Cortex-M コア対応エミュレータ
デバッグ制御インターフェースとトレース機能

2013 年 3 月 第 2 版

株式会社 コンピューテックス

CS0015(B)1303

目 次

第1章 はじめに

1.1 はじめに

Cortex-M コアのデバッグ環境を揃えられる際に、JTAG、SWD、ETM、SWV という言葉がキーワードとしてでてきます。
本書では、デバッグ制御インターフェース(JTAG、SWD)やトレース機能(ETM、SWV)について解説を行い、これらの機能を使用するには、どのモデルをどのように使用すればよいのかについても解説します。
本書の内容をご覧いただき、JTAG エミュレータ本体モデルとターゲットインターフェースをご選択ください。

1.2 Cortex-M コアのデバッグ機能

Cortex-M コアのデバッグ機能には様々な機能がありますが、その機能はプログラムのダウンロードや実行／ブレークといった CPU の制御を行なうデバッグ制御とプログラムの実行履歴を出力するトレース機能に、大きく分けられます。

前者の CPU のデバッグ制御を行なう機能としては、CPU が持つデバッグ制御インターフェースの JTAG と SWD があります。これらを通じて ICE 本体から Cortex-M コアを制御することで各種デバッグ機能を実現しています。
後者のトレース機能は、ETM と SWV という機能です。トレース機能は、デバッグ制御部分に付加される形となり、デバッグ制御インターフェースの端子に加えて、トレース端子を接続することで使用可能となります。

これらのデバッグ制御、トレース機能を、それぞれに分けて機能の解説を行っていきます。

。

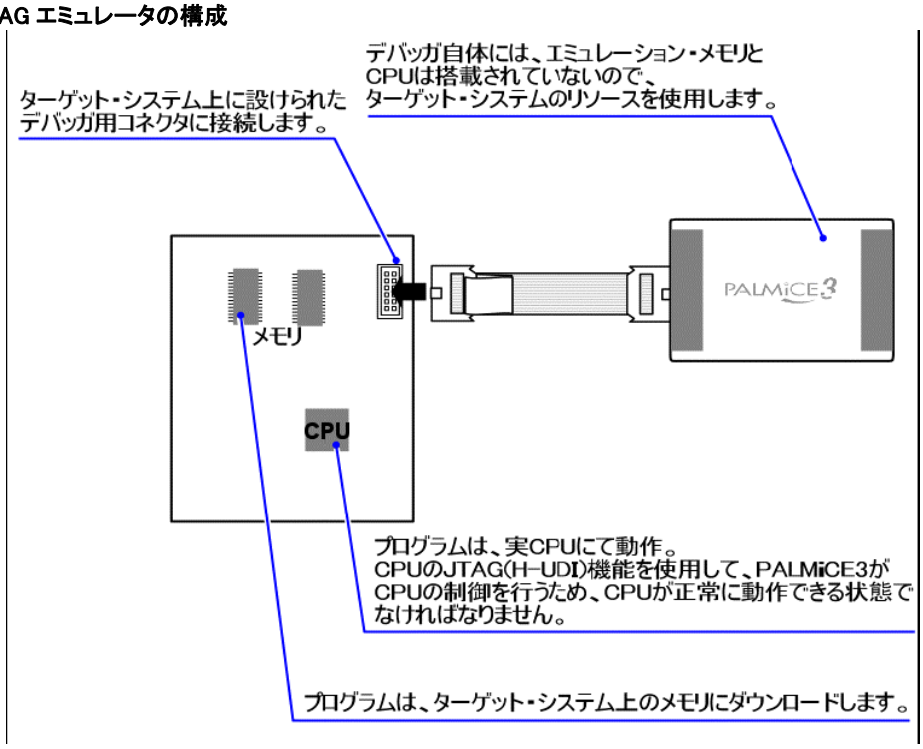
第2章 デバッグ制御インターフェース

本章では、プログラムのダウンロードや実行／ブレークの制御など、デバッガの制御を行なうためのデバッグ制御インターフェースについて説明を行ないます。

2.1 JTAG エミュレータ

現在の Cortex-M コアに対応するデバッガは、JTAG エミュレータが主流となっています。
JTAG エミュレータは、CPU が搭載しているデバッグ機能を、JTAG インターフェースを使用して ICE から制御を行なう方式のデバッガで、実際にターゲットシステム上に製品チップを実装してデバッグを行なうため、製品に近い状態でデバッグを行なうことができることが特徴で、インターフェース名から、JTAG エミュレータと呼ばれます。
この JTAG エミュレータで使用する Cortex-M コアの持つデバッグ機能が、JTAG、SWD、ETM、SWV なのです。

・JTAG エミュレータの構成



・JTAG エミュレータのモデル

Cortex-M コアに対応する JTAG エミュレータには、基本となる JTAG モデルと JTAG モデルにトレース機能を追加した ETM モデル(トレースモデル)の2種類のモデルがあります。

◎JTAG モデル

JTAG モデルは、デバッガの基本機能を実現する JTAG／SWD を搭載したモデルになります。
ターゲットボードとの接続は 20 ピン、もしくは、10 ピンで行ないます。
デバッグ制御インターフェースは、JTAG 接続、SWD 接続に対応、トレース機能は、SWV に対応します。

