ARM Tech Symposia 2017

京都マイクロコンピュータ株式会社

LLVMを用いた統合開発プラットホームと、
Armv8アーキテクチャのシステム性能の解析

ITRON以上Linux未満を実現する開発環境

SOLID

ベアメタル・RTOS用の
ソフトウェア開発プラットフォーム

バグの自動検出も可能な新開発環境 SOLID

組込みアプリケーションソフトウエア開発の普遍的な課題

- 納期どおりに開発する
- 品質を確保する
- 快適な操作性
- 出来るだけ安価に

これらを実現するための開発環境として生まれたのが、SOLID開発プラットフォームです。

バグの自動検出も可能な新開発環境 SOLID

開発ツールベンダーである京都マイクロコンピュータの着目点

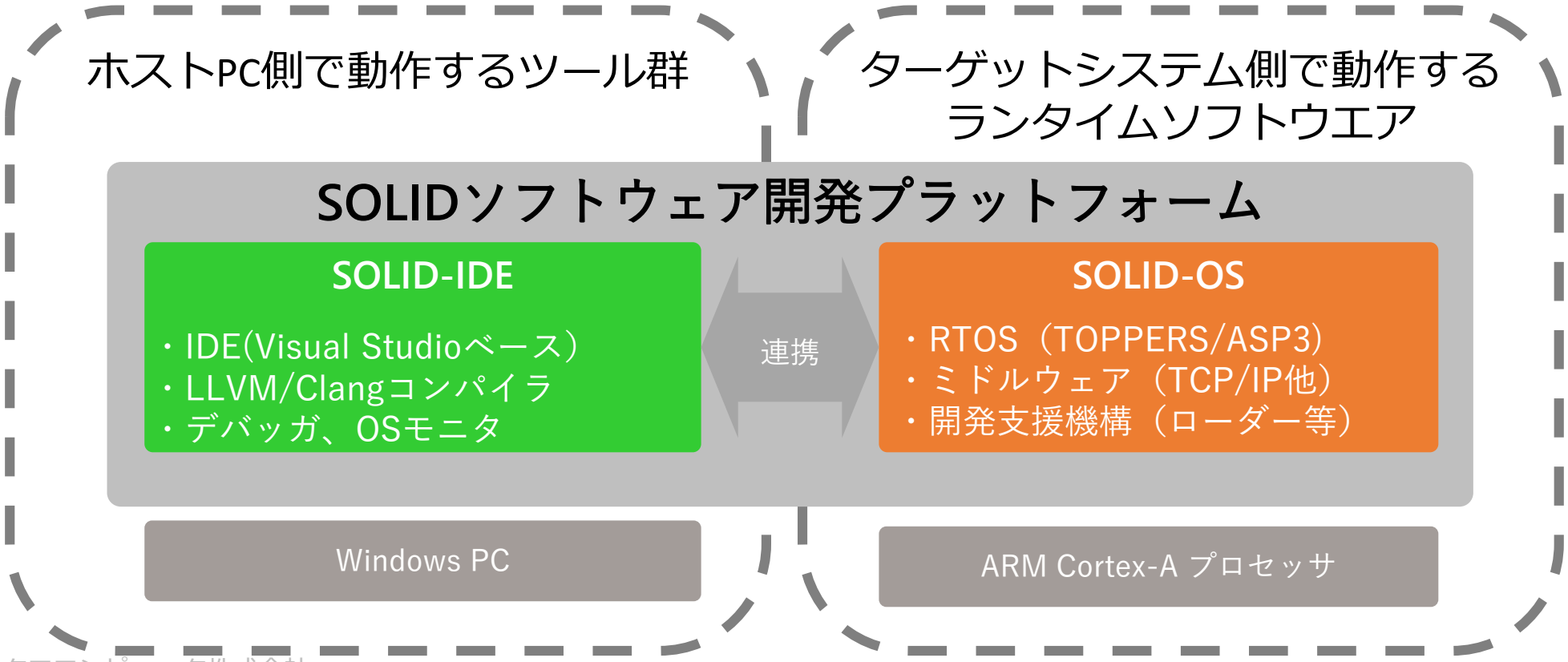
- ITRON開発環境への新提案という切り口で、
- ITRON含むランタイムと開発環境を専用設計
- 構成要素は　コンパイラ、ITRON+ランタイム、IDE、デバッガ

いわゆる、RTOSベースの統合環境が誕生しました。

「これを一つ用意するだけで、開発がスタートできる！」

バグの自動検出も可能な新開発環境 SOLID

SOLIDの構成



バグの自動検出も可能な新開発環境 SOLID

同じようなものは、今までもあったのではないか？

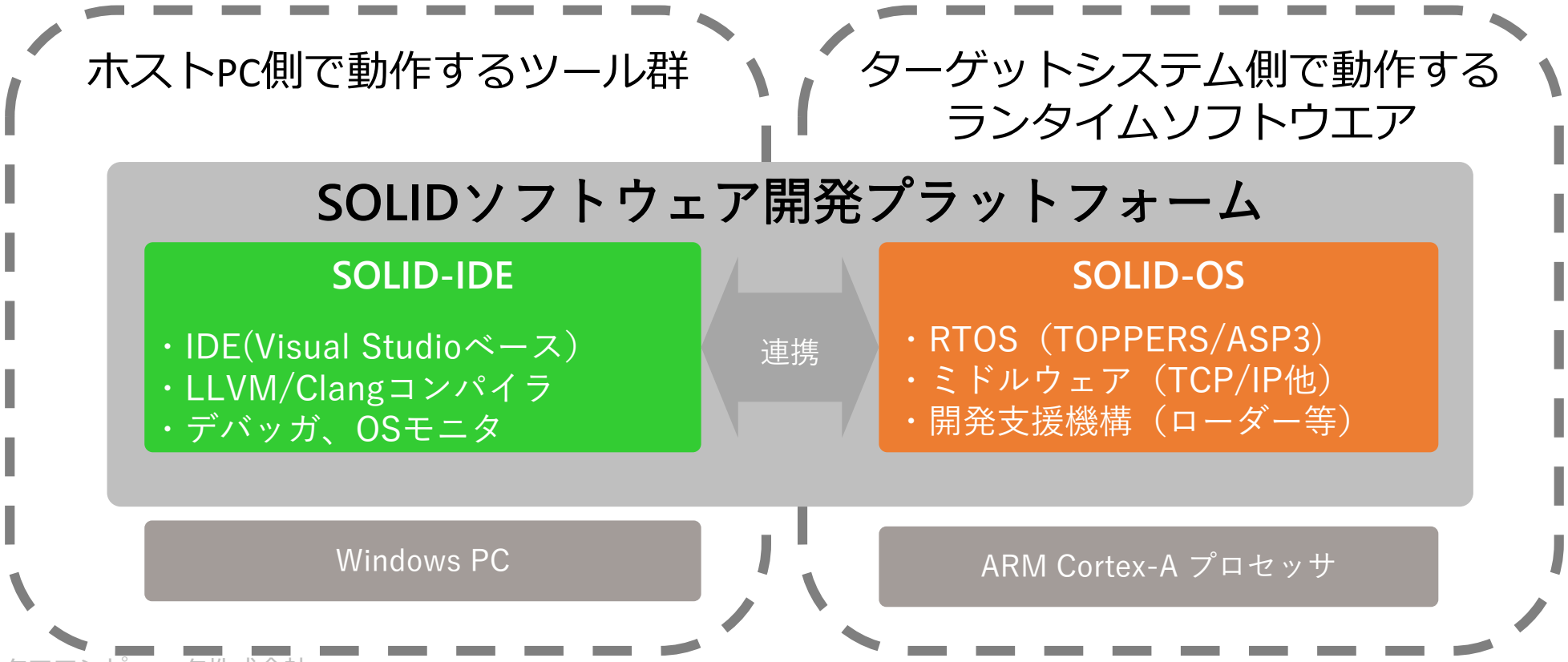
何が新しく、何が違うのか？

バグの自動検出も可能な新開発環境 SOLID

ここが新しい！

SOLIDの構成

ホストPCとターゲット、双方の**ソフトウエアの専用設計**が特徴です



バグの自動検出も可能な新開発環境 SOLID

連携というより、むしろ、各々のツールが専用設計されている

- RTOS実装とコンパイラの専用化
- RTOS実装とデバッガの専用化
- コンパイラとデバッガの専用化
- IDEを、RTOS,コンパイラ,デバッガに合わせて専用化

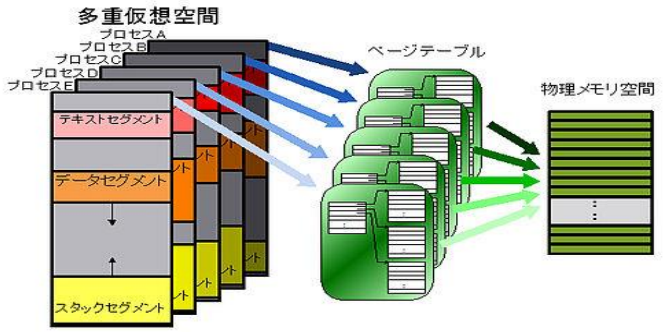
とても排他的だが、その分、使う時には一体化によるメリットが大きく享受される

バグの自動検出も可能な新開発環境 SOLID

MMUの役割：アドレス変換、メモリ保護、キャッシュの設定を簡単に!!

ARM Cortex-A9などV7A/V8A CPUには、高機能なメモリ管理機構(MMU)が搭載されています。
MMUの主な機能は次のとおりです。

- **仮想アドレスと物理アドレスのアドレス変換機能**
- **プロセスのための仮想空間を実現するための機能**
- **実行モードに応じたメモリ保護**



Linux/Androidなどは、この高機能なメモリ管理機能を使って、
多重仮想空間のオーバーラップしたプロセスを物理メモリ空間に
割り付けて実行する事が特徴のオペレーティングシステムです。

ARM V7A/V8Aプロセッサでは、データキャッシュを有効にするためにはメモリ管理機構を設定（MMU有効化）する必要があります。SHマイコンのように、番地によるキャッシュアクセス／非キャッシュアクセス
を区別する方式ではありません。

