Informatik aktuell

W. A. Halang
H. Unger (Hrsg.)

Industrie 4.0 und Echtzeit

Echtzeit 2014

Springer Vieweg

Informatik aktuell

Herausgegeben im Auftrag der Gesellschaft für Informatik (GI)

Wolfgang A. Halang
Herwig Unger (Hrsg.)

Industrie 4.0 und Echtzeit

Echtzeit 2014

Fachtagung des gemeinsamen Fachausschusses
Echtzeitsysteme von
Gesellschaft für Informatik e.V. (GI),
VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA) und
Informationstechnischer Gesellschaft im VDE (ITG)
Boppard, 20. und 21. November 2014

GESELLSCHAFT FÜR INFORMATIK E.V.

VDI VDE VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik

ITG INFORMATIONSTECHNISCHE GESELLSCHAFT IM VDE

Springer Vieweg

CR Subject Classification (2001): C3, D.4.7

ISSN 1431-472X

ISBN 978-3-662-45108-3 ISBN 978-3-662-45109-0 (eBook)
DOI 10.1007/978-3-662-45109-0

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist eine Marke von Springer DE.

Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media

www.springer-vieweg.de

Vorwort

„Industrie 4.0“ ist in aller Munde und wird heftig diskutiert. Deshalb hat das Programmkomitee die diesjährige Fachtagung Echtzeit unter das Leitmotiv „Industrie 4.0 und Echtzeit“ gestellt. Aber was bedeutet dieser schillernde Begriff überhaupt? Der Lenkungskreis Plattform Industrie 4.0 gibt sich vollmundig:

> „Der Begriff Industrie 4.0 steht für die vierte industrielle Revolution.“

Ein solcher Anspruch ist sicher vermessen und unseriös, denn „bemerkenswert ist die Tatsache, dass erstmalig eine industrielle Revolution ausgerufen wird, noch bevor sie stattgefunden hat“[1]. Das emotional aufgeladene Konzept Industrie 4.0 wird mit Heilserwartungen verknüpft und in vielfältigster Weise interpretiert, weil es an einer wissenschaftlich exakten Definition mangelt. Das Bundesministerium für Bildung und Forschung (BMBF) umreißt grob, worum es gehen soll[2]:

> „Die realen Abläufe und ihre Steuerung und Optimierung durch virtuelle IT-gestützte Prozesse werden derzeit durch ein technisches Bindeglied verkoppelt: Der Einbau vernetzter, leistungsfähiger eingebetteter Systeme – so genannter Cyber-Physical Systems – in viele Alltagsgegenstände stellt die direkte Verbindung von realer Welt mit intelligenten Steuerungsprozessen im so genannten Internet der Dinge und Dienste her. ... In der Industrie ermöglicht diese verteilte, aber vernetzte Intelligenz bessere Monitoring- und autonome Entscheidungsprozesse, um Unternehmen und ganze Wertschöpfungsnetzwerke in *nahezu Echtzeit* steuern und optimieren zu können. Möglich werden damit individualisierte Produkte zu den Bedingungen einer hoch flexiblen Großserienproduktion.“

Die in diesem Zitat enthaltenen und für den GI/GMA/ITG-Fachausschuss Echtzeitsysteme relevanten Termini sind Echtzeit sowie eingebettete und cyber-physikalische Systeme. Der letztgenannte Begriff ist zwar recht neu, damit bezeichnete Systeme sind aber in der Automatisierungstechnik seit rund vier Jahrzehnten gang und gäbe, und im Echtzeitbetrieb arbeitende, in Messanlagen „eingebettete“ digitale Datenverarbeitungssysteme gibt es sogar bereits seit 70 Jahren[3]. Während das Hauptaugenmerk des Fachausschusses auf Systeme mit harten Echtzeitbedingungen, die darüber hinaus oft auch noch sicherheitskritisch sind, gerichtet ist, stellen im Rahmen von Industrie 4.0 zu entwickelnde Systeme nach der Aussage des BMBF nur weiche Echtzeitanforderungen.

