

## Whitepaper

# LabVIEW Embedded mit Linux auf ARM und FPGA



Für alle, die sich näher über den Einsatz von LabVIEW im Embedded-Bereich und die zugrunde liegende Technologie informieren möchten, gräbt dieses Whitepaper etwas tiefer in den fachlichen Teil. Es bietet zusätzliche Einblicke und Erkenntnisse, die einige von LabVIEW vermutlich nicht erwarten würden. Auf der Bildstrecke begleitet uns ein reales Projekt basierend auf dem NI System-on-Module (SOM). Es handelt sich um das elektronische Herz und Hirn der «Sonnenblume» (siehe oben). Das ist ein kompaktes Solarkraftwerk mit einem 80% Wirkungsgrad der Energienutzung. Der Clou: die Kombination von Solarkonzentration mit Wärmegegewinnung.

## Inhalt

1	Was ist LabVIEW Embedded?.....	2
2	Die Flexibilität von Linux.....	2
3	Meine eigene LabVIEW-Hardware, bitte!.....	3
4	Herstellung im SMT-Maschinenpark in Losgrösse [1].....	5
5	Eigene LabVIEW-Hardware aus dem One-Stop-Shop.....	5
6	Low-Level-Gerätetreiber für den Mikrocontroller entwickeln.....	6
7	Den FPGA mit der Maus programmieren.....	7
8	Die Balance zwischen Mikrocontroller und FPGA.....	7
9	Möglichkeiten und Grenzen.....	8
10	Anhang: Cyber-Security Low-Hanging Fruit.....	9
11	Glossar.....	10

# 1 Was ist LabVIEW Embedded?

LabVIEW wird oft in einem Zug mit Arbeiten im Labor oder beim Testen genannt. LabVIEW Embedded hingegen skaliert weiter, und zwar in den Bereich der Embedded-Systeme hinein. Der Vorteil liegt auf der Hand: LabVIEW auf eigener Hardware für grafisch programmierbare Seriengeräte nutzen! Der Haken: viele komfortable Merkmale, die wir von NI Hardware «ab Stange» schätzen, müssen selbst entwickelt werden. Dazu gehört etwa das Einbinden anwendungsspezifischer Analogwandler und deren Treiberentwicklung. Ausserdem wird an die Anwendungsentwicklung spezielle Anforderungen gestellt, etwa eine skalierbare Architektur und konsequentes Fehlermanagement.

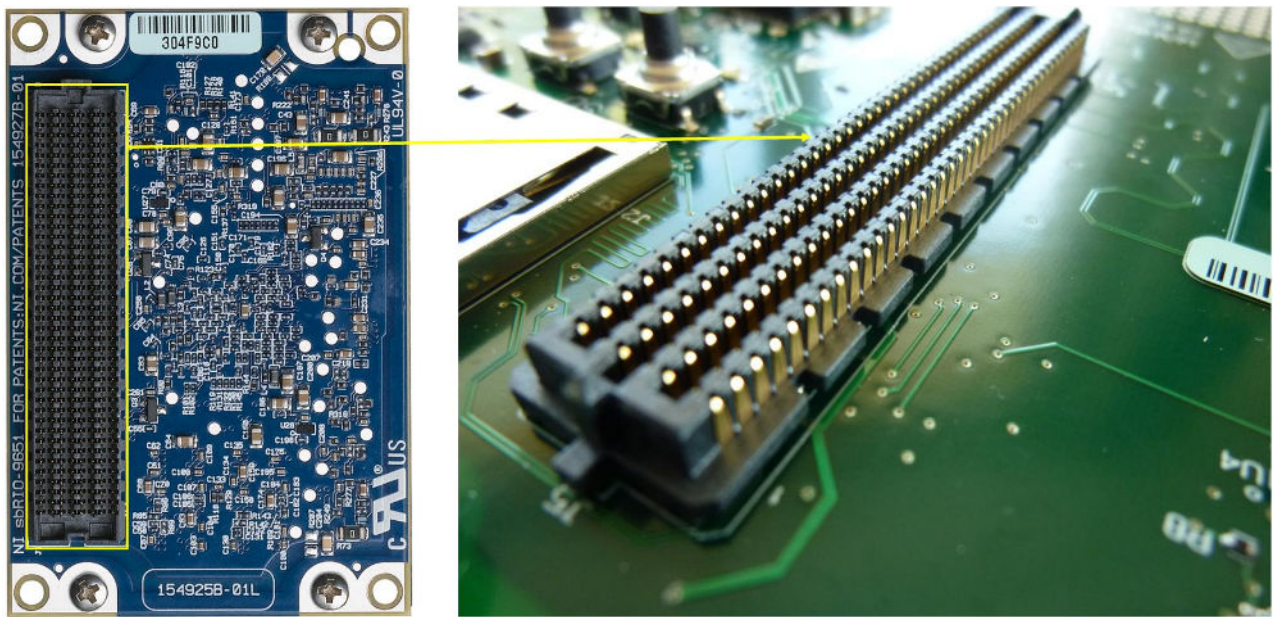


Bild 1 | Der 320-polige Searay-Stecker zwischen dem SOM und dem Baseboard ist ein BGA (Ball Grid Array).

Dieses Whitepaper gibt am Beispiel eines realen Projekts einen Überblick, wie sich der NI-Standard mit kundenspezifischen Bedürfnissen kombinieren lässt. Dieser Standard ist hier gleichbedeutend mit dem NI System-on-Module, sbRIO9651 oder kurz: dem SOM (Bild 1 links). Dieses Modul mit dem Dual-Core ARM-Cortex A9 und Artix-7 (7020) FPGA ist wieder langzeitverfügbar. Beschrieben wird der Zweiboard-Ansatz mit diesem universellen, flexiblen SOM, eingesteckt in ein Baseboard mit I/O nach eigener Wahl (Bild 1 rechts).

