ARM DS-5

バージョン 5.21

EB FVP リファレンスガイド

ARM DUI0424OJ

ARM DS-5

EB FVP リファレンスガイド

リリース情報

ドキュメント履歴

発行	日付	機密保持ステータス	変更点
A	31 8 月 2008	非機密扱い	RealView Development Suite v4.0 Professional、System Generator v4.0 SP1。
B	31 12 月 2008	非機密扱い	Fast Models v4.1。ARM_RTSM_PATH 関連の変更を追加しました。
C	31 3 月 2009	非機密扱い	Fast Models v4.2。テキストを若干変更しました。device-accurate-tlb パラメータについての説明を追加しました。
D	30 11 月 2009	非機密扱い	RealView Development Suite v4.0 Professional エディション。
E	31 5 月 2010	非機密扱い	RealView Development Suite v4.1 Professional エディション。
F	30 9 月 2010	非機密扱い	RealView Development Suite v4.1 SP1 Professional エディション。
G	31 10 月 2012	非機密扱い	DS-5 v5.12。
H	31 12 月 2012	非機密扱い	DS-5 v5.13。
I	31 3 月 2013	非機密扱い	DS-5 v5.14。
J	30 6 月 2015	非機密扱い	DS-5 v5.15。
K	30 9 月 2013	非機密扱い	DS-5 v5.16。
L	31 12 月 2013	非機密扱い	DS-5 v5.17。
M	30 6 月 2014	非機密扱い	DS-5 v5.19。
N	30 9 月 2014	非機密扱い	DS-5 v5.20。
O	20 3 月 2015	非機密扱い	DS-5 v5.21。

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to ARM’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow ARM’s trademark usage guidelines at *http://www.arm.com/about/trademark-usage-guidelines.php*

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

非機密著作権情報

本書は、著作権などの権利により保護されており、本書に含まれる手順または実装に関する情報は 1 つ以上の特許または申請中の特許により保護されている可能性があります。本書のいかなる部分も、ARM から事前に書面による明示的な承諾なく、何らかの形式や方法で無断複製することは許可されていません。**特に記載がない限り、明示的であるか黙示的であるかを問わず、また禁反言やその他いかなる知的財産権のライセンスを許諾するものではありません。**

本書の情報には、実装により、いかなる第三者の特許も侵害されないことを確認する目的で情報を使用せず、第三者にもそれを許可しないと承諾することを条件としてアクセスすることができます。

本書は、「現状」のまま提供されます。ARM は、明示的、黙示的、または制定法上のいずれを問わず、いかなる表明も保証も行いません。これには、本書に関連した商品性、品質基準、非侵害、または特定目的への適合性に関する黙示的保証を含むが、これに限定されません。疑義を避けるため、ARM は第三者の特許、著作権、営業機密、または他の権利の範囲および内容に関して、いかなる表明も行わず、識別や理解のための分析も行いません。

本書には、技術的に不正確な箇所および誤記が含まれる場合があります。

法により禁止されていない限りにおいて、ARM は本書の使用により生じた直接的、間接的、特別、付随的、懲罰的、または結果的損害などを含むすべての損害に対して、たとえそのような損害の可能性が事前に告知されていた場合でも、その原因および責任理論の如何に関わらず一切の責任を 負わないものとします。

本書には、商品のみが含まれています。本書の使用、複製、または開示が関連するあらゆる輸出法および輸出規制に完全に準拠し、本書が全体であれ一部であれ、該当する輸出法に違反して直接的または間接的に輸出されることがないことを保証する責任を負うものとします。ARM のお客様に関連して「パートナー」という言葉が使用されている場合でも、他会社と提携関係を設立することや、言及することを意図するものではありません。ARM は、通知することなくいつでも本書を変更することができます。

本契約のいずれかの規定と、ARM と締結された本書の内容を含む署名済みの書面契約の間に矛盾がある場合、署名済みの書面契約を本契約の規定より優先するものとします。本書は、便宜上、他言語に翻訳される場合がありますが、本書の英語版と翻訳との間に矛盾がある場合、契約書の英語版に含まれる規定を優先することに同意するものとします。

記号 (® または ™) が付いた言葉およびロゴは、ARM Limited や関連会社の EU またはその他の国における登録商標および商標です。All rights reserved.本書に記載されている他の製品名は、各社の所有する商標です。ARM の 商標の使用に関する次のガイドラインに従ってください。*http://www.arm.com/about/trademark-usage-guidelines.php*

Copyright © [2008-2010, 2012-2015], ARM Limited or its affiliates. All rights reserved.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

機密保持ステータス

本書は非機密扱いであり、本書を使用、複製、および開示する権利は、ARM および ARM が本書を提供した当事者との間で締結した契約の条項に基づいたライセンスの制限により異なります。

無制限アクセスは、ARM 社内による分類です。

製品ステータス

本書の情報は最終版であり、開発済み製品に対応しています。

Web アドレス

http://www.jp.arm.com

目次
ARM DS-5 EB FVP リファレンスガイド

序章

本書について 7
ご意見、ご感想 8

第 章 1 *はじめに*

1.1 *システムモデルの概念* 1-10
1.2 *EB FVP の概念* 1-11
1.3 *EB FVP の構成* 1-12

第 章 2 *EB FVP の使い方*

2.1 *EB FVP の起動* 2-16
2.2 *EB FVP の設定* 2-17
2.3 *EB FVP CLCD ウィンドウ* 2-18
2.4 *EB FVP での Ethernet の使用* 2-21
2.5 *システムモデルによる端末の使用* 2-23
2.6 *仮想ファイルシステム* 2-25
2.7 *ビルド済みの FVP で VFS を使用するためのパス名* 2-27

第 章 3 *EB FVP に関するプログラマ用リファレンス*

3.1 *EB モデルメモリマップ* 3-29
3.2 *EB モデルコンフィギュレーションのパラメータ* 3-33
3.3 *EB と CoreTile ハードウェアとモデルの相違点* 3-39

序章

この前書きでは、次について紹介します。*ARM DS-5 EB FVP リファレンスガイド*.

このドキュメントは、次で構成されています。

- *本書について*(7 ページ).
- *ご意見、ご感想*(8 ページ).

本書について

本書では、エミュレーションベースボード固定仮想プラットフォームを設定して使用する方法について説明します。このモデルを使用して、EB と関連付けられている CoreTile の仮想実装上でソフトウェアアプリケーションを実行することができます。EB FVP は、ARM アプリケーションプロセッサのモデルです。

本書の構成

本書は以下の章から構成されています。

第 章 1 はじめに

本章では、*エミュレーションベースボード*(EB) *固定仮想プラットフォーム*(FVP)について説明します。

第 章 2 EB FVP の使い方

この章では、EB FVP の起動と設定、およびモデル上でのソフトウェアアプリケーションの実行の手順について説明します。この手順は、使用している ARM ソフトウェアツールによって異なります。

第 章 3 EB FVP に関するプログラマ用リファレンス

本章では、ペリフェラルおよびシステムコンポーネントモデルのメモリマップとコンフィギュレーションレジスタについて説明します。

表記規則

italic
: 重要用語、相互参照、引用箇所を示します。

bold
: メニュー名などのユーザインタフェース要素を太字で記載しています。また、必要に応じて記述リスト内の重要箇所、ARM プロセッサの信号名、重要用語、および専門用語にも太字を使用しています。

`monospace`
: コマンド、ファイル名、プログラム名、ソースコードなど、キーボードから入力可能なテキストを示しています。

`monospace`
: コマンドまたはオプションに使用可能な略語を示しています。コマンド名またはオプション名をすべて入力する代わりに、下線部分の文字だけを入力することができます。

`monospace italic`
: 引数が特定の値で置き換えられる場合のモノスペーステキストの引数を示しています。

`monospace bold`
: サンプルコード以外に使用される言語キーワードを示しています。

`<and>`
: コードまたはコードの一部のアセンブラ構文で置換可能な項が使用されている場合に、その項を囲みます。例えば、

```
MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
```

スモールキャピタル
: 「*ARM 用語集*」で定義されている専門的な意味を持つ用語について、本文中で使用されます。例えば、IMPLEMENTATION DEFINED、IMPLEMENTATION SPECIFIC、UNKNOWN、UNPREDICTABLE などです。

ご意見、ご感想

本製品に関するフィードバック

本製品についてのご意見やご提案がございましたら、以下の情報を添えて購入元までお寄せ下さい。

- 製品名
- 製品のリビジョンまたはバージョン
- 説明にはできるだけ多くの情報を含めて下さい。適宜、症状と診断手順も含めて下さい。

内容に関するフィードバック

内容に関するご意見につきましては、電子メールを *errata@arm.com* まで送信して下さい。その際には、以下の内容を記載して下さい。

- タイトル
- 文書番号（ARM DUI0424OJ）。
- 問題のあるページ番号
- 問題点の簡潔な説明

また、補足すべき点や改善すべき点についての全般的なご提案もお待ちしております。

—————— **注** ——————

ARM では、この PDF を Adobe Acrobat および Acrobat Reader でのみテストしており、その他の PDF リーダーを使用した場合の表示品質については、保証いたしかねます。

————————————

第 1 章
はじめに

本章では、*エミュレーションベースボード*(EB) *固定仮想プラットフォーム*(FVP)について説明します。

以下のセクションから構成されています。

- *1.1 システムモデルの概念*(1-10 ページ).
- *1.2 EB FVP の概念*(1-11 ページ).
- *1.3 EB FVP の構成*(1-12 ページ).

