

Debug & trace tools for new NXP

i.MX8 and S32 automotive processors

Le applicazioni automotive sono sempre più esigenti in termini di real-time, sicurezza, integrazione. Obiettivi che vengono raggiunti impiegando nuove architetture multi-core, nuovi standard software, nuovi sistemi operativi e hypervisor.

Questa presentazione offre una panoramica sulle novità che Lauterbach ha introdotto per supportare i propri clienti e vincere questa nuova sfida: TRACE32 integra il supporto ai nuovi chip i.MX8 e S32 (con core ARMv8 Cortex A72, A53, A35 e M4), estende il supporto ORTI al nuovo standard AUTOSAR ARTI, supporta applicazioni multi-core, multi-OS e Hypervisor.

La presentazione comprende live demo su eval board i.MX8 QM e sistema operativo linux multicore SMP



Agenda del seminario



Maurizio Menegotto
Lauterbach, relatore



Marco Ferrario
Lauterbach, Live Demo

- **Documentazione & news**
- TRACE32 tools per ARM Cortex ARMv7 & ARMv8
- Debug & trace support NXP S32
- Multicore debug & trace support NXP iMX8
- Live demo iMX8
- Supporto ai sistemi operativi e alla virtualizzazione
- Live demo SMP Linux i.MX8
- Conclusione, Q&A




Documentazione









Documentazione stampata:

- ✓ Brochure Lauterbach Company Profile
- ✓ Brochure Product Overview
- ✓ Flyer Debug & Trace ARM
- ✓ Flyer ARM Linux debug & trace
- ✓ Flyer uTrace Cortex-M
- ✓ Newsletter 2017



Nel USB Flash Disk:  Lauterbach_NXP_Automotive_Seminar_2017.pdf



-  Flyer
-  Manuali
-  Newsletters
-  Press Release
-  Seminars 2011--2016
-  Slides Lauterbach ARM Expert Day 2016
-  Slides Lauterbach Automotive Seminar 2...
-  Webinar Video

News 2017

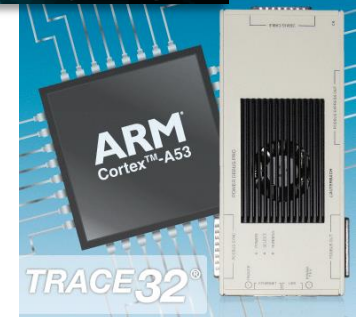
TRACE32 software news

- ✓ Debug di **Hypervisor**: una nuova debug extension per TRACE32
- ✓ Supporto **PikeOS** aggiornato alla release 4.2
- ✓ Supporto **Android** aggiornato per ARM64
- ✓ Supporto **QNX 7.0**
- ✓ Nuovo standard **AUTOSAR ARTI**
- ✓ Nuove funzionalità software e integrazioni



TRACE32 chip support news

- ✓ **ARMv8 Trust Zone** – supporto di sistemi **big.LITTLE**
- ✓ **NXP S32** – nuova serie di microcontrollori Cortex-M per automotive
- ✓ **NXP i.MX8** – nuova serie multicore **ARMv8** Cortex-A53/A72, Cortex-M



Agenda del seminario



Maurizio Menegotto
Lauterbach, relatore

- **Documentazione & news**
- **TRACE32 tools per ARM Cortex ARMv7 & ARMv8**
- Debug & trace support NXP S32
- Multicore debug & trace support NXP iMX8
- Live demo iMX8
- Supporto ai sistemi operativi e alla virtualizzazione
- Live demo SMP Linux i.MX8
- Conclusione, Q&A

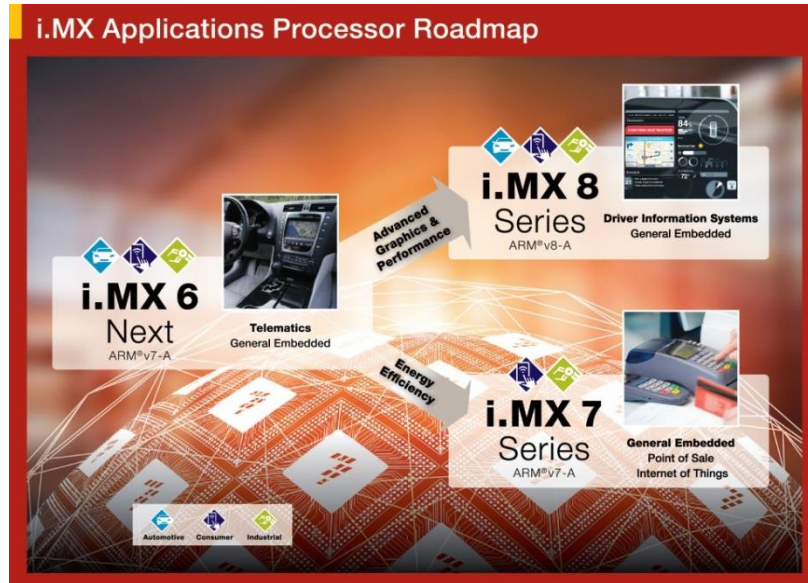


TRACE32 tools per ARM Cortex ARMv7 & ARMv8

Tra i chip più utilizzati per applicazioni automotive c'è NXP i.MX6, che è un SoC multicore con da 1 a 4 core Cortex-A9 e in alcuni casi Cortex-M4.

i.MX6 è basato su architetture ARMv7 32 bit, i tools TRACE32 per questi chip sono stati oggetto dei nostri seminari del [2015](#) e [2016](#).

La nuova generazione di core ARM Cortex ha architetture [ARMv8](#) con core 32/64 bit, nuovi processori basati su questi core sono già disponibili: **i.MX8**



Debug & Trace con ARM CoreSight™

Nei chip basati su ARM Cortex tutte le funzionalità di debug & trace sono gestite dal modulo **CoreSight**. [CoreSight](#) è una collezione di componenti hardware che possono essere implementati nel design di un chip.

Componenti CoreSight

Debug Access Port (**DAP**)

BenchMark Counter (**BMC**)

Cross Trigger Interface (**CTI**)

System Trace Macrocell (**STM**)

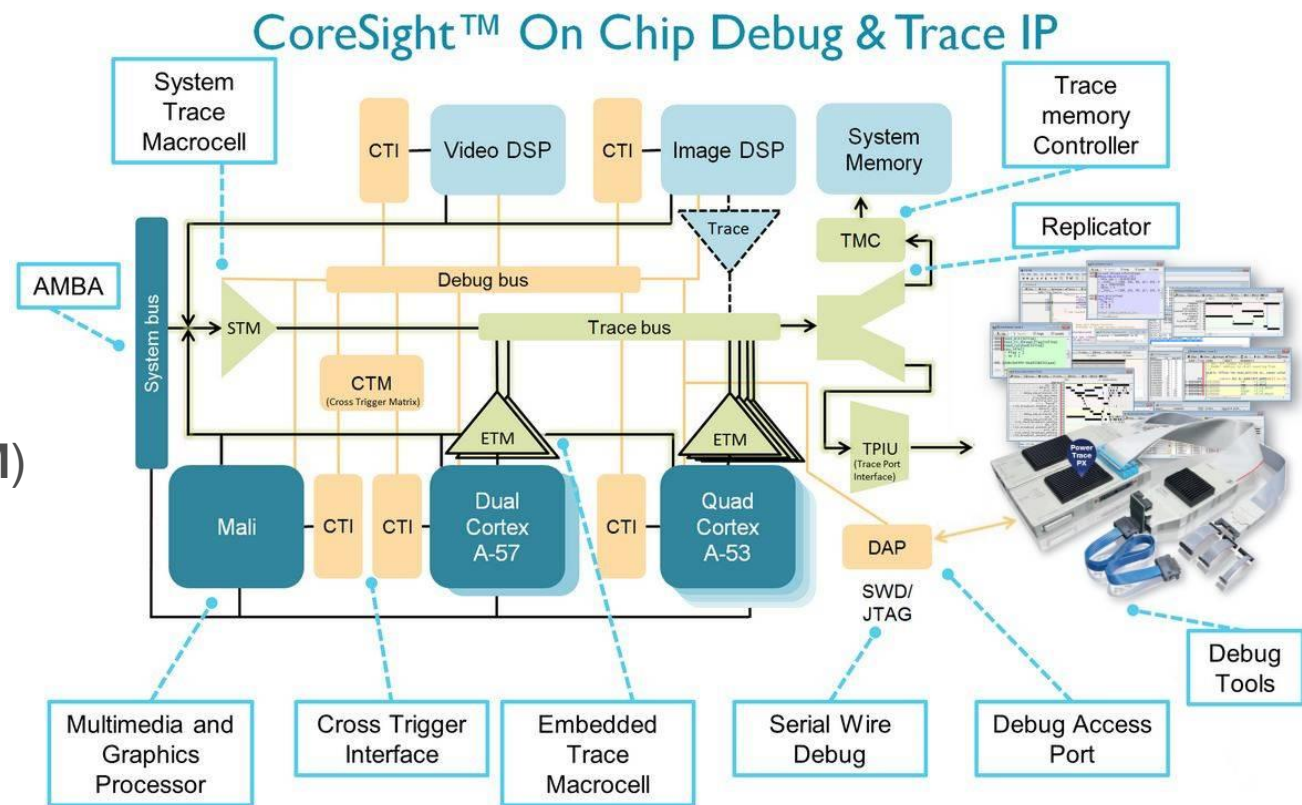
Embedded Trace Macrocell (**ETM**)

Embedded Trace Buffer (**ETB**)

Trace Port Interface Unit (**TPIU**)

Trace Memory Controller (**TMC**)

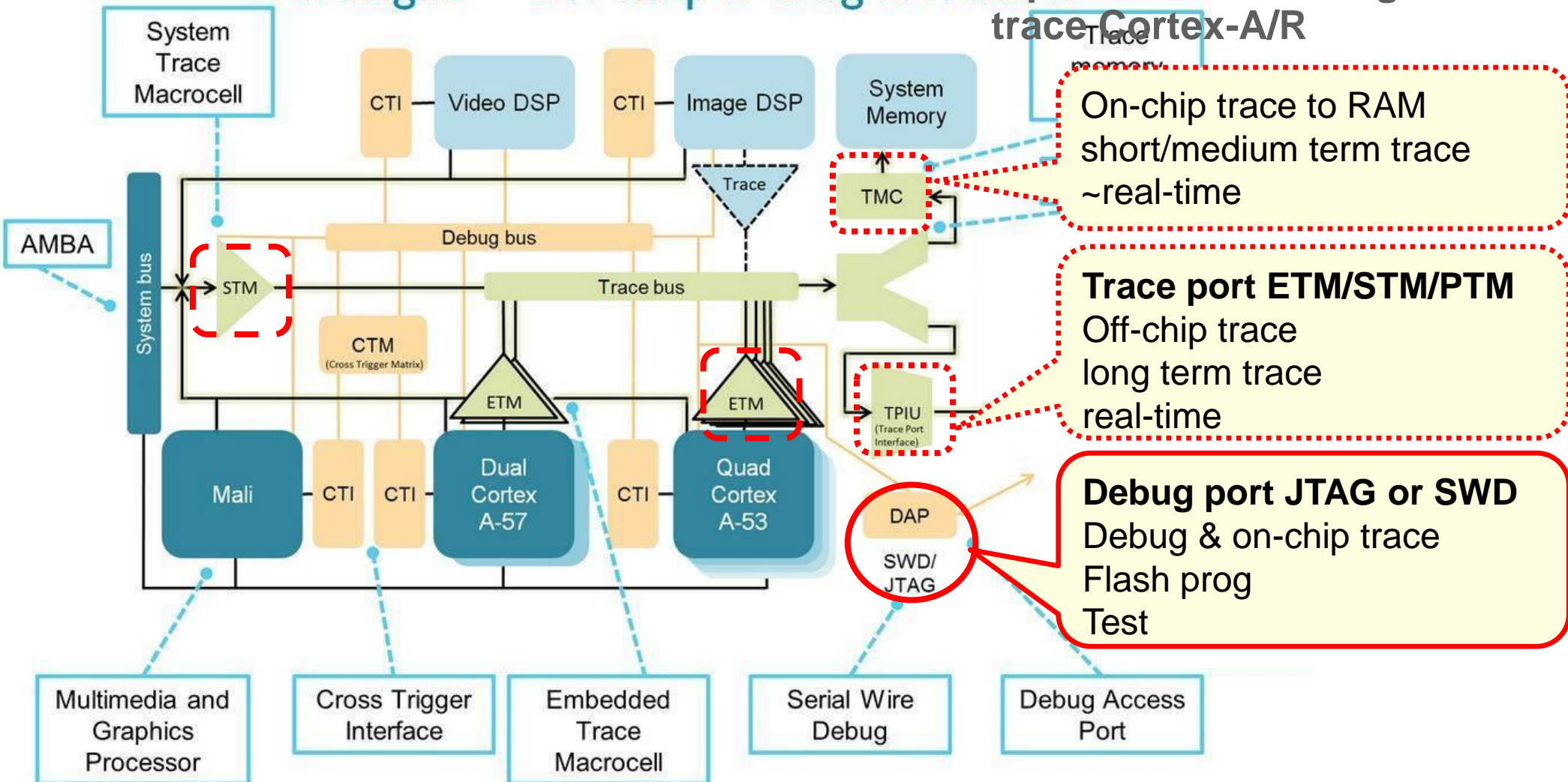
System Trace Macrocell (**STM**)



CoreSight components in a typical multicore ARMv8 SoC

Debug & Trace con ARM CoreSight™

CoreSight™ On Chip Debug & Trace

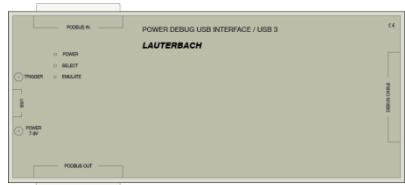


TRACE32 debug tools ARMv7 & ARMv8 Cortex-A/R

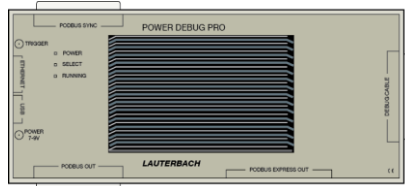
Un sistema debug è costituito da un modulo PowerDebug a cui si collega un JTAG Debugger per l'architettura ARM prescelta.

