dSPACE Release

新機能と移行手順

Release 2015-B – 2015 年 11 月

Embedded Success dSPACE

dSPACE へのお問い合わせ

dSPACE Japan 株式会社

Web:	http://www.dspace.jp
テクニカルサポート:	support@dspace.jp

dSPACE サポートへのお問い合わせ

問題点やご質問を dSPACE にお問い合わせいただく場合に、http://www.dspace.jp/go/jpn_supportrequest のお問い合わせフォームにサポートのお申し込みをご入力ください。

お問い合わせフォームは、サポートチームがより迅速かつ効果的に問題点を取り扱うのに役立ちます。

ソフトウエアのアップデートとパッチ

既存の dSPACE インストレーションに対して、最新のパッチをダウンロードしてインストールすることを強くお勧めします。ソフトウエアのアップデートとパッチについては、以下のサイトをご覧ください。http://www.dspace.jp/goto.cfm/ja_0903

重要なお知らせ

目次

本書について

目次

本書では、Release 2015-B に含まれるすべての dSPACE ソフトウエア製品の新機能について説明します。以前の dSPACE リリースからの変更がない、または変更が少ないソフトウエア製品についても概要を示します。また、以前の dSPACE リリース、特に以前の製品バージョンからの移行手順についても、必要に応じて説明します。

項目の一覧

本章の内容

本書で使用する記号と表記規則

記号

本書では次の記号を使用します。

	人身傷害につながる一般的な危険があることを示します。本書の指示に従って危険を回避しないと、けがをする可能性があります。
	感電の危険があることを示します。本書の指示に従って危険を防止しないと、死亡または重傷を負う可能性があります。
	物的な損害の危険があることを示します。本書の指示に従って危険を防止しないと、物的損傷を招く可能性があります。

	注意すべき重要な情報（故障を回避するための注意など）を示します。
	作業を円滑に進めるのに役立つヒントを示します。

表記規則

本書では次の省略表記と書式を使用します。

%name%　パーセント記号で囲まれた名前は、ファイルとパス名の環境変数を表します。

< >　山形括弧で囲まれた表記は、任意のファイル名やパス名などを表すワイルドカード文字またはプレースホルダを示します。

リンク先が別のドキュメントを参照する場合にドキュメントタイトルの前に付記されます。

リンク先が dSPACE HelpDesk で提供されている別のドキュメントを参照していることを示します。

特別なフォルダ

ControlDesk Next Generation や AutomationDesk などの一部のソフトウエア製品では、次の特別なフォルダを使用します。

共通プログラムデータフォルダ　アプリケーション固有の設定データ用の標準フォルダで、すべてのユーザが使用します。

```
%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>
```

ドキュメントフォルダ　ドキュメント用の標準フォルダで、各ユーザ固有のフォルダです。

```
%USERPROFILE%\My Documents\dSPACE\<ProductName>\
<VersionNumber>
```

ローカルプログラムデータフォルダ　アプリケーション固有の設定データ用の標準フォルダで、現在の非ローミングユーザが使用します。

```
%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\
<ProductName>
```

オンラインヘルプおよび PDF ファイルの利用

目的

dSPACE ソフトウエアをインストールすると、インストールした製品に関するドキュメントがオンラインヘルプまたは Adobe® PDF ファイルとして参照できるようになります。

オンラインヘルプ

オンラインヘルプ(dSPACE HelpDesk)を使用するには

Windows の[スタート]メニュー [スタート]メニューから[(すべての)プログラム] – [<製品名>] – [dSPACE HelpDesk (<製品名>)]を選択して、選択した製品のスタートページから dSPACE HelpDesk を開きます。また、インストールされている他のソフトウエア製品およびそれにサポートされるハードウエアのユーザマニュアルに移動して検索することもできます。

状況依存ヘルプ 現在アクティブなコンテキストのヘルプを表示するには、F1 キーを押すか、または dSPACE ソフトウエアの[Help]ボタンをクリックします。

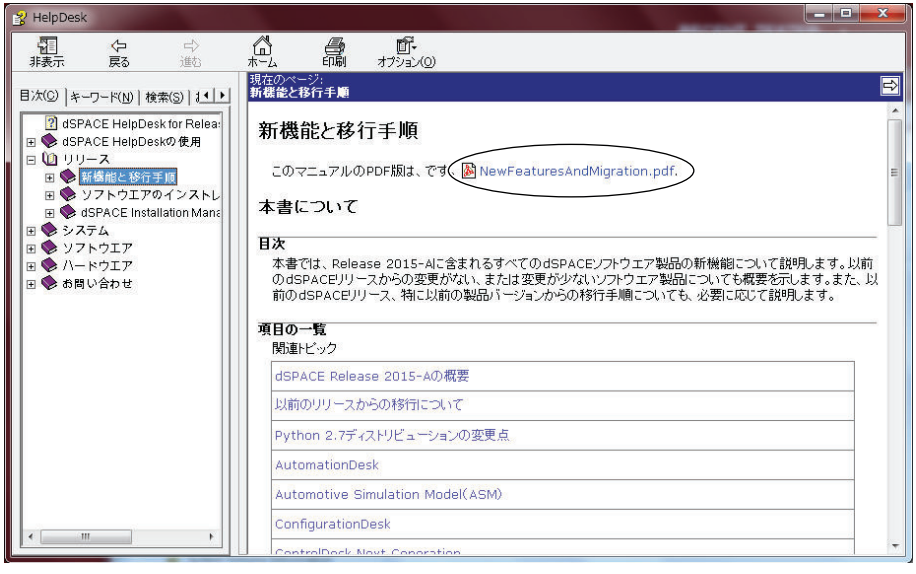
> いくつかのソフトウェア製品では、文脈依存ヘルプは入手可能ではない。

dSPACE ソフトウエアの[Help]メニュー メニューバーから[Help] – [Contents]または[Help] – [Search](すべてのソフトウエア製品で利用可能とはかぎりません)を選択して dSPACE HelpDesk を開きます。現在アクティブな製品のスタートページが表示されます。また、インストールされている他のソフトウエア製品およびそれにサポートされるハードウエアのユーザマニュアルに移動して検索することもできます。