1.1 システムモデルの概念

このセクションでは、*固定仮想プラットフォーム*(FVP)および *Programmer's View*(PV)モデルについて説明します。

FVP を使用すると、実際のハードウェアを使用せずにソフトウェア開発を行うことができます。

ソフトウェアでは、プロセッサおよびデバイスの PV モデルが使用できます。モデルは、実際のハードウェアと同じ機能動作を行います。

PV モデルは、絶対的なタイミング精度を犠牲にすることで、シミュレーション実行速度の高速化を達成します。PV モデルを使用してソフトウェアの機能を確認することはできますが、サイクルカウント、低レベルのコンポーネントインタラクション、またはその他のハードウェア固有の動作については確認できません。

1.2 EB FVP の概念

このセクションでは、*エミュレーションベースボード*(EB)、CoreTiles、および EB *固定仮想プラットフォーム*(FVP)について説明します。

EB と CoreTiles は、ARM® が開発したハードウェア開発プラットフォームです。

EB FVP は、システムのソフトウェアモデルです。ARM は、Fast Models ソフトウェアを使用して開発しています。

注

ARM では、EB FVP をプラットフォーム実装の例として提供していますが、特定の EB ハードウェアリビジョンを厳密に示していません。FVP は、本書に記載されているとおり、選択したペリフェラルをサポートします。提供される FVP は、EB ハードウェアと同じオペレーティングシステムイメージをブートするための完全性と精度を十分に備えています。

関連参照

1.3 EB FVP の構成(1-12 ページ).

1.3 EB FVP の構成

このセクションでは、EB リファレンスシステムの固定仮想プラットフォームについて説明します。

- プロセッサ CoreTile:
  — Cortex®-A8
- EB モデルは次のものを使用します。
  — 64MB フラッシュメモリ
  — 256MB RAM
  — Ethernet インタフェース
  — UART インタフェース
  — デバック DIP スイッチと LED
  — *リアルタイムクロック*(RTC)
  — タイムオブデイヤークロック
  — プログラム可能クロックジェネレータ
  — *同期シリアルポートインタフェース*(SSPI)
  — DMA コントローラコンフィギュレーションレジスタ
  — *スタティックメモリコントローラ*(SMC)。

EB FVP には、次の仮想コンポーネントも含まれています。

- *カラー LCD*(CLCD)ディスプレイ、キーボード、マウスの視覚化
- タッチスクリーンコントローラ
- 4 つの telnet 端末

EB リファレンスシステムの FVP は、以下のものから構成される階層モデルです。

- モデルのトップレベルビュー
- EB モデル
- システムモデルで使用されている CoreTile モデル

EB FVP は、ソフトウェア実行のために正確に機能するモデルを提供します。ただし、シミュレーション速度を上げるためにはタイミング精度が犠牲になります。実際のハードウェアとの主な相違点は次のとおりです。

- タイミングが概算である。
- バスが簡素化されている。

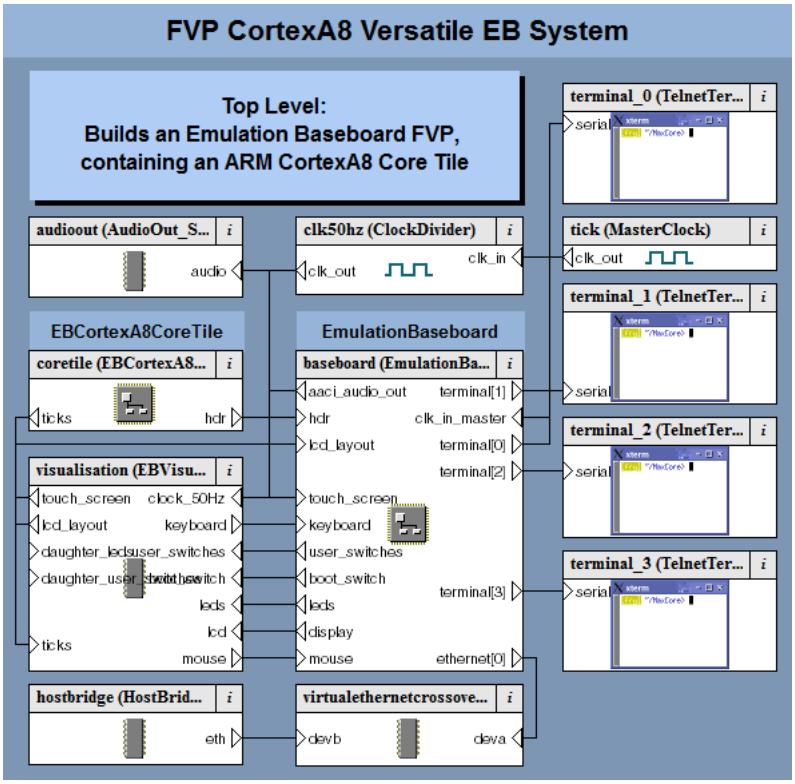

図 1-1 Cortex-A8 CoreTile を含むトップレベル EB モデルのブロック図

CoreTile コンポーネントは、プロセッサのバージョンと関連ポートを提供し、他のトップレベルのコンポーネントとの相互接続を可能にします。

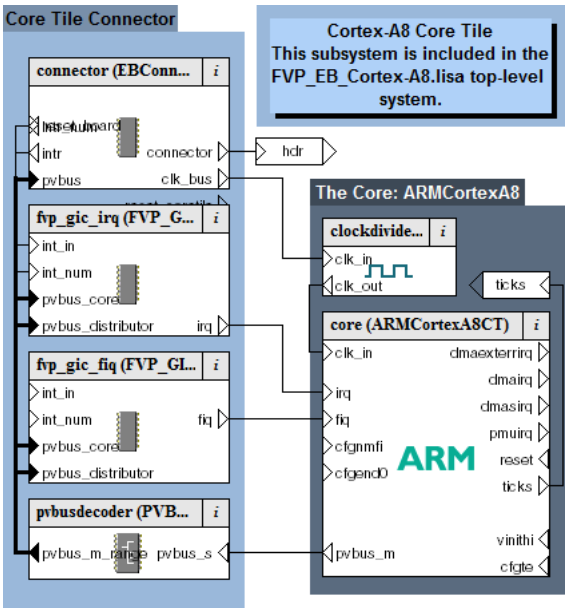

図 1-2 Cortex-A8 CoreTile モデルのブロック図

EB ベースボードモデルのブロック図は、ブロック形式、ポート、および相互接続におけるコンポーネントを示します。

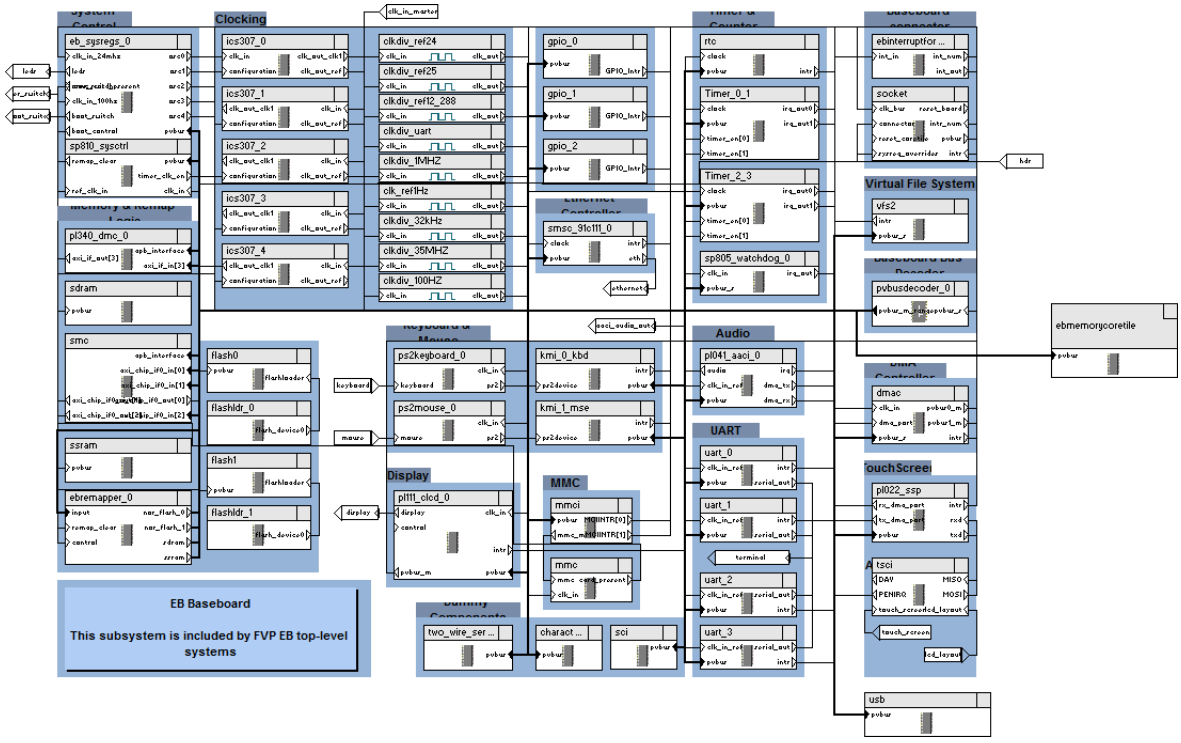

図 1-3 EB ベースボードモデルのブロック図

関連参照

3.2 EB モデルコンフィギュレーションのパラメータ(3-33 ページ).
3.3 EB と CoreTile ハードウェアとモデルの相違点(3-39 ページ).

第 2 章
EB FVP の使い方

この章では、EB FVP の起動と設定、およびモデル上でのソフトウェアアプリケーションの実行の手順について説明します。この手順は、使用している ARM ソフトウェアツールによって異なります。

以下のセクションから構成されています。

- *2.1 EB FVP の起動*(2-16 ページ).
- *2.2 EB FVP の設定*(2-17 ページ).
- *2.3 EB FVP CLCD ウィンドウ*(2-18 ページ).
- *2.4 EB FVP での Ethernet の使用*(2-21 ページ).
- *2.5 システムモデルによる端末の使用*(2-23 ページ).
- *2.6 仮想ファイルシステム*(2-25 ページ).
- *2.7 ビルド済みの FVP で VFS を使用するためのパス名*(2-27 ページ).