◎ETM モデル（トレースモデル）

ETM モデルは、JTAG／SWD に加えて、トレース機能である ETM を追加したモデルです。
Cortex M コアの CPU の場合、ターゲットボードとの接続は 20 ピン接続です。
（変換アダプタを使用しての接続となります。）
デバッグ制御インターフェースには、JTAG 接続、SWD 接続に対応、トレース機能は、ETM、SWV に対応します。
ETM モデルは、JTAG 機能にトレース機能が付加されたフルスペック機能のモデルです。

それぞれのモデルの対応機能は、以下の通りです。

	デバッグ制御インターフェース		トレース機能	
	JTAG	SWD	ETM	SWV (※)
JTAG モデル	○	○	×	○
ETM モデル	○	○	○	○

※SWV は、SWD 接続の場合にのみ使用可能

この表からもわかる通り、モデルの違いは、純粋に ETM トレース機能の有無です。
ですので、どちらのモデルを使用すればよいかは、ETM トレースを必要とされるかどうかになります。

なお、ETM モデルを使用されていて、評価ボード等で 20 ピンの JTAG コネクタしか実装されていない場合には、オプションの変換アダプタを使用することで JTAG モデルとして使用することが可能です。

2.2 デバッグ制御インターフェース

JTAG エミュレータは、CPU が内蔵するデバッグ機能を、デバッグ制御インターフェースを通して制御する事によって、デバッガの各機能を実現しています。この CPU と JTAG エミュレータを繋ぐデバッグ制御インターフェースが、JTAG であり、また、Cortex 世代から新たに実装された SWD です。
この2種類のデバッグ制御インターフェースがありますが、どのような違いがあって、どちらを選択すればよいのかという疑問があるかと思います。ここでは、それぞれのインターフェースについての解説します。

なお、JTAG エミュレータでは、デバッグ機能自体をそれぞれの CPU が持っているため、全ての CPU で同一の機能が提供されるのではなく、使用する CPU によって機能や仕様が若干異なります。

2.2.1 JTAG

JTAG とは、バウンダリスキャンテスト等の基板検査を行うための標準規格 IEEE1149.1 の通称名です。
規格を定めた業界団体 Joint European Test Action Group の名称の略から、JTAG と呼ばれています。
JTAG は、集積回路のピン間隔が狭くなったり、BGA などの表面実装が普及してきたりなど物理的にプローブを当てての検査が困難となってきたため、デバイス同士をデイジーチェーン接続にして、接続されたデバイスの状態を読み出す方式(バウンダリスキャンテスト)として規格化されました。
そして、この仕組をデバッガ制御に応用したエミュレータを総称して JTAG エミュレータ(JTAG ICE)と言います。元々の成り立ちが基板検査であるため、ARM コア CPU に限らず様々なアーキテクチャの CPU にて採用されている方式です。

CPU の制御には、基本的に TDI/TDO/TCK/TMS/TRST の 5 本の JTAG 信号線を使用して行ないます。

2.2.2 SWD (Serial Wire Debug)

SWD は ARM の Cortex 世代から採用された総合的なデバッグソリューションである CoreSight オンチップ・デバッグ／トレース・テクノロジーにて新たに採用された、デバッガの制御を双方向データ信号(SWDIO)とクロック(SWCLK)の 2 本で行なう事を可能になった ARM 独自の新インターフェースです。
JTAG に対して SWD を使用することでのメリットとデメリットを以下に示します。

・メリット

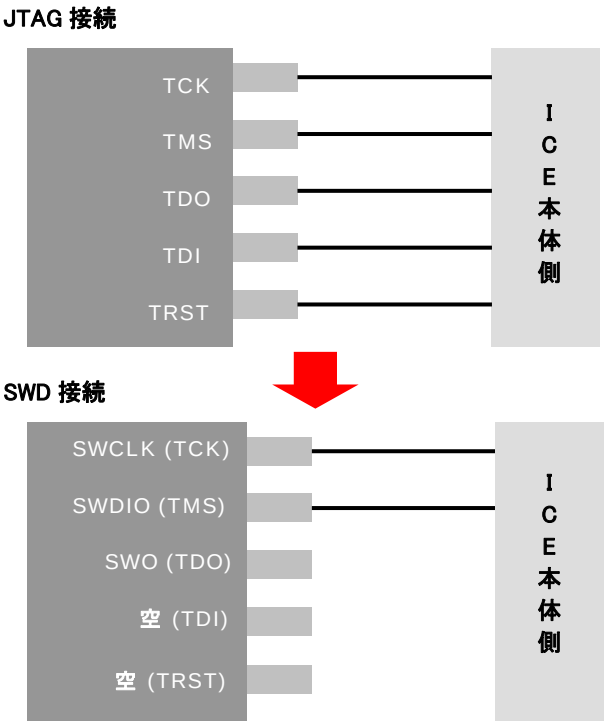
・端子数の減少
JTAG では、5 本の信号を使用して行っていた CPU のデバッグ制御を、SWD では 2 本の信号で行なうことが可能となります。端子数の制限が厳しい中、デバッグ時に JTAG エミュレータが占有する端子が少ないということは、今までデバッガに使用していた端子もユーザ側で使用できるようになり、大きなメリットとなります。

