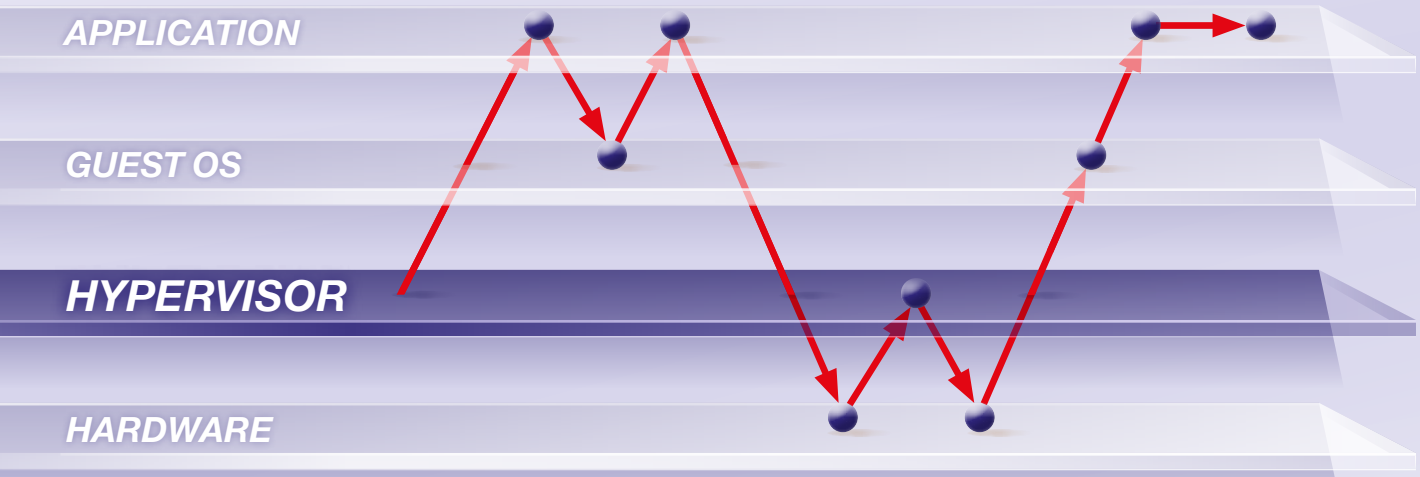
NEWS 2017

Edition Française

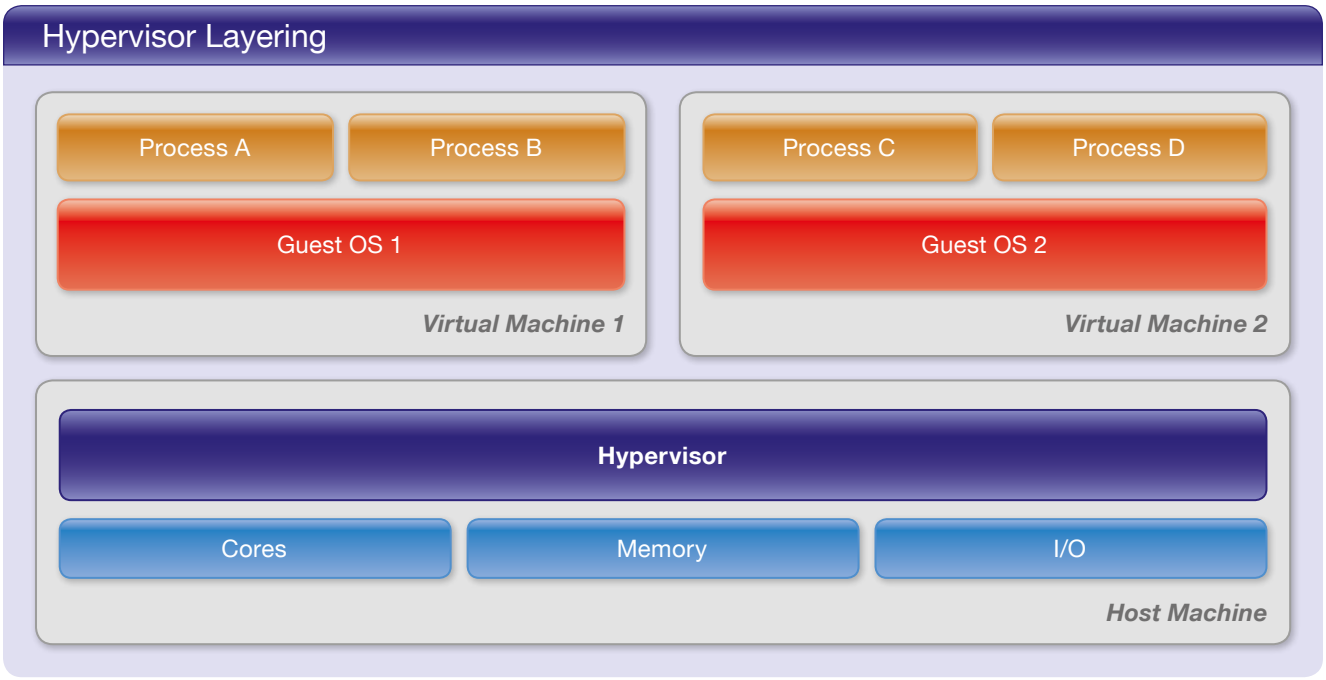


Seamless debugging through all software layers

Sommaire

LAUTERBACH
DEVELOPMENT TOOLS

Débugge Hyperviseur



En Avril 2017, Lauterbach dévoilera enfin les énormes performances de débugge, de son tout nouveau support pour Hyperviseur. Cet article présente une implémentation de référence dans lequel un hyperviseur Xen exécute deux instances linux sur une carte HiKey de chez LeMaker (Cortex-A53).

Virtualisation dans un System Embarqué

Le concept de virtualisation permet à plusieurs OS de s'exécuter en parallèle sur une plateforme. Aujourd'hui, la virtualisation est très utilisée, notamment dans les systèmes embarqués. L'exemple du cockpit d'une voiture : L'application temps-réel utilisant un OS AUTOSAR, s'exécute en parallèle d'une IHM basée sur Android. L'hyperviseur, qui est le cœur du système de virtualisation, s'assure que tout fonctionne de manière fiable et efficace.

L'Hyperviseur surveillant les VMs, est une couche logicielle remplissant entre autre deux tâches :

1. Démarrer et gérer les machines virtuelles (VMs),
2. Virtualiser les ressources matériels pour les VMs.

Un OS s'exécutant sur une VM est représenté comme un OS invité. L'invité accède aux ressources matérielles virtualisées, mappées physiquement par l'hyperviseur.

Virtualiser un CPU est important pour le débugge. Chaque VM est assignée à un ou plusieurs CPUs virtuels (vCPUs). Le nombre de vCPUs n'a pas besoin nécessairement d'être identique aux nombres de CPU physiques disponibles sur la plateforme.

Virtualiser la mémoire est primordial. Les VMs perçoivent la mémoire physique invité comme une mémoire virtualisée. L'hyperviseur gère une table de page pour chaque VM contrôlant l'accès à la mémoire physique. Les processus, sur des OS comme linux, fonctionnent avec des adresses virtuelles ; Nous avons donc à gérer deux étages de translation d'adresse :

- Mémoire virtuelle invité vers mémoire physique invité
- Mémoire physique invité vers mémoire physique host

