Model Shell for Fast Models

バージョン 6.1

リファレンスマニュアル

ARM®

ARM DUI 0457GJ（ID ０２０７１２）

Model Shell for Fast Models
リファレンスマニュアル

リリース情報

変更履歴

説明	発行	機密保持ステータス	変更点
2008 年 12 月	A	非機密扱い	初版
2009 年 3 月	B	非機密扱い	Fast Models 4.2 改訂版
2009 年 4 月	C	非機密扱い	Fast Models 5.0 改訂版
2009 年 9 月	D	非機密扱い	Fast Models 5.1 改訂版
2010 年 2 月	E	非機密扱い	Fast Models 5.2 改訂版
2010 年 10 月	F	非機密扱い	Fast Models 6.0 初版
2011 年 5 月	G	非機密扱い	Fast Models 6.1 初版

著作権

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。

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.arm.com

目次
Model Shell for Fast Models リファレンスマニュアル

序章

本章では、*『Model Shell for Fast Models リファレンスマニュアル』*（本書）について概説します。以下のセクションから構成されています。

- *「本書について」*（v ページ）
- *「ご意見、ご感想」*（vii ページ）.

本書について

本書では、Fast Models に含まれている信号、クロック、バス、汎用ペリフェラル、およびプロセッサコンポーネントについて説明します。これらのンポーネントについては、プロセッサおよびペリフェラルコンポーネントに対する*プログラマの所見*（PV）があります。

対象読者

本書は、開発環境の一部として ARM® のコンポーネントを使用する ARM® ベース製品の開発を支援するために、経験豊かなハードウェアおよびソフトウェア開発者を読者を対象としています。

構成

本書は以下の章から構成されています。

第 1 章 ***はじめに***

Model Shell の概要を説明します。

第 2 章 ***Model Shell コマンド***

Model Shell のコマンドラインオプションの詳細を説明します。

表記規則

表記規則は次のとおりです。

italic	重要事項、重要用語、相互参照、引用箇所を斜体で記載しています。
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。
`mono`space	コマンドまたはオプションに使用可能な略語を示します。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。以下に例を示します。 `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

参考資料

ここでは、ARM およびサードパーティから入手できる関連出版物を示します。

ARM マニュアルの入手方法については、*ARM Infocenter*, http://infocenter.arm.com/help/index.jsp を参照して下さい。

ARM の出版物

本書では、本製品固有の情報を提供しています。その他の関連情報については、以下のマニュアルを参照して下さい。

- *『ARM サイクル精度デバッグインタフェース開発者ガイド』*（ARM DUI 0444）
- 『ARM アーキテクチャリファレンスマニュアル』, http://infocenter.arm.com/help/index.jsp
- *『ARM FLEXnet ライセンス管理ガイド』*（ARM DUI 0209）
- *『Fast Models ユーザガイド』*（ARM DUI 0370）

ご意見、ご感想

ARM では、本製品および本書に関するフィードバックをお待ちしております。

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。必要に応じて、症状を含めて下さい。

本書に関するフィードバック

本書の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 DUI 0457GJ）
- 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。

第 1 章
はじめに

本章では、CADI 準拠モデルを設定および実行するためのコマンドラインツールである Model Shell の主要な機能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。

- *「概要」*（1-2 ページ）
- *「統合シミュレータのターゲット」*（1-3 ページ）.

1.1 概要

Model Shell は、CADI 準拠モデルを設定および実行するためのコマンドラインツールです。

Model Shell を CADI 準拠モデルに接続すると、以下の機能が実行されます。

- stdio のセミホスティング
- CADI ログの記録
- デバッガ、プロファイラ、およびオペレーティング環境に適したプラットフォームの起動

以下の方法で、Model Shell は CADIServer を起動してその他のデバッガをモデルに接続することができます。

- シミュレーションは初期化されますが、実行はされません。外部デバッガは、そのシミュレーションを制御する必要があります（デフォルト）。
- シミュレーションは初期化され、即座に実行されます。外部デバッガは、起動後にシミュレーションに接続できます。