また、MMU有効下では、CPU命令からのアクセスは常に仮想アドレスになる点にも注意が必要です。

バグの自動検出も可能な新開発環境 SOLID

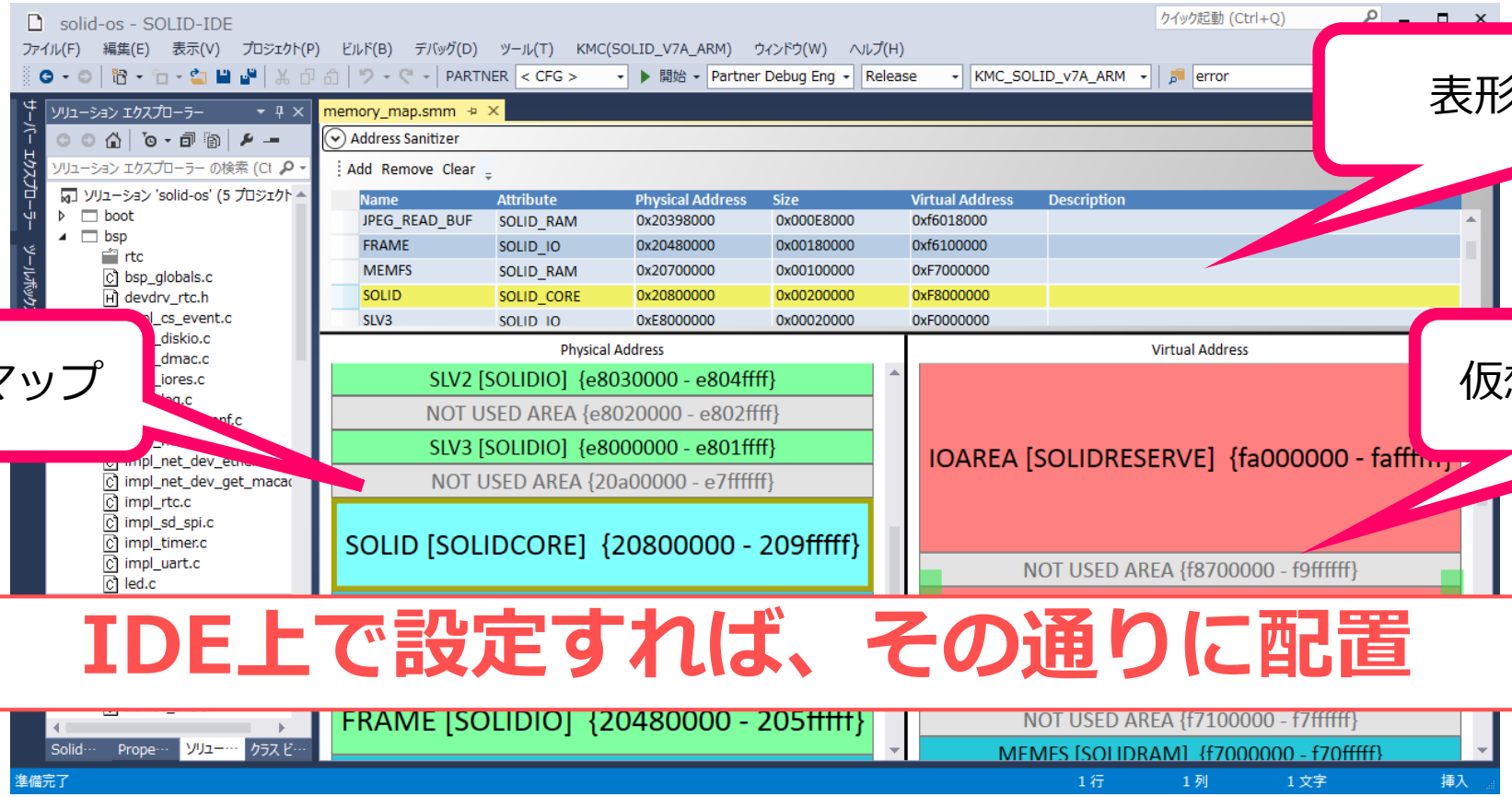
- OS側ランタイムとツールの専用設計で、MMUを簡単に活用
  - Cortex-AプロセッサのMMUを、設定だけで簡単に活用できます
    - CP15レジスタの設定を行うプログラムや、ページテーブルの作成は不要
  - 専用設計による次の仕組みで、MMUを簡単に活用できます
    1. IDE上のGUIで、物理アドレスと仮想アドレスのメモリマップを設定
       - アドレスやサイズ、また属性を設定
    2. ビルド時にIDEが、設定内容に基づくMMU設定テーブルを作成
    3. ARM側には上記MMU設定テーブルを元に、MMUを有効化するライブラリが動作
    4. 仮想アドレスやキャッシュ、プロテクションが動作

- MMUが簡単に使えるため、SOLIDはMMUを積極的に活用
  - SOLIDではリンカスクリプトと連動するプロテクション、やスタックフェンス、ELFローダーなど、各種機能がMMUを積極的に利用しています

enjoy Development

バグの自動検出も可能な新開発環境 SOLID

IDEから出来るMMUの設定



IDE上で設定すれば、その通りに配置

バグの自動検出も可能な新開発環境 SOLID

SOLIDを使うと。。。。。

- **立ち上げ簡単で、開発の開始コストを低減**
- **自動バグ検出で、デバッグ・テストコストを低減**
- **チーム開発の悩みを解決できる!!**

立ち上げ簡単で、
開発開始コストを低減!!

ツールとRTOSがセットで検証確認不要。

RTOSも数個のAPI実装で動作可能。

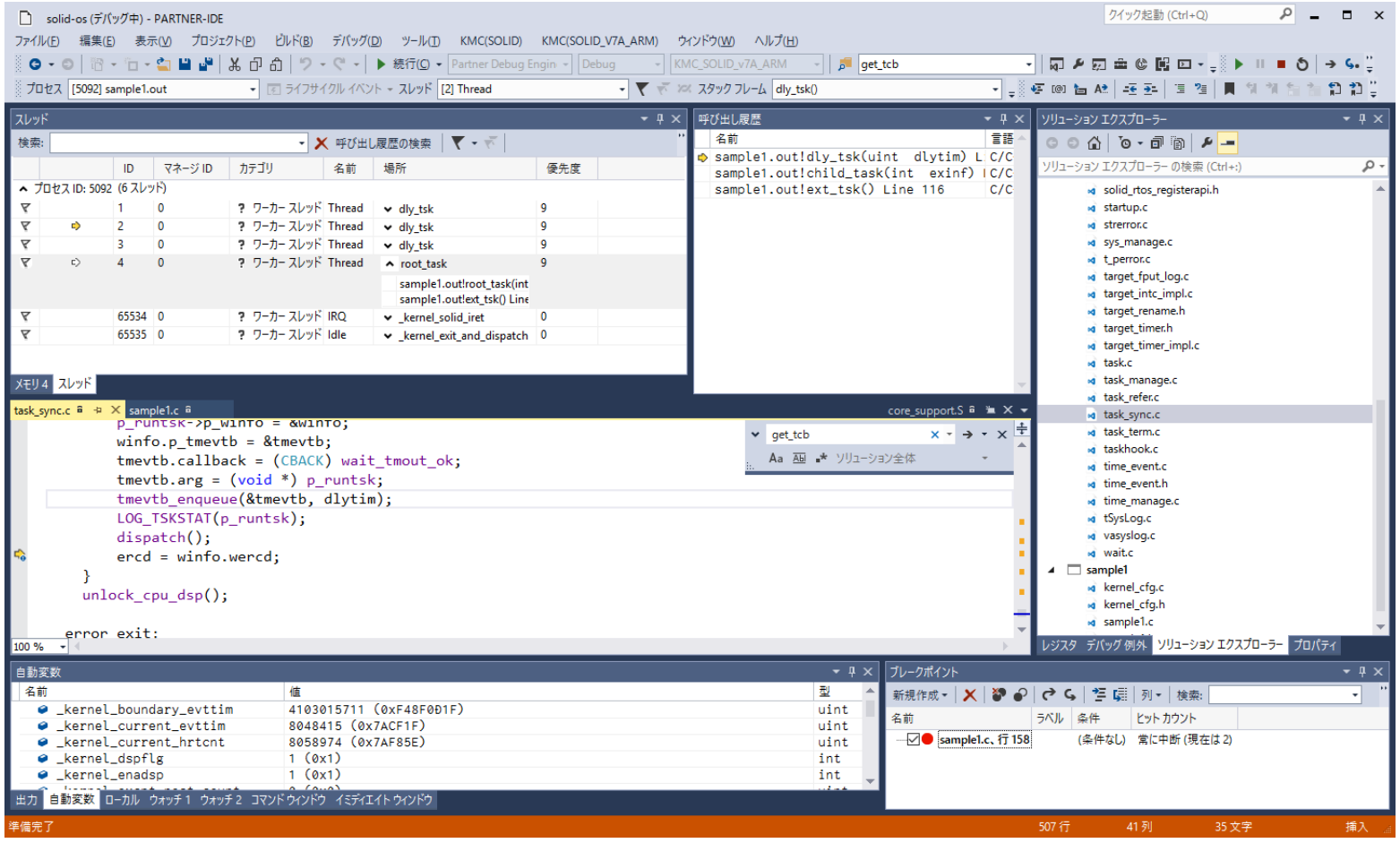
専用のARM Cortex用ライブラリと連携出来るGUI設定。

バグの自動検出も可能な新開発環境 SOLID

- **使いやすくて簡単**なVisualStudio
- RTOSの**立ち上げがきわめて簡単**
  - ITRONカーネル（TOPPERS/ASP3）を、デバイスやボード依存を排除した状態で供給
    数個（タイマと割り込み関係）の専用APIを実装するだけで、簡単にRTOSが動作