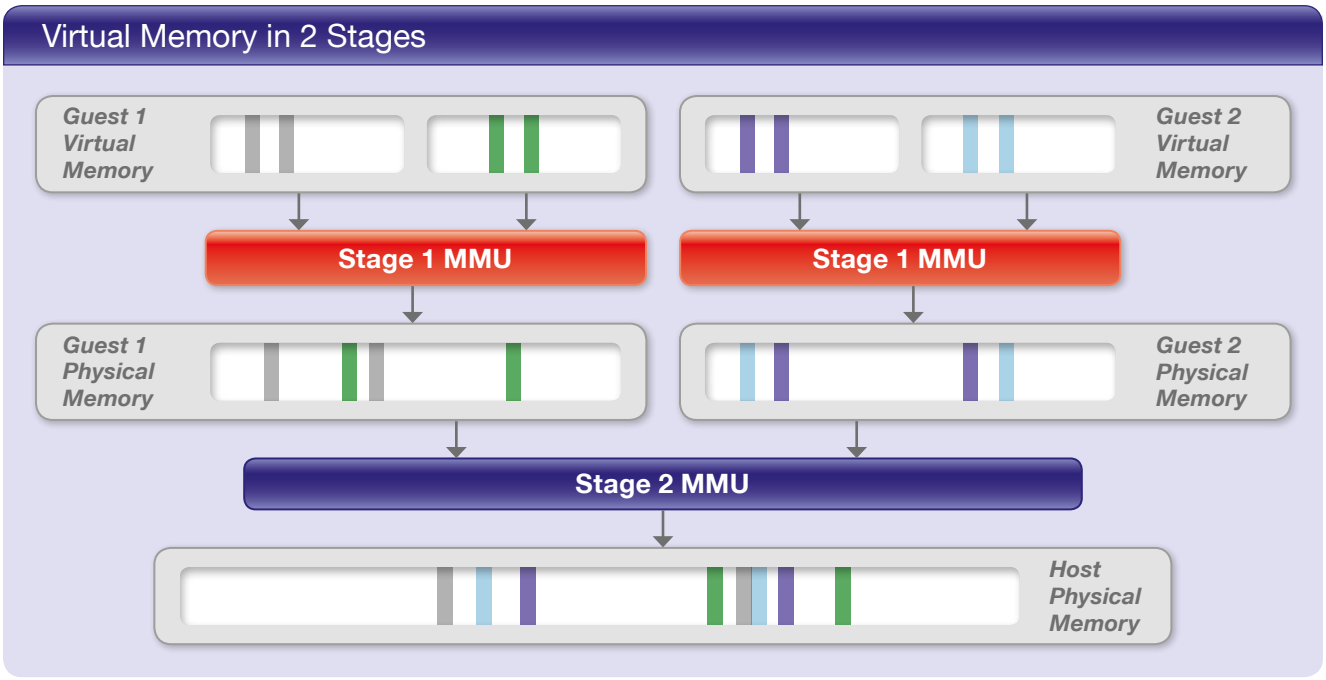
Voir le diagramme « Mémoire Virtuel à 2 étages » sur la page opposée :

- 1er étage : les tables de page des OS invités fournissent les informations du mapping MMU
- 2ème étage : la MMU utilise les tables de page de l'hyperviseur

Concepts de débugge étendu

TRACE32 a évolué tout au long de l'année 2016 afin de permettre à ses clients d'obtenir des capacités de débugge illimitées avec un hyperviseur. Les extensions suivantes ont été ajoutées :

- Un ID machine a été ajouté à la syntaxe de commande



TRACE32. L'ID machine permettra un accès au contexte de la VM active mais aussi au contexte de toutes les VMs inactives. Une machine virtuelle est considérée comme active lorsque qu'un cœur est alloué à son exécution.

- Utilisant le support de l'hyperviseur, le débuggeur détecte et affiche les VMs de l'hyperviseur.
- Il n'était possible de débugger qu'un seul OS. Désormais nous permettons d'en débugger plusieurs.
- Auparavant il était possible d'accéder uniquement aux tables de page de l'OS invité actif. Désormais le débuggeur peut accéder à toutes les tables de page de tous les OS invités inactifs.

L'objectif de cette extension est le débugge d'un système complet. Lorsque le système est stoppé sur un point d'arrêt, on peut vérifier et changer l'état de chacun des processus, de toutes les VMs, de l'hyperviseur ainsi que de la plateforme. De plus, vous pourrez poser un point d'arrêt sur n'importe quelle zone du code.

Point de départ

Les possibilités de débugge illimité qu'offrent les outils Lauterbach depuis plus de 20 ans pour des OS comme Linux, ont fourni le point de départ pour l'implémentation de ces extensions. Voici, un bref aperçu des concepts de débugge les plus importants :

Les Processus s'exécutent dans un OS et dans un espace d'adressage privé.