Model Shell には、標準ストリームについてのみ入出力をセミホスティングするプロセスが用意されています。

- CADIServer が起動すると、セミホスティングされている出力は Model Shell コンソールを経由してすべてのデバッガに移動します。
- デバッガが接続されていると、デバッガは入力のセミホスティングを実行します。接続されていない場合は、Model Shell が入力を提供します。

1.2 統合シミュレータのターゲット

Fast Models は、Model Shell をモデルの CADI ライブラリに静的にリンクすることによって、*統合シミュレータ*（isim）のターゲットを作成できます。詳細については、*『Fast Models ユーザガイド』*を参照して下さい。

`--model` を除いたすべての Model Shell コマンドラインオプションは、isim ターゲットと組み合わせて使用することができます。モデルはターゲットに統合されるため、モデルをコマンドラインで指定する必要はありません。

第 2 章
Model Shell コマンド

本章では、Model Shell の使用方法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。

- *「Model Shell オプション」*（2-2 ページ）
- *「文字列構文」*（2-6 ページ）
- *「モデルパラメータの定義」*（2-7 ページ）
- *「SMP のサポート」*（2-10 ページ）
- *「Model Shell の停止」*（2-11 ページ）
- *「model_shell および isim_system からのライセンス確認エラーの検出」*（2-13 ページ）.

2.1 Model Shell オプション

Model Shell のオプションを使用し、さまざまな用途に合わせて Model Shell の動作をカスタマイズすることができます。これらのオプションは、表 2-1 （2-3 ページ）にリストされています。

コマンドラインから Model Shell を起動するには、model_shell に続けて必要なオプションを入力します。

Model Shell を起動する際には、以下のすべての構文が使用可能です。

```
model_shell [options] model [application_list]

model_shell [options] -m model [application_list]

model_shell -m model [options] [application_list]

model_shell -m model -a app1.axf [options]
```

各項目には以下の意味があります。

options
: 表 2-1 （2-3 ページ）のコマンドラインオプションの一覧。

model
: モデルのファイル名（拡張子を含む）。
モデル名が以下に該当する場合、-m オプションは不要です。
- 1 つのコア、つまり、単一のコアまたは SMP コアを備えているシステムのモデル名である場合
- 行末にある場合
- 行の最後の項目に隣接しており後にはアプリケーションファイルの名前しかない場合

モデルファイルには、.so または .dll の拡張子が付きます。

application_list
: モデルにロードする単一のアプリケーション、または複数のアプリケーションの一覧。
アプリケーションリストが行末にある場合、-a オプションは不要です。
アプリケーションを特定のシステムインスタンスにロードするには、表 2-1 （2-3 ページ）に記載されている -a *instance*=*filename* オプションを使用します。
アプリケーションファイルには、.axf 拡張子が付きます。

注

Windows では、PATH に、Model Shell の実行可能ファイルが置かれるディレクトリを追加する必要がある場合があります。この場所は通常、次のようになります。

```
C:\Program Files\ARM\FastModelTools version\bin
```

