



インテル® System Studio による IoT プラットフォーム上でのシステムの 起動と組み込みソフトウェアのデバッグ

2017 年 9 月

ソフトウェア & サービスグループ
開発製品部門

林 侃

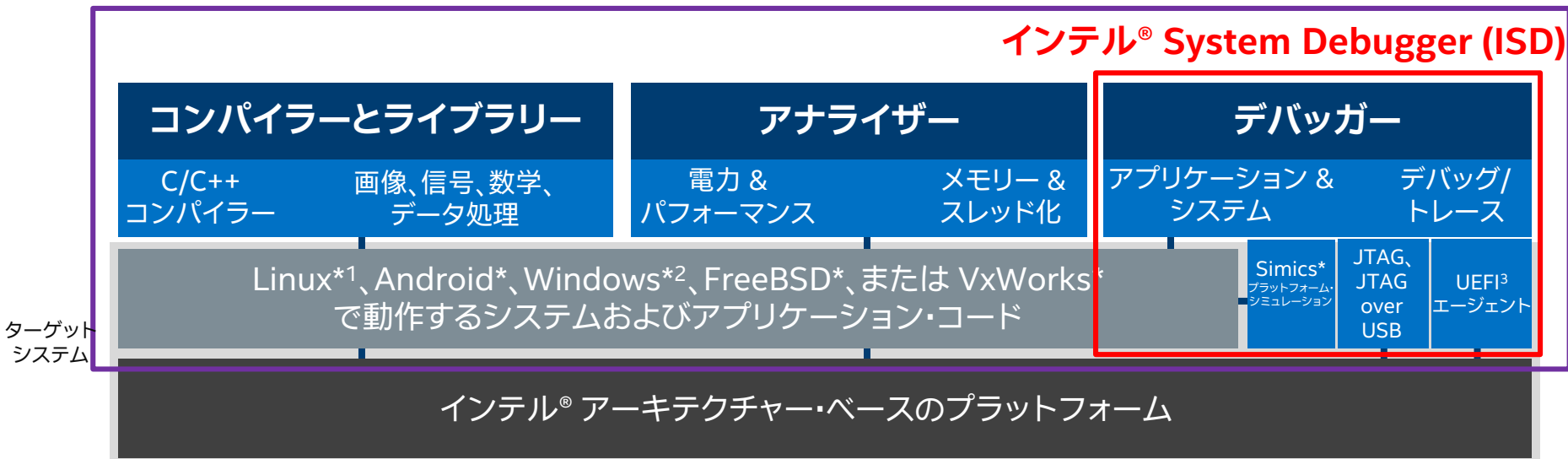
内容

- 1) インテル® System Debugger (ISD) 2017 の概要
- 2) デバッグに利用可能なプローブ
- 3) システムデバッグ (XDB)
- 4) システムトレース
- 5) WinDbg 拡張
- 6) eGDB
- 7) まとめ

インテル® System Debugger (ISD) 2017 の概要

インテル® System Studio 2017

インテル® System Debugger (ISD)



¹ Linux*, 組み込み Linux*, Wind River* Linux*, Yocto Project*

² Windows*: WinDbg 拡張は Windows* ターゲットでのみ動作

³ UEFI: Unified Extensible Firmware Interface

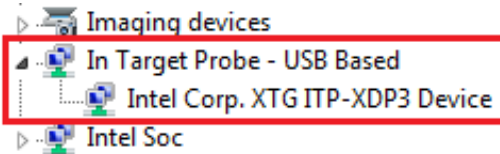
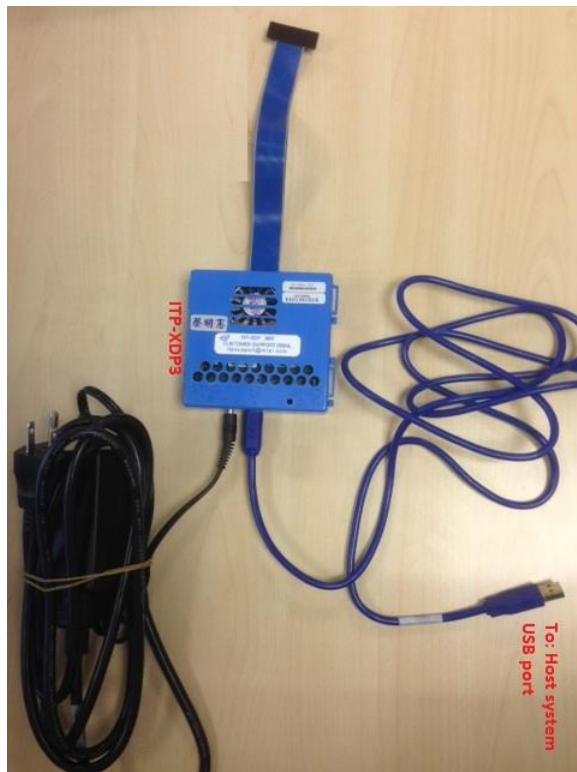
インテル® System Debugger (ISD) 2017 の概要

