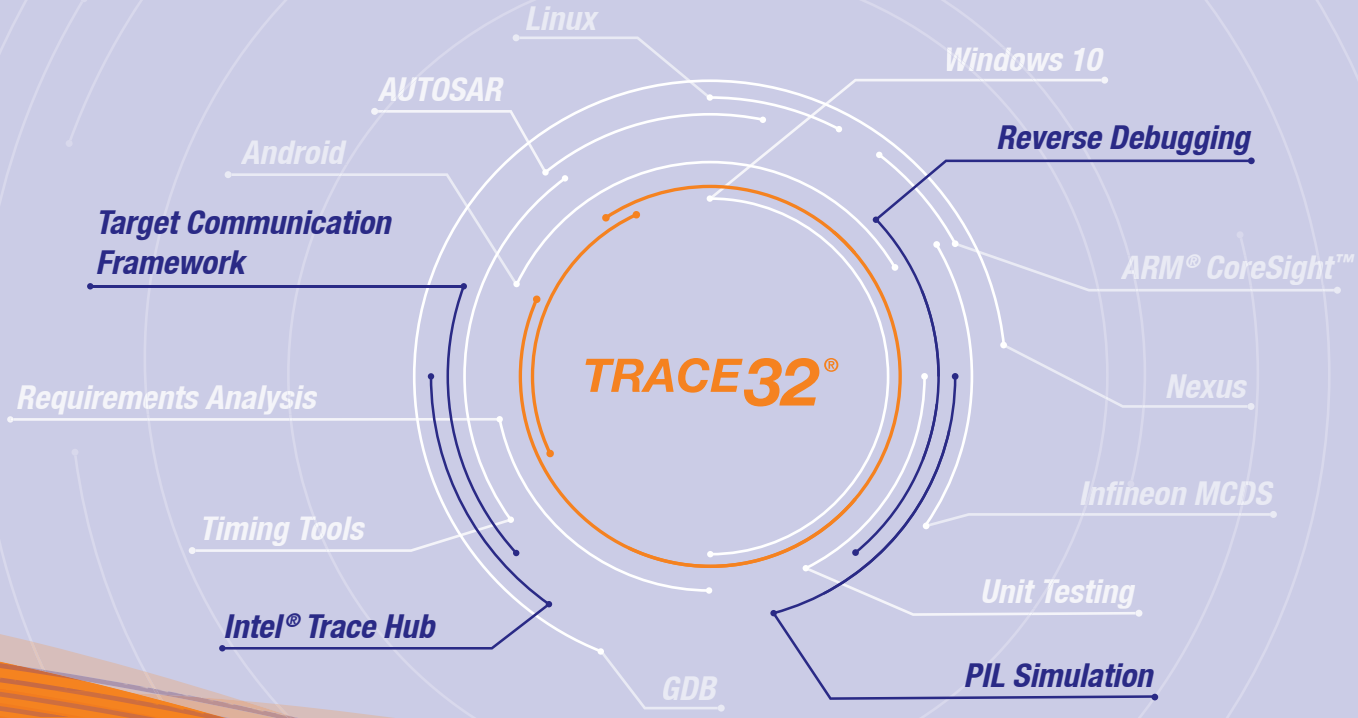
NEWS 2016



INHALT

LAUTERBACH
DEVELOPMENT TOOLS

PIL-Simulation mit TRACE32

Auf der embedded world 2016 wird Lauterbach sein neues Simulink Plug-In für die PIL-Simulation vorstellen. Mit Hilfe dieses Plug-Ins kann die Modellierungsumgebung über einen TRACE32 Debugger direkt mit dem Target kommunizieren.

In Laufe der letzten Jahre haben modellbasierte Methoden in der Software-Entwicklung stetig an Bedeutung gewonnen. Modellbasierte Methoden bieten den großen Vorteil, dass sich das Softwaredesign bereits sehr früh und kontinuierlich verifizieren lässt. Ein wichtiger Verifikationsschritt ist die Processor-in-the-Loop (PIL) Simulation.

