Elektrobit

Software-Entwicklung

AUTOSAR und funktionale Sicherheit – ein Widerspruch?

Während bei sicherheitskritischen Anwendungen Einfachheit im Vordergrund steht, bietet AUTOSAR mit seinen über 6.000 Konfigurationsparametern und weit mehr als 100.000 Zeilen Code einen schier ungeahnten Variantenreichtum. Dieser große Konfigurationsraum macht eine Safety-Analyse und die Durchführung der in hochsicherheitskritischen Anwendungen (ASIL-C, D) geforderten Verifikationsmaßnahmen fast unmöglich. Doch es gibt einen Ausweg aus diesem Dilemma: die Aufteilung der Software in sicherheitsrelevante und nicht-sicherheitsrelevante Teile – verbunden mit einer strikten Trennung beider Bereiche.

Von Dr. Alexander Mattausch

Immer mehr sicherheitskritische Steuergeräte setzen auf softwareimplementierte Sicherheitsmechanismen auf der Grundlage von AUTOSAR. Dabei verlassen sie sich auf Funktionen, die die AUTOSAR-Basissoftware (BSW) bereitstellt. Diese wurden jedoch in der Regel nach reinen Qualitätsmanagement-Richtlinien (QM-Richtlinien) entwickelt. Wie aber lässt sich auf diesem Fundament eine sichere Softwarearchitektur darstellen? Die einzig sinnvolle Lösung, die sich hier anbietet, ist die Aufteilung der gesamten Software in kleine Einheiten. So kommen lediglich bei einem kleinen Teil der gesamten Software Sicherheitsanforderungen zum Tragen, was die Menge des nach dem höchsten Sicherheitslevel zu verifizierenden Codes deutlich verringert. Dies vereinfacht die Entwicklung, reduziert die Komplexität und erhöht die Sicherheit.

Auch der AUTOSAR-Stack selbst lässt sich in sicherheitsrelevante und nicht-sicherheitsrelevante Teile zerlegen. Der Knackpunkt dabei ist die Rückwirkungsfreiheit zwischen den einzelnen Komponenten, in der ISO26262 „Freedom from Interference“ genannt. Es muss sichergestellt sein, dass zwischen den sicherheitskritischen Komponenten und der restlichen Software Wechselwirkungen ausgeschlossen sind beziehungsweise sicher erkannt werden. Dieses Prinzip der strikten Trennung gilt auch dann, wenn verschiedene Softwarekomponenten mit unterschiedlichen ASILs (Automotive Safety Integrity Levels) auf einem Steuergerät integriert werden. Erst wenn Interferenzen ausgeschlossen sind, lassen sich die sicherheitsrelevanten Anteile so klein wie möglich halten und sauber kapseln.

Das Schlüsselkonzept: Freedom from Interference

Die ISO 26262 unterscheidet zwischen drei verschiedenen Arten von Rückwirkungsfreiheit:

- Spatial Freedom from Interference
- Temporal Freedom from Interference
- Exchange of Information

Die „Spatial Freedom from Interference“ bezieht sich auf den Speicher: Die Daten sicherheitskritischer Komponenten müssen vor dem Zugriff anderer Komponenten geschützt sein. Damit sich eine sicherheitskritische Komponente darauf verlassen kann, dass ihre Daten immer korrekt sind, kann sie ihre Daten beispielsweise doppelt und bit-invertiert im Speicher ablegen und bei jedem Zugriff auf Konsistenz prüfen. Dieses Vorgehen erlaubt allerdings lediglich die Erkennung veränderter Daten, bietet aber keine Prävention. Alternativ lassen sich die Daten mithilfe der in modernen Prozessoren verfügbaren „Memory Protection Unit“ (MPU) schützen. Sie erkennt und verhindert einen fehlerhaften Zugriff, die Applikation kann also mithilfe des Betriebssystemkernels auf Fehler reagieren. Für große zu schützende Datenmengen und insbesondere für höhere ASILs ist dies die bevorzugte Variante.