## 2 Die Flexibilität von Linux

«LabVIEW Embedded» läuft auf Linux. Dank dieses weit verbreiteten Betriebssystems und des zugehörigen Ökosystems eröffnen sich der Entwicklergemeinde neue Möglichkeiten. Auf dem SOM installiert ist die für den Embedded-Bereich optimierte und schlanke SUMO-Distribution mit

einem Repository auf den Servern von NI und dem Linux Kernel 4.14. Der LabVIEW-Code wird entsprechend dem Linux-Standard auf das Betriebssystem abgebildet. Dort lässt sich das Linux-Ökosystem mit den üblichen Funktionen auch in LabVIEW nutzen. Die grafische Umgebung erhält etwa über ein VI direkten Zugriff auf die «command-line», womit sich direkt Systembefehle ausführen lassen. Auch besteht die Möglichkeit, über das C-API (Application Programming Interface) auf Betriebssystem-Bibliotheken zuzugreifen. Darüber hinaus haben versierte Programmierer sogar die Möglichkeit, den Kernel individuell zu konfigurieren. Über eine der Shells lässt sich das System mit externem Terminal wie PuTTY einfach verwalten (Bild 6). Noch einfacher geht es mit dem webbasierten, grafischen Konfigurationstool WebDAV.

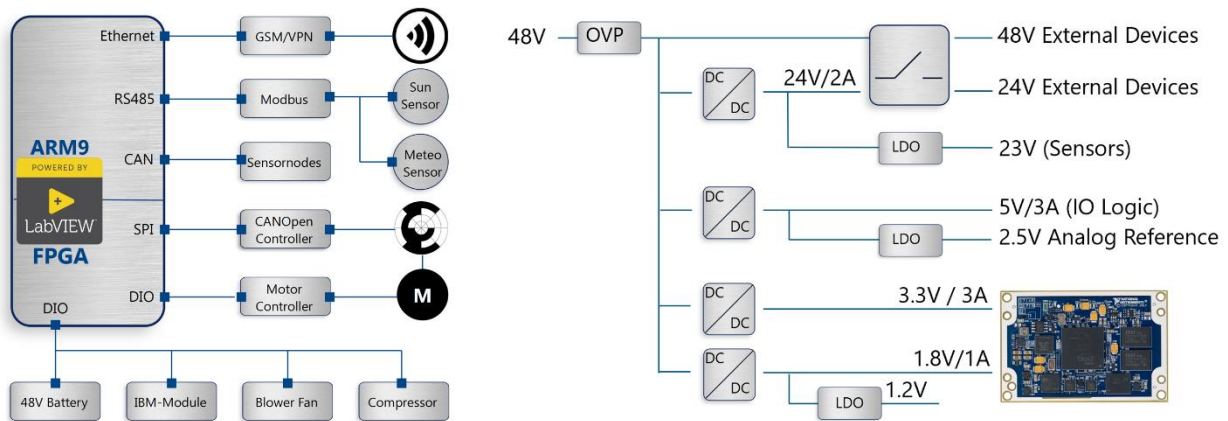
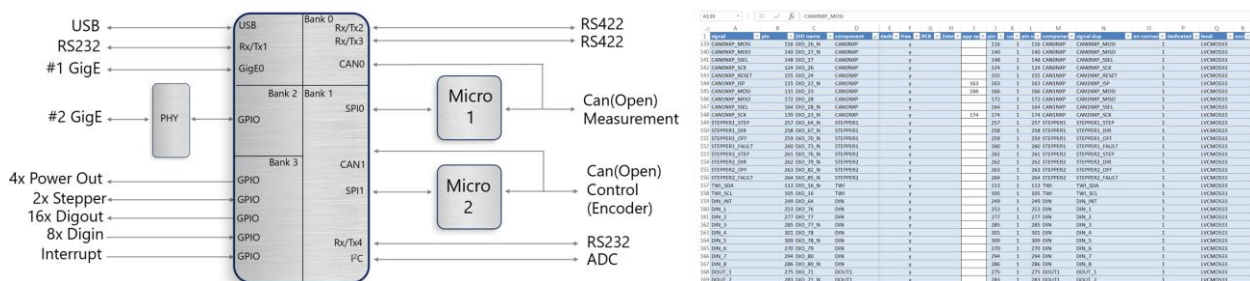


Bild 2 | Als erstes wird das Design des Baseboards entworfen. Links sind die Funktionen, rechts das Powerkonzept.

### 3 Meine eigene LabVIEW-Hardware, bitte!

Nahezu jeder am Markt verfügbare I/O-Baustein lässt sich an das SOM anbinden und mit LabVIEW ansteuern (Bild 2 links). Die Möglichkeiten sind vielfältig: digitaler I/O (memory mapped), synchrone (SPI, I<sup>2</sup>C) und asynchrone (UART) serielle Schnittstellen oder parallele Highspeed-Bussysteme. Typische Beispiele solcher Bausteine sind Analogwandler.



Der Vorteil: die Hardware lässt sich in Form und Funktion an jede beliebige Aufgabenstellung anpassen. Diese Flexibilität hat ihren Preis. Hier kommt nämlich der in der Embedded-Branche beliebte Zweiboard-Ansatz zum Zug. Es ist immer zuerst Hardware in Form eines Baseboards zu entwickeln, denn das SOM selbst ist nicht funktionsfähig. Das Risiko dabei ist überschaubar, denn die zeitkritischen High-Speed-Schaltungen rund um Mikrocontroller und Speicher befinden sich bereits auf dem eingesteckten SOM.