PDF ファイル

PDF ファイルは、次の方法で利用することができます。

dSPACE HelpDesk ドキュメント名の先頭にある PDF リンクをクリックします。



dSPACE Release 2015-B の概要

目的	Release 2015-B の主な新機能について説明します。また、変更のない製品に関する情報についても紹介します。
項目の一覧	本章の内容

一般的な機能拡張および変更

目的	複数の dSPACE 製品に関係する新機能と変更を下記に示します。
新しい dSPACE ハードウエアのサポート	dSPACE Release 2015-B では、以下の新しい dSPACE ハードウエアが導入されています。 ■ *DS2655M2 I/O Module* この SCALEXIO モジュールは、DS2655 FPGA Base Board に取り付けることができ、RTI FPGA Programming Blockset でサポートされています。「RTI FPGA Programming Blockset 3.0 の新機能」(167 ページ)を参照してください。

32 ビットおよび 64 ビットソフトウエアの提供

dSPACE ソフトウエアは 2 つの DVD セットで提供されます。各 DVD の内容は同じですが、次の点が異なります。

- 2 枚組の 32 ビット DVD セットには、(MATLAB の 32 ビットバージョンをサポートするなどの)dSPACE ソフトウエア製品の 32 ビットバージョンのみが含まれます。
- 2 枚組の 64 ビット DVD セットには、次の内容が含まれます。
  - MATLAB の 64 ビットバージョンをサポートする MATLAB を使用するすべての dSPACE 製品
  - MATLAB の 64 ビットバージョンをサポートするすべての 32 ビットバージョンの dSPACE 製品
  - MATLAB に関連しない 32 ビットバージョンの dSPACE 製品すべて(ControlDesk Next Generation など)

したがって、インストール時には、64 ビット DVD セットを 32 ビット DVD セットに取り替えなくても dSPACE ソフトウエアをインストールすることができます。

64 ビット dSPACE DVD セットに含まれるすべての dSPACE 製品の一覧とそれぞれの MATLAB サポート状況については、「64 ビット dSPACE DVD セットに含まれる製品とその MATLAB サポート」(261 ページ)を参照してください。

DVD セットの内容

dSPACE ソフトウエアは各 DVD セット(32 ビットと 64 ビット)の 2 枚の DVD で提供されます。DVD には、以下の dSPACE ソフトウエアパッケージとメインの製品が収録されています。

- ディスク 1:
  - AutomationDesk 5.1
  - ControlDesk Next Generation(ControlDesk 5.5)
  - TargetLink 4.1
  - Model Compare 2.6

この製品は米国での使用が禁止されています

米国では Model Compare を使用することはできません。この製品を米国内で使用することも第三者に使用させることも米国の法律に違反します。

  - SystemDesk 4.5(AUTOSAR 4.x をサポート)
  - VEOS 3.5
  - dSPACE ソフトウエアのその他各種ツール

- ディスク 2:
  - RCP and HIL Software

    RCP and HIL Software は、RTI、ConfigurationDesk、MotionDesk、ModelDesk などのさまざまな dSPACE ソフトウエア製品が含まれるソフトウエアパッケージを指す総称です。

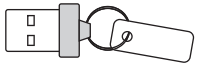
    ディスク 2 には、その他の dSPACE ソフトウエア製品は収録されていません。

ドングルライセンスの新しいハードウエアドングル

dSPACE Release 2014-B から、ドングルライセンスのハードウエアドングルが WibuKey から CodeMeter に変更されています。両方とも WIBU-SYSTEMS 社の製品であり、外観は下図のとおりです。

WibuKeyドングル

CodeMeterドングル



dSPACE Release 2014-B では、新しい CodeMeter ハードウエアドングルは、新規の dSPACE システムを初めて導入する場合に提供されます。

次の互換性情報にご注意ください。

- 通常、既存の WibuKey ドングルで dSPACE Release 2015-B をご利用いただけます。dSPACE Release 2014-B では、両バージョンのドングルドライバがホスト PC にインストールされます。ドライバソフトウエアがご使用のドングルを自動的に検出します。他の作業は必要ありません。
- 新しい CodeMeter ドングルで dSPACE Release 2014-A 以前のバージョンを使用する場合は、ご使用のホスト PC に dSPACE Installation Manager 3.8(以降)をインストールする必要があります。このバージョンには、新しいドングルのドライバが含まれています。dSPACE Installation Manager の最新のバージョンは http://www.dspace.jp/goto.cfm/IMUpdate_jp からダウンロードすることができます。
- dSPACE Release 6.3 以前のバージョンでは、新しい CodeMeter ドングルのテストは行っていません。必要に応じて、dSPACE サポートにご連絡ください。

dSPACE HelpDesk 使用時の制限事項

dSPACE HelpDesk は、`C:\Program Files\Common Files\dSPACE`(32 ビット版オペレーティングシステム)または `C:\Program Files(x86)\Common Files\dSPACE`(64 ビット版オペレーティングシステム)のリリースごとのフォルダにインストールされます。たとえば、dSPACE Release 2015-A の製品と dSPACE Release 2015-B の製品をインストールした場合、2 つの dSPACE HelpDesk を使用することができます。

以下の制限事項に注意してください。

ドキュメントへのリンクが機能せず、エラーメッセージ*"Selection is not associated with any topics."*が返される場合があります。これは、次のいずれかの理由が考えられます。

- 製品がライセンスキーに含まれていないため、製品のドキュメントがインストールされていない。
- 製品のドキュメントが別の dSPACE HelpDesk にインストールされている。たとえば、現在の dSPACE Release にある製品が変更されていない場合、そのユーザマニュアルは製品セットアップが作成されたバージョンの dSPACE HelpDesk にインストールされています。

  dSPACE Release 2015-B をインストールした場合、以下の製品のユーザマニュアルは dSPACE HelpDesk 2015-A に格納されます。

  - dSPACE ECU Flash Programming Tool 2.2.6
  - SYNECT Server 1.4.1

  製品のユーザマニュアルの場所が不明な場合は、Windows の[スタート]メニューから製品固有の[dSPACE HelpDesk]ショートカットを使用してオンラインヘルプを開いてください。

印刷版のユーザマニュアル

dSPACE Release 2015-B には、印刷版のユーザマニュアルは付属していません。必要な印刷版のマニュアルをユーザが指定できるようになっています。印刷版のユーザマニュアルについては、http://www.dspace.jp/go/request_jp_documentation を参照してください。