PIL-Simulation

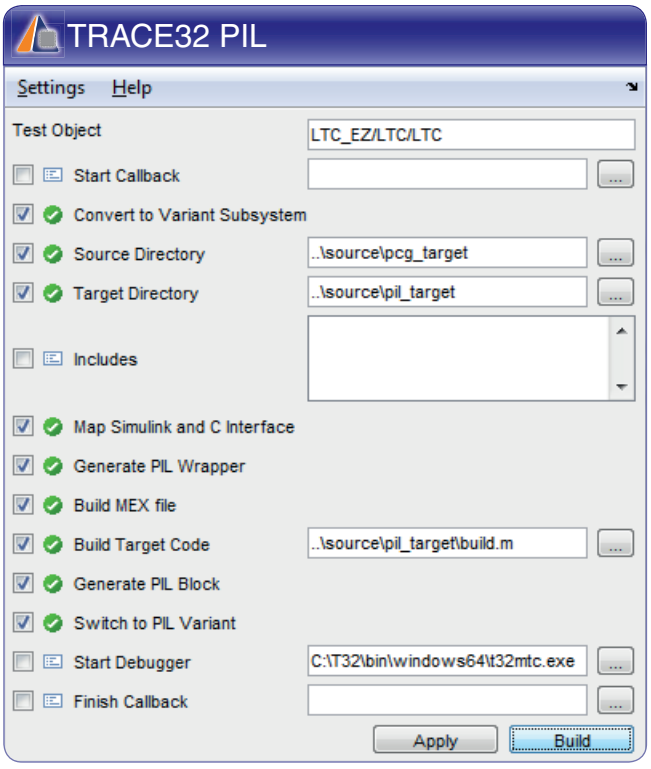
Die Durchführung von PIL-Simulationen soll die korrekte Funktionsweise entwickelter Algorithmen in der Target-Umgebung sicherstellen. Dieser Verifikationsschritt kann auf einem der folgenden Targetsysteme durchgeführt werden:

- Finale Zielhardware
- Evaluation Board
- Virtuelles Target
- Core-Simulator
- *TRACE32 Instruction Set Simulator*

Für die Durchführung der PIL-Simulation wird der zu testende Algorithmus in der Modellierungsumgebung durch einen PIL-Block ersetzt.

TRACE32-PIL Plug-In

Das TRACE32-PIL Plug-In (siehe *TRACE32 PIL* Screenshot) dient dazu, die PIL-Simulation zu konfigurieren. Die wichtigsten Konfigurationsschritte dieses Plug-Ins sollen hier kurz vorgestellt werden.