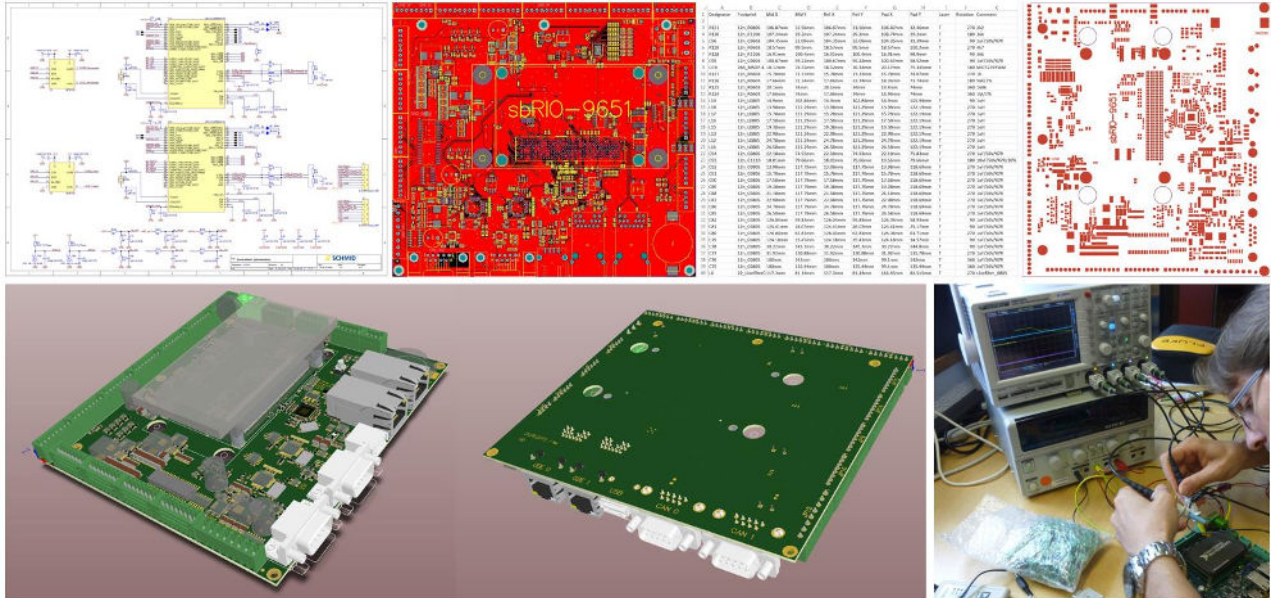


Bild 4 | Die Entwicklungsschritte für Embedded-Systeme: Schema erfassen, Layout erstellen, Produktionsdaten erzeugen (Pick & Place, Schablonendaten), Inbetriebnahme nach Produktion

Beim Baseboarddesign ist infolge des dicht gepackten BGA-Steckers trotzdem fortgeschrittenes Entwicklerwissen gefragt. Üblicherweise durchläuft ein Hardware-Entwickler von der Idee bis zum Baseboard folgende Schritte:

1. Analysieren der Anforderungen, Evaluieren der I/O-Bausteine, z.B. A/D-Wandler
2. Design des Baseboards (Bild 2) und zuordnen der SOM-Pins (Bild 3)
3. Die Funktion, Betriebs- und Grenzwerte sowie Timings in den Datenblättern finden
4. Das Schema erstellen und dabei Wissen aus dem Datenblatt und den Application Notes einfließen lassen
5. Die Material- und Stückliste für den Einkauf und die Produktion generieren
6. Baueinformen nach Produktionsvorgaben und IPC-Normen definieren (Footprint)
7. Boardgeometrie erstellen sowie Stecker und Montagebohrungen platzieren
8. Masse- und Speisungskonzept auf die Mehrlagenstruktur abbilden
9. Elektronische Komponenten platzieren und Verbindungen entflechten
10. Layout routen, Ground/Power-Flächen füllen, Signalintegrität prüfen
11. Produktionsdaten für Leiterplattenhersteller und SMT-Linie erstellen.

## 4 Herstellung im SMT-Maschinenpark in Losgrösse [1]

Der 320-polige Searay-Stecker zwischen dem SOM und dem Baseboard ist ein BGA (Ball Grid Array, Bild 1). Die Lötstellen sind von aussen also nicht zugänglich. Deshalb erfolgt die Produktion vorteilhaft bei einem EMS (Electronic Manufacturing Serviceprovider) mit professionell eingerichteten SMT-Maschinenpark (Bild 5). Zu Beginn einer Entwicklung sind die Stückzahlen oft gering, also wählt man vorteilhaft einen Elektronikfertiger, der Prototyping, NPI (New Product Introduction) und die Losgrösse [1] technisch, terminlich und auch wirtschaftlich beherrscht.

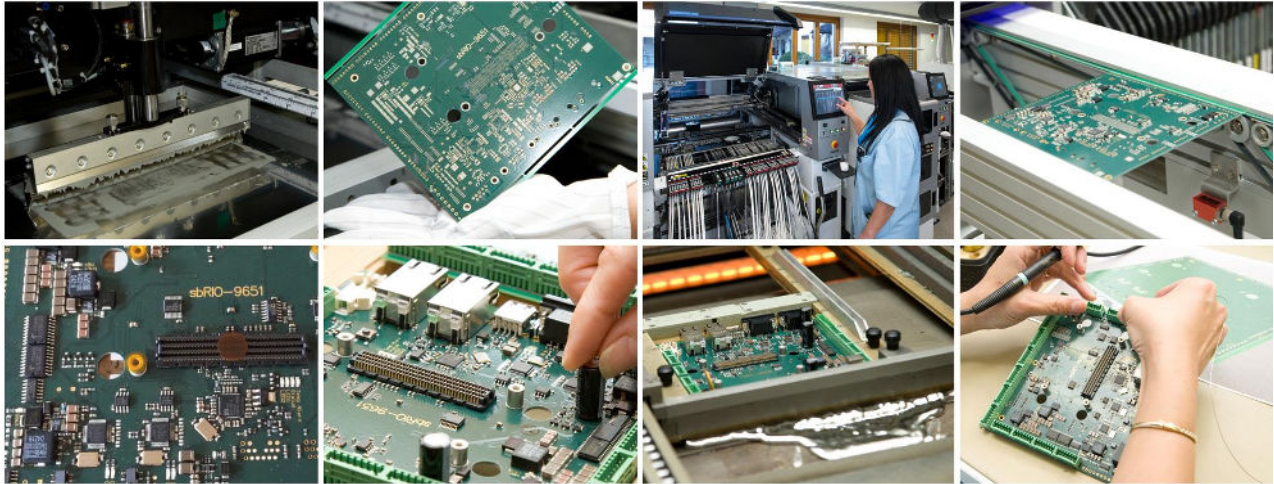


Bild 5 | So wird ein Embedded-System hergestellt: Lotpastendruck, automatische Bestückung, Reflowlöten, Handbestückung, Wellenlöten und Reparaturen