・コネクタの実装面積の減少
デバッガを接続するための端子数が減るため、ターゲットボードとの接続が、JTAG の 20 ピンでの接続の所が、10 ピンでも接続が可能となり、コネクタの実装面積が少なくすむこともメリットになります。

・SWV の使用
後述する SWV トレースは、JTAG 接続との組合せで使用することができません。SWD 接続の場合にのみ、SWV トレースを使用することができます。

・デメリット

・SWD は、デバッグ専用ポートですので、JTAG 本来の機能である検査ポートとしては使用できません。



SWD と JTAG は、ハードウェアから見れば制御方式が異なるためインターフェース部分が異なりますが、デバッガのソフトウェア上から見た場合、それぞれの接続方式によって、デバッガ上で使用できる機能の違いは全くなく、インターフェースの違いを認識される所はありません。また、信号数が減って、データ通信がシリアル通信となることで通信速度の低下の影響を懸念されるかもしれませんが、実際には CPU との通信以外の制御部分が処理の大半を占めるため、デバッガの動作速度の低下を体感することは、ほとんどなくお使い頂けます。

なお、SWD と JTAG は、兼用端子として機能を兼用している CPU の場合、JTAG 信号をインターフェースとして設計されているターゲットであれば、SWD 接続に切り替えて使用することも可能です。この場合、ハードウェア上にスイッチ等を設けていただく必要はなく、CSIDE 上の設定を変更するだけで、起動処理中に機能を切り替えて起動させることも可能です。
それぞれの端子が別れている CPU、どちら一方の機能しか実装されていない CPU もありますので、ご使用になる前に CPU の仕様をご確認ください。

2.2.3 CSIDE での設定方法

CSIDE は、初期設定では JTAG 接続で起動するようになっています。
SWD 接続でデバッガを起動させる場合には、以下の設定にチェックを入れます。

ターゲットシステムの設定ダイアログ［初期値の設定］-［SWD/SWV］-［シリアル・ワイヤ・デバッグ・モードを使用する］

この設定にチェックを入れるだけで、SWD 方式での動作となります。

2.3 どちらの方式を選べばよいのか？

SWD、JTAG というデバッグ制御インターフェースは、制御方法が JTAG 信号を使用するのか、SWD 信号を使用するのかの違いがあるだけで、CSIDE 上で実現できる機能や動作速度に違いはないため、どちらの方式を採用されても遜色ありません。
デバッガ専用インターフェースとしては、占有する端子数が少なく、かつ、JTAG に対してのデメリットが SWD 接続にはありませんので、使用する CPU が SWD インターフェースを搭載している場合には、今後は SWD インターフェースを採用されることをお薦めします。
ただ、バウンダリスキャンの検査ポートとしての使用できませんので、検査ポートとしての JTAG が必要となるのであれば、1 つのコネクタで兼用できる JTAG をご使用ください。

第3章 トレース機能

Cortex-M コアには、実行したプログラムの実行状況を解析する方法として、ETM、SWV というトレース機能が実装されています。本章では、このトレース機能について解説していきます。

3.1 ETM／SWV とは？

Cortex-M コアには、実行したプログラムの履歴を確認する方法として、ETM、SWV というトレース機能があります。
まずは、これらの機能の特徴を説明します。

ETM (Embedded Trace Macrocell)

ETM とは、CPU コア内部の実行命令の情報を専用のトレース・バスを介して外部にトレース情報として出力させるためのユニットです。実行するユーザプログラムに影響を与えることなく、プログラムの実行履歴やデータアクセスの情報などを取得することができます。
CPU の単なる実行／ブレークの繰り返しでは発見出来できない、システムがフルスピードで可動した際にハードウェアとソフトウェアの微妙なタイミングで起こるような不具合を調査する場合など、リアルタイム性が要求される組み込み開発環境において、正確な実行履歴を知ることができる非常に有用な機能です。
ただ、ETM は半導体メーカーごとのオプション機能であるため、ETM ユニットを搭載することで、出力用端子が必要となること、CPU 単価が上がること、ETM に対応した ICE が必要となることなどから、全ての CPU に搭載されていません。

SWV (Serial Wire Viewer)

SWV は、SWD と共に Cortex コア世代からの CoreSight オンチップ・デバッグ／トレース・テクノロジーにて採用されたトレース技術で、SWV も、プログラムに影響を及ぼすことなく、実行中の CPU コアにアクセスして、トレース情報を出力させます。SWV は、ETM に比べ、必要となる端子が少なく、ETM のように対応モデルの ICE も必要としないため、低コストで様々な CPU の情報を得ることの出来る機能です。
特に、ピン制限が厳しい少ピンタイプの CPU やチップ単価に敏感な CPU では、ETM を搭載されることが難しいため、このような分野では特に有効となるデバッグ機能です。

3.2 ETM と SWV の違い

ETM、SWV 共に、実行中のプログラムに影響を及ぼすことなく、トレース情報を出力するものですが、出力されるデータの特性に違いがあり、使用用途が異なります。本項では、ETM、SWV の違いについて解説を行ないます。