印刷版のマニュアルを注文しない場合は、ご使用の製品に関する新機能、拡張機能、安全上の注意事項などの情報については、dSPACE HelpDesk または PDF ファイルをご使用ください。

ソフトウエアサポートの廃止

32 ビットソフトウエアサポートの廃止予定 dSPACE Release 2015-B は、32 ビットオペレーティングシステムおよび 32 ビット MATLAB バージョンをサポートする最後のリリースです。dSPACE Release 2016-A 以降、dSPACE ソフトウエアは 64 ビットオペレーティングシステムおよび 64 ビット MATLAB バージョンのみをサポートします。

MicroAutoBox ソフトウエアサポートの廃止予定 dSPACE Release 2015-B は、MicroAutoBox バージョン 1401/1501、1401/1504、1401/1505/1506、1401/1505/1507、および 1401/1507 をサポートする最後のリリースです。dSPACE Release 2016-A 以降、dSPACE ソフトウエアは、MicroAutoBox II バージョン 1401/1507、1401/1511、1401/1513、1401/1511/1514、および 1401/1513/1514 のみをサポートします。

64 ビットバージョンの RCP and HIL Software

目的	RCP and HIL Software 製品は MATLAB 64 ビットバージョンをサポートしています。
RCP and HIL（64 ビット）Software での製品サポート	RCP and HIL（64 ビット）Software には通常、dSPACE Release 2015-B（32 ビット）DVD で入手可能な RCP and HIL Software と同じ製品が含まれています。 RCP and HIL およびその他の dSPACE ソフトウエア製品での 64 ビット MATLAB のサポートについては、「64 ビット dSPACE DVD セットに含まれる製品とその MATLAB サポート」(261 ページ)を参照してください。
サポートされる MATLAB バージョン	RCP and HIL（64 ビット）Software は以下をサポートします。 ■ MATLAB R2014a（64 ビット） ■ MATLAB R2014b（64 ビット） ■ MATLAB R2015a（64 ビット） ■ MATLAB R2015b（64 ビット） 「サポートしている MATLAB リリース」(256 ページ)も参照してください。
サポートされている MEX コンパイラ	MEX 関数をビルドする場合、RCP and HIL（64 ビット）Software は Microsoft Windows SDK 7.1 のみをサポートします。 このコンパイラは、Microsoft 社 Web サイトから無料でダウンロードすることができます。このコンパイラを使用するには、.NET framework 4.0 が必要です。こちらも Microsoft 社から無料で提供されています。コンパイラとフレームワークのダウンロード、またはその他の詳細については、http://www.mathworks.com/support/compilers/R2015a/index.html を参照してください。 MEX コンパイラが必要な次のような RCP and HIL Software 製品を使用する場合は、このコンパイラをインストールして MEX コンパイラとして設定します。 ■ RTI CAN MultiMessage Blockset ■ RTI LIN MultiMessage Blockset ■ Automotive Simulation Model ■ MotionDesk Blockset
システム要件	RCP and HIL（64 ビット）Software には、Windows 7 Enterprise Service Pack 1（64 ビット版）が必要です。その他の 64 ビット版のオペレーティングシステム（Windows XP および Windows Vista）はサポートされません。

ホスト PC のメインメモリは、4 GB RAM 以上である必要があります。8 GB RAM 以上をお勧めします。

「オペレーティングシステム」(257 ページ)も参照してください。

製品バージョンの概要

目的

次の表に、各製品の最新のリリースおよび過去 3 回のリリースのバージョン履歴を示します。新機能が追加されている場合は、本書での参照先を示しています。

製品名	dSPACE Release			
	2014-A	2014-B	2015-A	2015-B
AutomationDesk	4.1	4.1	5.0	5.1 「AutomationDesk」(49 ページ)を参照してください。
Automotive Simulation Model	6.0	7.0	8.0	8.1 「Automotive Simulation Model（ASM）」(53 ページ)を参照してください。
ConfigurationDesk	5.1	5.2	5.3	5.4 「ConfigurationDesk」(89 ページ)を参照してください。
Container Manager	4.2	4.3	4.3	4.4 「コンテナ管理」(97 ページ)を参照してください。
ControlDesk Next Generation	5.2	5.3	5.4	5.5 「ControlDesk Next Generation」(99 ページ)を参照してください。
DCI Configuration Tool	3.2.2	3.3	3.4	3.5 「DCI Configuration Tool」(123 ページ)を参照してください。
dSPACE CAN API	2.7.1	2.7.1	2.7.1	2.7.4
dSPACE ECU Flash Programming Tool	2.2.5	2.2.5	2.2.6	2.2.6
dSPACE FlexRay Configuration Package	3.3	3.4	3.5	3.6 「dSPACE FlexRay Configuration Package」(125 ページ)を参照してください。
dSPACE HIL API .NET	1.6	1.6	1.8	2.0 「dSPACE HIL API .NET」(127 ページ)を参照してください。

製品名	dSPACE Release			
	2014-A	2014-B	2015-A	2015-B
dSPACE Python Extensions	1.6	1.7	1.8	2.0 「dSPACE Python Extensions」(129 ページ)を参照してください。
dSPACE XIL API	–	2.0	2015-A	2015-B 「dSPACE XIL API」(133 ページ)を参照してください。
ECU Interface Manager	1.4.1	1.5	1.6	1.7 「ECU Interface Manager」(135 ページ)を参照してください。
Firmware Manager	1.1	1.2	1.3	2.0 「Firmware Manager」(139 ページ)を参照してください。
Model Compare	2.4	2.5	2.5	2.6 「Model Compare」(141 ページ)を参照してください。
ModelDesk	3.2	4.0	4.1	4.2 「ModelDesk」(145 ページ)を参照してください。
Model Interface Package for Simulink	–	–	3.0	3.1 「Model Interface Package for Simulink」(147 ページ)を参照してください。
MotionDesk	3.4	3.5	3.6	3.7 「MotionDesk」(149 ページ)を参照してください。
MotionDesk Blockset	2.3	2.3.1	2.3.2	2.4 「MotionDesk」(149 ページ)を参照してください。
Real-Time Testing	2.3	2.4	2.5	2.6 「Real-Time Testing」(151 ページ)を参照してください。
RTI[1)]	7.2	7.3	7.4	7.5 「RTI/RTI-MP および RTLib」(153 ページ)を参照してください。
RTI-MP[2)]	7.2	7.3	7.4	7.5 「RTI/RTI-MP および RTLib」(153 ページ)を参照してください。
RTI AUTOSAR Package	1.3.1	–	–	–