- インテル® マネジメント・エンジン (ME)/BIOS コードのトレース、SoC レジスターの読み取り/書き込み、システムログの取得を行う **デバッグツール** を含む
 1. システムデバッグ (XDB)
 2. システムトレース
 3. WinDbg 拡張
 4. eGDB
- サポートするホストからターゲット・プラットフォームへの接続 **プローブ**
 1. ITP-XDP3
 2. CCA (Closed Chassis Adapter)
 3. DbC (DCI USB 3.0 Debug Class)

デバッグに利用可能なプローブ

- ITP-XDP3

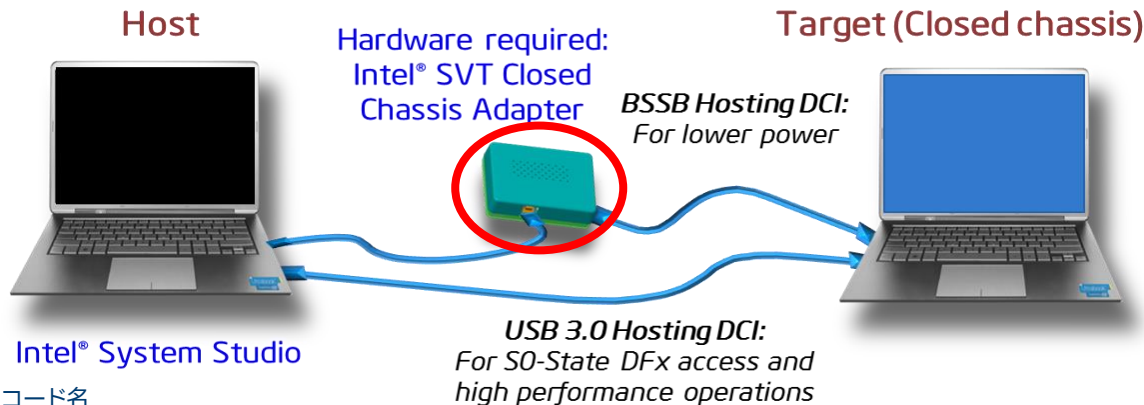
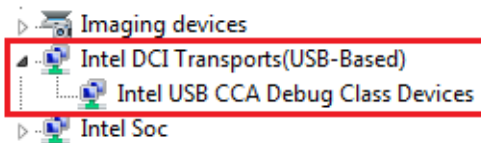
- 価格: お問い合わせください
- インターフェイス: XDP
- ターゲット:
 - 開発コード名 BayTrail (BYT)
 - 開発コード名 CherryTrail (CHT)
 - 開発コード名 Haswell (HSW)
 - 開発コード名 Broadwell(BDW)
 - 開発コード名 Skylake (SKY)
 - 開発コード名 Skylake Server
 - ほか



デバッグに利用可能なプローブ

- インテル® SVT Closed Chassis Adapter

- 価格:お問い合わせください
- インターフェイス: DCI over USB 3.0
- ターゲット:
 - インテル® Core™ プロセッサ・ベースのプラットフォーム (Skylake+ 以降、Kaby Lake+ などを含む)
 - インテル® Xeon® プロセッサ・ベースのプラットフォーム (Skylake Server+ 以降)