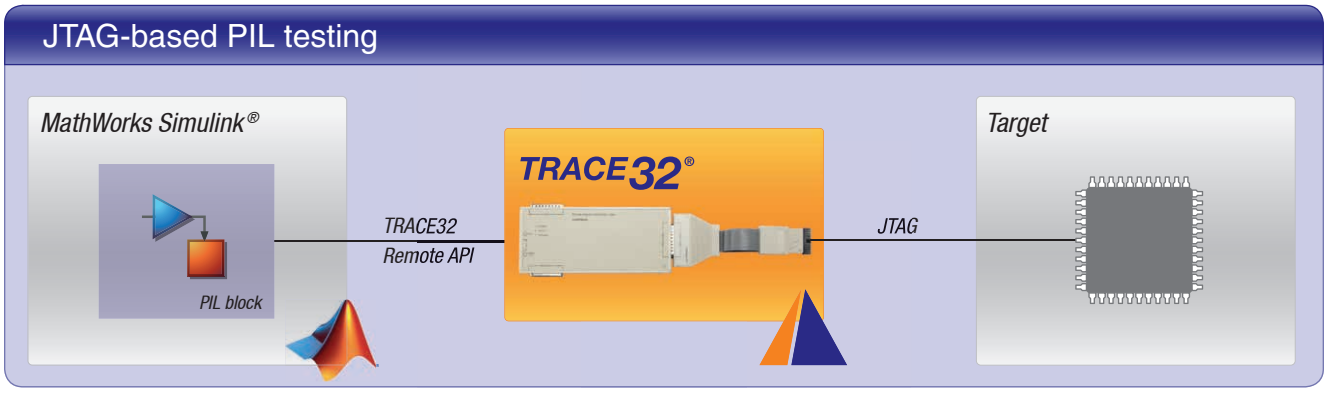


Checkbox: Map Simulink and C Interface

Vor der Generierung des PIL-Blocks muss die Schnittstelle zwischen Simulink und der Target-Applikation konfiguriert werden. Dazu erzeugt die Mapping GUI automatisch einen Vorschlag zur Abbildung der Callbacks der S-Funktion (Benutzer-definierter Block) auf die entsprechenden C-Funktionen. Dieser Vorschlag kann über einen Dialog geprüft und korrigiert werden. Die gleiche Abbildung muss auch zwischen Modell-Parametern und C-Variablen durchgeführt werden.

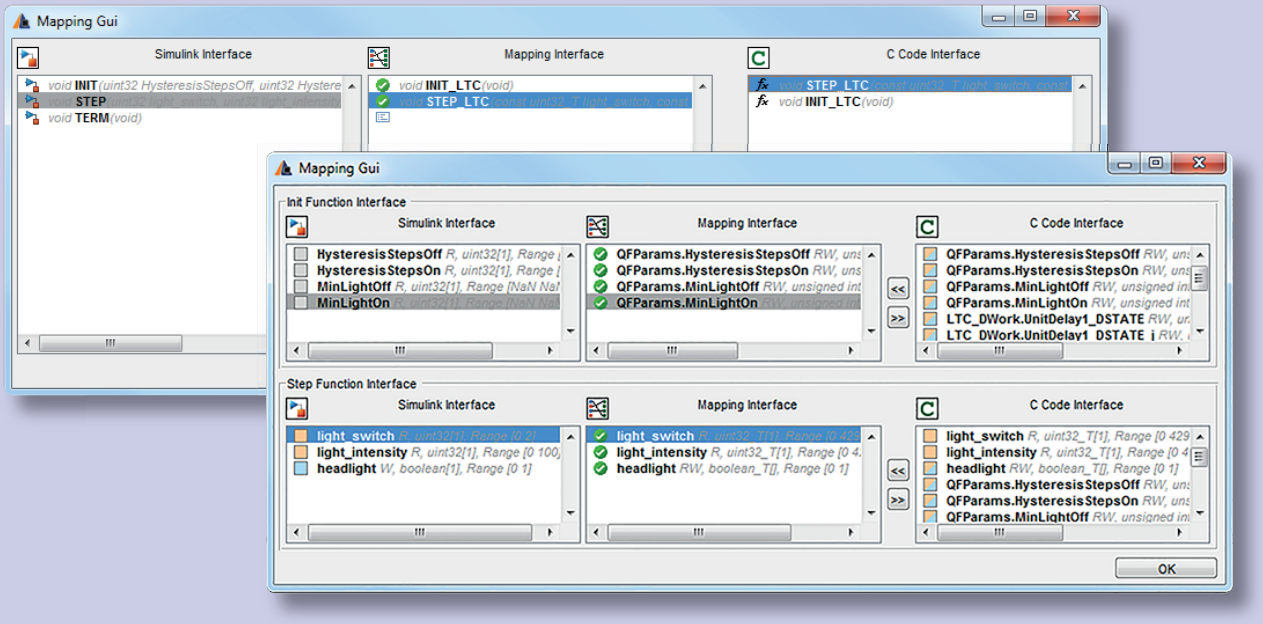
Checkbox: Generate PIL Wrapper

Auf Basis der Abbildungen werden folgende Schnittstellen automatisch erzeugt:



Mapping Simulink and C Interface

Mapping Simulink callbacks ⟶ C functions



Mapping Simulink parameters ⟶ C variables

- Umsetzung der Callbacks auf dem Targetsystem
- Schnittstelle des Modells zur *TRACE32 Remote API*

Anschließend kann der ausführbare Code erzeugt werden.