PowerDebug USB3

Modulo debug base
non espandibile a PowerTrace
Link USB 2/3

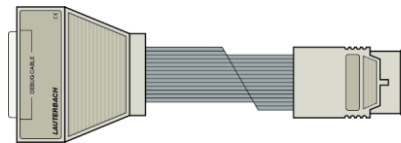


OR



PowerDebug PRO

Modulo debug universale
Espandibile a PowerTrace
Link USB 2/3 & Ethernet 10/100/1000



JTAG debug cable ARM
es. per ARM9 o ARM11



Si possono applicare estensioni software per:

- Cortex-A/R ARMv7 32 bit
- Cortex-A/R ARMv8 32/64 bit
- Cortex-M
- Multicore
- On-chip Trace (ETB o ETR)

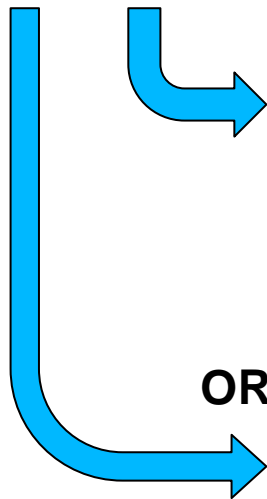
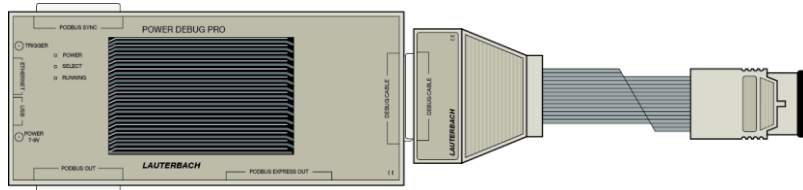
O add-on per:

- altri core non ARM (es. DSP)

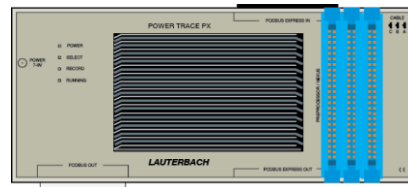
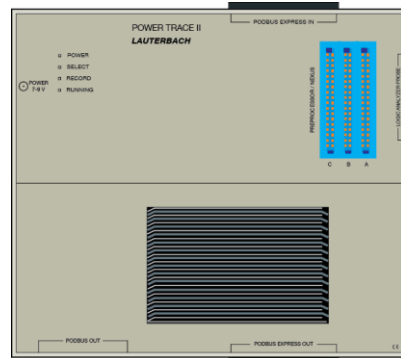
TRACE32 trace tools ARMv7 & ARMv8 Cortex-A/R

Un sistema debug+trace è costituito da un modulo PowerDebug PRO/JTAG ARM a cui si collega un modulo PowerTrace-PX o PowerTrace-II e un trace probe ARM.

PowerDebug PRO JTAG ARM

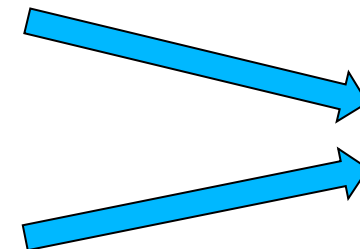


OR



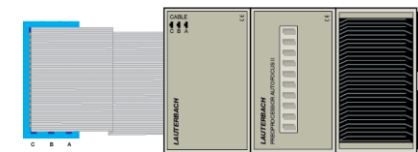
PowerTrace II

Trace storage 1/2/4 Gbyte
Trace streaming & RTS
Logic/protocol analyzer



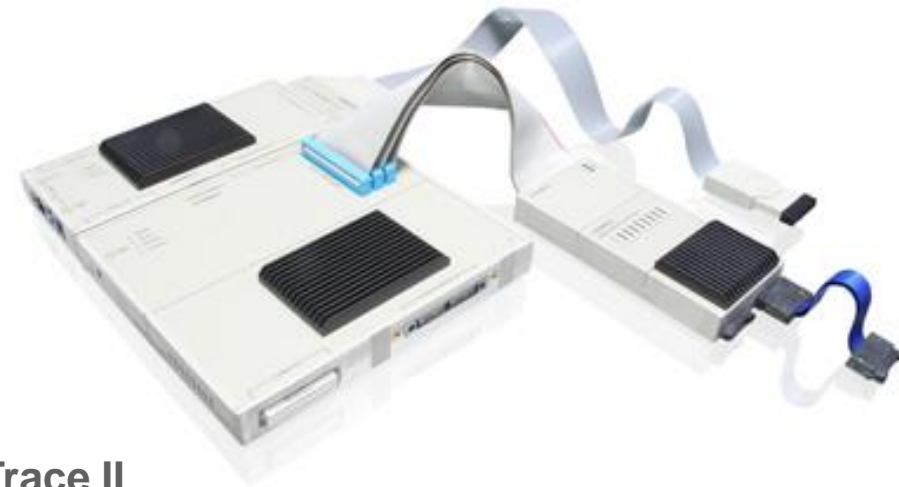
PowerTrace PX

Trace storage 512MByte



Trace probe ARM

Preprocessor ETM Autofocus-II



TRACE32 PowerView ARMv8

Gli ARMv8 sono architetture a 32/64 bit, i core 64 bit sono supportati da un eseguibile **TRACE32 PowerView** specifico:

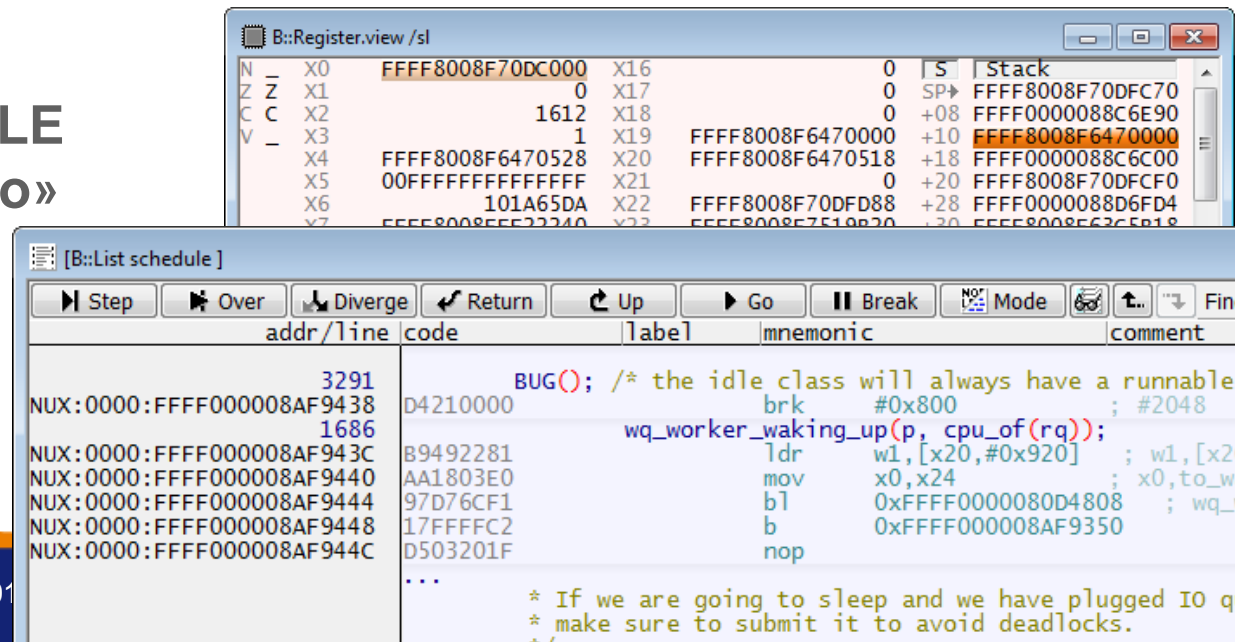
t32marm.exe PowerView per tutti gli ARM/Cortex **32 bit**
(ARM7, ARM9, ARM11, Cortex-A/R ARMv7, Cortex-M)

t32marm64.exe PowerView per i ARM/Cortex **64 bit**
(Cortex-A/R ARMv8)



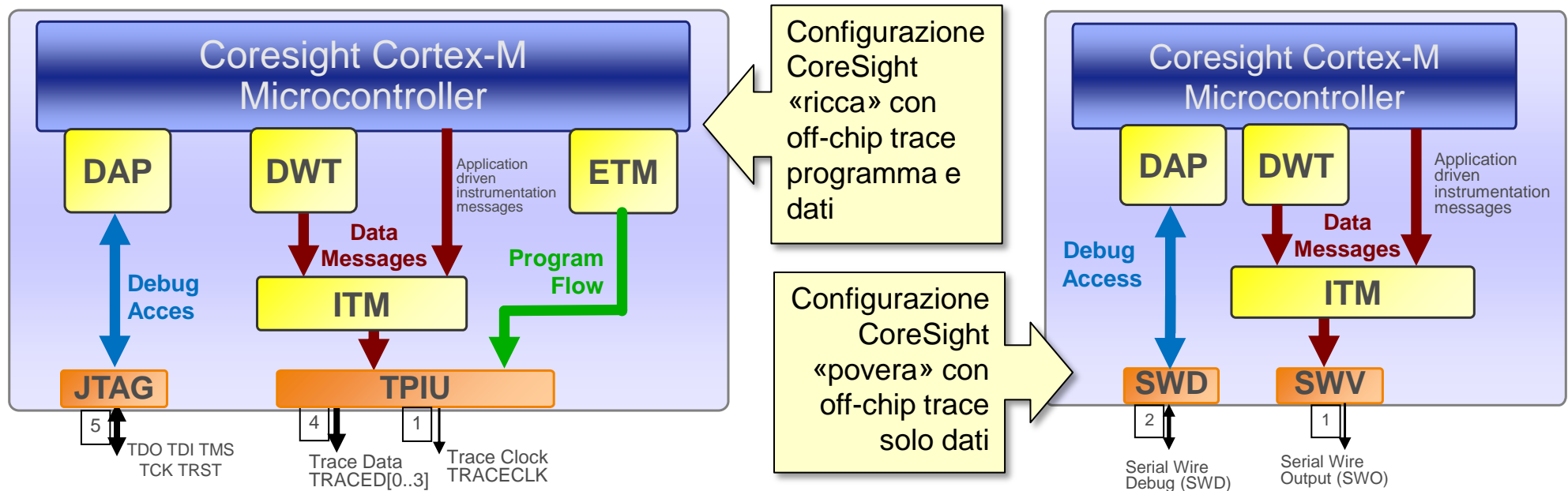
➔ Tra le novità introdotte in TRACE32 con le nuove architetture ARM:

- ✓ Supporto ARM **TrustZone®**
- ✓ Supporto Architettura **Big.LITTLE**
- ✓ Supporto multicore **SMP «misto»**
- ✓ Supporto **ARM 64 bit**
- ✓ ARM trace **ETR ETF**



TRACE32 debug & trace tools ARM Cortex-M

Anche i Cortex-M hanno risorse CoreSight per il debug e il trace. Le configurazioni «modulari» TRACE32 per Cortex-A/R viste nelle pagine precedenti possono essere usate anche per i Cortex-M, sia da soli che integrati in SoC multicore.



Ma nel caso in cui serva il solo supporto per Cortex-M, come ad esempio nei chip NXP Kinetis e S32K, o nei nuovi i.MX RT, allora è possibile usare soluzioni «dedicate» e più economiche.

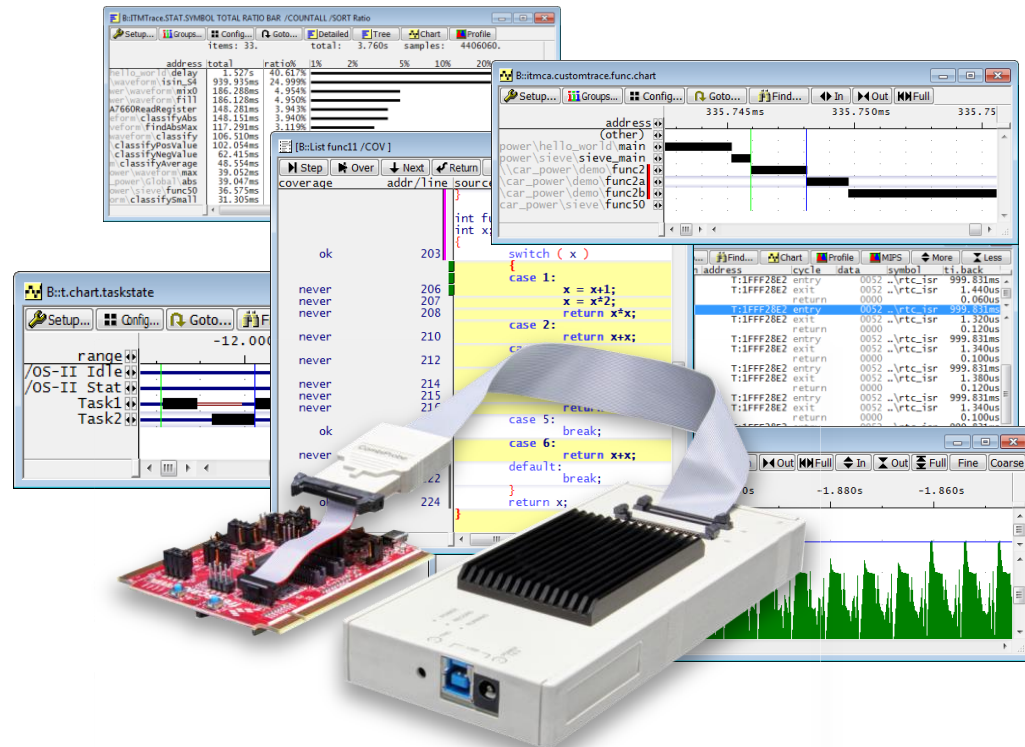
Approfondimento debug & trace Cortex-M

Per approfondimenti sui tools per Cortex-M, raccomando la visione delle slides e del video del nostro webinar in italiano dal titolo:

Debugging ARM Cortex-M con μ Trace



www.lauterbach.com/tut-i_utrace.html

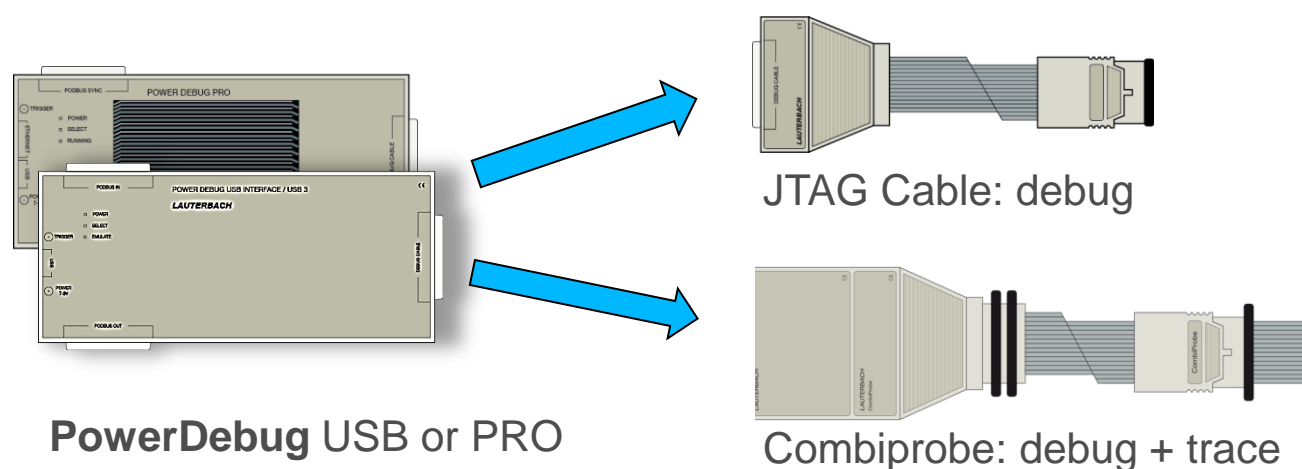


TRACE32 debug & trace tools ARM Cortex-M

I sistemi PowerDebug o PowerTrace per un qualsiasi ARM o Cortex-A/R, possono essere aggiornati per il supporto Cortex-M aggiungendo la debug license per Cortex-M (**sw-extension**).

➔ Questa è la soluzione migliore soprattutto nel caso di debug di SoC multicore Cortex-A/R e Cortex-M.

➔ Se invece ho un PowerDebug con JTAG debugger non ARM, è necessario acquistare il JTAG cable Cortex-M (debug) oppure un Combiprobe Cortex-M (debug+trace), serve cioè un **hw-upgrade**.