† 開発コード名



最適化に関する注意事項

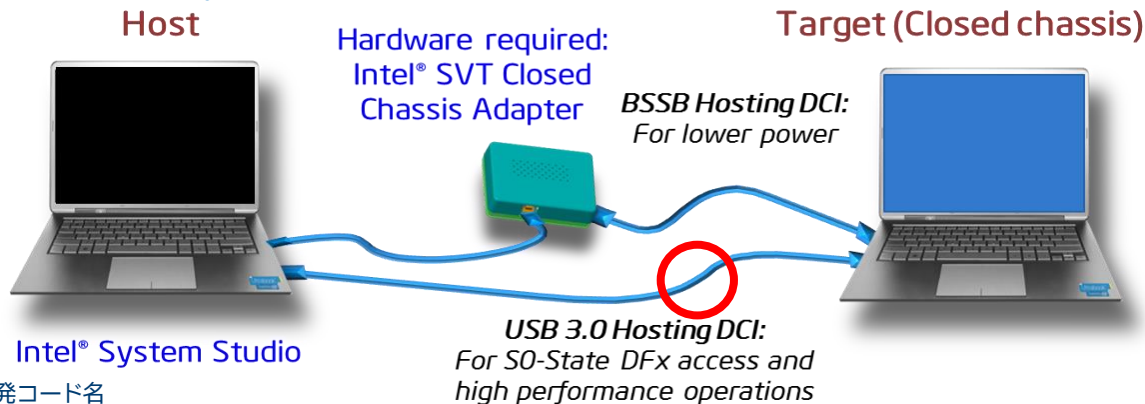
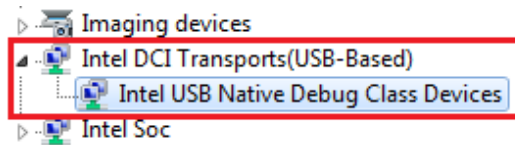
© 2017 Intel Corporation. 無断での引用、転載を禁じます。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

デバッグに利用可能なプローブ

- DbC (DCI USB 3.0 Debug Class)

- 価格:お問い合わせください
- インターフェイス: DCI over USB 3.0
- ターゲット:
 - Intel Atom® プロセッサ・ベースのプラットフォーム (Apollo Lake⁺ 以降)
 - インテル® Core™ プロセッサ・ベースのプラットフォーム (Skylake⁺ 以降)
 - インテル® Xeon® プロセッサ・ベースのプラットフォーム (Skylake Server⁺ 以降)



† 開発コード名

最適化に関する注意事項

© 2017 Intel Corporation. 無断での引用、転載を禁じます。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

システムデバッグ (XDB)

• 主な機能

- 接続ルートを確立: XDB – プローブ – プラットフォーム
- 実行制御
- SMM (システム・マネジメント・モード) デバッグ
- 現在の位置、PEI、および DXE モジュールの EFI シンボル (ソースコード) のロード
- PCI スキャン
- カーネルモジュールのデバッグ
- プロセッサ命令のトレース

システムデバッグ (XDB)

- 現在の位置、PEI、および DXE の EFI シンボル (ソースコード) のロード

アセンブラビュー

ソースビュー

The screenshot displays the Intel(R) System Debugger interface. The left pane shows the 'Assembler' view for the address range 0x0038:0x0000000078AFC80A to 0x0038:0x0000000078AFC97A. The right pane shows the 'Source' view for the file debuglib.c, with line 149 highlighted.

Trail	Address	Opcodes	Source
	0x0038:0x0000000078AFC80A	49 FF C0	inc r8
	0x0038:0x0000000078AFC80D	48 85 C9	test rcx, rcx
	0x0038:0x0000000078AFC810	75 E2	jnz 0x78AFC7F4 <DebugAssert(ch...
	0x0038:0x0000000078AFC812	48 C7 84 24 58 01 ...	mov qword ptr [rsp+0x158], 0xE...
	0x0038:0x0000000078AFC81E	48 8B 84 24 58 01 ...	mov rax, qword ptr [rsp+0x158]
	0x0038:0x0000000078AFC826	48 85 C0	test rax, rax
	0x0038:0x0000000078AFC829	74 F3	jz 0x78AFC81E <DebugAssert(ch...
	0x0038:0x0000000078AFC82B	48 81 C4 38 01 00 00	add rsp, 0x138
	0x0038:0x0000000078AFC832	C3	ret
	0x0038:0x0000000078AFC833	CC	int3
	0x0038:0x0000000078AFC834	40 53	push rbx
	0x0038:0x0000000078AFC836	48 83 EC 20	sub rsp, 0x20
	0x0038:0x0000000078AFC83A	48 8B 05 37 44 00 00	mov rax, qword ptr [rip+0x4437...

```
139 // Send the print string to the Console Output device
140 //
141 SerialPortWrite ((UINT8 *)Buffer, AsciiStrLen (Buffer));
142
143 //
144 // Generate a Breakpoint, DeadLoop, or NOP based on PCD settings
145 //
146 if ((PcdGet8(PcdDebugPropertyMask) & DEBUG_PROPERTY_ASSERT_BREAKPOINT_ENABLED) != 0)
147     CpuBreakpoint ();
148 } else if ((PcdGet8(PcdDebugPropertyMask) & DEBUG_PROPERTY_ASSERT_DEADLOOP_ENABLED) != 0)
149     CpuDeadLoop ();
150 }
151 }
```