Checkboxen:
Generate PIL Block/Switch to PIL Variant

Nachdem die Funktionsschnittstelle des Modells zur *TRACE32 Remote API* erzeugt wurde, kann nun der PIL-Block erzeugt und für die Simulation in das Modell eingeblendet werden. Sobald TRACE32 dann gestartet wurde, ist die Konfiguration der PIL-Simulation abgeschlossen und es kann losgehen.

Vorteile

1. Offen für jede Art von Code

Durch die Möglichkeit, die Schnittstelle für die Callbacks dynamisch zu konfigurieren, ist die Lauterbach Lösung offen für alle Codegeneratoren sowie für manuell erzeugten Code.

2. Direkte Anpassung an neue Targetsysteme

TRACE32 unterstützt eine Vielzahl von Prozessorarchitekturen und Compilern. Startup Skripte für die Targetsysteme können direkt vom Entwickler erstellt werden. Es muss nicht auf ein Software-Update gewartet werden. Sobald der Debugger mit dem Target kommunizieren kann, ist die Basis für die PIL-Simulation gelegt.

3. Sofortiges Debuggen

Für den Fall, dass ein Simulationsergebnis vom erwarteten Resultat abweicht, ist das Debuggen der C-Funktion direkt möglich.

4. Lizenzierung

Entwickler, die bereits einen hardware-basierten TRACE32 Debugger bzw. ein virtuelles Target mit einer *TRACE32 Floating License* im Einsatz haben, benötigen lediglich eine *TRACE32 PIL License*. Kunden, die für die PIL-Simulation einen *TRACE32 Instruction Set Simulator* einsetzen wollen, benötigen zusätzlich die neue *TRACE32 Simulator License*.

Umstellung auf neue Basismodule 2015 abgeschlossen

Alle PowerDebug Module verfügen nun über eine USB 3 Schnittstelle. PowerDebug PRO bietet zusätzlich eine Gigabit-Ethernet-Schnittstelle, sowie eine PodBus-Express-Schnittstelle für den Anschluss der TRACE32 Tracemodule PowerTrace PX (neu) sowie PowerTrace II. Selbstverständlich können die bisherigen Basismodule auch für neue Chips/Prozessoren uneingeschränkt weiterverwendet werden.