Per Cortex-M «poveri» senza alcuna risorsa trace off-chip. (al limite con trace on-chip ETB).

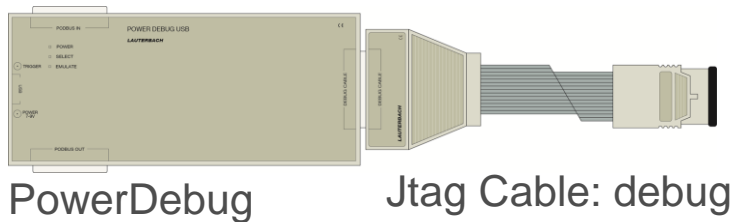
Per Cortex-M «ricchi» con risorse trace on-chip e off-chip (ETM, ITM, DWT)

TRACE32 debug & trace tools ARM Cortex-M

Le soluzioni debug+trace «a basso costo» per Cortex-M sono ben 3 (!)

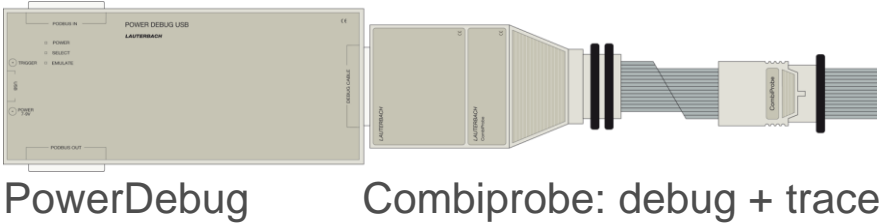
Se ho un PowerDebug posso aggiornarlo in due modi:

Aggiornamenti



Se ho un Cortex-M senza trace port, o se sono interessato al solo debug, è sufficiente aggiungere il JTAG Cable Cortex-M

Best Price

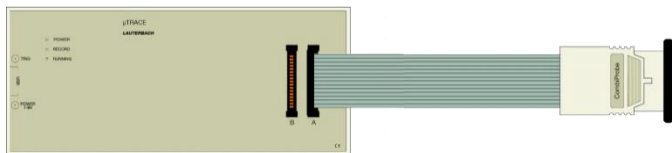


Se il chip ha una trace port ETM/ITM o anche solo SWV per ITM, posso aggiungere o aggiornare un Combiprobe

Best Performance

Se devo acquistare un nuovo sistema dedicato al solo Cortex-M è conveniente scegliere **µTrace** indipendentemente dalle risorse del chip

Nuovo



µTrace Cortex-M
Jtag debugger + trace 256MB

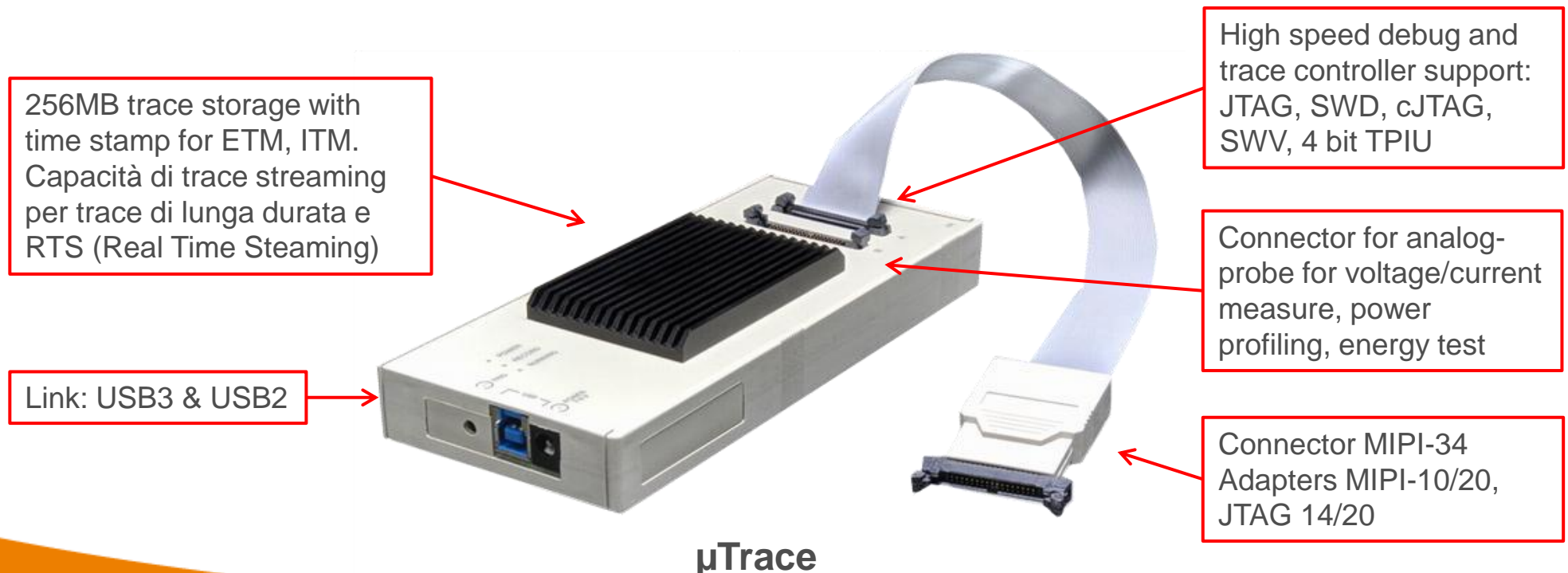
Best Price
Best Performance
For Cortex-M only

µTrace: la soluzione all-in-one per Cortex-M

µTrace è un prodotto hardware dedicato, progettato specificamente per il supporto a tutti i chip basati su ARM Cortex-M.

µTrace utilizza l'ambiente software PowerView comune a tutti i sistemi Lauterbach.

In questo modo è stato possibile ridurre notevolmente i costi ed offrire così un prodotto ad altissime prestazioni ad un prezzo competitivo.



Agenda del seminario



Maurizio Menegotto
Lauterbach, relatore

- **Documentazione & news**
- **TRACE32 tools per ARM Cortex ARMv7 & ARMv8**
- **Debug & trace support NXP S32**
- Multicore debug & trace support NXP iMX8
- Live demo iMX8
- Supporto ai sistemi operativi e alla virtualizzazione
- Live demo SMP Linux i.MX8
- Conclusione, Q&A



Debug & trace support NXP S32

Gli NXP S32 sono una famiglia di microcontrollori per automotive, basati core ARM Cortex.

La famiglia degli **S32K** è basata sui Cortex-M, in particolare Cortex-M0+ per gli S32K11x e Cortex-M4F per gli S32K14x.

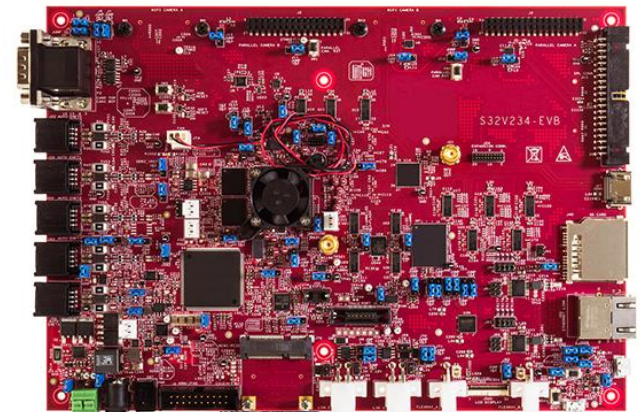
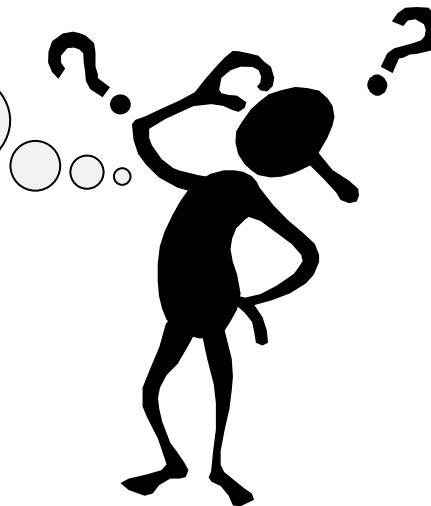
La famiglia degli **S32V** è multicore, basata sui Cortex-A e Cortex-M, in particolare nel S32V234 vengono impiegati 4 Cortex-A53 e un Cortex-M4.

- S32
 - S32K116
 - S32K118
 - S32K142
 - S32K144
 - S32K146
 - S32K148
 - S32V234
 - Debug & Off-Chip Trace
 - Debug & On-Chip Trace

Che TRACE32
usare per
S32K o S32V ?



EVB S32K144



EVB S32V234



Debug & trace support NXP S32K series

Dal punto di vista del debug e trace, gli **S32K11x** (Cortex-M0+) e gli **S32K14x** (Cortex-M4F) sono del tutto simili agli NXP Kinetis, e vengono supportati dagli stessi debug e trace tools TRACE32 per Cortex-M.

In particolare tutti i sistemi TRACE32 per Cortex-M prima descritti supportano gli S32K.

Es. eval board EVBS32K144:

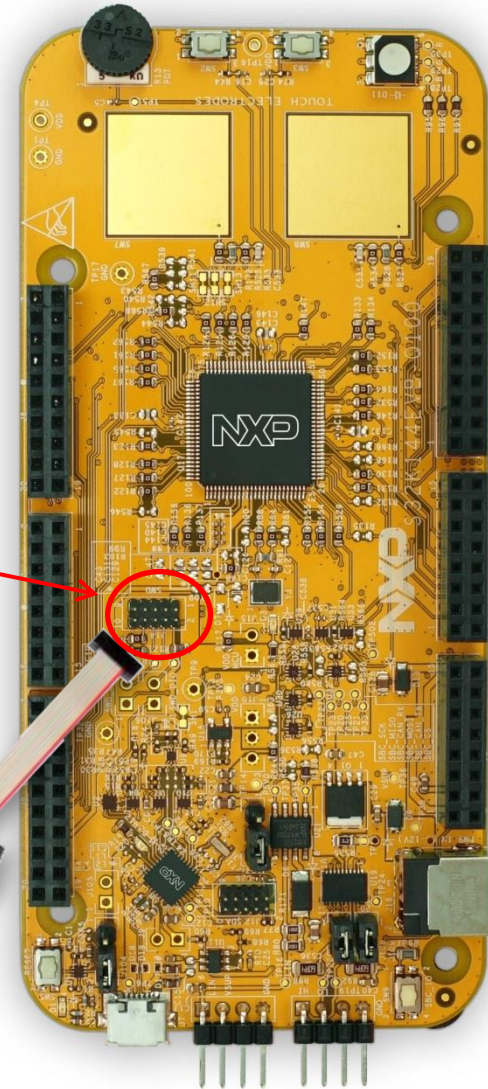
- Debug port MIPI-10
- Supporto debug via SWD
- Off-chip data trace via SWV

Tools TRACE32 raccomandati:



µTrace OR CombiProbe

MIPI-10 debug port

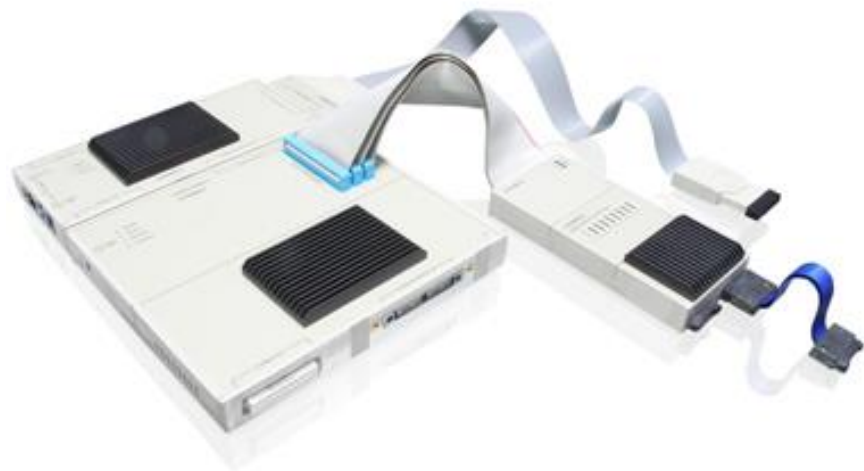


EVB S32K144

Debug & trace support NXP S32V series

Gli **S32V** sono una serie basata su core diversi e che operano a velocità molto superiori.

Ad esempio **S32V234** è un SoC multicore con 4 ARMv8 Cortex-A53 ed un Cortex-M4.

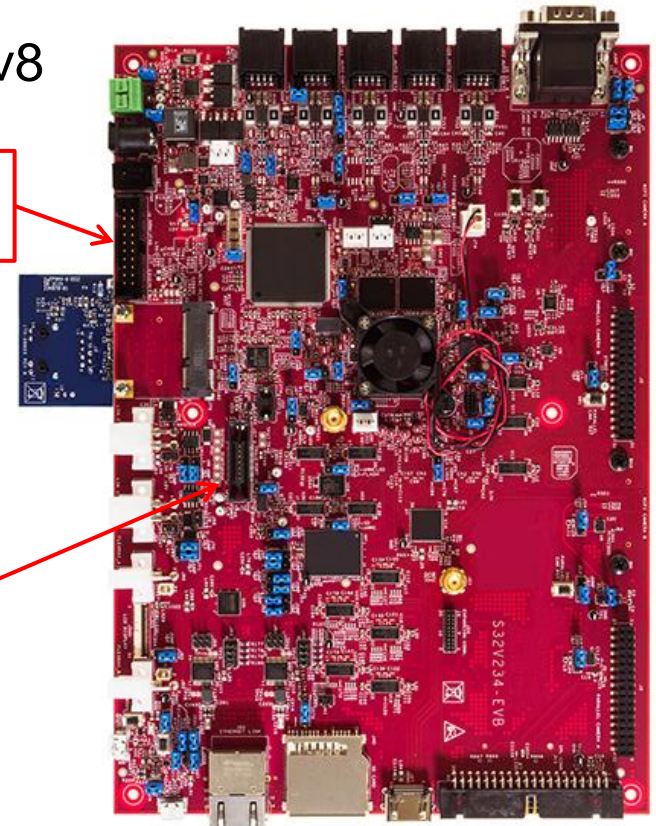


Sistema PowerTrace-II

ARM JTAG port
(standard ARM 20 pin)

ARM TRACE port
(standard Mictor 38)

EVB S32V234



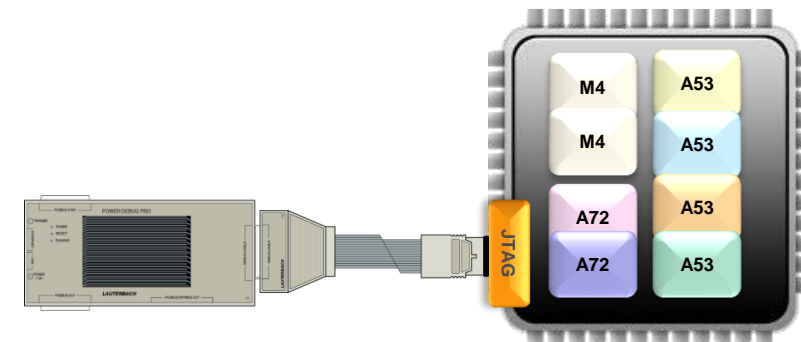
➔ I tools di sviluppo per il supporto di questo chip sono esattamente gli stessi che si utilizzano per il debug della nuova serie **i.MX8**, che tratteremo nel prossimo capitolo.