最適化に関する注意事項

© 2017 Intel Corporation. 無断での引用、転載を禁じます。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

システムデバッグ (XDB)

- SW/HW ブレークポイントの設定 (シンボルを使用しない場合)

The screenshot displays the Intel XDB interface with several callouts explaining the steps to set a breakpoint:

- ステップ 1: [Breakpoints] ウィンドウをクリック**: Points to the Breakpoints icon in the top toolbar.
- ステップ 2: アセンブラーまたはソースコードをダブルクリック**: Points to a line of assembly code in the Assembler window.
- ステップ 3: [Breakpoints] ウィンドウにブレークポイントが表示される**: Points to the newly added breakpoint in the Breakpoints window.

Trail	Address	Opcodes	Source
	0x0010:0xFFFE3F5	CC	int3
	0x0010:0xFFFE3F6	CC	int3
	0x0010:0xFFFE3F7	CC	int3
	0x0010:0xFFFE3F8	56	push esi
	0x0010:0xFFFE3F9	57	push edi
	0x0010:0xFFFE410	FA	mov esi, dword ptr [esp+0x10]
	0x0010:0xFFFE411	FE	mov edi, dword ptr [edi+0xC]
	0x0010:0xFFFE412	B2	mov edx, dword ptr [edx+14]
	0x0010:0xFFFE413	96	lea eax, ptr [edx+esi]
	0x0010:0xFFFE414	9A	cmp esi, edi
	0x0010:0xFFFE415	9C	jnb 0xFFFFDE412 <>
	0x0010:0xFFFE416	9E	cmp eax, edi
	0x0010:0xFFFE417	9C	jnb 0xFFFFDE41E <>
	0x0010:0xFFFE418	9E	mov ecx, edx
	0x0010:0xFFFE419	83	and ecx, 0x3
	0x0010:0xFFFE41A	C1	shr ecx, 0x2
	0x0010:0xFFFE41B	F3	rep movsd dword ptr [edi], dword ptr [eax]
	0x0010:0xFFFE41C	E8	jmp 0xFFFFDE425 <>
	0x0010:0xFFFE41D	8B	mov esi, eax
	0x0010:0xFFFE41E	80	lea edi, ptr [edx+edi*1-0x1]
	0x0010:0xFFFE41F	FD	std

Id	Address	Function	File
0	0x0010:0xFFFE417		

最適化に関する注意事項

© 2017 Intel Corporation. 無断での引用、転載を禁じます。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

システムデバッグ (XDB)

- SW/HW ブレークポイントの設定 (シンボルを使用する場合)

ステップ 1: [Breakpoints] ウィンドウでダブルクリック

ステップ 2: シンボルをロード後に関数名を検索

[Hard] オプションをオンにするとアクセスできないメモリー位置にもブレークポイントを設定可能

Symbol Browser Search Results:

Symbol Name	Address	Function	File	Skip Count
PchBaseInit(void)	0xFFFFDC84C	PchBaseInit(void)	pplatform.c:499	0