Somit stehen auch aus Sicht der Echtzeitsysteme alle Informations- und Kommunikationstechniken zur Verfügung, die für Industrie 4.0 gebraucht werden. „... Jetzt gilt es (nur), diese Technologien für die Industrie zu erschließen“[1].

---

[1] R. Drath: Industrie 4.0 – eine Einführung. *open automation*, 3/14, 16–21, 2014

[2] `http://www.bmbf.de/de/9072.php`

[3] K. Zuse: Vorwort zu *Constructing Predictable Real Time Systems* von W.A. Halang und A.D. Stoyenko, Boston-Dordrecht-London: Kluwer Academic Publishers 1991

Dass es sich aus der Echtzeitperspektive bei Industrie 4.0 nicht um die vierte industrielle Revolution, sondern einen „alten Hut“ handelt, mag auch erklären, warum nur recht wenige Beiträge zu diesem Leitmotiv eingereicht wurden. Diese beschäftigen sich mit der semantischen Integration von Feldgerätedaten im Rahmen durchgängiger Fabrikvernetzung, der Organisation kollaborativer Fertigung eines Produkts mittels eines Multiagentensystems zur Vernetzung der durchaus unterschiedlichen Steuerungssysteme der beteiligten Maschinen mit dem Ziel, die Losgröße eins zu erreichen, sowie der Genauigkeit der bei der automatischen Umkonfiguration flexibler Produktionssysteme erforderlichen zeitlichen Synchronisation verteilter Steuerungen. Die im Zuge von Industrie 4.0 geplante vollständige Vernetzung wird zu enormen informationellen Sicherheitsproblemen führen. Deshalb ist ein Beitrag ihrer Lösung hinsichtlich Gewährleistung der Übertragungssicherheit in der mobilen Steuerungs- und Überwachungstechnik unter Echtzeitbedingungen und ein anderer dem Schutzaspekt durch Authentisierung und Autorisierung in der Logistik und im Gesundheitswesen gewidmet.

Als aktuelle Anwendungen werden eine als Knoten in Sensornetzen verwendbare Komponente für selbständig ausgeführte Fluoreszenzmessungen sowie eine auf Tablet-PCs laufende Geschwindigkeitsregelung beschrieben, mit der das Echtzeitverhalten des Betriebssystems Android in Verbindung mit Drahtlosübertragung für Einsatzmöglichkeiten in Automobilen untersucht wird.

Eine modulare und erweiterbare Plattform erlaubt echtzeitfähige Simulation verschiedenster Sensoren mit unterschiedlichen digitalen Schnittstellen in Test- und Entwicklungssystemen für Steuergeräte. Um in solchen Umgebungen gemessene Werte verschiedener Signalquellen zeitlich zu synchronisieren, werden neue Ansätze vorgestellt. Der Simulation unmittelbar zugänglich werden auch im Echtzeitbereich immer häufiger zur Software-Spezifikation eingesetzte UML-Beschreibungen durch automatische Übersetzung in Simulink-Modelle.

Die Programmierumgebung OpenPEARL90 ist gedacht, PEARL insbesondere für Ausbildungszwecke unter Linux zur freien Verfügung zu stellen. Sie nimmt eine Zwischenübersetzung nach C++ vor und ihr Laufzeitsystem bietet umfassende Ein-/Ausgabemöglichkeiten sowie eine offene Treiberschnittstelle. Um die Erstellung hoch verläßlicher und verifizierbarer Software für sicherheitskritische Anwendungen zu unterstützen, wurde eine erweiterte Teilmenge von PEARL definiert, die den Sicherheitsanforderungen der Stufe SIL 3 gemäß IEC 61508 genügt und die in die nächste Version der PEARL-Norm eingehen wird.

