SOLID IDEのイベントトラッカ機能を使ってみよう！

2018.06.08

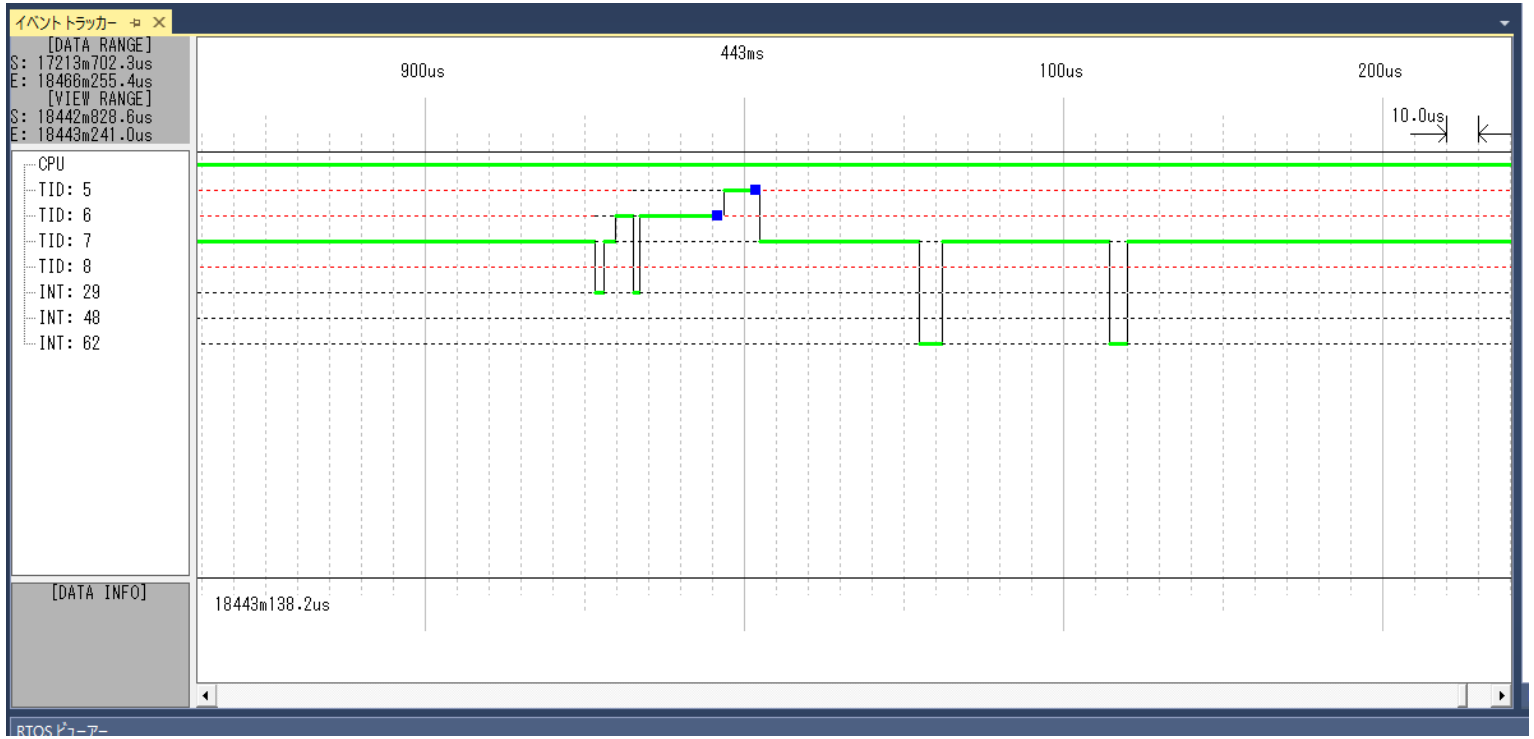
京都マイクロコンピュータ

SOLIDイベントトラッカ機能

何ができるか

イベントトラッカの概要

- タスク、割り込み、タイマハンドラの実行時間や遷移を、この画面のように時間軸に沿って記録・表示する機能です。



イベントトラッカの概要

- タスクや割り込み間の遷移が設計通りになっているか？
- タスクや割り込みの処理時間が想定通りになっているか？
- タスクや割り込みの処理間隔が想定通りになっているか？

等の確認や、

- 実際の処理時間のざっくりした評価

等に使える機能です。

SOLIDイベントトラッカを使うための準備

使うために前もって設定すべき内容

イベントトラッカを使うための準備

- 既にイベントトラッカが使用するタイマーの実装が済んでいることを前提として説明します。
- タイマー以外の下準備は必要ありません。

※新しい環境でタイマーを実装する必要が有る場合は、本資料末尾の付録を参照して実装してください。

SOLIDイベントトラッカの使い方

実行する際の手順

イベントトラッカの使い方

- ソリューションのプロパティファイル(.props) の <LibEventTracker> というプロパティを “true” に設定。無効にする場合には “false”に変更します
- プロパティ<LibEventTracker>が無い場合には巻末の付録の説明に従い追加して下さい