Le support des OS TRACE32 et le support de la MMU de TRACE32 permet aux utilisateurs de débugger de manière transparente à travers les limites qu'impose un processus :

- Grâce au space ID, on accède à l'espace d'adressage virtuel de chaque processus.
- Grâce à l'option "TASK", on affiche l'état des registres et de la pile d'appel de chacun des processus.

L'ID Machine

Comment peut-on étendre ce concept si l'OS s'exécute dans une machine virtuelle ?

1. Il est nécessaire d'identifier de manière unique chaque VM. Pour accomplir cette tâche, TRACE32 assigne à chaque VM un nombre appelé ID machine. Celui de l'hyperviseur est 0. Le Space ID est utilisé pour identifier l'espace d'adressage virtuel d'un processus, l'ID machine lui est utilisé pour identifier l'espace d'adressage privé d'une VM.
2. Pour afficher l'état des registres et de la pile d'appel de n'importe quel processus, le débuggeur doit savoir sur quel VM et sur quel OS invité le processus s'exécute. L'option MACHINE a été développée pour répondre à ce besoin.

Ces deux extensions sont suffisantes pour permettre au débuggeur d'accéder à toutes les informations traversant les limites des processus. L'aperçu des

TRACE32 Commands

Traditional OS-Aware Debugging	`Data.dump`	`<space _ id>:<virtual _ address>`
	`Data.LOAD.Elf <file>`	`<space _ id>:<virtual _ address>`
	`Register.view`	`/TASK <process _ name>`
	`Frame.view`	`/TASK <process _ name>`
< 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>`

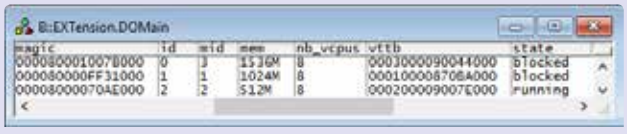
commandes TRACE32 qui précède, fourni une comparaison entre la syntaxe de commande concernant le débugge hyperviseur et la syntaxe de commande traditionnelle pour le débugge des OS.

Support des Hyperviseurs

- Comme les fonctionnalités de support des OS, il existe désormais les fonctionnalités de support des hyperviseurs. Ces fonctionnalités fournissent au débuggeur toutes les informations sur l'hyperviseur s'exécutant sur la plateforme matériel. Cependant le support de l'hyperviseur exige que les symboles de débugge soient chargés. Le débuggeur peut alors créer une vue globale de tous les Guest. La copie d'écran "Guest List" de notre implémentation de référence — Xen, Cortex-A53 montre les informations suivantes :
- VM IDs et états des VM, nombres de vCPUs par VM
- Adresse de départ de l'étage 2 de la table de page (vttb)

Les fonctionnalités de support pour un hyperviseur spécifique sont développées par Lauterbach pour être fournies à ses clients. Un aperçu de tous les hyperviseurs supportés est fourni dans le tableau "Currently Supported Hypervisors" en page 5.