## 5 Eigene LabVIEW-Hardware aus dem One-Stop-Shop

Falls die EDA-Infrastruktur (**E**lectronic **D**esign **A**utomation), das nötige Hardware-, Software- und Betriebssystem-Knowhow oder der SMT-Park fehlen, bietet der NI Partner Schmid Elektronik einen One-Stop-Shop an. Dieser umfasst einen schnellen Entwicklungs- und Produktionsservice für individuelle Baseboards mit kundenspezifischer industrieller Elektronik. Dazu steht ein für Low-Volume/High-Mix optimiertes Produktionsteam und ein High-Tech-Maschinenpark für Prototyping und Serienfertigung bereit, ergänzt mit Test- und EMV-Labor. Softwareseitig gehören Anpassungen des Linux-Kernels ebenso dazu wie das Entwickeln individueller Gerätetreiber und deren Einbinden in die LabVIEW Umgebung. Alles zusammen also ein komfortabler Service für jeden, der LabVIEW auf eigener Hardware betreiben möchte. Ein Kunde drückte es einmal so aus:

*«Wir schätzen das gegenseitige Vertrauen und die professionelle Zusammenarbeit mit unserem Partner Schmid Elektronik, der absolut verlässliche Qualität sowohl in der Entwicklung von Hard- und Software als auch in der Fertigung und Prüfung bietet. Die agile und aufgeschlossene Arbeitsweise ermöglicht flexible, schlanke Prozesse, insbesondere für unsere Prototypen und Kleinserien. Schmid Elektronik ist einer unserer Stützpfiler bei der Anpassung der NI-Plattform an unsere speziellen Bedürfnisse im Bereich der Hochspannungs-Gleichstrom-Übertragung.»*  
(Julian Lange, Siemens Energy)

## 6 Low-Level-Gerätetreiber für den Mikrocontroller entwickeln

Sobald die Hardware vorliegt, geht's an die Entwicklung der Low-Level-Treiber. Damit werden externe I/O-Bausteine angesprochen und mit einem High-Level-VI abstrahiert. Sind die Bausteine an den 160 FPGA-Pins angeschlossen, schreibt man die Treiber grafisch mit LabVIEW auf dem FPGA. Sind sie direkt mit dem Mikrocontroller verbunden, so gibt es unter Linux drei Möglichkeiten:

1. Liegt der Treiber als Executable vor, lässt er sich aus LabVIEW heraus direkt über das Command-Line-VI ausführen. LabVIEW ist in diesem Fall bei Linux als «lvuser» angemeldet (Bild 6, oben links)
2. Liegt der Treiber als C-Source vor, wird das Executable über die Eclipse-IDE gebaut. Die erreichbaren Antwortzeiten liegen infolge des Zugriffs über das Betriebssystem im zweistelligen Millisekundenbereich.
3. Schliesslich kann mit Eclipse eine Bibliothek (Shared Object) erzeugt und von LabVIEW aus über das C-API-VI angesprochen werden, ähnlich wie eine DLL unter Windows. Dank Direktzugriff auf die C-Bibliothek sind Antwortzeiten im zweistelligen Mikrosekundenbereich möglich (Bild 6 unten links).

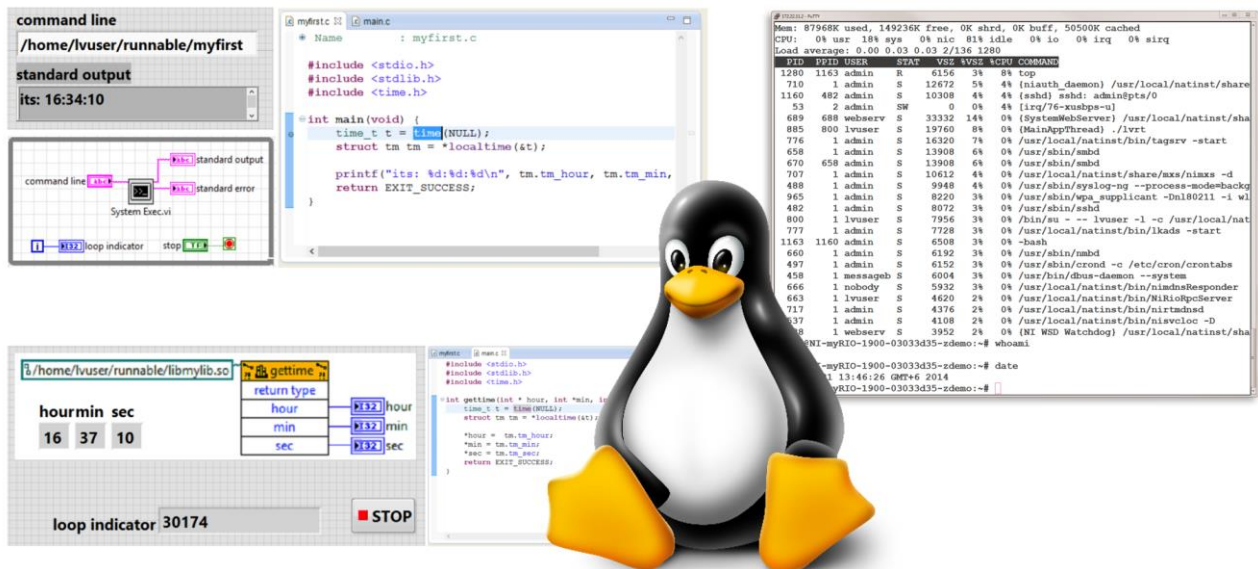


Bild 6 | Schnittstellen zwischen LabVIEW und Linux: oben links die Möglichkeit 1) und unten links die Möglichkeit 3. Rechts das bei Linux übliche externe Terminal PuTTY.

## 7 Den FPGA mit der Maus programmieren

Der Artix-7-FPGA im SOM ist eine softwarekonfigurierbare, parallel arbeitende Hardware. An ihn werden zeitkritische Aufgaben und I/O's delegiert, um den Mikrocontroller zu entlasten. Entscheidende Eckdaten sind dabei die realisierbare Funktionalität und deren Timing. Bisher waren Spezialisten nötig, um FPGAs zu programmieren. Mit LabVIEW erhalten auch Ingenieure ohne diese Erfahrung Zugang zur mächtigen Technologie rekonfigurierbarer Logik (Bild 7). Über intuitive Funktionsblöcke lässt sich ein breites Spektrum analoger, digitaler und serieller Prozesssignale einbinden, verknüpfen und parallel vorverarbeiten, bevor sie in den Mikrocontroller weitergeleitet werden.