2.1 EB FVP の起動

このセクションでは、EB FVP の起動方法について説明します。

FVP を起動するには、モデルファイルが存在するディレクトリを変更し、コマンドプロンプトで以下を入力します。

```
model_name [--cadi-server] [--config-file filename] [-C instance.parameter=value] [--application app_filename]
```

`model_name`
: モデルファイルの名前。デフォルトでは、このファイル名は、`FVP_EB_processor` です。

`--cadi-server`
: ARM DS-5 デバッガなどの CADI 対応デバッガを実行中のモデルに接続できるように、CADI デバッグサーバを起動するオプション。起動するまでデバッガ接続を待機するように設定することができます。

`filename`
: オプションのプレーンテキストコンフィギュレーションファイルの名前。コンフィギュレーションファイルは複数パラメータの管理を簡略化します。

`instance.parameter=value`
: コンフィギュレーションパラメータのオプションの直接設定。

`app_filename`
: 起動時にロードするモデルのイメージのファイル名。

モデルを起動すると FVP CLCD ディスプレイが開きます。

関連参照

2.2 EB FVP の設定(2-17 ページ).
2.3 EB FVP CLCD ウィンドウ(2-18 ページ).

関連情報

モデルオプション、*Model Shell for Fast Models リファレンスマニュアル*.

2.2 EB FVP の設定

このセクションでは、EB FVP の初期状態をコマンドラインで設定する方法について説明します。

コンフィギュレーションファイルの使用

コマンドラインで、`--config-file` オプションを使用し、コンフィギュレーションファイルで FVP を起動します。

オプションのプレーンテキストコンフィギュレーションファイルのコメント行は `#` 文字で始めます。

コンフィギュレーションファイルのアクティブな各行には以下が含まれます。

- コンポーネントインスタンスの名前
- パラメータ名とその値

  ブール値は、`true`/`false` または `1`/`0` のいずれかを使用して設定します。文字列は、ホワイトスペースを含む場合は二重引用符で囲みます。

代表的なコンフィギュレーションファイル

```
# Disable semihosting using true/false syntax
coretile.core.semihosting-enable=false
#
# Enable the boot switch using 1/0 syntax
baseboard.sp810_sysctrl.use_s8=1
#
# Set the boot switch position
baseboard.eb_sysregs_0.boot_switch_value=1
```

このファイルは、以下のコマンドラインでモデルを起動するのと同じ効果があります。

```
-C coretile.core.semihosting-enable=false
-C baseboard.sp810_sysctrl.use_s8=1
-C baseboard.eb_sysregs_0.boot_switch_value=1
```

コマンドラインの使用

`-C` スイッチを使用すると、モデルの起動時にモデルパラメータを定義できます（`--parameter` は `-C` スイッチと同じ意味です）。コンフィギュレーションファイルについても同じ構文を使用しますが、各パラメータの前に `-C` スイッチを付けます。

ブートオプションを使用したフラッシュイメージからのモデルのブート

EB FVP の設定の例

```
# Boot from a flash image
FVP_EB_Cortex-A8 \
    --parameter "baseboard.flashldr_0.fname=flash.bin" \
    --parameter "baseboard.eb_sysregs_0.user_switches_value=4" \
    --parameter "visualisation.disable_visualisation=false" \
    --parameter "visualisation.rate_limit-enable=0" \
```

関連参照

2.1 EB FVP の起動(2-16 ページ).
3.2 EB モデルコンフィギュレーションのパラメータ(3-33 ページ).

2.3 EB FVP CLCD ウィンドウ

このセクションでは、EB FVP の CLCD ウィンドウについて説明します。

FVP が起動すると、FVP CLCD ウィンドウが開きます。このウィンドウに、シミュレートされたカラー LCD フレームバッファの内容が表示されます。CLCD ペリフェラルレジスタで設定された水平および垂直解像度に合致するよう自動的にサイズが変更されます。

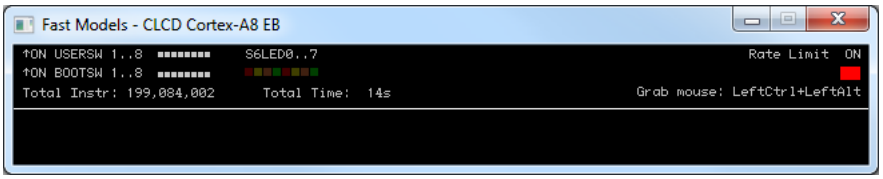

図 2-1 起動時の CLCD ウィンドウ

CLCD ウィンドウの最上部には、このステータス情報が表示されます。

USERSW

8 個の白いボックスは EB ユーザの DIP スイッチの状態を示します。

これらは EB ハードウェアのスイッチ S6、USERSW[8:1] を表します。これは、アドレス `0x10000004` の SYS_SW レジスタのビット [7:0] にマップされます。

デフォルトではスイッチの位置はオフです。状態を変更するには、白いボックスの上または下の領域をクリックします。

BOOTSW

8 個の白いボックスは EB のブート DIP スイッチの状態を示します。

これらは EB ハードウェアのスイッチ S8、BOOTSEL[8:1] を表します。これは、アドレス `0x100000004` の SYS_SW レジスタのビット [15:8] にマップされます。

デフォルトではスイッチの位置はオフです。

注

- ARM は、CLCD インタフェースを使用するのではなく、`boot_switch` モデルパラメータを使用してブート DIP スイッチを設定することを推奨します。
- モデルの実行中にブート DIP スイッチの位置を変更すると、予想できない結果になる可能性があります。

S6LED

8 個の色付きボックスは EB ユーザの LED の状態を示します。

これらは EB ハードウェアの LED D[21:14] を表します。これらは、アドレス `0x10000008` の SYS_LED レジスタのビット [7:0] にマップされます。ボックスは EB ハードウェアの赤/黄/緑の LED に対応します。

Total Instr

実行された命令の合計数を示すカウンタ。
FVP モデルはプログラマの視点からシステムを見るため、CLCD には合計コアサイクル数ではなく、合計命令数が表示されます。タイミングは、以下の理由でハードウェアごとに大きく変わる可能性があります。

- バスファブリックが簡略化されている。
- メモリレイテンシが最小化されている。
- プログラマビューコアとペリフェラルモデルが使用されている。

一般的に、モデル内での操作のタイミングは正確ではありません。

合計時間

合計経過時間（単位は秒）を示すカウンタ。

これは実時間であり、シミュレーション時間ではありません。

Rate Limit

シミュレーションが実時間よりも早く実行されないようにする機能。

システムモデルが高度に最適化されているため、実際のハードウェア以上に高速でコードが実行される可能性があります。この違いによりタイミングの問題が発生する場合があります。

デフォルトでは、レートリミットは有効になっています。実際の時間に近くなるよう、シミュレーション時間が制限されます。

レートリミットを無効または有効にするには、四角のボタンをクリックします。レートリミットが無効になるとテキストはオンからオフに変わり、色付きボックスは暗くなります。

注

モデルをインスタンス化する際に、 `rate_limit-enable` パラメータを使用してレートリミットを有効にするかどうか制御できます。

CLCD で **Total Instr** または **Total Time** の項目をクリックすると、ディスプレイは **Inst/sec**（1 秒当たりの命令）と **Perf Index**（パフォーマンスインデックス）を表示するようになります。項目上でもう一度クリックすると、これらの表示が切り替わります。

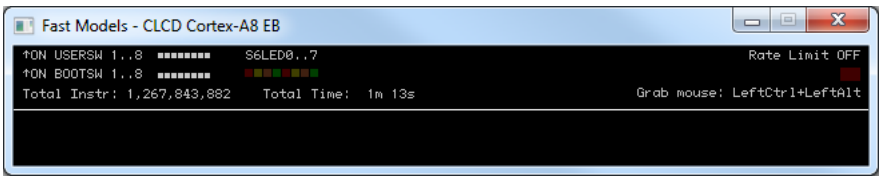

図 2-2 レートリミットをオフにした CLCD ウィンドウ

Instr/sec

実時間の 1 秒当たりに実行される命令数を示します。

Perf Index

シミュレーション時間に対する実際の時間の比率。比率が大きいほど、シミュレーションの実行が速くなります。レートリミット機能を有効にすると、Perf Index は 1 に近づきます。

モデルをリセットするとシミュレーションカウンタをリセットできます。

CLCD ウィンドウにフォーカスがある場合：

- すべてのキーボード入力は PS/2 キーボードデータに変換されます。
- ウィンドウ内でのすべてのマウス動作は PS/2 相対マウスモーションデータに変換されます。このデータはその後 KMI ペリフェラルモデル FIFO に渡されます。

注

シミュレータは相対マウスモーションイベントだけをモデルに送信します。結果として、ホストのマウスポインタがターゲット OS のマウスポインタと一致する必要はありません。

左側 Ctrl+左側 Alt キーを押すと、ホストのマウスポインタを非表示にできます。キーをもう一度押すと、ホストのマウスポインタが再表示されます。**左側 Ctrl** キーだけが機能します。キーパッドの右側の**右側 Ctrl** キーを押しても同じ効果はありません。

別のキーを使用する場合は、`trap_key` コンフィギュレーションオプションを使用します。

関連参照

スイッチ S6（3-34 ページ）.

