

SOLID IDEのイベントトラッカ機能を使ってみよう！

2019.07.08

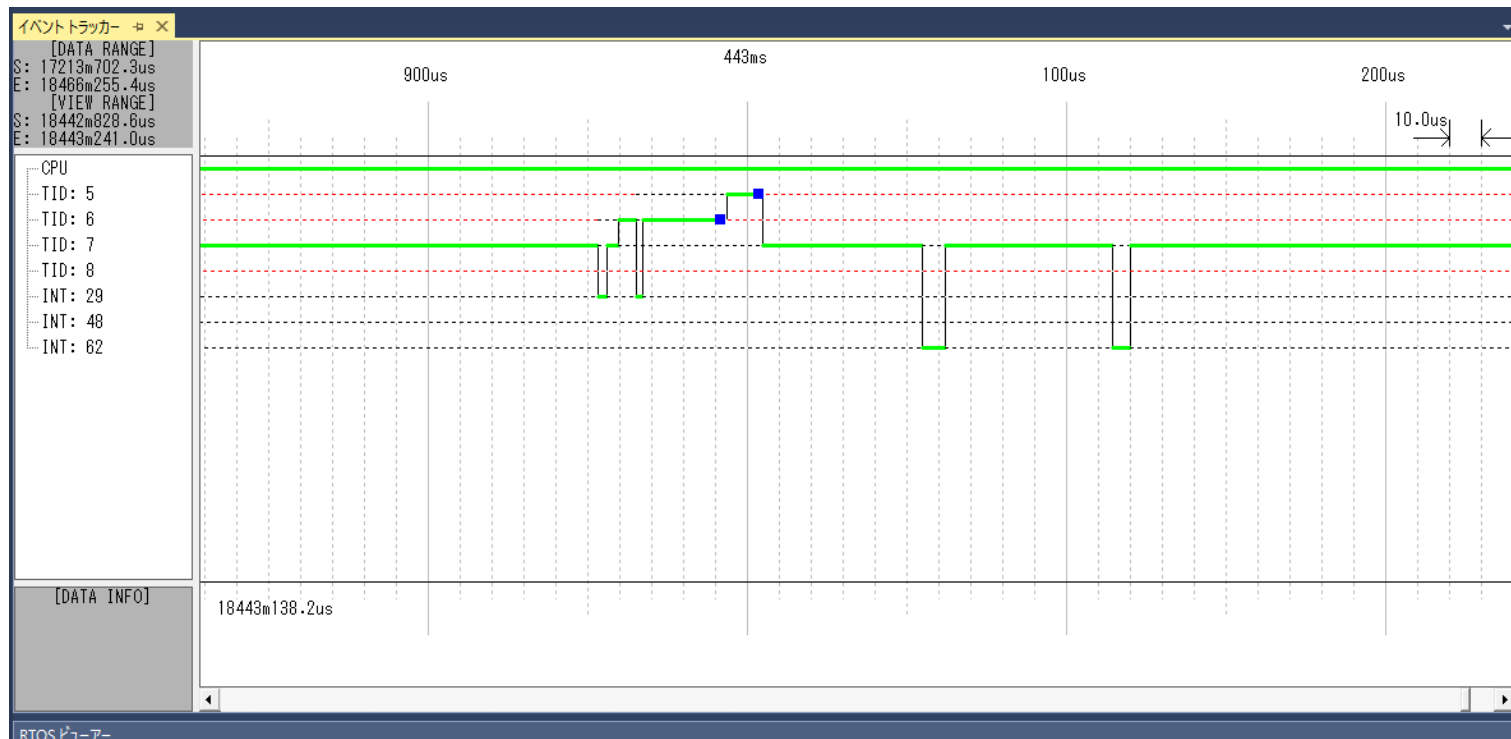
京都マイクロコンピュータ

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。



イベントトラックの概要

- 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イベントトラッカーを 使うための準備

使うために前もって設定すべき内容

イベントトラックを使うための準備

- 既にイベントトラック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ある場合は、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。