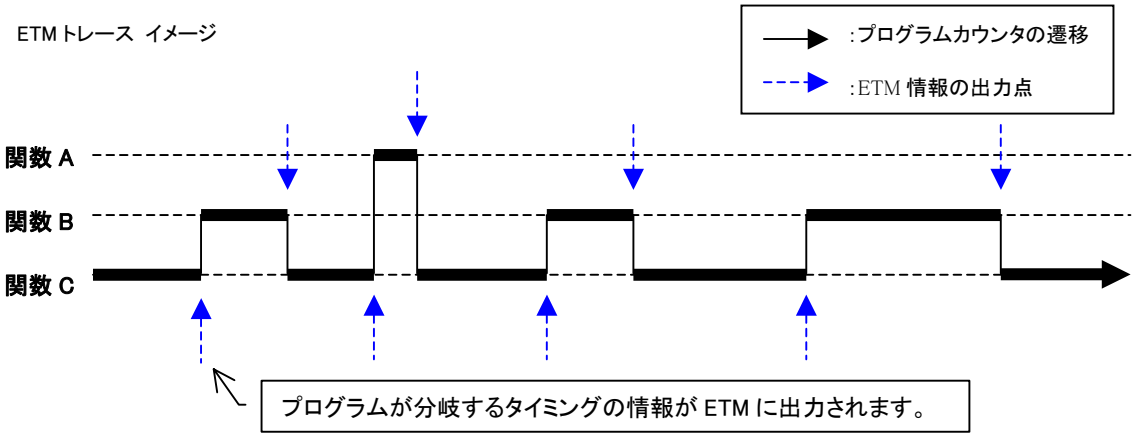
3.2.1 トレース内容の違い

ETM と SWV は、どちらも実行されたプログラムがどこを実行していたのか、履歴を知ることができますが、そのトレース内容には違いがあります。

ETM

ETM トレースは、実行されたプログラム中の分岐命令の情報がトレースデータとして出力され、その分岐情報を元に、デバッガが分岐間に実行された内容を補完して、実行履歴として表示を行ないます。分岐時の分岐元アドレスと分岐先アドレスが出力されますので、プログラムが実行された経路を正確に把握することができます。
また、プログラム中でのデータアクセスの履歴も合わせて出力されるため、データアクセスの状況も把握することが出来ます。

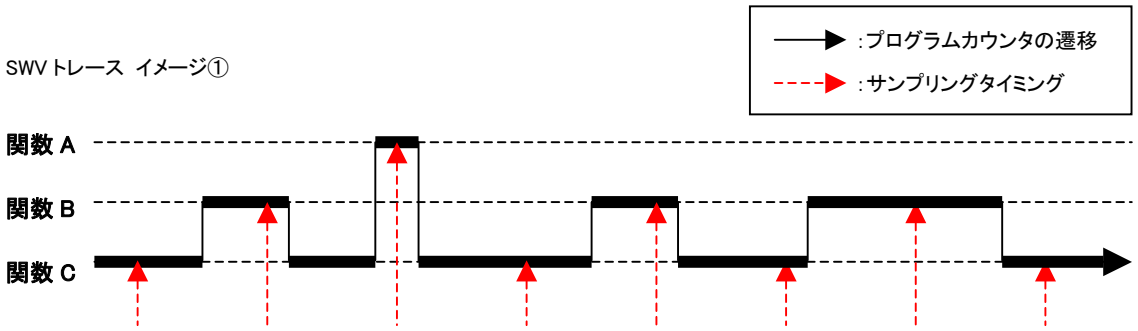
ETMトレースのイメージとしては、下図のようにプログラムが分岐する際の矢印の地点で、どこからどこへ分岐するかの情報が出力されるため、プログラムの動きを正確に知ることができます。



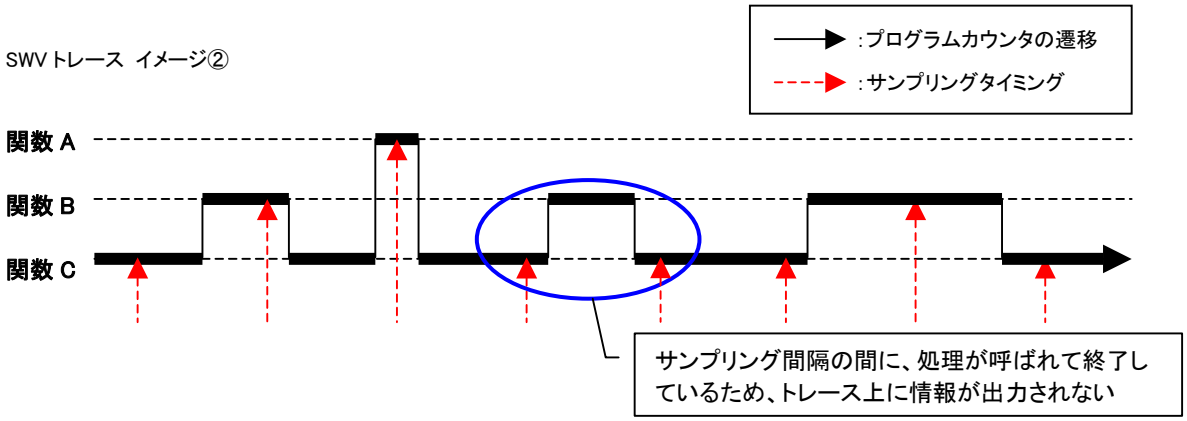
SWV

SWV トレースは、指定したサンプリング・サイクル間隔で指定したデータを取得(サンプリング)する機能です。
例えば、SWV でプログラムの実行の履歴を見る場合、プログラムカウンタの値を出力するように設定することで、設定したサンプリング・サイクル数ごとにその時点のプログラムカウンタの値がトレースとして出力されます。