※変更後は、一旦 SOLID-IDE を終了し、ソリューションを読み直す必要があります。

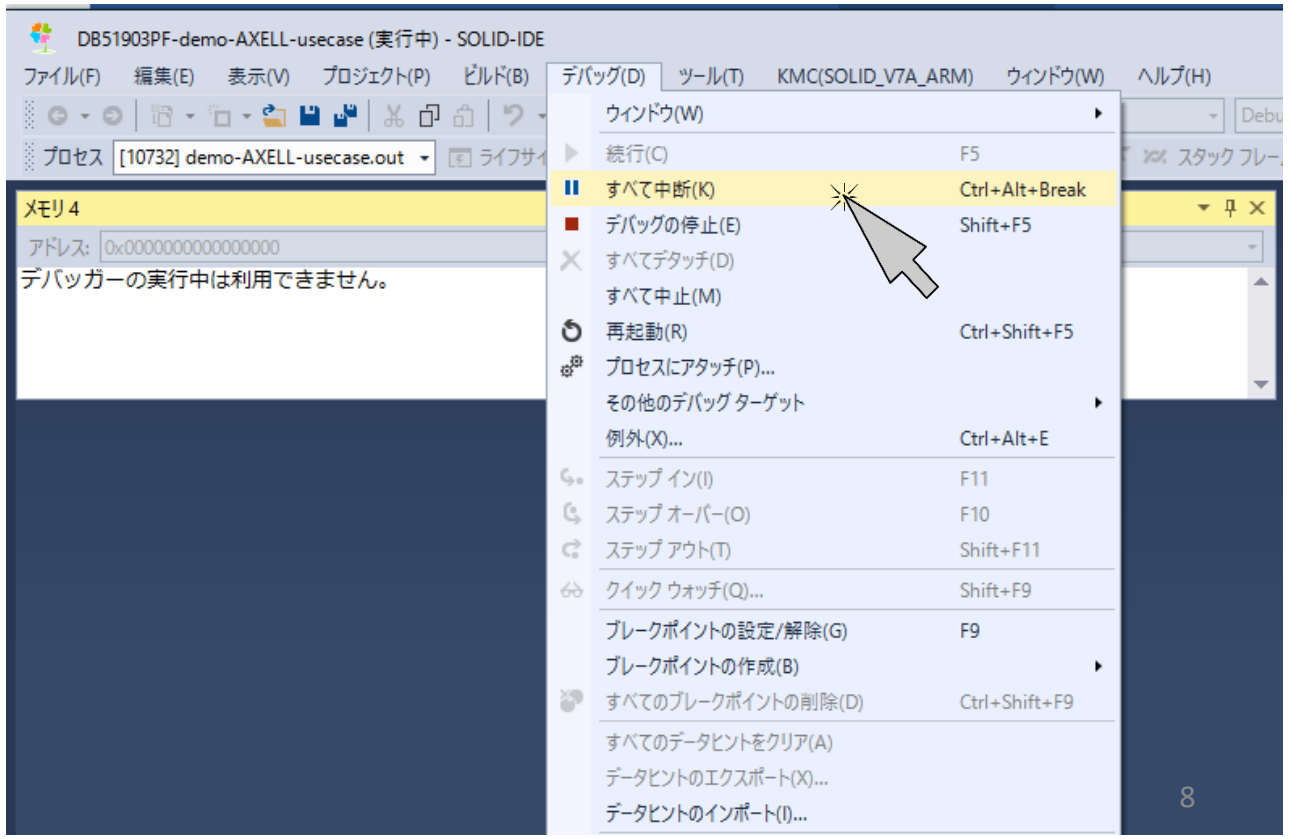
[～～.propsの変更内容]

```
. . .
<PropertyGroup Label=”UserMacros”>
. . .
<LibEventTracker>true</LibEventTracker>
. . .
</PropertyGroup>
. . .
```