Frau Dipl.-Ing. Jutta Düring sei ganz herzlich für die überaus sorgfältige Überarbeitung der eingegangenen Texte sowie die schöne Gestaltung des Tagungsbandes und dem Springer-Verlag für die verbesserten Konditionen zu seiner Publikation gedankt. Ganz besonderer Dank gebührt in diesem Jubiläumsjahr Konrad Zuse, der vor 70 Jahren en passant die Echtzeitsysteme erfunden hat.

Hagen, im August 2014

Wolfgang A. Halang
Herwig Unger

Inhaltsverzeichnis

Zeitsynchronisation

Weiterentwicklung von PEARL

Simulation

Plattformen

Aktuelle Anwendungen

Zeitsynchronisation von Echtzeitmessungen verschiedener Signalquellen für Hardware-in-the-Loop-Testverfahren

Florian Spiteller und Kristian Trenkel

iSyst Intelligente Systeme GmbH
90411 Nürnberg
{florian.spiteller|kristian.trenkel}@isyst.de

Zusammenfassung. Dieser Beitrag stellt verschiedene Synchronisationsmöglichkeiten für Echtzeitmessungen in einem Steuergeräte-Test vor. Zunächst wird das, in der Automobilindustrie etablierte, Hardware-in-the-Loop-Verfahren sowie die für den Test notwendigen Messarten erläutert. Davon abgeleitet werden die durch unterschiedliche Zeitbasen entstehenden Herausforderungen. Vorgestellte und diskutierte Lösungsansätze sind die Messung unterschiedlicher Signalquellen mit einer gemeinsamen Zeitachse, das Einbringen von Statussignalen in vorhandene Messkanäle sowie die Nutzung nicht relevanter, aber messbarer Größen als Synchronisationssignale. Durch ein reales Praxisbeispiel wird sowohl die Umsetzbarkeit gezeigt als auch die gewonnenen Erfahrungen dargelegt.

1 Einleitung

Durch den Einsatz eingebetteter Systeme für sicherheitskritische Anwendungen, beispielsweise im Automobilbereich, ergibt sich die Forderung nach fortschrittlichen Testmethoden. Nur so kann die gewünschte Funktionalität sicher verifiziert werden. Dabei gehört zu einer korrekten Funktion, neben der richtigen Reaktion auf ein Ereignis, auch die rechtzeitige Reaktion auf dieses. Für den Test eines eingebetteten Systems, beispielsweise eines Automobil-Steuergerätes, ist es daher nötig, unter Echtzeitbedingungen sowohl den Eintritt eines Ereignisses (z.B. hohe Beschleunigungskräfte in Folge eines Unfalls), die interne Verarbeitung (Berechnung der Unfallschwere, Entscheidung zum Auslösen des Airbags) und die externe Reaktion (Signal zum Auslösen des Airbags) zu messen. Die Messdaten können anschließend ausgewertet und bewertet werden.

Das gerade in der Automobilindustrie häufig verwendete Hardware-in-the-Loop-Testverfahren (HIL) erlaubt es dabei, die Umgebung des zu testenden Steuergerätes und damit auch beliebige Fahrsituationen zu simulieren. Der Test des Steuergerätes kann so bereits parallel mit der Entwicklung beginnen, da die für eine korrekte Funktion benötigten und noch nicht real vorhandenen Gesamtsystemkomponenten (z.B. Partnersteuergeräte, Sensoren) modelliert und anschließend simuliert werden können. Durch die synthetische Umgebung können die

W. A. Halang, H. Unger (Hrsg.), *Industrie 4.0 und Echtzeit,* Informatik aktuell,
DOI 10.1007/978-3-662-45109-0_1,

Randbedingungen eines Tests beliebig oft exakt reproduziert werden, wodurch sich Änderungen in der Software isoliert prüfen lassen. Das Umgebungsmodell wird graphisch mithilfe von MATLAB®/Simulink® modelliert und anschließend automatisiert in optimierten Code für das verwendete Echtzeitsystem (beispielsweise ein dSPACE Rechner). Wahlweise kann der HIL-Simulator durch reale Komponenten, wie zum Beispiel Sensoren und Aktoren, ergänzt werden. Abbildung 1 zeigt den schematischen Aufbau eines HIL-Testsystems.