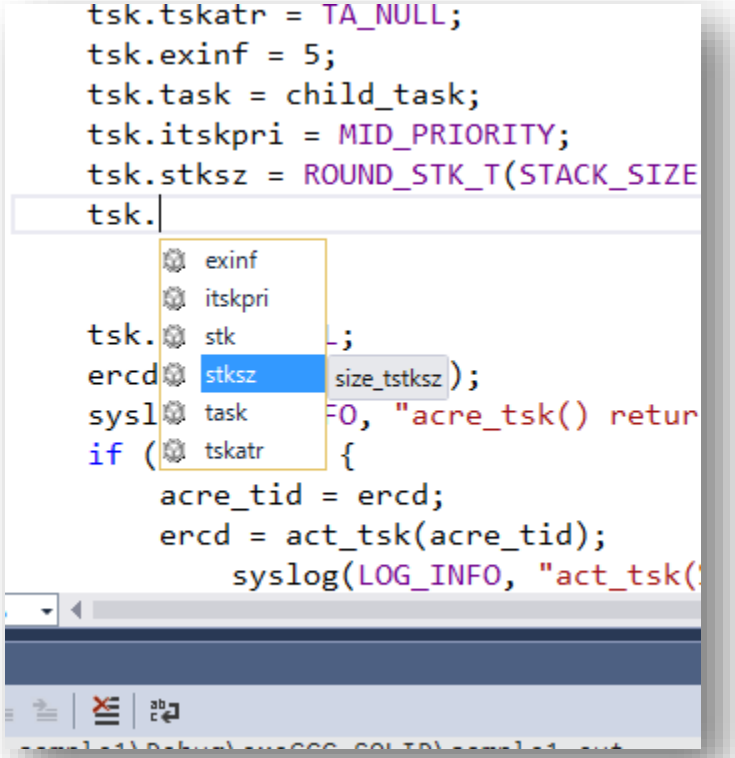
バグの自動検出も可能な新開発環境 SOLID

Visual Studio によるIDE

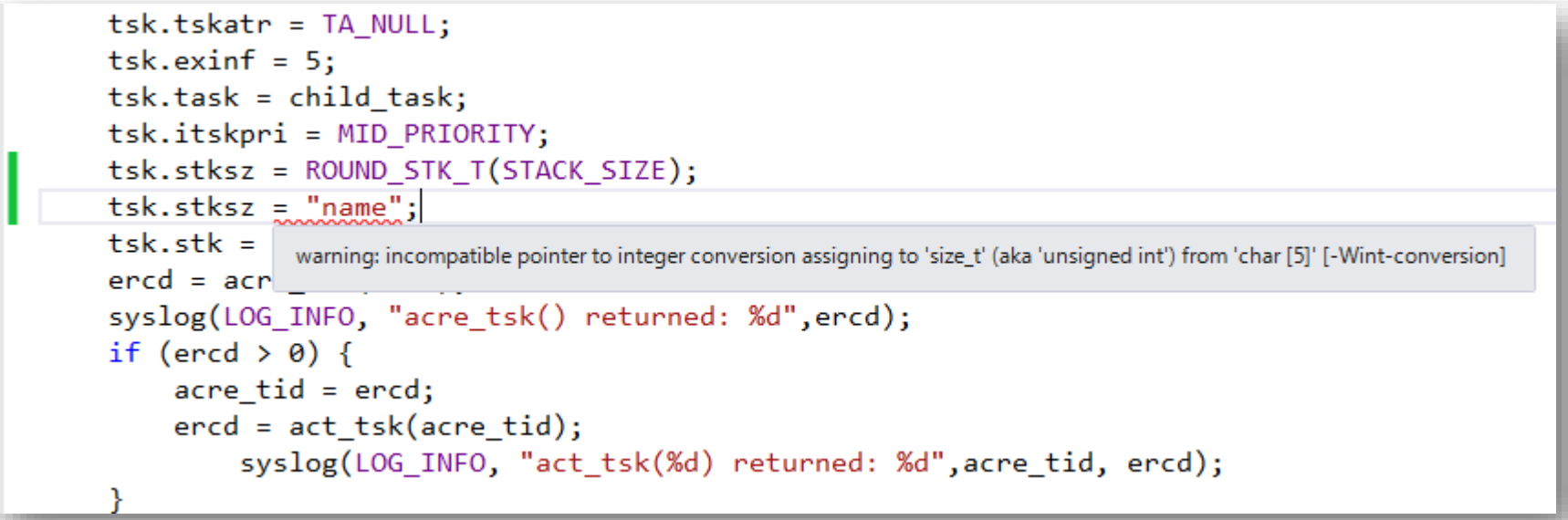


バグの自動検出も可能な新開発環境 SOLID

確実なコーディングで、build ←→editの手戻りを少なくします



インテリセンス機能



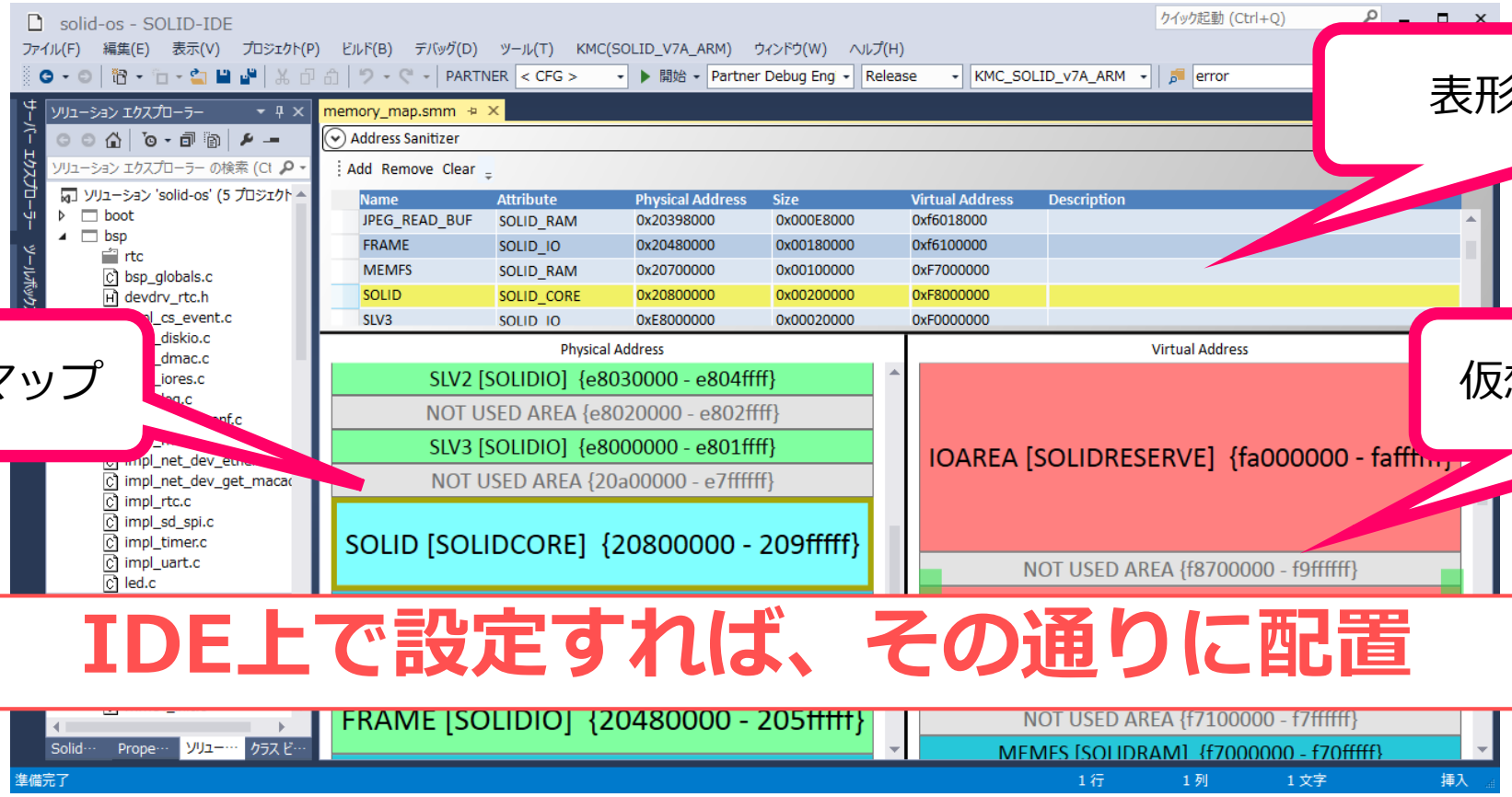
入力時にリアルタイムにコンパイルエラー・警告を指摘
（バックグラウンドでコンパイラが動作）

バグの自動検出も可能な新開発環境 SOLID

- ツールチェインを使いこなす設定や実装も
  あらかじめ**準備済み**
  - リンカスクリプトと実行ライブラリが連携するので、
    何もしなくても実行命令領域を”書き換え不可”などに設定
- Arm Cortex-AのMMUを、**GUIで設定するだけ**で高機能に利用
  - これもツールの設定処理と、実行ライブラリの連携

バグの自動検出も可能な新開発環境 SOLID

IDEから出来るMMUの設定



IDE上で設定すれば、その通りに配置

自動バグ検出で、デバッグ・テストコストを低減!!

ソースコード解析が標準付属。

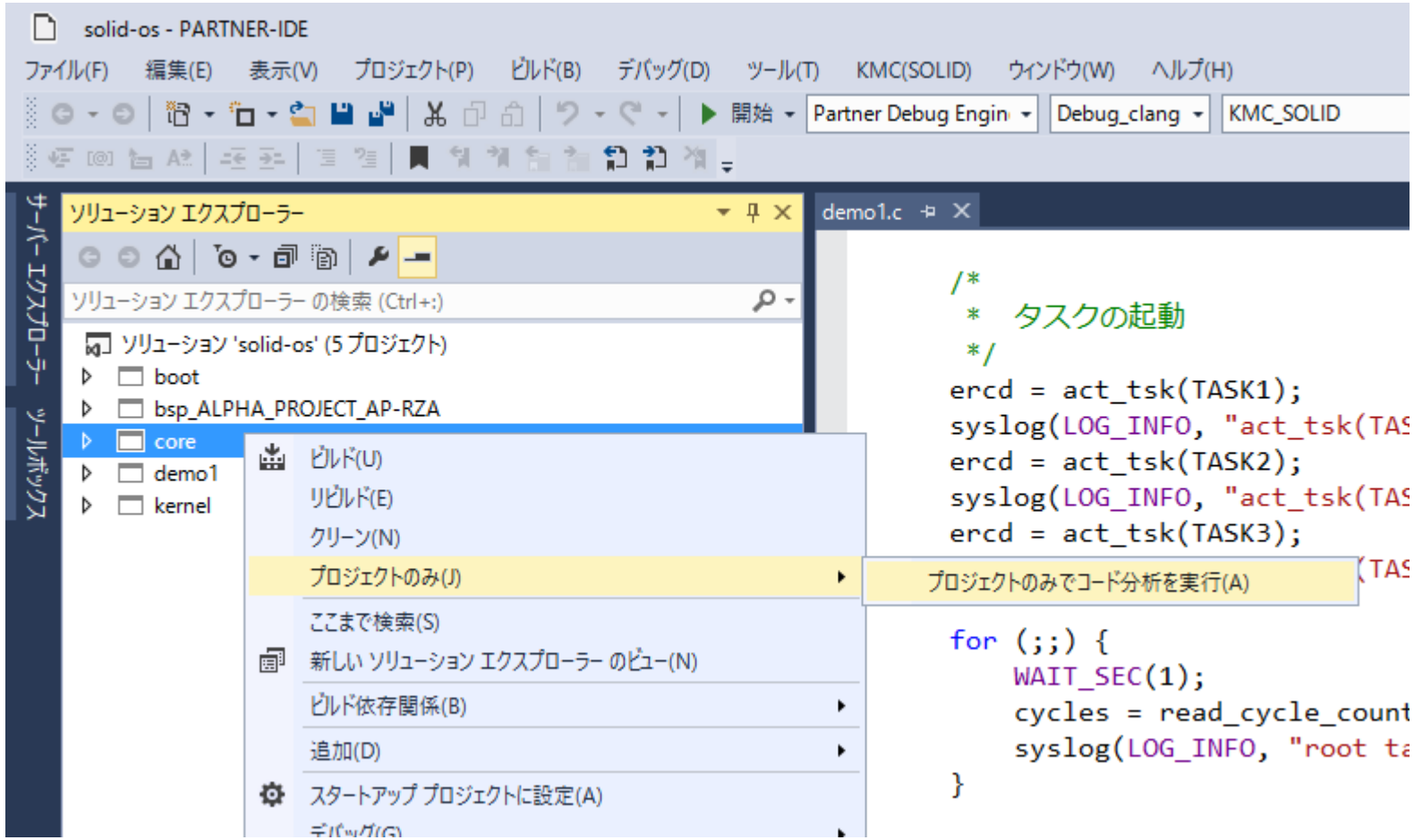
先進的な実行時メモリバグ検出機能や、メモリプロテクションで、隠れたバグを簡単にあぶり出し。