※※スターターキット(BSP 2.0.1 以降必須)では常時有効なので、この設定は不要です。

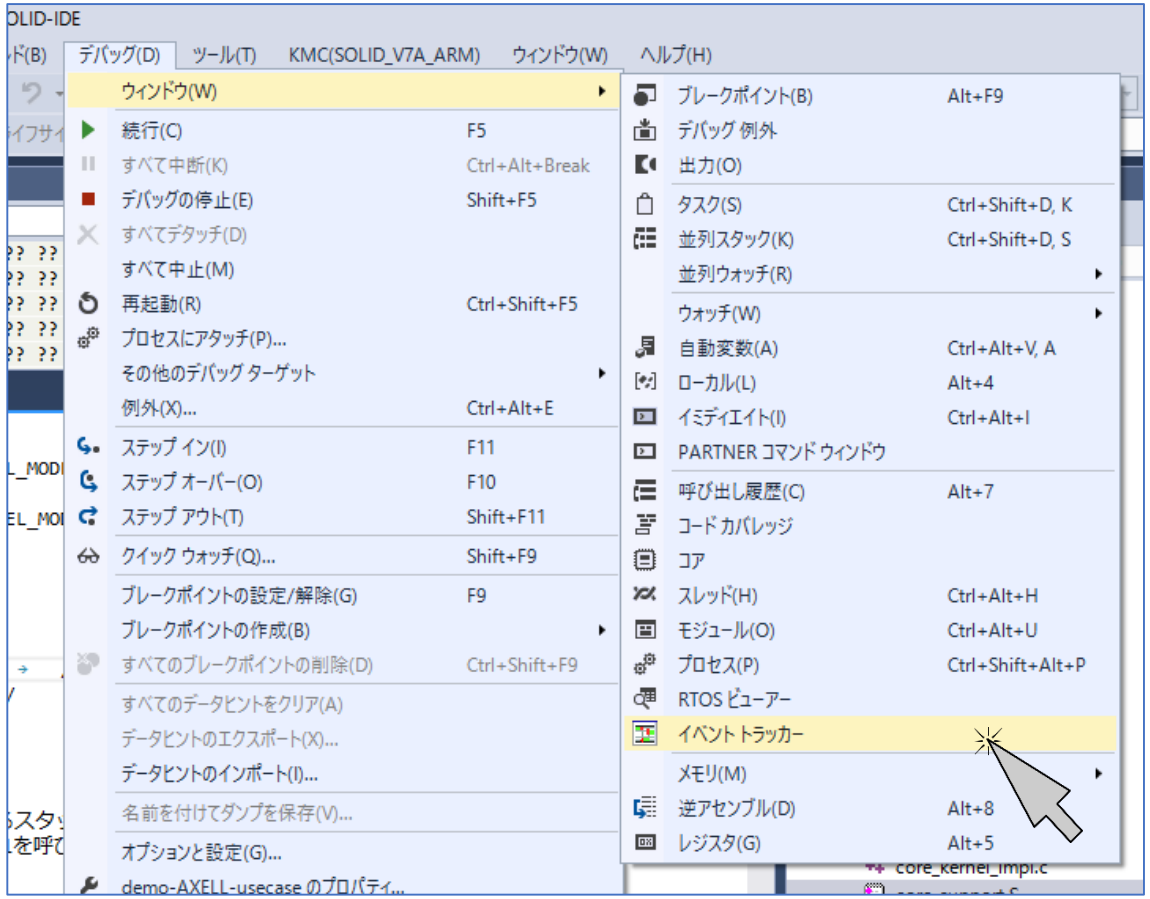
イベントトラッカの使い方

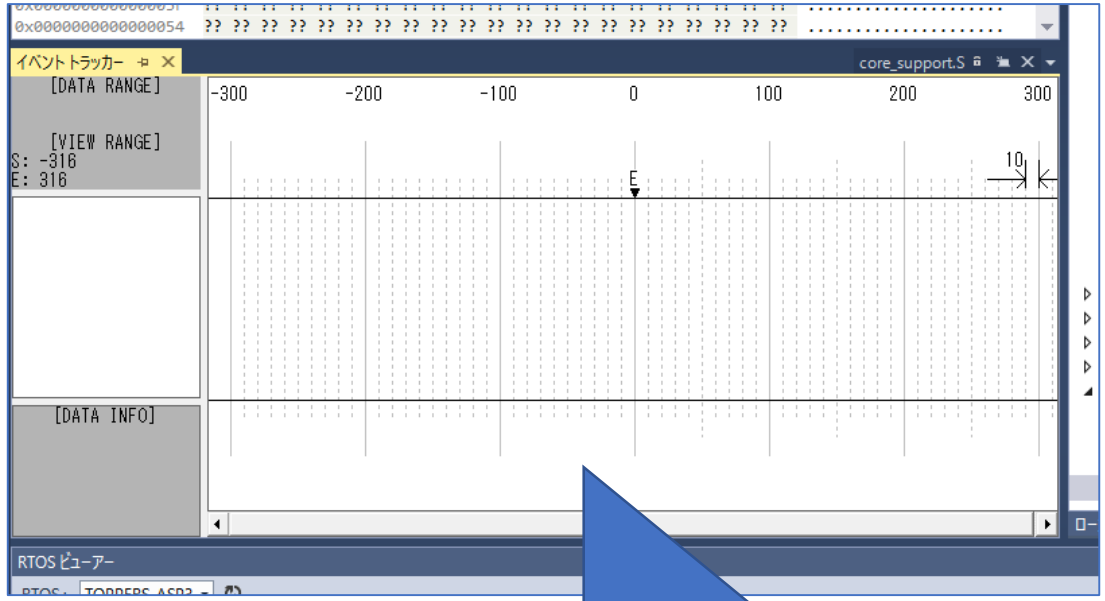
1. 全ページの設定をしてビルド＆実行
2. 適当なタイミングで、[デバッグ]-[すべて中断]