製品名	dSPACE Release			
	2014-A	2014-B	2015-A	2015-B
RTI Bypass Blockset	3.2	3.3	3.4	3.5 「RTI Bypass Blockset」(157 ページ)を参照してください。
RTI CAN Blockset	3.2	3.3	3.4	3.4.1
RTI CAN MultiMessage Blockset	3.0	4.0	4.1	4.2 「RTI CAN MultiMessage Blockset」(161 ページ)を参照してください。
RTI Electric Motor Control Blockset	–	1.0	1.1	1.2 「RTI Electric Motor Control Blockset」(165 ページ)を参照してください。
RTI Ethernet Blockset	1.0	1.1	1.2	1.2
RTI Ethernet (UDP) Blockset	1.3	1.3	1.4	1.4
RTI FPGA Programming Blockset	2.7	2.8	2.9	3.0 「RTI FPGA Programming Blockset」(167 ページ)を参照してください。
RTI LIN MultiMessage Blockset	2.3	2.4	2.5	2.5.1 「RTI LIN MultiMessage Blockset」(173 ページ)を参照してください。
RTI RapidPro Control Unit Blockset	2.2	2.2	2.2.1	2.2.1
RTI USB Flight Recorder Blockset	1.1	1.2	1.2	1.2
RTI Watchdog Blockset	1.0	1.0	1.0	1.0
SCALEXIO Firmware	3.0	3.1	3.2	3.3 「SCALEXIO Firmware」(175 ページ)を参照してください。
SYNECT Server	1.3.1	1.4	1.4.1	1.4.1
SystemDesk 4.x[3)]	4.2	4.3	4.4	4.5 「SystemDesk」(177 ページ)を参照してください。
TargetLink/TargetLink Data Dictionary	3.5	4.0	4.0	4.1 「TargetLink」(191 ページ)を参照してください。
Variable Editor	1.8	1.8	2.1	2.2

製品名	dSPACE Release			
	2014-A	2014-B	2015-A	2015-B
VEOS	3.2	3.3	3.4	3.5 「VEOS」(249 ページ)を参照してください。

[1] 標準の I/O ブロックセットを含みます。
[2] RTI Gigalink Blockset を含みます。
[3] AUTOSAR 4.x をサポート

定期的に更新を行っていない場合は、新機能と必要な移行手順について、上記の各 dSPACE Release の『*新機能と移行手順*』マニュアルを参照してください。

各製品の主な新機能

目的

ここでは、各製品の主な新機能の概要を示します。詳細については、各製品のセクションを参照してください。

本章の内容

AutomationDesk

AutomationDesk の主な新機能は次のとおりです。

- 新しい時間タグ機能など、信号ベースのテストの拡張
- AutomationDesk API の拡張
- ユーザビリティの強化：式と条件を指定する新しいエディタなど

	■ トレースファイル生成の機能強化をサポートする新しい Variable Browser ■ DS1007 PPC Processor Board および MicroLabBox に対するシグナルジェネレータのサポートの強化 新機能の詳細については、「AutomationDesk 5.1 の新機能」(49 ページ)を参照してください。
ConfigurationDesk (Implementation Version)	ConfigurationDesk の主な新機能は、次のとおりです。 ■ MATLAB R2014a で導入されたパラメータ処理に関する Simulink Coder の機能のサポート ■ プリコンパイル済み SIC ファイルのサポート 新機能の詳細については、「ConfigurationDesk – Implementation」(90 ページ)を参照してください。
コンテナ管理	コンテナ管理の新機能は、次のとおりです。 ■ SystemDesk 4.5 でのコンテナファイルへのエレメントの割り当ての改善 新機能の詳細については、「コンテナ管理の新機能」(97 ページ)を参照してください。
ControlDesk Next Generation	ControlDesk Next Generation(ControlDesk 5.5)の主な新機能は、次のとおりです。 ■ プラットフォーム／デバイスの拡張: ■ 新しい DCI-CAN2 のサポート(CAN FD のサポートを含む) ■ CAN バスモニタリングデバイス:FIBEX および AUTOSAR システムデスクリプションファイルのサポート ■ FlexRay バスモニタリングデバイス:AUTOSAR システムデスクリプションファイルのサポート ■ 自動再接続をサポートするプラットフォームの追加 ■ シミュレーション時間グループのマスターの指定 ■ 仮想検証シナリオの改善 ■ 変数管理の拡張: ■ 新しい Variable Browser ■ 構造体と構造体配列のサポート ■ 元の接続パスの表示

- 計器およびビジュアル表示の向上:
  - 計器への変数の接続割り当てのカスタマイズ
  - Time Plotter と Index Plotter の拡張:
    - Time Plotter データを新しい計測として保存(Time Plotter のみ)
    - x 軸の同期
  - マルチスイッチの拡張
- 計測および記録の拡張:
  - 新しいデフォルトの交換フォーマット ASAM MDF 4.x
  - 多数の計測ラスタの処理
  - Measurement Data API の拡張
- データセット管理の拡張:
  - 新しいデフォルトの交換フォーマット CDFX
- Bus Navigator の拡張:
  - Bus Manager 設定用のバス計器の生成
  - MicroLabBox での CAN バス通信の再生
  - SCALEXIO および DCI-CAN2 での CAN FD のサポート
  - CAN バスモニタリングデバイス:FIBEX および AUTOSAR システムデスクリプションファイルのサポート
- Signal Editor の拡張:
  - DS1007 の完全なサポート
  - DS1202 MicroLabBox のサポート
- 電気的欠陥シミュレーション(欠陥シミュレーション)の拡張:
  - 新しい XIL API EESPort グラフィカルユーザインターフェース(ControlDesk の Failure Simulation Module の後継)
- 自動化の拡張:
  - ルックアップテーブルの計測と記録(マップと曲線)
  - 計測信号リストに信号を追加または信号を削除する際のイベント

新機能の詳細については、「ControlDesk Next Generation の新機能(ControlDesk 5.5)」(100 ページ)を参照してください。

