お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

H8/300L SLP シリーズ

開発ツール/ハードウェア開発ツールの効果的な使用方法 (UseDevTool)

要旨

このアプリケーションノートの目的は，ユーザが，ハードウェア開発ツールを正しく選択できるようにすることです。

このアプリケーションノートでは，各種ハードウェアツールを分類して，それぞれの長所と短所を紹介します。ユーザにより深く理解していただくために，ツールの適用についても説明します。

紹介する各種ツールを以下に示します。

- CPU ボード，スタータキット
- ROM エミュレータ
- JTAG エミュレータ
- フルスペック ICE (インサーキットエミュレータ)
- シミュレータ

紹介する機能を以下に示します。

- ブレークポイント
- トレース
- より高度な機能

このアプリケーションノートではハードウェアを中心に説明しますが，説明をわかりやすくするために，ソフトウェアの開発支援ツールの特長についても簡単に紹介します。

動作対象デバイス

すべての H8/300L SLP MCU

目次

1. 概要

開発ツールの目的は，MCU の動作を理解し，必要な制御やモニタリングの手段を提供することです。開発ツールを使用し, ユーザは応用システムを設計しますが, 費やす時間のほとんどはバグの修正です。したがって，一般的に，デバッガまたはデバッグ用ツールと呼ばれます。

ユーザは開発ツールを使用するともどかしく感じるかもしれません。エミュレータに多額の費用を費やし，ターゲットボードに接続するのに何ヶ月もの時間を要するからです。さらに，最悪なことは，十分なエミュレーションをするために何ヶ月もコーディングして, そのプログラムを OTP (One Time PROM) にダウンロードしても機能的なアプリケーションが保証されないかもしれません。または，JTAG エミュレータなどを使用して，制限事項に気づくかもしれません。

一方では，開発ツールの設計者は，ユーザのシステムのトラブルシューティングにサポート業務の 50%を費やします。ほかの 40%の時間は，開発ツールの使い方を間違えたユーザへの対応に追われます。

したがって，これらの問題を回避する必要があります。ユーザは開発を始める前に各開発ツールの長所と短所を理解し，デバッグ戦略を計画しなければいけません。

このアプリケーションノートは, ユーザが各種開発ツール, 適用方法, 機能を理解するのに役に立ちます。

各種ハードウェア開発ツールは主に 3 つに分類できます。

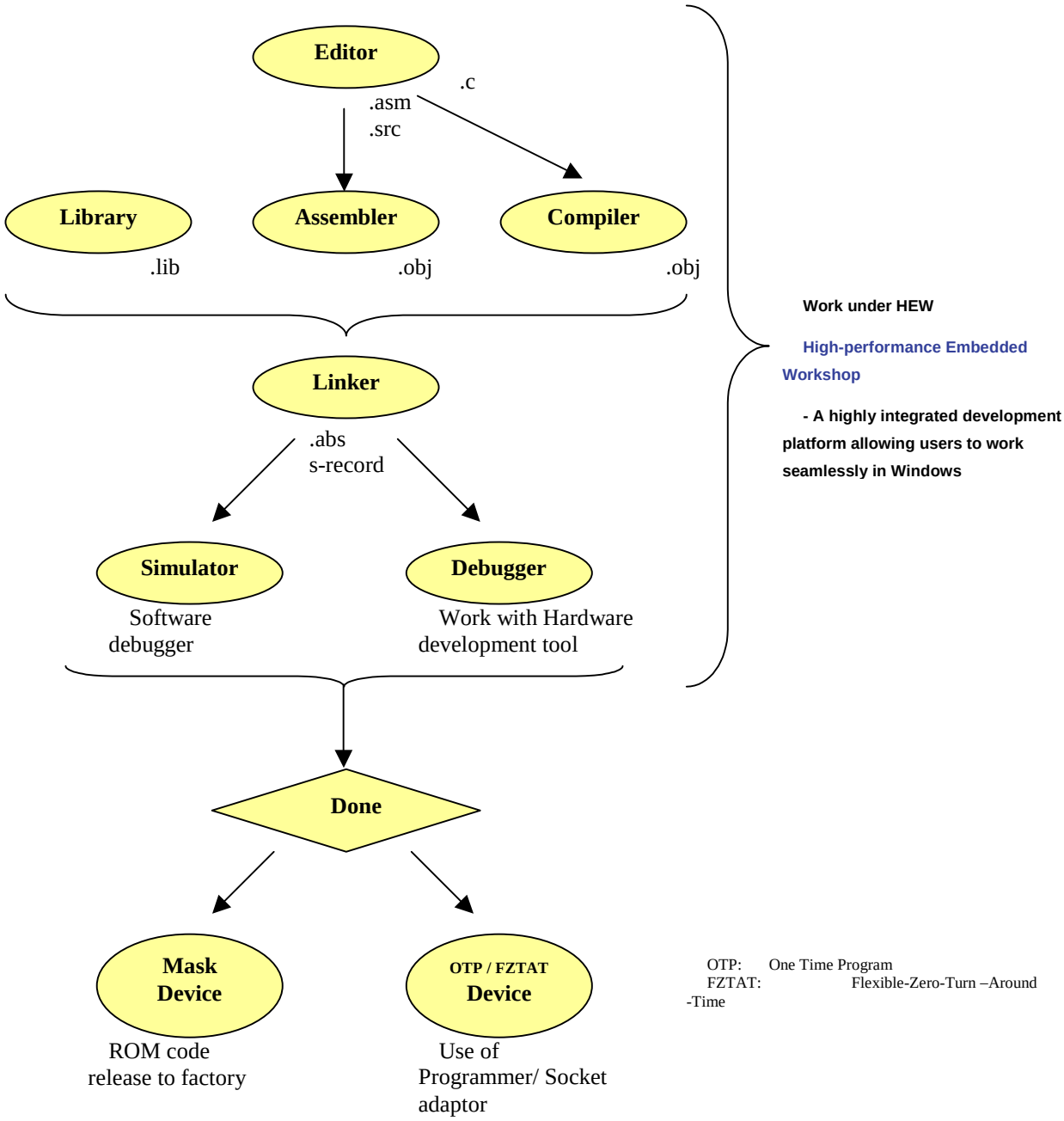
各分類では，以下の項目について説明します。

- 概要
- 特性
- 適用
- 注意点
- 使用方法
- 長所と短所

後半の章では, 3 つの範囲のツールにおけるエミュレーション機能 (ステップ, ブレーク, トレースなど) の特性の詳細を説明します。

2. 開発フロー

以下に開発フローの概要を示します。



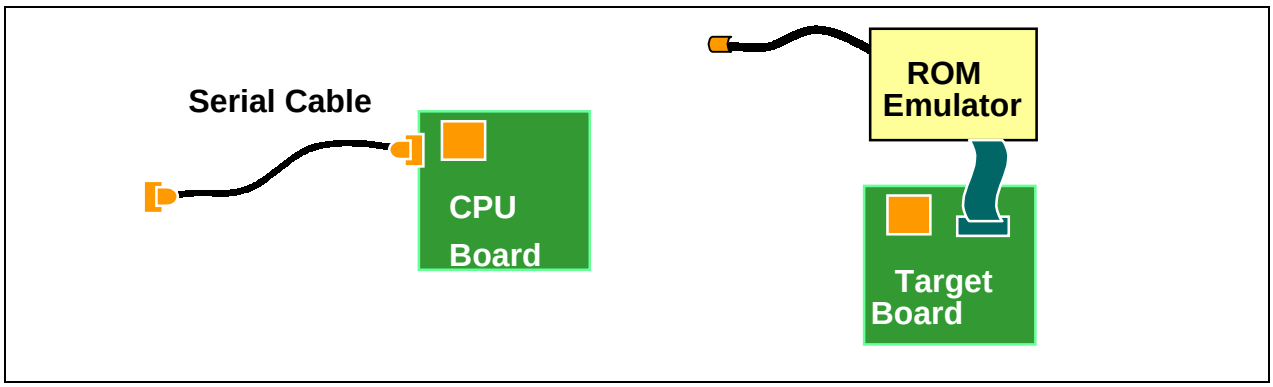
定義

1. **HEW (High-performance Embedded Workshop)**
   ルネサスの統合開発プラットフォーム。アプリケーションの編集，アセンブル，コンパイル，リンク，シミュレーション，デバッグを一貫して行うことができる。
2. **エディタ**
   ソースコード (C 言語またはアセンブリ言語) 作成用のテキストエディタ。
3. **アセンブラ**
   人が解読できるコードから機械語への変換 (オペコード) -> オブジェクトファイルの生成。
   例： MOV R1, R2 => H’1112
4. **コンパイラ**
   高級言語（例：C 言語）から機械語への変換-> オブジェクトファイルの生成。
5. **リンケージエディタ**
   すべての (アセンブラやコンパイラにより出力された) オブジェクトファイルを, 管理できる 1 つのファイルに統合する。 -> MOT, ABS ファイル
6. **デバッガ** (ハードウェアの開発ツールと共に使用。例：E6000, E7, CE, ALE, その他)
   リンケージエディタから出力されたファイルをハードウェア開発ツールにロードして，エミュレーションや，ステップ，ブレーク，トレースなどのデバッグを行う。
7. **シミュレータ**
   実チップの動作をシミュレーションするエミュレータのソフトウェア版。
   リンケージエディタから出力されたファイルを実チップではなく PC 上で実行する。
8. **フラッシュライタ/プログラマ** (ハードウェア開発ツールと共に使用。)
   リンケージエディタから出力されたファイルを MCU のフラッシュメモリに書き込むツール。
   例：FDT (Flash Development Toolkit)
9. **ほかのユーティリティ**
   CallWalker (HEW 内蔵)
   MapViewer (HEW 内蔵)
10. **ソケットアダプタ**
    MCU パッケージ用を，市販のプログラマに取り付けるためのアダプタ。