最適化に関する注意事項

© 2017 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

システムトレース

• 主な機能

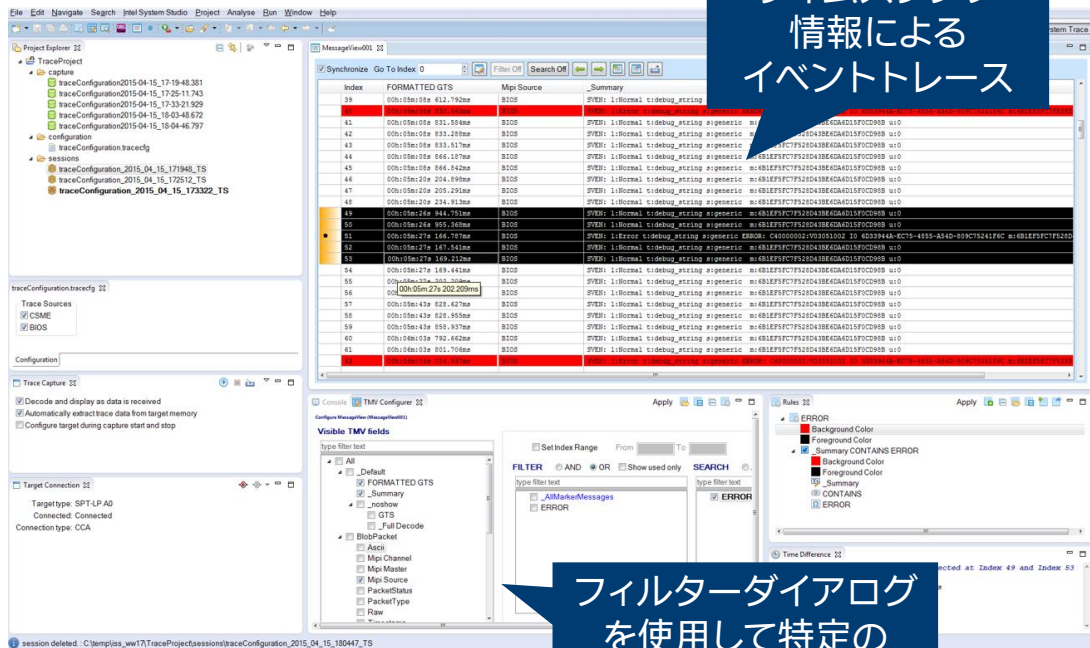
- 接続ルートを確立: システムトレース – プロブ – プラットフォーム
- システム・トレース・データをキャプチャー
- トレースデータのキーワードをフィルター/ハイライト表示
- ログをマーク
- イベント分布