ここで、*version* は Fast Models のリリースバージョンです（5.0 など）。

表 2-1 Model Shell コマンドラインオプション

ショート	ロングオプション	説明
-	--cpulimit *n*	*n* として実行する最大 CPU 秒数を指定します。秒の端数は指定できますが、残りの時間は 100ms の解像度に対してのみテストされます。 また、*n* を省略した場合、デフォルトは無制限になります。
-	--cyclelimit *n*	*n* として実行する最大サイクル数を指定します。 また、*n* を省略した場合、デフォルトは無制限になります。
-	--data *file@address*	指定されたアドレスにロードする未加工データを指定します。完全な形式は以下のとおりです。 --d [*INST*=]*file*@[*memspace*:]*address*
-	--dump *file@address,size*	メモリセクションを *file* にダンプします。完全な形式は以下のとおりです。 --u [*INST*=]*file*@[*memspace*:]*address,size*
-	--list-instances	ターゲットのインスタンスをリストします。
-	--list-params	ターゲットのインスタンスとそのパラメータをリストします。このオプションは、コンフィギュレーションファイルの正しい構文を特定しやすく、ターゲットによって何が提供されるのかを把握するために使用します。
-	--list-memory	モデルのメモリ情報を標準出力に出力します。
-	--start *address*	PC を指定されたアプリケーション開始アドレスに初期化します。完全な形式は以下のとおりです。 --start [*INST*=]*address*\1*address*
-	--stat	シミュレーションの最後に統計情報を出力します。
-	--timelimit *n*	*n* として実行する最大秒数を指定します。 また、*n* を省略した場合、デフォルトは無制限になります。 **注** *n* が 0 として指定された場合、Model Shell は以下の内容を実行します。 • システムを初期化します。 • すべてのアプリケーションとデータをロードします。 • ブレークポイントと PC を設定します。 • モデルを実行せずに即座に終了します。 このオプションは、アプリケーションを未加工のバイナリに変換するために使用します。以下に例を示します。 `model_shell --timelimit 0 -m mymodel.dll -a app.axf -u app.raw@0x8000,0x10000`

表 2-1 Model Shell コマンドラインオプション （続き）

ショート	ロングオプション	説明
-a	--application *filename*	アプリケーション *filename* をロードします。 特定のシステムインスタンスをロードするには、-a *instance=filename* を使用します。*「SMP のサポート」*（2-10 ページ）および *「文字列構文」*（2-6 ページ）を参照して下さい。 アプリケーションのファイル名がコマンドラインの末尾にある場合、-a オプションは不要です。
-b	--break *address*	指定されたアドレスにプログラムブレークポイントを設定します。 特定のシステムインスタンスにブレークポイントを設定するには、-b *instance=address* を使用します。
-C	--parameter *parameter=value*	パラメータを指定された値に設定します。詳細については、*「モデルパラメータの定義」*（2-7 ページ）を参照して下さい。 特定のシステムインスタンスのパラメータを設定するには、-C *instance*.parameter=*value* を使用します。
-d	--disassemble	Core Generator モデルの場合のみ、stdout でプログラムを逆アセンブルして終了します。
-f	--config-file *filename*	コンフィギュレーションファイル *filename* のモデルパラメータを使用します。詳細については、*「コンフィギュレーションファイル」*（2-8 ページ）を参照して下さい。*「文字列構文」*（2-6 ページ）も参照して下さい。
-h	--help	Model Shell コマンドラインオプションの一覧を出力して終了します。
-K	--keep-console	完了後にコンソールウィンドウを開いた状態に保ちます。
-L	--cadi-log	XML ログファイルのすべての CADI 呼び出しのログを作成します。
-m	--model *filename*	モデル *filename* をロードします。詳細については、*「文字列構文」*（2-6 ページ）を参照して下さい。 モデル名がコマンドラインの末尾にあり、オプションのアプリケーションファイル名以外のテキスト文字列が後に続いていない場合、-m オプショは不要です。
-o	--output *filename*	--list-instances コマンドと --list_params コマンドから出力をファイルに転送します。詳細については、*「文字列構文」*（2-6 ページ）を参照して下さい。 ファイルの内容は、入力として使用するために --config-file オプションによって正しくフォーマットされます。
-P	--prefix	ターゲットインスタンスの名前の先頭にセミホスティングしている出力を付けます。