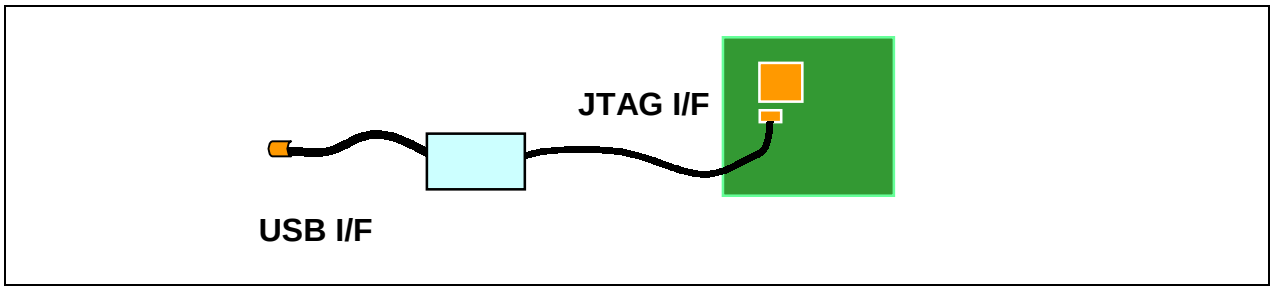
3. ハードウェア開発ツールの分類と比較

ハードウェア開発ツールの市場では多数の定義や名称が氾濫しています。これらのツールの特性に基づいて，ハードウェア開発ツールは大きく 3 つに分類できます。

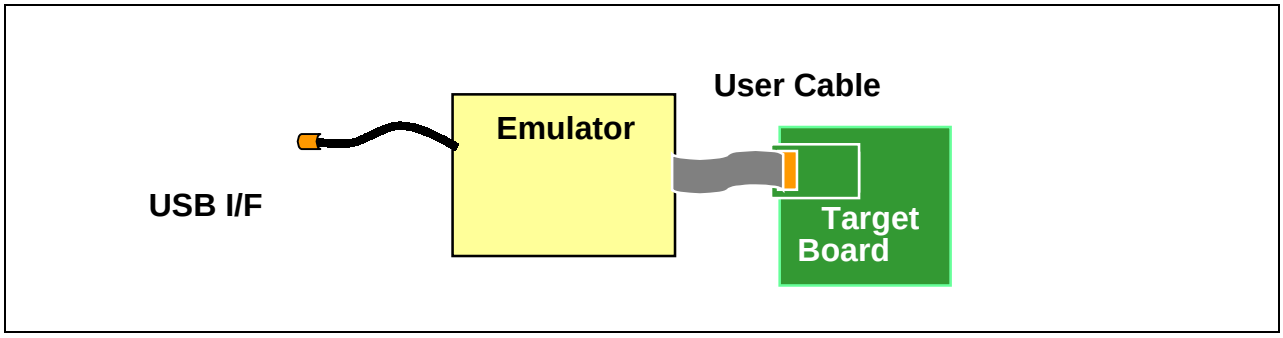
1. CPU ボード，スタータキット，評価ボード，ROM エミュレータ → **CPU ボード**
   実際の CPU を搭載したボードで， PC と通信して簡易なデバッグができるように，ソフトウエアモニタのコードを内蔵しています。



2. JTAG エミュレータ，オンチップデバッガ, BDM (Background Debug Monitor), N-wire → **JTAG 類**
   JTAG ライクに類似したデバイスと通信することを目的とし，デバイスにデバッガ機能が内蔵されています。通常，簡易な配線で接続されエミュレーションをします。



3. ICE (インサーキットエミュレータ) → **ICE**
   エバチップ（評価チップ）を搭載する複雑なシステムで，実チップをシミュレーションできます。これらのシステムは，高度な制御および監視機能を持ちます。ターゲットボードに接続するための専用のケーブルが必要です。



以下の表にハードウェア開発ツールの一般的な特性のリストを示します。

種類	CPU ボード	JTAG 類	ICE
チップ資源	一部使用	未使用	未使用
実チップ	使用	使用	未使用
エミュレーション機能	基本 [30%]	中程度 [50%]	複雑 [100%]
	ロード/実行/編集/ ブレーク	ロード/実行/編集/ ブレーク/トレース	ロード/実行/編集/ ブレーク/プログラム実行中のメモリのトレース/複雑なブレークとトレース/時間測定
使用方法	基本システム，ベンチマーキング	中程度のシステム	複雑なシステム開発
価格	ほぼ無償	低価格	高価格
製品化に要する時間	高速，開発が容易	高速	低速，設計が困難
備考	実際のアプリケーションボードに移植可能。無償ツール。	動作保証の経済的な基本開発ツール (ICE とは異なる)	特に，プロジェクトを一から始めるときなど，ハイエンドなアプリケーションに必要不可欠なツール。
例	SLP CPU board, BAG など	E7, E10	E6000, CE, ALE

一般的な用語を使用しましたが，以下の例外があります。

- RTOS プロジェクトの開発に，CPU ボードの概念を使用する。
- JTAG エミュレータにはトレース機能がないものもある。
- JTAG エミュレータには ICE より高価なものもある。

ハードウェア開発ツールのエミュレーション機能の概要を以下に示します。

特長	CPU ボード	JTAG ライク類	ICE
ダウンロード速度	低速 (シリアルインタフェース使用が多い)	低速 (シリアルインタフェース使用)	高速 (USB, PCI などの使用)
ステップ	ステップ（シミュレーション）	ステップ（制御）	ステップ（制御）
トレース	トレースなし	制限付きトレース (例：4 ジャンプトレースアドレス)	トレースのフル制御 (バス，フィルタ，サブルーチン，タイムスタンプデータなど)
ハードウェアブレーク	デバイスの UBC (User Break Controller) 依存による	制限付きハードウェアブレーク	ブレークのフル制御 (複雑，条件付きなど)
エミュレーション制御	なし	なし	ガード領域アクセス禁止，書き込み保護
時間測定	なし	なし	あり
使用チップ資源	シリアルポート，NMI の使用	マルチプレクスポートピン，メモリ資源の使用	なし
ターゲット	CPU ボード	実チップ搭載のターゲットボードが必要	ターゲットなしで動作可能。ターゲットがない場合，オプションのターゲットメモリを提供
プログラム実行中のメモリアクセス	なし	なし	あり
高度機能	なし	なし	カバレッジ パフォーマンス測定，端子アナライザ

4. 開発ツールの適用と特長

4.1 CPU ボード，スタータキット，評価ボード，ROM エミュレータ

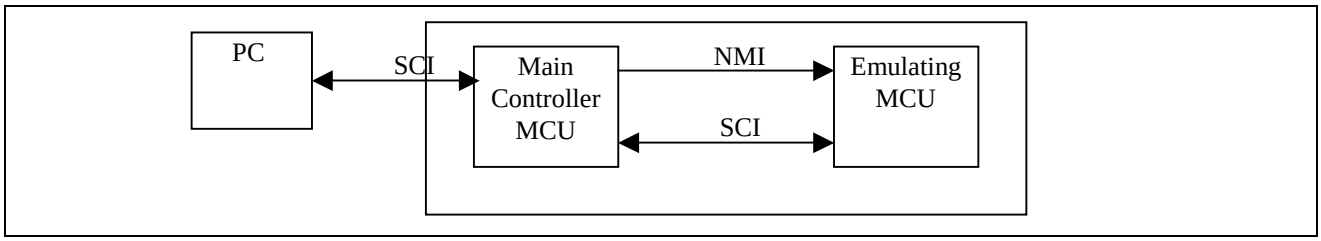
4.1.1 概要

実際の CPU を搭載した基本的なボードで，デバッグ用に PC と通信するためのソフトウェアモニタプログラムを内蔵します。

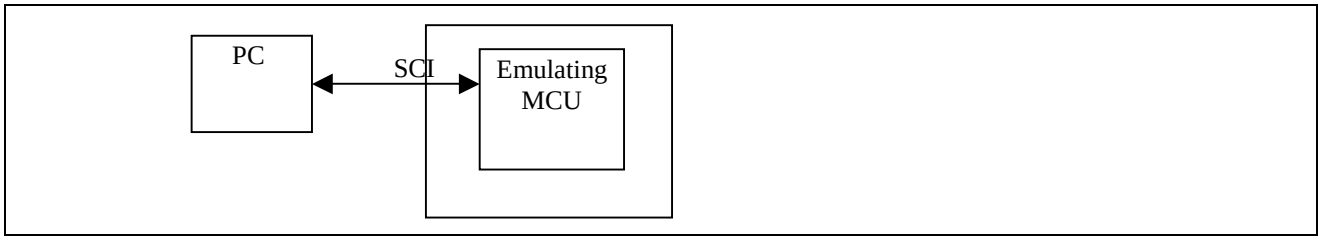
4.1.2 共通の特性

一般的に，この分類のハードウェア開発ツールは，実チップを使用してエミュレーションを行います。PC との通信に，モニタプログラムとシリアルインタフェースを用います。使用方法によっては，CPU ボードは強力で使い勝手の良いツールになります。SLP CPU ボード (H8/38024) を例に示します。モニタプログラムがルネサスの HEW と通信します。この PC GUI を使い C ソースレベルでのデバッギングが可能です。

CPU ボードで PC を制御するにはいくつかの方法があります。



1 つの例は，他の MCU をマスタコントローラとして PC との通信に使用して，エミュレーションする CPU に NMI 割り込みを発生させることです。この設計では，エミュレーションする CPU の NMI を使用します。(エミュレーションする MCU のシリアルポートは PC からコマンドやデータを取得するために使用します。)