DCI Configuration Tool

DCI Configuration Tool の主な新機能は次のとおりです。

- A2L ファイル適合の改善

新機能の詳細については、「DCI Configuration Tool 3.5 の新機能」(123 ページ)を参照してください。

dSPACE FlexRay Configuration Package	dSPACE FlexRay Configuration Tool の主な新機能は、次のとおりです。 ■ AUTOSAR System Template 4.2.1 のサポート ■ 不明確なバイトオーダー形式のサポート 新機能の詳細については、「dSPACE FlexRay Configuration Package 3.6 の新機能」(125 ページ)を参照してください。
dSPACE HIL API .NET	dSPACE HIL API .NET の主な新機能は次のとおりです。 ■ トレースファイル生成の機能強化のサポート ■ スティミュラスサポートの拡張 新機能の詳細については、「dSPACE HIL API .NET」(127 ページ)を参照してください。
dSPACE XIL API	dSPACE XIL API の主な新機能は次のとおりです。 ■ トレースファイル生成の機能強化のサポート ■ スティミュラスサポートの拡張 新機能の詳細については、「dSPACE XIL API 2015-B の新機能」(133 ページ)を参照してください。
ECU Interface Manager	ECU Interface Manager の主な新機能は、次のとおりです。 ■ DPMEM プラグオンデバイス(POD)用に設定された組込みの dSPACE Calibration and Bypassing Service のサポート ■ 実行制御の挿入が失敗した場合の動作の指定 新機能の詳細については、「ECU Interface Manager 1.7 の新機能」(135 ページ)を参照してください。
Firmware Manager	Firmware Manager の主な新機能は次のとおりです。 ■ SCALEXIO システムのサポート 新機能の詳細については、「Firmware Manager 2.0 の新機能」(139 ページ)を参照してください。
Model Compare	Model Compare の主な新機能は、次のとおりです。 ■ ダンプファイルの作成を制御できるフック機能 ■ Property Inspector でのプロパティ値のコピー ■ 最も重要な機能をまとめた Model Compare Quick Guide 新機能の詳細については、「Model Compare 2.6 の新機能」(141 ページ)を参照してください。

ModelDesk	ModelDesk の主な新機能は、次のとおりです。 ■ 新しいシミュレーションプラットフォームのサポート：DS1007 PPC Processor Board および MicroLabBox ■ 処理とプロットのために拡張したツール自動化 新機能の詳細については、「ModelDesk 4.2 の新機能」(145 ページ)を参照してください。
MotionDesk	MotionDesk の主な新機能は、次のとおりです。 ■ 新しい計器機能：計器パネル ■ 新しくサポートされるプラットフォーム：MicroLabBox ■ 計器を複数のオブザーバに割り当て可能 ■ オブザーバを処理するように拡張されたツール自動化 新機能の詳細については、「MotionDesk 3.7 の新機能」(149 ページ)を参照してください。
Python Extensions	MAPort の dSPACE HIL API Python Implementation の主な新機能は次のとおりです。 ■ トレースファイル生成の機能強化のサポート ■ スティミュラスサポートの拡張 matlablib2 の主な新機能は次のとおりです。 ■ MATLAB および MATFile インスタンスの新しい拡張されたメソッドとプロパティ 新機能の詳細については、「dSPACE Python Extensions」(129 ページ)を参照してください。
Real-Time Testing	Real-Time Testing の主な新機能は、次のとおりです。 ■ 新しくサポートされるプラットフォーム：MicroLabBox 新機能の詳細については、「Real-Time Testing 2.6 の新機能」(151 ページ)を参照してください。

RTI、RTI-MP、RTLib	RTI、RTI-MP、RTLib の主な新機能は、次のとおりです。 ■ MATLAB R2015b で導入された、トレースファイル生成の機能強化に反映される新しい Simulink Coder 機能のサポート ■ MicroLabBox での RTI サポートの拡張 ■ MicroLabBox および DS1007 での不揮発性データ処理（NVDATA）のサポート 新機能の詳細については、「RTI/RTI-MP および RTLib の新機能」(153 ページ)を参照してください。
RTI Bypass Blockset	RTI Bypass Blockset の主な新機能は次のとおりです。 ■ XCP 1.3 のサポート ■ CAN FD のサポート 新機能の詳細については、「RTI Bypass Blockset 3.5 の新機能」(157 ページ)を参照してください。
RTI CAN MultiMessage Blockset	RTI CAN MultiMessage Blockset の主な新機能は次のとおりです。 ■ SCALEXIO システムでの CAN FD のサポート ■ ISO CAN FD プロトコルのサポート ■ CAN FD メッセージのアービトレーションフェーズとデータフェーズのサンプリングポイント ■ 不明確なバイトオーダー形式のサポート 新機能の詳細については、「RTI CAN MultiMessage Blockset 4.2 の新機能」(161 ページ)を参照してください。
RTI Electric Motor Control Blockset	RTI Electric Motor Control Blockset の主な新機能は、次のとおりです。 ■ EnDat インターフェースベースの位置計測用エンコーダのサポート 新機能の詳細については、「RTI Electric Motor Control Blockset 1.2 の新機能」(165 ページ)を参照してください。
RTI FPGA Programming Blockset	RTI FPGA Programming Blockset の主な新機能は次のとおりです。 ■ Xilinx®ソフトウエアのサポートの拡張 ■ DS2655 FPGA Base Board 向けの FPGA フレームワークの拡張 ■ DS2655M2 Digital I/O Module 向けの新しい FPGA フレームワーク ■ DS2655M1 Multi-I/O Module 向けの FPGA フレームワークの拡張 新機能の詳細については、「RTI FPGA Programming Blockset 3.0 の新機能」(167 ページ)を参照してください。