Agenda del seminario



Maurizio Menegotto
Lauterbach, relatore

- **Documentazione & news**
- **TRACE32 tools per ARM Cortex ARMv7 & ARMv8**
- **Debug & trace support NXP S32**
- **Multicore debug & trace support NXP iMX8**
- Live demo iMX8
- Supporto ai sistemi operativi e alla virtualizzazione
- Live demo SMP Linux i.MX8
- Conclusione, Q&A



TRAC32 support i.MX8 series (& S32V)

I chip NXP **i.MX8** sono una nuova serie di SoC multicore basati su un mix di core ARMv8 64 bit e Cortex-M 32 bit.

- **i.MX 8** 2x Cortex-A72 + 4x Cortex-A53 + 2x Cortex-M4F
- **i.MX 8M** 4x Cortex-A53 + 1x Cortex-M4F
- **i.MX 8X** 4x Cortex-A35 + 1x Cortex-M4F

Anche gli **S32V** hanno una simile configurazione multicore

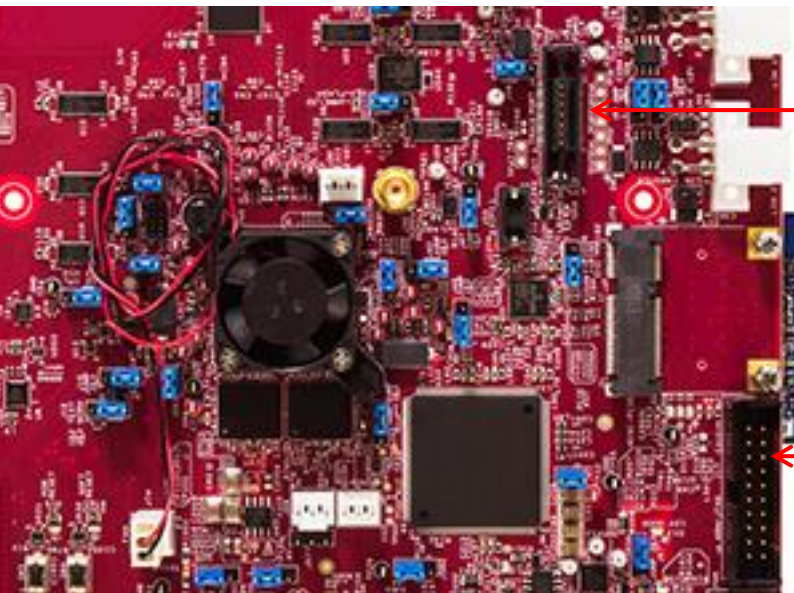
- **S32V234** 4x Cortex-A53 + 1x Cortex-M4F

➔ Documentazione USB Flash Disk



Debug & trace support i.MX8 series (& S32V)

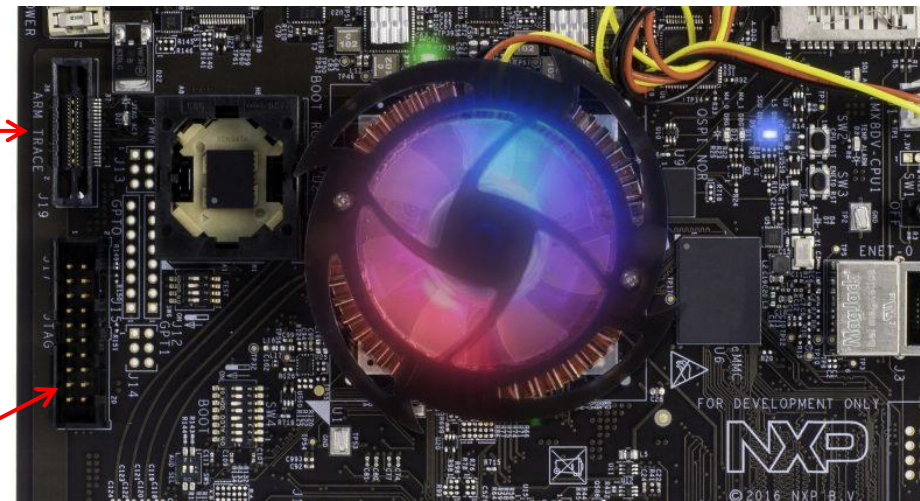
Gli **i.MX8** e **S32V** sono tutti dotati di una porta **JTAG** ARM Coresight standard per il debug e del supporto trace. Il trace può essere trasmesso off-chip tramite una trace port **ETM**, oppure può essere registrato su ram tramite **ETR** (Embedded Trace Router) o **TMC** (Trace Memory Controller).



Particolare evaluation board **S32V234**

ARM TRACE
standard
port
Mictor 38

ARM JTAG
standard
port
ARM 20 pin



Particolare evaluation board **iMX8**

➔ **TRACE32** supporta sia gli **i.MX8** che gli **S32V** con lo stesso sistema offrendo le stesse funzionalità di debug e trace.



JTAG MIPI-10
converter

Quali tools TRACE32 per i.MX8 & S32V ?

Le configurazioni PowerTools utilizzabili per queste architetture sono le stesse per i.MX6 (Cortex-A9), descritte nel seminario 2016.

La sola differenza è il JTAG Debugger, che deve poter supportare anche ARMv8 e Cortex-M, e che si può aggiornare con una estensione software.

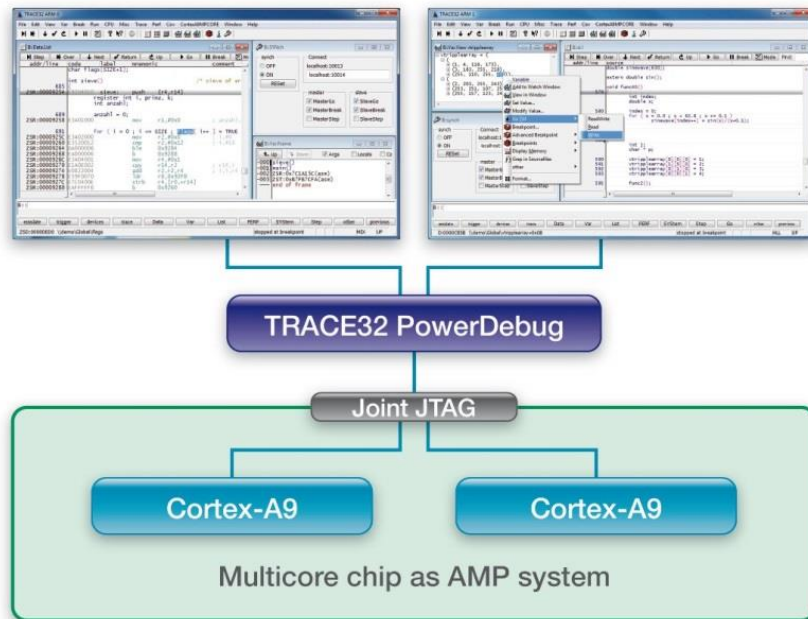
➔ Tutto il resto del sistema non cambia consentendo di supportare queste nuove architetture con un costo minimo.



Tutte le configurazioni modulari per debug e debug+trace possono essere utilizzate

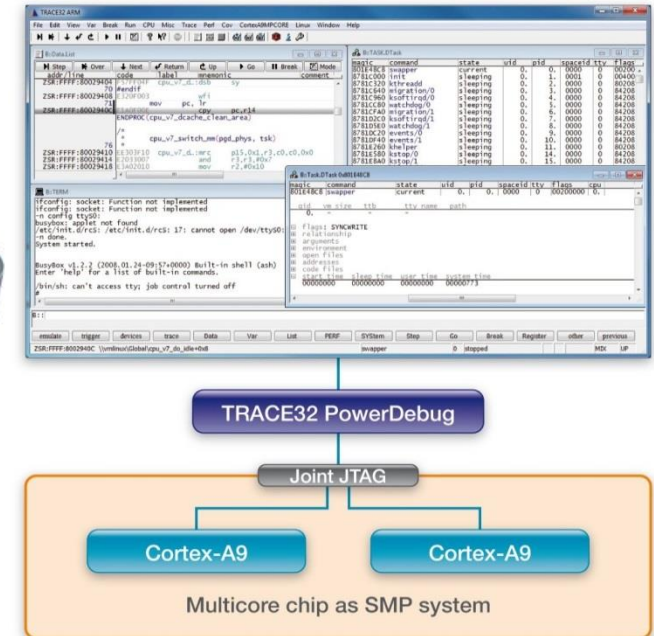
Supporto multicore in TRACE32

Con un unico PowerDebug hardware TRACE32 consente il debug multicore sia in modo SMP «System view» che in modo AMP «Core view».



Modo **AMP** (*Asymmetric Multi Processing*)

Una GUI PowerView per ogni core, ogni core esegue un programma diverso.



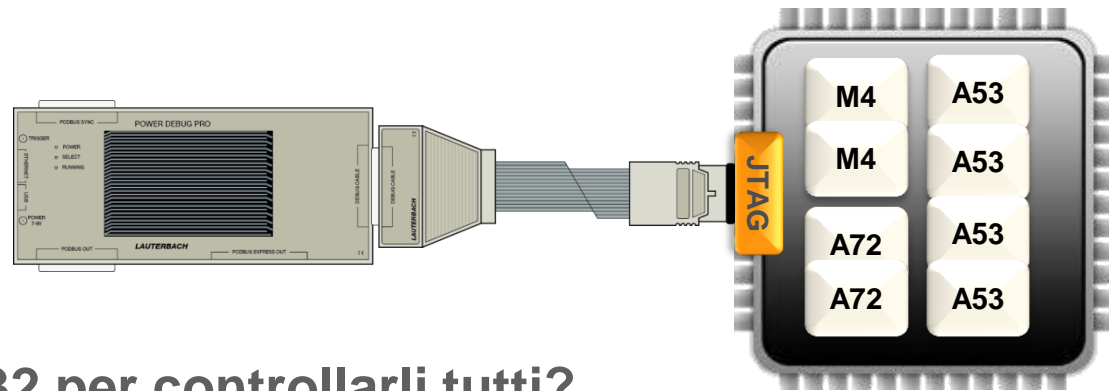
Modo **SMP** (*Symmetric Multi Processing*)

Una sola GUI PowerView per il controllo dell'intero sistema, tutti i core eseguono lo stesso programma.

Quale supporto multicore per i.MX8 & S32V ?

i.MX8QM ha ben 8 core:

2 Cortex-A72
2 Cortex-M4
4 Cortex-A53

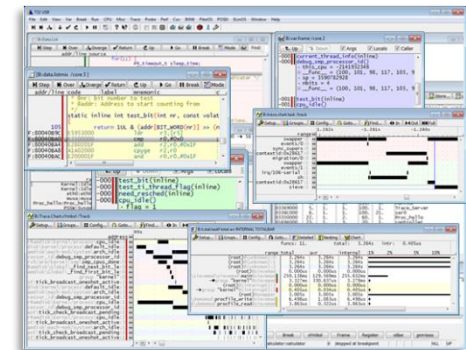
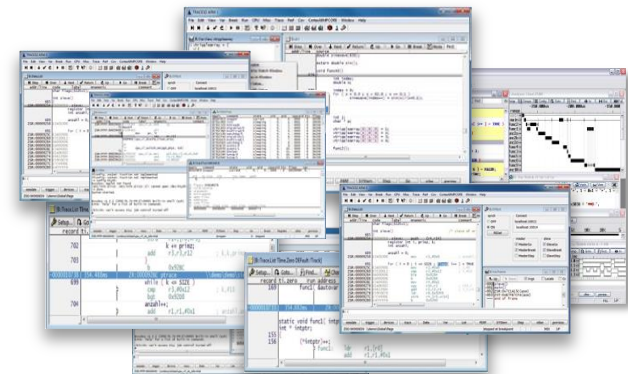


Come posso configurare TRACE32 per controllarli tutti?

➔ Una configurazione «AMP» prevederebbe ben 8 istanze PowerView, una per ogni core.

Con TRACE32 si può fare, ma è molto scomodo da gestire soprattutto se alcuni di questi core eseguono in modalità SMP.

➔ Una configurazione «SMP» prevederebbe una sola istanza PowerView per tutti i core. Ma questo non è possibile in quanto i core sono basati su architetture diverse, e queste richiedono diversi eseguibili TRACE32.



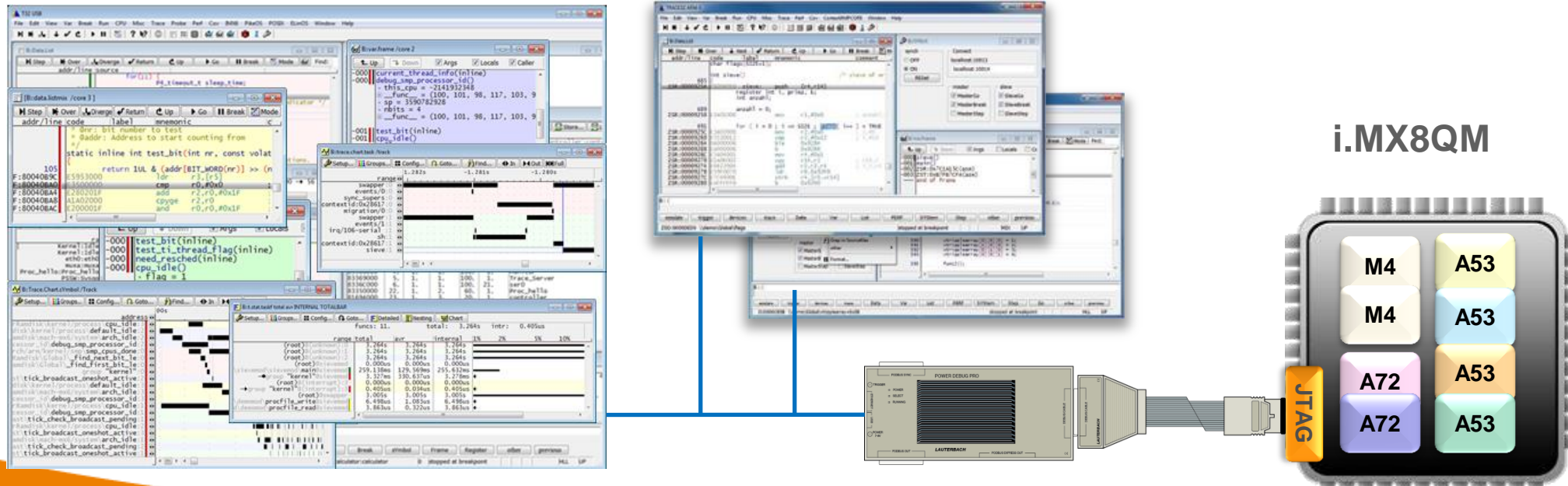
Supporto multicore per i.MX8 & S32V

L'approccio più adatto per il debugging dipende dal tipo core e da come l'applicazione è distribuita sui diversi core.

In questi casi è necessario considerare una configurazione mista AMP e SMP, che rispecchi l'architettura software.