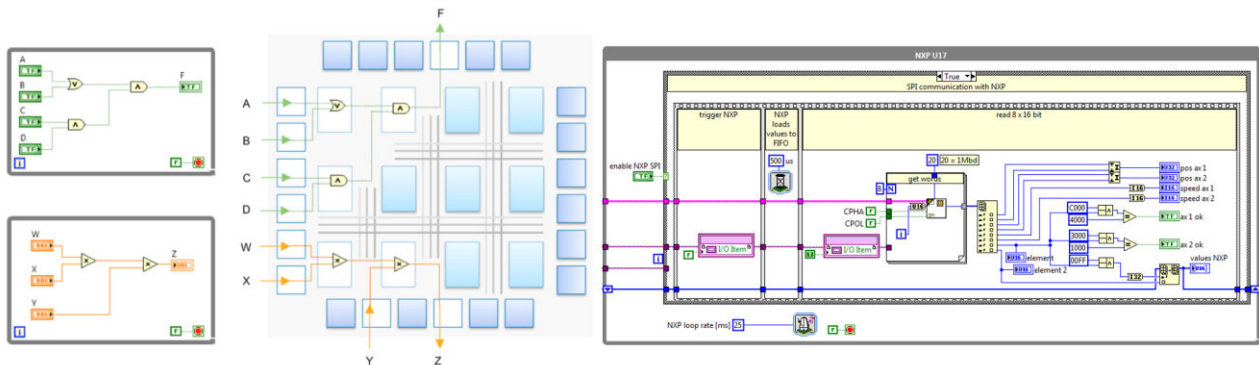


Bild 7 | Entwickeln von Low-Level-Treibern mit LabVIEW FPGA.

Beispiel: die Kommunikation zwischen dem FPGA und einem externen Mikrocontroller, der einen Drehgeber via CANOpen einbinden kann

Die Funktionalität der jeweils abzubildenden Anwendung ist direkt begrenzt durch die zur Verfügung stehende Anzahl an Gattern. Bekannte Kennwerte, welche Operation (Addition, Filter, FFT) wie viele Gatter benötigt, liefern wertvolle Entscheidungsgrundlagen für den maximal möglichen Funktionsumfang. Beim Timing denkt auch der grafisch denkende Anwendungsprogrammierer in «Ticks», dem kleinsten FPGA-Zeitinkrement in der Größenordnung von Nanosekunden. Dabei ist definiert, wie viel Zeit von den logischen und mathematischen Operationen und den I/O-Zugriffen benötigt wird. Daraus ergeben sich Richtwerte für das gesamte System-Timing.

## 8 Die Balance zwischen Mikrocontroller und FPGA

Beim Softwaredesign gilt es, die Embedded-Anwendung sorgfältig auf den Mikrocontroller und den FPGA aufzuteilen. Ersterer ist verantwortlich für die Highlevel-Hauptfunktionen. Low-Level-Details wie Gerätetreiber, zeitkritischer Code, digitale Filter, kombinatorische und sequentielle Logik, Skalierungen, Fixed-Point- und Integer-Arithmetik werden besser auf den FPGA ausgelagert. Das LabVIEW-FPGA-Diagramm wird schlussendlich in VHDL-Code übersetzt, mit den FPGA-Tools in ein Bitfile (Firmware) kompiliert und über LabVIEW Realtime in den FPGA geladen.

## 9 Möglichkeiten und Grenzen

Aufgrund der oft sehr hohen Projektanforderungen unserer Kunden sind wir in den letzten 15 Jahren oft an technische Grenzen gestossen. Im Multitasking-Betrieb robust funktionierende Gerätetreiber erforderten viel Knowhow und waren oft regelrechte Zeitfresser. Dann und wann machte uns der grafische Overhead einen Strich durch die (Performance-) Rechnung und es galt, gezielt Code zu optimieren. Konsequente Fehlerbehandlung stellte sich als Schlüssel heraus. Sieht man von den Grenzen in Tabelle 1 ab, so bietet Schmid Elektronik mit dem Zbrain eine flexible Plattform mit Möglichkeiten, welche den gesamten Bereich der Wertschöpfung abdecken kann und das mit nur einem einzigen Programmierparadigma: Machbarkeitsprüfung, Minimum Viable Products (MVPs), Prototypen, Serienprodukte und Test.

Möglichkeiten	Grenzen (No-Mans-Land)
<ul style="list-style-type: none"><li>• Komplexe Applikationslogik, Multitasking</li><li>• Robuster 24/7-Echtzeitbetrieb</li><li>• ADC-Blocksampling Standard 450kHz, skalierbar bis 40MHz</li><li>• Flexible Datenkommunikation</li><li>• Kundenspezifische I/O-Bausteine</li><li>• Floating-Point-Arithmetik</li><li>• Sicherheitsfunktionen (im FPGA)</li></ul>	<ul style="list-style-type: none"><li>• Kontinuierliche Lowest-Power-Anwendungen in mW. Low-Power für LabVIEW kann mit einer Kombination aus dem SOM mit einer RTC oder einem Low-Power-Coprozessor gelöst werden.</li><li>• LabVIEW auf eigenem Mikrocontroller. Das war früher mit dem C-Generator möglich. Dieser ging 2017 in die End-Of-Life-Phase (EOL).</li><li>• Damit liegen 8Bit-Kleinsttargets außer Reichweite</li><li>• Preissensible Stückzahlen</li></ul>

Tabelle 1 | Möglichkeiten und Grenzen von LabVIEW Embedded

# 10 Anhang: Cyber-Security Low-Hanging Fruit

## Sicherheit vom SOM gegenüber externen Manipulationen und Angriffen

Es kommt darauf an, wie stark man das SOM gegen solche Angriffe absichern möchte. NI bietet, wie unten in den Links zu den Security Best Practices dargestellt wird, verschiedene Stufen der Sicherheit: von «Offen» (also gar nicht sicher, dafür komfortabel zu arbeiten), über «Recommended» bis «Extreme» (komplett abgeriegelt). Da wird auch auf das Thema "Netzwerk" eingegangen. Mit SELinux (Security Enhanced Linux) kann man den Zugriff einschränken. Damit wird der Zugriff von Prozessen auf die minimal notwendigen Ressourcen beschränkt. Je höher die Sicherheitsstufe, desto komplexer die Konfiguration und Verwaltung. Dokumentation auf der NI Website:

- [SELinux – Addressing Access Control Security in LabVIEW RIO Devices - NI](#)
- [NI Linux Real-Time Security User Guide](#)
- [Overview of Best Practices for Security on RIO Systems - NI](#)
  - [Best Practices for Security on RIO Systems: Part 1 Recommended](#)
  - [Best Practices for Security on RIO Systems: Part 2 Optional](#)
  - [Best Practices for Security on RIO Systems: Part 3 Extreme](#)

## Informationen zum Kernel und dem NI RT System Image

- Singleboard-RIOs basierend auf der ARM-Architektur, darunter fällt das NI System-on-Module, wird mit dem Linux Kernel 4.14 und der Sumo Distribution ausgeliefert.
- Das RTOS-Team gibt [hier](#) wichtige Änderungen und Korrekturen über das Open-Source-NILRT-Repository auf GitHub bekannt.
- NI Linux RT Images haben eine Rückwärtskompatibilität von 4 Jahren. Das Image 2022Q4 beispielsweise unterstützt LabVIEW 2022, 2021, 2020 und 2019.
- Ein NI Linux RT Image kann [hier](#) heruntergeladen werden.
- Die NI Hardware und Software-Betriebssystemkompatibilität ist [hier](#) beschrieben.
- [Hier](#) gibt es regelmässige Updates über kritische Sicherheitsfunktionen.
- Firmen wie [neosoft.ca/en/](#) bieten weitere Sicherheitskomponenten an, z.B. einen Firewall
- Siehe auch [NI Linux RT Knowledge Base](#) von Hampel Software Engineering

NI-Hardware- und Software-Betriebssystemkompatibilität

Wählen Sie aus der Dropdown-Liste ein SO-Produkt aus. Eine begrenzte Set von einige Hardwaremodelle von mehreren Treibern unterstützt werden und für jeden Treiber, der unterstützt, ein Beitrag enthalten ist.

**Wichtige Informationen zur Unterstützung älterer Betriebssysteme finden Sie in der Datei „NI Hardware and Software Operating System Compatibility“ im Abschnitt Downloads am Ende dieser Seite.**

68R90-3851 (CompactRIO)

In den folgenden Tabellen sehen Sie die jeweils älteste Version des Treibers, für die Unterstützung verfügbar ist.\*

Installiert auf	NI CompactRIO RIO					Hersteller	
Windows	Windows 11 (64 Bit)	Windows 10 (64 Bit)	Windows Server (2022) (64 Bit)	Windows Server (2019) (64 Bit)	Windows Server (2016) (64 Bit)	Microsoft	
Linux Desktop	RHEL 9	RHEL 8	OpenSUSE 15.5	OpenSUSE 15.4	Ubuntu 22.04	Ubuntu 20.04	Red Hat, SUSE, Canonical
Linux Real-Time	Linux Real-Time					Fluxus	
Linux Real-Time	15.1					Microsemi	

\* OpenSUSE, Redhat, etc. -> [Geben Sie ein](#)  
Before using NI-RTD on Linux, you must begin development on a computer running Windows. Refer to the NI-RTD readme in the form [for the Linux Device Driver Source](#).

### NI Linux RT System Image

Um das Betriebssystem NI Linux Real-Time auf dem meisten Real-Time-Controllern von NI auszuführen, ist das NI Linux RT System Image erforderlich. [Mehr lesen](#)

#### DOWNLOADS

Unterstütztes OS: Windows [Versionshinweise anzeigen](#)

Version: 2024 Q2

Enthaltene Editionen:

2024 Q2	21.0
2024 Q1	20.7
2023 Q4	20.6
2023 Q3	20.5
2023 Q2	20.20.06
2023 Q1	21.0
2022 Q4	20.20.03
2022 Q3	20.19.12
21.0	20.19.12
21.5	20.19.12
21.3	20.19.09

#### NI Linux RT System Image 2024 Q2

Veröffentlichungsdatum: 16.04.24

Enthaltene Versionen: 2024 Q2

> Unterstütztes OS

> Prüfsumme

[HERUNTERLADEN](#)

[OFFLINE INSTALLIEREN](#)

Dateigröße: 5.83 MB

# 11 Glossar

<b>Kapitel 1</b>	Embedded Systeme	Kombination aus Hard- und Software, die innerhalb eines grösseren Systems, in dem sie eingebettet sind, eine Teilfunktion ausführt.
	SOM, sbRIO9651	Das NI <b>System-On-Module</b> in der Grösse einer Visitenkarte.
	Dualcore	Ein Mikrocontroller mit zwei Rechenkernen.
	ARM-Cortex A9	Ein Dualcore 32-Bit Mikrocontroller mit einem ARM v7-Instruktionssatz.
	FPGA	Rekonfigurierbare Logik: <b>Field Programmable Gate Array</b> .
	Zweiboard Ansatz	Hier: ein Scheckkartenmodul enthält nur den Mikrocontroller und seinen Chipsatz und ist in der Regel in Form und Funktion rückwärts- und vorwärts-kompatibel. Es wird in ein kundenspezifisches Baseboard eingesteckt. Das ist ein in der Embedded-Branche beliebter Ansatz, da Mikrocontroller und Speicher meistens schnelllebig sind, im Gegensatz um Anwendungs-I/O.
	Baseboard	Enthält alle anwendungsspezifischen Bausteine und nimmt über einen hochpoligen Stecker, hier ein 320-Pol-SEARAY, das Mikrocontrollermodul auf.
<b>Kapitel 2</b>	Linux	Weit verbreitetes Opensource Betriebssystem.
	Kernel	Die Hauptkomponente eines Betriebssystems, hier Linux.
	Linux Distribution	Eine Auswahl aufeinander abgestimmter Software, Tools, Apps und Services um einen Linux Kernel.
	Repository, Repo	Eine zentrale Ablage, die Entwickler verwenden, um Änderungen am Quellcode einer Anwendungen oder eines Images vorzunehmen.
	Command-Line	Hier ist es ein Linux-Interface, welches Textzeilen entgegennimmt.
	C-API	<b>Application Programming Interface</b> . In diesem Kontext ist es ein spezielles LabVIEW VI, welches über externen C-Code auf Hardware zugreifen kann.
	Shell	Ein Kommandozeilen-Interface zum Linux-Betriebssystem, zB PuTTY.
	PuTTY	Ist eine <b>Secure Shell</b> (SSH), um eine Verbindung zu Linux herzustellen.
	WebDAV	<b>Web-based Distributed Authoring and Versioning</b> : Datenaustausch mit einem Linux-System.
<b>Kapitel 3</b>	I/O-Baustein	Hier: elektronische Bausteine, welche Sensoren und Aktoren einbinden.
	Memory Mapped	Schneller, paralleler Zugriff vom Mikrocontroller auf einen I/O-Baustein.
	Synchron	Hier: Zeitgetaktete serielle Kommunikation, z.B. SPI.
	Asynchron	Hier: Zeitversetzte Kommunikation, Synchronisierung mit Start/Stop-Bit.
	SPI	Serieller synchroner und sehr schneller Bus für I/O-Bausteine.
	I <sup>2</sup> C	TWI (Two-Wire-Interface), ursprünglich I2C: Zweidrahtbus für I/O.
	UART	<b>Universeller, Asynchroner Receiver &amp; Transmitter</b> für asynchrone Datenübertragung
	A/D-Wandler	Analogwandler, genauer : <b>Analog</b> zu <b>Digital</b> Wandler.
	Mikrocontroller	Hier beim NI SOM handelt es sich um einen ARM-Cortex-A9.