イベントトラッカの使い方

3. メニューの[デバッグ]-[ウインドウ]-[イベントトラッカ]を選択

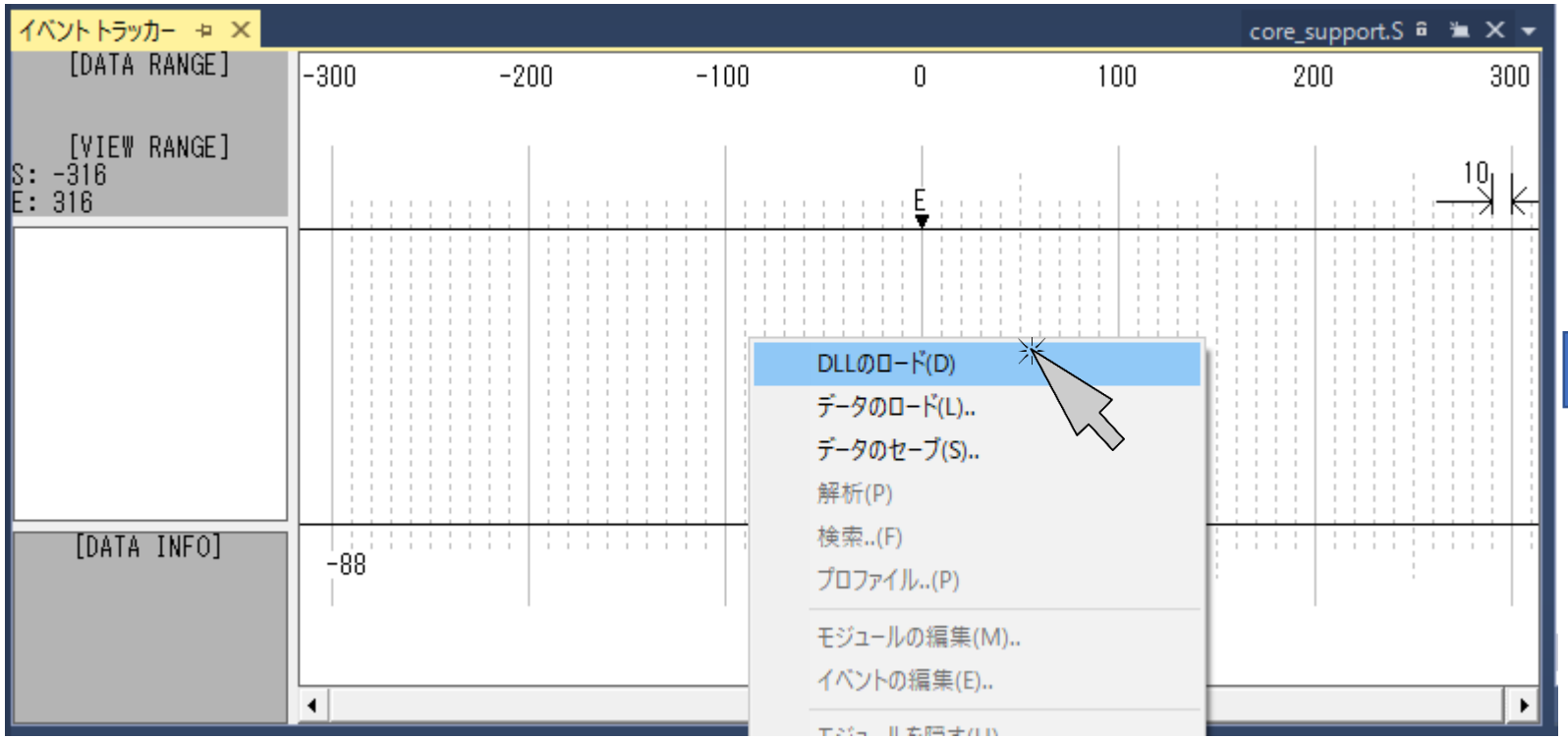




これがイベントトラッカの
表示画面です

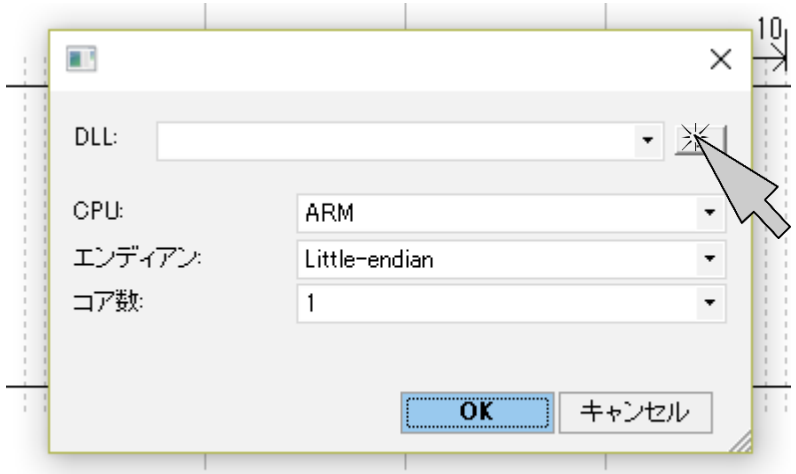
イベントトラッカの使い方

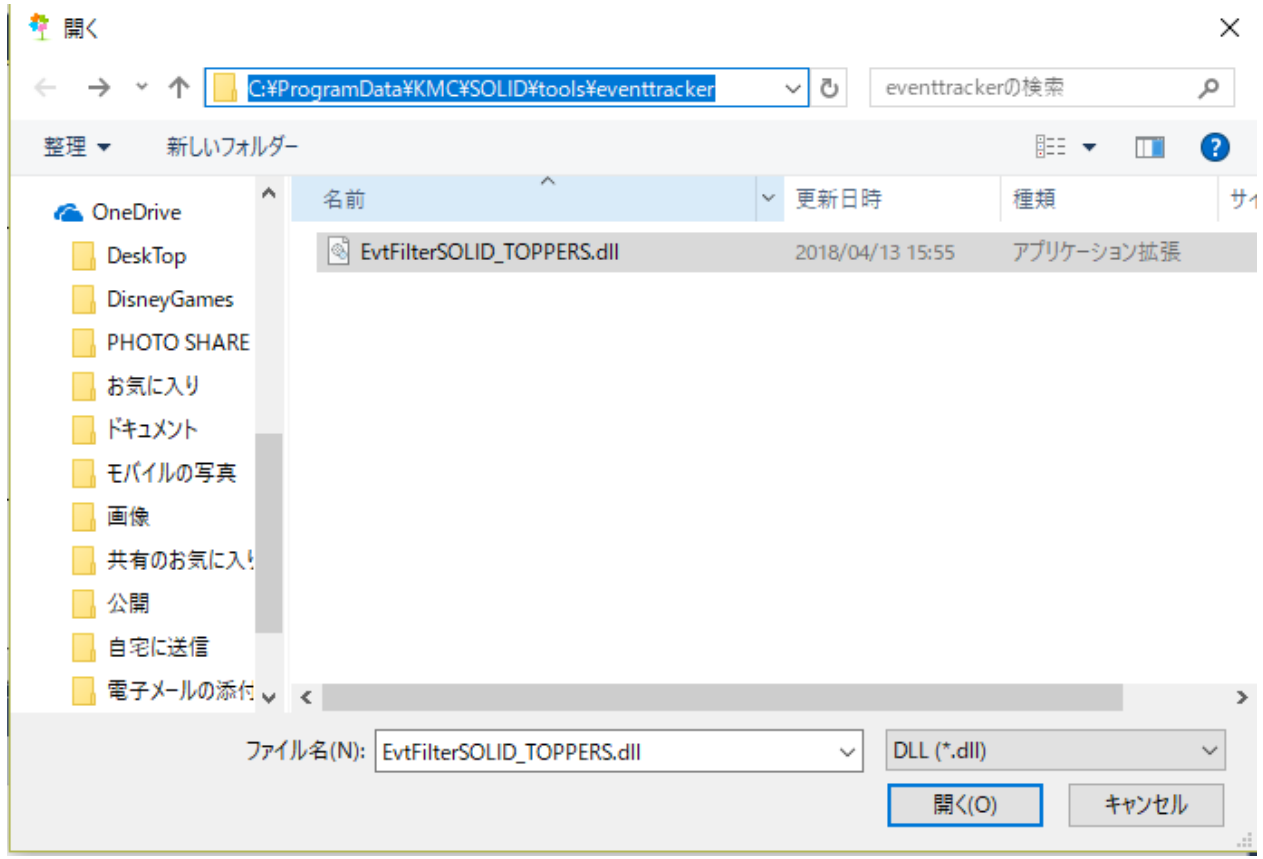
4. イベントトラッカの画面にマウスカーソルを移動し、右クリックで表示されたメニューから[DLLのロード(D)]を選ぶ