Una istanza PowerView **SMP** che controlla tutti i core Cortex-A53 e A72 che eseguono Linux SMP

Due istanze PowerView **AMP** che controllano i due core Cortex-M con sistema operativo FreeRTOS

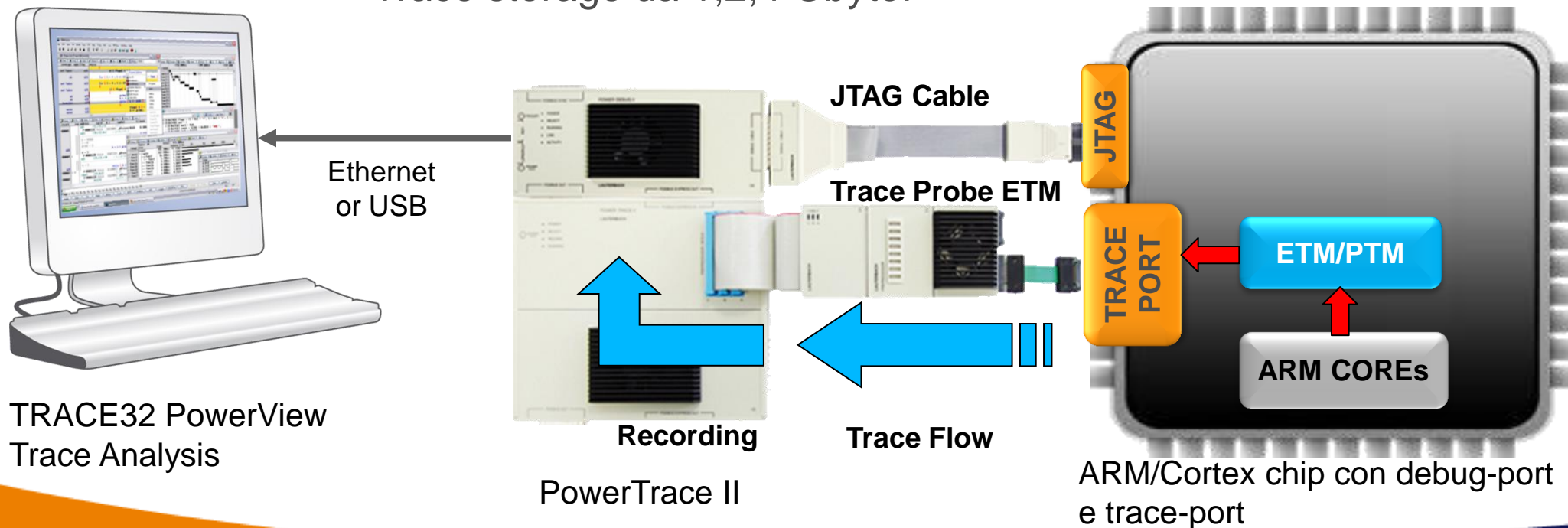


Supporto trace per i.MX8 & S32V

Negli i.MX8 e S32V è prevista una modalità trace ARM ETM «off-chip» ed una modalità «on-chip».

Nella modalità «off-chip» il trace (ETM o PTM) viene trasmesso in tempo reale attraverso una trace port parallela, per catturarlo serve un sistema PowerTrace.

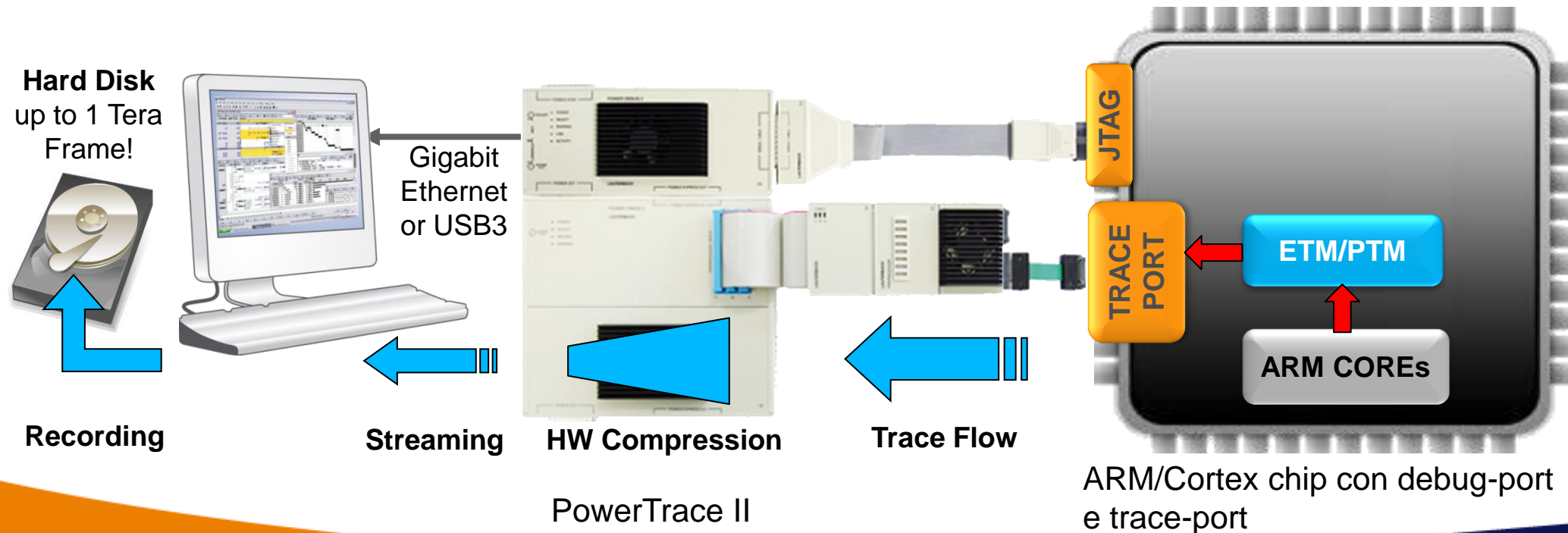
- + Tecnica assolutamente NON intrusiva
- + Richiede pochi pin dedicati (trace-port)
- + Trace storage da 1,2,4 Gbyte!



Trace streaming mode

Per prolungare enormemente il tempo di registrazione si usa il **TRACE STREAMING**. In questo modo il trace-flow viene compresso dal PowerTrace II e trasferito via gigabit ethernet o USB3 al computer dove viene registrato su HDD.

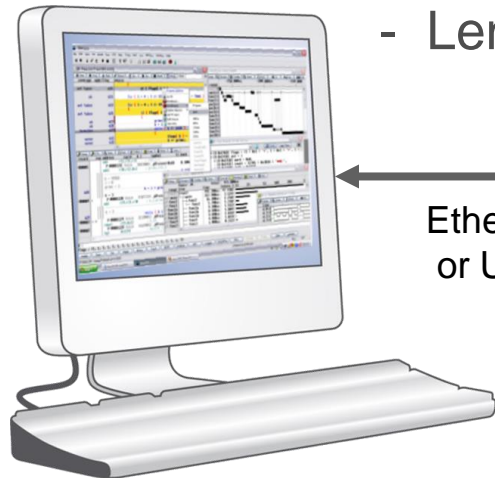
- + Tutti i vantaggi dell'off-chip trace
- + Trace di lunghissima durata, fino a 1 Tera Frame!
- + Compressione per ridurre occupazione hard-disk



On-Chip Trace ETR

Il chip ha la logica **ETM** per generare il trace e ha **ETR** (Embedded Trace Router) per indirizzarlo verso la memoria ram, che può essere una DDR esterna oppure una RAM interna. Dopo il break il trace buffer viene letto via JTAG dal debugger.

- + Non richiede trace-port, solo debug-port
- + Trace real-time con intrusione minima
- + Trace di lunga durata (DDR esterna)
- Trace di breve durata (RAM interna)
- Lentezza lettura trace lunghi



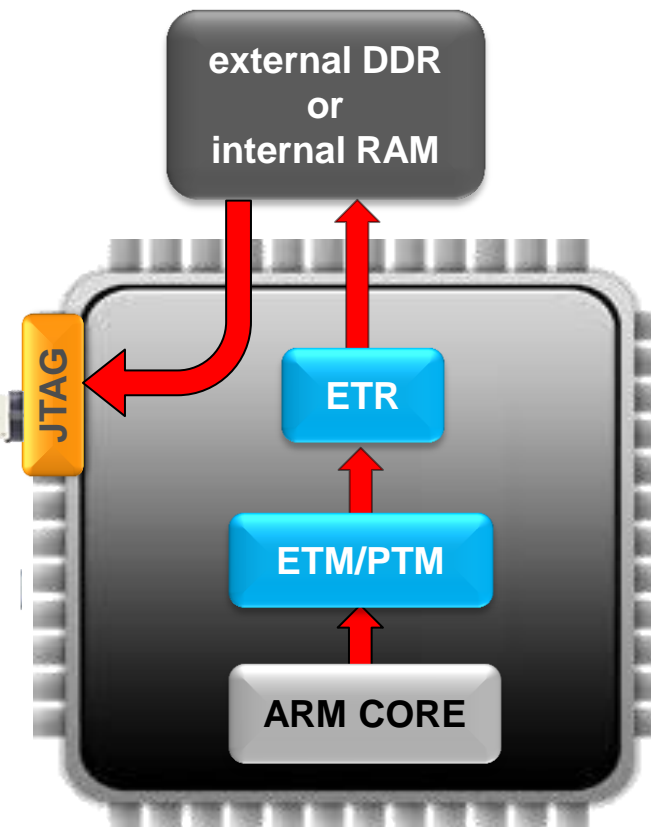
Ethernet
or USB

TRACE32 PowerView
Trace Analysis



JTAG Cable

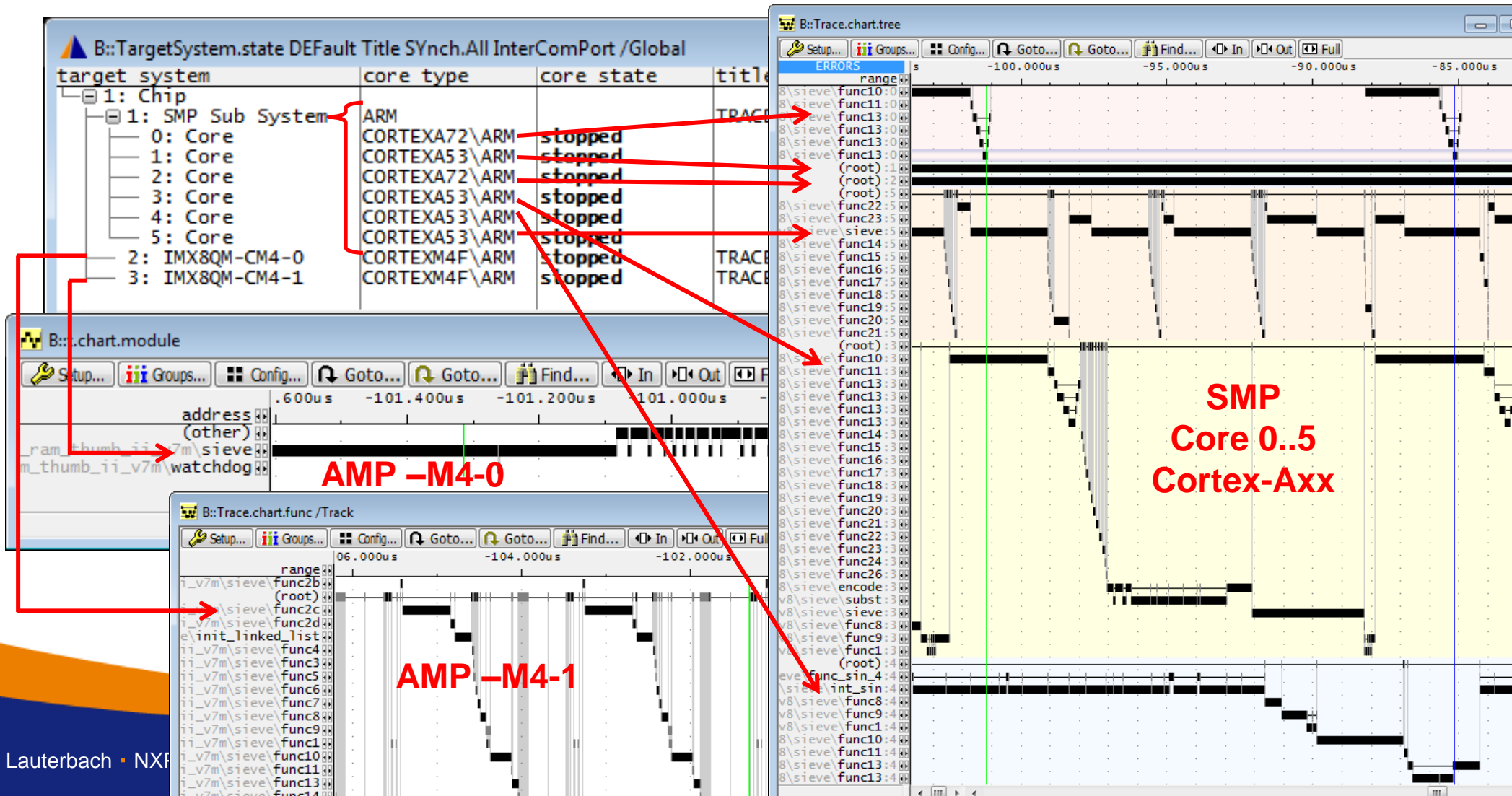
PowerDebug JTAG con
ARM Trace License



ARM/Cortex chip con debug-port
e on-chip trace (ETB)

Supporto trace per i.MX8 & S32V

Il supporto trace negli **i.MX8** è impressionante, il sistema consente il trace real-time multicore simultaneo di tutti i core. Nel **i.MX8QM** (8 cores di 3 tipi diversi) si può osservare e analizzare il comportamento run-time dell'intero sistema software.

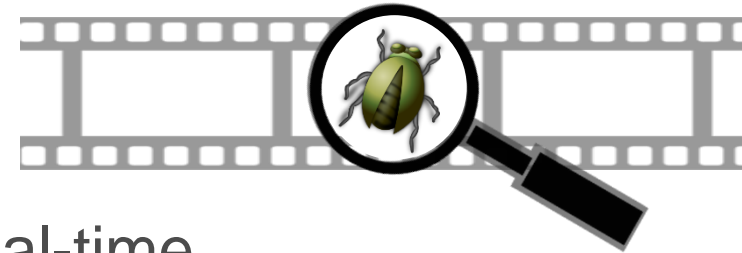


In TRACE32 il Real-Time Trace si usa per:

1) **Trace-based Debugging**

Debug rapido senza fermare la CPU

Trovare bugs che appaiono solo in real-time



2) **Ottimizzazione** con misure temporali

Analizzare le performance del codice

Analizzare eventi esterni



3) **Qualificazione**

Dimostrare il rispetto dei requisiti real-time

Verificare il code coverage



1) Trace Based Debugging: Trace.List

Il trace è di grande aiuto: è sempre attivo, ogni esecuzione in run o step viene registrata ed è immediatamente visibile per comprendere cosa è accaduto.

Ci si può fermare **DOPO** che il problema è accaduto e tornando indietro nel trace (cioè nel tempo) si può osservare il comportamento del programma nel punto in cui ha sbagliato:

The screenshot shows the Trace.List window with the following columns: record, run, address, cycle, data, symbol, and ti.back. The trace contains several instructions, including a while loop and a function call. A magnifying glass highlights the instruction at address 0x0002334, which is a ptrace instruction. A stick figure points to the trace, and a large orange arrow on the right points downwards, labeled 'TEMPO'.