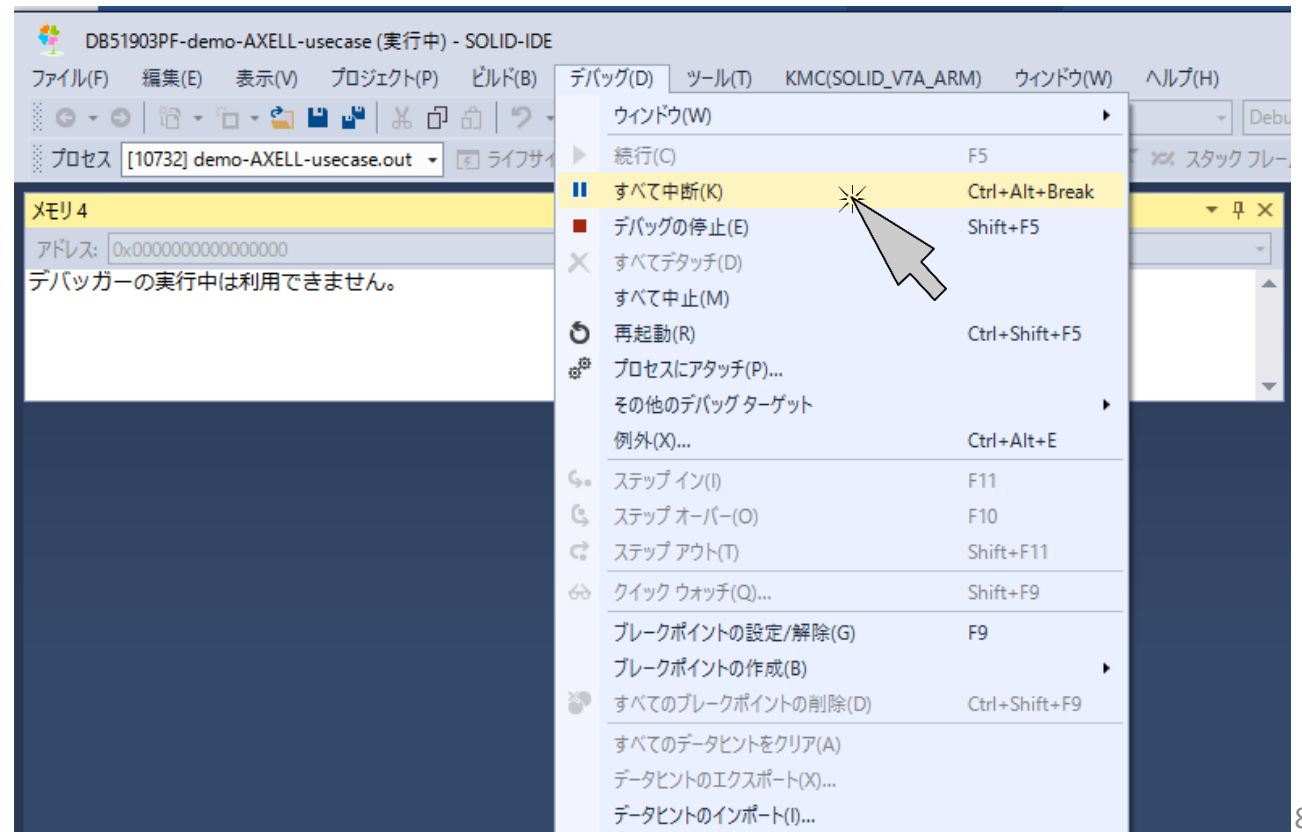
[~~.propsの変更内容]

```
...
<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。

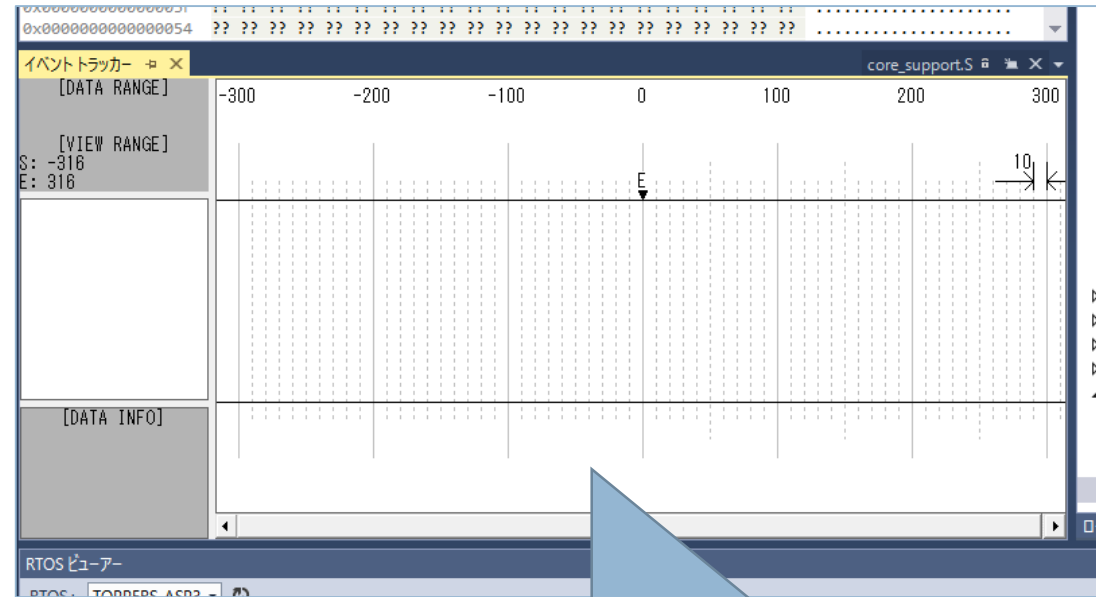
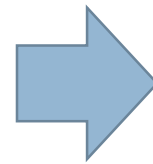
イベントトラッカの使い方

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

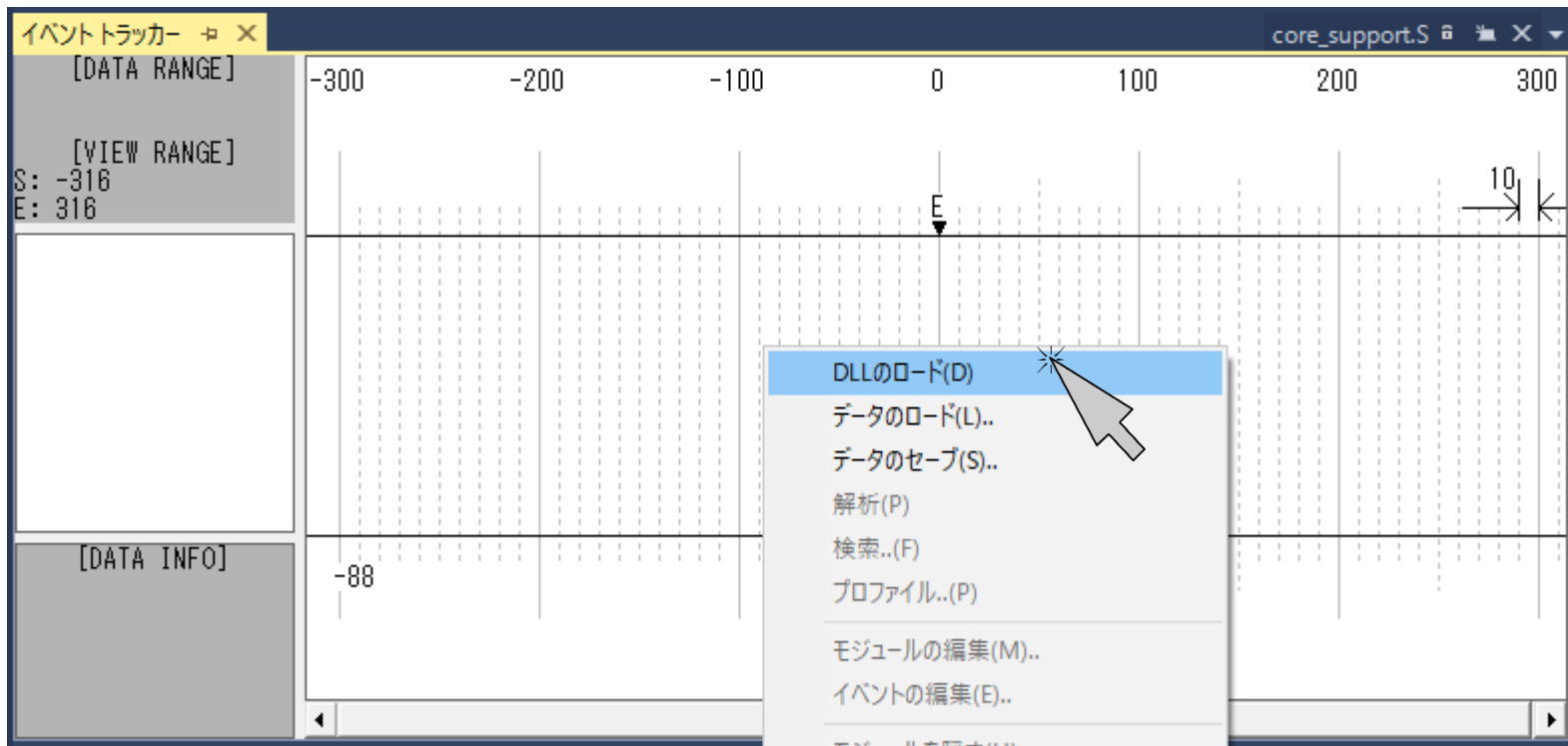
3. メニューの[デバッグ]-[ウィンドウ]-[イベントトラッカ]を選択



これがイベントトラッカの
表示画面です