SWV は、下のイメージ図のように、プログラムの動きに関係なく一定間隔でサンプリングして情報が出力されます。



この機能は、実行中の CPU の状態を定期的にサンプリングする機能になるため、サンプリングとサンプリングの間にプログラムがどこを実行しているのかわかりません。このためサンプリング間隔の間に、終了してしまう処理では、トレース上に情報が出力されないということになります。



この場合、サンプリング間隔を短くして補うということが考えられますが、サンプリング間隔を短くすることで取得するトレースデータが増加するため、トレースデータの転送が追いつかず、後述するトレースデータの欠落が頻繁に発生してしまい補うことはできません。
ただ、SWV はプログラムカウンタ以外にも、例外情報やデータアクセスの情報を出力させることも出来るため、様々なデータのサンプリングトレースとして使用することができます。

3.2.2 接続端子数

ETM

ETMトレースを取得するには、デバッグ制御インターフェースに加えて、ETM 用のクロック端子とデータ端子の接続が必要となります。データ端子の本数は、ARM コアごとに対応する出力ビット数が異なります。また、端子が他の機能と兼用となっていることが多く、ユーザターゲットにて ETM のデータ端子として使用できるかを予め確認しておく必要があります。

Cortex-M コアの場合、ETM は 1 ビット、2 ビット、4 ビットでの出力に対応しています。
最低限、クロック端子1本とデータ端子1本を接続すれば ETM トレースを取得することが出来ます。
接続するデータ端子の本数によって、ARMコアから出力される情報が変わるということはありませんので、1本でも取得は行なえます。ただし、接続本数が少ない場合、デバッガへのデータ転送のラインが少ないため、転送が間に合わずにデータがオーバーフローしてしまい、データが欠落する可能性が高くなります。
多くのデータ端子を接続していただくことで、データが欠落することを防ぐ事ができますので、ETM を使用される場合には、できるだけ多くのデータ端子を接続してのご使用をお薦めいたします。

SWV

SWV は、1 本の端子で取得することが可能です。
SWD インターフェースの端子と合わせて、合計 3 ピンで、デバッグ制御、SWV トレースの各機能を使用できるようになり、ユーザターゲットの端子を多数占有することなく様々な機能を使用することができます。

3.2.3 リアルタイム性

ETM

ETM トレースは、プログラム実行中に ARM コアから出力される ETM 情報を ICE 内のトレースメモリ上に貯めて、プログラムのブレーク時に表示が行われます。実行中にリアルタイムにトレース内容を確認することはできません。
一度に取得出来るトレース容量は、ICE 本体に搭載されているトレースメモリ容量分となります。

SWV

SWV トレースは、実行中のプログラムに影響を与えることなく、リアルタイムに取得、表示を行なうことができます。
また、SWV トレースは、実行中にリアルタイムにデータを ICE、ホスト PC へ送って表示を行なうため、ETM のように取得できる容量に限りはありません。(ただし、ホスト PC の性能によってデータの欠落が発生する可能性があります。)

3.2.4 トレースの取得条件

ETM

Cortex-M コアの ETM の仕様はアドレストレースのみとなっており、データトレースを取得することはできません。
指定した番地が実行されたらトレースの取得を開始することはできますが、特定範囲内にデータアクセスがあった場合にトレースを取得するという条件を指定することはできません。

SWV

SWV においてトレース条件を指定する場合、データアクセスが条件となります。
このため特定の番地が実行されたらトレースを取得することは特定のアドレスにアクセスがあった場合にトレースを取得するという形での条件設定となります。

3.2.5 タイムスタンプの正確性

ETM

ETM では、CPU から出力される情報の中に、タイムスタンプは含まれていないため、ICE 側で ETM 情報を取得したタイミングでタイムスタンプを付加して表示を行っています。このためトレースの開始点から終了点までのトータルの時間については、正確な値となりますが、表示される各命令のタイムスタンプには誤差が含まれるため注意が必要です。

SWV

SWV の情報の中に情報を取得した際のタイムスタンプを付加することができるため、トレース上に表示されるタイムスタンプは、実際に CPU から出力された正確な値になっています。このため、データアクセスの検出の間隔を計測される場合などには、非常に有効な手段となります。

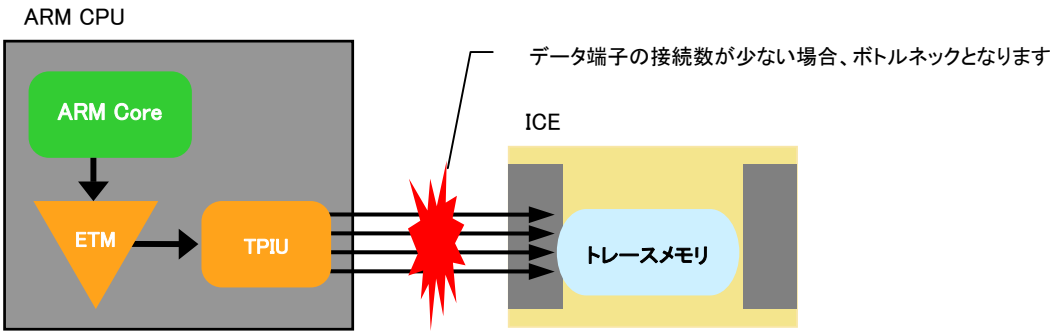
3.2.6 トレースデータの欠落

SWV、ETM は共に、トレースデータの出力時にデータの欠落が発生することがあります。
ターゲット上の CPU の動作速度に対して、CPU と ICE、ICE とホスト PC の通信速度は相対的に遅いため、トレースデータが集中して出力される状況となると、ICE 側へのデータ転送が間に合わなくなり、トレースデータの欠落が発生します。