スイッチ S8 (3-34 ページ).
3.3.9 タイミングの注意事項(3-42 ページ).
3.2.7 視覚化パラメータ(3-37 ページ).

関連情報

CLCD モデルコンポーネント、Fast Models リファレンスマニュアル.

2.4 EB FVP での Ethernet の使用

このセクションでは、EB FVP で Ethernet を使用する方法について説明します。

以下のサブセクションから構成されています。

- *2.4.1 ホストの要件*(2-21 ページ).
- *2.4.2 ターゲットの要件*(2-21 ページ).
- *2.4.3 Ethernet の設定*(2-22 ページ).

2.4.1 ホストの要件

EB FVP の Ethernet 機能を使用するには、ホストコンピュータをセットアップする必要があります。

関連情報

Fast Models ユーザガイド.

2.4.2 ターゲットの要件

EB FVP には仮想 Ethernet コンポーネントがあります。

SMSC91C111 Ethernet コントローラコンポーネント

このモデルの SMSC91C111 Ethernet コントローラは、TAP デバイスを使用してネットワークと通信します。

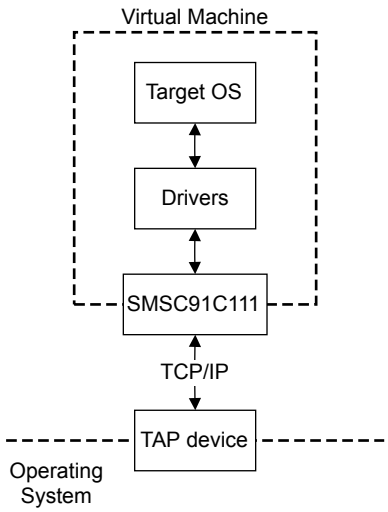

図 2-3 モデルネットワーク構造のブロック図

HostBridge コンポーネントは TAP デバイスの読み出しと書き込み操作を行うよう設定します。HostBridge コンポーネントは仮想のプログラマビューモデルです。これはネットワークゲートウェイのように動作し、ホスト上で Ethernet パケットを TAP デバイスと交換して、パケットを NIC モデルに転送します。

デフォルトでは、Ethernet コンポーネントは無効になっています。ターゲット OS にこのデバイス用のドライバが必要です。SMSC チップを使用するようにカーネルを設定します。Linux では SMSC91C111 をサポートしています。

`enabled`

`enabled` パラメータのデフォルト状態は `false` です。デバイスを無効にすると、カーネルはデバイスを検出できません。

`mac_address`

`mac_address` には 2 つのオプションがあります。MAC アドレスを指定しない場合、シミュレータが実行されるときに、Ethernet デバイスのデフォルトの MAC アドレスを使用します。これにより、ローカルネットワーク上の複数のホストでモデルを実行する場合に、このアドレスは一意性の可能性を高めるためにランダムに生成されます。

`promiscuous`

デフォルトの状態は `true` です。これは、指定されていないデバイス向けのものも含めて、すべてのネットワークトラフィックを受信することを意味します。複数の MAC アドレスに 1 つのネットワークデバイスを使用する場合は、デフォルトを使用します。例えば、ホスト OS と EB FVP Ethernet コンポーネント間で同じネットワークカードを使用する場合は、これを使用して下さい。

関連参照

3.2.3 Ethernet のパラメータ(3-35 ページ).

関連情報

SMSC_91C111 コンポーネント、*Fast Models リファレンスマニュアル*.

2.4.3 Ethernet の設定

Microsoft Windows または Linux から FVP 上で Ethernet インタフェースへの接続を設定できます。

関連情報

Fast Models ユーザガイド.

2.5 システムモデルによる端末の使用

このセクションでは、端末コンポーネントを使用する方法について説明します。この仮想コンポーネントは、ホスト上の TCP/IP ソケットとターゲット上のシリアルポート間で UART データを転送できるようにします。

以下のサブセクションから構成されています。

- *2.5.1 システムモデルの端末コンポーネント*(2-23 ページ).
- *2.5.2 Microsoft Windows 7 での Telnet のインストール*(2-24 ページ).

2.5.1 システムモデルの端末コンポーネント

TelnetTerminal ブロックは、端末コンポーネントパラメータを定義するときに設定します。仮想マシンは EB FVP です。

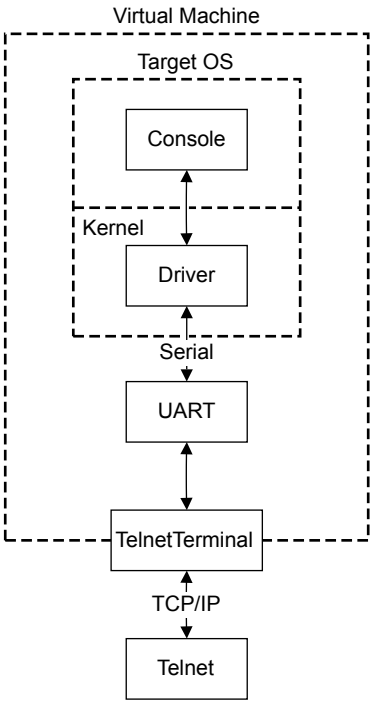

図 2-4 端末ブロック図

ターゲット側では、ターゲット OS により呼び出されるコンソールプロセスは、適切なドライバの存在に依存します。そのようなドライバは通常、OS カーネルの一部です。ドライバは UART 経由でシリアルデータを渡します。データは TelnetTerminal コンポーネントに転送され、FVP の外部の世界に TCP/IP ポートを公開します。このポートは、例えば、ホスト上の telnet プロセスによって接続できます。

デフォルトでは、モデルが初期化されるときに、EB FVP は 4 つの telnet 端末を起動します。4 つの端末のそれぞれの起動方法を変更するには、対応するコンポーネントパラメータを変更します。

例えば、クライアントの Telnet セッションが閉じられるなどで端末接続が切れた場合、ホスト上のポートが開きます。元のポート番号が使用できない場合は、このポートのポート番号が変わってしまう可能性があります。最初にデータにアクセスする前に、選択したクライアントをネットワークソケットに接続できます。最初にデータにアクセスする際に既存の接続がなく、`start_telnet` パラメータが `true` の場合、ホストの Telnet セッションは自動的に起動します。

端末のポート番号は、FVP の起動時に定義できます。各端末が使用するポートの実際の値は起動または再起動時に宣言されます。そのポートが既に使用されている場合は、指定した値にならない可能性があります。ポート番号は、モデルを起動したホストのウィンドウに表示されます。

端末コンポーネントは、telnet モードまたは raw モードで起動できます。

Telnet

Telnet モードの端末コンポーネントは、RFC 854 プロトコルのサブセットをサポートします。この状態では、ホストとクライアントの間でサポートされているもの、サポートされていないものに関するネゴシエーションに端末が加わることを意味しますが、フロー制御は実装されていません。

Raw

Raw モードを使用すると、ホストとターゲット間でバイトストリームを変更せずに渡すことが可能になります。この状態では、ホストとクライアント間の初期の機能のネゴシエーションに端末が加わらないことを意味します。TCP/IP ポートのように動作します。この機能を使用して、端末経由でターゲットに直接接続することができます。

2.5.2 Microsoft Windows 7 での Telnet のインストール

Microsoft Windows 7 クライアントで端末コンポーネントを使用するには、まず Telnet をインストールします。

デフォルトでは、Telnet アプリケーションは Microsoft Windows 7 にはインストールされていません。

Microsoft の Web サイトから次の手順でアプリケーションをダウンロードします。「Windows 7 Telnet」を検索して、Telnet FAQ ページを見つけます。Telnet をインストールするには、以下の手順に従います。

1. **[スタート] [コントロールパネル] [プログラムと機能]**の順に選択します。プログラムのアンインストールまたは変更を有効にできるウィンドウが開きます。
2. バーの左側の**[Windows の機能の有効化または無効化]**を選択します。これにより[Microsoft Windows 機能]ダイアログが開きます。**[Telnet クライアント]**チェックボックスを選択します。
3. **[OK]**をクリックします。Telnet のインストールの完了に数分かかる場合があります。

2.6 仮想ファイルシステム

仮想ファイルシステム(VFS)を使用すると、ターゲットは、ターゲット OS 固有のドライバと MessageBox というメモリマップされたデバイスを経由してホストファイルシステムの部分にアクセスできます。