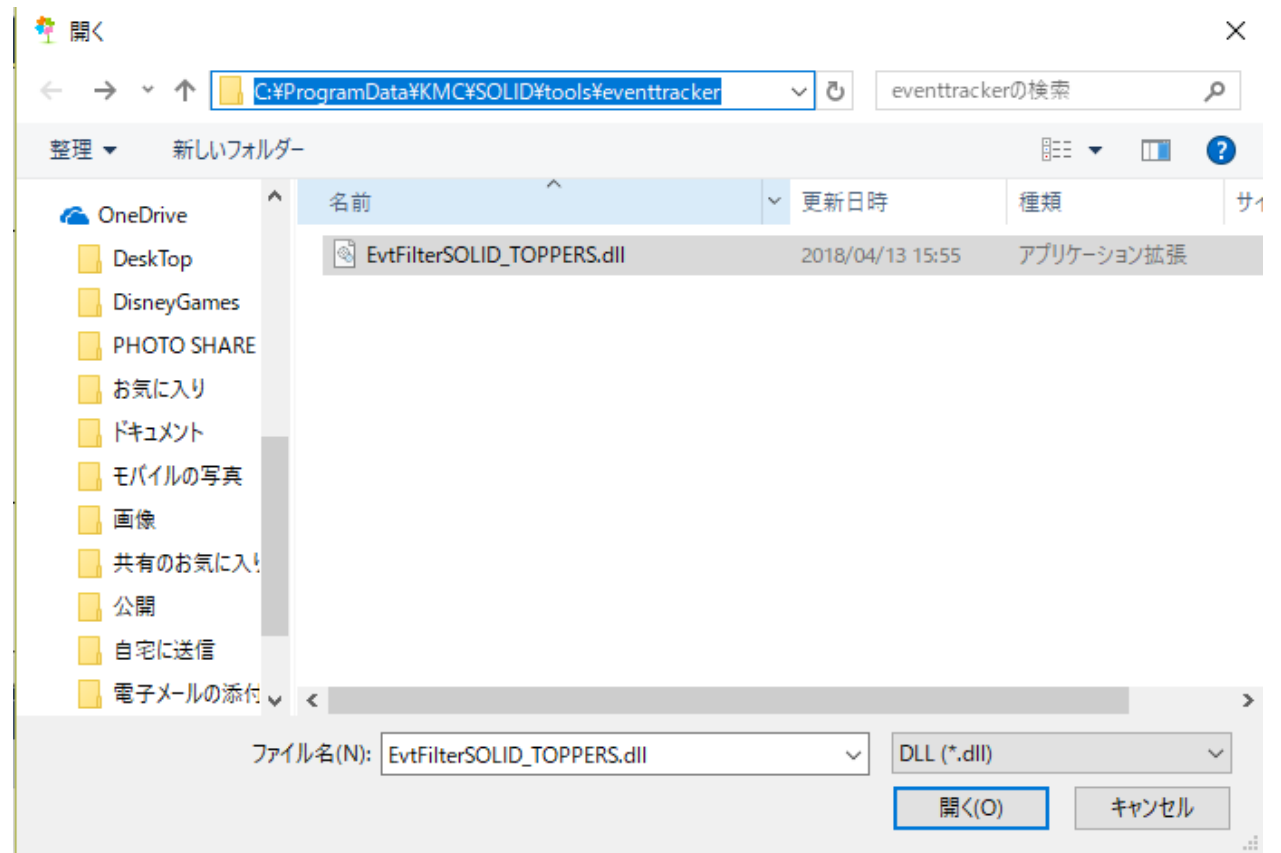
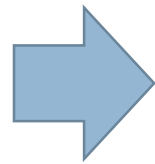
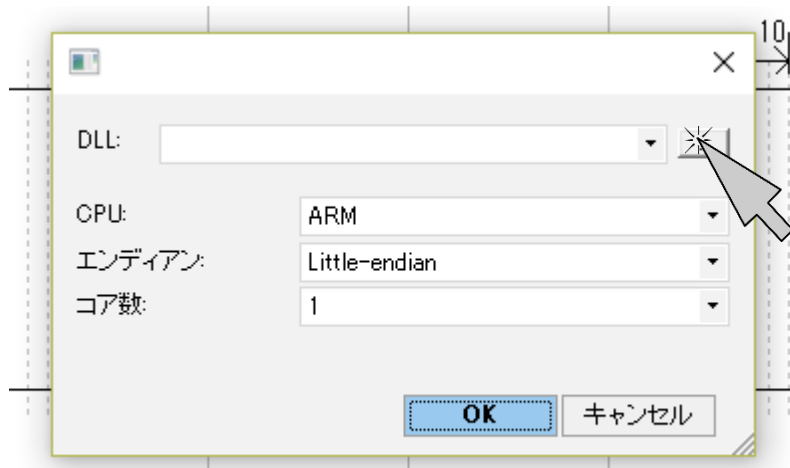
イベントトラッカの使い方

- 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



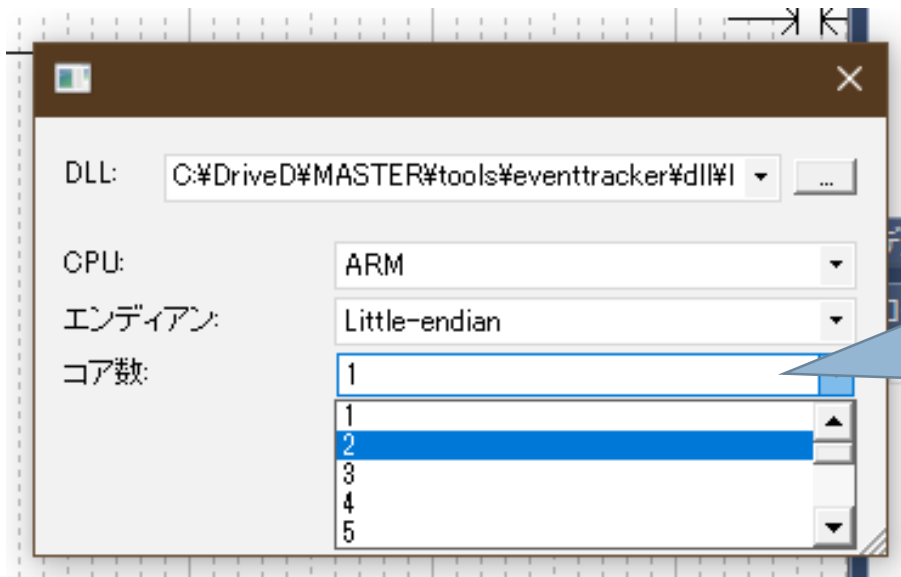
イベントトラッカの使い方

5. EvtFilterSOLID_TOPPERS.dll を選ぶ (所在はP.13)



選択後 OK をクリック。

TOPPERS/FMP の場合はコア数指定必要



DLL を選択した後、

[コア数]のリストボックスから、
SOLID-OSを実行しているコアの
数を選択してから、

[OK] を押します