複雑なシステムの問題を迅速に切り分け

総合的なシステムワイドのハードウェア/ソフトウェア・イベントトレース

インテル® System Debugger

- タイムスタンプ付きの関連付けされたトレース情報を使用して問題を効率良く特定
- ソフトウェアとハードウェア間の複雑な相互作用を解析



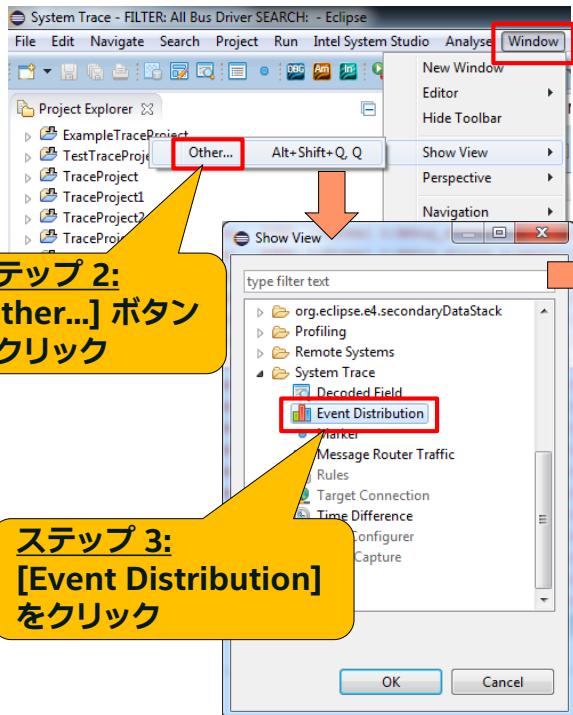
第 6 世代インテル® Core™ プロセッサ・ファミリー以降で利用可能

最適化に関する注意事項

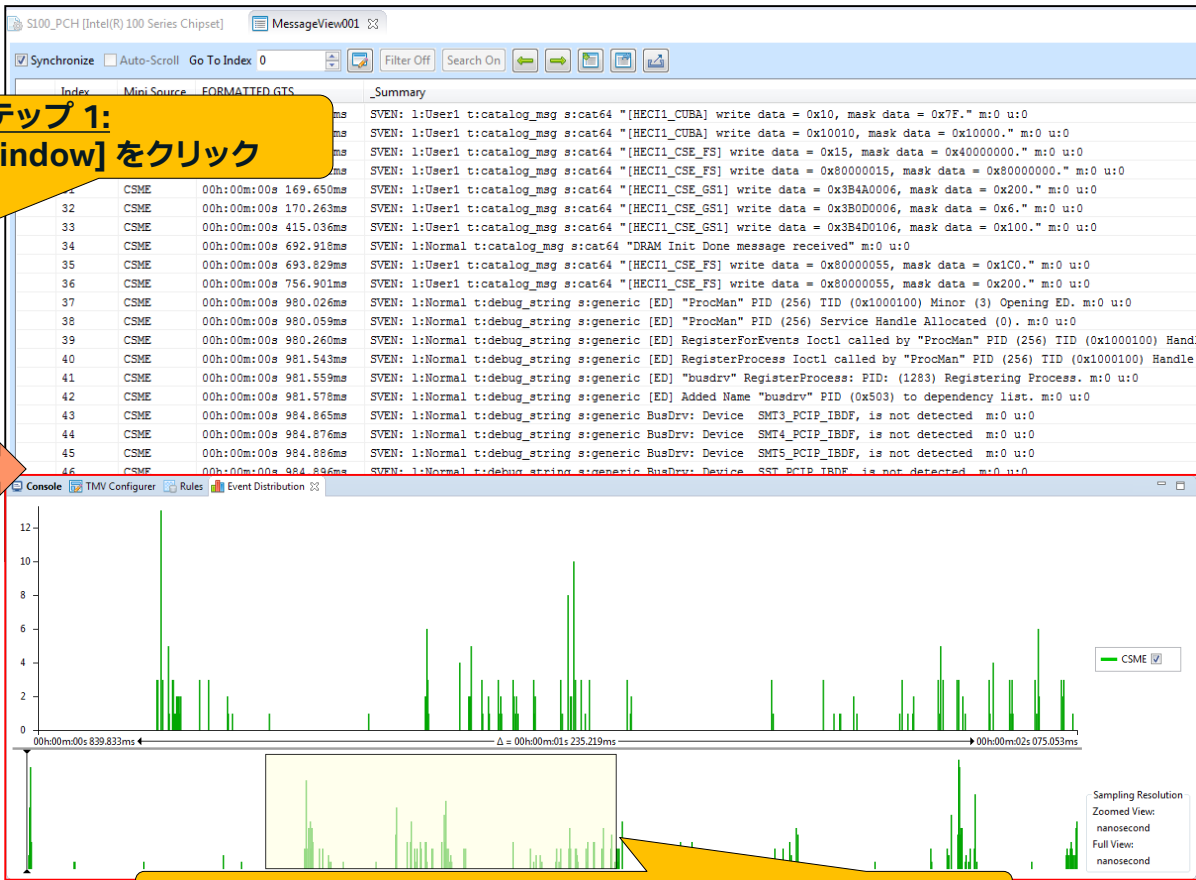
© 2017 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

システムトレース

・ イベント分布



ステップ 1:
[Window] をクリック



最適化に関する注意事項

© 2017 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

WinDbg 拡張

- 概要

- インテル® Debug Extension for WinDbg は、WinDbg 向けに JTAG ベースのトランスポート層を提供
- 初期ブートやハイバネーションのデバッグなどの追加のユースケースに対応できるように WinDbg を拡張

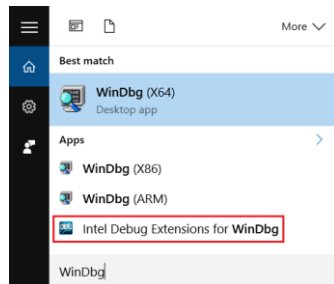
- 使用シナリオ

- … 初期ブートとウェイクアップ・フェーズ
- … システム回復、Windows* インストーラー



WinDbg 拡張

- スタートメニューからインテル® Debug Extensions for WinDbg を開始



- 設定を選択 - 例:

```
Please choose target architecture:
[0] - x64
[1] - x86
0
Please choose connection layer: Intel(R) DAL (SKL, KBL) or OpenIPC (VLV2, BXTP, DNV)
[0] - c:\Intel\DAL
[1] - c:\Intel\OpenIPC\Bin
0
```

