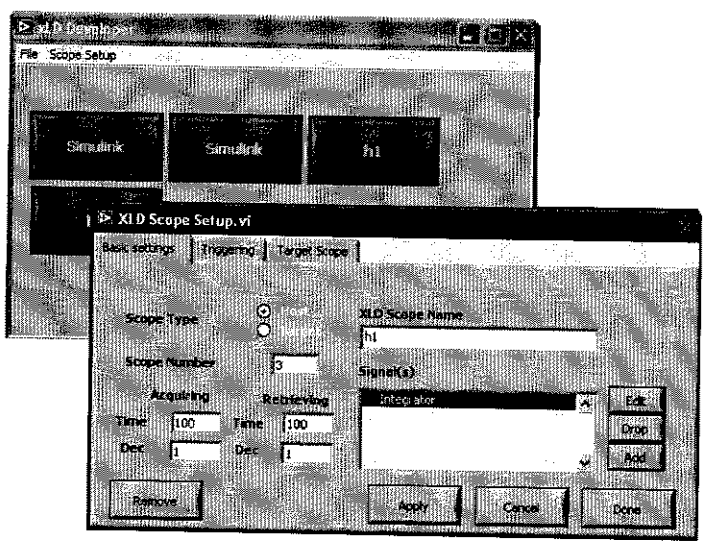

Abb 5: Einstellung der Parameter eines Scopes

Letztlich, dient der xLD Developer über den *xLD Capturer* zur Datenaufzeichnung definierter Simulationsdaten. (siehe Abb.6). Hierzu werden die gewünschten Signale ausgewählt und Angaben zur Aufzeichnungszeit getätigt. Die Daten können anschließend in einem entsprechenden Format (ASCII, MAT oder DIAdem) abgespeichert werden.

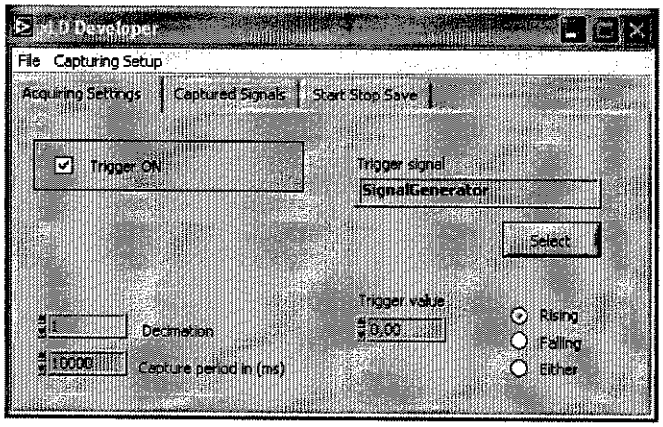

Abb 6: xLD Capturer zur Datenaufzeichnung

5. xLD Anwendung

Eine xLD-Anwendung wird vom Benutzer je nach gestalterischen und funktionellen Anforderungen erstellt. Durch Anordnung geeigneter Bedien- und Anzeigeelemente auf dem Front-Panel entsteht eine für die jeweilige Anwendung angepasste graphische Benutzeroberfläche. Bei der Erstellung einer MMI Anwendung in *xPCLabDesk* kann der Benutzer auf das ganze Spektrum von LabVIEW zurückgreifen.
Das Blockdiagramm der Anwendung ist grundsätzlich in vier Bereiche eingeteilt, die jeweils den graphischen Code für Parameteranpassung, Paramtererfassung, Signalerfassung und Signalvisualisierung enthalten. Allerdings können, um die Benutzerfreundlichkeit der MMI Schnittstelle zu erhöhen oder zusätzliche Funktionen in das Programm zu integrieren, zusätzliche LabVIEW-Codes hinzugefügt werden. So eine Flexibilität der Programmierung wird durch Verwendung spezieller *xLD Driver* ermöglicht. Dadurch werden die im xLD Developer spezifizierten Parameter und Signale nach xPC Target geschrieben beziehungsweise von dort ausgelesen.