PowerDebug USB 3

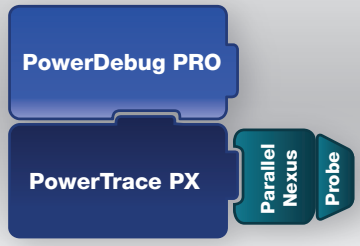
Previous generation PowerDebug USB 2

PowerDebug PRO

Previous generation PowerDebug Ethernet or PowerDebug II

PowerDebug PRO + PowerTrace PX

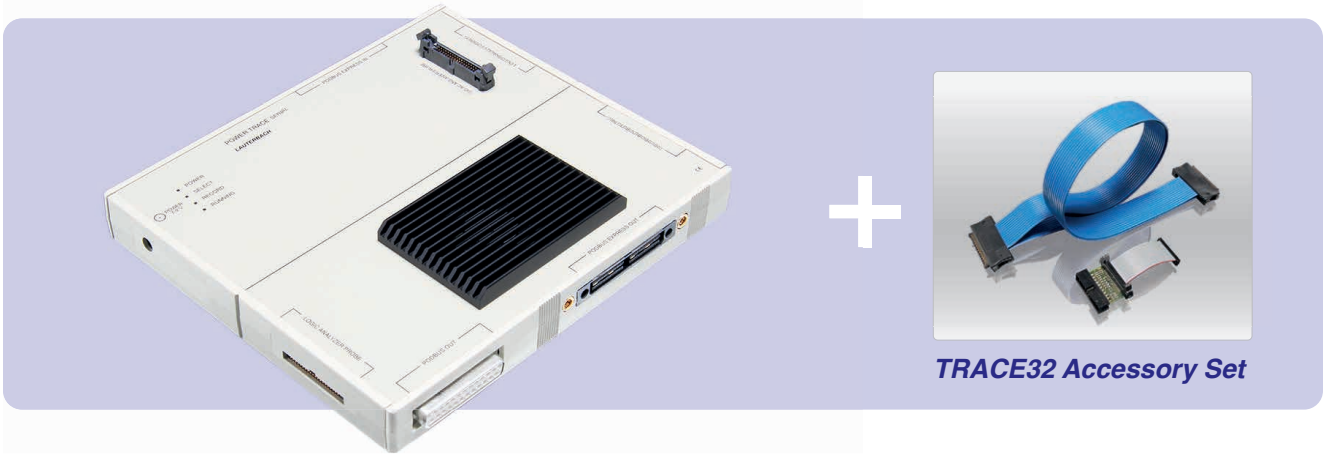
Previous generation PowerTrace Ethernet



PowerDebug PRO + PowerTrace II

Previous generation PowerDebug II + PowerTrace II

PowerTrace Serial



TRACE32 Accessory Set

Lauterbach wird auf der embedded world 2016 sein neues Basismodul TRACE32 PowerTrace Serial vorstellen. Dieses wird dann ab Q2/2016 lieferbar sein.

Seit 2008 unterstützt Lauterbach für verschiedene Prozessorarchitekturen serielle Traceschnittstellen mit einer seriellen Trace-Probe. Diese hat folgende Kenndaten:

- Bis zu 4 RX Kanäle
- Je Kanal 6,25 GBit/s bei bis zu 3 Kanälen
- Je Kanal 3,125 GBit/s bei bis zu 4 Kanälen
- Für Trace-Protokolle, die Aurora nutzen

Für die volle Sichtbarkeit der internen Abläufe in komplexen Multicore-Systemen reichen die aktuellen Bandbreiten serieller Traceschnittstellen nicht immer aus. Deshalb entwerfen erste Prozessorhersteller Schnittstellen mit höheren Datenraten und mehr Kanälen. Zudem wird *PCI Express* als Traceexport-Schnittstelle immer häufiger ins Gespräch gebracht.

Kenndaten

Das TRACE32 Basismodul PowerTrace Serial folgt mit seinen Kenndaten den neuen Anforderungen.

- Bis zu 8 Kanäle
- Bis zu 12,5 GBit/s je Kanal
- Xilinx Aurora sowie andere Protokolle, allen voran *PCI Express*
- 4 GigaByte Tracespeicher

Da diese Kenndaten nur mit einem sehr großen und leistungsfähigen FPGA realisiert werden konnten, wird der PowerTrace Serial als eine All-in-One Lösung angeboten. Das bedeutet, die bisher separat verfügbare serielle Trace-Probe ist nun bereits in den PowerTrace Serial integriert. Um den PowerTrace Serial an das Target anzuschließen, bietet Lauterbach verschiedene *Accessory Sets* an. Diese bestehen in der Regel aus einem passenden Flex-Kabel sowie den notwendigen Adaptern.

Der PowerTrace Serial ist bei der Auslieferung bereits für die Dekodierung von Core-Traceinformationen einer Prozessorarchitektur lizenziert. Die Trace-Dekodierung für weitere Architekturen lässt sich jederzeit einfach nachlizenzieren.

Stecker

Der PowerTrace Serial verfügt über folgende Schnittstellen:

Serial Trace Port 0 (Samtec ERF8, 40-Pins)

- Für Trace-Protokolle, die Aurora nutzen
- 6 RX Kanäle
- Referenzclock 0,325 - 6,25 GHz

Target Debug Port (34-Pin MIPI Connector)