Istruzioni eseguite

Istruzioni non eseguite

Data read/write

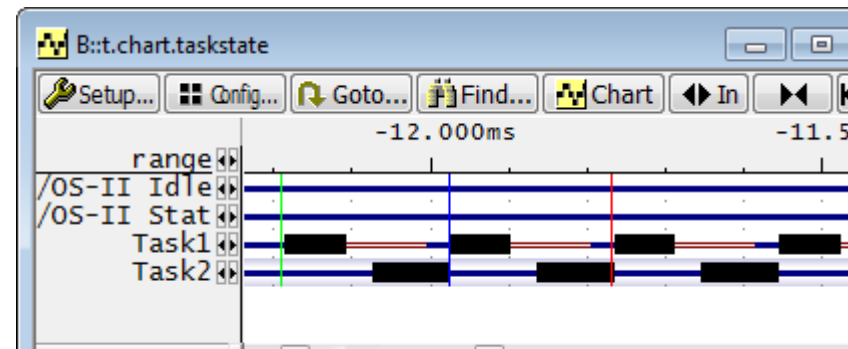
Tempo relativo

TEMPO

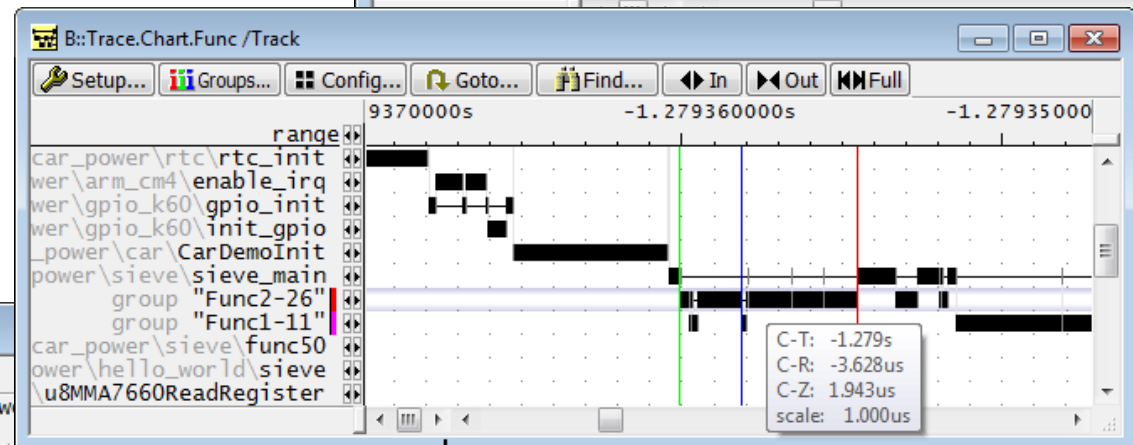
2) Ottimizzazione: Trace.Chart & Trace.Stat

L'individuazione dei punti dove ottimizzare il programma è molto facilitata dall'analisi grafica e statistica del trace registrato.

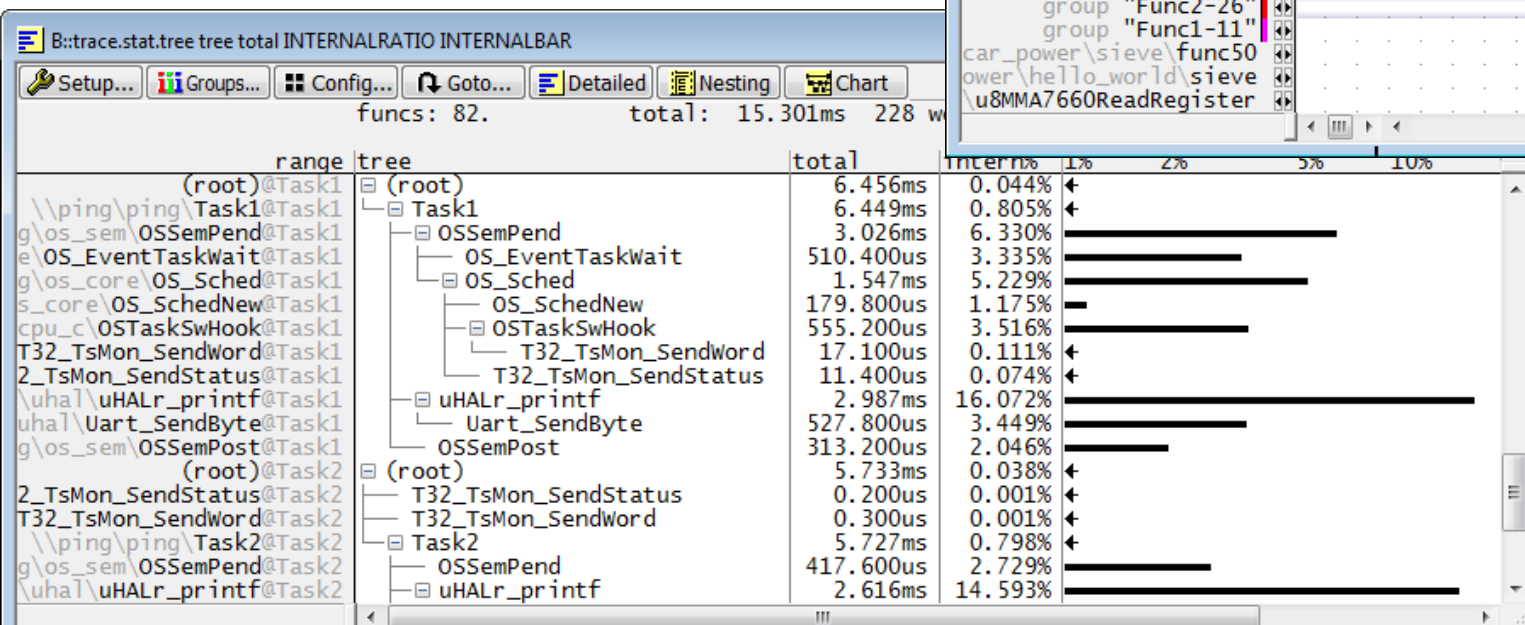
Task State Timing



Misure di durata e
analisi function
nesting



Statistiche e profiling



3) Qualificazione: Code Coverage

Il Code Coverage è una misura diretta di quale parte del codice di una applicazione è stato eseguito. «Statement Coverage» e «Condition Coverage» sono due tipi di copertura richiesti dagli standard di certificazione. Con TRACE32 vengono derivati direttamente dal program trace ETM in modo non intrusivo.

coverage	addr/line	source
ok	203	int func11(x) int x; { switch (x) { case 1: x = x+1; x = x*2; return x*x; case 2: return x+x; case 3: return x-x; case 4: x = x+1; x = x*2; return x*x; case 5: break; case 6: return x+x; default: break; } return x; }
never	206	
never	207	
never	208	
never	210	
never	212	
never	214	
never	215	
never	216	
ok	218	
never	220	
ok	222	
ok	224	

Il Coverage può essere analizzato in modo interattivo con diversi livelli di dettaglio: asm, linea, funzione, file, programma

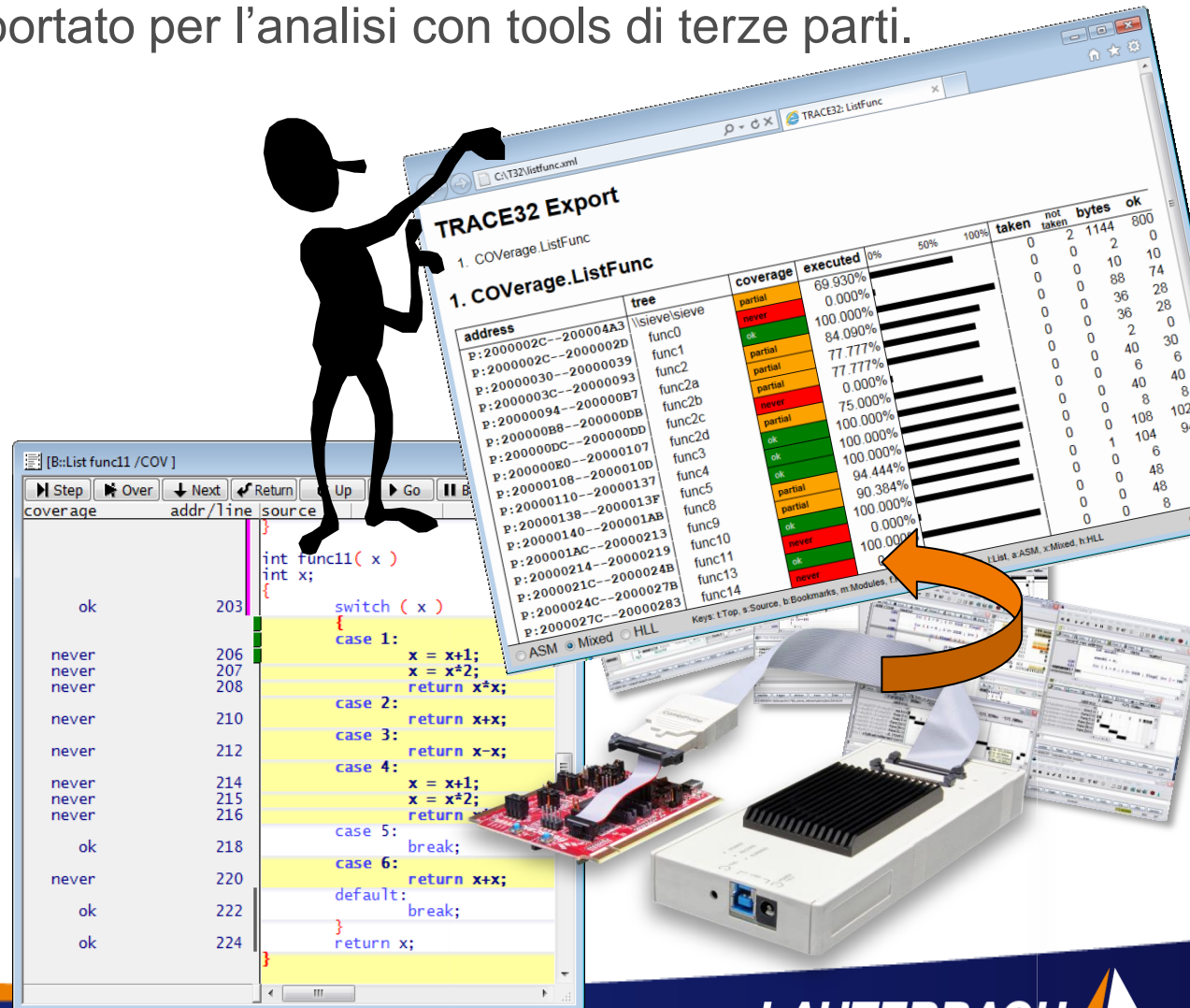
tree	coverage	executed	0%	50%	100%
-1FFF044B	never	0.000%			
-1FFF225D	partial	96.094%			
-1FFF23F3	ok	100.000%			
-1FFF25E1	partial	70.815%			
-1FFF26B5	not exec	94.545%			
-1FFF284F	partial	55.940%			
-1FFF295B	partial	24.137%			
-1FFF2A77	partial	37.301%			
-1FFF2CFB	partial	50.000%			
-1FFF2B41	never	0.000%			
-1FFF2CFB	partial	64.253%			
-1FFF3017	partial	80.769%			
-1FFF360B	partial	24.794%			
-1FFF36AF	never	0.000%			
-1FFF36AF	never	0.000%			

3) Qualificazione: Code Coverage EXPORT

Il Code Coverage, organizzato per file sorgenti, funzioni, linee di codice, può essere esportato in formato XML e analizzato con un comune web browser, convertito in PDF o HTML o esportato per l'analisi con tools di terze parti.

TRACE32 realizza lo «Statement Coverage» e il «Condition Coverage», entrambi basati sul trace.

Lauterbach fornisce kit per la qualificazione di TRACE32 per i test di certificazione secondo gli standard **ISO-26262** e **DO-178**



Agenda del seminario



Marco Ferrario
Lauterbach, Live Demo

- **Documentazione & news**
- **TRACE32 tools per ARM Cortex ARMv7 & ARMv8**
- **Debug & trace support NXP S32**
- **Multicore debug & trace support NXP iMX8**
- **Live demo iMX8**
- Supporto ai sistemi operativi e alla virtualizzazione
- Live demo SMP Linux i.MX8
- Conclusione, Q&A



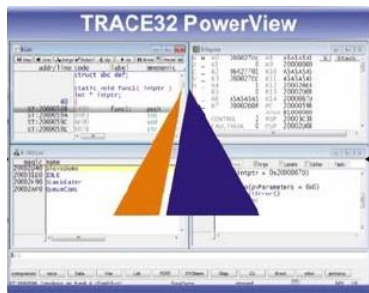
Live demo i.MX8



Marco Ferrario
Lauterbach, Live Demo

Focus sulle key-feature:

- ✓ Sistema PowerTrace JTAG + trace ETR
- ✓ Eval board i.MX8QM QUAD
- ✓ Debugger ARMv8 64 bit
- ✓ Configurazione multicore mista AMP + SMP
- ✓ Synch Debug & trace



TRACE32® PowerView
as a TCF Agent

USB/
NET



Agenda del seminario



Maurizio Menegotto
Lauterbach, relatore

- **Documentazione & news**
- **TRACE32 tools per ARM Cortex ARMv7 & ARMv8**
- **Debug & trace support NXP S32**
- **Multicore debug & trace support NXP iMX8**
- **Live demo iMX8**
- **Supporto ai sistemi operativi e alla virtualizzazione**
- **Live demo SMP Linux i.MX8**
- **Conclusione, Q&A**



Sistemi operativi multicore e virtualizzazione

Tipicamente con i sistemi multicore vengono impiegati dei sistemi operativi capaci di gestire il parallelismo, oppure si utilizzano diversi sistemi operativi su più core.

In base al tipo e ai requisiti di una applicazione, ed al livello di sicurezza che si deve raggiungere, si possono scegliere diverse soluzioni:

Sistema operativo SMP

Un kernel SMP gestisce tutti i core suddividendo tra loro il carico elaborativo.

Es. *SMP Linux* o *VxWorks SMP*



Sistemi operativi multipli

Diversi sistemi operativi eseguono in parallelo sui diversi core.

Es. *Linux + FreeRTOS*,
Linux + ERIKA RTOS
([workshop 2015](#))

Virtualizzazione

Un Hypervisor esegue diversi OS «guests» dedicati a funzioni specifiche.

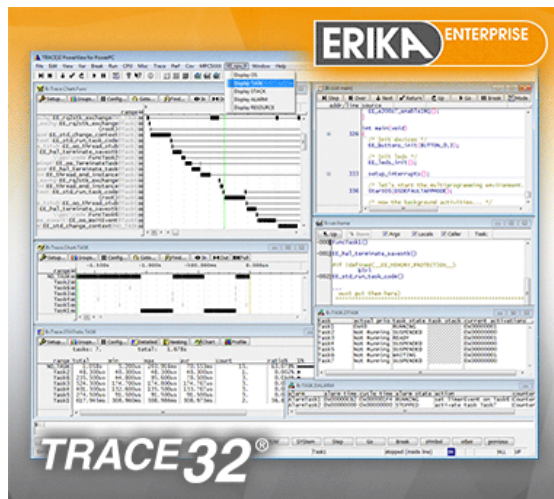
Es. *KVM*, *Xen*, *Jailhouse*,
Windriver Hypervisor,
SYSGO PikeOS

➔ E' necessario che il debugger supporti più sistemi operativi, **simultaneamente**

Supporto ai sistemi operativi e alla virtualizzazione

Lauterbach supporta oltre [30 RTOS/Kernel](#) sia free che commerciali.