RTI LIN MultiMessage Blockset	RTI LIN MultiMessage Blockset の主な新機能は次のとおりです。 ■ 不明確なバイトオーダー形式のサポート 新機能の詳細については、「RTI LIN MultiMessage Blockset 2.5.1 の新機能」(173 ページ)を参照してください。
SCALEXIO Firmware	SCALEXIO Firmware の主な新機能は次のとおりです。 ■ DS2655M2 Digital I/O Module のサポート 新機能の詳細については、「SCALEXIO Firmware 3.3 の新機能」(175 ページ)を参照してください。
SystemDesk	SystemDesk 4.5 の主な新機能は、次のとおりです。 ■ AUTOSAR 4.2.2、4.2.1、4.1.3、4.1.2、4.1.1、および 4.0.3 のサポート ■ SystemDesk のグラフィカルモデリング用ダイアグラムの改善 ■ AUTOSAR に準拠したベーシックソフトウエアモジュール記述テンプレートのサポート ■ RTE 生成の改善 ■ 仮想検証のための NVRAM マネージャのサポート ■ 多様なバリアントを含むモデルのバリアントバインディング 新機能の詳細については、「新しい一般機能」(178 ページ)を参照してください。
TargetLink	TargetLink の主な新機能は、次のとおりです。 ■ AUTOSAR ■ Revision 4.2.1 のサポート ■ NvData 通信 ■ データ変換 ■ ポートにより定義される引数値 ■ Activation Reason ■ 参照先モデル(複数のインスタンスを含む)とインクリメンタルサブシステムをサポートする改善された関数再利用 ■ 生成された量産コードを FMI 準拠のツールでシミュレートするための TargetLink サブシステムの FMU のエクスポート ■ バスの初期状態の構造体を含む、Simulink の簡易初期化モードのサポート ■ Stateflow のバスのサポート

- TargetLink ブロックと Stateflow Chart オブジェクトから参照できる、DD RequirementInfo オブジェクトを介した Data Dictionary への要件情報の保存のサポート

すべての新機能の詳細については、「TargetLink 4.1 および TargetLink Data Dictionary 4.1 の新機能」(192 ページ)を参照してください。

TargetLink の移行に関する(TargetLink、TargetLink AUTOSAR モジュール、TargetLink Data Dictionary)詳細については、「TargetLink 4.1 および TargetLink Data Dictionary 4.1 への移行」(218 ページ)を参照してください。

VEOS

VEOS の主な新機能は、次のとおりです。

- より直感的なユーザインターフェース
- デバッグ情報の生成の有効化／無効化(MSVC、GCC)
- 例外時の呼び出しスタック情報へのアクセス

新機能の詳細については、「VEOS」(249 ページ)を参照してください。

以前のリリースからの移行について

目的	最新の dSPACE Release の製品をインストールした後、いくつかの追加手順が必要になる場合があります。前回の dSPACE Release から移行する場合に必要な手順は、本書の製品固有の移行トピックスに記載されています。それ以前の dSPACE Release から移行する場合は、関連する『*新機能と移行手順*』を参照してください。
項目の一覧	本章の内容 他章の参照情報

dSPACERelease 2015-B への移行

目的	Release 2015-B をインストールした後、いくつかの追加手順が必要な場合があります。
dSPACE Release 2015-A からの移行	**製品固有の移行手順**　製品ごとに必要な移行手順は、通常、製品ごとに自動的に実行されます。例外については、製品ごとの移行に関する説明を参照してください。
dSPACE Release 2014-B 以前のリリースからの移行	dSPACE Release 2014-B 以前のリリースから Release 2015-B への移行を行うには、その間の dSPACE Release バージョンへの移行手順も併せて実行する必要があります。Release 2015-B をインストールした状態で、移行に必要なすべての手順を実行することができます。

必要な移行手順の詳細については、各 dSPACE Release バージョンの『*新機能と移行手順*』ドキュメントを参照してください。

以前のリリースのドキュメント

以前のリリースの PDF ファイルの名前は、NewFeaturesAndMigrationxx.pdf（xx はリリース番号）です。

以前のリリースの『*新機能と移行手順*』は次の場所にあります。

- 最新の dSPACE HelpDesk インストールフォルダの C:\Program Files<(x86)>\Common Files\dSPACE\HelpDesk 2015-B\Print\PreviousReleases を参照してください。
- dSPACE DVD の場合は、\Doc\Print\PreviousReleases を参照してください。
- http://www.dspace.jp/goto.cfm/supver.rcphil からダウンロードしてください。ここには、かなり以前のリリースの『*新機能と移行手順*』もあります。

TRC ファイル生成の変更

項目の一覧

本章の内容

TRC ファイルの変更の基礎

目的

コード生成機能の強化は、実行アプリケーションのシミュレーション動作の改善につながります。dSPACE ソフトウエアでこのような改善を実現できるように、TRC ファイルの生成機能が強化されました。

生成される TRC ファイルの機能強化

MATLAB/Simulink R2014a リリースでは、シミュレーション動作を最適化するために Simulink® Coder™ により機能強化されたコード生成が導入されました。これにより、すべてのパラメータをより簡単な動作で調整でき、また参照先モデルがサポートされます。MATLAB R2015b で導入された Simulink Coder の追加機能により、dSPACE は機能強化された TRC ファイル生成を介してこれらの新機能を完全にサポートできるようになりました。

TRC ファイル生成の機能強化の主な利点を次に示します。

- MATLAB ワークスペースと TRC ファイルでモデルパラメータを同じように表示

  MATLAB ワークスペース変数で定義するすべての調整可能なモデルパラメータは、TRC ファイルの最上位の Tunable Parameters グループで使用することができます。これにより、グローバルパラメータに迅速に、モデル階層に関係なくアクセスすることができます。後でモデル階層を変更しても、レイアウト接続やテストスクリプトに指定済みの変数パスには影響しません。

- MATLAB 構造体の使用

  MATLAB 構造体が Simulink Coder 規則に従って調整可能な場合、構造体レベルと構造体フィールドがコード内に生成されます。

  これには、以下のような意味があります。

  - 構造化されたパラメータを TRC ファイルで使用することができます。
  - 非仮想 Simulink バスが、TRC ファイルでより効率的に再現されます。
  - TRC ファイルでバス配列を使用することができます。

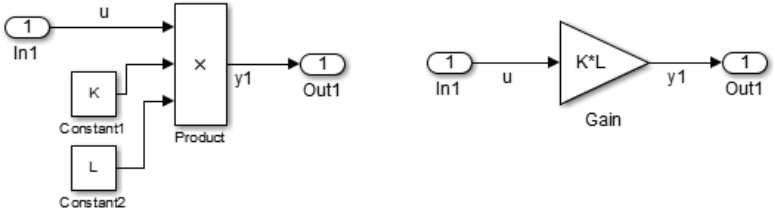
- パフォーマンスが大幅に向上

  非仮想 Simulink バスでは、コード生成とコンパイルのパフォーマンスが大幅に向上します。

- 調整可能なパラメータ式を使用するモデルのサイズ縮小

  パラメータ式をモデリングする複雑な解決策を、たとえば以下のモデルで示すように単純化することができます。MATLAB ワークスペース変数 K と L は、調整可能なパラメータとして自動的に生成されます。