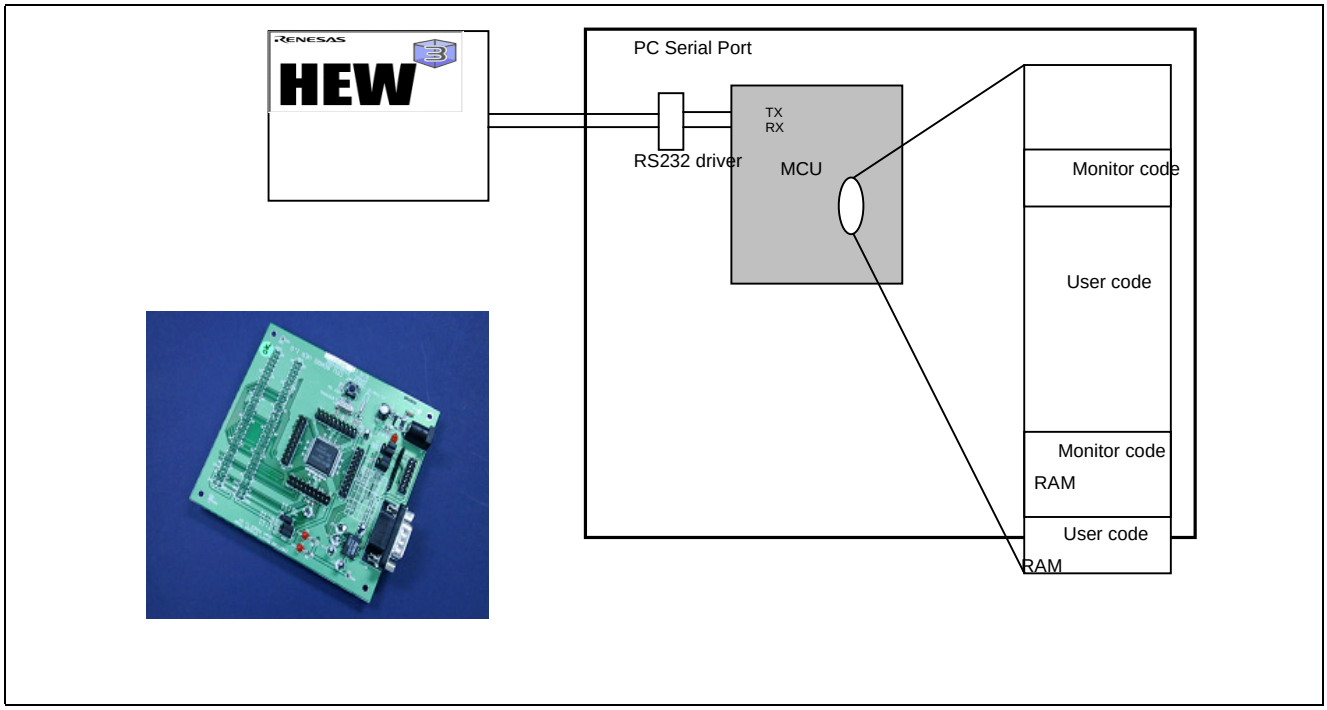
もう 1 つの例は, エミュレーションする MCU のシリアルポートの割り込みを使用して PC と通信する方法です。この場合，ユーザプログラムによって割り込みを禁止してはいけません。PC と MCU との通信ができなくなります。

この主な短所は，メモリやシリアルポートなどの資源を使用することです。アプリケーションでこれらの資源を必要としない場合は影響はありません。ユーザの必要条件は，モニタプログラムをリロケータブルにして, ユーザはアプリケーションをより良く制御できるようになります。通常のモニタプログラムは，フラッシュメモリを 3K バイト程度使用します。シングルステップ機能を削除することにより，1K バイトに低減できます。

MCU の多くは，オンチップのブレークコントローラを内蔵していませんが，ルネサスの TINY シリーズでは，UBC (User Break Controller)を内蔵しています。ユーザは，UBC を利用して，CPU ボードのデバッギング機能を強化できます。また, UBC により, アドレスやデータの値を検出して, ブレーク条件を作成できます。

適用方法や特長が類似しているので，ROM エミュレータはこの分類に属します。

4.1.3 一般的な適用方法



4.1.4 共通の問題点

ユーザが直面する問題を以下に示します。

1. 割り込み処理
   SLP CPU ボードでデバッグする間, HEW は常に CPU ボードをチェックして CPU ボードが「ブレークモード」(ブレークポイント設定) に遷移したかどうかを確認します。CPU ボードが，長い期間，より優先度の高い割り込み処理を実行すると，HEW はタイムアウトエラーを発行します。
2. ソースプログラム
   場合によっては，外付けメモリが RAM ではなく，フラッシュメモリのような，書き込み用のアルゴリズムが必要なデバイスのときがあります。ユーザはこの外付けメモリをアクセスするために PC GUI の使用を希望するかもしれません。この場合，CPU のソースプログラムは編集する必要があります。しかし，通常，モニタプログラムは，ソースプログラムではなくライブラリフォーマットや S タイプレコードフォーマットで提供されます。
3. 制限付きデバッグ機能
   トレース機能なし。ソフトウェア手段で提供される以外では，トレース機能なし。
4. OS との共存
   OS ベースのアプリケーションに移植するのは困難です。この主な理由は，両方のモジュール（モニタプログラムと OS）がシステムをフルに制御しようとするからです。これにより，タイムアウトと資源との競合が起こります。しかし，モニタプログラムを OS の一部のモジュールにすれば，移植が可能です。

4.1.5 使用方法

1. 周辺機能の即座の評価
   — CPU ボードには他の部品が搭載されていないため，PWM，複雑なタイマ，A/D 変換器などの周辺機能の評価に適しています。
2. 基本デバッグ用に実際のアプリケーションボードへの移植
   — CPU ボードにはシリアルポート 1 本だけを必要とし，ユーザは以下のことが可能です。
   (a) ボード上にシリアル RS-232C ドライバとコネクタを配置して，PC と通信する。
   (b) ボード上に 4 ピンコネクタ (VCC, GND, TX, RX) を配置して，ボードの空間と電力を低減するため，外付けのシリアルドライバを組み込む。
3. パラメータの調整のために実際のアプリケーションボードへの移植
   — 生産するフロアでシステム全体がセットアップされた後で，アナログのアプリケーションで，いくつかのパラメータを微細調整する必要がある。
4. ベンチマーキング
   — ベンチマークのソフトウェアを実行して外部の手段 (ポート端子をトグル動作する) で実行時間を測定する。

4.1.6 長所と短所

長所：

- 低価格または無償
- 簡易な評価を提供
- 実際の場所でのデバッグが可能
- 膨大なプロジェクトグループ内の 2 次的ツールを提供
- 実チップでの高速動作のサポート

短所：

- 制限されたデバッグ機能 (トレースなし，ハードウェアブレークなしなど)
- ユーザ資源 (ROM，シリアルポートなど) の使用
- 監視用に CPU のサイクルスチール
- フラッシュメモリ搭載デバイスでのみ動作可能
- モニタプログラムを考慮しなければプログラム破壊の可能性

4.2 JTAG エミュレータ，オンチップデバッガ, N-wire

4.2.1 概要

テスト中のデバイスは，JTAG ライク類のデバイスと通信するために，オンチップのデバッグ機能を持つことが必要です。通常は，簡単な配線を介してエミュレーションを実行します。

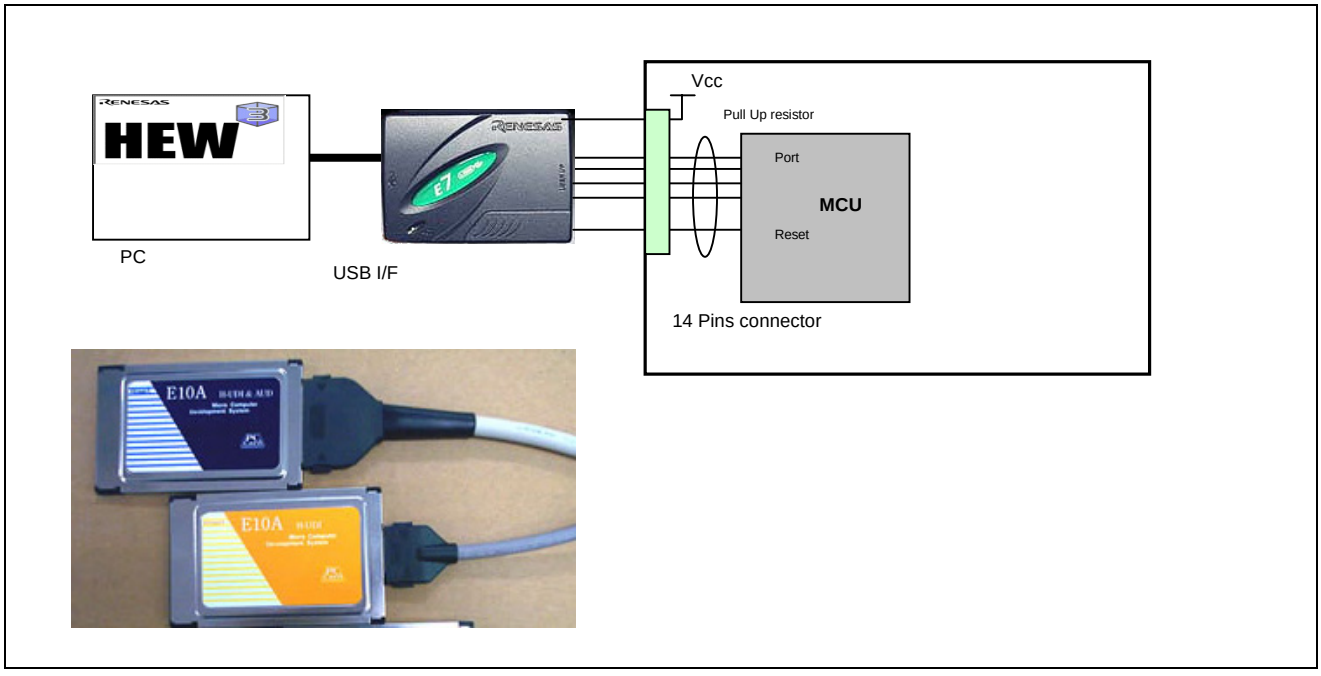
4.2.2 共通する特性

MCU がオンチップ機能をサポートするように設計されていれば, この種類の開発ツールを使用することができます。一般的に，この種類の開発ツールは通信用に 2, 3 本の配線を使用します。制御はシリアルで行われるため，ダウンロード速度が制限されます。したがって，リアルタイムトレース (PC にデータを返すのが遅いため) や詳細なトレース記録は不可能です (実チップの空間を占有するため)。他の制限には, エミュレーション機能の制限があります。この種類の開発ツールでは，ロード，実行，ステップ，ブレーク，トレースの基本機能しか使用できません。このため，ユーザは複雑なバグを検出するためにはさまざまな技術を必要とします。