Guest List

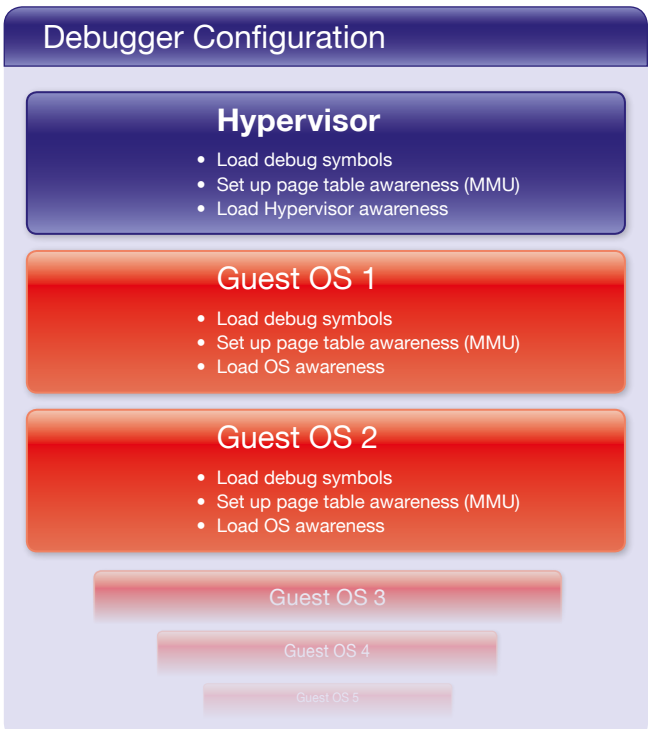


Configuration du Débuggeur

Comment ce concept d'extension affecte le débugge avec TRACE32. Commençons par regarder la configuration. Les étapes suivantes sont nécessaires pour configurer un hyperviseur de même qu'un OS Guest seul :

1. Charger les informations de débugge
2. Configurer les tables de pages (MMU)
3. Charger le support hyperviseur de TRACE32 et ensuite le support OS de TRACE32

Debugger Configuration

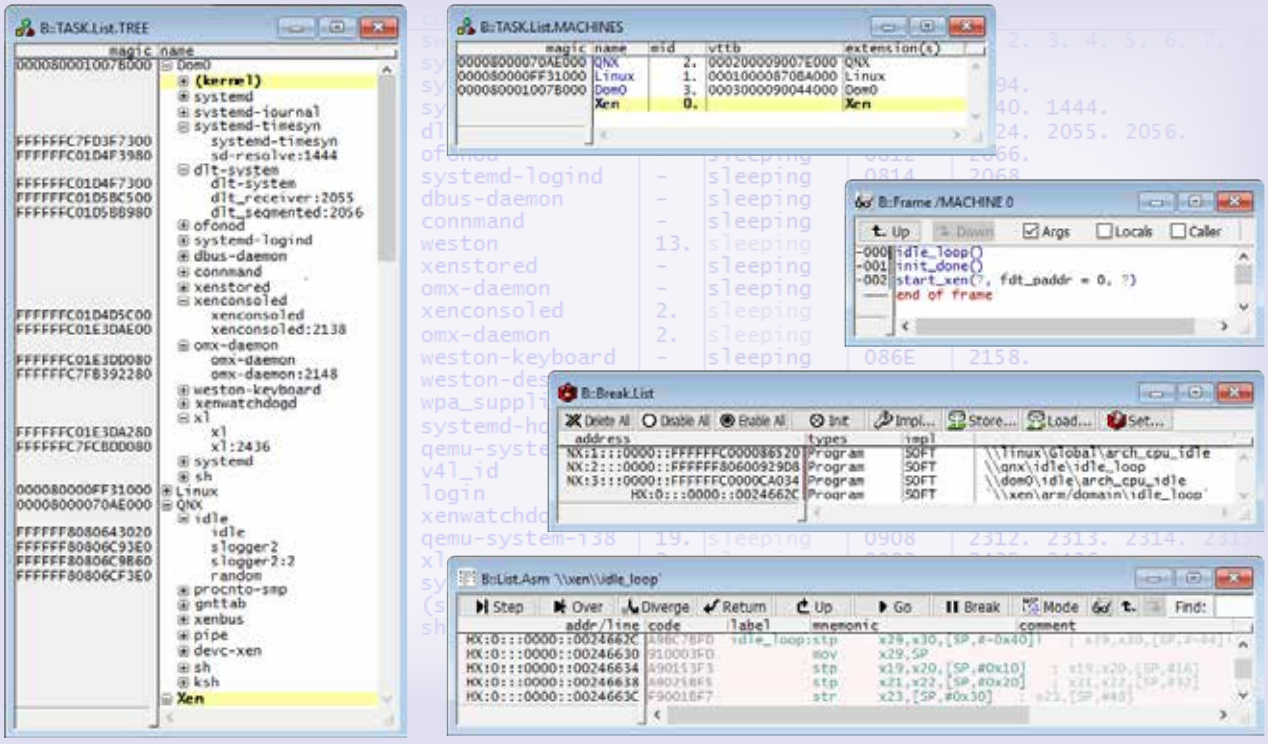


LAUTERBACH DEVELOPMENT TOOLS

Xen Hypervisor on Cortex-A53

Global Task List

Virtual Machine List



Le diagramme “Débugger Configuration” montre un aperçu de chaque étape de configuration.