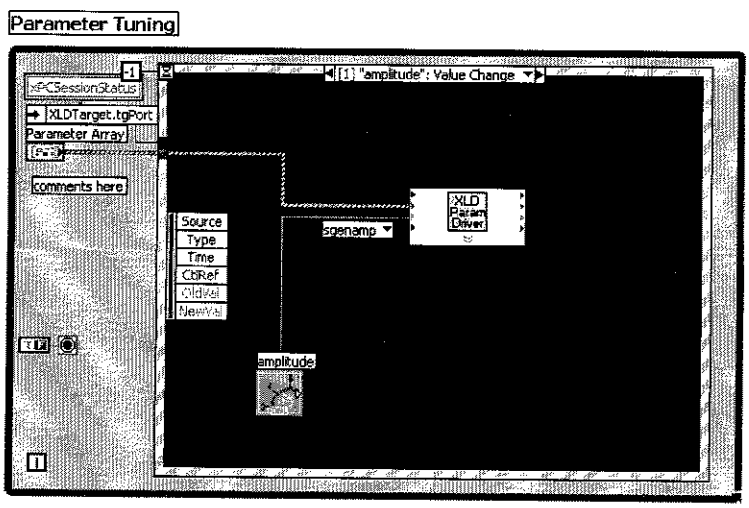

Abb 7: LabVIEW Code zur Paramteranpassung

Der Codebereich für Parameteranpassung (*Abb.7*) besteht aus einer Ereignisstruktur, die den Code, der sich innerhalb dieses Rahmens befindet, nur dann ausführt, wenn ein bestimmtes Ereignis, d.h. eine Änderung des Parameters im FrontPanel, eintritt. Man beachte, dass sowohl die Anpaßung von Skalar- als auch von Array-Parameter unterstützt wird.
Die Signalerfassung folgt einem ähnlichen Ablauf. Da die Signalwerte allerdings während der Simulation ständig ausgelesen werden, benötigt der graphische Code hier eine *While*-Schleife. In *Abb.8* beispielsweise wird der Code zur Signalerfassung auf einem Indikator gezeigt. Dabei wird der xLD Treiber eines Signals und eines Signalpakets (gleichzeitige Erfassung von mehreren Signalen) verwendet.

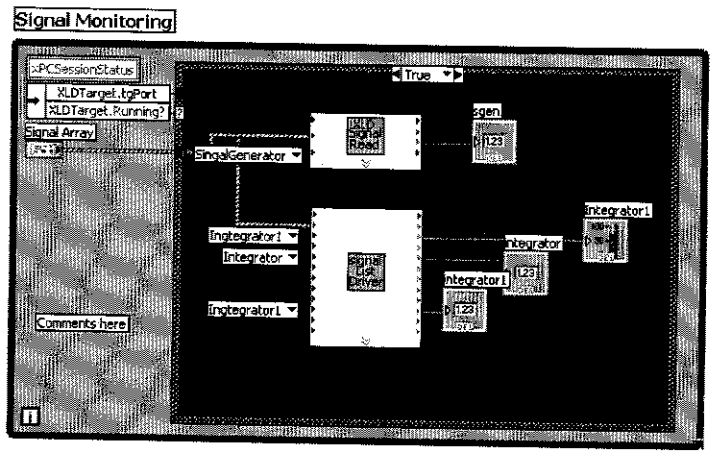

Abb 8: LabVIEW Code zur Signalerfassung

Schließlich, wird in *Abb.9* eine im *xPCLabDesk* entwickelte MMI Schnittstelle gezeigt, die als Cockpit für eine Fahrdynamiksimulationsumgebung (mit ve-DYNA von Fa. Tesis) dient.

6. Zusammenfassung

xPC Target bietet eine flexible Umgebung für RCP und HIL Simulationen auf einer kostengünstigen PC-Hardware. Jedoch fehlt zu diesem Zeitpunkt eine kommerzielle Lösung, die eine komfortable, flexible und echtzeitfähige Mensch-Maschine Interaktion ermöglicht. Das Softwarewerkzeug *xPCLabDesk,* das in Rahmen dieses Beitrags kurz vorgestellt wird, nutzt die leistungsfähige LabVIEW Programmierumgebung um komfortable, flexible und echtzeitfähige Mensch-Maschine Interaktion zu ermöglichen. Damit wird die xPC Target Simulationsumgebung mit LabVIEW vollkommen gekoppelt und dem Benutzer eine mächtige Umgebung für RCP und HIL Simulationen bereitgestellt

8. Literatur

[1] N. Bajcinca, „ xPCLabDesk: Eine Mensch-Maschine-Schnittstelle für Rapid-Control-Prototyping und Hardware-in-the-Loop Simulationen mit xPC Target ", VIP 2004 , München, 2004.