TRACE32 è stato scelto da molte importanti aziende come il sistema di debug di riferimento per i loro sistemi operativi, tra queste:



[Press:](#) Lauterbach and **Evidence** collaborate on OSEK VXD tool chain



[Press:](#) Lauterbach and **Wind River** join strategic partnership



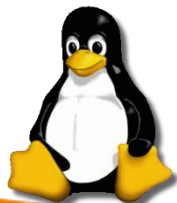
[Press:](#) TRACE32 support **SYSGO's** PikeOS and ELinOS (linux)

Per approfondimento vedi [seminario 2016](#) dal titolo:

**Multi-OS and multi-core debug&trace
for automotive ARM Cortex microcontrollers**

Supporto ai sistemi operativi

Tra i sistemi operativi supportati da TRACE32 e che possono essere di interesse per applicazioni automotive segnalo:



- RTOS OSEK ORTI, come ERIKA di Evidence
- VxWorks e Wind River Linux
- SysGO PikeOS
- FreeRTOS
- Android
- QNX 7
- Linux



Linux SMP sarà il kernel multicore impiegato per la demo TRACE32 con i.MX8

→ Doc. USB Flash Disk





Supporto a Linux

Il supporto a Linux offerto da TRACE32 è molto sofisticato e ricco di funzionalità. L'estensione TRACE32 per Linux consente il debug e il trace di ogni componente di un sistema linux: uboot, kernel, moduli, librerie dinamiche, processi e threads. Include il supporto per SMP Linux (multicore), le estensioni RT Linux, device tree blob.

Approfondimento

Su questo argomento abbiamo tenuto due webinar, sul debug e trace di sistemi Linux.

Le slides ed i link ai video sono disponibili qui:

www.lauterbach.com/tut-i_linux.html

www.lauterbach.com/tut-i_linux_trace.html

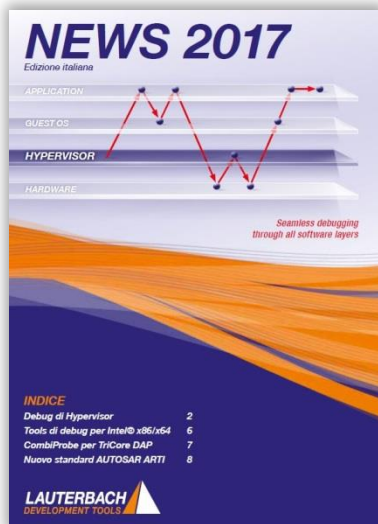


Supporto alla virtualizzazione

Il concetto di virtualizzazione permette a più sistemi operativi di girare in parallelo su una sola piattaforma hardware.

L'**hypervisor** è il nucleo software che consente la virtualizzazione.

La virtualizzazione è sempre più utilizzata nei sistemi embedded, è necessario che il TRACE32 supporti sempre più hypervisors.



➔ Nella newsletter 2017 viene presentata la nuova implementazione del supporto TRACE32 per il [debug di Hypervisors](#).



Supporto alla virtualizzazione

E' ora possibile configurare in TRACE32 una estensione per l'hypervisor, che è l'«host os» e una o più estensioni per i «guest os».

Il sistema di indirizzamento della memoria in TRACE32 viene quindi esteso per poter indicare lo **<space_id>** che identifica gli indirizzi logici di un task, e il **<machine_id>** che identifica il sistema operativo virtualizzato:

< NEW > Hypervisor Debugging	Data.dump	<machine_id>:::<space_id>::<virtual_address>
	Data.LOAD.Elf <file>	<machine_id>:::<space_id>::<virtual_address>
	Register.view	/MACHINE <machine_id> /TASK <process_name>
	Frame.view	/MACHINE <machine_id> /TASK <process_name>

TRACE32 si occupa dell'intero processo di conversione degli indirizzi virtualizzati-logico-fisici e del supporto MMU e di tutti i core in modo da offrire all'utente una vista completa del sistema e di ogni singolo OS, processo, contesto.

➔ **Approfondimento** [Hypervisor debugging article](#)



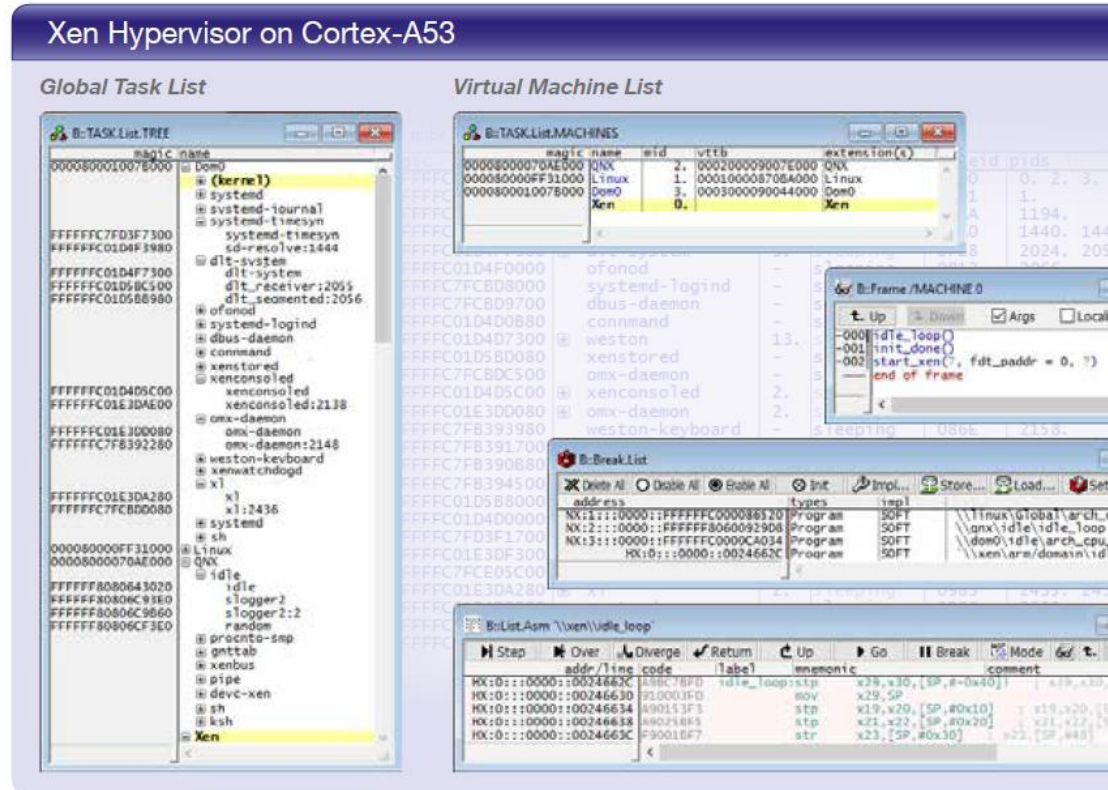
Hypervisors supportati

Il supporto a nuovi Hypervisors è in continua evoluzione, al momento sono disponibili estensioni TRACE32 per:

- ✓ Xen
- ✓ KVM
- ✓ VxWorks 653 v3 [link](#)
- ✓ Wind River Hypervisors 2.x [link](#)
- ✓ PikeOS [link](#)
- ✓ Jailhouse
- ✓ QNX Hypervisor [link](#)

Altri Hypervisor possono essere supportati su richiesta degli utenti.

➔ L'estensione per gli Hypervisor, come quella per gli OS/Kernel, è gratuita e inclusa con il normale software TRACE32.

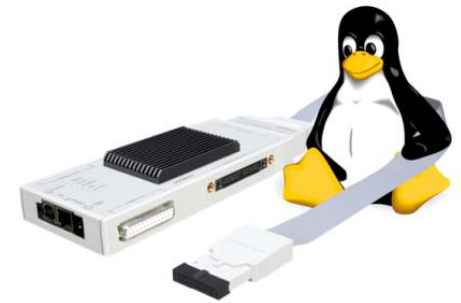


Agenda del seminario



Marco Ferrario
Lauterbach, Live Demo

- Documentazione & news
- TRACE32 tools per ARM Cortex ARMv7 & ARMv8
- Debug & trace support NXP S32
- Multicore debug & trace support NXP iMX8
- Live demo iMX8
- Supporto ai sistemi operativi e alla virtualizzazione
- Live demo SMP Linux i.MX8
- Conclusione, Q&A



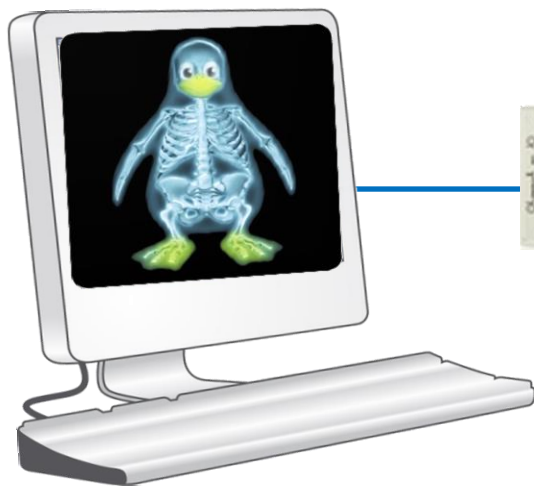
Live demo SMP Linux i.MX8



Marco Ferrario
Lauterbach, Live Demo

Focus sulle key-feature:

- ✓ Linux SMP
- ✓ Debug big.LITTLE su 6 core: 4x A53 + 2x A72
- ✓ Debug multicore 64bit
- ✓ Task trace su 6 core



Agenda del seminario

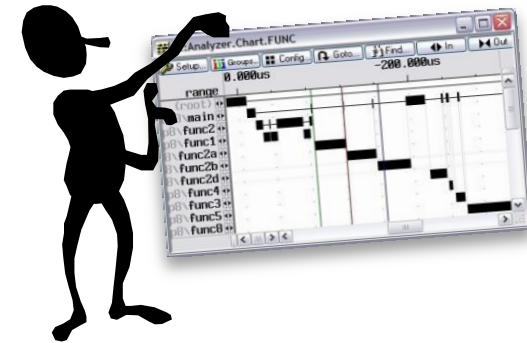
- Documentazione
- Multicore debug & trace ARM Cortex
- TRACE32 TCF-Agent per Eclipse
- Demo TRACE32 integrato con Eclipse, iMX6 QUAD by NXP
- Supporto ai sistemi operativi e alla virtualizzazione
- Demo hypervisor multi-os support, PikeOS by SYSGO
- **Conclusione, Q&A**



Conclusione

Lauterbach produce da circa 38 anni i sistemi di debug e trace più avanzati.

Un sistema Lauterbach TRACE32 è lo strumento che permette di «vedere» cosa accade realmente durante l'esecuzione della vostra applicazione.



Per il vostro progetto TRACE32 va considerato come:

- ➔ Lo strumento per abbattere i tempi di sviluppo e debug
- ➔ La garanzia di scoprire e risolvere rapidamente i problemi sw

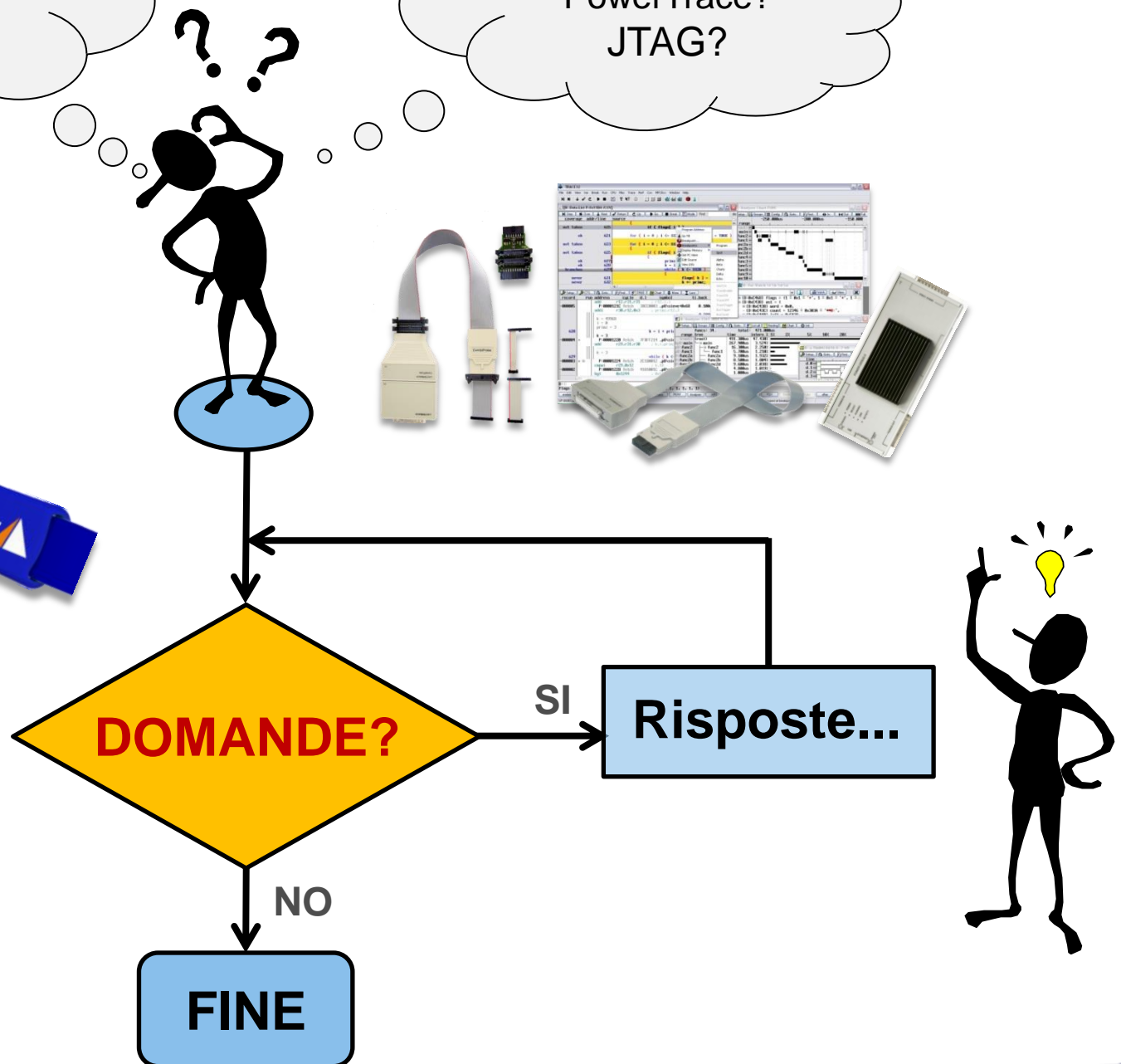
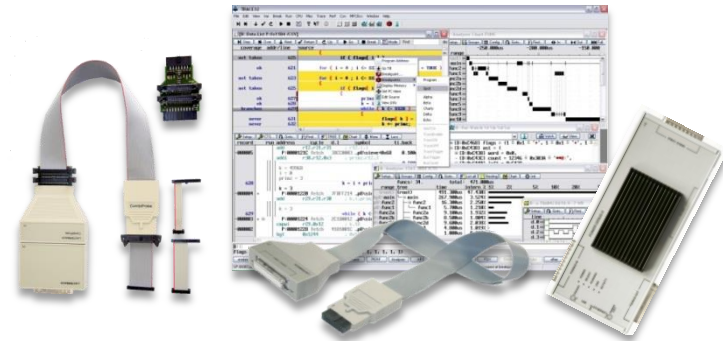
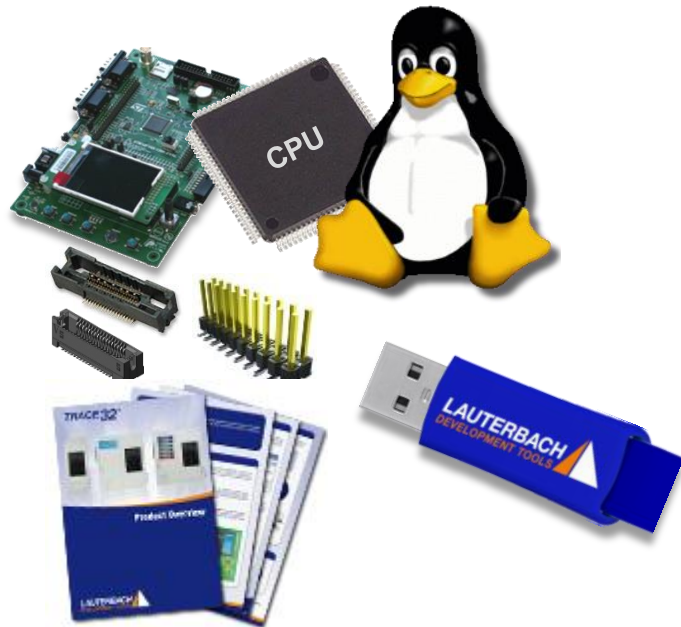


TRACE32 è utile non solo per sviluppo e debug, ma anche per velocizzare i test del software, necessari per le certificazioni safety critical.

