Application Note

文書番号: ESC-APN-008-02

[SMT] 通過ポイントにマーク(目印)を入れる方法

マーク関数の挿入により、プログラム実行状況を確認する

はじめに

「[SMT] 通過ポイントにマーク(目印)を入れる方法」(以下、本書)では、ユーザープログラム中のある事象が実行されているか否かを確認する方法として、システムマクロトレースの活用例をご紹介します。

本書の対象となるユーザー

- macroTRACE-VIEWER をご使用になる、すべてのユーザー。

本書の対象となる環境

- adviceLUNA システムマクロトレースハードウェア
  - AP511, AP512, AP514 (adviceLUNA 本体)
  - AQ720 (SD カードインターフェース対応システムマクロトレースプローブ)
  - AQ721 (パケット/外部バスインターフェース対応システムマクロトレースプローブ)
  - AQ723 (USB(device)インターフェース対応システムマクロトレースプローブ)
- TRQer (TRQer 本体)
- TLA000 (システムマクロトレース ソフトウェアキット) Ver.1.00 以降。

注意事項

- 本書で使用している画面は、Windows 7 の環境で作成しています。ほかの環境をお使いの場合、表示や操作手順が異なることがあります。
- 本書の操作手順は、TLA000 に含まれる make_cp, CheckPointEditor, macroTRACE-VIEWER を使用して解説しています。お使いの機種によっては、操作手順内での名称や参照しているマニュアル名が異なる場合があります。
- advice シリーズの製品を安全にお使いいただくために重要な情報は、『adviceLUNA ユーザーズマニュアル（共通編）』（adviceLUNA_podm_jpn.pdf）に記載されています。
- 本書に記載されている会社名・製品名は、各社の登録商標または商標です。

アイコンについて

本ガイドで使用しているアイコンには、以下の意味があります。

	特に重要な情報を記載しています。操作する際は十分に注意してください。
	操作を進める上で役に立つ情報やアドバイスなどの補足事項を記載しています。
	本ガイドのほかのページやほかのマニュアルなどの参照情報を記載しています。

Application Note

目次

はじめに 1
本書の対象となるユーザー 1
本書の対象となる環境 1
注意事項 2
アイコンについて 2
1. 概要 4
1.1. マーク関数について 4
1.2. 必要な環境 4
2. 手順 5
2.1. トレース対象ソースの編集 5
2.2. ビルド 5
2.3. macroTRACE-VIEWER の操作 6
3. 制限・注意事項 8
改訂履歴 9

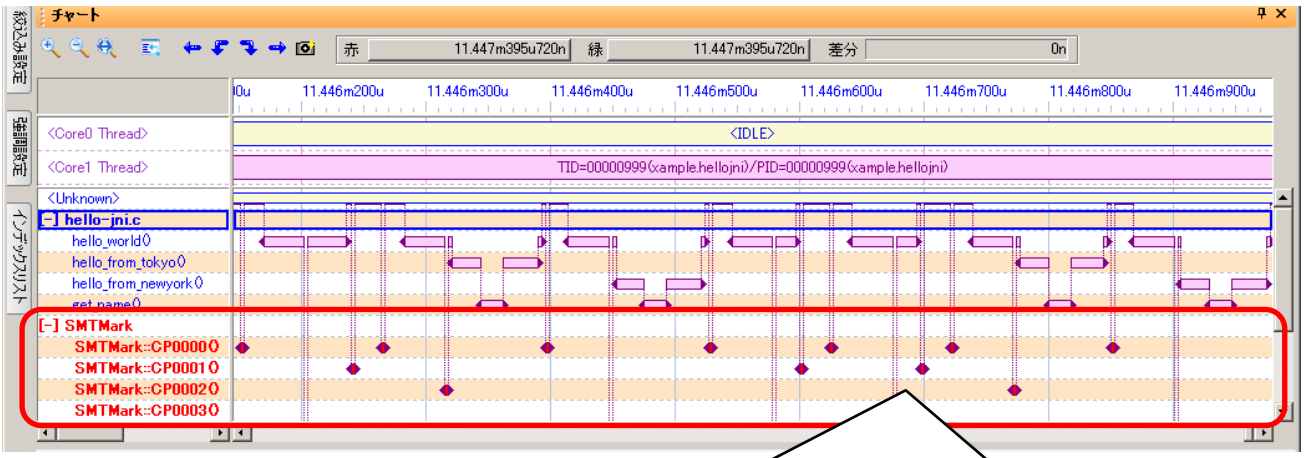
Application Note

1. 概要

1.1. マーク関数について

システムマクロトレースでは、macroTRACE-VIEWER の関数遷移チャート表示より、ユーザープログラムの実行履歴を「関数単位」で確認することができます。
該当するプログラムのポイントに、マーク(目印)となる関数を挿入することによって、次のようなケースも観測することができます。