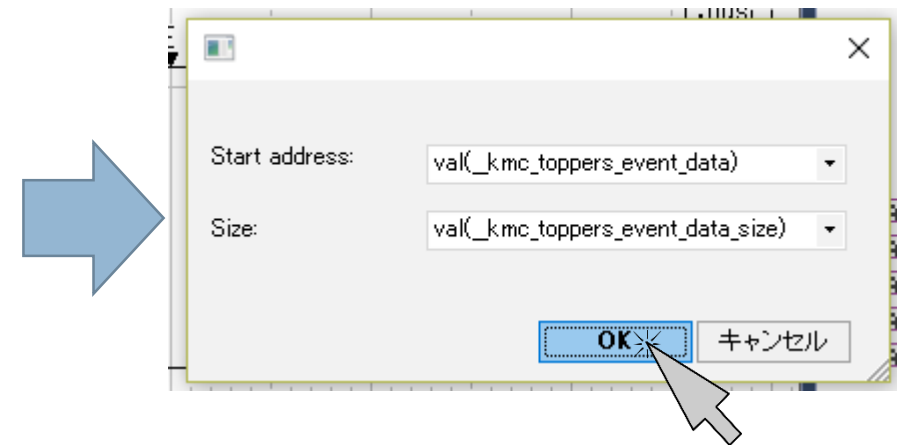
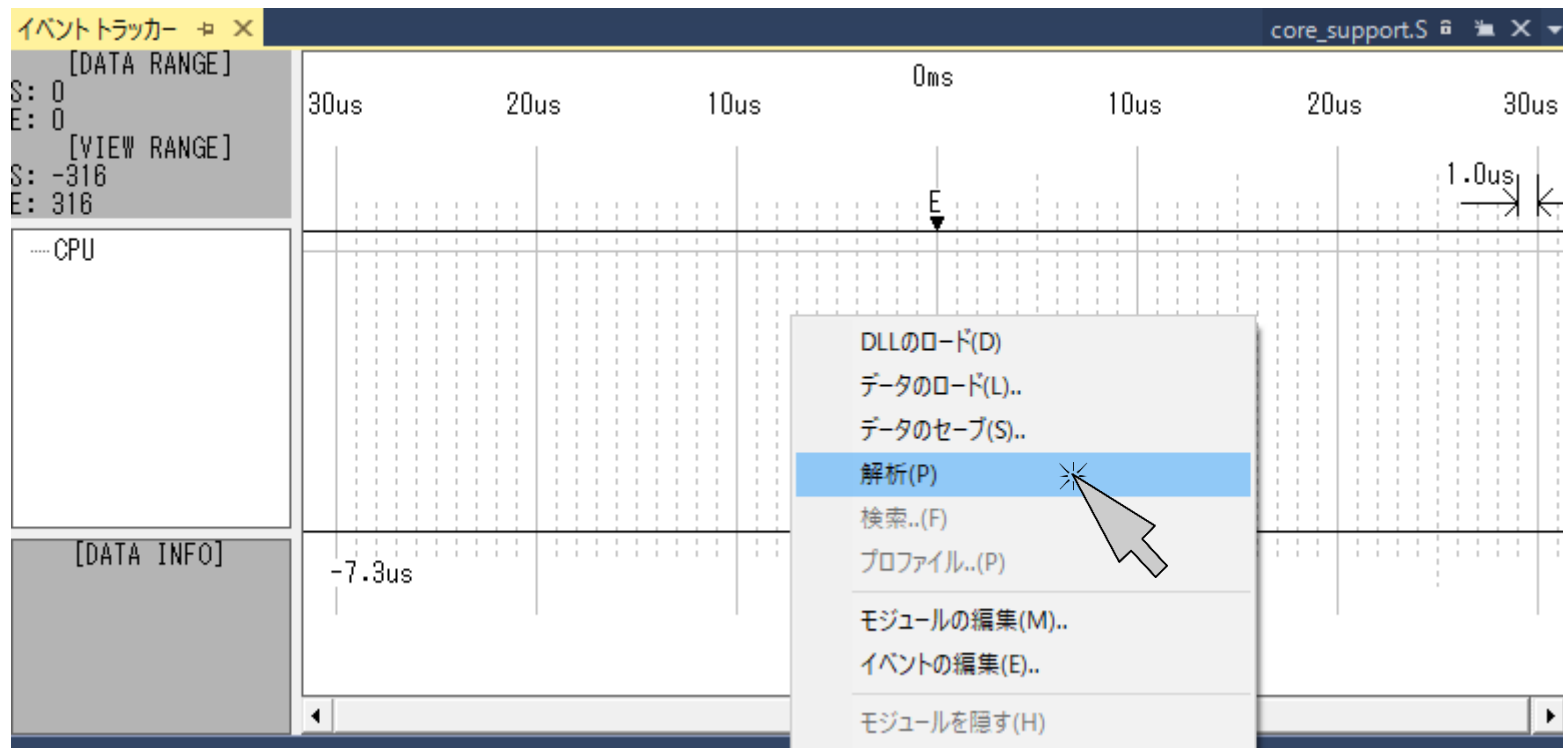
イベントトラッカの使い方

EvtFilterSOLID_TOPPERS.dll の所在

- 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

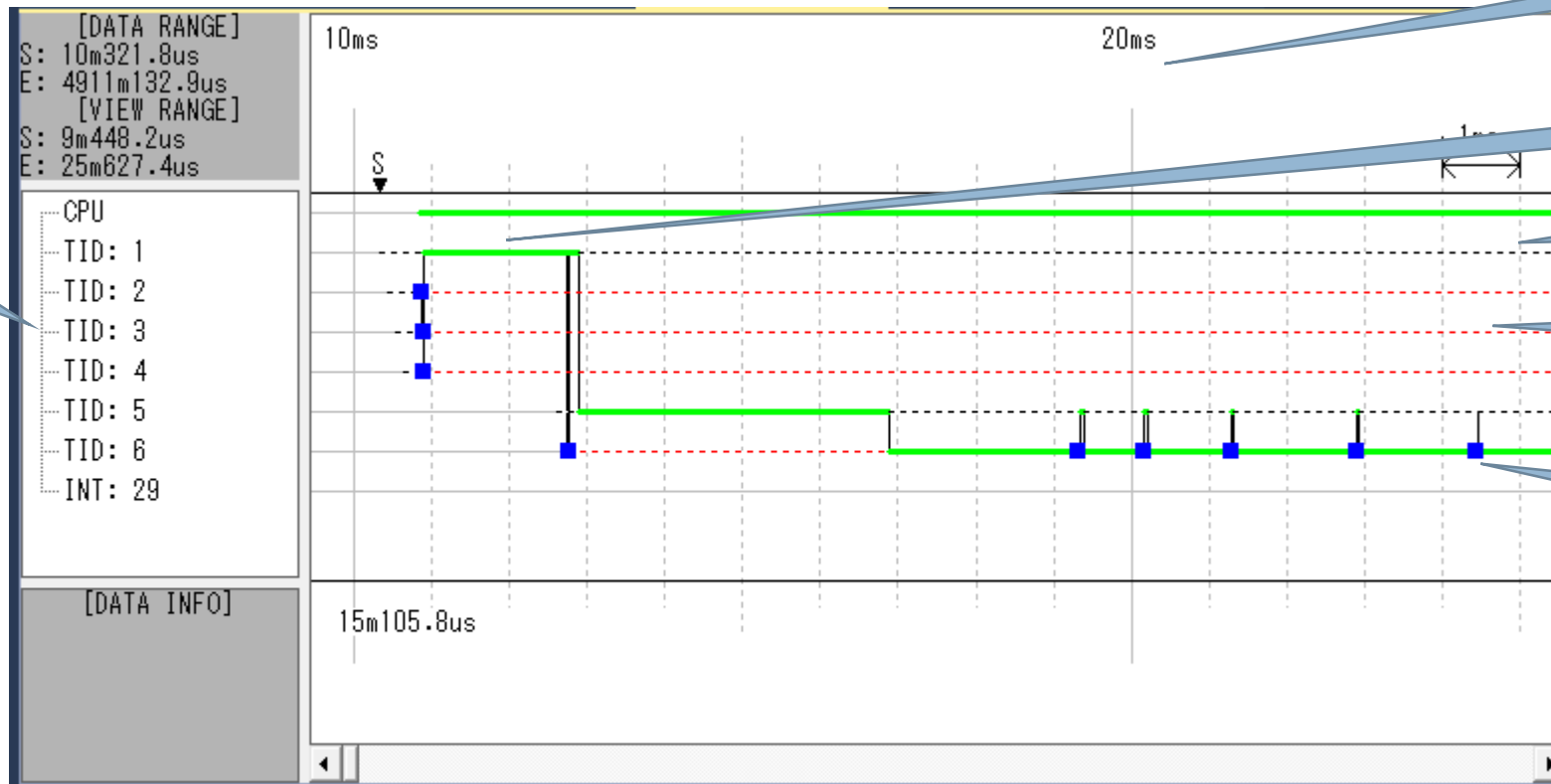
6. イベントトラッカの画面にマウスカーソルを移動し、右クリックで表示されたメニューから[解析(P)]を選ぶ



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

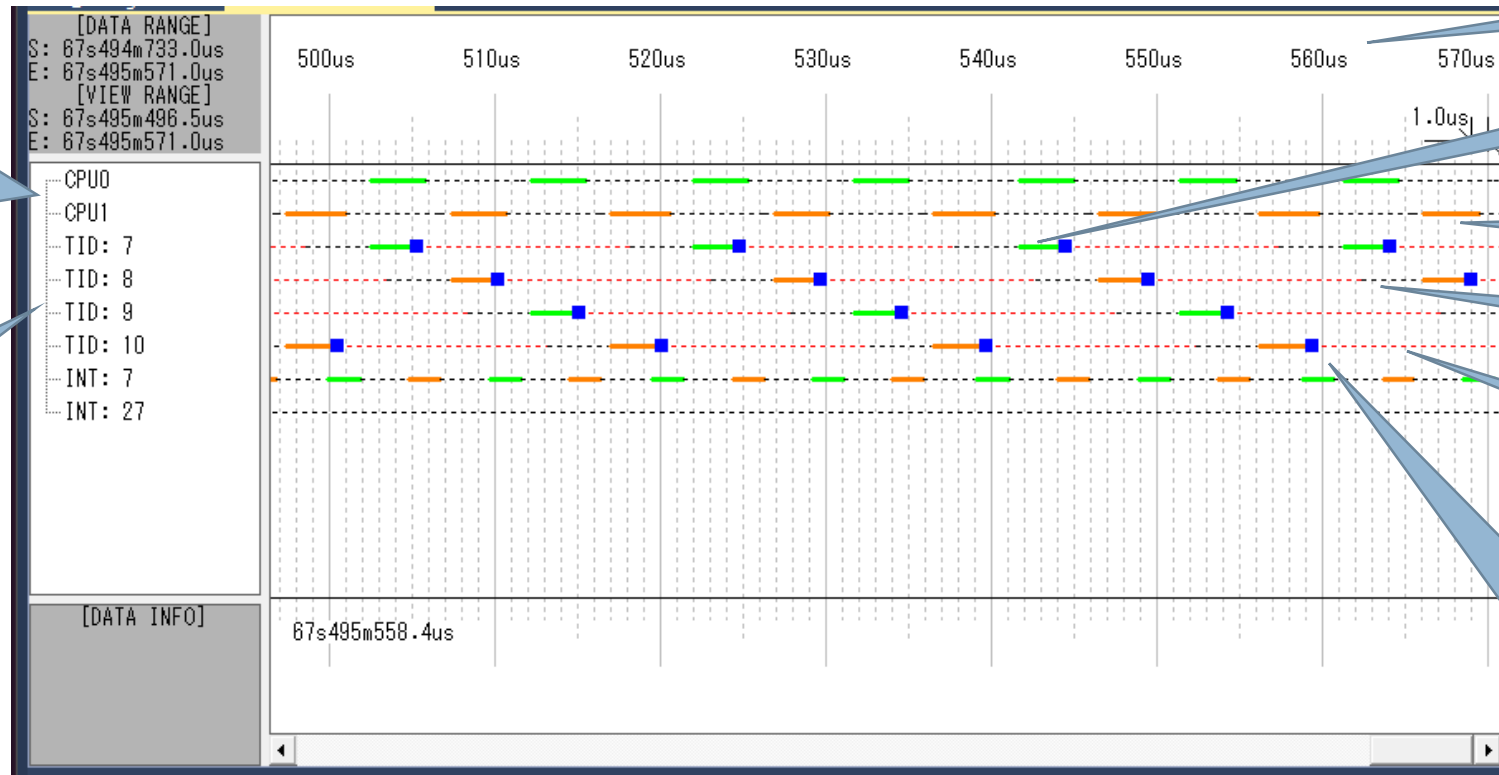
黒破線: CPU実行可能