VFS を使用している場合、ホストファイルシステムへのアクセスは共有ネットワークドライブへのアクセスと似ています。同じように動作することが予想できます。

VFS ファイルシステム動作

getattr
ファイル、ディレクトリ、またはシンボリックリンクのメタデータを取得します。

mkdir
ディレクトリを作成します。

remove
ファイル、ディレクトリ、またはシンボリックリンクを削除します。

rename
ファイル、ディレクトリ、またはシンボリックリンクの名前を変更します。

rmdir
空のディレクトリを削除します。

setattr
ファイル、ディレクトリ、またはシンボリックリンクのメタデータをセットします。

—————— 注 ——————

`setattr` は実装されていません。

———————————

VFS ではシンボリックリンクはサポートしていません。モデルではハードリンクを作成できませんが、ホストのオペレーティングシステムで作成されたハードリンクは正常に動作します。

VFS マウントポイント

closemounts
`openmounts` から戻された反復子ハンドルを解放します。

openmounts
使用可能なマウントのリストに対する反復子ハンドルを取得します。

readmounts
マウント反復子 ID からエントリを 1 つ読み込みます。

VFS では、以下のディレクトリ反復子がサポートされています。

closedir
`opendir` で取得されたディレクトリ反復子ハンドルを解放します。

opendir
指定されたディレクトリに対する反復子ハンドルを取得します。

readdir
ディレクトリ反復子から次のエントリを読み込みます。

—————— 注 ——————

日付スタンプは、1970 年 1 月 1 日 00:00 UTC の VFS 時点からの時間をミリ秒で表したもので、ホストの日付スタンプです。

———————————

VFS ファイル操作

closefile
`openfile` で開かれたハンドルを解放します。

filesync
すべてのファイルデータを永久ストレージにフラッシュするようホスト OS を強制します。
getfilesize
ファイルのサイズを返します（バイト単位）。
openfile
ファイルのハンドルを返します。
readfile
ファイルからデータブロックを読み取ります。
setfilesize
ファイルのサイズを、切り捨てるかまたはゼロを付けて拡大し、バイト単位で設定します。
writefile
ファイルにデータブロックを書き込みます。

関連参照

2.7 ビルド済みの FVP で VFS を使用するためのパス名（2-27 ページ）.

関連情報

VFS2 コンポーネント、*Fast Models リファレンスマニュアル*.
WritingADriver.txt、*%PVLIB_HOME%\VFS\docs*.

2.7 ビルド済みの FVP で VFS を使用するためのパス名

EB FVP の VFS 機能を使用するには、モデルの起動時に `baseboard.vfs2.mount` コンフィギュレーションパラメータを設定します。

パラメータを設定して、モデルがアクセスするホストファイルシステムディレクトリのパスを固定します。すべてのパス名は、次の完全修飾パスでなければなりません。

```
mountpoint:/path/to/object
```

付属の EB FVP には必要な VFS コンポーネントが含まれています。

関連参照

2.6 仮想ファイルシステム(2-25 ページ).

関連情報

Fast Models リファレンスマニュアル.

第 3 章
EB FVP に関するプログラマ用リファレンス

本章では、ペリフェラルおよびシステムコンポーネントモデルのメモリマップとコンフィギュレーションレジスタについて説明します。

以下のセクションから構成されています。

- *3.1 EB モデルメモリマップ*(3-29 ページ).
- *3.2 EB モデルコンフィギュレーションのパラメータ*(3-33 ページ).
- *3.3 EB と CoreTile ハードウェアとモデルの相違点*(3-39 ページ).

3.1 EB モデルメモリマップ

このセクションでは、EB 固定仮想プラットフォームで使用されるメモリ、ペリフェラル、コントローラの場所と割り込みについて説明します。

メモリの EB FVP 実装では、メモリコントローラを正しい値でプログラミングする必要はありません。ハードウェアでアプリケーションエラーを回避するには、メモリコントローラを正しく設定して下さい。

表 3-1 標準ペリフェラルのメモリマップと割り込み

ペリフェラル	モデリング	アドレス範囲	バス	サイズ	GIC Int[a]	DCCI Int[b]
ダイナミックメモリ	可	0x00000000 – 0x0FFFFFFF	AHB	256MB	-	-
システムレジスタ	可	0x10000000 – 0x10000FFF	APB	4KB	-	-
SP810 システムコントローラ	可	0x10001000 – 0x10001FFF	APB	4KB	-	-
TwoWire シリアルバスインタフェース	なし	0x10002000 – 0x10002FFF	APB	4KB	-	-
予約	-	0x10003000 – 0x10003FFF	APB	4KB	-	-
PL041 *アドバンストオーディオ CODEC インタフェース*(AACI)	部分的	0x10004000 – 0x10004FFF	APB	4KB	51	51
PL180 *マルチメディアカードインタフェース*(MCI)	部分的 [c]	0x10005000 – 0x10005FFF	APB	4KB	49, 50	49, 50
キーボード/マウスインタフェース 0	可	0x10006000 – 0x10006FFF	APB	4KB	52	7
キーボード/マウスインタフェース 1	可	0x10007000 – 0x10007FFF	APB	4KB	53	8
文字 LCD インタフェース	なし	0x10008000 – 0x10008FFF	APB	4KB	55	55
UART 0 インタフェース	可	0x10009000 – 0x10009FFF	APB	4KB	44	4
UART 1 インタフェース	可	0x1000A000 – 0x1000AFFF	APB	4KB	45	5
UART 2 インタフェース	可	0x1000B000 – 0x1000BFFF	APB	4KB	46	46
UART 3 インタフェース	可	0x1000C000 – 0x1000CFFF	APB	4KB	47	47
同期シリアルポートインタフェース	可	0x1000D000 – 0x1000DFFF	APB	4KB	43	43
スマートカードインタフェース	なし	0x1000E000 – 0x1000EFFF	APB	4KB	62	62

[a] ペリフェラル(SPI)の割り込み番号に 32 を追加して、GIC に表示される割り込み番号を形成します。GIC 割り込みの 0 ～ 31 は内部使用されます。
[b] DCCI システムで使用される割り込み番号。
[c] PL180 の実装は制限されているので、すべての機能が存在するわけではありません。

表 3-1 標準ペリフェラルのメモリマップと割り込み（続き）

ペリフェラル	モデリング	アドレス範囲	バス	サイズ	GIC Int[a]	DCCI Int[b]
予約	-	0x1000F000 – 0x1000FFFF	APB	4KB	-	-
SP805 ウォッチドッグインターフェイス	可	0x10010000 – 0x10010FFF	APB	4KB	32	32
SP804 タイマモジュール 0 および 1 インタフェース（タイマ 1 は 0x10011020 で開始）	可	0x10011000 – 0x10011FFF	APB	4KB	36	1
SP804 タイマモジュール 2 および 3 インタフェース（タイマ 3 は 0x10012020 で開始）	可	0x10012000 – 0x10012FFF	APB	4KB	37	2
PL061 GPIO インタフェース 0	可	0x10013000 – 0x10013FFF	APB	4KB	38	38
PL061 GPIO インタフェース 1	可	0x10014000 – 0x10014FFF	APB	4KB	39	39
PL061 GPIO インタフェース 2（さまざまなオンボード I/O）	可	0x10015000 – 0x10015FFF	APB	4KB	40	40
予約	-	0x10016000 – 0x10016FFF	APB	4KB	-	-
PL030 リアルタイムクロックインタフェース	可	0x10017000 – 0x10017FFF	APB	4KB	-	-
ダイナミックメモリコントローラコンフィギュレーション	部分的	0x10018000 – 0x10018FFF	APB	4KB	-	-
PCI コントローラコンフィギュレーションレジスタ	なし	0x10019000 – 0x10019FFF	AHB	4KB	-	-
予約	-	0x1001A000 – 0x1001FFFF	APB	24KB	-	-
PL111 カラー LCD コントローラ	可	0x10020000 – 0x1002FFFF	AHB	64 KB	55	55
DMA コントローラコンフィギュレーションレジスタ	可	0x10030000 – 0x1003FFFF	AHB	64 KB	-	-
汎用割り込みコントローラ 1（CPU インタフェース）	あり[d]	0x10040000 – 0x10040FFF	AHB	4KB	-	-
汎用割り込みコントローラ 1（ディストリビュータインタフェース）	可	0x10041000 – 0x10041FFF	AHB	4KB	-	-
汎用割り込みコントローラ 2［CPU インタフェース（タイル 1 用 nFIQ）］	なし[d]	0x10050000 – 0x10050FFF	AHB	4KB	-	-
汎用割り込みコントローラ 2（ディストリビュータインタフェース）	可	0x10051000 – 0x10051FFF	AHB	4KB	-	-
汎用割り込みコントローラ 3［CPU インタフェース（タイル 2 用 nIRQ）］	なし[d]	0x10060000 – 0x10060FFF	AHB	4KB	-	-

[a] ペリフェラル（SPI）の割り込み番号に 32 を追加して、GIC に表示される割り込み番号を形成します。GIC 割り込みの 0 ～ 31 は内部使用されます。
[b] DCCI システムで使用される割り込み番号。
[d] EB FVP GIC は、レジスタマップが異なるため、EB ハードウェア GIC と同じではありません。