- 次のコマンドで WinDbg セッションを初期化
`wdbg = itpkd.WindbgShell()`
- 接続後に WinDbg を開始
`wdbg.run()`

WinDbg 拡張

現在の位置のアセンブリ・コード

編集可能なノート - いくつかの役立つ KD コマンドが追加されている

カーネルデバッグ (KD) の出力コンソール

カーネルデバッグ (KD) のコマンド入力

The screenshot shows the WinDbg interface with several windows:

- Disassembly Window:** Shows assembly code for kernel mode. A red box highlights the instruction `movzx eax, dword ptr [eax]` at address `fffff8001a72f994`. The command line below indicates the kernel debugger path: `Command - eXDI \exdiid:\{66664C88-5320-4686-837D-7D17679728E6}\Kd-VerAddr:18446735293415284160>DataBreaks=Exdi - WinDbg:10.0.10586.567`.
- Registers Window:** Shows the current register values, including `eax` and `ecx`.
- Kernel Debugging Output Window:** Displays the output of the kernel debugger, including system information like `fffff8003e42d440` and `fffff803a3dbab40`.
- Scratch Pad Window:** Contains a list of useful KD commands such as `!process`, `!thread`, `!analyze`, `!dump`, and `!help`.
- Registers - eXDI Window:** Shows a table of registers and their values, including `eax`, `ecx`, `edx`, `ebx`, `esp`, `ebp`, `esi`, `edi`, `eip`, `cr2`, and `cr4`.
- Memory Window:** Shows the current memory address and its contents.

最適化に関する注意事項

© 2017 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

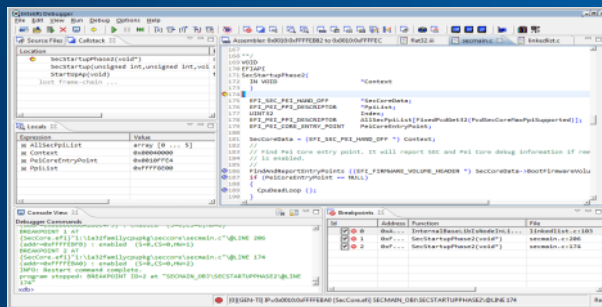


デバッグとトレース

GDB アプリケーション・デバッガ

- インテル® Processor Trace 向けの GNU* GDB サポート

インテル® System Debugger



インテル® Processor Trace

トレースを実行する新しいハードウェア機能

- 簡潔なトレース形式
- 低オーバーヘッドのトレース収集
- タイムスタンプを含むトレース

GDB 使用モデル:

- クラッシュ: なぜクラッシュしたのか?

その他の使用モデル:

- サンプリング: サンプル周辺のコンテキストを提供
- プロファイル: 基本ブロックレベルのタイミング
- 応答性: 時間 X 頃にアプリケーションが何をしていたか?

まとめ

- インテル® System Studio のインテル® System Debugger は以下の場合に利用可能
 - IoT システムの起動、評価、システム統合
 - 組み込みソフトウェアのデバッグ、クローズドシャーシのデバッグ
- プロセッサ・トレースのサポート
- システムトレースのサポート
- 関連情報:
 - <https://www.isus.jp/intel-system-studio/>

法務上の注意書きと最適化に関する注意事項

本資料の情報は、現状のまま提供され、本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他、知的財産権の侵害への保証を含む) をするものではありません。

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサ用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴ、Intel. Experience What's Inside、Intel. Experience What's Inside ロゴ、Intel Atom、Intel Core、Xeon は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

最適化に関する注意事項

インテル® コンパイラーでは、インテル® マイクロプロセッサに限定されない最適化に関して、他社製マイクロプロセッサ用に同等の最適化を行えないことがあります。これには、インテル® ストリーミング SIMD 拡張命令 2、インテル® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサ依存の最適化は、インテル® マイクロプロセッサでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサ用のものがあります。この注意事項で言及した命令セットの詳細については、該当する製品のユーザー・リファレンス・ガイドを参照してください。

注意事項の改訂 #20110804