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(FMP)

- 機能豊富ですが、ここでは基本的な操作のみ説明します。



CPUコア

時間軸

緑実線: CPU0で実行中

橙実線: CPU1で実行中

黒破線: CPU実行可能

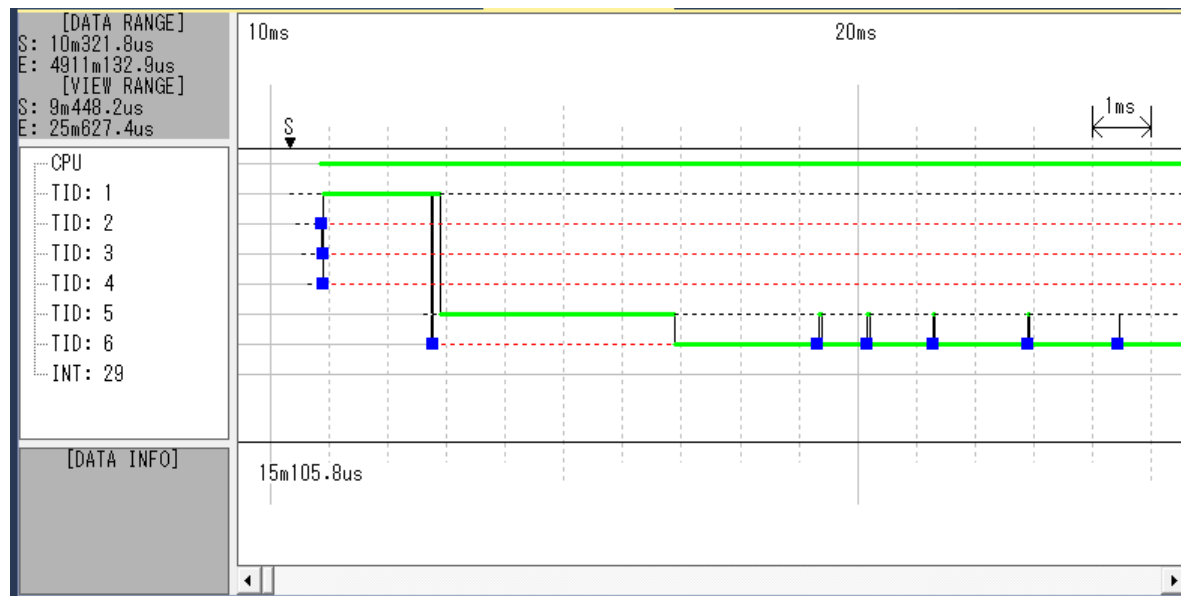
赤破線: 待ち状態

青四角: 待ち遷移イベント

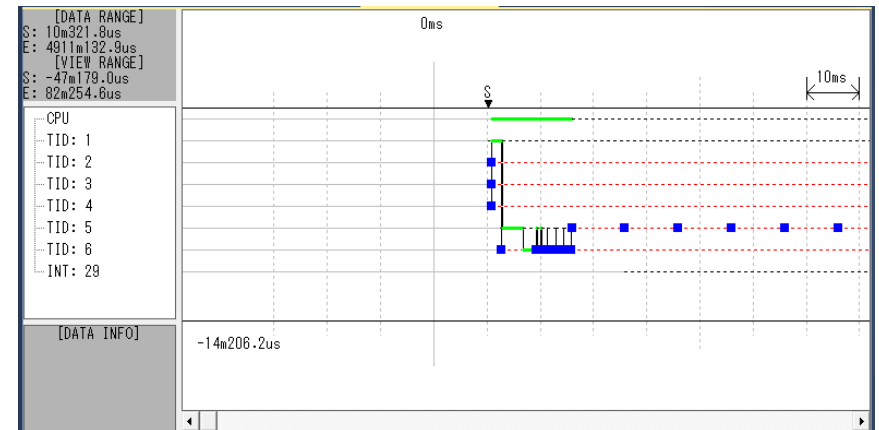
モジュール名
TID: タスク
INT: 割り込み
CYC: 周期ハンドラ

イベントトラッカの結果を表示

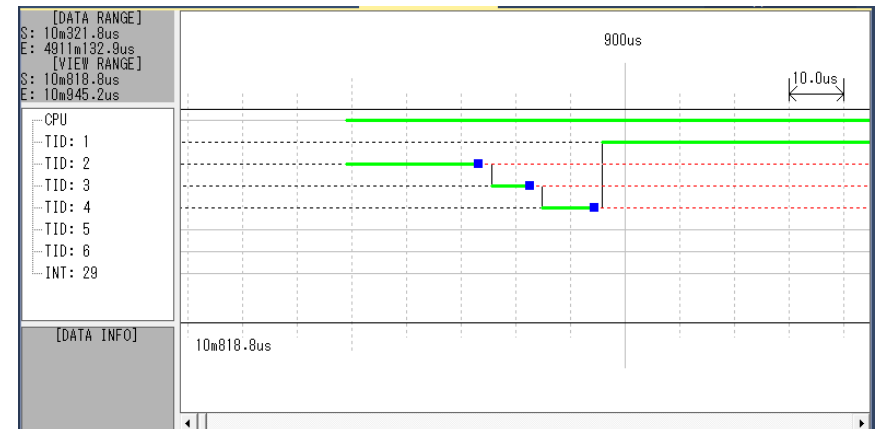
- マウスのホイールを回すと、時間軸が Zoomin, Zoomout します。