イベントトラッカの使い方

5. EvtFilterSOLID_TOPPERS.dll を選ぶ (在処は次ページ)





選択後 OK をクリック。

イベントトラッカの使い方

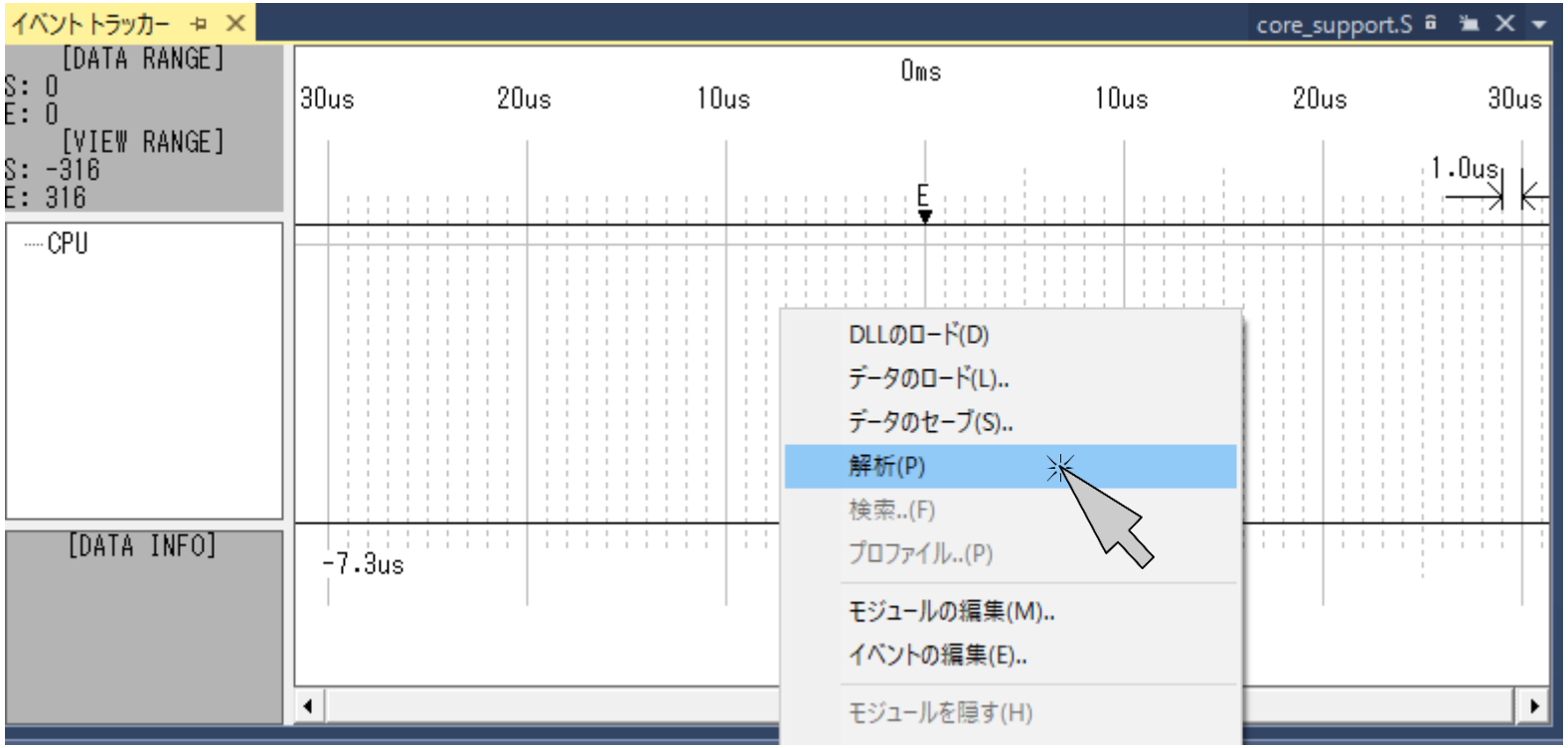
EvtFilterSOLID_TOPPERS.dll の在処

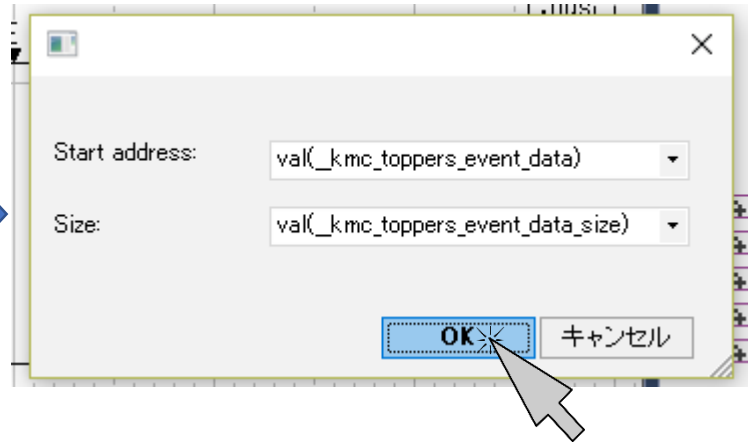
- StarterKit の環境
  C:¥ProgramData¥KMC¥SOLID¥tools¥eventtracker¥