  パラメータ式なし　　パラメータ式あり



- グローバルパラメータ[Default parameter behavior] = [Tunable]または [Inlined]（以前の[Inline Parameters]オプションのオフとオン）の処理

設定された調整可能なワークスペース変数と Simulink.Parameter オブジェクト間のマッピング、および生成されるコード内の変数は、[Default parameter behavior]オプション（以前の[Inline Parameters]オプション）には依存しません。

- モデル参照のサポートの改善

  Simulink 参照先モデルは、[Inline Parameters]オプションを[On]（MATLAB R2015b：[Default parameter behavior]を[Inlined]）に設定した使用に制限されていました。MATLAB R2015b を使用する場合、dSPACE ツールチェーンでは、[Default parameter behavior]オプションを[Tunable]に設定した参照先モデルの使用もサポートされるようになりました。

- Simulink マスクパラメータのサポート

  TRC ファイルに Simulink マスクパラメータを使用できるようになり、ControlDesk Next Generation などの dSPACE ソフトウエアでアクセスすることができます。

- Simulink シミュレーションと dSPACE プラットフォームで実行するシミュレーションの同じ動作

  上述した、パラメータ調整の一貫性に関わる機能強化の結果として、Simulink シミュレーションと dSPACE プラットフォームでのシミュレーションが同じ動作になります。

Simulink Coder 拡張のサポート

生成される TRC ファイルでのコード拡張をサポートするために、MathWorks 社と dSPACE は協力して追加のビルド機能を開発し、MATLAB R2015b でリリースしました。その結果として追加された TRC ファイル構文により、TRC ファイルを生成および使用するすべての dSPACE 製品で複雑な修正が必要になりました。

これらの拡張の完全なサポートは、dSPACE Release 2015-B と MATLAB R2015b を組み合わせて使用することで実現されます。dSPACE Release 2015-B を以前の MATLAB バージョンで使用する場合、コード生成は以前とほぼ同じです。

dSPACE リリースを変更しても、MATLAB リリースはそのままの場合、移行は必要ありません。

動作の違いについては、次の表を参照してください。

dSPACE Release 2015-B で使用する MATLAB リリース			
R2014a	**R2014b**	**R2015a**	**R2015b**
Simulink Coder のコード生成と dSPACE の TRC ファイル生成の動作は、MATLAB R2013b 以前での動作と同じです。	*構造化されていない* MATLAB ワークスペース変数によって、式を*使用せず*に定義されたブロックパラメータに対し、同じパラメータ変数を複数のブロック間で共有することができます。 他のすべてのブロックパラメータ定義は、以前と同じ動作になります。		前述した Simulink coder 機能の完全なサポート
RTI または Model Interface Package for Simulink を使用する前に、Simulink コマンド `revertInlineParametersOffToR2013b` を実行して、動作を強制する必要があります。	コード変更に合わせた内部調整が自動的に行われます。		標準の Simulink Coder 動作が使用されます。
[Inline Parameters]オプションを `Off` に設定した参照先モデルの使用はサポートされません。			[Inline Parameters]オプションを `Off` に設定した参照先モデルの使用がサポートされます。[1]

[1] MATLAB R2015b では、この設定は[Default parameter behavior]を `Tunable` に設定した場合と同じになります。

MATLAB R2015b で導入された TRC ファイル変更の詳細

dSPACE Release 2015-B および MATLAB R2015b では、次の変更が行われています。

Model Root グループ Model Root グループ内のエントリが次のように変更されました。

- パフォーマンスと操作性を向上させるために、Simulink 仮想バスと多重信号（`Out1{SubArray1}`など）のエントリは、変数記述ファイルに生成されなくなりました。

  このことは、これらの信号のラベルにも適用されます。

  これは、手作業での移行が必要な互換性のない変更です。「TRC ファイルを生成するソフトウエアでの変更の移行」(41 ページ)を参照してください。

- Simulink 非仮想バスのエントリは、変数記述ファイルに 1 つの構造体変数として生成されるようになりました。たとえば、`Out1{MyField}` は `Out1.MyField` に変更されました。

  これは、Simulink 非仮想バスのラベルにも適用されます。

これは、手作業での移行が必要な互換性のない変更です。「TRC ファイルを生成するソフトウエアでの変更の移行」(41 ページ)を参照してください。

- Simulink マスクパラメータは、関連するマスクサブシステムのエントリ時に変数記述ファイルに生成されるようになりました。
- 信号シンクブロックの入力信号は、ConfigurationDesk または VEOS をビルドプロセスに使用する場合にも、変数記述ファイルに生成されるようになりました。
- [Include states]および[Include derivatives]オプションを、ConfigurationDesk と VEOS にも使用できるようになりました。

Tunable Parameters グループ　Tunable Parameters グループ内のエントリは次のように変更されました。

- MATLAB ワークスペース変数と、モデル内でブロックパラメータとして使用される Simulink.Parameter オブジェクトは、Tunable Parameters グループ内のグローバル変数として生成されるようになりました。コード生成時の内部的な最適化により、変数は変数記述ファイルに生成されなくなります。

  ブロックのパラメータ定義に式が含まれる場合、ローカルブロックパラメータは使用できなくなります。これは、手作業での移行が必要な互換性のない変更です。「TRC ファイルを生成するソフトウエアでの変更の移行」(41 ページ)を参照してください。
- 構造化されたワークスペース変数とモデル内でブロックパラメータとして使用される Simulink.Parameter オブジェクトは、Tunable Parameters グループ内のグローバル構造体パラメータとして生成されるようになりました。

  構造体は、調整可能な構造体パラメータに関する Simulink Coder 条件を満たす必要があります。
- 以前は、階層を参照するモデルの各参照先モデルに、独自の Tunable Parameters グループがありました。このようなグループは生成されなくなりました。