Die sogenannte „Temporal Freedom from Interference“ behandelt die zeitliche Komponente: Es muss sichergestellt sein, dass die sicherheitskritische Software auch die benötigte Rechenzeit erhält und in der erwarteten Reihenfolge ausgeführt wird. In der Regel geschieht das mit Hilfe externer intelligenter Watchdogs: Werden bestimmte Checkpoints nicht in der vorgegebenen Reihenfolge zu den gegebenen Zeitpunkten erreicht, wird dies als Fehler erkannt. Auch die Aktivierung von Applikations-Tasks fällt in diesen Bereich: Ein prioritäts-basiertes Scheduling ist essentiell für die korrekte Ausführung der Tasks in der vorausbestimmten Reihenfolge. Allerdings kann auf diesem Weg das Betriebssystem die sicherheitskritische Applikation beeinflussen, sodass es durch die fehlende Rückwirkungsfreiheit den ASILs der Applikation „erbt“.

Als dritten Bereich, für den Rückwirkungsfreiheit gewährleistet sein muss, nennt die ISO 26262 den Datenaustausch („Exchange of Information“). Sicherheitskritische Daten und Signale müssen so abgesichert sein, dass verfälschte oder fehlende Daten erkannt werden.

Eine wichtige Grundvoraussetzung für all diese Funktionen ist allerdings, dass sich die Software auf den Prozessor verlassen kann. Der Prozessor muss in der Lage sein, Fehler selbsttätig mit der für den ASILs geforderten „Diagnostic Coverage“ zu erkennen. Für niedrigere ASILs wie A oder B reicht hier in der Regel eine Selbsttest-Software, die vom Prozessorhersteller zur Verfügung gestellt und periodisch aufgerufen wird. Bei höheren ASILs greift man in der Regel auf Lockstep-Prozessoren mit ECC-Speicher zurück: Zwei Prozessorkerne führen automatisch alle Berechnungen parallel aus und melden einen Fehler, wenn die Ergebnisse voneinander abweichen. Fehler im Speicher werden bei einem Zugriff durch die falsche ECC-Prüfsumme erkannt. Eine wichtige Grundregel hierbei ist: Erfüllt ein Prozessor die Voraussetzungen für den gewünschten ASIL nicht, lässt sich das durch die Software nicht reparieren.

Der Betriebssystemkernel als zentraler Baustein

Auch die AUTOSAR-Basissoftware lässt sich mittels der Freedom-from-Interference-Mechanismen von der restlichen Software auf dem Steuergerät abkoppeln. Die Basissoftware läuft dann gekapselt in einer eigenen Speicher-Partition, die verschiedenen Software-Komponenten auf der ECU sind dabei mithilfe der MPU voreinander geschützt. Der zentrale Baustein ist dabei der Betriebssystemkernel (siehe Bild 1).