しかし，JTAG 類のエミュレータは，効率的で使い勝手の良い開発ツールです。コンパクトな本体で，信頼できるコネクタで接続して簡易なデバッグができます。実チップを使用するため，デバッグ後の動作条件の整ったアプリケーションが保証されます。

トレースの欠点を補うために，ルネサスの E10 JTAG エミュレータには，オプションで AUD インタフェースがあります。これは実行中の外部バスサイクルを取得するためのものです。これによって，エミュレータの機能を強化できます。(キャッシュの動作などの内部バスサイクルは取得できません。) エミュレータが使用するメモリ資源は少ないです。

4.2.3 一般的な適用例



4.2.4 共通の問題点

ユーザが直面する共通の問題点は，以下のとおりです。

- 制限されたエミュレーション機能
- 不十分なトレース情報
- ファイルのダウンロード速度の遅さ

4.2.5 使用方法

JTAG エミュレータは，ほぼすべての種類の開発状況において使用できます。

4.2.6 長所と短所

長所：

- 低価格
- 簡易で信頼性のある接続
- 実チップでの高速動作をサポート
- エミュレータ未接続時の動作の保証

短所：

- 高度なエミュレーション機能の不足 (複雑なトレース，イベント，ブレークなど)
- 低通信速度 (シリアル)

4.3 ICE (インサーキットエミュレータ)

4.3.1 概要

評価チップを搭載した複雑なシステムで，実チップを実際にシミュレーションできます。システムはより高度な制御は監視機能を持つことができます。ターゲットボードに接続する専用のユーザケーブルが必要です。

4.3.2 共通の特性

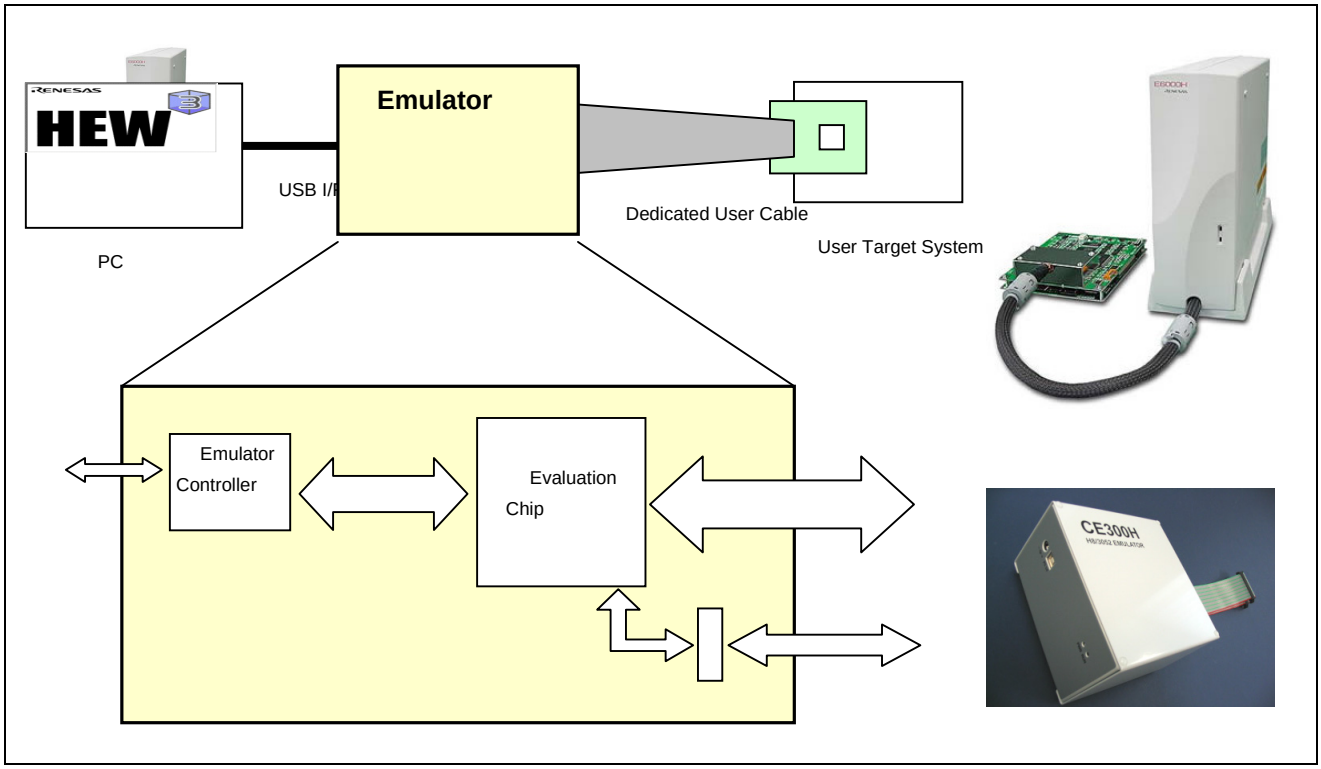
エミュレータは最も強力なハードウェア開発ツールです。複雑なため，初心者が使用するのは難しいかもしれません。この開発ツールに対する不満の多くは，価格とターゲットボードとの接続に関することです。また，一般的に言われることは，利用できる高度な機能を多くのユーザが十分に使いこなせていないということです。

必然的に，エミュレータは，特に新規プロジェクトの開発においては，代替の効かない重要な開発ツールです。複雑な状況を制御するため，リアルタイムトレース，プログラム実行中のメモリアクセス機能，イベントブレークポイントなどの機能が必要になります。これらの機能は他の範囲のツールでは達成できません。

エミュレーションをコントロールするため，エミュレータの制御範囲はターゲット全体に及びます。たとえば，エミュレータはリセットのマスクを取り除いたり，クロックを制御したり，プログラム実行中にメモリをアクセスしたり，ソフトウェアブレークを設定したりできます。これらの機能を理解しないと，外部のウォッチドッグリセットがかからない, 間違ったクロック周波数で動作する, リアルタイム実行ができない, ブレークできない，などの不明な問題が起こります。

近年，MCU の動作速度は 4MHz から 40MHz の範囲へと，10 倍に増加しています。さらに，新規デバイスの立ち上げは，より早くなっています。これらの動向により，エミュレータの設計の難易度を高め，市場への出荷時期に対する需要は，これらの開発ツールにマイナスのインパクトを与えています。

4.3.3 一般的な適用例



4.3.4 共通の問題点

ユーザが直面する共通の問題点は，以下のとおりです。

1. 実チップがエミュレーションしたプログラムで動作しない
2. ユーザケーブルの接続性が悪い
3. ユーザケーブルが重く大きい
4. 外部からの発振クロックがない
5. ターゲットボードからの NMI やリセットがない
6. 信号が不定である
7. 実チップで性能が異なる
8. ブレークしない

4.3.5 使用方法

エミュレータはすべてのアプリケーションで使用できます。しかし，価格や効率を考慮すると，エミュレータは，RTOS やタイミングが重要なプロジェクトなど，新規の複雑なプロジェクトに適しています。

エミュレータの使用の詳細は，次の章のエミュレーション機能についての記述を参照してください。

4.3.6 長所と短所

長所：

- 不良または不完全なハードウェア上でも動作できる
- ターゲット MCU の資源を使用しない
- ターゲットボードなしのスタンドアロン型で動作できる
- 複雑な問題を解決できる
- 高度な機能を提供できる
  — 複雑なトレースおよびブレーク制御
  — パフォーマンス分析とカバレッジ機能

短所：

- 高価格
- 効率良く使用するには高度な理解が必要である
- 専用ユーザケーブルであっても，接続に関する問題が起きる可能性がある

5. エミュレーション機能

この章では，ユーザがエミュレーション機能を使用により理解が深まるように，あらゆる範囲の開発ツールの，共通して使用できる各種機能を説明します。また，エミュレータだけが持つ特長や機能についても説明します。

5.1 ロード

ユーザプログラムを MCU にロードする機能は必須です。しかし，その能力は使用の PC ソフトウェアにより異なります。SLP の CPU ボード，E7 や E6000 エミュレータなどの多くのルネサスの開発ツールは，共通のユーザインタフェース HEW (High-Performance Embedded Workshop) を使用します。この GUI (Graphical User Interface) により，ユーザは，HEX, Bin, S レコード (mot), Elf/Dwarf (abs) などのさまざまな形式のファイルをダウンロードできます。Elf/Dwarf ファイルには C ソースレベルデバッグ用の特定の情報を含みます。