表 2-1 Model Shell コマンドラインオプション （続き）

ショート	ロングオプション	説明
-R	--run	ロード後に即座にシミュレーションを実行します。CADI サーバが起動されている場合でもシミュレーションは即座に実行されます。
-q	--quiet	Model Shell の出力を抑制します。
-S	--cadi-server	CADI サーバを起動します。これにより、デバッガを接続してシミュレーションでターゲットをデバッグできます。 サーバをシャットダウンするには、モデルを起動するために使用したコマンドウィンドウに戻り、**Ctrl + C** を押して CADI サーバを停止します。Model Shell プロセスは前面に表示してからシャットダウンする必要があります。
-t	--cadi-trace	CADI 呼び出しおよびコールバックからの診断出力を可能にします。
-	--trace-plugin	トレースプラグインを指定します。このコマンドラインオプションまたは環境変数 FM_TRACE_PLUGINS で指定されるすべてのプラグインがロードされます。
-V	--verbose	ModelShell メッセージクラスを詳細なメッセージにすることができます。このコマンドは、以下のコマンドと同じ効果があります。 model_shell --enable-verbose ModelShell
-v	--version	Model Shell のバージョン番号を出力して終了します。

2.2 文字列構文

コマンドラインに入力されるファイル名および類似の文字列は、文字列にホワイトスペースがある場合、二重引用符で囲む必要があります。以下に例を示します。

```
model_shell -a “my application file.axf” ...
```

ただし、パラメータが単語でスペースがない場合は引用符を使用する必要はありません。以下の 2 つの形式は有効です。

```
model_shell --list-instances output.txt ...
```

```
model_shell --list-instances “output.txt” ...
```

2.3 モデルパラメータの定義

Model Shell を起動するために使用するモデルパラメータは、コマンドラインで定義できます。

定義は、以下の形式である必要があります。

`--parameter [`*instance*`.]`*parameter_name*`=`*value*

各項目には以下の意味があります。

`instance` コンポーネントインスタンスの名前。インスタンスが 1 つしか存在しない場合は、`instance` 指定子を省略できます。

instance は、各レベルがピリオド（.）文字で区切られた階層パスになる可能性があります。

`parameter_name`

変更対象のパラメータ。

階層システムの場合、インスタンスが 1 つしか存在しない場合でも完全な名前でパラメータを指定する必要があります。

`value` パラメータに割り当てる値。

value が文字列である場合、その他のフォーマット規則が適用される可能性があります。詳細については、*「文字列構文」*（2-6 ページ）を参照して下さい。

ブール値は、`true`/`false` または `1`/`0` を使用して設定できます。

`--parameter` オプションは、短縮形の `-C` で置き換えることが可能です。

2.3.1 コンフィギュレーションファイル

以下のコマンドを使用し、オプションのプレーンテキストのコンフィギュレーションファイルにリファレンスをインクルードすることによって、Model Shell を使用してコマンドラインから起動するモデルを設定できます。

```
model_shell --config-file my_configuration_file.txt ...
```

コンフィギュレーションファイルの各行には、コマンドラインの割り当てに使用される構文と同一の *instance.parameter=value* 構文が必要です。

コメントまたは空白テキストの前に # 文字を使用することによって、コンフィギュレーションファイルにコメント行および空白行を含めることもできます。

コンフィギュレーションファイルを生成するには、コマンドラインで --list-instances および --list-params オプションを使用します。コマンドラインにはパラメータの割り当てを含めることもできます。以下に、コマンドラインの例を示します。

```
model_shell --list-params --list-instances -C top-mm=0x3  -C top-mm=0x3  -o file.config -m model.so
```