Zoomout

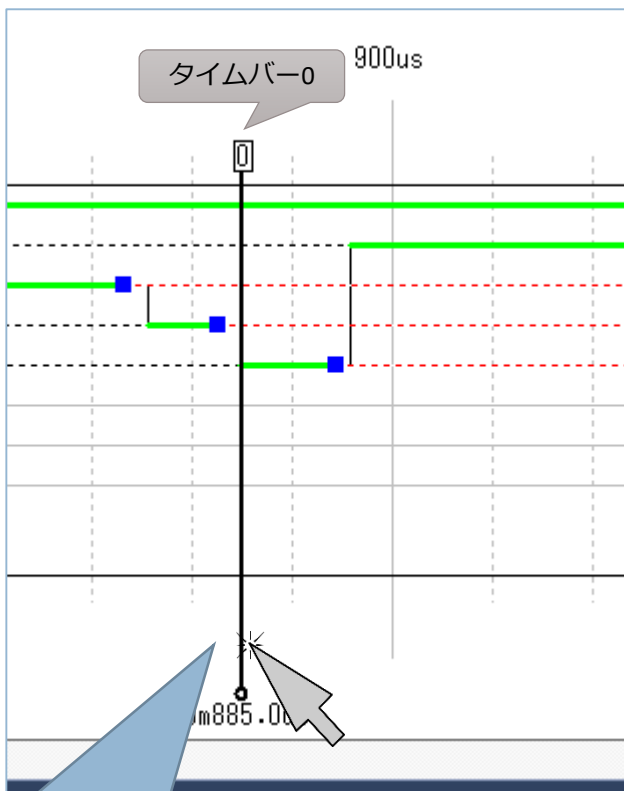


Zoomin

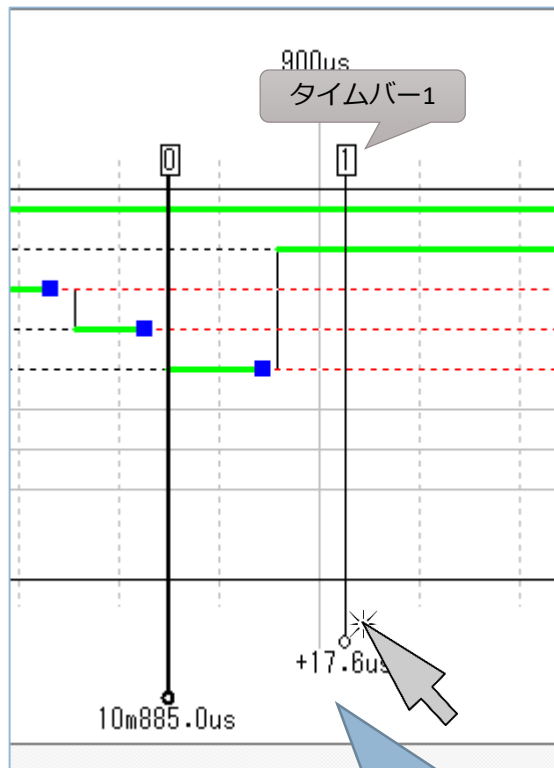


イベントトラッカの結果を表示

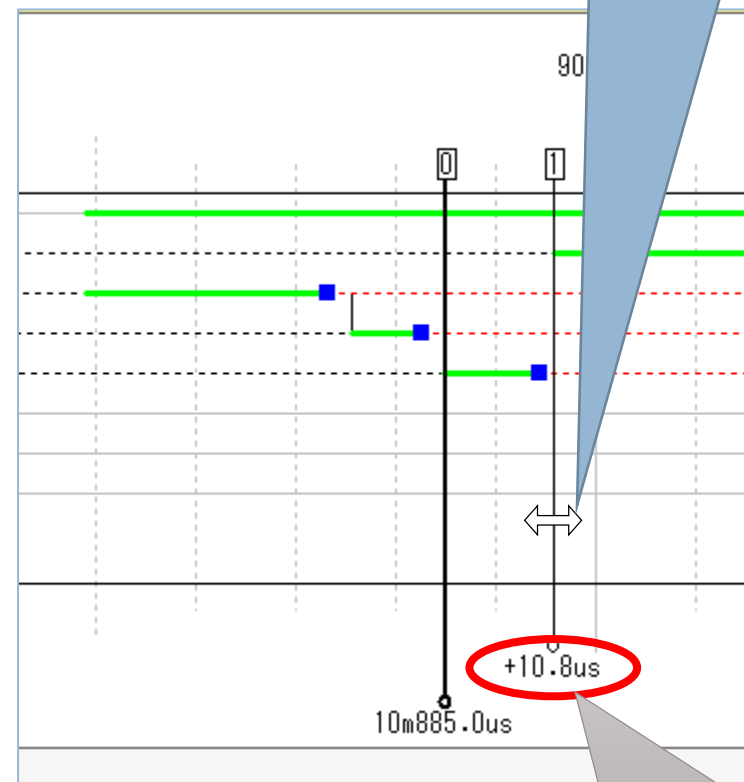
- タイムバーを表示し、経過時間を計測できます



下のエリアでダブルクリックすると
タイマーが出ます



さらにダブルクリックすると新しい
タイマーが出ます



タイマーは左クリックしながら
左右に移動できます

タイマー0からタイマー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)
橙実線	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 $(24\text{byte} * \text{最大イベント数}) + 32\text{byte}$ の領域を使用します。
注意: 最大イベント数は 2 の n 乗の数字である必要があります
- 処理時間オーバーヘッド
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的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、bsp プロジェクトの impl_evtrk.c の以下のマクロを変更してください

```
#define MAX_EVENT_NUM 0x400
```

- このマクロで保存できるイベント数を指定しますが、処理効率化の為、MAX_EVENT_NUM の値は、必ず 2 の n 乗の数にして下さい。守らない場合は、イベントが正しく記録されません

.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、以下の変更も追加します。

```
...  
<ItemGroup>  
...  
  <BuildMacro Include="LibEventTracker">  
    <Value>$(LibEventTracker)</Value>  
  </BuildMacro>  
...  
</ItemGroup>  
...
```

以上です



更新履歴

日付	更新内容
2019.07.08	<ul style="list-style-type: none">• FMP向け記述の追加• 制限事項や設定内容について記述を追加• 現状の実装とマッチしない記述内容を修正
2018.06.08	<ul style="list-style-type: none">• 初期リリース