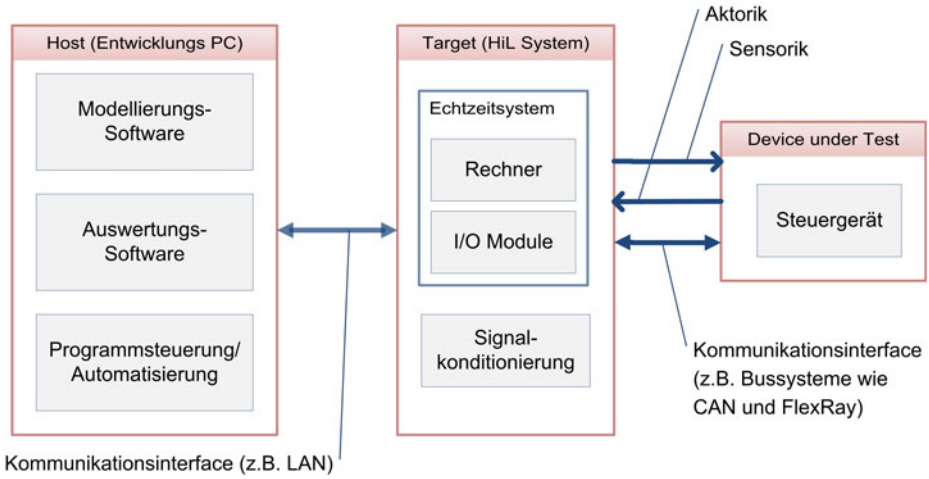

Abb. 1. Schematische Darstellung eines HIL-Testsystems

2 Messen und Testen

Je nach Anwendungsfall und zu testendem Steuergerät ist es nötig, gleichzeitig Signale aus verschiedenen Quellen mit hoher zeitlicher Auflösung zu erfassen. Anhand von vier unterschiedlichen Signalquellen, wie sie auch in einem typischen HIL-Steuergerätetest vorkommen, werden in folgendem Abschnitt die verschiedenen Messmöglichkeiten und ihre jeweiligen Eigenheiten näher erläutert. Auch die Notwendigkeit für eine Echtzeitmessung wird dargestellt.

Aus den Erfahrungen der Praxis und den gegebenen Anforderungen lassen sich Rückschlüsse auf die benötigte Messgenauigkeit ziehen. Übliche Reaktionszeiten liegen zwischen $10\,ms$ und $100\,ms$. Die Messung des Testablaufes sollte mindestens doppelt so schnell erfolgen können.

Bussysteme Moderne Automotive-Steuergeräte kommunizieren über ein oder mehrere Bussysteme, beispielsweise den CAN- oder FlexRay-Bus. Vereinfacht dargestellt erfolgt die Kommunikation dabei über zyklische Sende- und Empfangsbotschaften, denen jeweils eine gewisse Anzahl an Signalen zugeordnet ist. Das

Steuergerät muss also sowohl externe Signale verarbeiten als auch Informationen für andere, an dem selben Bus angeschlossene, Steuergeräte senden.

Aus Sicht eines Softwaretesters interessant ist dabei neben der korrekten Übertragung der Werte auch die benötigte Zeitdauer, gerade im Fall von übertragenen Fehlerwerten. Moderne Softwaretools, beispielsweise CANalyzer®, erlauben es, die Buskommunikation zu überwachen und aufzuzeichnen.

Sensorik Neben den empfangenen Busdaten dienen Werte von direkt an das Steuergerät angeschlossenen Sensoren häufig als weitere Eingangsgröße für den Funktionsalgorithmus. Dies können beispielsweise Sensoren für Temperatur, Beschleunigung oder die Einfedertiefe eines Rades sein. Auch das Betätigen eines Schalters durch den Fahrer (z.B. das Öffnen eines Fensters) zählt zur Sensorik.