例 2-1 にリストされている file.config コンフィギュレーションファイルが生成されます。

例 2-1 生成されたコンフィギュレーションファイル

```
# Instances:
# Instance id:instance name (SW:y/n, component, type, version) :description
# instance.parameter=value       #(type, mode) default = 'def value' :description :[min..max]
#-------------------------------------------------------------------------------------------
# Instance  0:(SW:no , NoCore, , 1.0) :Regression test system without PVLIB usage.
    top-p=0x2          # (int   , init-time) default = '0x2'    :test display name
    top-str="empty"    # (string, init-time) default = 'empty'  :test string param
    top-mm=0x3         # (int   , init-time) default = '0x6'    :test min(2) max(6) param :[0x2..0x6]
# Instance  1:a1    (SW:no , A, , 1.0) :
    a1.p1=0x2          # (int   , init-time) default = '0x2'    :A parameter p1
    a1.p2=0            # (bool  , run-time ) default = '0'      :A parameter p2
# Instance  2:a1.b  (SW:no , B, , 1.0) :
    a1.b.p1=0x2        # (int   , init-time) default = '0x2'    :B parameter p1
    a1.b.p2=""         # (string, run-time ) default = ''       :B parameter p2
# Instance  3:a2    (SW:no , A, , 1.0) :
    a2.p1=0x2          # (int   , init-time) default = '0x2'    :A parameter p1
    a2.p2=0            # (bool  , run-time ) default = '0'      :A parameter p2
# Instance  4:a2.b  (SW:no , B, , 1.0) :
    a2.b.p1=0x2        # (int   , init-time) default = '0x2'    :B parameter p1
    a2.b.p2="test"     # (string, run-time ) default = ''       :B parameter p2
#-------------------------------------------------------------------------------------------
```

例 2-1 では、ファイルを Model Shell の入力ファイルとして使用できるようにインスタンスの情報がコメント行の形式で記されています。top-mm および a2.b.p2 パラメータの値は、これらのパラメータの新しい値がコマンドラインで指定されたためにデフォルト値から変更されています。

例 2-2 （2-9 ページ）では、実行時パラメータを指定する別の例を示します。

例 2-2 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 VFP at reset using 1/0 syntax
coretile.core.vfp-enable_at_reset=1
#
# Set the baud rate for UART 0
baseboard.uart_0.baud_rate=0x4800
```

2.4 SMP のサポート

以下の 2 つの使用パターンがあります。

シンプル SMP コアが 1 つしかないシステムでのみ使用できます。

同じアプリケーションがすべての CPU にロードされます。このアプリケーションは、コマンドラインの残余にリストされます。

```
model_shell smp_model.so app.axf
```

標準 あらゆる事例で使用でき、-a オプションを使用して CPU ごとにアプリケーションをリストします。

CPU ごとに個別のアプリケーションをロードするだけではなく、さらに、-a オプションを使用すると、すべての CPU に同一のアプリケーションをロードすることもできます。

すべての CPU での -a を使用した同一アプリケーションのロード

インデックスの代わりにコアの接尾文字として * を使用します。以下のコマンドのいずれかを使用します。

```
model_shell -m smp_model.so -a core.cpu*=app.axf
model_shell -m smp_model.so -a "core.*"=app.axf
```

注

Unix の場合、* 文字をエスケープするために引用符が必要になる場合があります。

各 CPU への -a を使用した異なるアプリケーションのロード

各 CPU の完全なインスタンス名を使用して対象の CPU 用のアプリケーションを指定します。