書き込み可能なメモリ空間があるときだけファイルのロードができます。

内蔵 ROM メモリは，エミュレータの内蔵 RAM で代用しているので，ロードできます。

以下の領域にはロードできません。

- 外付けメモリやオプションの貸し出しメモリがない ROM レス領域
- 初期化されていない DRAM 領域

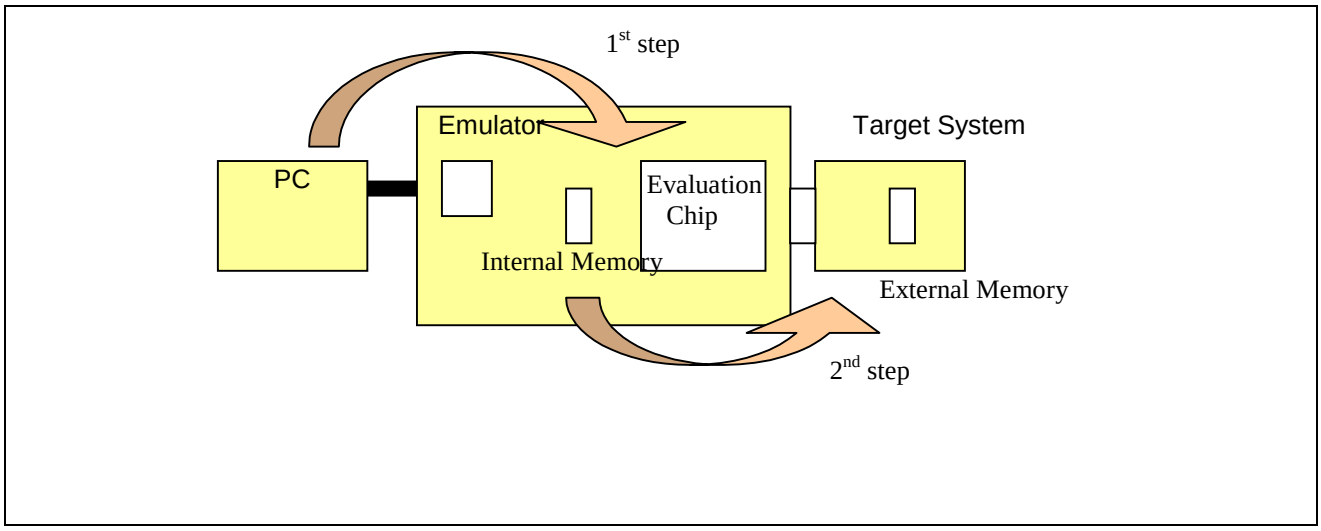
5.2 メモリアクセス

メモリは，バイト，ワード，および浮動小数点の形式でアクセスできます。

5.3 ダウンロード/アップロードの速度

シリアルインタフェースのエミュレータの方が, PCI バスインタフェースのエミュレータより, ダウンロード速度が低速です。

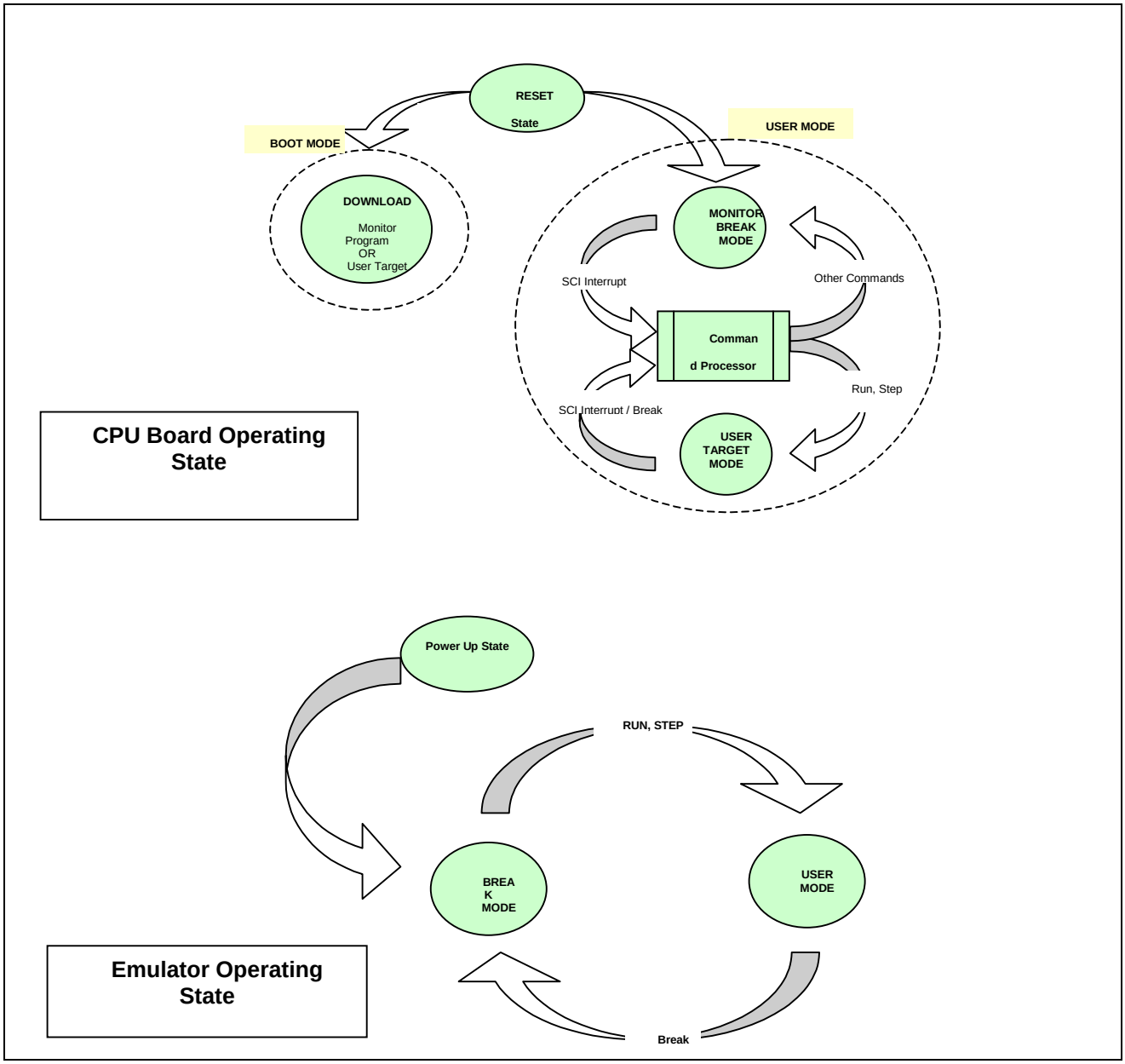
ダウンロード先によっても，ダウンロード速度が異なります。下図にこの理由を説明します。



ダウンロード先がエミュレータの内部メモリの場合，高速です。ダウンロード先が外部のユーザ領域の場合，PC からエミュレータ内部へ，そしてエバチップを介して外部のユーザ領域にデータを転送しなければなりません。この場合，ユーザがエバチップの動作周波数を低く設定すると，さらに時間がかかります。

5.4 実行

「実行」はすべての開発ツールに必要な基本のコマンドです。JTAG エミュレータや一般のエミュレータでは，このコマンドを実行すると，ターゲットボードが実行されます。もしそうでなければ，MCU は他のモード (一般的には「ブレーク」モードと言われる) で動作します。CPU ボードの場合，MCU は電源を投入されると動作を開始し，「コマンド待ち」状態で動作します。ホストコンピュータから「実行」コマンドを受け取ると，MCU はユーザプログラムのルーチンに飛んで実行します。



5.5 ブレーク

「ブレーク」は, 実行コード (ユーザモード) を終了してモニタ状態またはコマンド待ち状態 (ブレークモード) に遷移する割り込み動作です。

ブレークには，ハードウェアブレークとソフトウェアブレークの 2 種類があります。

これらは，基本的な適用方法をもとに，名付けられています。

この 2 つのブレークの特長は，ブレーク時の PC 条件です。

- ハードウェアブレーク時，PC はイベント発生後に停止します。
  例：「アドレス=H'1000」の条件のとき→PC アドレス H'1002 で停止 (MCU による)。
- ソフトウェアブレーク時，PC は設定条件で停止します。
  例：「アドレス=H'1000」の条件のとき→PC アドレス H'1000 で停止。

5.5.1 ハードウェアブレーク

ハードウェアブレークはハードウェアの手段によって実行され，ブレーク前に条件が満たされる必要があります。この複雑なブレーク条件により，評価に必須な条件をキャプチャできます。

ブレーク条件には，以下のものが含まれます。

1. アドレスまたはアドレス範囲の検出
2. データまたはデータのマスクの検出
3. アクセス種別 (書き込み，読み出し)
4. アクセス領域 (ROM, RAM, DRAM)

設定には，“AND”，“OR”，“IF-ELSE”条件を用いることができます。E6000 エミュレータでは，複雑なイベントを制御でき，イベントを 6 レベルまで連続的につなげることができます。イベント発生回数，条件成立後のサイクル数などの条件も設定できます。

これにより，以下の複雑なイベント情報も取得できます。

- 可変な COUNT (アドレス=H'1234) にデータ 10 を書き込む
- データテーブル(アドレス範囲：H'1000～H'2000) にデータ H'FF を書き込む
- 可変な LOG に 100 回アクセスする
- ISR (アドレス=H'2024) を終了して 20 サイクル後ブレークする (割り込み後のイベントをトレースするため)

5.5.2 ソフトウェア (PC) ブレーク

一般的に，PC ブレークと呼ばれます。

HEW では，ユーザがプログラムコードの行をクリックして，ブレークポイントを設定できます。PC ブレークポイントを設定すると，そのコードは特定のコードに置き換えられ，MCU が他のモードに遷移します。ある種類の CPU ボートでは，TRAP 命令が使用されます。

これはソフトウェアにより実行されるので，フラッシュメモリ，EEPROM などの読み出し専用メモリには使用できません。このような場合には，代わりにハードウェアブレークを使用してください。

通常，PC ブレークの数は，255 に制限されています。ユーザが 255 個の PC ブレークポイントを設定すると，PC GUI は実行するたびに，255 個の命令を置き換えなければなりません。これは，「実行」コマンドの速度を遅くします。この遅延は，CPU ボードでより顕著になります。

DRAM 動作でも重要な問題が起きます。DRAM の初期化の前に，PC ブレークポイントが DRAM 領域に設定されると，PC ブレークポイントは正しく設定されません。

5.6 エミュレーション制御 (ガード領域 & ライトプロテクト領域) /アドレスマッピング

この機能はエミュレータ特有です。MCU を選択すると, デフォルトのマッピングが設定されます。ここで, ROM 領域をライトプロテクト領域として, リザーブ領域をガード領域として定義します。デフォルトで, 外部メモリもガード領域として定義されます。ユーザが外部領域を使用するには, 外部読み出し/書き込み領域として設定する必要があります。（この手順をマッピングと呼びます。）

すべての MCU において, 5 つの主な領域と考えられるマッピングを以下の表に示します。

MCU 領域	マッピング例	備考
内蔵 ROM	**内蔵ライトプロテクト領域**	
内蔵 RAM	**内蔵書き込み/読み出し領域**	
IO	**内蔵書き込み/読み出し領域**	
外部領域	**ガード領域** **外部読み出し専用領域** **外部書き込み/読み出し領域**	**ユーザによる変更可能。**
	オプションの読み出し専用領域 **オプションの書き込み/読み出し領域**	**次章の説明を参照。** **ターゲットボードが準備できない場合，ユーザはこの領域を一時的なデバッグ用の領域として使用できる。**
リザーブ領域	**ガード領域**	