Zur Verifikation der korrekten Funktionalität ist die zeitliche Reaktion auf die sich ändernden Sensorsignale zu überprüfen. Die Messung unterscheidet sich dabei je nachdem ob der jeweilige Sensor real vorhanden oder Teil des Simulationsmodelles ist. Für letzteren Fall lassen sich die Sensorwerte durch Parametrisierung des Umgebungsmodells direkt aus der Testumgebung ändern, der Zeitpunkt der Änderung ist somit im Echtzeitmodell bekannt. Reale Sensoren müssen über Hilfsmittel wie z.B. einen Stellmotor oder eine Temperaturkammer stimuliert werden, dies erhöht die Komplexität sowohl im Bezug auf das Vorgeben eines genauen Wertes als auch auf die Ermittlung des exakten Zeitpunktes zu dem der geänderte Wert vorgegeben wurde.

Aktorik Steuergeräte mit angeschlossener Aktorik (z.B. der Motor eines Fensterhebers) regeln diese entsprechend ihres Funktionsalgorithmus unter Berücksichtigung der relevanten Eingangswerte.

Getestet werden muss auch hier die korrekte, d.h. richtige und rechtzeitige Funktionalität. Gemessen wird dabei ein zu stellender Strom oder eine zu stellende Spannung über eigene, im HIL-System verbaute Messkarten. Die Werte stehen anschließend in dem Echtzeitmodell und darüber in der Testumgebung zur Verfügung.

Systeminterne Werte Neben der Vorgabe und Messung von extern zugänglichen Signalen (Blackbox-Test) können auch Steuergeräte-interne Variablen gesetzt und gelesen werden (Whitebox-Test). Der eigentliche Programmcode der Software wird dafür mit einem speziellen Treiber erweitert, um mit Hilfe des XCP-Protokolls [1] auf die internen Variablen sowohl lesend als auch schreibend zugreifen zu können. Dies geschieht mit Hilfe des Tools CANape®. Neben der Kontrolle der internen Abläufe (z.B. durch die Überprüfung von Zustandsvariablen) können auch gezielt spezielle Zustände angeregt werden.

3 Unterschiedliche Zeitbasen

Durch die in Kapitel 1 gezeigte Architektur des HIL-Systems und die in Kapitel 2 dargelegten zu messenden Signalquellen ergeben sich verschiedene, nicht synchrone Zeitbasen. Die parallelen Abläufe lassen sich dabei wie folgt einteilen:

- Ablauf des Testskriptes (bei üblicherweise voll automatisierten Tests)
- Ablauf der Simulation (Umgebungsmodell auf dem Echtzeitrechner)
- Abläufe im Steuergerät (die zu testende Software, oft mehrere parallele Zeitscheiben)
- Ablauf der jeweiligen Messung bzw. Messungen

Abbildung 2 verdeutlicht die in einem, sogar vergleichsweise einfachen Test, vorkommenden, unterschiedlichen zeitlichen Bezugssysteme.