・関数内のある事象が実行されたことを確認したい
・関数内の条件分岐で、どの分岐が発生したかを知りたい



トレース対象ソース内の任意の箇所にマーク関数を挿入することにより、通常の関数トレースに加えて、マーク(目印)となる表示を行うことができます。

1.2. 必要な環境

- adviceLUNA、または、TRQer
- adviceLUNA システムマクロトレースプローブ (下記のいずれか)
  - AQ720 (SD カードインターフェース対応システムマクロトレースプローブ)
  - AQ721 (パケット/外部バスインターフェース対応システムマクロトレースプローブ)
  - AQ723 (USB(device)インターフェース対応システムマクロトレースプローブ)
- TLA000 (システムマクロトレース ソフトウェアキット) Ver.1.00 以降。
- マーク関数定義ヘッダファイル “SMTManualCP.h” (圧縮ファイルとして、本書と同梱)
- マーク関数用チェックポイントワークスペースファイル (圧縮ファイルとして、本書と同梱)

2. 手順

2.1. トレース対象ソースの編集

1) 別途提供するヘッダファイル“SMTManualCP.h”を、トレース対象ソースにインクルードします。
2) トレース対象ソースのマーク(目印)を挿入したい箇所に、
   _SMT_CP000();
   のように、関数マクロを追記します。

<例>

```
switch(city) {
case TOKYO:
    _SMT_CP0002();
    status = hello_from_tokyo(person);
    if (status != OK) {
        _SMT_CP0003();
        _SMT_PRINTF(LEVEL_DEBUG, "hello_from_tokyo() returns %d", status);
    }
    break;
case NEWYORK:
    _SMT_CP0004();
    status = hello_from_newyork(person);
    if (status != OK) {
        _SMT_CP0005();
        _SMT_PRINTF(LEVEL_DEBUG, "hello_from_newyork() returns %d", status);
    }
    break;
default:
    _SMT_CP0006();
    _SMT_PRINTF(LEVEL_DEBUG, "No such city %d", city);
    return (ERR_NO_CITY);
}
```