表 3-1 標準ペリフェラルのメモリマップと割り込み（続き）

ペリフェラル	モデリング	アドレス範囲	バス	サイズ	GIC Int[a]	DCCI Int[b]
汎用割り込みコントローラ 3（ディストリビュータインタフェース）	なし	0x10061000 – 0x10061FFF	AHB	4KB	-	-
汎用割り込みコントローラ 4［CPU インタフェース（タイル 2 用 nFIQ）	なし[d]	0x10070000 – 0x10070FFF	AHB	4KB	-	-
汎用割り込みコントローラ 4（ディストリビュータインタフェース）	なし	0x10071000 – 0x10071FFF	AHB	4KB	-	-
PL350 スタティックメモリコントローラコンフィギュレーション[e]	可	0x10080000 – 0x1008FFFF	AHB	64 KB	-	-
予約	-	0x10090000 – 0x100EFFFF	AHB	448MB	-	-
デバッグアクセスポート（DAP）ROM テーブル[f]	なし	0x100F0000 – 0x100FFFFF	AHB	64 KB	-	-
予約	-	0x10100000 – 0x1FFFFFFF	-	255MB	-	-
予約	-	0x20000000 – 0x3FFFFFFF	-	512 MB	-	-
NOR フラッシュ	あり[g]	0x40000000 – 0x43FFFFFF	AXI	64MB	-	-
ディスクオンチップ	なし	0x44000000 – 0x47FFFFFF	AXI	64MB	41	41
SRAM	可	0x48000000 – 0x4BFFFFFF	AXI	64MB	-	-
コンフィギュレーションフラッシュ	なし	0x4C000000 – 0x4DFFFFFF	AXI	32MB	-	-
イーサネット	あり[h]	0x4E000000 – 0x4EFFFFFF	AXI	16MB	60	60
USB	なし	0x4F000000 – 0x4FFFFFFF	AXI	16MB	-	-
PISMO 拡張メモリ	なし	0x50000000 – 0x5FFFFFFF	AXI	256MB	58	58
PCI インタフェースバスウィンドウ	なし	0x60000000 – 0x6FFFFFFF	AXI	256MB	-	-
ダイナミックメモリ（ミラー）	可	0x70000000 – 0x7FFFFFFF	AXI	256MB	-	-
メモリタイル（第 2 CoreTile）	可	0x70000000 – 0x7FFFFFFF	AXI	0GB から 1GB へ	-	-

[a] ペリフェラル（SPI）の割り込み番号に 32 を追加して、GIC に表示される割り込み番号を形成します。GIC 割り込みの 0 ～ 31 は内部使用されます。
[b] DCCI システムで使用される割り込み番号。
[e] EB ハードウェアは PL093 スタティックメモリコントローラを使用しますが、モデルは PL350 を実装します。これらのコンポーネントは機能的には同じです。
[f] 一部のデバッガは、ターゲットプロセッサと DAP テーブルからのデバッグチェーン上の情報を読み出します。
[g] EB FVP では、IntelStrataFlashJ3 コンポーネントにこのペリフェラルが実装されています。
[h] EB FVP では、SC91C111 コンポーネントにこのペリフェラルが実装されています。

関連参照

3.3.2 ベースボードモデルに部分的に実装されている機能(3-39 ページ).

3.3.6 ステータスとシステムコントロールレジスタ(3-41 ページ).

3.3.7 汎用割り込みコントローラ(3-41 ページ).

関連情報

コントローラとペリフェラル、エミュレーションベースボードユーザガイド(リードフリー).

3.2 EB モデルコンフィギュレーションのパラメータ

このセクションでは、EB モデルコンフィギュレーションパラメータについて説明します。

以下のサブセクションから構成されています。

- *3.2.1 EB モデルコンフィギュレーションのパラメータについて*(3-33 ページ).
- *3.2.2 EB FVP ベースボードのパラメータ*(3-33 ページ).
- *3.2.3 Ethernet のパラメータ*(3-35 ページ).
- *3.2.4 システムコントローラのパラメータ*(3-35 ページ).
- *3.2.5 UART パラメータ*(3-36 ページ).
- *3.2.6 端末のパラメータ*(3-36 ページ).
- *3.2.7 視覚化パラメータ*(3-37 ページ).
- *3.2.8 FVP_EB_Cortex-A8 CoreTile パラメータ*(3-38 ページ).

3.2.1 EB モデルコンフィギュレーションのパラメータについて

このセクションでは、実行時に定義できるパラメータについて説明します。ビルド時間にのみ変更できるパラメータや、ハードウェアでユーザが通常変更することがないパラメータは、ここでは説明しません。

関連参照

3.1 EB モデルメモリマップ(3-29 ページ).

3.3 EB と CoreTile ハードウェアとモデルの相違点(3-39 ページ).

3.2.2 EB FVP ベースボードのパラメータ

ベースボードには、モデル起動時に変更できる、インスタンス生成時パラメータがあります。

コンフィギュレーションファイルで使用する構文は、次のとおりです。

```
baseboard.component_name.parameter=value
```

表 3-2 EB ベースボードモデルインスタンス化パラメータ

コンポーネント	パラメータ	型	使用できる値	デフォルト値	説明
eb_sysregs_0	user_switches_value	int	-	0	S6 設定の切り替え。
eb_sysregs_0	boot_switch_value	int	-	0	S8 設定の切り替え。
flashldr_0	fname	string	有効なパス	""	フラッシュイメージファイルへのパス。
	fnameWrite	string	有効なファイル名	""	フラッシュイメージファイルの名前。
flashldr_1	fname	string	有効なファイル名	""	フラッシュイメージファイルへのパス。
	fnameWrite	string	有効なファイル名	""	フラッシュイメージファイルの名前。
mmc	p_mmc_file	string	有効なファイル名	mmc.dat	マルチメディアカードのファイル名。
pl111_clcd_0	pixel_double_limit	int	-	0x12c	フレームバッファに送られるピクセルとなる水平ピクセルのしきい値は、両方の寸法のサイズが 2 倍になる。
sdram_size	sdram_size	int	0 ～ 0x40000000	0	2 番目にインストールされた CoreTile 上の SDRAM のサイズ。
sp805_watchdog_0	simhalt	bool	true、false	false	ARM ウォッチドッグモジュール (SP805) の有効化または無効化。

表 3-2 EB ベースボードモデルインスタンス化パラメータ（続き）

コンポーネント	パラメータ	型	使用できる値	デフォルト値	説明
sp810_sysctrl	use_s8	bool	true、false	false	boot_switches_value を読み出すかどうかを示す。
vfs2	mount	string	有効なファイル名	""	mount ディレクトリ名。

スイッチ S6

スイッチ S6 は、EB ハードウェアでブートモニタ設定スイッチに相当します。

標準 ARM ブートモニタフラッシュイメージをロードした場合は、スイッチ S6-1 の設定によりモデルのリセット時に発生するものが変更されます。それ以外の場合、スイッチ S6 の機能は実装に依存します。

スイッチの位置を S6 パラメータに書き込むには、スイッチ設定をバイナリから同等の整数値に変換します。ここでは、1 がオンで 0 がオフになります。

表 3-3 EB システムモデルスイッチ S6 のデフォルトの位置

スイッチ	デフォルトの位置	デフォルトの位置の関数
S6-1	オフ	ディスプレイプロンプトは、システムの起動後にブートモニタコマンドの入力を可能にします。
S6-2	オフ	「*STDIO 転送*」を参照して下さい。
S6-3	オフ	「*STDIO 転送*」を参照して下さい。
S6-4 ～ S6-8	オフ	アプリケーション使用のために予約されています。

S6-1 がオンの位置にある場合は、ブートモニタはフラッシュにロードされたブートスクリプトを実行します。スクリプトがない場合は、ブートモニタプロンプトが表示されます。

S6-2 と S6-3 の設定は、モデルリセットの STDIO のソースとデスティネーションに影響を与えます。

表 3-4 STDIO 転送

S6-2	S6-3	出力	入力	説明
オフ	オフ	UART0	UART0	STDIO はセミホスティングの I/O または UART を使用するかどうかを自動検出します。デバッガが接続されている場合は、STDIO はデバッガの出力ウィンドウに転送され、接続されていない場合は、STDIO は UART0 に移動します。
オフ	ON	UART0	UART0	セミホスティングの設定に関わらず、STDIO は UART0 に転送されます。
ON	オフ	CLCD	キーボード	セミホスティングの設定に関わらず、STDIO は CLCD とキーボードに転送されます。
ON	ON	CLCD	UART0	セミホスティングの設定に関わらず、STDIO 出力は LCD に転送され、入力はキーボートに転送されます。

関連情報

ブートモニタのコンフィギュレーションとコマンド、エミュレーションベースボードユーザガイド (リードフリー).