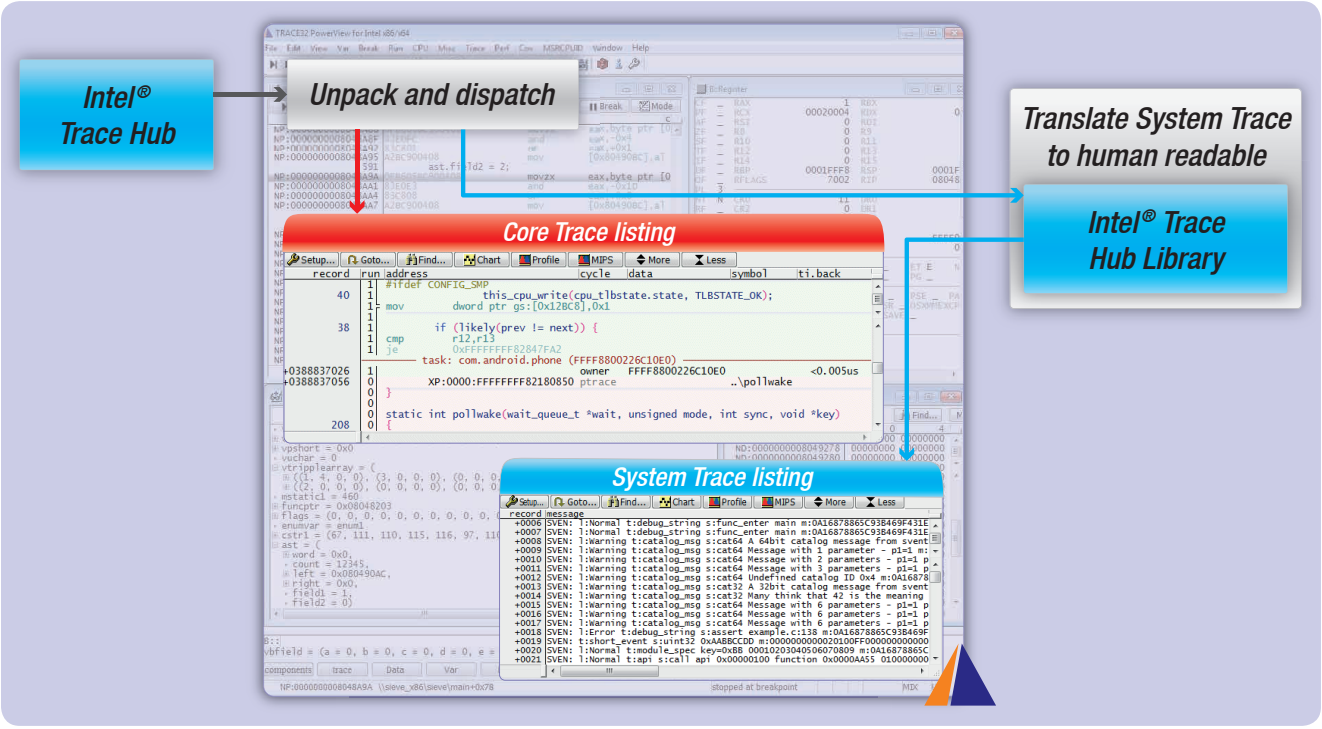
Sind auf den 40-Pin Samtec-Stecker auch Debug-Signale geroutet (JTAG/SWD/cJTAG), kann hier ein TRACE32 Debug Kabel angesteckt werden.

Serial Trace Port 1 (Samtec ERM-ERF, 80-Pins)

- Für alle Protokolle
- 8 RX/TX Kanäle
- Referenzclock 0,325 - 6,25 GHz oder 10 - 500 MHz

Durch die Bereitstellung beider Traceports kann der PowerTrace Serial sowohl in aktuellen als auch in zukünftigen Projekten eingesetzt werden.

Support für Intel® Trace Hub



Ab Mai 2016 wird TRACE32 den Intel® Trace Hub und das dazugehörige Framework unterstützen.

Intel® Trace Hub

Intel® Trace Hub (Intel® TH) ist der Name der Trace-infrastruktur, die Intel® in seine neuen Hardware-Platt-formen einbaut. Diese Traceinfrastruktur ermöglicht es, die *Intel® Processor Trace* Daten der einzelnen Cores sowie System-Trace-Informationen aus den unterschiedlichsten Quellen mit einem korrelierbaren Zeitstempel zu versehen, mittels des MIPI STPv2.1 Protokolls zu einem Tracestream zusammenzufassen und diesen dann an die ausgewählte Tracesenke weiterzuleiten.