[2] W. Henner, „ Aufbau eines komfortablen Rapid-Control-Prototyping - Simulators ", Diplomarbeit, DLR, 2003.

[3] N. Bajcinca, „A driving Simulator for Steer-by-Wire Control Design", International Automotive Conference, Stuttgart, 2002.

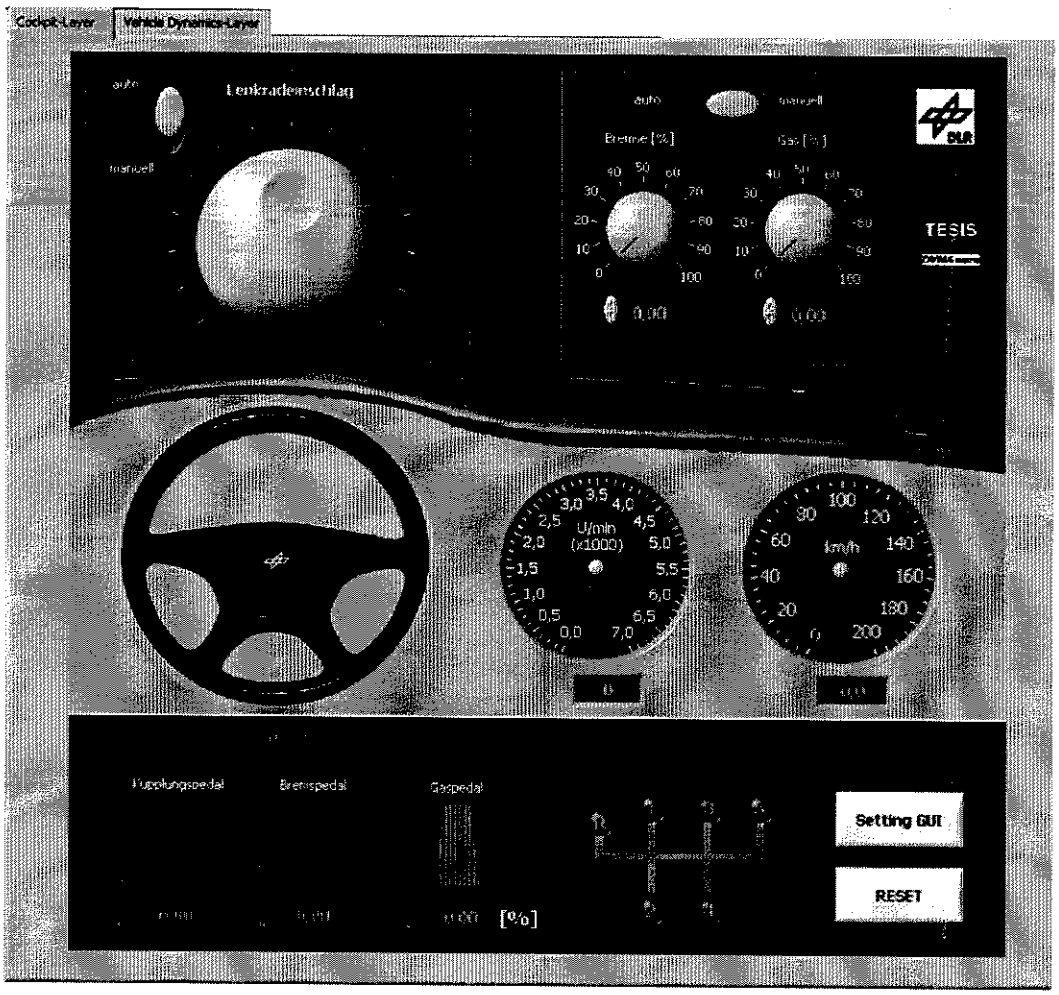

Abb 9: Das Front-Panel eines Fahrzeugcockpits mit ve-DYNA