Débugger un Processus

Pour ce type d’opération, un débuggeur doit souvent résoudre des besoins contradictoires. Certains utilisateurs demanderont une utilisation simple et intuitive pendant que d’autres voudront un maximum de flexibilité et un scripting complet. Regardons en premier lieu l’utilisation la plus intuitive. L’idée de base est actuellement très simple : si le débuggeur s’arrête sur un point d’arrêt, alors l’interface graphique affiche le processus applicatif qui est à l’origine du point d’arrêt.

Si vous êtes intéressé par un processus applicatif différent, ouvrez simplement la liste globale des tâches dans TRACE32. Toutes les tâches s’exécutant sur l’ensemble du système seront listées à cet endroit. Il suffira de double-cliquer sur la tâche concernée pour qu’elle s’affiche en détail sur l’interface graphique. La liste globale des tâches offre aussi un moyen simple de positionner un point d’arrêt programme directement sur une tâche spécifique. Comme les symboles de débugge sont associés à un ID machine et à un space ID lorsque le fichier .elf est chargé, les fonctions et variables peuvent être accédées directement par leur nom, comme toujours lors d’une session de débugge.
Une flexibilité maximum et une capacité de scripte complète peuvent être obtenues en utilisant les commandes de script TRACE32. La syntaxe étendue pour ce type de commande a été présentée ci-dessus.

Conclusion

Puisque Lauterbach a systématiquement étendu le concept très connu de son support de débugge OS vers le débugge hyperviseur, il sera très facile aux utilisateurs TRACE32 de le mettre en œuvre.

Currently Supported Hypervisors

KVM	Wind River Hypervisor 2.x
VxWorks 653.3x	Xen
PIKEOS SYSGO	*(more to follow)*

Intel® x86/x64 – Mise à jour des outils

Au mois de Janvier de cette année, Lauterbach a présenté son nouveau Combiprobe MIPI60-Cv2. Le Combiprobe TRACE32 et le Quadprobe TRACE32 offrent maintenant les mêmes fonctionnalités de débugge avec l'interface « Converged Intel® MIPI60 ».

- JTAG Standard, Intel® debug hooks avec Pmode, et bus I2C
- Port de débugge fusionné (deux chaines JTAG)
- Fonctionnalités Intel® Survivability (threshold, slew rate, etc…)

Ces deux outils de débugge s'utilisent dans des domaines d'application différents. Le **QuadProbe TRACE32** a été développé pour les processeurs de type serveur, il permet de débugger en mode SMP une centaine de threads sur cible jusqu'à quatre connecteurs JTAG séparés.

Le Combiprobe **TRACE32 MIPI60-Cv2**, supporte les processeurs Intel de type client ainsi que ceux de type mobile, il peut capturer et analyser de la trace système. Il supporte, un port de trace 4 bit et un port de trace 8 bit.

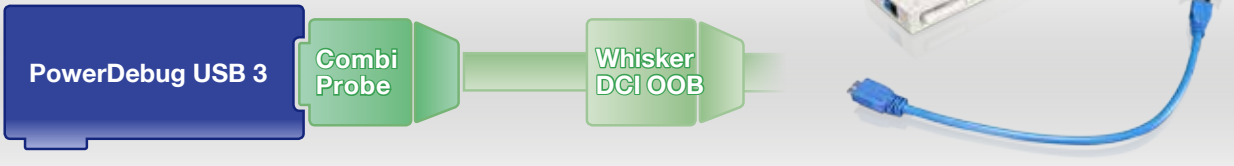
Le **Combiprobe TRACE32 DCI OOB** permet de débugger et tracer des processeurs n'ayant pas de connecteur de débugge. Grâce au système de gestion DCI, nous pouvons échanger des messages de débugge et de trace directement via l'interface USB3. Le protocole DCI, supporte le standard JTAG et le débugge hooks Intel® de même que les messages de trace système.