ETM

ETM は、プログラムの実行中に ETM を通じてトレース情報が出力され、ICE 内のトレースメモリにデータを転送しています。
ETM は、SWV に比べて転送速度が速く、また、最大 4bit のデータ端子を使用してデータ転送を行なえるため、SWV と比較するとデータの欠落は発生しにくいのですが、間接分岐を繰り返されるなどして出力する ETM トレースの情報量が集中した場合には、トレースメモリへのデータ転送が間に合わなくなり、トレースデータの欠落が発生します。
(間接分岐は、分岐先アドレスの情報も ETM に出力されるため、これを多重に繰り返すことで情報量が増えます。)

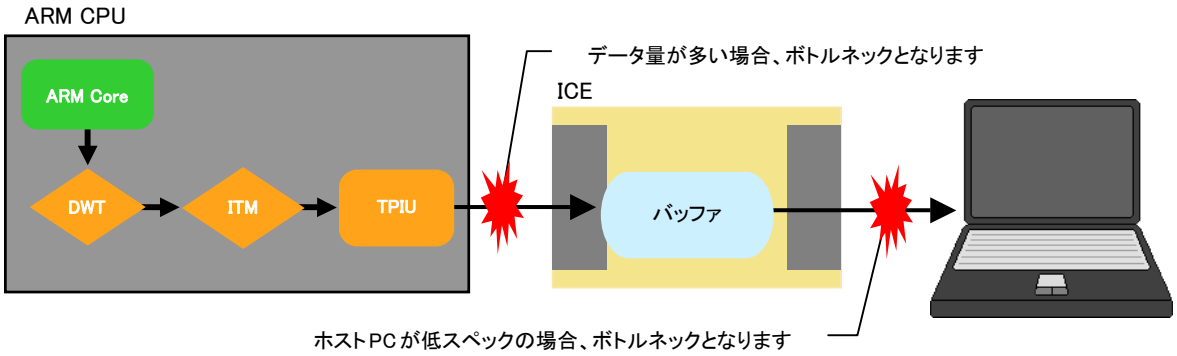
また、接続されている ETM のデータ端子が少ない場合、時間当りの転送量が減るため、ボトルネックとなりデータの欠落が発生しやすくなります。このため ETM データ端子は、できるだけ多くの端子を接続されることをお薦めいたします。



SWV

SWV は、CPU から ICE 本体へのトレースデータの転送を、リアルタイムに1本の信号線のシリアル通信で行い、また、通信速度も遅いため、ETM トレースと比較してデータの欠落が発生しやすくなります。
また、SWV ではトレースデータの ICE からホスト PC へのデータ転送もリアルタイムに行なうため、ホスト PC の性能が低い場合には、こちらもボトルネックとなり、データの欠落が発生する要因になります。

ただ、SWV は、出力させるトレースデータの内容の選択やサンプリング間隔を設定することができます。もし、SWV を使用して頻繁にトレースデータの欠落が発生する場合には、一度に複数の情報を出力させないようにしたり、トリガー条件を指定し必要な情報だけを取得するようにしたり、データ取得のサンプリング間隔を空けるようにしたりなど、出力されるデータ量が減少するよう工夫して使用することで、データの欠落をある程度避ける事が出来ます。



3.2.7 まとめ

これまでの ETM と SWV のそれぞれの違いをまとめると以下の表となります。

	ETM	SWV
①トレース	実行履歴	サンプリング
②接続端子数	JTAG or SWD 接続に加えて +2 ピン or3 ピン or5 ピンで取得可能	SWD 接続に加えて +1ピンで取得可能
③表示	ブレーク時のみ表示可能	リアルタイムに表示可能
④トレース取得条件	アドレス条件	データアクセス
⑤タイムスタンプ	若干の誤差が含まれる	CPU の正確な値が取得可能
⑥トレースデータの欠落	少ない	多い
⑦その他機能	-	例外トレース、printf 機能など

これまでの説明の中で、ETM と SWV というは、どちらも Cortex-M コアのトレースソリューションなのでプログラムの実行履歴を示すものではありますが、デバッグ時の使用目的が全く異なる機能であることをご理解いただけたのではないかと思います。

ETM は、プログラム実行履歴を正確に確認することができるため、実行履歴からプログラムの不具合箇所を特定する用途に適していますし、SWV は、実行中のプログラムがおおよそ正常に動作しているかをリアルタイムに確認したり、例外の発生間隔やデータアクセスの発生間隔などプログラムの測定等の用途に適しています。