Q&A...

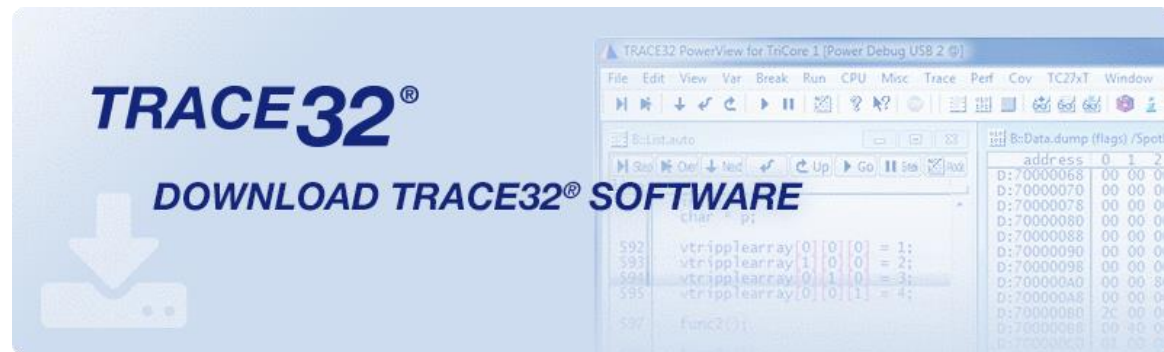
Kernel? ARMv8?
Hypervisors?
Multicore?

Debug? Trace?
PowerTrace?
JTAG?



News 2017 – nuovo software TRACE32

- ➔ A breve verrà rilasciata la nuova release software TRACE32, la **R.2017.09**. Questa release è attualmente disponibile in beta per il test (su richiesta)
- ➔ E' possibile sia il download di una installazione completa per windows, per mac-os o per linux, che il download di aggiornamenti specifici per architettura



News 2017 – aggiornato TRACE32 linux/Qt

L'ambiente di sviluppo TRACE32 PowerView è disponibile per PC Windows, linux MacOS-X e workstation Unix, sia per sistemi 32 bit che 64 bit.

Per linux TRACE32 è disponibile sia con la vecchia GUI **Motif** che la nuova GUI **Qt** con grafica più moderna, molto simile alle versioni per Windows ma più personalizzabile e con nuove funzionalità.

Nuova GUI Linux Qt

Vecchia GUI Linux Motif

The image displays two side-by-side screenshots of the TRACE32 PowerView for TriCore development environment. The left window shows the older Motif GUI, which has a more traditional, less polished look. The right window shows the newer Qt GUI, which is more modern and feature-rich.

TRACE32 PowerView for TriCore (Qt GUI) Details:

- File Edit View Var Break Run CPU Misc Trace Perf Cgv TC27x Window Help**
- Step Over Next Return Up Go Break Mode Find: ..task.c**
- addr/line code label mnemonic comment**

P:70001422	1000F00B		add	d1,d0,d15	; d1,i,primz
P:70001426	DA3C		j16	0x7000143A	
P:70001428	F7000091		movh.a	a15,#0x7000	
P:7000142C	1030FFD9		lea	a15,[a15]0x70	
P:70001430	F600F101		addsc.a	a15,a15,d1,#0x0	; a15,a15,k,#0
- B::List.auto**
- B::Register.view /sl**
- B::Trace.Chart.syr**
- emulate trigger devices trace Data Var List PERF other previous**
- stopped at breakpoint**

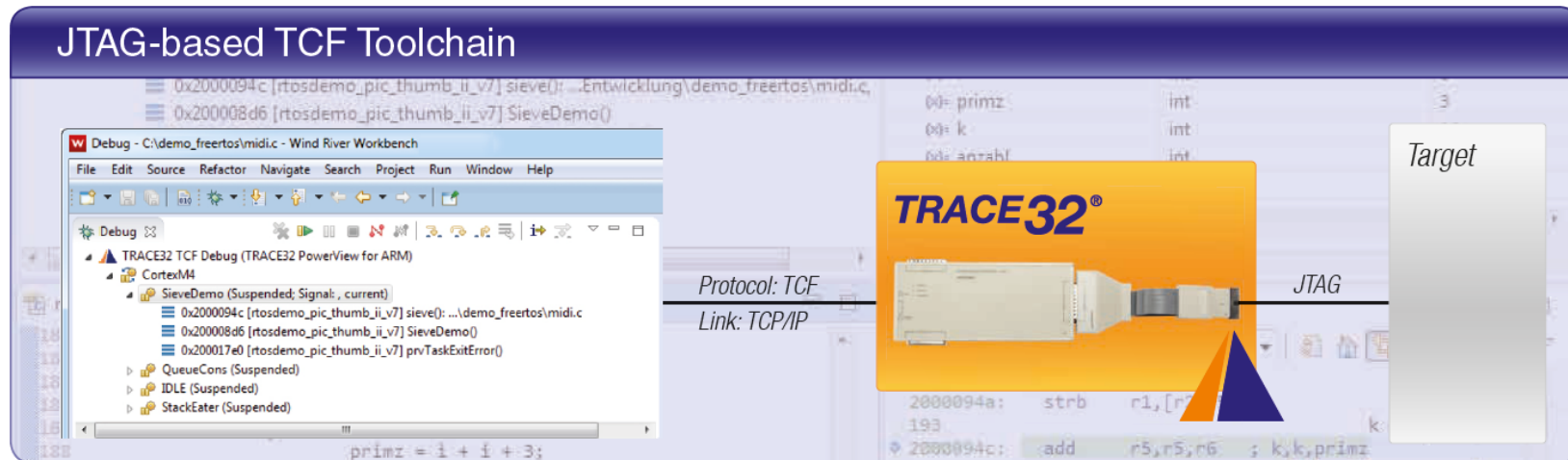
TRACE32 PowerView for TriCore (Motif GUI) Details:

- File Edit View Var Break Run CPU Misc Trace Perf Cgv TrICore Window**
- Step Over Next Return Up Go Break Mode Find:**
- addr/line code label mnemonic comment**

P:D4000854	F0000091		movh.a	a15,	
P:D4000856	8123FFD9		lea	a15,	
P:D400085E	F240		mov16.a	a2,a	
P:D4000860	01DA		mov16	d15,	
- B::Data.List**
- B::b.set**
- emulate trigger devices trace Data Var List PERF other previous**
- stopped at breakpoint**

TRACE32 TCF-Agent, integrazione con Eclipse

TRACE32 può operare come TCF Agent. Questo rende possibile l'utilizzo di ambienti di sviluppo e debug basati su Eclipse, TRACE32 svolge il ruolo di debugger back-end per la connessione al target ed il suo controllo.



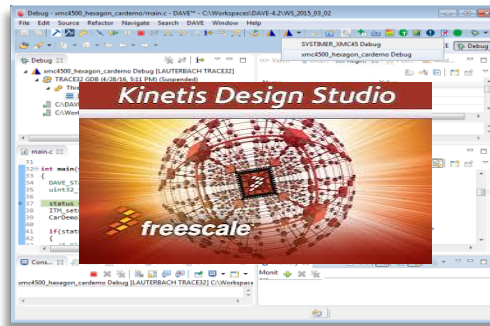
- TRACE32 agisce come TCF agent (Target Communication Framework)
- Applicabile a tutti i sistemi TRACE32 e per tutte le architetture supportate
- Sincronizzazione viste tra PowerView e C/C++ debugger in Eclipse (anche ASM!)
- Sincronizzazione go/break/step e breakpoint
- Right-click→edit in TRACE32 apre il sorgente in edit in Eclipse (**novità!**)
- Supporto multi-core e multi-progetto

Integrazione con Eclipse, perchè è importante?

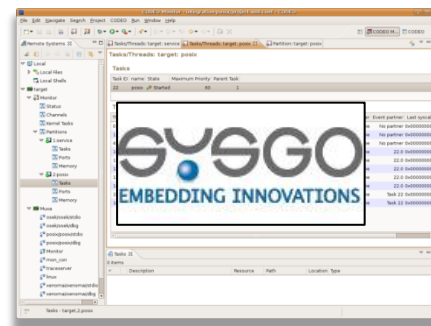
Perché Eclipse è sempre più utilizzato da molti ambienti di sviluppo embedded



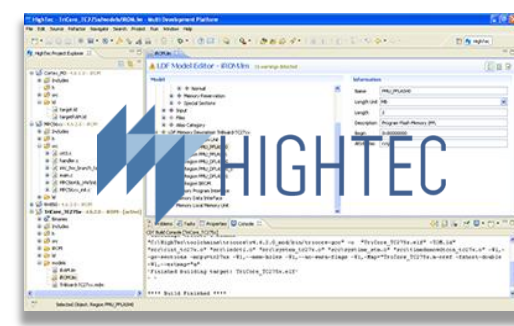
Wind River Workbench



NXP Kinetis Design Studio



SYSGO Codeo for PikeOS



Hightec compiler IDE

➔ L'integrazione semplifica la gestione del progetto e unisce il meglio dei due ambienti di debug: TRACE32 può essere usato sia con Eclipse che con PowerView.



Eclipse IDE

Protocol: TCF
Link: TCP/IP



TRACE32® PowerView
as a TCF Agent

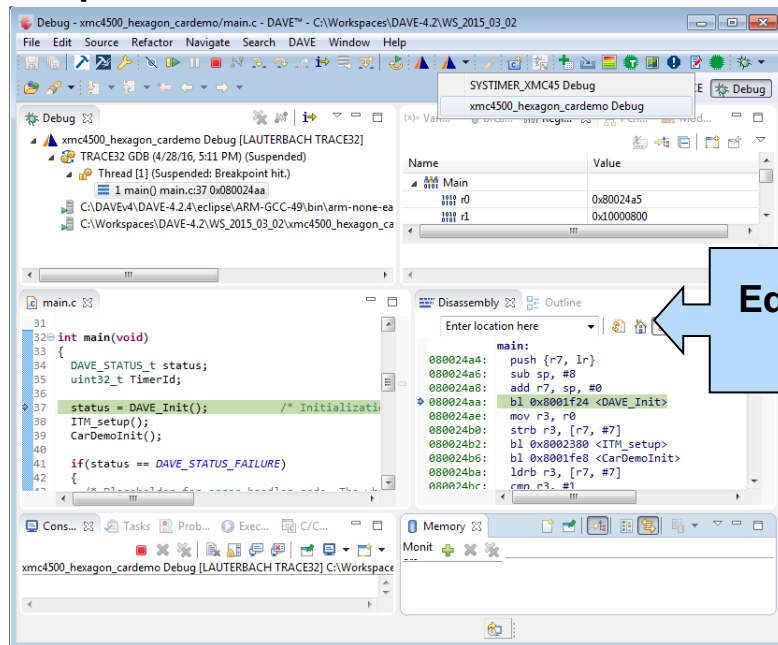
USB/
NET



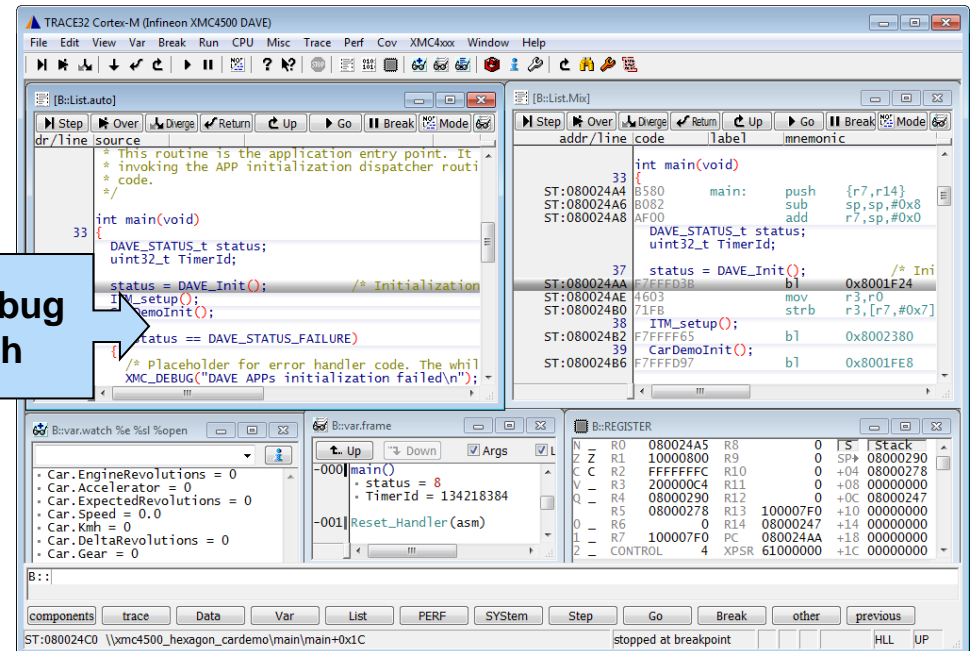
TRACE32® Hardware

TRACE32 TCF-Agent, integrazione con Eclipse

Eclipse IDE



TRACE32 PowerView



Edit & Debug
in synch



➔ Per approfondimenti rimando alla [Newsletter 2016](#), pag.7

➔ Installazione: www.lauterbach.com/eclipse/tcf/

➔ manuale: [app_tcf_setup.pdf](#)

Debug & trace tools for new NXP i.MX8 and S32 automotive processors

Lauterbach Italia:

Maurizio Menegotto, maurizio.menegotto@lauterbach.it

Marco Ferrario, marco.ferrario@lauterbach.it

Tel. +39 02 45490282 web www.lauterbach.com

Documentazione web:

- ✓ Lauterbach Company: www.lauterbach.com/profile.html
- ✓ Lauterbach Products: www.lauterbach.com/products.html
- ✓ TRACE32 PowerView: www.lauterbach.com/powerview.html