	BGA-Stecker	<b>Ball-Grid-Array:</b> eine Gehäuseform, bei der die Anschlüsse auf der Unterseite des Bauteils liegen. Und zwar in Form kleiner Lotperlen, die nebeneinander in einem Raster aus Spalten und Zeilen angeordnet sind. Auf diese Weise sind sehr viele Anschlüsse auf wenig Raum möglich. Allerdings können solche Bauteile nur maschinell gelötet werden.
	Schema	Die grafische Darstellung einer elektronischen Schaltung mit Bauteilen.
	Layout	Geometrisches Anordnen von Komponenten/Bauteile auf einer elektronischen Platine und deren Verbinden gemäss dem Schema.
	Datenblatt	Hier ist es eine Produktbeschreibung mit Eigenschaften, Funktionen, Spezifikationen und Materialien eines elektronischen Bauteils.
	Application Notes	Ausführliche Beschreibung, wie ein im Datenblatt beschriebenes elektronisches Bauteil in einem bestimmten Anwendungsfall eingesetzt wird.
	Stückliste	Eine Liste aller elektronischen Bauteile, die in einer Baugruppe enthalten sind. Es ist die Schnittstellen zwischen Entwicklung und Produktion.
	IPC-Norm, Footprint	IPC erstellt Standards für die Bauelemente-Formen im Layout, damit sie problemlos bei verschiedenen Elektronikfertigern hergestellt werden können.
	Mehrlagenstruktur	Eine Leiterplatte, die aus mehreren Schichten (4,6,8,12, etc) aufgebaut ist.
	Signalintegrität	Wenn sich die Signale auf der Leiterplatte entlang der Leiterbahnen in Spannung und Strom genau so verhalten, wie es gewünscht ist.
	SMT und SMD	Oberflächenmontierte Technologie/Bauteile. <b>SMT: Surface Mounted Technology. SMD: Surface Mounted Devices.</b>
<b>Kapitel 4</b>	SEARAY	Ist hier ein 320-Poliger Stecker zwischen zwei Leiterplatten, der schnelle Signale ermöglicht. Das Raster des BGA (siehe oben) ist 1.27mm.
	EMS	<b>Electronic Manufacturing Services</b> , ein Elektronikfertiger.
	NPI	Neuprodukteinführung (engl. <b>New Product Introduction</b> ). Eine Produktinnovation wird optimal in die Serienproduktion überführt.
	Losgrösse 1	Herstellung von Einzelstücken im One-Piece-Flow. Der Hintergrund ist ein wachsendes Kundenbedürfnis, Produkte individuell zu konfigurieren. Die Rüstkosten werden dank Lean-Prozessen auf ein Minimum reduziert.
	Lotpastendruck	Das ist der erste Schritt in der SMT-Bestückung. Die Lotpaste – eine Mischung aus kleinsten Lotkugeln und Flussmittel – wird durch eine Metallschablone präzise auf die zu bestückende Leiterplatte aufgetragen.
	SMT-Bestückung	Ein Bestückroboter holt sich mit einem Greifer die Bauteile aus einer Rolle, misst sie aus und bestückt sie auf die Platine. Deshalb heisst es Pick & Place. Die Bauteile bleiben bis zum Lötten auf der klebrigen Lotpaste haften.
	Reflowlöten	Die vom Lotpastendruck aufgetragene Lotpaste wird in einem Ofen über ein kontrolliertes Temperaturprofil wieder aufgeschmolzen und verbindet die Lötanschlüsse der Bauteile mit den Pads/Landeflächen der Leiterplatte.
	THT-Bestückung	Heisst « <b>Through-Hole Technology</b> » und kommt zum Einsatz, wenn bedrahtete Bauteile in die Leiterplatte gesteckt und über die Welle gelötet werden.
	Wellenlöten	Eine vorher handbestückte Leiterplatte führt mit einem Förderband durch ein Lotbad. Durch eine wortwörtliche Welle aus Lotzinn wird die gesamte Unterseite der Platine benetzt.