これらの機能の特性をご理解いただき、用途に応じて使い分け、効率よくデバッグを行っていただければと思います。

3.3 デバッグ制御インターフェースとトレース機能の組合せ

CSIDE にて、デバッグ制御インターフェースとトレース機能の使用可能な組合せを以下に示します。

JTAG	+	ETM	+		→	○
JTAG	+	SWV	+		→	×
JTAG	+	SWV	+	ETM	→	×
SWD	+	ETM	+		→	○
SWD	+	SWV	+		→	○
SWD	+	SWV	+	ETM	→	○

（但し、ETM と SWV は切り替えての使用になります。同時に使用出来ません）

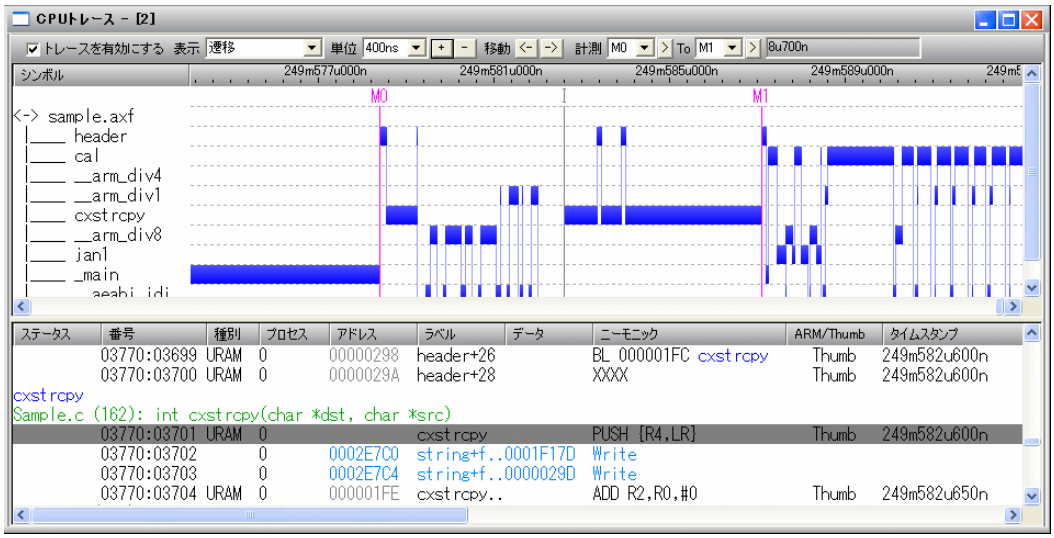
3.4 ETM／SWV を使用した CSIDE の機能

ETM/SWV を使用することで、CSDIE 上で以下のデバッグ機能を使用することが可能となります。

ETM

・CPU トレース機能

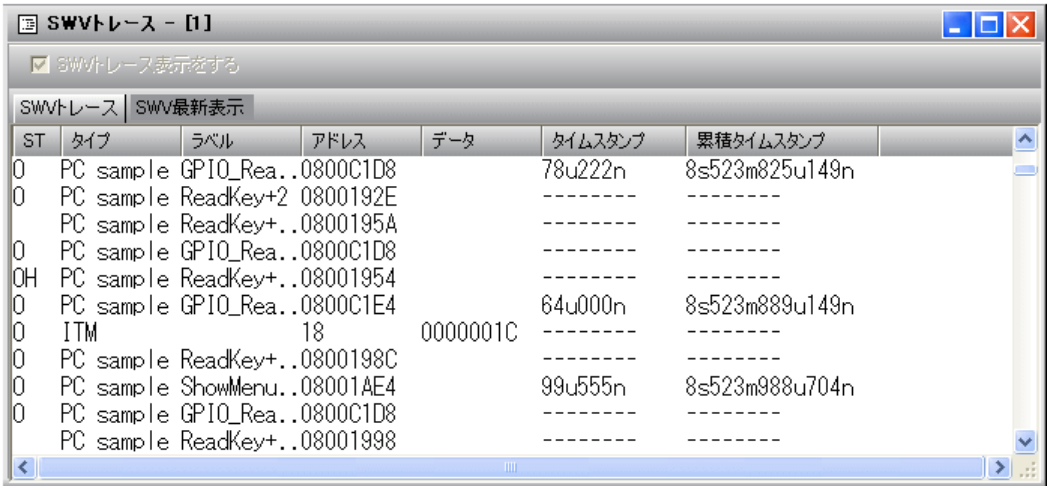
取得した ETM トレースを表示します。
関数の遷移図と逆アセンブルコードが表示されます。遷移状況を確認しつつ、下の逆アセンブル表示にして、問題点を調べることが出来ます。