バグの自動検出も可能な新開発環境 SOLID

- ソースコード**静的解析機能**を標準搭載
  - 追加設定なく、ビルドの設定さえあれば、解析可能
  - 0除算や不正ポインタアクセスの可能性などを検出

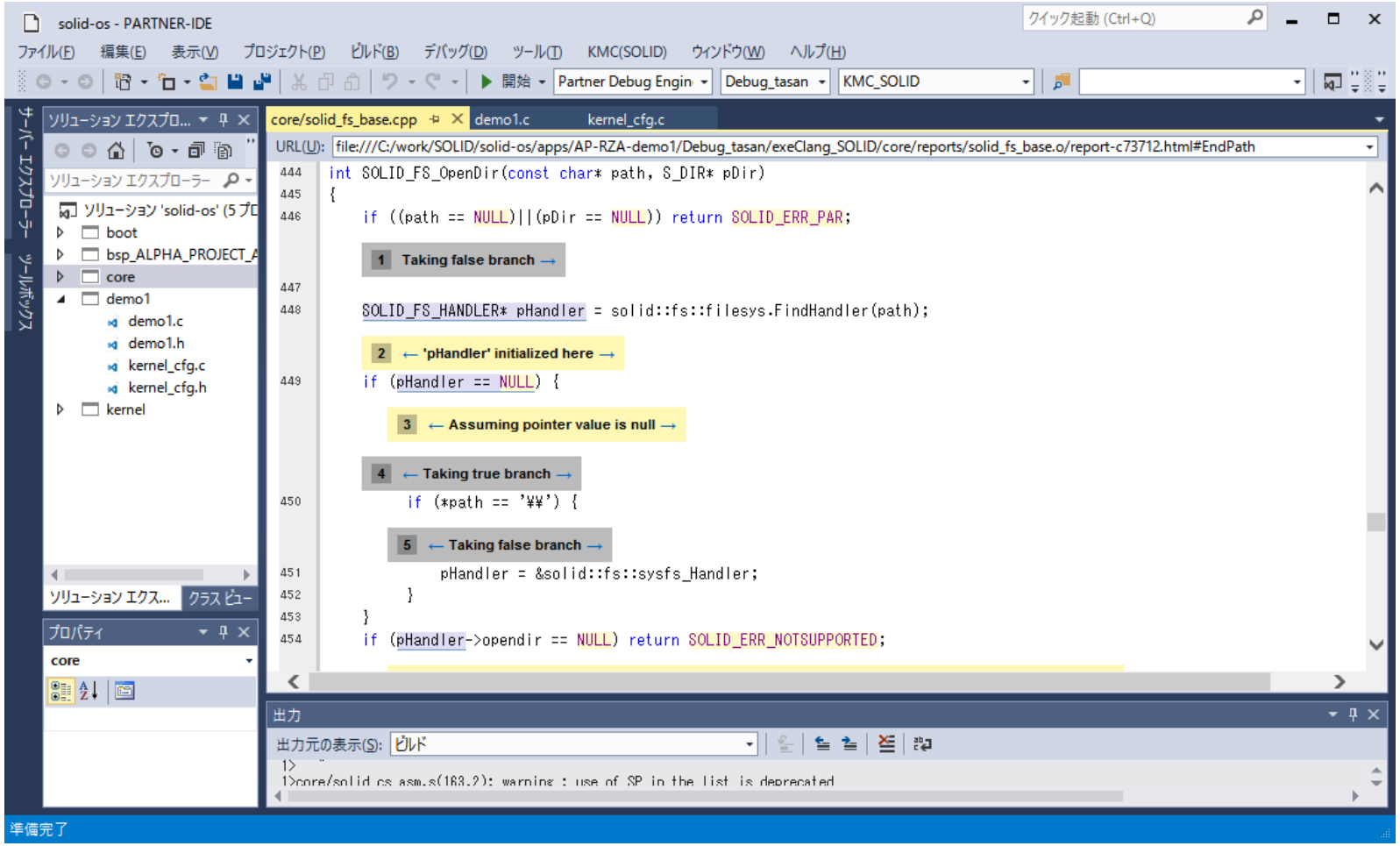
バグの自動検出も可能な新開発環境 SOLID

ソース静的解析も、メニューからワンクリックで実施



バグの自動検出も可能な新開発環境 SOLID

実行前のバグ検出が可能。デバッグを効率良くします



バグの自動検出も可能な新開発環境 SOLID

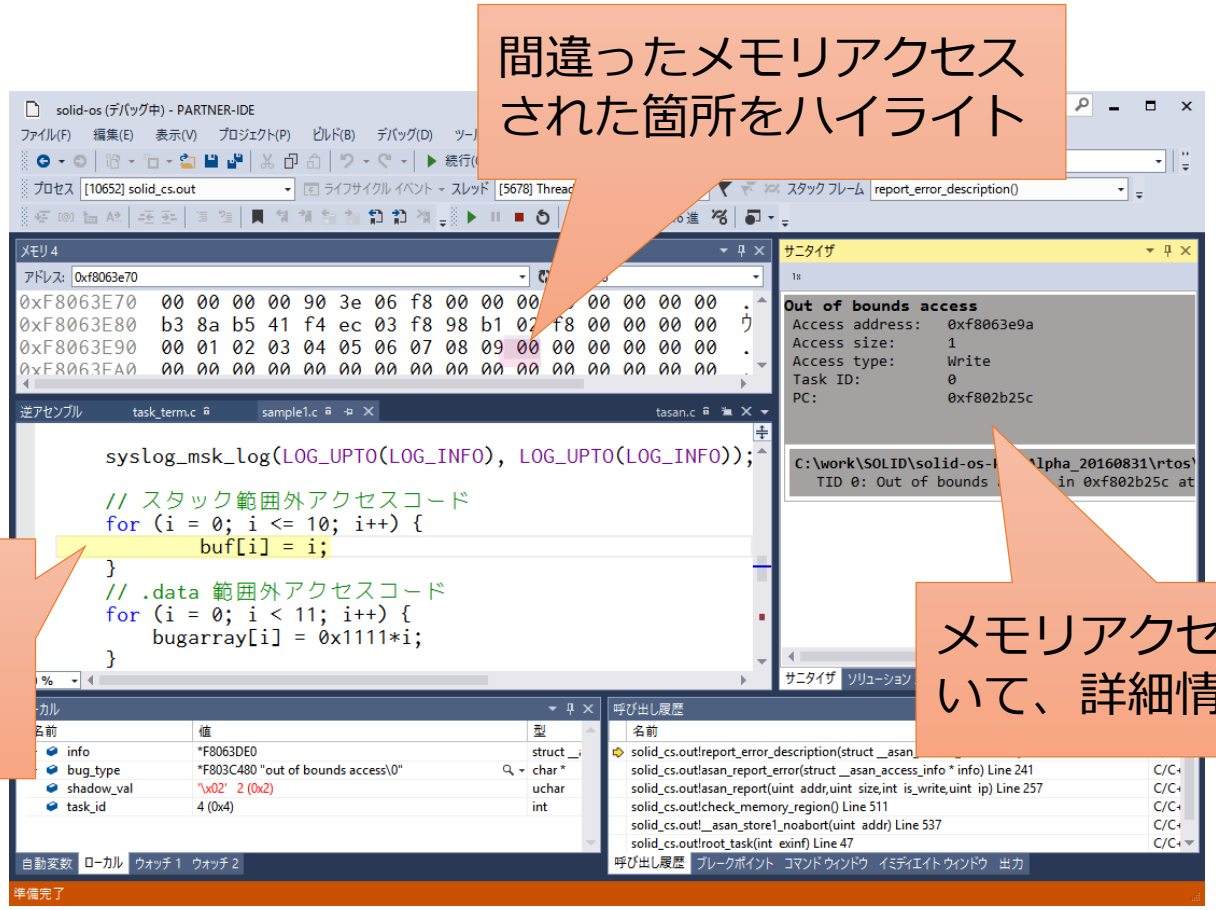
- MMUとリンカスクリプトが連携する**メモリプロテクション**
  - スタックの突き抜けもプロテクション
  - プログラム的に存在しない領域や、データ領域の実行など、不正アクセスを即時に検出
  - 単一アドレス空間によるプロテクション機構、既存のRTOS開発モデルと同じ
- **アドレスサニタイザ**による実行時メモリバグ検出機能
  - iOSアプリ開発環境などで定評の**アドレスサニタイザ**を、組み込みで初めて実現
  - バッファオーバーランや、解放後のメモリアクセスなど、**不正アクセスを自動検出**

バグの自動検出も可能な新開発環境 SOLID

メモリアクセスバグの自動検出！！

- バッファオーバーラン
- 不正ポインタ
- 二重解放など

手順はとてもシンプル
サニタイザモードでビルド＆実行するだけです



チーム開発の悩みを解決!!

分散開発に適した、アドレス解決可能なローダーを標準搭載。

バグの自動検出も可能な新開発環境 SOLID

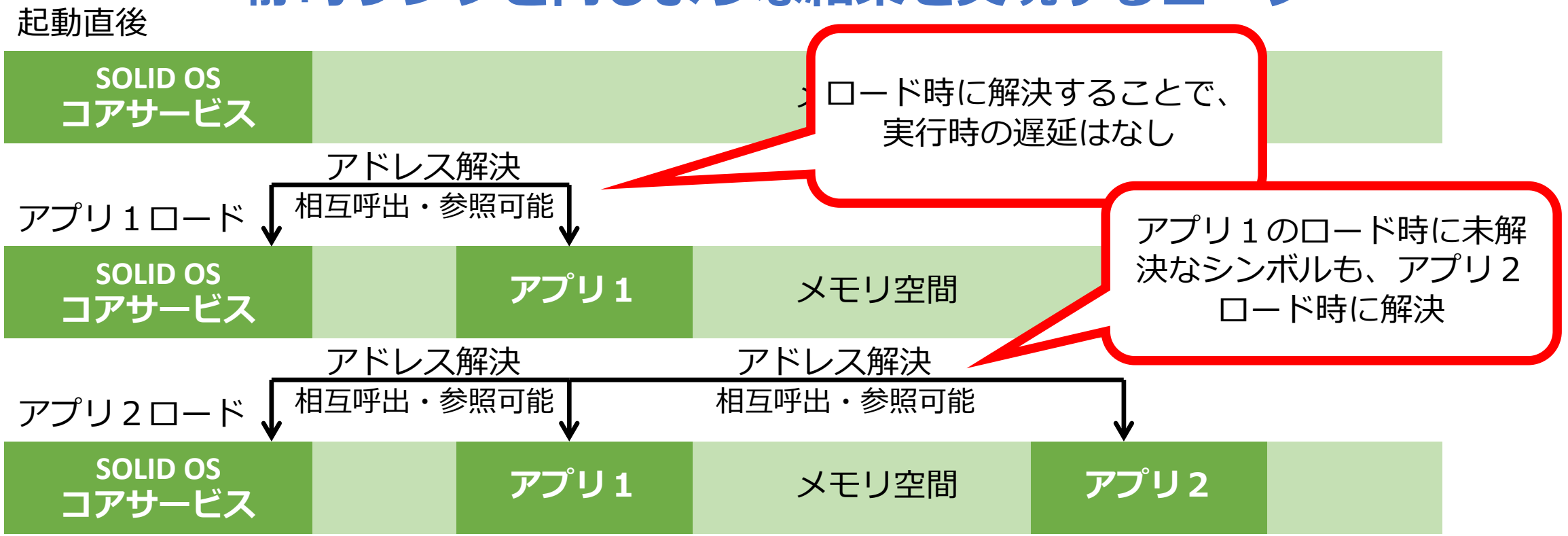
- **アドレス解決可能なELFローダー**を標準搭載
  - モジュール間で外部シンボルを実行時に解決
    - リンカに機能追加と、専用ローダーを用意
    - ロードした追加ELFモジュールをアンロードする事も可能
  - MMUと連携して、効率良くメモリを利用
- IDEとの連携
  - メモリマップエディタとの連携
  - デバッガとの連携
    - メインモジュールはROMのまま、**追加ELFのみを改変**してソースデバッグが可能

分散開発に便利

バグの自動検出も可能な新開発環境 SOLID

組込み向き!!