Debugger and System Trace for Intel® Converged MIPI60 Connector

(devices and client applications)

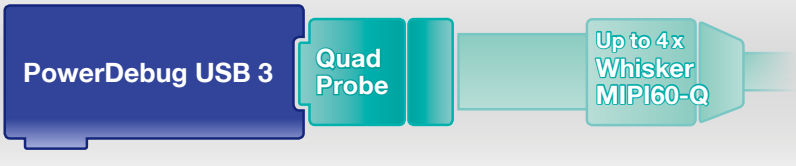
Debugger and System Trace for USB3 Connector

(all applications)



Debugger for Intel® Converged MIPI60 Connector

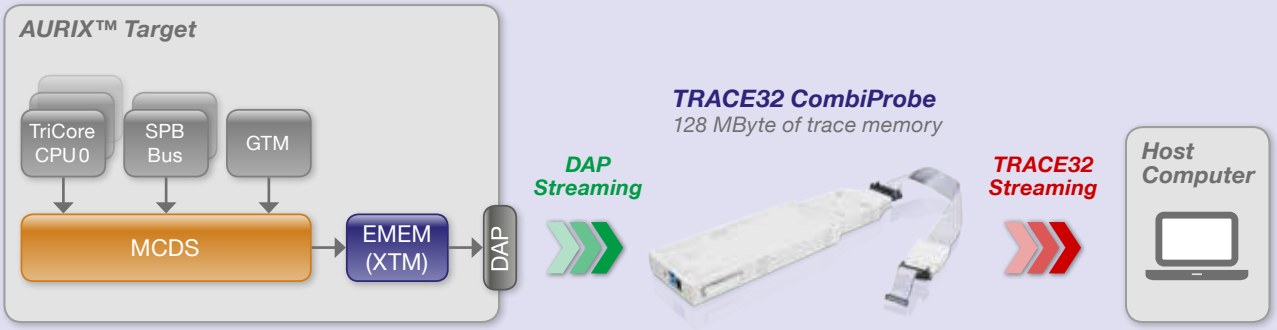
(server)



LAUTERBACH DEVELOPMENT TOOLS

CombiProbe TRACE32 pour TriCore DAP

DAP Streaming and TRACE32 Streaming



Lauterbach propose la nouvelle Combiprobe Tricore DAP pour la famille AURIX™ d'Infineon depuis Octobre 2016. Ce qui veut dire que TRACE32 offre désormais une analyse du run-time compréhensive pour tous les utilisateurs d'AURIX dont les cartes matérielles ne possèdent pas d'interface AGBT.

DAP Streaming

La Combiprobe implémente une nouvelle technologie appelée DAP streaming : le contenu de la mémoire de trace on-chip est accédé à la volée pendant que le programme s'exécute. Il sera ensuite enregistré dans les 128Mo de trace du Combiprobe. L'interface DAP AURIX™ possède les caractéristiques suivantes : Fréquence DAP jusqu'à 160Mhz, bande passante maximum de 30Mo/s. Cette dernière n'est pas suffisante pour transmettre la totalité du code exécuté, mais une analyse étendue peut néanmoins être faite :