スイッチ S8

デフォルトでは、スイッチ S8 は無効になっています。これを有効にするには、モデルを起動する前に、パラメータ `baseboard.sp810_sysctrl.use_s8` の状態を `true` に変更します。

ブートモニタフラッシュイメージをロードした場合は、スイッチ S8 でブートメモリを再マップすることができます。

リセット時には、EB ハードウェアは `0x0` でコードを実行し始めます。これは、通常、揮発性の DRAM です。EB FVP CLCD に S8 スイッチを設定することによって、この場所に不揮発性 RAM の内容を指定することができます。設定は、モデルのリセット時に有効になります。

表 3-5 EB システムモデルスイッチ S8 の設定

スイッチ S8[4:1]	メモリ範囲	説明
0000	`0x40000000` – `0x43FFFFFF`	`0x0` に再マップされた NOR フラッシュ(`flash0`、IntelStrataFlashJ3)
0001	`0x44000000` – `0x47FFFFFF`	`0x0` に再マップされた NOR フラッシュ(`flash1`、IntelStrataFlashJ3)
0010	`0x48000000` – `0x4BFFFFFF`	`0x0` に再マップされた SRAM(`ssram`, RAMDevice)

関連参照

3.2.2 EB FVP ベースボードのパラメータ(3-33 ページ).

3.2.3 Ethernet のパラメータ

Ethernet コンポーネントには、変更できるインスタンス生成時パラメータがあります。

コンフィギュレーションファイルまたはコマンドラインで使用する構文は、次のとおりです。

```
baseboard.smsc_91c111_0.parameter=value
```

表 3-6 Ethernet のコンフィギュレーションパラメータ

Name	型	使用できる値	デフォルト値	説明
`enabled`	`bool`	`true`、`false`	`false`	ホストインタフェース接続有効。
`mac_address`	`string`	None、`auto`	`00:02:f7:ef:31:11`	ホスト/モデルの MAC アドレス。
`promiscuous`	`bool`	`true`、`false`	`true`	Ethernet コントローラをホスト OS と共有するなどの場合に、ホストを無差別モードに切り替えます。

`mac_address`

- MAC アドレスを指定しないと、シミュレータの実行時に、デフォルトの MAC アドレスが使用されたり変更されたりします。最後の 2 バイト[`00:02`]が、ホストワークステーション上にあるいずれかのアダプタの MAC アドレスの最後の 2 バイトに置き換えられます。これにより、ローカルネットワーク上の複数のホストでモデルを実行する場合に、MAC アドレスが一意になる可能性が高くなります。
- MAC アドレスを自動で指定した場合、シミュレータが実行されるたびに、ローカル MAC アドレスがランダムに生成されます。アドレスは、ローカル管理のユニキャスト MAC アドレスであることを示すため、最初のバイトでビット[1] が設定され、ビット[0] がクリアされます。

  注

  IP アドレスを割り当てるために DHCP サーバが使用されますが、提供されている MAC アドレスが使用されることがあるため、無作為の MAC アドレスを使用すると、DHCP サーバと競合することがあります。

3.2.4 システムコントローラのパラメータ

システムコントローラには、モデル起動時に変更できるインスタンス生成時パラメータがあります。

コンフィギュレーションファイルまたはコマンドラインで使用する構文は、次のとおりです。

```
baseboard.sp810_sysctrl.parameter=value
```

表 3-7 システムコントローラのコンフィギュレーションパラメータ

Name	型	使用できる値	デフォルト値	説明
sysid	int	0、[i]1、[j][k][l]2	0x00000000	システム識別レジスタの値。
use_s8	bool	true、false	false	スイッチ S8 を有効にするかどうかを選択。

3.2.5 UART パラメータ

UART には、変更できるインスタンス生成時パラメータがあります。

コンフィギュレーションファイルまたはコマンドラインで使用する構文は、次のとおりです。

```
baseboard.uart_x.parameter=value
```

ここで x は、UART 識別子 0、1、2 または 3 になります。

表 3-8 UART コンフィギュレーションパラメータ

Name	型	使用できる値	デフォルト値	説明
generic_uart	bool	true、false	false	SBSA 仕様で汎用 Uart として定義されているレジスタのサブセットにのみアクセスを許可します。
revision	string	-	"r1p4"	シミュレートするリビジョン（ID レジスタおよび FIFO 容量に影響します）。
shutdown_on_eot	bool	true、false	false	EOT（ASCII 4）char を送信したときのシャットダウンシミュレーション。
shutdown_tag	string	-	""	文字列を送信したときのシャットダウンシミュレーション。ランタイムパラメータ。
untimed_fifos	bool	true、false	true	クロックレートを無視し、シリアルデータをすぐに送受信します。

3.2.6 端末のパラメータ

端末には、モデル起動時に変更できる、インスタンス生成時パラメータがあります。

コンフィギュレーションファイルまたはコマンドラインで使用する構文は、次のとおりです。

```
terminal_x.parameter=value
```

ここで x は、端末 ID 0、1、2 または 3 になります。

表 3-9 端末インスタンス化パラメータ

Name	型	使用できる値	デフォルト値	説明
モード	string	telnet、raw	Telnet	端末動作モード。
quiet	bool	true、false	false	stdout および stderr への出力を抑制します。

[i] SYS_ID レジスタ値 = 0x0225f500 、REV_A に対応。
[j] SYS_ID レジスタ値 = 0x12257500 、REV_B に対応。
[k] SYS_ID レジスタ値 = 0x22252500 、REV_C に対応。
[l] その他の値の場合：SYS_ID レジスタ値 = 0x0 。

表 3-9 端末インスタンス化パラメータ（続き）

Name	型	使用できる値	デフォルト値	説明
start_telnet	bool	true、false	true	システム起動時に端末を有効にします。
start_port	int	有効なポート番号	5000	システム起動時に端末が使用するポート。指定されたポートがフリーでない場合、フリーなポートが見つかるまで、ポートの値が 1 ずつ増加します。

3.2.7 視覚化パラメータ

視覚化コンポーネントには、モデル起動時に変更できる、インスタンス生成時パラメータがあります。

コンフィギュレーションファイルで使用する構文は、次のとおりです。

```
visualisation.parameter=value
```

表 3-10 視覚化インスタンス化パラメータ

Name	型	使用できる値	デフォルト値	説明
cpu_name	string		""	ウィンドウのタイトルに表示される名前。
daughter_led_count	int	0-32	0	ドーターボードにある LED の数。
daughter_user_switch_count	int	0-32	0	ドーターボードにあるスイッチの数。
disable_visualisation	bool	true、false	false	モデル起動時に EBVisualisation コンポーネントを無効にします。
rate_limit-enable [m]	bool	true、false	true	できる限り高速で実行するのではなく、シミュレーション時間が実際の時間に近くなるように、シミュレーション速度を制限します。
trap_key	int	有効な ATKeyCode キー値 [n]	74[o]	**左側 Ctrl** キーで操作する、マウスポインタ表示を切り替えるためのトラップキー。
window_title	string		"Fast Models - CLCD %cpu%"	ウィンドウのタイトル（%cpu% は cpu_name に置き換えられます）。

関連参照

2.3 EB FVP CLCD ウィンドウ(2-18 ページ).

関連情報

EBVisualisation コンポーネント、Fast Models リファレンスマニュアル.

[m] インスタンス生成時にパラメータを設定する代わりに、CLCD で[Rate Limit]ボタンをクリックできます。
[n] Fast Models をインストールしている場合は、ATKeyCode 値のリストについて、ヘッダファイル `%PVLIB_HOME%\components\KeyCode.h` を参照して下さい。Linux の場合は、ファイル `$PVLIB_HOME/components/KeyCode.h` を参照してください。
[o] これは**左側 Alt** キーに相当します。

3.2.8 FVP_EB_Cortex-A8 CoreTile パラメータ

Cortex-A8 CoreTile FVP には、モデル起動時に変更できるパラメータがあります。

記載されているすべてのパラメータはインスタンス生成時パラメータです。この CoreTile FVP は、Cortex-A8 プロセッサの r2p1 に基づいています。

コンフィギュレーションファイルで使用する構文は、次のとおりです。

```
coretile.core.parameter=value
```

また、Cortex-A8 CoreTile FVP には GIC が含まれますが、インスタンス生成時に設定することはできません。

表 3-11 FVP_EB_Cortex-A8 CoreTile パラメータ

Name	型	使用できる値	デフォルト値	説明
semihosting-cmd_line	string	メモリ以外は制限なし	""	コマンドラインは、SVC セミホスティング呼び出しに使用できます。
semihosting-debug [p]	bool	true、false	false	SVC セミホスティング呼び出しのデバッグ出力を有効にします。
semihosting-enable	bool	true、false	true	セミホスティング SVC トラップを有効にします。
semihosting-ARM_SVC	int	24 ビット整数	0x123456	セミホスティングの ARM SVC 番号。
semihosting-Thumb_SVC	int	8 ビット整数	0xAB	セミホスティングの Thumb SVC 番号。
semihosting-heap_base	int	0x00000000 - 0xFFFFFFFF	0x0	ヒープベースの仮想アドレス
semihosting-heap_limit	int	0x00000000 - 0xFFFFFFFF	0x0F000000	ヒープトップの仮想アドレス。
semihosting-stack_base	int	0x00000000 - 0xFFFFFFFF	0x10000000	下降スタックの仮想ベースアドレス。
semihosting-stack_limit	int	0x00000000 - 0xFFFFFFFF	0x0F0000000	スタックリミットの仮想アドレス。

[p] 無視されます。

3.3 EB と CoreTile ハードウェアとモデルの相違点

このセクションでは、モデルに実装されていないか、実装に違いのあるエミュレーションベースボードと CoreTile ハードウェアの機能について説明します。

以下のサブセクションから構成されています。

- *3.3.1 ベースボードモデルにない機能*(3-39 ページ).
- *3.3.2 ベースボードモデルに部分的に実装されている機能*(3-39 ページ).
- *3.3.3 プロセッサモデルの制約*(3-39 ページ).
- *3.3.4 再マッピングと DRAM のエイリアス生成*(3-40 ページ).
- *3.3.5 ダイナミックメモリの特性*(3-41 ページ).
- *3.3.6 ステータスとシステムコントロールレジスタ*(3-41 ページ).
- *3.3.7 汎用割り込みコントローラ*(3-41 ページ).
- *3.3.8 GPIO2*(3-41 ページ).
- *3.3.9 タイミングの注意事項*(3-42 ページ).