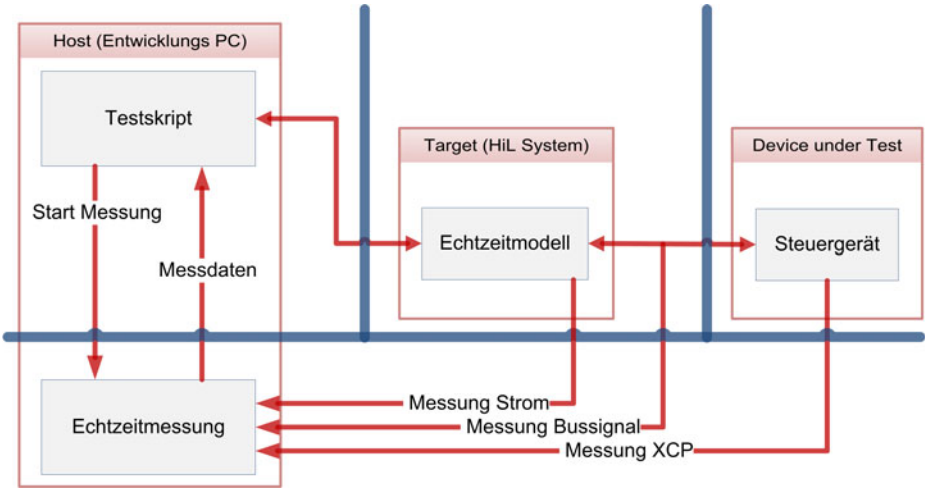

Abb. 2. Darstellung getrennter Bezugssysteme

Der Test von Steuergeräten erfordert häufig die Analyse von Ereignissen, die alle oben genannten Zeitbasen durchlaufen. Ein typischer Testschritt könnte beispielsweise folgenden zeitlichen Ablauf haben, dargestellt am Beispiel Reaktion auf ein am Kommunikationsbus empfangenes Signal:

1. Parametrisierung der Umgebungssimulation (durch das Testskript) für das Senden eines geänderten Signalwertes an das Steuergerät.
2. Das Umgebungsmodell auf dem Echtzeitrechner generiert den Botschaftsinhalt und schreibt diesen in den Sendepuffer der Kommunikationshardware.
3. Im nächsten Sendezyklus (10 ms) wird die neue Botschaft auf dem Bus geschickt.

4. Das Steuergerät empfängt die Botschaft (1 ms Zeitscheibe), wertet den Inhalt aus (5 ms Zeitscheibe) und triggert als Reaktion einen geänderten Sollstrom (2 ms Zeitscheibe) an einem der Aktoren und schaltet zusätzlich in einen bestimmten internen Betriebszustand (10 ms Zeitscheibe).
5. Der geänderte Strom wird von der Analogmesskarte des HIL-Systems gemessen und steht im Echtzeitmodell zur Verfügung.
6. Der Stromwert wird von dem Testskript aus dem Echtzeitmodell gelesen und bewertet.
7. Das Testskript liest über das XCP-Protokoll die interne Zustandsvariable aus.

Das gewählte Beispiel zeigt die Vielzahl an nötigen Schritten des Testablaufes. Die zu verifizierende Anforderung schreibt häufig eine maximale Zeitdauer zwischen Eintritt des Ereignisses (geändertes Signal auf dem Bus) und der gewünschten Reaktion (gestellter Strom am Aktor) vor. Aus Sicht des Testers beziehungsweise des Testskriptes bei einem automatisiertem Test ergeben sich folgende Unsicherheiten in Hinblick auf eine präzise Aussage über den zeitlichen Verlauf von Aktion zu Reaktion:

- Die Zeitdauer ab der Vorgabe des Signalwertes bis zum Sendezeitpunkt auf dem Bus kann nicht exakt bestimmt werden. Dies liegt einerseits in der Natur der zyklischen Übertragung und der nicht vorhandenen Synchronisation zwischen geänderter Vorgabe und Sendezyklus. Weitere mögliche Verzögerungen entstehen durch die Kommunikation des Testskriptes mit dem Echtzeitmodell, durch die Verarbeitung innerhalb des Modelles und dem aktuellen Inhalt des Sendepuffers.
- Die kontinuierliche Messung des Stromwertes zur Erkennung einer Änderung verwendet andere Zeitstempel als das Testskript (Messung erfolgt über das Echtzeitmodell).
- Die Messung der internen Zustandsvariable steht nicht in direktem Bezug zum Eintritt des Ereignisses (geänderter Signalwert auf dem Bus). Es lässt sich zwar einfach eine Aussage darüber treffen, ob die Variable den richtigen Wert angenommen hat, nicht aber, ob sie den Wert auch rechtzeitig angenommen hat.