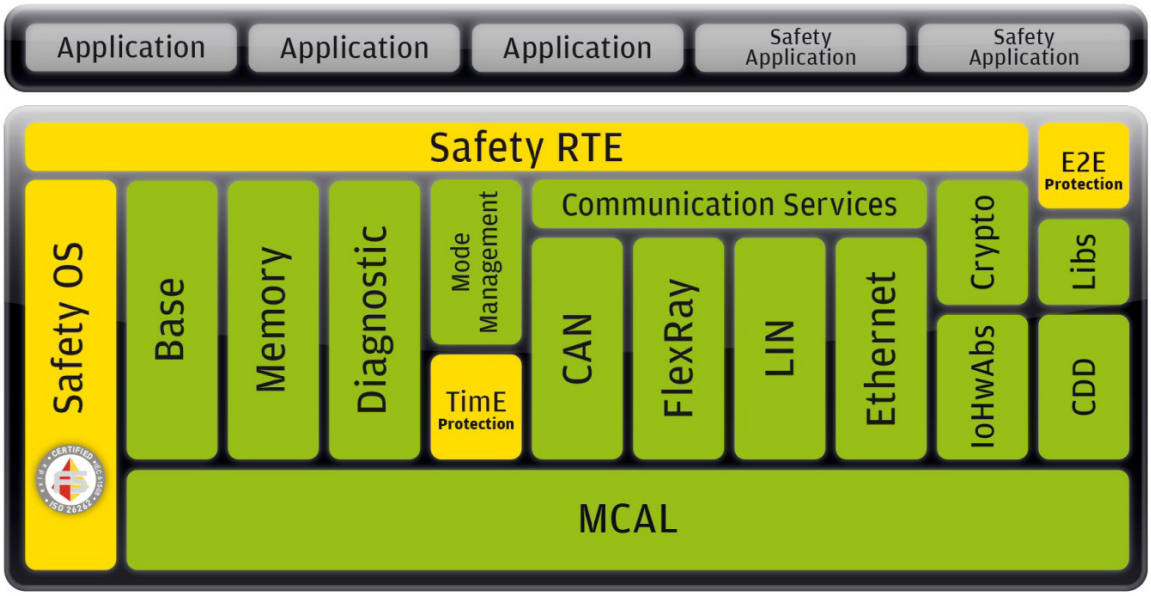

Bild 1. *AUTOSAR-Architektur mit den EB tresos Safety-Produkten (gelb). Durch eine geschickte Auswahl der nach der ISO 26262 entwickelten Module lässt sich der Anteil des sicherheitsrelevanten Codes so klein wie möglich halten.*

Da der Kernel weiß, wann welcher Task zu aktivieren ist, ist dort auch die Verwaltung der MPU am besten angesiedelt. Wenn man die Task- und Interrupt-Verwaltung mit der Programmierung der MPU kombiniert und nach dem höchsten ASIL im System entwickelt, lassen sich Sicherheitsmechanismen direkt auf diesen Kernel aufsetzen. Diese sind gleichzeitig durch die passende Einstellung der MPU vor unautorisiertem Datenzugriff geschützt. Sichere Systeme wie das EB tresos Safety OS bieten die für die genannten Bedingungen notwendige Basisfunktionalität eines Betriebssystems zusammen mit einer flexiblen Speicher-Partitionierbarkeit, die komplett nach ASIL-D entwickelt wurde. Da der Kernel auf ein System-Call-Interface aufsetzt und sich selbst auch mittels Memory Protection schützt, entkoppelt er sich komplett von der Applikation. Er nutzt damit die Unterscheidung von privilegiertem und nicht-privilegiertem Modus mit reduzierten Rechten, wie moderne Prozessoren sie bieten. Dieses Design macht ihn äußerst robust gegen jegliche Fehler in der Applikation.

Doch nicht nur die Kombination aus Task-, Interruptverwaltung und Speicherschutz macht ein sicheres Betriebssystem aus. Komplexe Softwarearchitekturen nutzen geteilte Datenbereiche, die gegen gleichzeitigen Zugriff geschützt werden müssen, sowie Benachrichtigungsmechanismen, um andere, schlafende Teile der Software über ein eingetretenes Ereignis zu informieren.

Insbesondere das AUTOSAR Runtime Environment (RTE) nutzt diese Mechanismen häufig, wenn Software-Komponenten miteinander kommunizieren. Aus diesem Grund bietet das EB tresos Safety OS auch den OSEK-Ressourcen-Mechanismus, der mit Hilfe des Priority-Ceiling-Protokolls eine Deadlock-freie Implementierung eines Locking-Mechanismus darstellt. Die OSEK-Events stellen Funktionen zur Benachrichtigung schlafender Tasks als nach ASIL-D implementierte Funktionalität bereit. Für den Applikationsentwickler bietet das den Vorteil, dass alle für die Funktionalität einer komplexen Softwarearchitektur grundlegenden Mechanismen in einer A-SIL-D-Variante zur Verfügung stehen und eine weitere Absicherung innerhalb der Applikation nicht mehr nötig ist. Auch die oft implizit angenommene Grundfunktionalität eines Betriebssystems, der Taskwechsel und die Interrupt-Behandlung, ist Teil des ASIL-D-Kernels. Der Applikationsentwickler kann sich damit auf das Wesentliche – die Funktionalität in der Anwendung – konzentrieren.