単一アドレス空間で静的リンクと同じような結果を実現するローダー



まとめると

enjoy Development

バグの自動検出も可能な新開発環境 SOLID

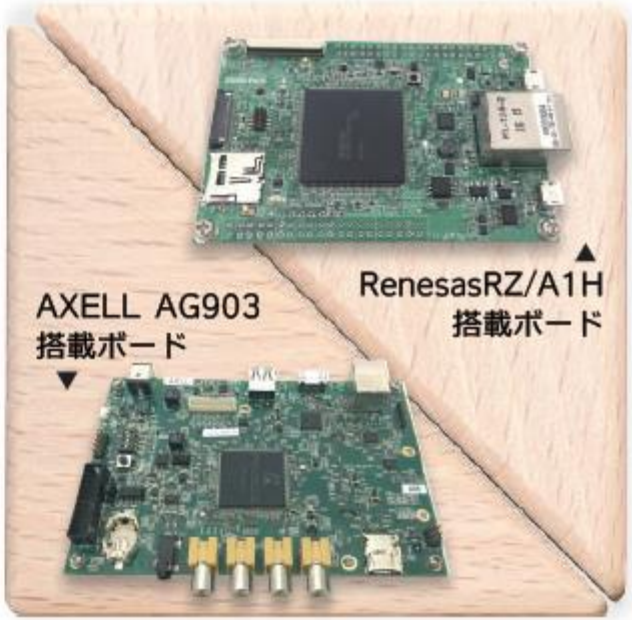
SOLIDは、

ITRON以上Linux未満を実現する開発環境

従来のプログラムモデルのまま、
Linuxライクな効率良い開発が可能

¥50,000-で、SOLIDを体験してみよう

SOLID スターターキット



IDE,コンパイラ,JTAG,ボードと、PC以外の必要な物全てが含まれます。
イーサネットドライバなどのBSPも付属して、購入してスグにSOLIDの開発環境を体験できます!!

スターターキットWEB →

SOLIDの進化

バグの自動検出も可能な新開発環境 SOLID

皆様のご要望を受けてSOLIDは、

TOPPERS/FMPベースのマルチコアシステム

に対応し、大規模なソフトウェア開発をさらに強力に支援

タスク優先度を最大256まで拡張
タスクなどOS資源の動的生成にも対応
そして特徴的な機能として・・・・

展示コーナーで動作デモ実施中

バグの自動検出も可能な新開発環境 SOLID

新機能： プロセッサが各々固有の値を保持する

Processor Local Storage

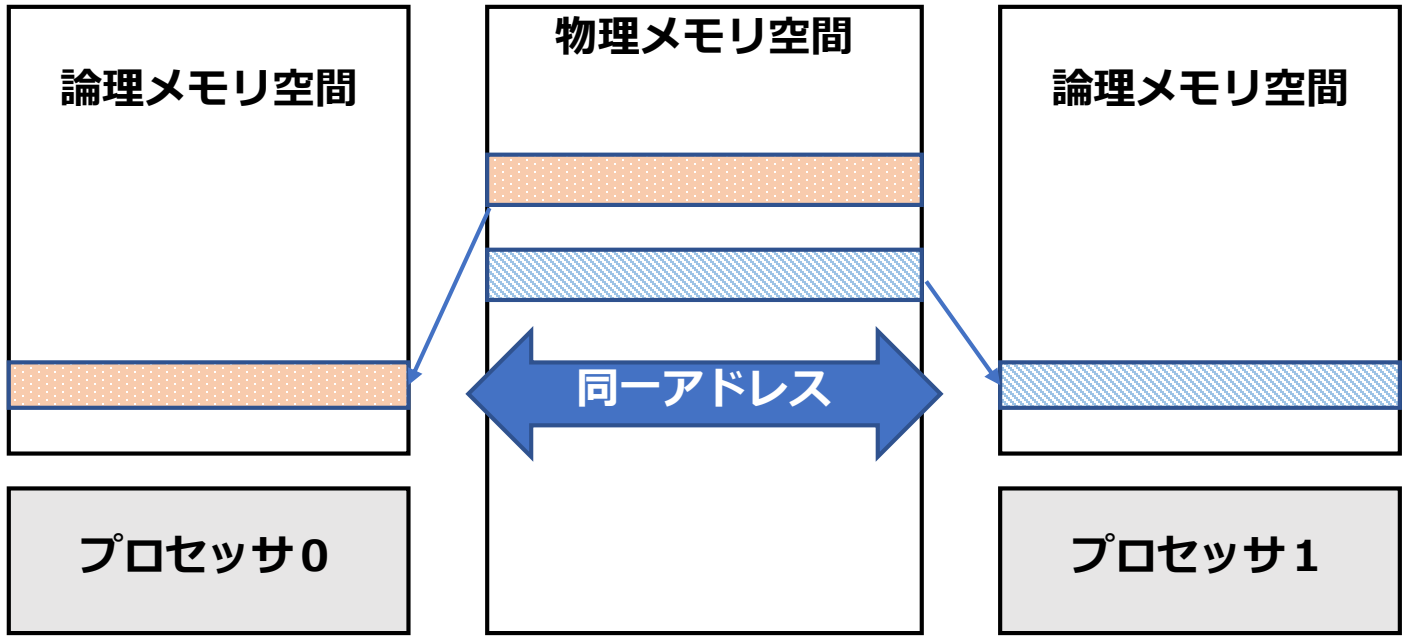
c 言語では同じ変数でありながら、

プロセッサごとに固有の値を持つことが可能になるため、

マルチコアによる並列動作を効率良く実行できる

バグの自動検出も可能な新開発環境 SOLID

SOLID がMMU の設定を行うランタイムを生成し自動的にメモリを割り付けます



Processor Local Storage 機構により、
同一の論理メモリアドレスに
固有の物理メモリデータを割り付ける

バグの自動検出も可能な新開発環境 SOLID

TOPPERS/FMPベースのマルチコアシステム向けのSOLID開発プラットフォームは、2018年度リリース予定

展示コーナーで
動作デモ実施中

2018年1月には Armv8-A(AArch32)用SOLIDのリリースも予定しています

enjoy Development

Linuxアプリ専用開発環境 LIQUID

- Linuxアプリケーション開発の専用環境
  - SOLIDのコンセンプトを引き継ぐ
  - VisualStudio IDEの採用
  - Clangコンパイラによる静的解析や、メモリバグ自動検出
  - IDEからアプリケーションだけを簡単に開発・デバッグできる
    - perfや straceなどとの統合
- 2018年度提供予定
  - デモンストレーションセットがあるので、ご興味のある方はご連絡ください

PARTNER-Jet2

ARM V8プロセッサ 64bit対応デバッグについて

PARTNER Jet2

PARTNER-Jet2　Model 10
USBバスパワー対応で、
低価格で導入しやすい普及モデル

4Gバイト/8Gバイトトレースメモリ
搭載の高機能モデル
PARTNER-Jet2　Model 20/30

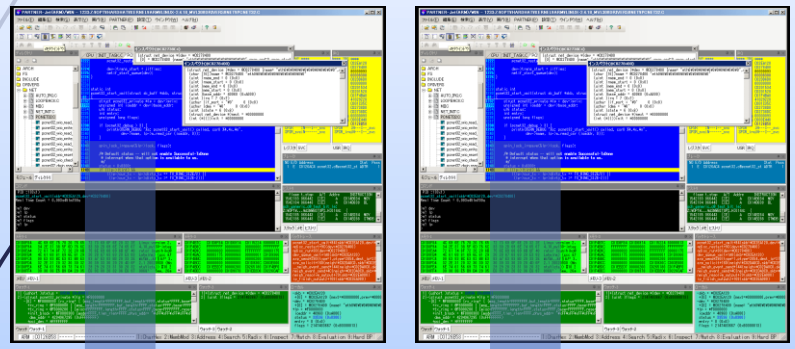
PARTNER-Jet2の利便性

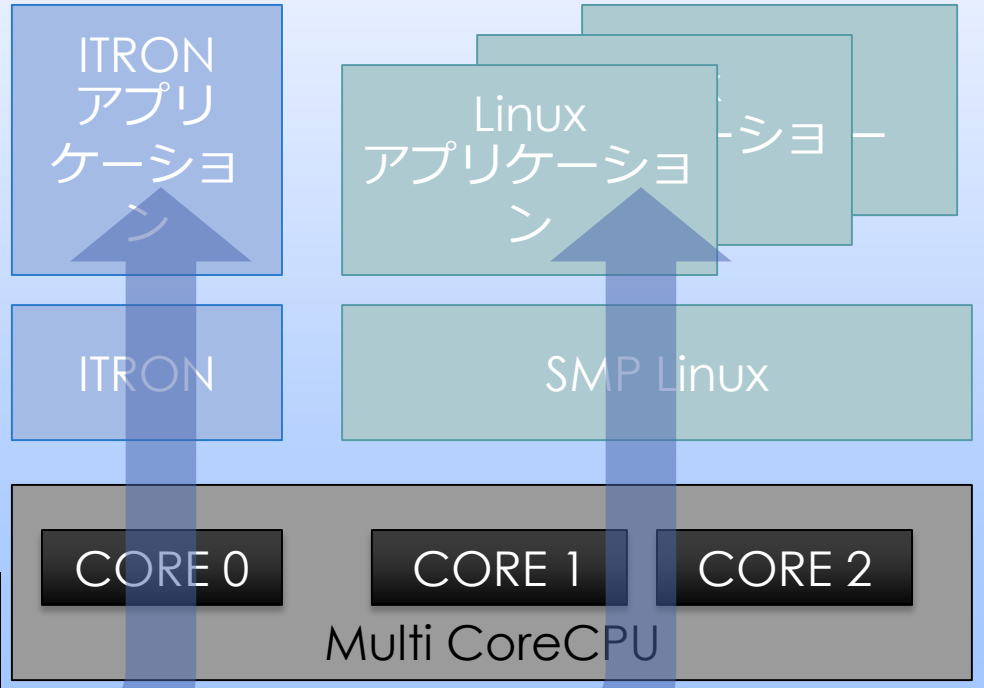
- USBバスパワー対応
  - PARTNER-Jet2 Model10は、USBバスパワーで動作
  - USB3.0のみならず、多くのPCではUSB2.0でも動作可能
    - 電圧低下時はLEDでそれを表示。その場合はACアダプタをご利用下さい

- プローブホットプラグ対応
  - ターゲット動作中に、後からプローブを接続してデバッグを開始したり、取り外したりする事が可能
  - 障害が発生してから、JTAGで状態確認が可能
  - ARMプロセッサーとIntelプロセッサーで対応