Um einen möglichst präzisen Test zu ermöglichen, müssen zwei Hindernisse überwunden werden. Zum einen ist eine genaue Abschätzung der Zeitverluste durch den Testablauf und die Teststeuerung notwendig. Zur Einordnung der späteren Testergebnisse berechnet man hierfür sowohl den theoretisch minimalen (Best Case) als auch maximalen (Worst Case) Zeitverlust. In die Berechnung gehen dabei auch empirisch ermittelte Werte mit ein. Andererseits ist es notwendig, die Zeitstempel der verschiedenen Abläufe in Bezug zu einander zu stellen, also eine Synchronisation zwischen den einzelnen Zeitbasen herzustellen. Möglichkeiten hierfür sollen in diesem Beitrag aufgezeigt werden.

4 Synchronisationsmechanismen für Echtzeitmessungen

Statt einer Synchronisierung verschiedener, paralleler Messungen wird bisher häufig lediglich der durch die nicht vorhandene gemeinsame Zeitbasis entstehende Fehler abgeschätzt und das Ergebnis dadurch entsprechend ungenau. Im Extremfall kann eine belastbare Aussage nur noch bezüglich der Korrektheit der Signalwerte, nicht aber bezüglich der zeitlichen Vorgaben getroffen werden. In diesem Abschnitt werden drei Lösungsansätze für die in Kapitel 3 näher erläuterte Problemstellung aufgezeigt.

4.1 Messung unterschiedlicher Signalquellen mit einer gemeinsamen Zeitachse

Teilweise ist es möglich, auch Signale aus verschiedenen Quellen in einer Messung zu erfassen und so direkt eine gemeinsame Zeitachse zu erhalten. Geeignete Toolketten erlauben es beispielsweise, gleichzeitig interne Steuergeräte-Variablen (über das XCP Protokoll) und Signale des Restbusses (über CAN oder FlexRay) zu messen. Die Transportschicht der XCP Daten ist dabei ebenfalls eines der verwendeten Bussysteme. Gestartet wird die eigentliche Messung in diesem Fall vor den relevanten Testschritten und läuft parallel zu diesen ab. Ein Vorteil dieser Methode ist die vergleichsweise einfache Umsetzbarkeit, da keine weiteren Hilfsmittel benötigt werden. Abbildung 3 zeigt ein sich änderndes Bussignal (untere Linie) und die Reaktion der intern gemessenen Statusvariable (obere Linie). Die gemeinsame Zeitachse ermöglicht nun eine direkte zeitliche Auswertung der Fehlerreaktion.

Da die eigentliche Synchronisation innerhalb der verwendeten Messsoftware erfolgt, fällt es allerdings schwer, eine genaue Abschätzung über den vorhandenen Restfehler zu treffen. Erfahrungen zeigen, dass eine Genauigkeit von $\pm$ 5 ms erreicht werden kann.

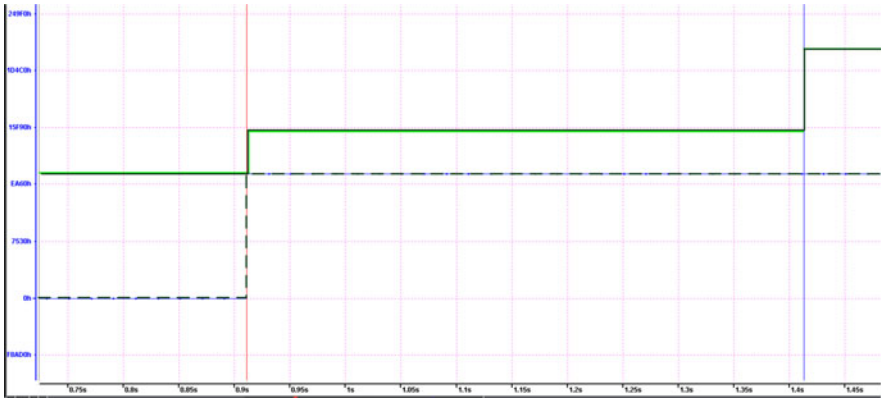

Abb. 3. Messung eines Bus- und XCP Signals