```
model_shell -m smp_model.so -a core.cpu0=A.axf -a core.cpu1=B.axf
```

2.5 Model Shell の停止

ユーザによる操作または条件付きでコマンドライン引数を指定することによって、シミュレーションの実行を停止し、Model Shell を終了することがきます。

2.5.1 Model Shell の手動シャットダウン

手動で Model Shell の実行を停止するには、以下のいずれかを実行します。

- **Ctrl+C** を押します。プログラムはシミュレータのシャットダウンを開始し、シャットダウンが完了してから終了します。再び **Ctrl+C** を押し、Model Shell を即座に終了します。

  注
  一部のモデルは、Model Shell の動作をオーバーライドする独自の Control C ハンドラを割り当てることができます。

- **Ctrl** と**改行**を押し（Windows のみ）、即座に Model Shell を終了します。
- LCD ウィンドウを閉じるなどしてシミュレーションを停止します。

2.5.2 Model Shell の自動シャットダウン

Model Shell を自動的にシャットダウンするには、コマンドラインに停止条件を指定します。通常、条件は、最初に満たされた条件によって実行が中されるように組み合わされます。

注
すべての停止条件を CADI サーバモードで使用できるとは限りません。

Model Shell の自動シャットダウンでは、以下の条件がサポートされます。

- シミュレーションが停止します。

  注
  デフォルトの条件の場合、CADI サーバが起動されていない場合。
  CADI サーバが `--cadi-server` オプションを使用して起動されていた場合、この条件は無視されます。

- シミュレーションがブレークポイントにヒットします。`--break` オプションを使用してブレークポイントを指定します。

  注
  CADI サーバが `--cadi-server` オプションを使用して起動されていた場合、この条件は無視されます。
  このオプションは、--cyclelimit オプションと組み合わせて使用することはできません。

- サイクル数が超過しています。`--cyclelimit` オプションを使用してサイクル制限を指定します。

注

このオプションは、--cadi-server オプションと組み合わせて使用することはできません。

このオプションを使用すると、実行速度が低下する可能性があります。

ブレークポイントは無視されます。

- 制限時間が超過しています。--timelimit オプションを使用して、実時間の秒単位で実行制限時間を指定します。
- cpu 制限が超過しています。--timelimit オプションを使用して、CPU 秒単位で実行制限時間を指定します。

注

0.1 秒の時間粒度で指定された時間をテストし、パフォーマンスの低下を回避します。

ユーザおよびカーネル時間の両方が CPU 経過時間の測定値に含まれます。

2.6 model_shell および isim_system からのライセンス確認エラーの検出

Windows および Linux で、model_shell および isim_system からライセンス確認エラーを検出することができます。そのためには、以下の手順を実行します。

1. 以下のいずれかの場合、model_shell/isim_system の終了ステータスを確認します。
   - システムが起動して実行されている場合、実行可能ファイルが終了するまで終了ステータスは取得できません。
   - 終了ステータスが 0 の場合、エラーがないことを意味しています（シミュレータのクリーンシャットダウンなど）。
   - 終了ステータスが 1 の場合、エラーが発生したことを意味しています（ライセンス確認またはファイル損失など）。
2. 終了ステータス 1 が Parse stderr of model_shell/isim_system という結果になった場合は、「ERROR: License check failed!」の行を探します。このメッセージが表示されたら後続の行を確認します。そこには、ライセンス確認エラーに関する詳細なエラーメッセージがされています。メッセージは GUI で表現され、次の行が WARNING、ERROR、または Fatal Error から始まる前にメッセージが終わっている場合があります。エラーメッセージは、数回繰り返される場合があります。

   ライセンスの期限切れが間近になると、stdout に警告が出力されます。シミュレーションは正常に起動するので、「WARNING」の行を見つけて下さい。このメッセージが表示されたら後続の行を確認します。そこには、ライセンス確認モジュールからの実際の警告メッセージが記されています。メッセージは GUI で表現され、次の行が WARNING、ERROR、または Fatal Error から始まる前にメッセージが終わっている場合があります。エラーメッセージは、数回繰り返される場合があります。

エラーおよび警告メッセージの例については、例 2-3、例 2-4、および例 2-5 （2-14 ページ）を参照して下さい。

例 2-3 ライセンス確認エラーのエラーメッセージの例

```
ERROR:License check failed!
Either the license file or the license server could not be found.
Please set the environment variable 'ARMLMD_LICENSE_FILE'
to your license file location or refer to the ARM_FLEXnet_Guide
for instructions on where to obtain a license file, where to install
the license file, and how to setup a license server.
Error Code :-1
```

例 2-4 ライセンス確認エラーのエラーメッセージの例

```
ERROR:License check failed!
No licenses 'FM_Simulator' available.
No such feature exists.
License files searched:
h:\tmp\win\warningtest_ANY_04-feb-2011.dat
Error Code :-5
```

例 2-5 警告メッセージの例

```
警告 :
Licenses 'SG_ARM1176_CT' expire in 0 days.
Please contact ARM support to renew your license or to receive a new license.
License files searched:
h:\tmp\win\warningtest_ANY_04-feb-2011.dat
警告 : General warning:
```