ITRONとSMP Linuxを同時にデバッグ

ITRONのデバッグ　Linuxのデバッグ





一台のPARTNER-Jetで二つのOSを同時にデバッグ

PARTNER-Jet2
ARMv8 64bit対応（デバッガ機能）

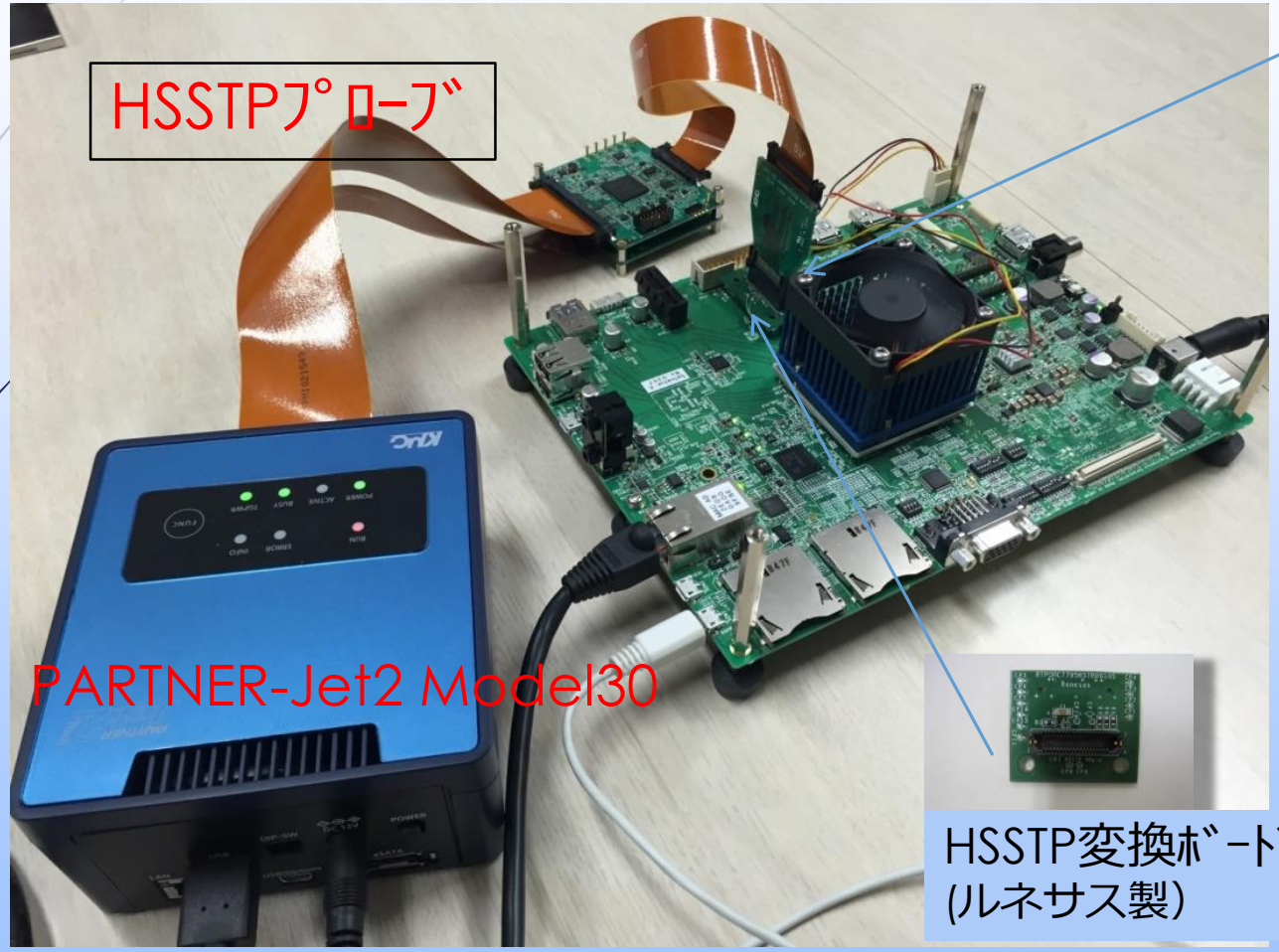
- 対応済み64bitプロセッサ
  - Cortex-A53, Cortex-A57, Cortex-A72, A73, A35
- 新しいETMトレース（ETM v4）対応
  - 従来のETMv3.5よりも、パケット効率がよくなった
  - トレースバッファ（ETB）では対応、動作確認済み
  - LVDS化されたETM HSSTPへの対応
    - R-Car H3（Cortex-A57/A53）で対応、動作確認済み
- PARTNER-Jet/Jet2で提供してきた各種機能への対応
  - スナップショットデバッグ
  - ホットプラグ接続
  - プロファイル機能
  - その他…

HSSTPプローブ

- ARM High Speed Serial Trace Port仕様対応のプローブ
- 4 Lane HSSTP　サポート
- 2.5Gbps/5.0Gbps対応
  - ARM ETMv4.0対応
  - ARM STM対応　　400MByte/sec

  PARTNER-Jet2/Model30との組み合わせで使用

R-Car H3 Salvator HSSTP接続



HSSTP
(5Giga bps Trace)

STM 転送実績
H3 → Jet2
440Mbyte/s

H3 → Jet2 → PC
150Mbyte/s

HSSTPで何を出すか？

- HSSTPは、トレースデータ出力の転送フォーマット
  - ETMやSTMのトレースを出力するのが主か？
  - ETM/PTM
    - CPUの実行状況のトレース（分岐トレース）
    - 命令実行の履歴などを再現できる
    - トレースデータの中身は決まっているので、仕様に従い解析・表示機能を作成できる
  - STM
    - 色々あるが、簡単なのは任意のメモリデータを出力することができる
    - どういうデータを出力するのかにより、解析・表示は異なる

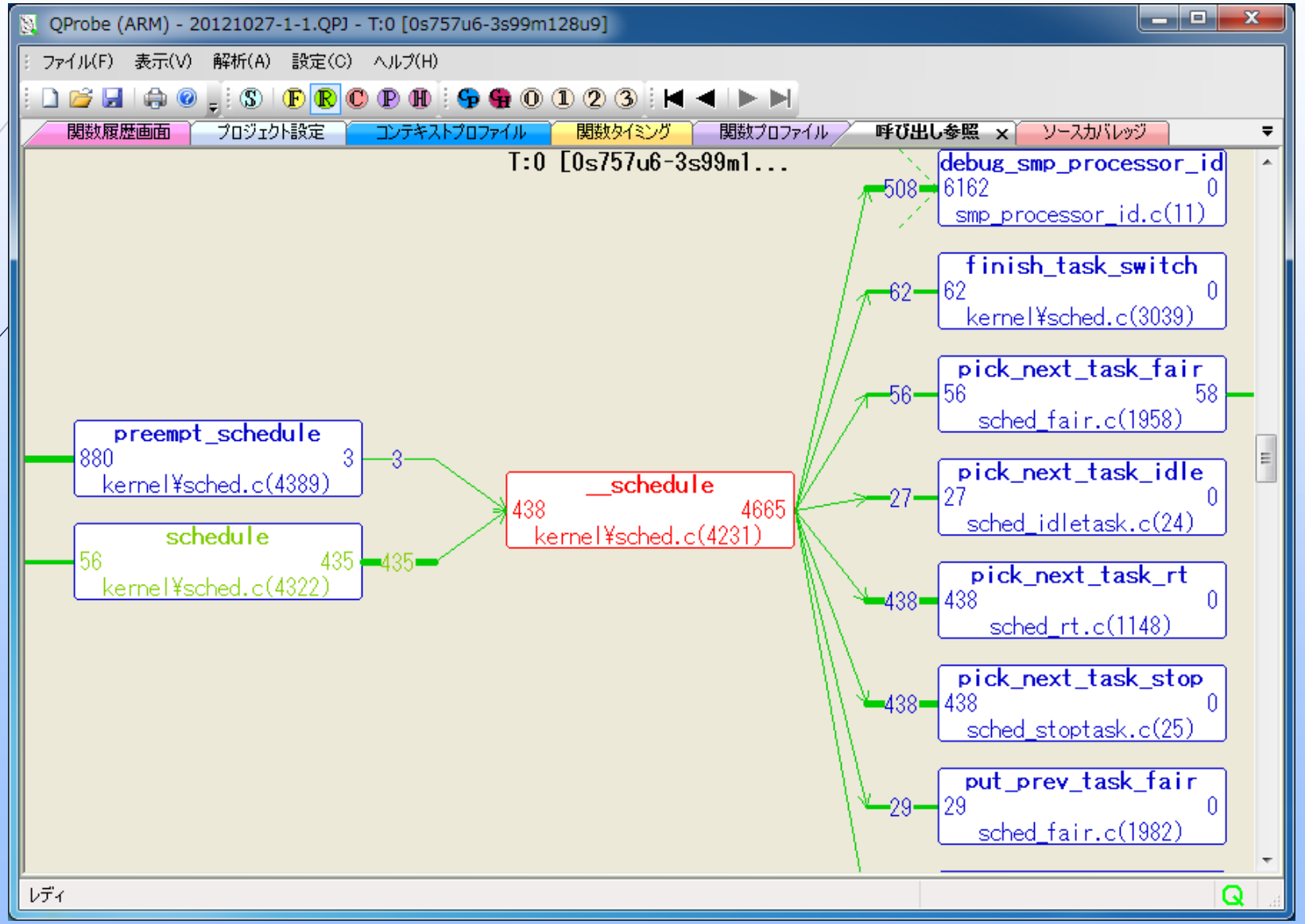
QProbe

動的性能解析ソフトウェア

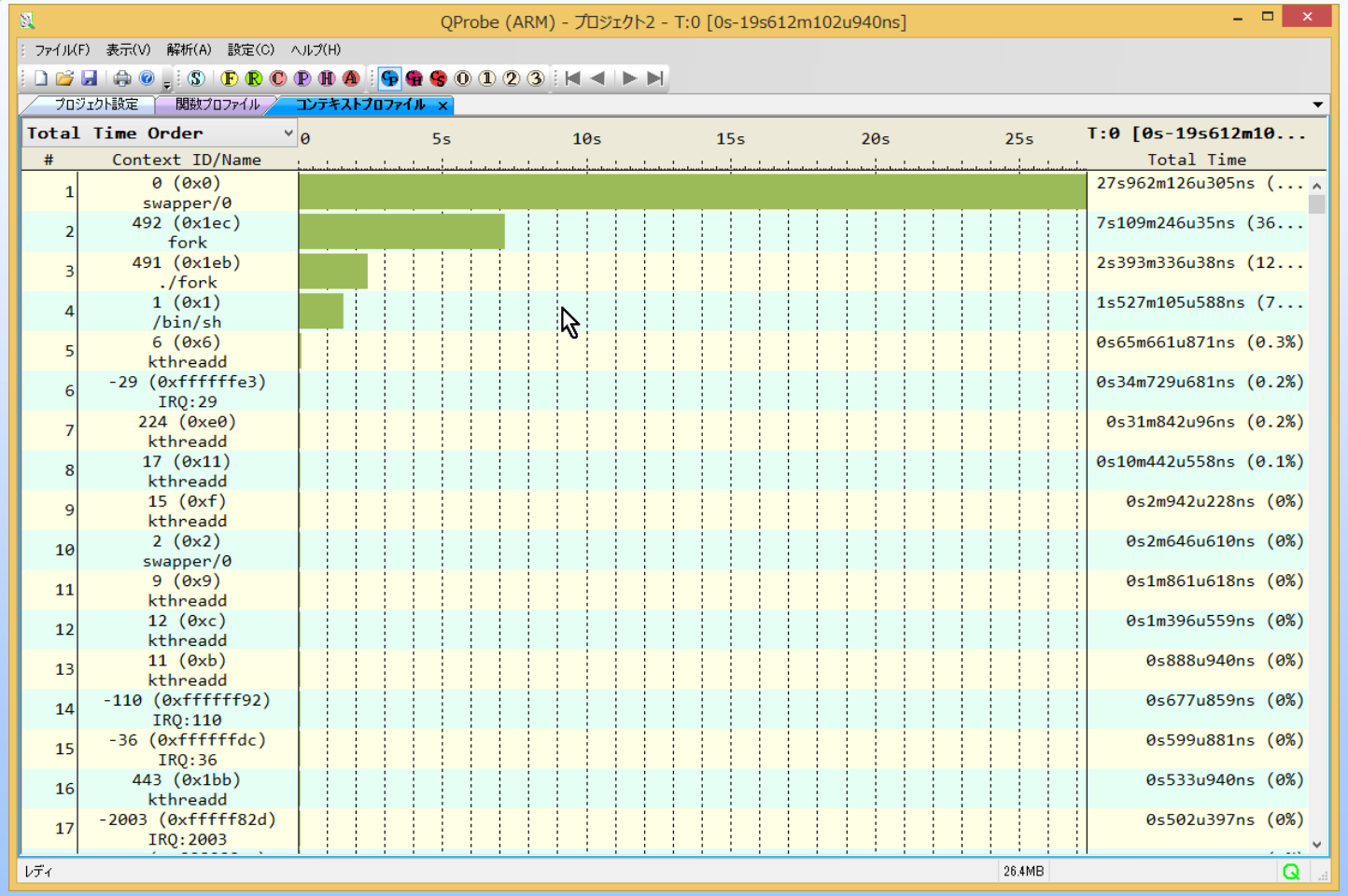
解析ソフトウェア QProbe

- プロファイルや実行履歴、実行偏差などはQProbeを利用して解析
  - トレースログのタイミング表示
  - 関数プロファイル
  - スレッド、プロセスプロファイル
  - 関数実行順序、プロセス実行順序
  - 関数実行偏差、統計解析
  - コンテクストのプロファイル