ガード領域やライトプロテクト領域の目的は，実行しているプログラムに問題があるとユーザに警告することです。たとえば，プログラムの暴走，ROM 領域への書き込みなどです。プロジェクト管理によっては，ユーザは，個別のコードを他のコードと重なることを避けるために，特定の領域をガード領域やリザーブ領域に設定することができます。

5.7 貸し出しメモリ/オプションのメモリ

貸し出しメモリとは，エミュレータの内部メモリを指す一般的な言葉です。ユーザは MCU の内蔵 ROM, RAM をエミュレーションするために使用する SRAM を貸し出しメモリと呼ぶ場合があります。しかし，混乱を避けるため，MCU の内部メモリを内蔵 ROM, 内蔵 RAM と呼ぶのが適切です。

エミュレータでは，貸し出しメモリを「拡張」することができます。この場合，この拡張貸し出しメモリとは, ターゲットボードがまだ準備できないときに, 一時的なエミュレーションのために使用するエミュレータ内のメモリを指します。

この拡張貸し出しメモリは，MCU のメモリマップの外部領域にマッピングされなければなりません。したがって，外部読み出し／書き込み領域または外部ライトプロテクト領域にマッピングする代わりに，ユーザはオプションの読み出し／書き込み領域またはオプションのライトプロテクト領域にマッピングする必要があります。

この場合，ターゲットボードが接続されていると，エミュレータのマッピングはオプションのメモリ (内部) に設定されます。エミュレータは内部のオプションのメモリをアクセスするので，競合は起きません。

5.8 トレース

トレースは，ユーザが最新のブレーク条件までに実行されたコードを評価するために必須の機能です。

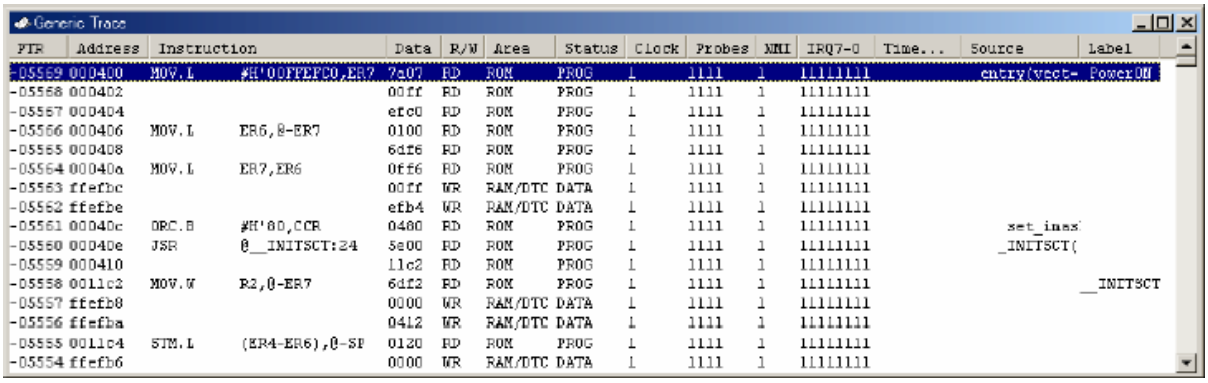
E6000 エミュレータのトレース例：

- トレース範囲: 256k サイクル
- 各サイクルに含まれる情報は以下のとおりです。
  — アドレス
  — データ
  — アクセス種別 (読み出し/書き込み)
  — アクセス領域 (ROM, RAM, DRAM など)
  — 外部プローブ
  — 割り込み (IRQ[0-7])
  — タイムスタンプ

使用例：

- ライトプロテクトブレークが発生すると，トレース結果により，ユーザがデータを書き込み中にポインタがデクレメントされすぎ，ROM 領域に達したことがわかります。
- ルーチンが外部ソースからデータパターンを読み込むという，事前に設定された条件に達したことを示します。トレース結果により，計算結果の誤りがわかる場合があります。

E6000 エミュレータによって取得されたトレース情報の例を下図に示します。



5.8.1 トレース範囲

技術的に，トレース範囲は広ければ広いほど，多くの情報が得られます。

しかし，多くの場合，最新の数サイクルだけが検討されます。これは，プログラムの実行を停止した条件が, 事前に定義された条件であること (イベントブレーク, マッピングブレークなど)に起因します。そして，通常，停止の理由は最新の数サイクルでトレース取得されます。

連続したイベントの解析などの RTOS アプリケーションでは，トレース範囲が広いほど，有益です。

5.8.2 トレースのフィルタリング

通常，トレースできる範囲には制限があります。したがって，フィルタを設定することにより，ユーザは同じトレース範囲でより多くの情報を取得できます。また，これにより，不要な情報の代わりに，ユーザが特定のルーチン解析に焦点を当てられるため，トレース情報が読みやすくなります。

5.8.3 タイムスタンプ

タイムスタンプとは，命令のための取得時間です。タイムスタンプは蓄積保管されます。タイムスタンプは，起動時のルーチン実行時間の測定に便利で,特にトレースフィルタは顕著です。この場合，いくつかの命令がトレースバッファから除去されることがあります。

5.9 シングルステップ (ステップイン/アウト/オーバ)

シングルステップ実行により，ユーザはコーディングの流れを静的な条件の下でモニタをコントロールする環境を提供します。シングルステップ機能では，現在のプログラムカウンタの命令を実行します。割り込みを検出しても，割り込みサービスルーチンは実行されません。

ステップ実行中，ステップモードで (HEW などでは)，どのようにシングルステップ実行をするかを選択します。

ステップモードには 3 つのモードがあります。

- **自動：**実行モードはアクティブウィンドウに依存します。つまり，C ソースウィンドウでステップ実行する場合，C ソースレベルのステップ実行が起動されます。
- **ソース：**ステップ実行中，ユーザは C ソースレベルのステップを（あれば）モニタできます。つまり，バックグランドで，一連のアセンブリ言語のコードが実行されます。
- **アセンブリ：** ステップ実行中，現在のプログラムカウンタの位置にあるアセンブリ言語のコードが実行されます。現在のウィンドウが C ソースウィンドウの場合，逆アセンブリウィンドウが表示されます。

シングルステップ実行には，3 つの種類があります。

- **ステップイン：** 1 命令だけ実行します。簡単に言うと，次の C 言語またはアセンブリ言語の命令にステップします（ステップモードの選択によります）。
- **ステップアウト：** 現在のルーチンから分岐するまで実行します。
- **ステップオーバ：** 関数呼び出し（およびその関数によって呼ばれた関数呼び出し）を実行して，次の命令で停止します。

C 言語の命令のシングルステップ実行の間，エミュレータの種類によって，動作が異なります。C 言語の命令を解読できるアセンブリ言語コードのすべてをシングルステップ実行する場合があります。または，C 言語の命令を解読するアセンブリ言語のコードを実行する場合もあります。技術的には，どちらの方法も，テストの静的なモードを提供するだけなので，有効です。

CPU ボードでのシングルステップ実行は，他のツールの場合に比べてより複雑です。CPU ボードでは，ステップ実行はシミュレーションされたステップです。したがって，割り込みイベントが発生すると，複雑な問題が発生します。

5.10 MCU への介入/リアルタイム性

すべてのツールはリアルタイム性があるように設計されています。しかし，実行中の MCU から情報を取得しようとするとき，ある程度の MCU へのプロービングが必要となります。技術的に，MCU の通常動作を停止せずに MCU の情報 (たとえば，I/O レジスタ，ターゲットメモリデータなど) をアクセスすることは不可能です。

MCU はユーザモードからブレークモードへ切り替わり，アクセス (レジスタ/メモリの書き込み/読み出し) して，可能な限り短時間でユーザモードに戻ります。これはよく「パラレルオンザフライ」 (POTF) と呼ばれます。以下の章でこの説明をします。

ユーザがこの機能を使用しないとき，MCU は独自の速度で動作できます。POTF はエミュレータの設定で禁止できます。

5.11 モニタ/制御 – パラレルオンザフライとバスモニタ

開発中，ユーザがプログラム実行中に変数の値 (メモリ) をアクセスしたいと希望する場合，以下がその 2 つの方法と機能です。

1. パラレルオンザフライ (POTF) – メモリ内容の介入の書き込み/読み出しを許可する
2. バスモニタ – メモリ内容の非介入読み出しを許可する

POTF - HEW のメモリウィンドウをリフレッシュすると，MCU は割り込みされ，最新のデータを更新します。メモリウィンドウの内容を変更すると，MCU は割り込みされ，メモリ内容を変更します。

バスモニタ - アドレスバスやデータバスは変更内容をモニタリングされます。読み出し/書き込みがあれば，バスモニタデータウィンドウが更新されます。モニタは実行中の MCU への割り込みを引き起こしません。

POTF の介入的側面は，アプリケーションによっては悪影響を与えます。通信のアプリケーションでは，割り込みが原因で通信データを失うことがあります。しかし，ユーザが POTF 機能の使い方を知っていれば，有効的な機能となります。

5.12 外部プローブ

外部プローブはエミュレータの外部のイベントを検出するために用います。外部プローブはターゲットボードの回路に取り付けることができます。プローブ 1 が Low などのイベントが起きると，ブレークを起こすように設定できます。使用できる外部プローブの条件には，Low, High, エッジトリガの 3 種類があります。これにより，外部でイベントが起きた場合のプログラムの追跡がスムーズになります。