SWV

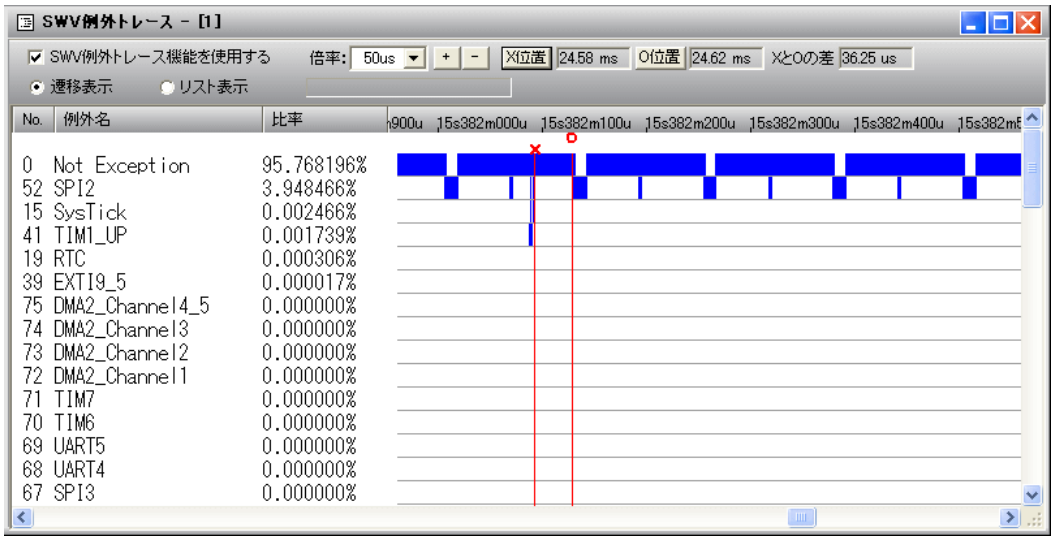
・SWV トレース

CPU から出力されたトレース情報を履歴として確認することができるウィンドウです。
条件指定により、プログラムカウンタのサンプリング、周辺レジスタなど特定のデータの観察、CPU のアクセスデータ、例外情報など様々な情報を得ることが出来ます。



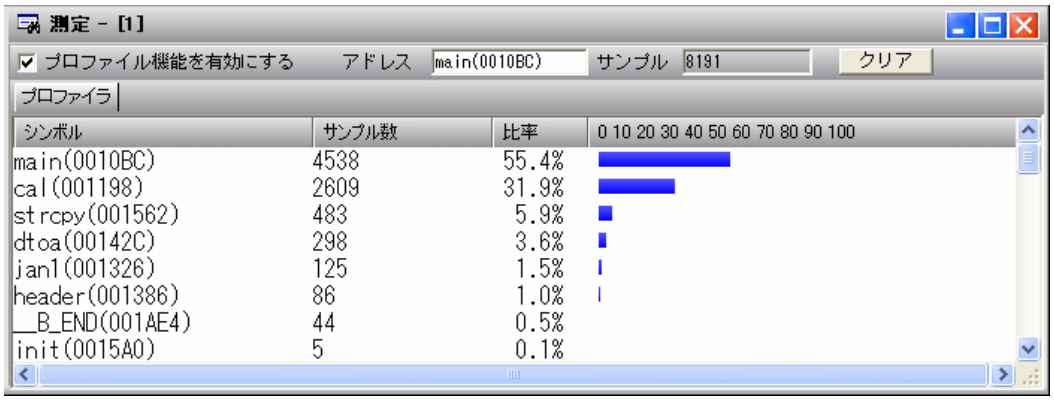
・SWV 例外トレース

SWV トレースを使用して得た例外の遷移状態や発生状況等の情報を、専用のウィンドウで確認することができます。
また、タイムスタンプの取得設定を行なうことで、例外の処理時間を表示させることも出来ます。



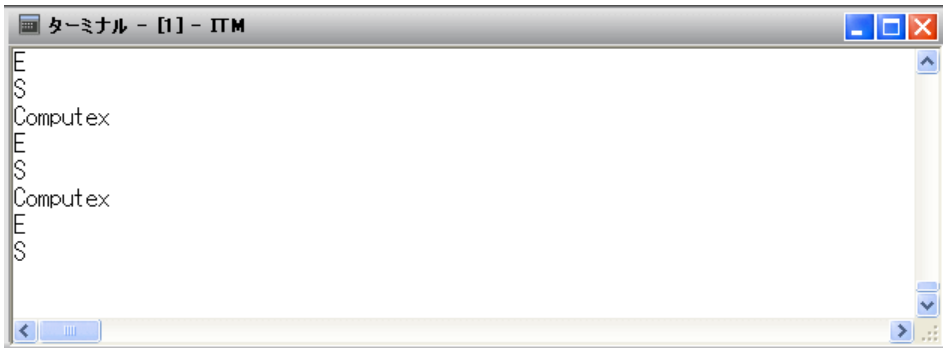
・プロファイラ

プログラムカウンタのサンプリング情報を元に、モジュールごとの実行比率をグラフ表示します。
使用頻度の高い関数、時間占有率の高い関数を調査するのに便利な機能です。



・Printf デバッグ機能

SWV の ITM(計装トレース・マクロセル)を使用して、プログラム中から任意の文字列、数値を CSIDE 上のターミナルウィンドウ上に表示させることができます。



※CSIDE の RTOS デバッグ拡張機能である RTOS タスクトレース機能は、ETM、SWV を使用した機能ではありません

Computex

株式会社コンピューテックス

本 社
〒605-0846 京都市東山区五条橋東四丁目 432-13 對嵐坊ビル
TEL.075(551)0528(代表) FAX.075(551)2585

営業部
〒101-0047 東京都千代田区内神田二丁目 15-2 内神田 DNK ビル 7F
TEL.03(3253)2901(代表) FAX.03(3253)2902

テクニカルセンタ
TEL.075(551)0373 FAX.075(551)2585

Cortex-M コア対応エミュレータ デバッグ制御インターフェースとトレース機能
2013 年 03 月 第 2 版 CS0015(B)1303