Intel® stellt ein Software-Framework bereit, damit Debug- und Trace-Tools wie TRACE32 diese neue Traceinfrastruktur einfach unterstützten können.

Intel® Trace Hub Configuration API

Ziel der *Intel® Trace Hub Configuration API* ist es, die Konfiguration der Traceinfrastruktur durch das Debug-Tool zu vereinfachen. TRACE32 muss die Plattform-spezifische Programmiersequenz nicht kennen, sondern kann eine Konfigurationsanfrage an die *Intel® TH Configuration API* absetzen. Diese liefert dann die passende Programmiersequenz. TRACE32 schreibt diese dann über die JTAG-Schnittstelle in die Kontrollregister.

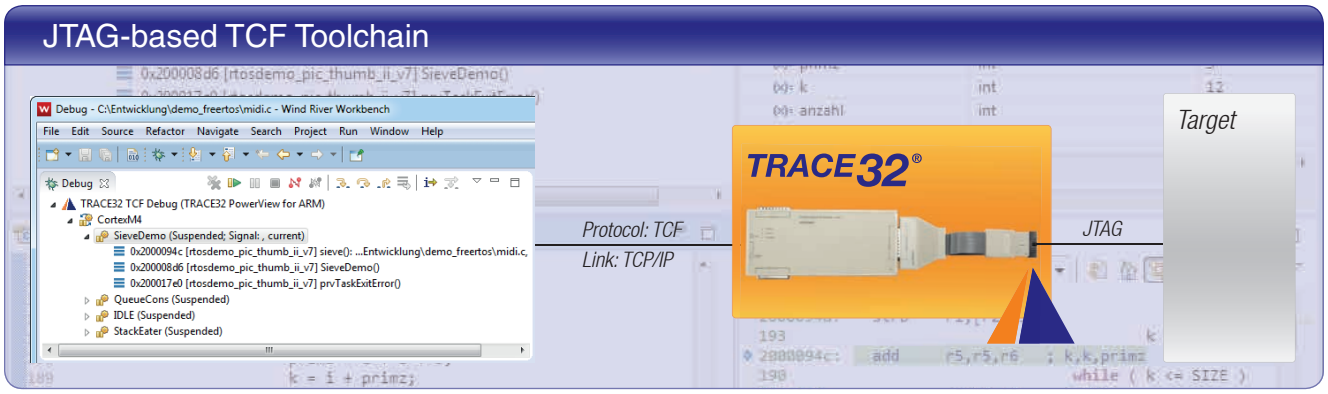
Intel® Trace Hub Library

Die *Intel® TH Library* wandelt System-Trace-Pakete in für Menschen verständliche Trace-Messages um. Da der *Intel® TH* alle Traceinformationen in das MIPI STPv2.1 Protokoll verpackt hat, müssen diese zu-nächst ausgepackt und dann zur Dekodierung verteilt werden (siehe Bild):

- *Intel® Processor Trace* Pakete verbleiben in TRACE32 und werden dort direkt für die Auswertung dekodiert.
- Die System-Trace-Pakete werden zur Dekodierung an die *Intel® TH Library* weitergeleitet. Die von der *Intel® TH Library* an TRACE32 zurückgelieferten verständlichen Trace-Messages, können dann in TRACE32 über eine eigene Kommandogruppe dargestellt und ausgewertet werden.

Die einfache Möglichkeit einer zeitkorrelierten Dar-stellung aller Traceinformationen in TRACE32 Power-View erlaubt einen schnellen Überblick über die Platt-form-weiten Abläufe.

TRACE32 als TCF-Agent



Seit Oktober 2015 arbeitet TRACE32 auch als TCF-Agent. Damit ist es nun möglich, die Wind River Workbench beziehungsweise den Eclipse Debugger als IDE und einen TRACE32 Debugger als Debug-Backend zu verwenden.

TCF

Das *Target Communication Framework* (TCF) wurde innerhalb der *Eclipse Foundation* als Protokoll-Framework mit der Zielsetzung entwickelt, für das Debuggen ein einheitliches Kommunikationsprotokoll zwischen IDE und Targetsystem zu definieren.