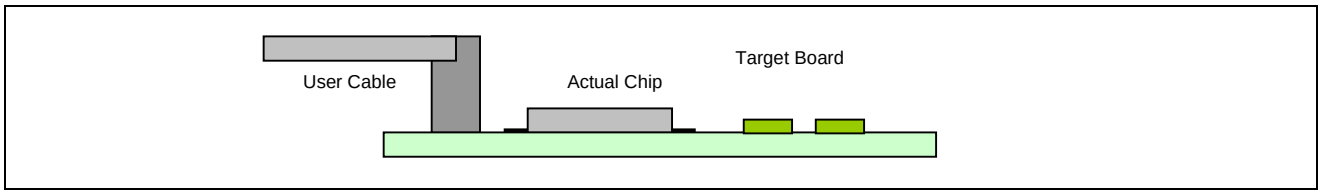
外部プローブのロジックレベルがトレースメモリに取得されるので，この機能はロジックアナライザとして機能できます。外部プローブをターゲットのターゲットボードの異なる位置に付けると，ユーザは外部プローブによって取得されたイベントに対する実行中のプログラムを解析できます。

5.13 ターゲットボードとの接続

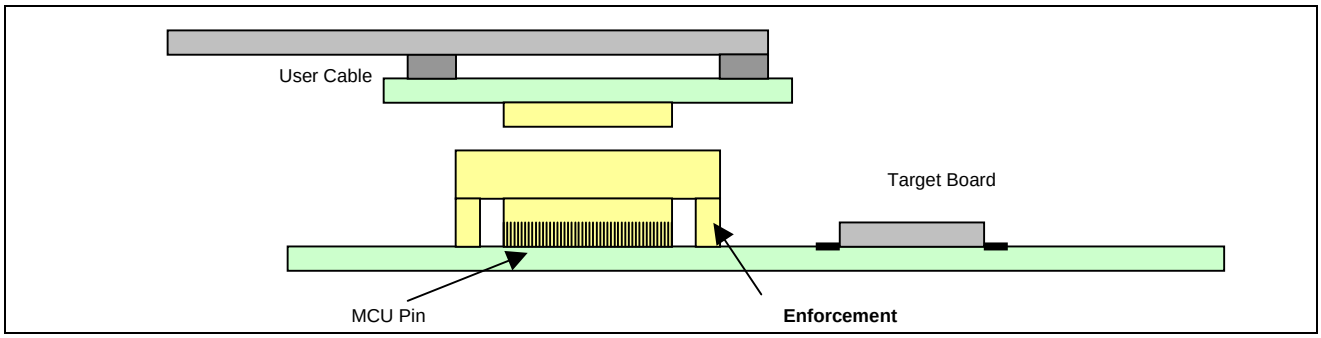
JTAG エミュレータとエミュレータとの接続を以下の図に示します。

JTAG エミュレータの場合のターゲットボードへの接続は，MCU との n-wire 接続です。この接続は信頼性があります。

JTAG エミュレータ接続の例：



専用ユーザケーブルを介したエミュレータとターゲットボードとの接続：

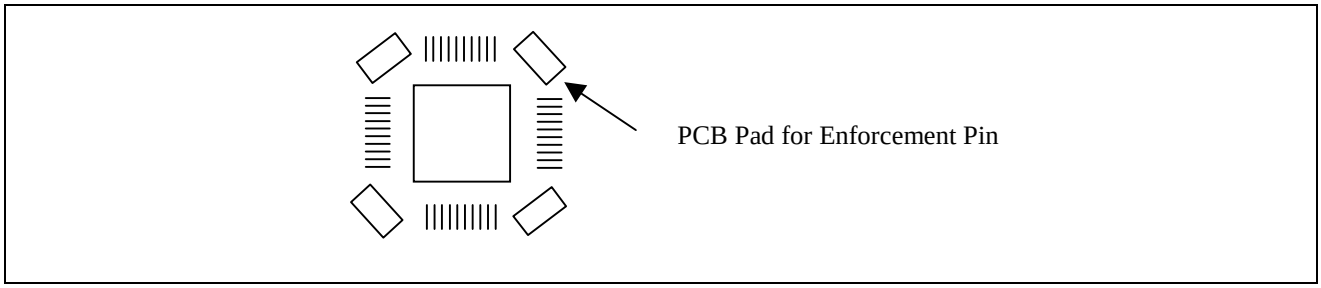


エミュレータの場合のターゲットボードとの接続の問題点を以下に示します。

1. 断続接続
2. 遅延やノイズマージン
3. 信号制御 (NMI, RESET, MODE, STANDBY, CLOCK)
4. 電源電圧フォロワ
5. 発振クロック
6. プルアップ抵抗とケーブル

5.13.1 断続接続

ユーザは，ターゲットボードとの接続には十分注意しなければなりません。また，ターゲットボードとの接続のためのプリント基板の設計は非常に重要です。それらを留意することで，コネクタにとっての基本的な強い基盤を提供することができます。接続を強化するために強化用ピンを考慮し，PCB パターン設計することを強く推奨します。強化用ピンをパッドにはんだ付けすると，グリップが強くなります。ユーザケーブルが太くて大きい場合は特に考慮すべきです。



5.13.2 遅延とノイズマージン

エミュレータの設計において，遅延やノイズを最低限に抑えるために最大限の努力は払いますが，完全に取り除けるものではありません。ユーザはターゲットボードをデバッグする際，これらの問題を考慮してください。一般的には，エミュレータとターゲットボードとの間のケーブル間の遅延は 4ns と推測されています。もし，その間にインタフェース回路が挿入された場合，追加で 4ns の遅延が発生します (使用デバイスにより異なります)。

5.13.3 信号制御 (NMI, RESET, MODE, STANDBY, CLOCK)

他のツールと異なり，エミュレータはターゲットボードをフルに制御します。NMI, RESET, MODE 端子，スタンバイ，クロックなどのターゲットボードの信号が制御されます。

ターゲットボードが使用できないときでも，エミュレータは開発の要求に従って，エミュレーションを制御できます。

例： - 動作モードを拡張モードに設定して，動作周波数を 14MHz に設定する。

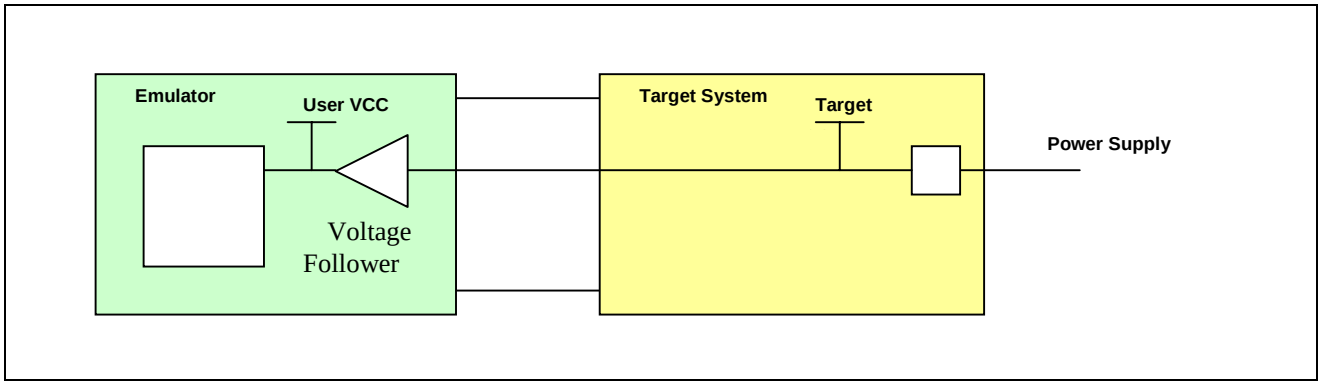
ターゲットボード接続時，ユーザはエミュレータの設定ではなくターゲットボードをベースに設定することができます。しかし，ターゲットボード未接続時には正しいエミュレーションは保障されません。この場合，ユーザはターゲットボードのトラブルシューティングのために調整が必要なこともあります。

例： - エミュレータ内部のより低い動作周波数に変更してシステムからの応答をみる。

- エミュレーションを制御することを目的として，ターゲットボードの外部ウォッチドッグリセット，または例外の NMI などのマスクを外す。

5.13.4 電源と電圧フォロワ

ターゲットボードをエミュレータに接続すると，ユーザはターゲットボードの電源がエミュレータにロードされることを注意しなければなりません。

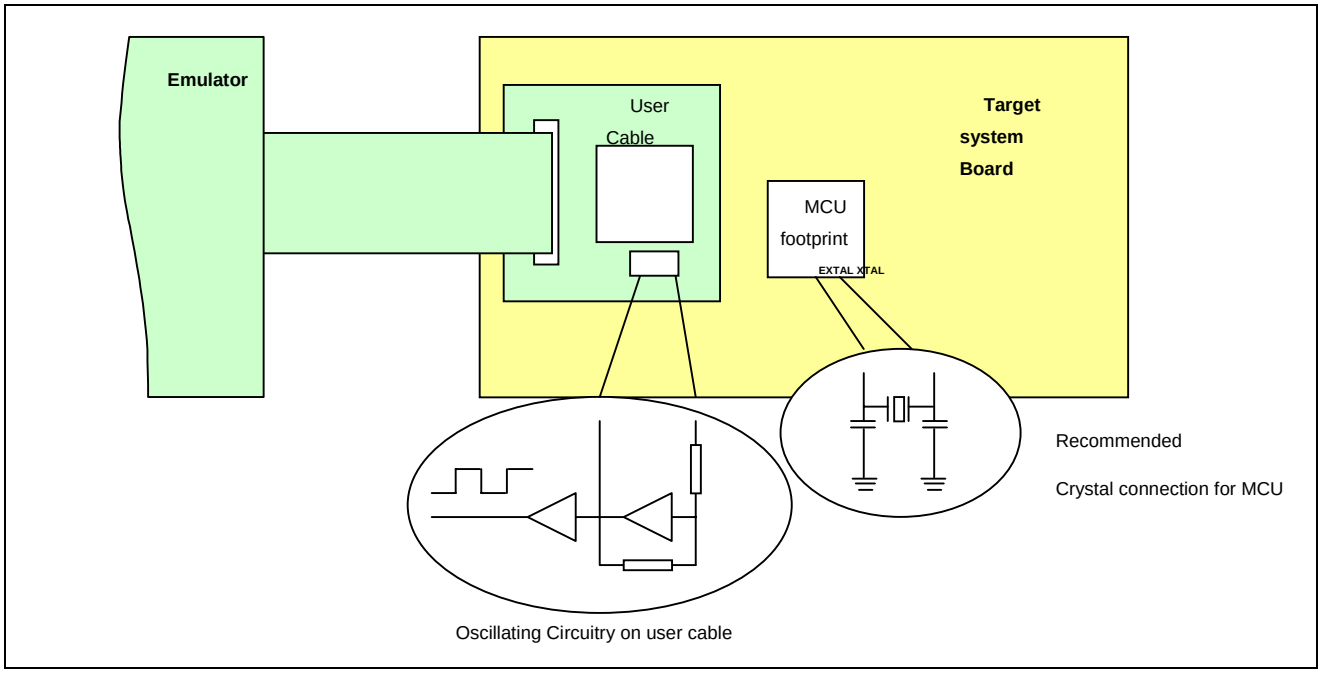


上記の図から，ハードウェア設計者は，ターゲットボードからエミュレータへ電流が流れないことを確認できます。エミュレータは，ターゲットボードの電源 (ターゲット Vcc) をもとに，内部動作用に内部電圧 (ユーザ Vcc) を生成します。ターゲットボードの電流測定をするとき，ユーザは，エミュレーション中の MCU がターゲットボードの電力を消費しないよう考慮しなければなりません。