3.3.1 ベースボードモデルにない機能

システムモデルには、エミュレーションベースボードのハードウェアのいくつかの機能が実装されていません。

- TwoWire シリアルバスインタフェース
- 文字 LCD インタフェース
- スマートカードインタフェース
- PCI コントローラコンフィギュレーションレジスタ
- デバッグアクセスポート
- ディスクオンチップ
- コンフィギュレーションフラッシュ
- USB
- PISMO 拡張メモリ
- PCI インタフェースバスウィンドウ
- UART モデムハンドシェークシグナル
- VGA のサポート

関連参照

3.1 EB モデルメモリマップ(3-29 ページ).

3.3.2 ベースボードモデルに部分的に実装されている機能

システムモデルには、エミュレーションベースボードのハードウェアのいくつかの機能が部分的に実装されています。

すなわち、一部のコンポーネントは存在しますが、機能が完全にはモデル化されていません。モデルリリースノートで、最新情報を参照してください。

サウンド

EB FVP は、PL041 AACI PrimeCell およびオーディオコーデックを EB ハードウェアと同様に実装しますが、サンプルレートの数に制限があります。

DMC

EB FVP でモデル化されていますが、*ダイナミックメモリコントローラ*(DMC)は、直接メモリアクセスをすべてのペリフェラルに提供するわけではありません。オーディオと同期シリアルポートインタフェースのコンポーネントだけが DMC を介してアクセスできます。

3.3.3 プロセッサモデルの制約

このセクションでは、プロセッサモデルの制限について説明します。

プロセッサモデルの一般的な制約

ARM プロセッサの FVP 実装には、いくつかの一般的な制限事項が適用されます。

- Fast Model ソフトウェアは、正確な命令のタイミングをモデル化しません。プロセッサは、シミュレーション時点で一連の命令(「「quantum」」)を発行し、次の quantum を実行するまで待機します。プロセッサは、クロック Tick あたりの 1 つの命令の平均を求めます。
  - モデルで実行するソフトウェアの体感的パフォーマンスは、実際のソフトウェアとは異なります。具体的には、メモリアクセスと算術演算にはすべて同じ時間がかかります。
  - 例えば、プログラムは高解像度タイマをポーリングすることによって、プロセッサの量子化された実行動作を検出することができます。
  - quantum のすべての命令は、事実上、アトミックです。この原子性により、ソフトウェアで一部の競合状態のバグがマスクされる可能性があります。
- モデルにはキャッシュ制御レジスタが含まれますが、通常、レジスタアクセス許可を確認できるだけです。キャッシュフラッシュ操作はサポートされていますが、何の影響も与えません。結果として、キャッシュのエイリアス生成問題のため、実際のハードウェア上では失敗する可能性のあるコードを EB FVP 上で問題なく実行できる場合があります。
- モデル上での VFP と NEON™ の命令セットの実行は、高性能ではありません。
- 書き込みバッファはモデリングされません。
- TLB 動作の大部分はモデルで実装されます。アーキテクチャ v7 モデルの場合、ステートフルキャッシュが有効になると、TLB メモリ属性設定が使用されます。
- MicroTLB は実装されません。
- 1 つのメモリアクセスポートが実装されます。このポートには、命令、データ、DMA、ペリフェラルのアクセスが組み込まれています。ペリフェラルのポートのメモリマップレジスタの構成は無視されます。
- すべてのメモリアクセスはアトミックであり、プログラマの表示順に実行されます。すべてのメモリトランザクションは、最大 32 ビット幅です。非境界整列アクセスは常にバイト転送として実行されます。
- 割り込みはすべての命令バウンダリで受け入れられません。
- semihosting-debug コンフィギュレーションパラメータは無視されます。
- 統合レジスタとテストレジスタは実装されません。
- 一部のプロセッサモデルでは、1 つの CP14 デバッグコプロセッサレジスタには CP14 DSCR が含まれます。レジスタは 0 を読み出し、書き込みを無視します。他の CP14 レジスタにアクセスすると、未定義命令例外が生成されます。FVP をデバッグするには、外部デバッガを使用します。
- モデルが直接的にサポートしているブレークポイントのタイプは次のとおりです。
  - 1 つのアドレスの無条件命令ブレークポイント
  - 1 つのアドレスの無条件データブレークポイント
  - 無条件命令のアドレス範囲のブレークポイント
- デバッガの擬似レジスタは、プロセッサ例外ブレークポイントをサポートしています。例外レジスタをゼロ以外の値に設定すると、関連した例外ベクタに対するエントリの実行が停止されます。
- モデルには、命令カウンタ以外の*性能管理ユニット*(PMU)は実装されていません。

関連情報

EB ハードウェアとシステムモデルの相違点、*Fast Models リファレンスマニュアル*.

FVP_EB_Cortex-A8 CoreTile の制約

Cortex-A8 プロセッサの固定仮想プラットフォーム実装には、追加制限が適用されます。

- モデルには 2 つの 4GB アドレス空間が表示されます。1 つはセキュアモードから表示され、もう 1 つは標準モードから表示されます。アドレス空間にはゼロウェイト状態メモリとペリフェラルが含まれていますが、空間の多くはマップされていません。
- PLE モデルは、純粋にレジスタベースで、実装動作がありません。
- MMU が無効の場合、非境界整列アクセスはデータアボートを発生させません。

3.3.4 再マッピングと DRAM のエイリアス生成

このセクションでは、ハードウェアとシステムモデルの相違点について説明します。

EB ハードウェアは、大幅なメモリ再マップ機能を提供します。

このブートの再マッピング時、一番下の 64MB の物理アドレスマップは以下のとおりです。

- NOR フラッシュ
- スタティック拡張メモリ

再マップ機能を提供することに加えて、ハードウェアはすべての 256MB システム DRAM を `0x70000000` でエイリアス化します。

再マッピングは、通常、システムモデルに適用しません。ただし、NOR フラッシュをモデル化し、再マップすることはできます。

メモリマップでは、モデルは、ペリフェラルまたはメモリによって明示的に占領されていないメモリ領域はマップしません。この簡素化により、実装されていないペリフェラルが理論的に占領する領域や、予約領域が含まれます。ホストプロセッサからこれらの領域にアクセスすると、モデルに警告が表示されます。

関連参照

スイッチ S8(3-34 ページ).

3.3.5 ダイナミックメモリの特性

このセクションでは、ハードウェアとシステムモデルの相違点について説明します。

エミュレーションベースボードのハードウェアには、PL340 DMC が含まれます。このコンポーネントにより、メモリマップのアドレス `0x10030000` にコンフィギュレーションインタフェースが表示されます。

システムモデルは、DRAM の汎用領域を設定し、PL340 をモデル化しません。この簡素化により、シミュレーションを高速化できます。

3.3.6 ステータスとシステムコントロールレジスタ

このセクションでは、ハードウェアとシステムモデルの相違点について説明します。

エミュレーションベースボードのハードウェアバージョンの場合、ステータスとシステムコントロールレジスタを使用すると、プロセッサでその環境を決定し、一部のオンボード操作を制御できます。

すべての EB システムレジスタは、オシレータのテストレジスタである SYS_TEST_OSC[4:0] 以外のシステムモデルに実装されています。実装されていないレジスタはメモリとして機能し、レジスタに書き出す値によってモデルの動作が変更することはありません。

ほとんどの EB FVP 機能は、起動時の設定によって決まります。

関連参照

2.2 EB FVP の設定(2-17 ページ).

3.3.7 汎用割り込みコントローラ

EB FVP に付属の*汎用割り込みコントローラ*(GIC)は、エミュレーションボードのファームウェアの GIC と本質的に異なります。

新しいデバイスのプログラマモデルは、概して下位互換性があります。モデル GIC は、PL390 PrimeCell を実装したものです。

関連情報

PrimeCell® Generic Interrupt Controller (PL390) テクニカルリファレンスマニュアル.

3.3.8 GPIO2

このセクションでは、ハードウェアとシステムモデルの相違点について説明します。

システムモデルは、GPIO を汎用 IO デバイスとして提供します。EB ハードウェアでは、GPIO2 は、USB、プッシュボタン、MCI ステータスシグナルを処理します。EB FVP では、USB と MCI は実装されていないため、プッシュボタンもモデル化されていません。

3.3.9 タイミングの注意事項

このセクションでは、ハードウェアとシステムモデルの相違点について説明します。

シミュレーション速度の高速化とタイミング精度との相対的なバランスのために、モデルが期待どおりに動作しない場合があります。

FVP は、機能的に正確なシミュレーションでソフトウェアアプリケーションを実行できる環境です。ただし、タイマやキーボードなどの実際のデバイスとコードがやり取りする場合、タイミングに差が生じることがあります。モデル化されたデバイスでは実環境データを実時間（ウォールクロック時間）で受信しますが、シミュレーション時間は実時間よりもかなり高速になります。したがって、キーを 1 回しか押していなくても複数回連続で押したように解釈されたり、マウスを 1 回しかクリックしていなくても 2 回クリックしたと誤って解釈されたりすることがあります。

EB FVP には、シミュレーション時間と実時間を一致させるレートリミット機能があります。CLCD ディスプレイの［Rate Limit］ボタンを使用するか、 `rate_limit-enable` モデルインスタンス化パラメータを使用してレートリミット機能を有効にすると、モデルは強制的に実時間で実行されます。この機能により、2 つのクロックが異なったレートで動作する問題を回避できます。インタラクティブなアプリケーションの場合は、レートリミットを有効にすることを推奨します。