Auch die Basissoftware gewinnt durch diesen Ansatz. Sie lässt sich so ebenfalls als Applikation betrachten, „eingesperrt“ von einem kleinen Betriebssystemkernel. Da moderne Prozessoren in der Regel ermöglichen, auch mit reduzierten Rechten selektiv auf freigegebene Hardware zuzugreifen, spricht nichts dagegen, die BSW mit reduzierten Rechten und aktiviertem Speicherschutz zu betreiben. Die „Freedom from Interference“ zu den anderen Komponenten im System ist damit sichergestellt, gleichzeitig beschränkt sich der sicherheitskritische Anteil der Software auf den Kernel und die Applikation.

Wie lässt sich das in der Praxis umsetzen? Viele Autohersteller liefern Software-Komponenten für herstellerspezifische Funktionen, die auf den AUTOSAR-Stack aufsetzen. So bietet EB beispielsweise ein BMW-spezifisches AUTOSAR-Paket an, in dem die Basissoftware und die BMW-Komponenten im nicht-privilegierten Modus der CPU mit aktiviertem Speicherschutz laufen. Auf dem Leopard-Prozessor von Freescale und ST Microelectronics, auf dem die Referenzimplementierung stattgefunden hat und die jetzt in Projekten mit bis zu ASIL-D in Serie geht, gibt es zusätzlich noch Unterstützung durch das integrierte „Peripheral-Protection-Modul“. Dieses erlaubt, die Peripherie selektiv für den Zugriff im nicht-privilegierten Modus freizuschalten. Zusammen mit dem Speicherschutz lässt sich so eine saubere Trennung von Basissoftware, Applikation und Betriebssystem darstellen.

Ablaufverfolgung und Zeitüberwachung

Ein weiterer wesentlicher Teil der Absicherung einer sicherheitskritischen Software ist die sogenannte zeitliche „Freedom from Interference“. Der Kernel kümmert sich lediglich um die Task-Aktivierung, nicht jedoch um das Einhalten zeitlicher Randbedingungen. Da diese eng mit den Sicherheitsanforderungen der Applikation verknüpft sind, ist deren Überwachung in einer separaten Softwarekomponente oder direkt in der Applikation auch wesentlich besser aufgehoben als im Kernel.

In der Regel gibt es in der Sicherheitsentwicklung drei Grundanforderungen, die von einer Zeitüberwachung erfüllt werden müssen. Als allererstes muss sie erkennen, ob die Software überhaupt noch funktioniert. Dies ist die sogenannte „Alive Supervision“. Eine weitere wichtige Überwachungsaufgabe ist, dass sicherheitskritische Aktionen innerhalb einer festgelegten Zeit-

spanne ausgeführt werden – das sogenannte „Deadline Monitoring“. Zu guter Letzt muss sie eine Abweichung von der vorher festgelegten Ausführungsreihenfolge feststellen können – auch „Control Flow Monitoring“ genannt. Beides kann unter Umständen eine feinere Granularität benötigen, als der Betriebssystemkernel mit seinen Task-Aktivierungen bieten kann, ein weiterer Grund, weshalb eine Auslagerung in ein eigenes Modul angeraten ist.

In der EB tresos Produktfamilie bietet die EB tresos Safety TimE Protection die entsprechende Funktionalität (siehe Bild 2).