Das TCF-Kommunikationsmodell basiert auf der Idee von Services. Ein Service ist eine Gruppe von verwandten Befehlen, Events und einer dazugehörigen Semantik. Der *Memory Service* definiert beispielsweise eine Gruppe von Befehlen und Events für das Speicher-Lesen und -Schreiben. TCF definiert eine Reihe von Standardservices. Gleichzeitig ist das Framework offen für die Definition herstellerspezifischer Services.

TRACE32 TCF Plug-In

Nachdem die TRACE32 Software als TCF-Agent gestartet wurde, stellt sie ihre Services über TCP/IP der Wind River Workbench bzw. dem Eclipse Debugger zur Verfügung. Angeforderte Services werden von TRACE32 mit Hilfe des angeschlossenen Debuggers bedient. Dabei spielt es keine Rolle, ob ein hardwarebasierter Debugger via JTAG an das Targetsystem angeschlossen ist oder ein reiner Software-Debugger mit einem virtuellen Target kommuniziert. Aktuell bietet TRACE32 alle für das Debuggen relevanten Standard-Services an. Abhängig vom Kunden-Feedback ist geplant, zukünftig auch spezielle TRACE32 Services für erweiterte Funktionen zu entwickeln.

Für Kunden, die TRACE32 lieber aus der *Wind River Workbench* bzw. aus dem Eclipse Debugger heraus konfigurieren und starten wollen, bietet Lauterbach ein TRACE32 TCF Plug-In an.

Wind River Workbench

Für Entwickler, die bevorzugt mit der *Wind River Workbench* arbeiten, konnte Lauterbach bisher keine integrierte Debug-Lösung anbieten. Diese Einschränkung ist nun aufgehoben.

Eclipse Debugger

Die bisherige GDB-basierte Anbindung war leider auf die von GDB unterstützten Prozessorarchitekturen und Compiler beschränkt.

Über seine TCF-Services kann TRACE32 nun eine für alle Prozessorarchitekturen und Compiler offene Kommunikationsschnittstelle für das Debuggen aus Eclipse anbieten.

TRACE32 Support für Wind River

VxWorks 5/6/7

VxWorks 653 2.x

VxWorks 653 3.x

Wind River Linux

Wind River Hypervisor 2.x

VxWorks Microkernel Profile

WIND RIVER

TRACE32 als Frontend für UndoDB

Seit Mitte 2015 kann TRACE32 als Frontend für den reversible Debugger UndoDB eingesetzt werden. Unterstützt werden folgende Architekturen: ARM/ Cortex sowie Intel® x86/x64.

Mit dem UndoDB Target Server steht dem Linux-Entwickler ein Tool zur Verfügung, das es erlaubt, einen Anwendungsprozess zu debuggen sowie Details über seine Ausführung aufzuzeichnen.

Neben der Kontrolle des Debuggens übernimmt das TRACE32 Frontend die Aufgabe, die aufgezeichneten Daten des UndoDB Target Servers in der TRACE32 GUI darzustellen. Ähnlich wie bei einer Traceaufzeichnung hat der Entwickler damit die Möglichkeit, die Applikation vorwärts und rückwärts zu debuggen („reverse debugging“). Fehler im Anwendungsprozess lassen sich auf diese Weise einfach und schnell lokalisieren.

Um optisch zu verdeutlichen, dass das Debugging der Aufzeichnung aktiviert ist, wird in der TRACE32 Statuszeile mit *reverse* der dargestellte Aufzeichnungszeitpunkt referenziert. Zudem wechseln die Debug-Buttons im Source-Listing auf eine gelbe Darstellung. Die TRACE32 GUI wird automatisch auf folgenden Darstellungsmodus umgeschaltet:

- Im Source-Listing wird der Instruction Pointer auf seinen Wert zum dargestellten Aufzeichnungszeitpunkt zurückgesetzt (siehe Bild).
- Ebenso werden die Inhalte von Speichern und Variablen für den Aufzeichnungszeitpunkt dargestellt.