5.13.5 発振クロック

一般的に，水晶発振器と 2 つのコンデンサを MCU 端子である EXTAL と XTAL に接続します。水晶発振器を発振させるための回路が内蔵されているので発振できます。エミュレータでは，エミュレーションする MCU (エバチップ) をメイン基板またはユーザケーブル先端部に置くことができます。エバチップを主要部に置くと，発振しません。したがって，そのような場合，発振部をユーザケーブル (下図参照) に置く必要があります。この場合，発振クロックは，ユーザケーブルを通ってエミュレータの主要部に供給されます。

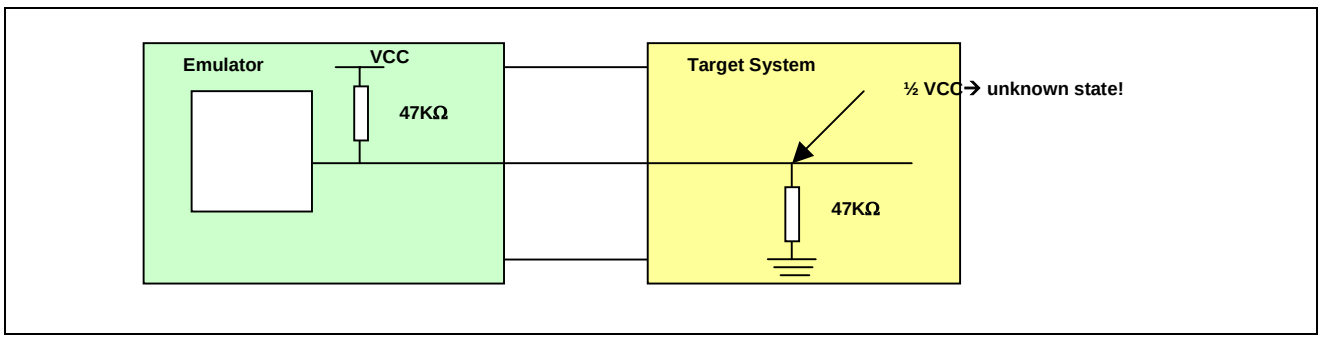
ユーザがエミュレータとターゲットボードの直接接続を望む場合，(これは実際のユーザケーブルの面積により，リンクが弱くなることを防止するためです。)ユーザは，クロック入力が発振することを理解しなければなりません。この場合，代わりに，ユーザはエミュレータの内部クロックを使用しなければいけません。



5.13.6 プルアップ抵抗

一般的に, エミュレータ内部のほとんどの入出力端子にプルアップ抵抗を付ける必要があります。これは, 検出された状態を保持すると, 長いユーザケーブルからターゲットボードへの駆動電流を増強するためです。アナログ端子 (ADC, DAC など) には，プルアップ抵抗はありません。このことは，2 つの問題を引き起こします。

- エミュレータから実チップに置き換えるとき，ユーザは抵抗の影響を見過ごす
- エミュレーション中，ユーザは同様のプルダウン抵抗を接続する。これにより，電圧分割が起こり，端子に予期せぬ状態が発生する

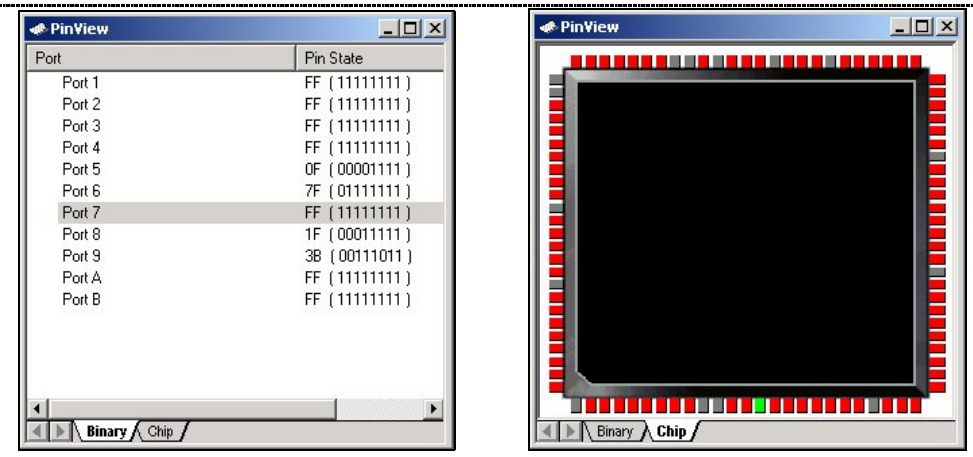


エミュレータのインタフェース詳細については, 各ユーザーズマニュアルを参照してください。場合によっては，ユーザはエミュレータのプルアップ抵抗を DIL ソケットから抜き取ることで削除できます。

5.14 Pinview

エミュレータとターゲットボードとの接続性はいつも懸念すべき要素です。ルネサス製の CE エミュレータシリーズでは，この問題に対処するために Pinview 機能があります。これは，その名が示すように端子の状態を監視する機能です。

例：CE300H Pinview ウィンドウの図



RED : HIGH
GREEN : LOW
GREY : NOT MONITOR

エミュレータは，ターゲットボードとの接続時に，ハードウェアですべての端子を常に監視します。したがって, エミュレーション中の MCU 動作へのサイクルスチールまたは介入は起きません。この情報は PC GUI (HEW) に反映されグラフィックで表示されます。これにより，ユーザはすべての端子の状態 (High または Low) をすぐ閲覧できます。また, 実行中のプログラムを測定する基本的な手段を提供できます。(たとえば, BUSY 信号は常にアサートされるなどです。) 静的モード (プログラム停止時) には，ユーザは，マルチメータを使わずに，各 MCU 端子を目視によりアクセスでき，的確に判断できます。

5.15 MCU とエミュレータの相違点

MCU とエミュレータでは，電源投入時とリセット状態が異なります。一般的には，起動時の MCU レジスタの状態は不定であるのに対して, エミュレータではレジスタの値をある特定の値に初期化します。しかし，C のプロジェクトで HEW を使用する場合には重要ではないことがあります。 (HEW のプロジェクトジェネレータはすべてのレジスタを設定します。)

例：H8 MCU とエミュレータの相違点

状態	レジスタ	エミュレータ	H8 MCU
エミュレータの初期化 (電源投入)	PC	**パワーオンリセットベクタ値**	**パワーオンリセットベクタ値**
	ER0 ～ ER6	H'00000000	**不定**
	ER7 (SP)	H'0FFF10	**不定**
	CCR	**I マスクビットが 1 にセットされ，残りのビットが不定** (B'1 xxx xxxx)	**I マスクビットが 1 にセットされ，残りのビットが不定** (B'1xxx xxxx)
エミュレータの初期化 (リセット CPU コマンド)	PC	**パワーオンリセットベクタ値**	**パワーオンリセットベクタ値**
	ER0 ～ ER6	**不定**	**不定**
	ER7 (SP)	**不定**	**不定**
	CCR	**I マスクビットが 1 にセットされ，残りのビットが不定** (B'1 xxx xxxx)	**I マスクビットが 1 にセットされ，残りのビットが不定** (B'1 xxx xxxx)

ユーザインタフェースの一般的な相違点を以下に示します。

- エミュレータのターゲットボードインタフェースには, プルアップ抵抗および (または) バッファがあるため，信号がわずかに遅延する
- プルアップ抵抗により，ハイインピーダンス信号が High になる
- A/D 変換の分解能がわずかに低下する
- エミュレータの負荷容量が実チップの負荷容量にくらべて大きい。
- 水晶発振器は，フットプリントタイプのユーザケーブルが使用できる場合のみ，使用できます (ボード上に発振回路があります)。ユーザがターゲットボードを直接接続する際は，実際のクロック信号線を EXTAL 端子に接続しなければなりません (**注：**XTAL 端子はありません)。

6. 要約

市場には，さまざまな要求に応えるための各種開発ツールがあります。ユーザにとってもっとも重要なことは，開発ツールへの理解を深めることです。それなしには，いくら高価で高機能な開発ツールであっても，開発作業に多くの混乱を招くことになります。

このアプリケーションノートでは，各種開発ツールのさまざまな特性を紹介しました。ユーザはこのノウハウを適用して開発ツールを選択してください。開発ツールを効率良く使用すれば，効率よい開発サイクルにつながります。

7. 参考文献

1. www.embedded.com
2. The Practice of Programming by Brain W.Kernighan & Rob Pike, Addison –Wesley
3. Fundamentals of Embedded Software where C and Assembly Meet by Daniel W.Lewis, Prentice Hall
4. Programming Embedded Systems in C and C++ by Michael Barr,O’REILLY

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.08.06	—	初版発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。