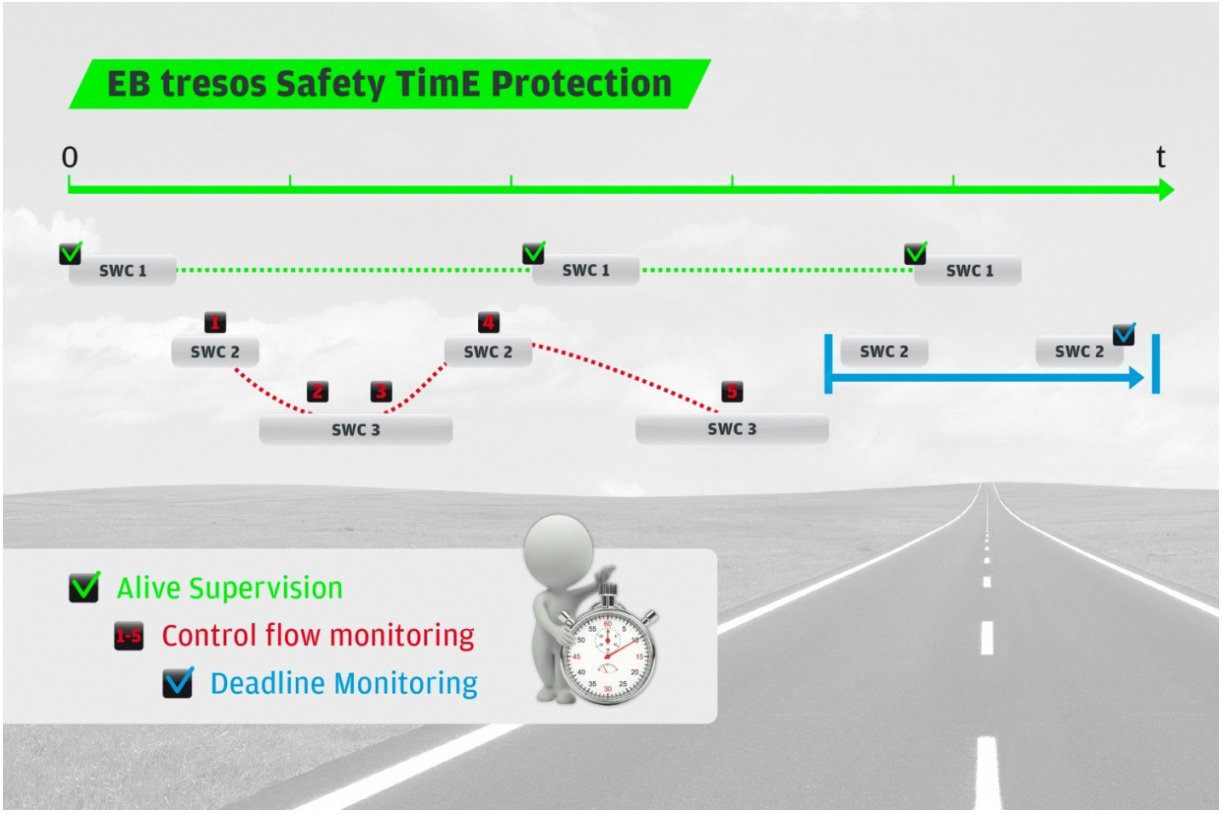

Bild 2. *Die EB tresos Safety TimE Protection bietet im Zusammenspiel mit einem Watchdog „Alive Supervision” durch die Überwachung regelmäßiger Kontrollpunkte, „Control Flow Monitoring“ durch die Überprüfung der Reihenfolge und „Deadline Monitoring“ durch den Abgleich mit einer maximal erlaubten Zeit. Das Bild zeigt dabei das korrekte Verhalten ohne Fehlerfälle.*

Sie implementiert den AUTOSAR Watchdog Stack und stellt mithilfe eines externen Watchdogs sicher, dass im Falle einer Abweichung von der vorgegebenen Aufrufreihenfolge oder bei einer Deadline-Verletzung der Übergang in den sicheren Zustand eingeleitet wird. Als Basis benötigt sie dafür einen Zeitstempel, der ihr entweder vom Applikationsentwickler oder dem Betriebssystemkernel zur Verfügung gestellt wird.

Für Applikationen mit einem kleinen Anteil sicherheitskritischer Daten und auch für niedrigere ASILs bietet die Ablaufkontrolle die Möglichkeit, die Freedom from Interference alleine darzustellen. Werden keine Sicherheitsanforderungen direkt auf dem Betriebssystem allokiert, kann

man mittels Prüfsummen oder redundanter Ablage die Integrität der Daten sicherstellen. Die korrekte Ausführung des Sicherheitsmechanismus kann der Entwickler dann mithilfe von Checkpunkten an geeigneter Stelle und einer Ablaufkontrolle überwachen. Dieser Ansatz stößt jedoch an seine Grenzen, sobald die Komplexität des Sicherheitsmechanismus und die Menge der zu schützenden Daten zunimmt. Des Weiteren steigen mit höheren Sicherheitsanforderungen und damit einhergehenden höheren ASIL-Werten natürlich auch die Anforderungen an die Robustheit des Systems, sodass eine Ablaufkontrolle als alleinige Absicherung meist auch nicht mehr ausreichend ist. So setzen Projekte mit ASIL-C oder D auf eine Kombination aus Safety OS, um die Speicherpartitionierung darzustellen, und einer Ablaufkontrolle zur Sicherstellung der zeitlichen Freedom from Interference.