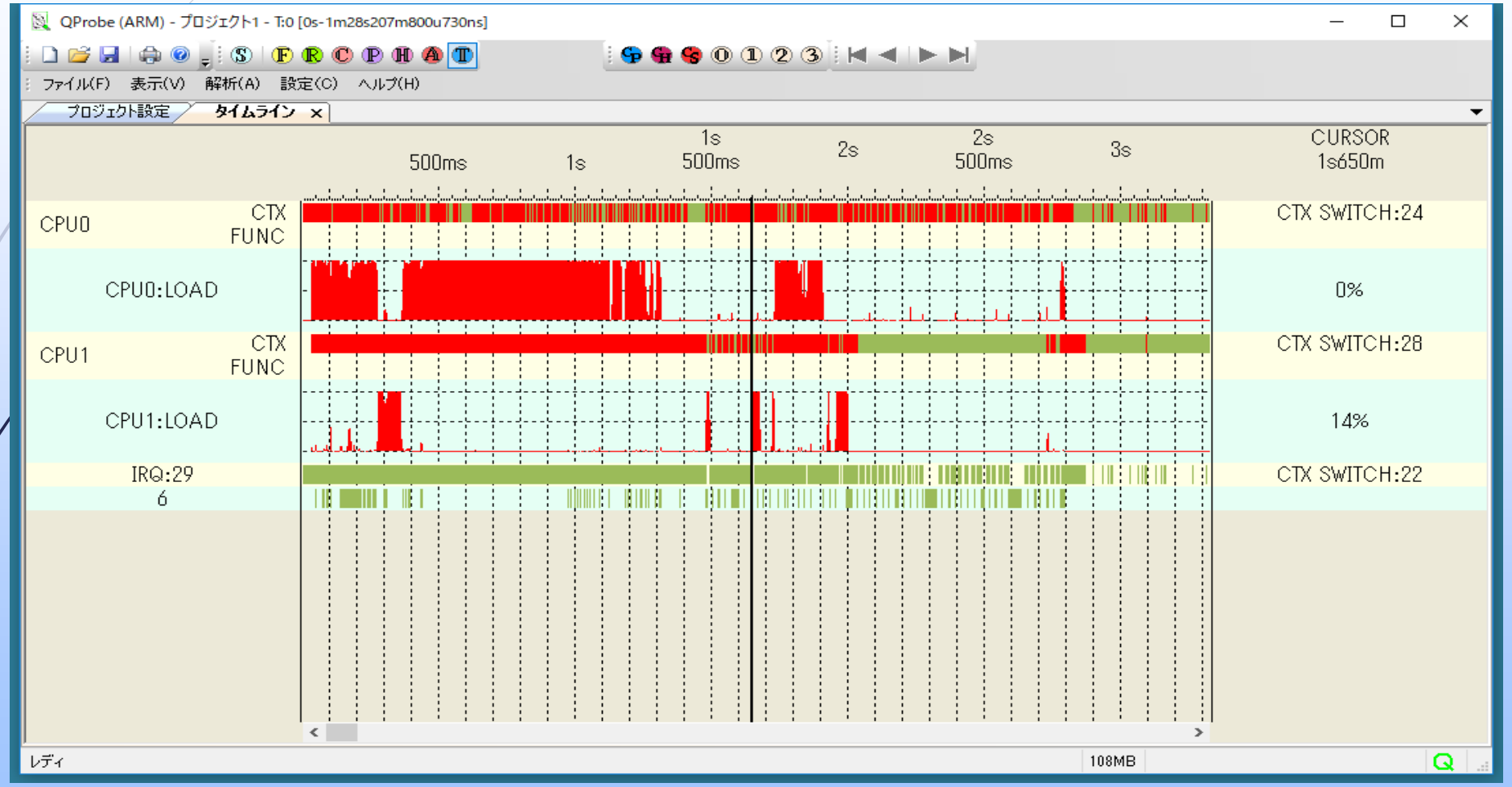
QProbe 呼び出し参照表示



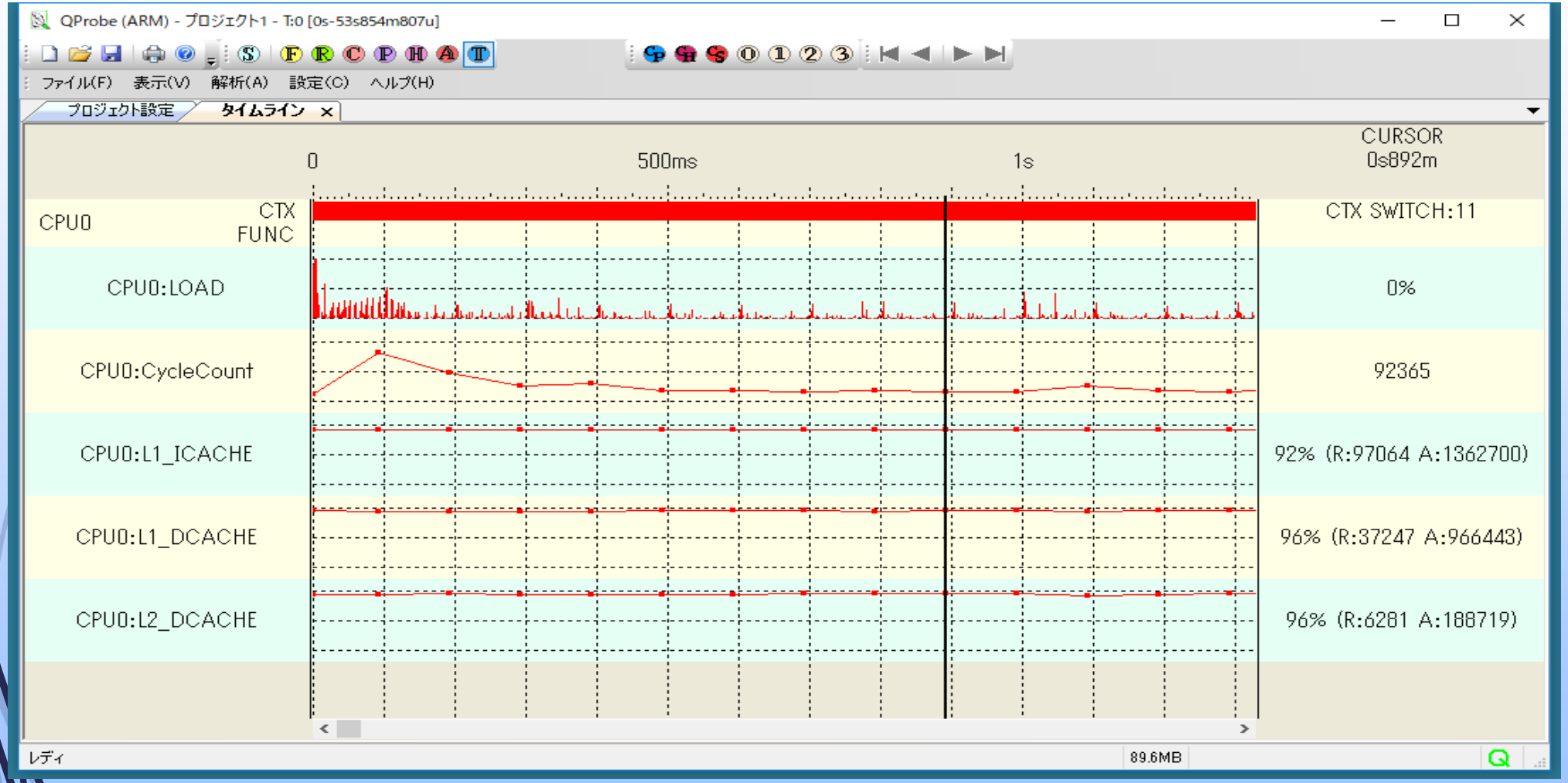
QProbe コンテキストプロファイル



QProbe CPUコア毎の負荷



QProbe CPU負荷とキャシュヒッ



SoC内部バストラフィック負荷計測ツール

概要

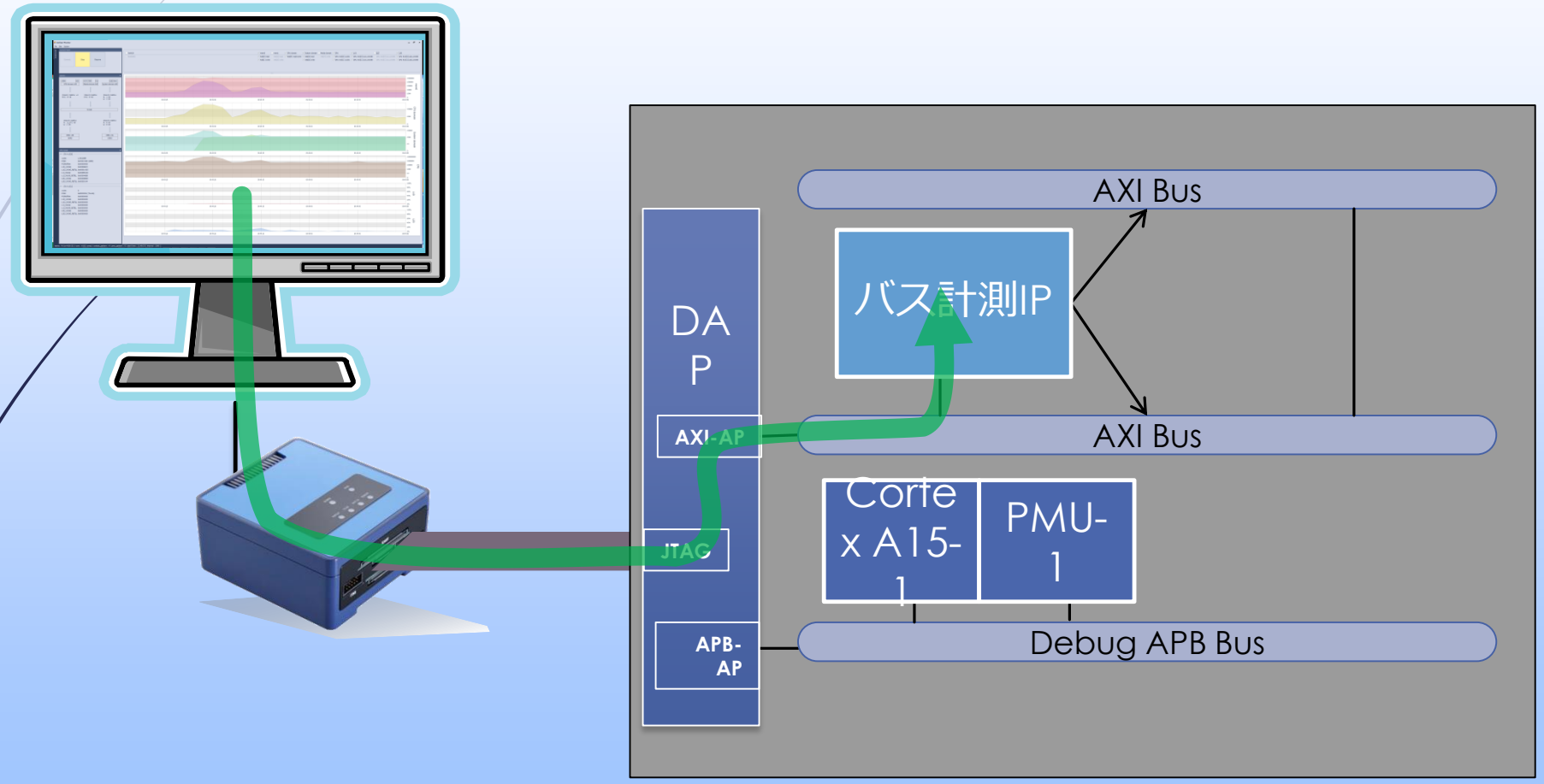
- ARM Cortex世代を搭載した、JTAGによるSoC内部情報取得ツール
- 測定について、CoreSightデバッグ専用ポートからバス負荷測定レジスタを直接に制御
  - デバッグ専用ポートを利用しますが、測定時にCPUをブレークさせません（CPUは止まらない）
  - CPUを介さないAXI-APという仕組みを使って、測定レジスタをJTAGから直接に操作
- 専用ハードウェアが周期的にSoC内のレジスタなどの情報を収集し記録
  - ミリ秒オーダーの周期（周期設定可変可能）でJTAGからAXI-AP介して読みとった測定結果を記録
  - PCにUSB介してリアルタイム表示
    - 測定結果をGUI表示
    - 設定した閾値を超えた時に判別しやすい表示
  - 記録したデータをファイル保存（CSV形式）
  - 保存したデータを再生表示