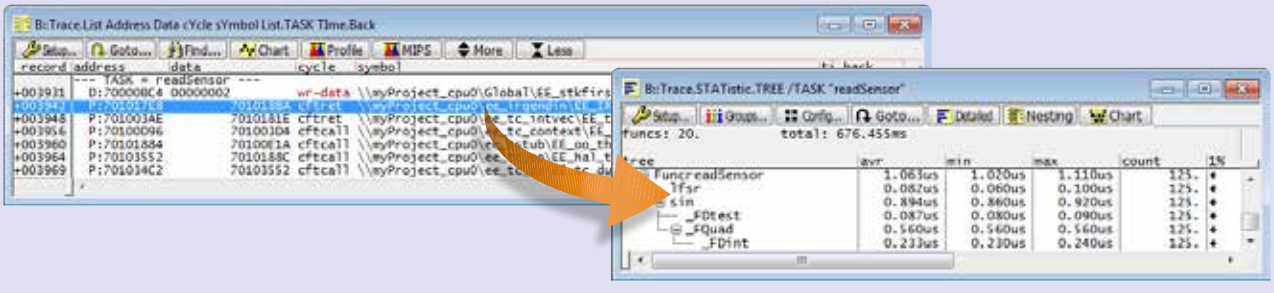
- Mesure du temps d'exécution des fonctions en utilisant la « Compact Function Trace » (CFT). Cette trace est une trace programme spécifique dans le sens ou les messages de trace ne sont générés que sur les appels de fonctions (cftcall) et les retours de fonctions (cftret). Le diagramme ci-dessous de la « Compact Function Trace » montre un exemple de l'arbre d'appel et des détails du temps d'exécution calculé par TRACE32 grâce à cette trace.
- Analyse du contenu d'une variable à travers le temps.
- Mesure des temps d'exécution des tâches, ISR, et services d'un OS.

Streaming TRACE32

Si le buffer de trace de 128Mo du Combiprobe ne suffit pas pour enregistrer toutes les données de trace, il est possible de combiner le streaming DAP avec le streaming TRACE32. Le streaming TRACE32 permet de transférer les données à la volée du combiprobe vers l'ordinateur hôte. Cela permet d'enregistrer en continue plusieurs Tera Octets de données de trace, dans le but d'une mesure sur du long terme.

Vous trouverez plus d'informations sur le sujet sur :
www.lauterbach.com/8467

Compact Function Trace



ARTI – Interface de mesure des temps AUTOSAR

Le nouveau standard ARTI sera spécialement développé pour faire face aux nouvelles spécifications de l'industrie automobile sur le débugge et la trace du support des OS. Lauterbach, en tant que partenaire de développement officiel AUTOSAR, est partie prenante dans le design de ce standard. Les premières publications sont prévues pour début 2018.

Le standard ORTI, qui a été utilisé dans toute l'industrie automobile depuis 2003, permet à des milliers de développeurs de débugger leurs systèmes AUTOSAR. Cette norme, qui a traversé les années, nécessite désormais des mises à jour pour répondre aux dernières exigences.

But

Nouvelles méthodes de développement logiciel, Multicoeur, système multi ECU et accroissement des spécifications sur la validation des systèmes temps réel critique – la nouvelle norme devra couvrir tout cela.

Ces nouvelles méthodes de débugge, de trace et d'analyses de performance, qui doivent être ajoutées au standard ARTI, sont déjà utilisées de nos jours comme des solutions propriétaires. Cela veut dire que ces solutions ont déjà fait leurs preuves. Ce qui manque, c'est une interface standardisée entre les différents outils utilisés dans le processus de développement. Voici un exemple :

Exporter les données de trace

Depuis 2014, Lauterbach a collaboré très souvent et avec différents fabriquants d'outils, utilisés dans la validation et l'optimisation de logiciel dans le domaine automobile. TRACE32 exporte ses données de traces temps réels, qui sont ensuite chargées dans un outil externe pour y être analysées de manière compréhensive. Donc que manque t-il ?

1. Le fichier ORTI créé par l'outil, contient uniquement les informations sur les tâches, les services de l'OS, et les ISRs. Mais aucune information concernant le début et la fin d'exécution d'une tâche ni sur l'état de son exécution. Ces informations seront ajoutées manuellement dans TRACE32 avant son exportation.
2. Il n'existe aucun format standardisé pour ces données de trace. Cela implique que l'outil externe doit être capable de lire le format propriétaire de TRACE32.

Le nouveau standard ARTI couvrira ces deux besoins. Le fichier ARTI fourni par l'outil de build, contiendra toutes les informations sur les objets AUTOSAR tout en standardisant l'exportation des données de trace.

Conclusion

Comme tous les plus importants fabricants d'outils, les utilisateurs travaillent de concert pour écrire le nouveau standard ARTI. Il sera sans nul doute un succès avec très certainement une aussi longue vie que son prédécesseur.