- 製品版環境
  (インストール先フォルダ)¥tools¥eventtracker¥dll¥

イベントトラッカの使い方

6. イベントトラッカの画面にマウスカーソルを移動し、右クリックで表示されたメニューから[解析(P)]を選ぶ

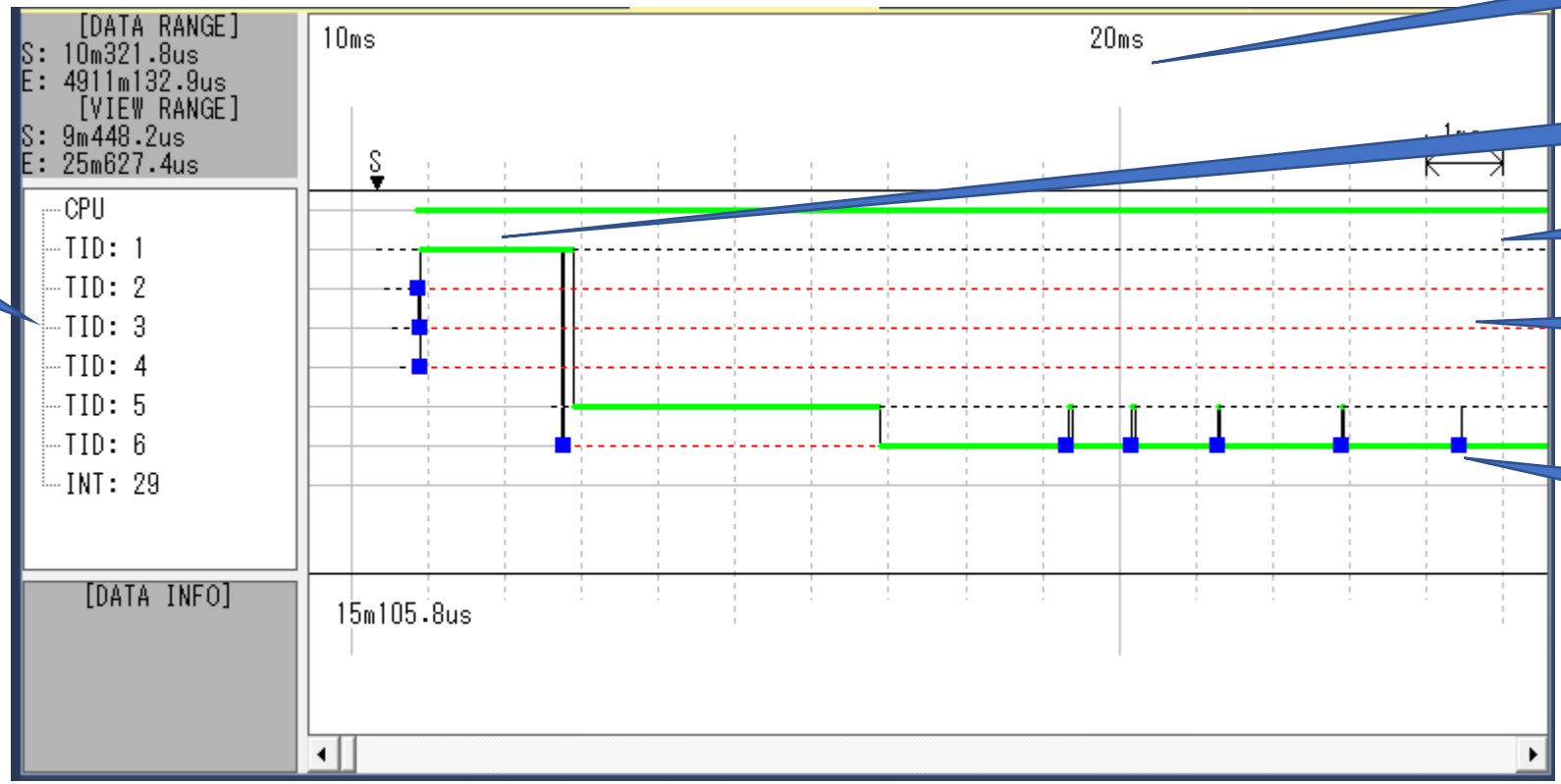




範囲指定ダイアログでは[OK]をクリック

イベントトラッカの結果を表示

- 機能が豊富にあるのですが、ここでは基本的な操作のみ説明します。



モジュール名
TID: タスク
INT: 割り込み
CYC: 周期ハンドラ

時間軸

緑実線:CPU実行中

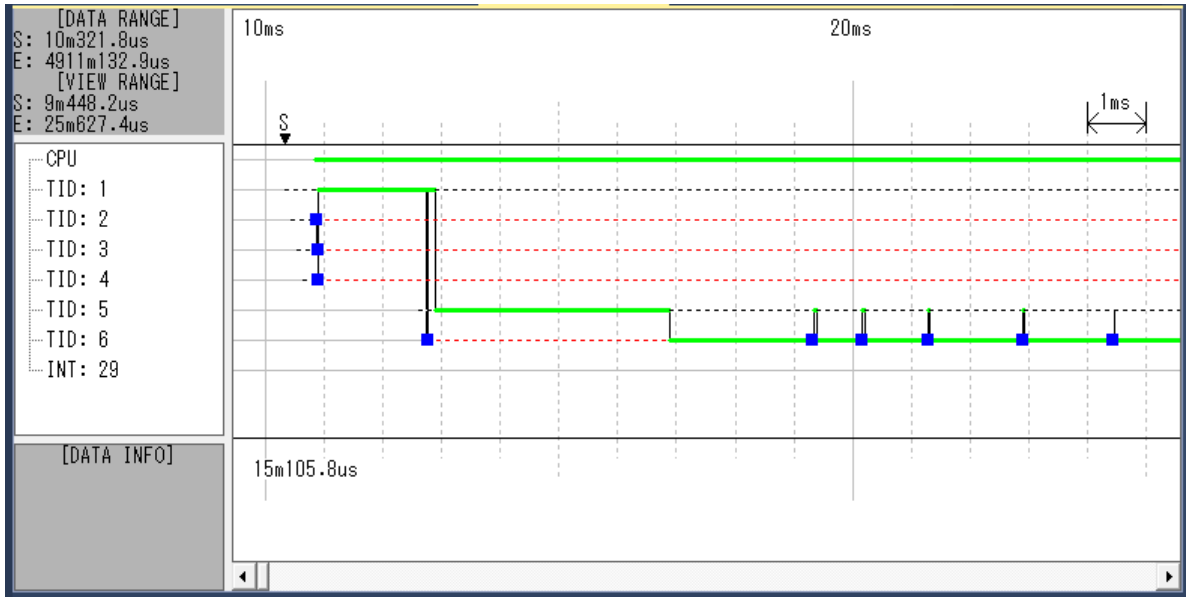
黒破線:CPU実行可能

赤破線:待ち状態

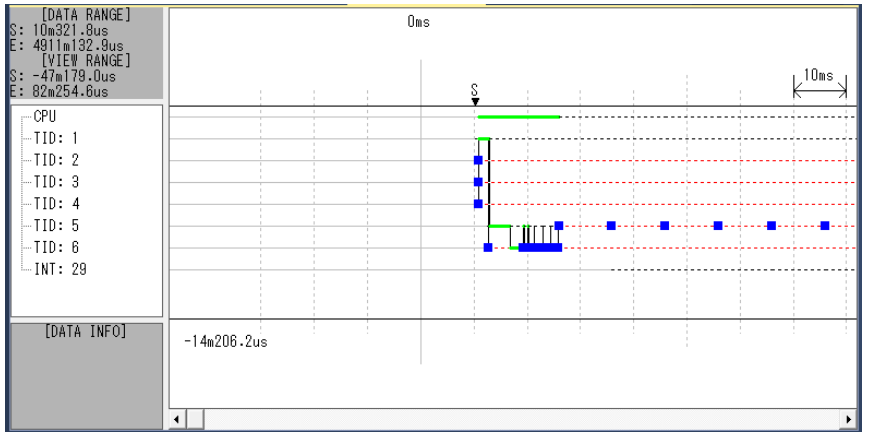
青四角:
待ち遷移イベント

イベントトラッカの結果を表示

- マウスのホイールを回すと、時間軸が Zoomin, Zoomout します。



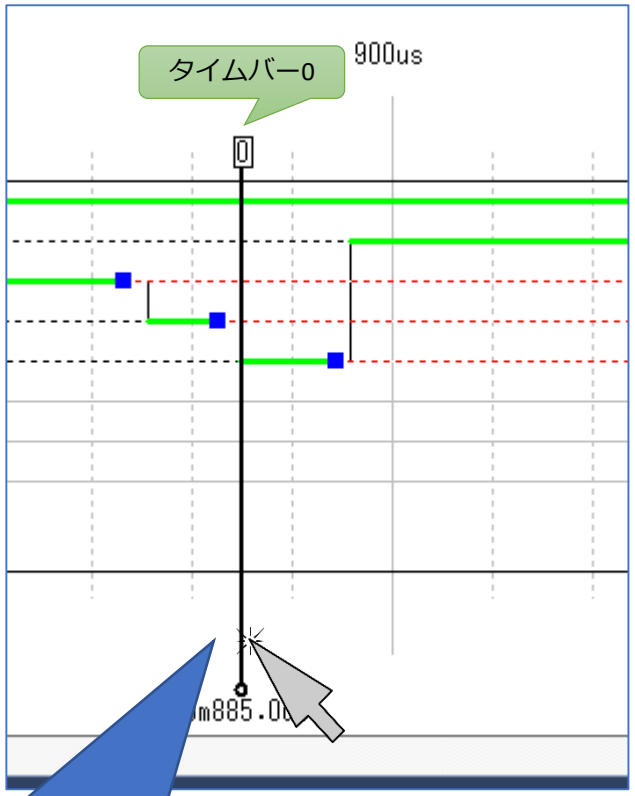
Zoomout



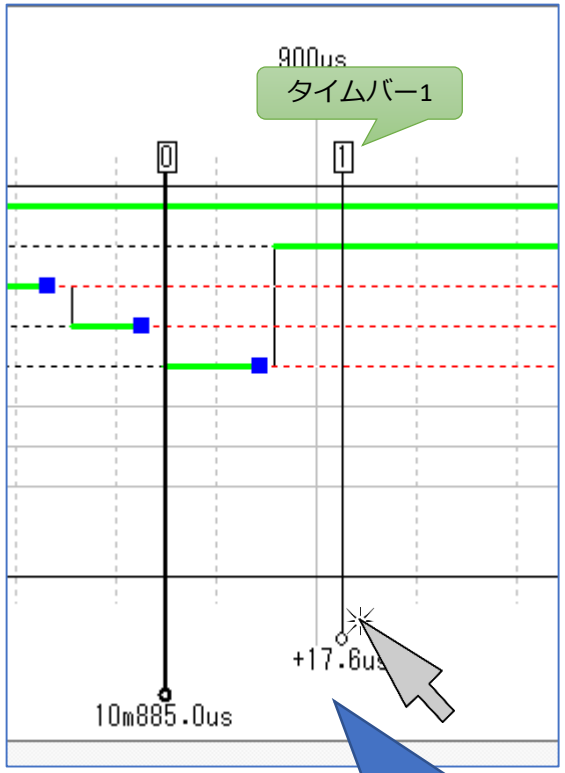
Zoomin

イベントトラッカの結果を表示

- タイムバーを表示し、経過時間を計測できます

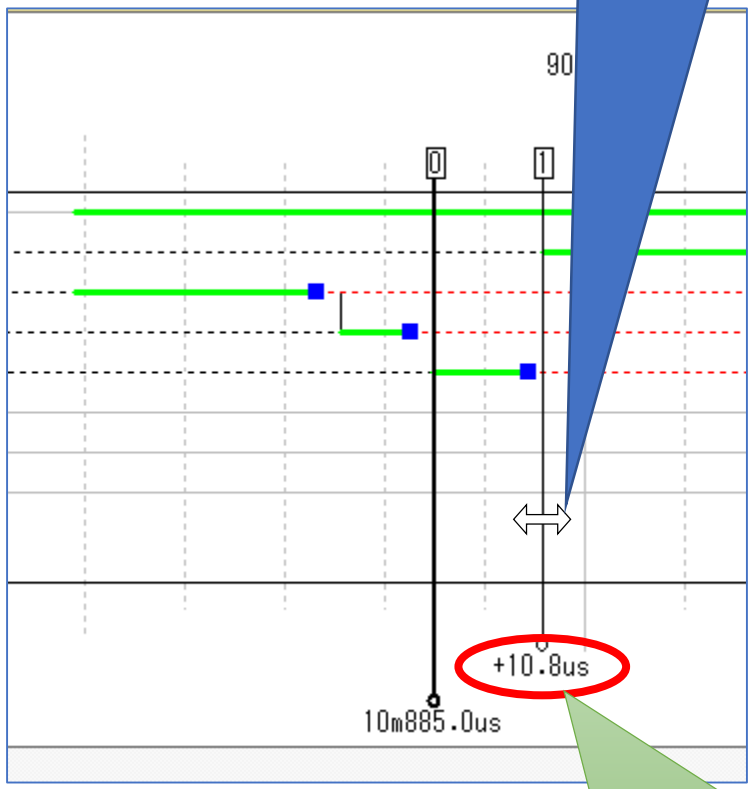


下のエリアでダブルクリックすると
タイムバーが出ます



さらにダブルクリックすると新しい