<b>Kapitel 5</b>	EDA	<b>Electronic Design Automation:</b> Software für die Entwicklung von elektronischer Hardware mit Schema und Layout. Teilgebiet des CAD.
	One-Stop-Shop	Ein Kunde erhält mehrere Dienste aus einer Hand. In diesem Fall kann er eine kundenspezifische LabVIEW-Hardware entwickeln und produzieren lassen und erhält dazu auch die Low-Level-Treiber. So kann er sich von der ersten Stunde an auf die LabVIEW-Anwendung konzentrieren.
	Low-Volume/ High-Mix	Hier: die Elektronikfertigung geringer Losgrößen (z.B. 50 Stück) verschiedener Produkte mit mehreren Umrüstaktionen während eines Arbeitstages.
	EMV-Labor	Ein Labor, in dem Geräte auf ihre Störfestigkeit und auf ihre Störaussendung überprüft werden. EMV: <b>Elektro-Magnetische Verträglichkeit</b> .
	Gerätetreiber	Hardwarenahes Ansteuern elektronischer Bausteine, z.B. AD-Wandler.
<b>Kapitel 6</b>	High-Level-VI	Ein Funktionsblock in LabVIEW, der Unterprogramme aufruft.
	FPGA	Rekonfigurierbare Logik: <b>Field Programmable Gate Array</b> .
	C-Source	In der Programmiersprache «C» geschriebener Quellcode.
	Eclipse-IDE	Eine integrierte Entwicklungsumgebung für viele Programmiersprachen, z.B. C.
	Shared-Object	Eine Code-Bibliothek unter Linux (*.so).
	DLL	<b>D</b> ynamic <b>L</b> inked <b>L</b> ibrary: eine Code-Bibliothek unter Windows (*.dll).
<b>Kapitel 7</b>	Artix-7-FPGA	Neben dem Mikrocontroller ein Hauptbestandteil der ZYNQ-7000-Familie.
	Gatter	Logische Schaltungen/Zellen in einem FPGA: etwa 85'000 im SOM.
	FFT	<b>F</b> ast <b>F</b> ourier <b>T</b> ransformation: wandelt Signale von der Zeit- in die Frequenzdomäne um.
	Ticks	Ein Clock-Zyklus im FPGA in der Größenordnung von Nanosekunden.
	Timing	Hier ein zeitliches Abstimmen verschiedener Softwareteile.
<b>Kapitel 8</b>	Digitale Filter	Bestimmte Frequenzen in einem Signal verstärken oder abschwächen.
	Kombinatorische Logik	Der Ausgangswert hängt direkt vom Eingangswert ab.
	Sequentielle Logik	Ausgangswerte hängen auch von gespeicherten Zuständen ab.
	Fixed-Point	Festkomma-Zahl: besteht aus einer festen Anzahl von Ziffern vor und nach dem Komma. Die Position des Kommas ist fest vorgegeben. Vorteil: kann massiv schneller berechnet werden als Floating-Point.
	Floating-Point	Gleitkomma-Zahl: wird aufgespalten in einen Exponenten für die Größenordnung und in eine Mantisse. Dient zur praktischen Schreibweise sehr grosser und sehr kleiner Zahlen.
	VHDL-Code	<b>V</b> ery <b>H</b> igh <b>S</b> peed <b>I</b> ntegrated <b>C</b> ircuits <b>H</b> ardware <b>D</b> escription <b>L</b> anguage: Hardware-Beschreibungssprache für FPGAs.
	Bitfile	Enthält alle Informationen des LabVIEW Blockschaltbildes, um die Logik des FPGA zu konfigurieren.
	Firmware	Software, die den Hardwarekomponenten Maschinenbefehle erteilt.

	LabVIEW Realtime	Eine LabVIEW-Variante, die in Echtzeit bis Mikrosekunde auf einem Mikrocontroller ausgeführt wird.
	LabVIEW FPGA	Eine LabVIEW-Variante, die in Echtzeit bis [ns] auf einem FPGA ausgeführt wird.
<b>Kapitel 9</b>	Multitasking	Ausführung mehrerer, unabhängiger Aufgaben gleichzeitig.
	Overhead	Daten, die nicht zu den Nutzdaten zählen, sondern als Zusatzinformation benötigt werden, z.B. bei der Abstraktion von grafischem Code.
	Fehlerbehandlung	Fehler gezielt abfangen und kontrolliert darauf reagieren.
	Zbrain	Plattform für grafisches Programmieren mit LabVIEW Embedded, kombiniert mit einem Baukasten aus modularer Hardware und skalierbarer Software aus dem Hause des NI-Partners Schmid Elektronik.
	MVPs	<b>M</b> inimum <b>V</b> iable <b>P</b> roducts enthalten nur die nötigsten Funktionen und werden oft benötigt, um eine Idee zu testen.
	24/7	Industrieller Dauerbetrieb: das System läuft rund um die Uhr.
	ADC-Blocksampling	Analogwerte werden in Blöcken erfasst. Innerhalb dieser Blöcke sind hohe Abtastraten möglich. Anschliessend werden diese Blöcke verarbeitet, bevor der nächste Block erfasst wird.
	Low-Power mW	Skalierbarer stromsparender Betrieb von Embedded Systemen.
	RTC	Echtzeituhr ( <b>R</b> eal- <b>T</b> ime- <b>C</b> lock).
	Coprocessor	Ein zweiter Mikrocontroller zusätzlich zum Haupt-Mikrocontroller. In diesem Kontext steuert ein stromsparender Mikrocontroller den viel leistungsfähigeren Haupt-Mikrocontroller. Das System kann je nach Bedarf zwischen Low-Power und Leistung hin- und herschalten.
	C-Generator	In diesem Kontext wird aus dem grafischen LabVIEW-Code C-Code erzeugt, der dann mit üblichen Tools in eine Firmware überführt wird. EOL seit 2017.
	8Bit-Kleinsttargets	Kleinst-Mikroprozessoren, die 8 Bit während einem Takt verarbeiten.
<b>Anhang</b>	Cyber Security	Massnahmen, um Computer gegen böswillige Angriffe zu schützen.
	SELinux	Eine Linux-Kernel-Erweiterung, die den Zugriff auf bestimmte Ressourcen kontrolliert.
	Best Practices	Beste Vorgehensweise, die sich an Erfahrung orientiert.
	ARM-Architektur	<b>A</b> dvanced <b>R</b> ISC <b>M</b> achine: das sind Mikrocontroller, die auf der RISC-Architektur basieren (RISC: <b>R</b> educed <b>I</b> nstruction <b>S</b> et <b>C</b> omputer) und in Embedded-Systemen dominieren. Diese gehen beim Design der Hardware anders vor als die bekanntere Architektur x86.
	Linux Kernel 4.14	Ein Kernel aus dem Jahr 2017 mit Langzeitsupport bis 2024.
	SUMO Distribution	Die NI Linux RT Distribution für das System on Module.
	RTOS	<b>R</b> eal- <b>T</b> ime <b>O</b> perating <b>S</b> ystem: Echtzeit-Betriebssystem.