二つの実装方法

- JTAGから制御レジスタを直接読み出し
  - JTAGから測定IPの制御レジスタを直接にリードライト
  - 測定対象のSoCで動作するファームウェアの改造不要
  - ツールの実装コストが大きい
  - 測定IPの仕様によっては、データ読み出しのロストの可能性がある

- ターゲット内に読み出しモニタを実装
  - 計測対象SoCのファームウェアに、測定IPを設定・読みとりする簡単なモニタを実装
  - ツールはJTAGとAXI-APを介して、モニタとの情報共有領域を読み出し
  - 計測タイミングと、ファームウェアのソフトの同期が可能（V-Sync単位など）
  - ツールの実装コストが小さい

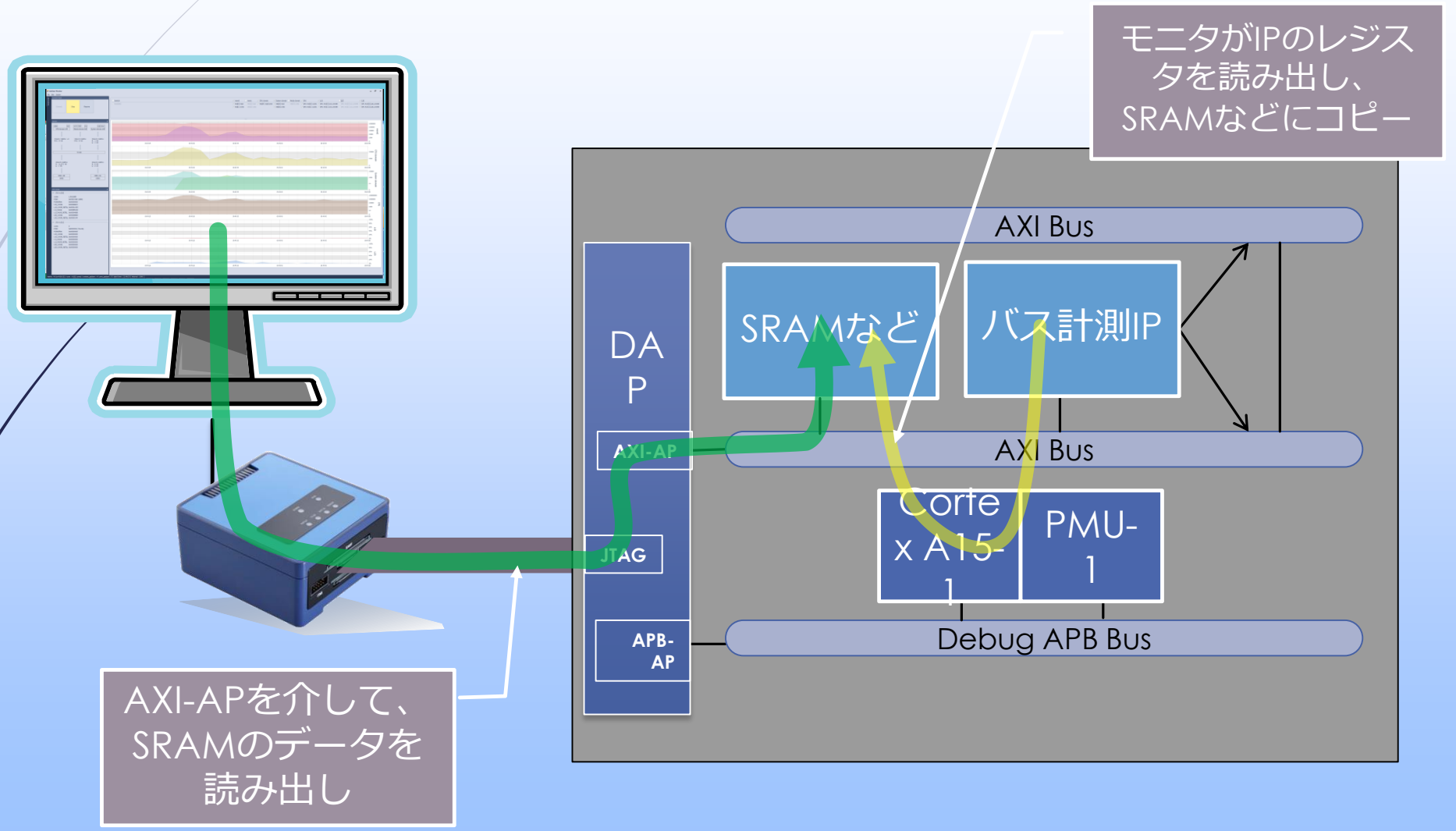
JTAGから制御レジスタを直接読み出し



JTAGから制御レジスタを直接読み出し

- バス計測IPのカウンタレジスタの要求仕様
  - 計測周期を設定出来る
  - 計測した周期単位で、カウンタレジスタのレジスタバンクが切り替わる
- JTAGからの読み取りは、ポーリングになるので、短い時間で正確に計測しようと思うと、上記が必要

ターゲット内に読み出しモニタを実装



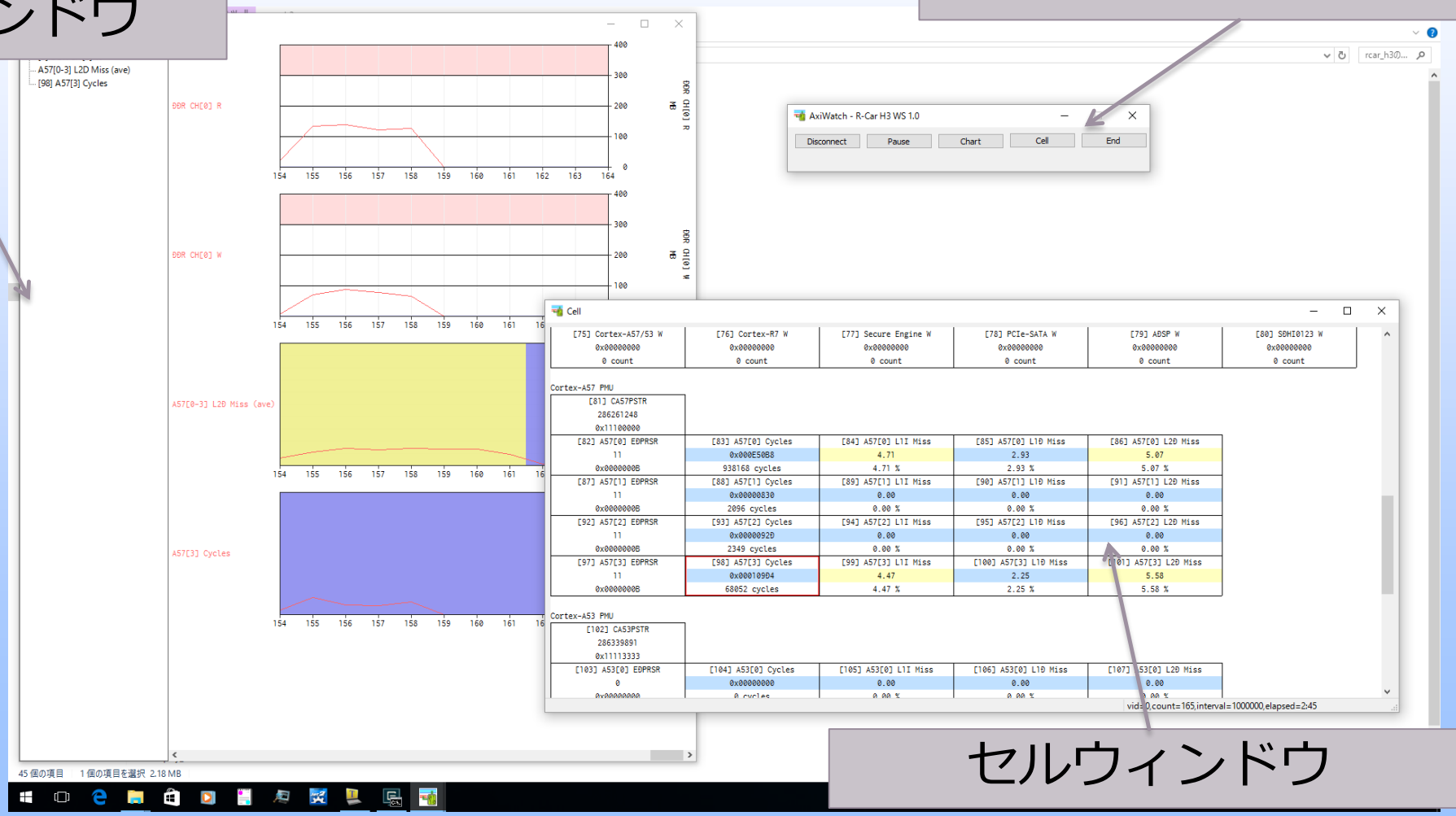
ターゲット内に読み出しモニタを実装

- 読み出しモニタと、ツールが使う共有メモリをSRAMなどに配置
- 共有メモリは、2面構成（ダブルバッファ）
- モニタは、計測毎に共有メモリにコピー
- コピー時に計測毎にインクリメントされるシリアル番号を付与する
  - ツールはこの番号を見て、更新されたか、またロストしたかを判断
- ツールはモニタの計測周期より短い周期で共有メモリをチェックする

バス負荷計測アプリ 画面全体図

チャートウィンドウ

コントロールウィンドウ



セルウィンドウ

ありがとうございました

Kyoto Microcomputer Co.,Ltd.

http://www.kmckk.co.jp