タイムバーが出ます

タイムバーは左クリックしながら
左右に移動できます



タイムバー0 から タイムバー 1 までの経過時間

タイムライン上の線と四角の色の意味

線種・色	状態
緑実線	CPU実行状態 (Running) ※FMPの場合 Core0実行状態
黒破線	CPU実行権獲得可能 (Ready)
赤破線	待ち状態 (Wait, Suspend, Suspend-wait)
オレンジ実線	FMPのみ Core1 実行状態 (Running)
紫実線	FMPのみ Core2 実行状態 (Running)
水色実線	FMPのみ Core3 実行状態 (Running)

四角の色	イベントの意味
青四角	待ち状態イベント (Wait に遷移)
赤四角	強制待ちイベント (Suspend に遷移)
紫四角	二重待ちイベント (Suspend-wait に遷移)
赤四角	メールボックス送信イベント

メモリ使用量、処理時間
オーバーヘッド

イベントトラッカのオーバーヘッド

- メモリ使用量
  .data セクションに デフォルト 24byte * 8kエントリのイベント記録用メモリを使用します。

- 処理時間オーバヘッド
  イベントを記録するための処理量は、全体の処理量と比較して十分に少ない（ディスパッチが起きる回数は十分少ない）ので、処理時間のオーバーヘッドは無視できます。

付録

必要なタイマの実装

- システム横断で一意なカウンタ値が読める64bit長タイマを選択し、以下の2つの関数を実装する必要があります。

```
#include “impl_globaltick.h”
void IMPL_GLOBAL_TICK_Init(IMPL_GLOBAL_TICK_INFO *pInfo);
{
        // 最低限必要な処理
        pInfo->tickspersec = (64ビットカウンタが1秒にカウントするカウント数);

        // 64bitカウンタの初期設定・起動処理が必要なら追加
}

unsigned long long IMPL_GLOBAL_TICK_GetCurrent()
{
        return (64bitカウンタの現在の値);
}
```

記録するイベントの個数の変更

- 記録イベント数の変更は、kernel_cfg.h で、以下のマクロを宣言してください。

```
#define SOLID_MAX_EVENT_TRACKER_LOG（イベントの個数）
```

.propsに <LibEventTracker> が無い場合

- 前述のエントリを一行追加します。

```
. . .
<PropertyGroup Label=”UserMacros”>
. . .
<LibEventTracker>true</LibEventTracker>
. . .
</PropertyGroup>
. . .
```

- 加えて、以下の変更も一行追加します。

```
. . .
<ItemGroup>
. . .
<BuildMacro Include=“LibEventTracker”/>
. . .
</ItemGroup>
. . .
```

以上です