Runtime-Environment und Kommunikation

In modernen ECUs setzen Applikationen in der Regel nicht mehr direkt auf den Betriebssystemkernel auf, sondern verwenden als AUTOSAR Software Component (SWC) das AUTOSAR Runtime Environment (RTE). Mittels des RTE werden die Basissoftware und die Prozessorhardware abstrahiert, sodass eine leichte Portierbarkeit der Applikation gewährleistet ist. Zwar ist das RTE lediglich eine Abstraktionsschicht, die auf die Kernel- und die BSW-APIs aufsetzt und damit die Funktionalität anderer AUTOSAR-Module der Applikation bereitstellt. Allerdings zeichnet sie für die Kommunikation zwischen den einzelnen Softwarekomponenten verantwortlich, und zwar sowohl innerhalb der ECU als auch über ECU-Grenzen hinweg. Sind die zu übertragenden Daten sicherheitsrelevant, bieten sich für beide Fälle unterschiedliche Absicherungsmöglichkeiten an.

Innerhalb der ECU lässt sich die Absicherung direkt über die RTE vornehmen, was in EBs Produktfamilie die EB tresos Safety RTE übernimmt. Die wichtigste sicherheitskritische Funktion stellt dabei der Datenaustausch dar, da diesen das RTE selbst durchführt und nicht an andere Module delegiert. Die Aktivierung der anderen Softwarekomponente oder den Schutz der gemeinsamen Daten bildet das RTE auf Betriebssystemfunktionalität ab, sodass sich die Absicherung in diesem Fall auf die Überprüfung der korrekten Anwendung der OS-API beschränkt. Die Voraussetzung ist natürlich ein nach dem entsprechenden ASIL entwickeltes Betriebssystem.

ECU-übergreifend gibt es eine Vielzahl von Fehlerquellen, die zu einem Ausfall in der Datenübertragung führen können. Sie liegen oftmals sogar außerhalb der ECU, sodass der Fehler softwareseitig durch eine passende Entwicklung gar nicht verhindert werden kann. Es bleibt lediglich die Fehlererkennung, wofür AUTOSAR das E2E-Modul (End-to-End Protection) spezifiziert hat (Siehe Bild 3). Mithilfe von Prüfsummen und Zählern erkennt dieses Modul verloren gegangene, doppelt gesendete oder fehlerhaft übertragene Datenpakete.