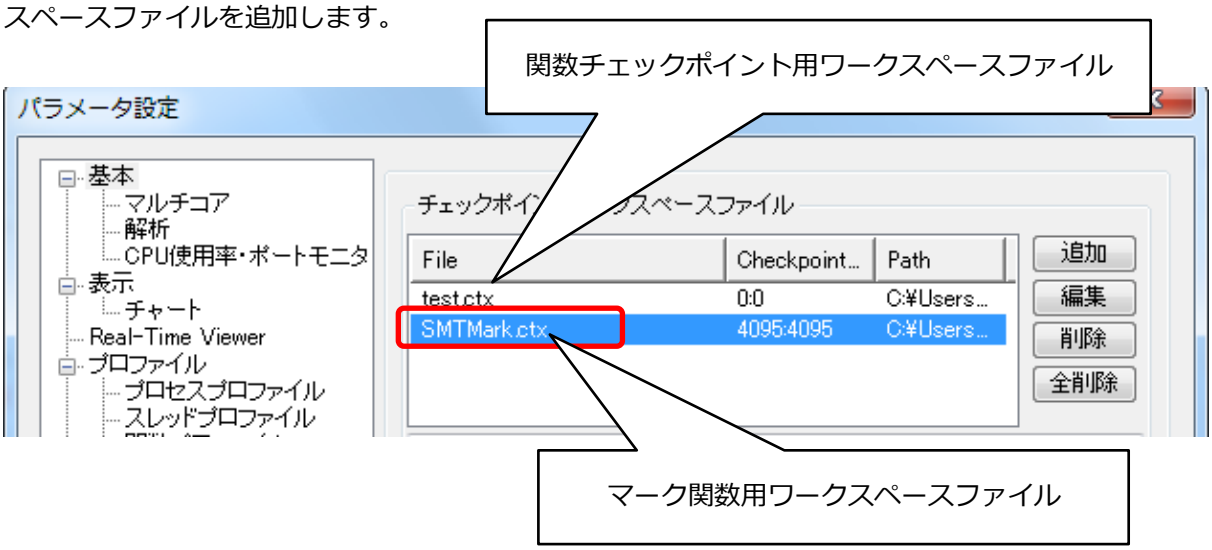
2.2. ビルド

make_cp を使用して、ユーザープログラムをビルドします。

マーク関数を挿入したソースは、CheckPointEditor(Cpe)でチェックポイント設定されている必要があります。

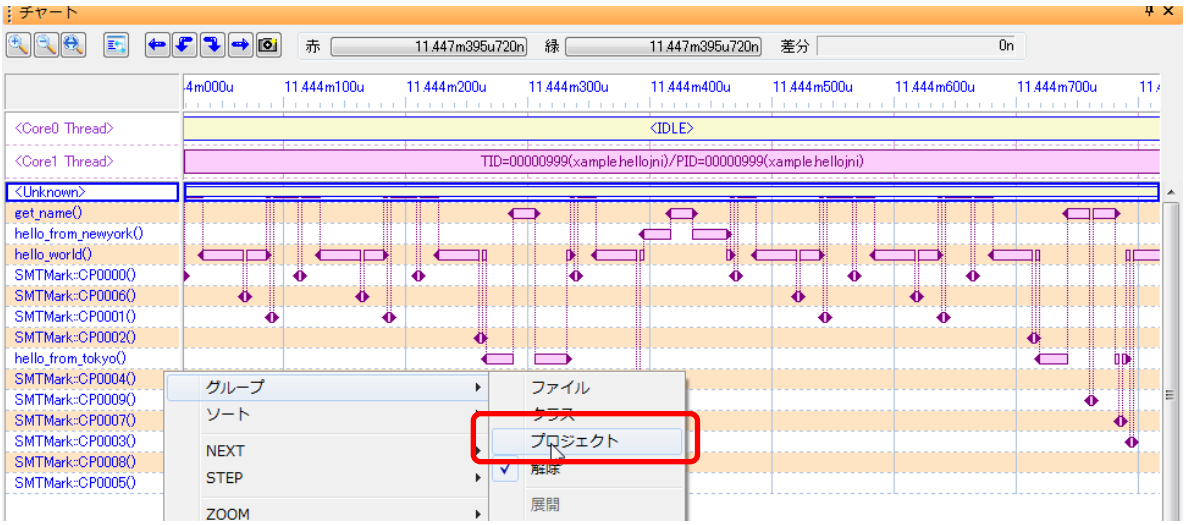
2.3. macroTRACE-VIEWER の操作

1) macroTRACE-VIEWER のパラメータ設定で、別途提供しているマーク関数用のチェックポイントワークスペースファイルを追加します。

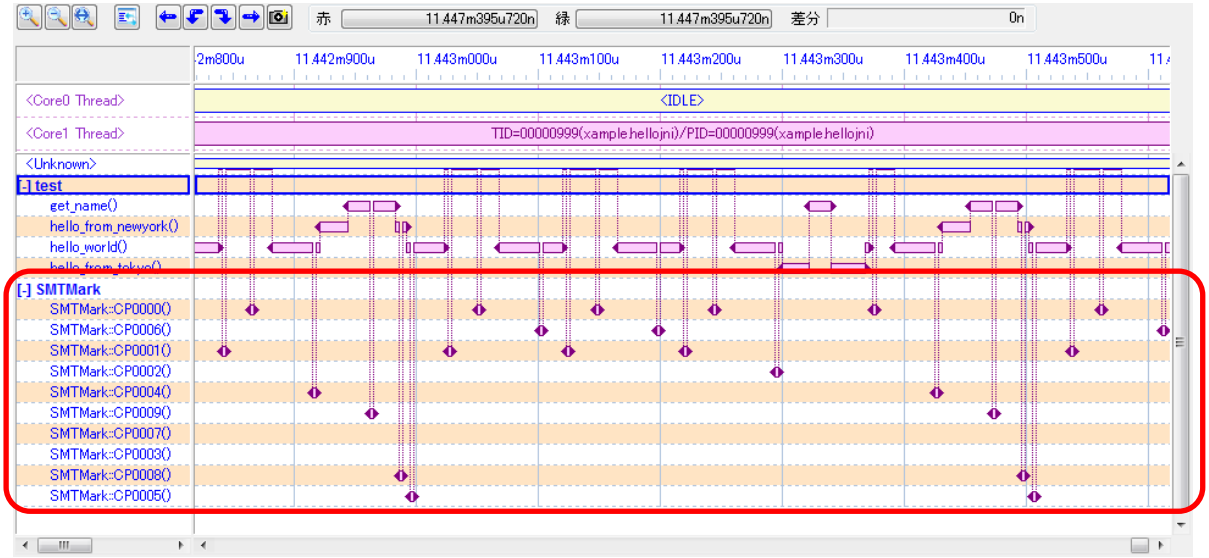


2) ツールバーのトレース開始 / 終了ボタンを使用して、関数トレースをおこないます。

3) 関数チャート表示上で、マーク関数をグループ化することにより、確認が容易になります。
関数名表示ペインを右クリックし、[グループ] - <プロジェクト>を選択します。



< グループ化した後の表示例>



3. 制限・注意事項

- マーク関数は、対象ソースの確認ポイントに手動で追加する(関数マクロ記述)必要があります。
- CheckPointEditor(Cpe)で、チェックポイントを挿入しているソースに対してのみ、マーク関数を実装することができます。
- 本活用例は、macroTRACE-VIEWER の関数チャート表示上で「ある事象を通過したか否かを確認」するためのものです。マーク間の時間を測定する場合、マーク関数の実行時間が含まれます。

改訂履歴

版	発行日付	変更内容
第 1 版	2013.06.15	新規発行
第 2 版	2017.04.01	新商号対応