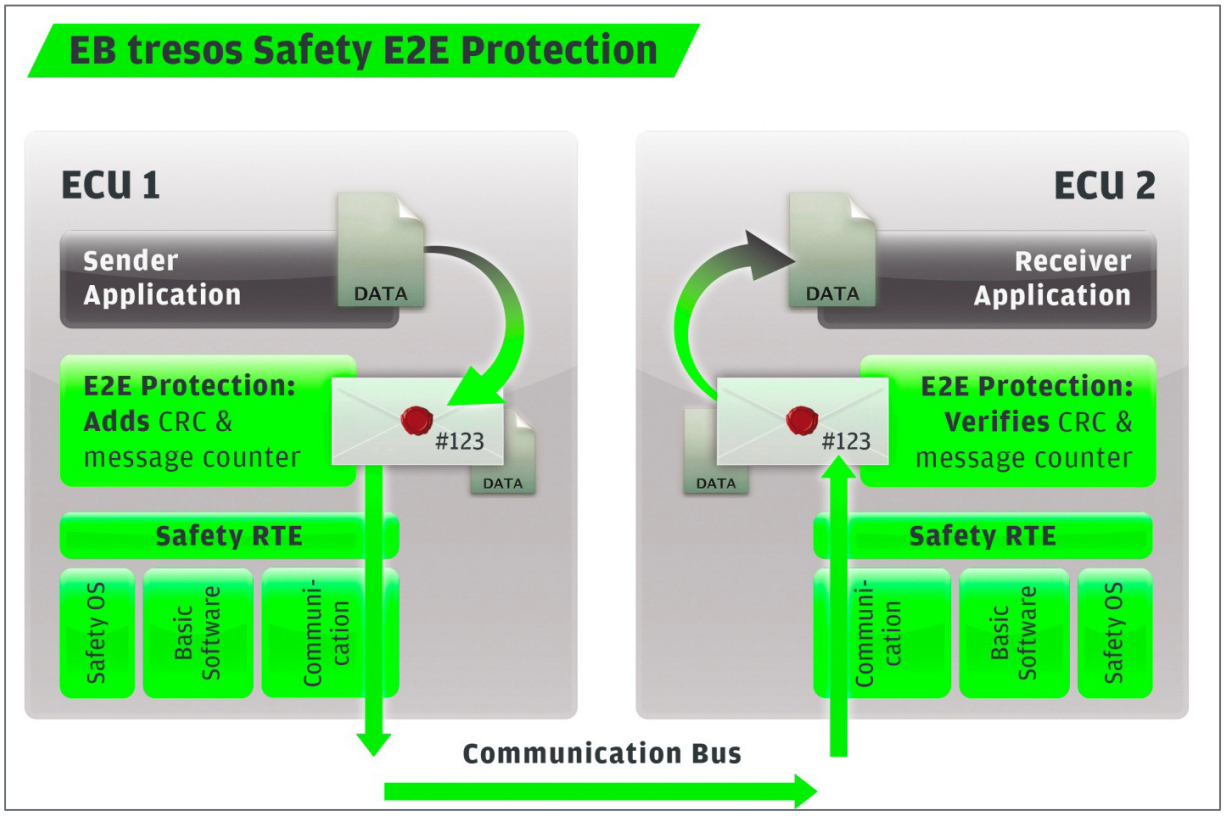

Bild 3. *Sichere Kommunikation lässt sich mittels Prüfsummen („CRC“) und Zählern („message counter“) darstellen. Verfälschte oder fehlende Nachrichten werden damit durch die EB tresos Safety E2E Protection sicher erkannt.*

AUTOSAR und sichere Steuergeräte

Wenn man sich bei AUTOSAR auf die prinzipiell notwendigen Module beschränkt und die hochkomplexe AUTOSAR Basis-Software und Applikationssoftware „einsperrt“, ist es möglich, mit AUTOSAR bis zum höchsten ASIL zu entwickeln und abzusichern. Während der Applikation und dem eigentlichen Sicherheitsmechanismus nach wie vor die Aufgabe obliegt, die eingelesenen Daten zu plausibilisieren und abzusichern, stellen AUTOSAR und insbesondere die EB tresos Safety Produktfamilie Mechanismen zur Funktionsabsicherung bereit. Eine Kompromittierung der Daten lässt sich mithilfe des Betriebssystemkernels verhindern und erkennen, während zeitliches Fehlverhalten durch Alive Supervision, Deadline Monitoring und eine Ablaufkontrolle identifiziert wird. Safety RTE und E2E Protection ergänzen das Paket durch die Absicherung der Kommunikation. Die Beschränkung der Safety-Komponenten auf ihre grundlegenden Funktionalitäten und die Abkapselung der restlichen AUTOSAR-Funktionen sorgt dafür, dass diese Module so klein und einfach wie möglich bleiben. Es ist also trotz der hohen AUTOSAR-Komplexität durchaus möglich, einfache Lösungen zu finden und sichere Steuergeräte zu entwickeln.

Bild 4. *Der „Automotive Safety Integrity Level“ (ASIL) bewertet die Sicherheitsrelevanz von Fehlfunktionen auf einer Skala von A bis D. Nach drei Kriterien: Häufigkeit der jeweiligen Fahrsituation, Beherrschbarkeit der Situation bei einer Fehlfunktion und Schwere der Auswirkung.*

Dr. Alexander Mattausch
studierte Physik an der Universität Erlangen-Nürnberg und hat dort 2005 seine Promotion in theoretischer Festkörperphysik abgeschlossen. Seit 2007 arbeitet er bei Elektrobit Automotive GmbH, wo er als Senior Project Manager für die Betriebssystementwicklung für Automotive-ECUs verantwortlich zeichnet.