  最上位モデルまたは参照先モデルで参照されるすべてのグローバルパラメータは、最上位モデルの Tunable Parameters グループに生成されます。

  これは、手作業での移行が必要な互換性のない変更です。「TRC ファイルを生成するソフトウエアでの変更の移行」(41 ページ)を参照してください。

n-D Lookup Table の処理　dSPACE Release 2015-B では、4x3x2 行列のような 2 次元を超える Lookup Table ブロックは 2 次元スライスに自動的に分割されなくなりました。

これは、手作業での移行が必要な互換性のない変更です。「TRC ファイルを生成するソフトウエアでの変更の移行」(41 ページ)を参照してください。

Data Stores グループ　パフォーマンスおよび他のブロックとのデータ整合性を改善するために、Data Stores グループは変数記述ファイルに生成されなくなりました。

これは、手作業での移行が必要な互換性のない変更です。「TRC ファイルを生成するソフトウエアでの変更の移行」(41 ページ)を参照してください。

構造体変数　非仮想バスや調整可能な構造体パラメータなどの構造体変数は、コード内に生成され変数記述ファイルに `struct` エレメントとして表示されます。構造体エレメント内のフィールドおよびメンバーの階層は、ドット表記で記述されます。例:`myStruct.mySubstruct.myValue[0][1].`.

リファレンス　変数記述ファイルに、ブロックパラメータがリファレンスとして含まれるようになりました。リファレンスのソースは、たとえば Tunable Parameters グループで使用可能な MATLAB ワークスペース変数などのグローバルパラメータや、マスクパラメータにすることができます。構造体パラメータの場合、リファレンスは構造体のフィールドを指定することができます。

構造体とリファレンスをサポートするために、TRC ファイル構文に次のキーワードが追加されました。

- `array-incr`
- `offs`
- `struct`
- `endstruct`
- `refvar`
- `refgroup`
- `refelem`
- `DEPRECATED`

これらのキーワードのいずれかを変数名として使用した場合、その変数名は TRC ファイルの生成中に検出されてファイルに追加されません。場合によっては、ユーザコード内の定義を確認する必要があります。そうしないと、その TRC ファイルを使用するソフトウエアでエラーが発生する可能性があります。

最新情報	TRC ファイル生成に関する詳細および最新の移行手順については、dSPACE の Web サイト：http://www.dspace.jp/go/trc を参照してください。

TRC ファイルを生成するソフトウエアでの変更の移行

目的	コード生成が複雑に変更されていますが、手作業での移行が必要なものはわずかです。機能強化に基づく変更の大部分は、dSPACE 製品によって自動的に移行されます。
MATLAB R2014a と dSPACE Release 2015-B を使用	Simulink Coder のコード生成と dSPACE の TRC ファイル生成の動作は、MATLAB R2013b 以前での動作と同じです。つまり、Simulink Coder の変更はいずれも有効になりません。 サポートには Simulink コマンド `revertInlineParametersOffToR2013b` が必要です。つまり、RTI でコードを生成する前、または ConfigurationDesk と VEOS で Model Interface Package for Simulink を使用する前にこのコマンドを実行する必要があります。このコマンドを実行するには、MATLAB の起動後に手作業で行うか、`startup.m` または `dsstartup.m` スクリプト内にこのコマンドを呼び出しとして追加します。 [Inline Parameters]オプションを `Off` に設定した参照先モデルの使用はサポートされません。
MATLAB R2014b と dSPACE Release 2015-B を使用	手作業での移行は不要です。変数記述ファイルには、Tunable Parameters グループ内に構造化されていないワークスペース変数の追加のグローバルパラメータが含まれます。これらのグローバルパラメータは、ブロックパラメータが式で定義されていない場合、対応するブロックパラメータと共有されます。新しい値をグローバルパラメータのいずれかに書き込むと、関連するブロックパラメータも変更されます。 [Inline Parameters]オプションを `Off` に設定した参照先モデルの使用はサポートされません。
MATLAB R2015a と dSPACE Release 2015-B を使用	注意事項は MATLAB R2014b の場合と同じです。

MATLAB R2015b と dSPACE Release 2015-B を使用

MATLAB R2015b を使用する場合、新しい Simulink Coder 機能は完全にサポートされます。互換性のない変更には、一般的には下記の移行手順が必要です。モデルの複雑さ、使用しているソフトウエア、テストスクリプトの内部構造など、さまざまな条件によって手順が異なるため、詳細な手順は示しません。そのため、一般的な移行方法を示す基本的な例のみを提供しています。

詳細については、http://www.dspace.jp/go/trc を参照してください。

TRC ファイルを生成するソフトウエアに必要な移行手順

RTI、ConfigurationDesk、VEOS など、TRC ファイルを生成する dSPACE 製品は、現状のままで新しい Simulink Coder の拡張をサポートしています。変数記述ファイル内の情報を提供するために、以下の変更のみ、手作業で移行する必要があります。

判定モードの更新（RTI のみ） rtiAssertionMode 変数は、変数記述ファイルに生成されなくなりました。ビルドプロセスを開始する前にモードを設定するには、[RTI simulation options]ページの[Assertion mode]設定を引き続き使用することができます。

Data Stores グループへのアクセスの更新 Data Stores グループは、変数記述ファイルに生成されなくなりました。Data Store Memory ブロックを使用する代わりに、読み取りアクセスには Data Store Read ブロック、書き込みアクセスには Constant ブロックと Data Store Write ブロックの組み合わせを使用する必要があります。Data Stores グループのエントリの代わりに、Model Root グループに Data Store Read ブロックまたは Constant ブロックのエントリが含まれます。

この移行手順では、dSPACE Release 2015-B を使用する必要はありません。それ以前のリリースの dSPACE を使用して行うこともできます。

TRC ファイルを使用するソフトウエアでの変更の移行

目的

ControlDesk Next Generation など、TRC ファイルを使用する製品では、生成された変数記述ファイルを使用して、ソフトウエア内のエレメントをシミュレーションアプリケーション内の変数と接続します。TRC ファイルの変更によって生じる変数パスの修正の大部分は、dSPACE 製品によって自動的に移行できますが、一部の変更については、ソフトウエア製品内で手作業で移行する必要があります。

TRC ファイルを使用するソフトウエアに必要な移行手順

ControlDesk、AutomationDesk、または変数パスを使用して変数にアクセスするあらゆる種類のテストスクリプトを既に使用しており、MATLAB R2015b を使用してシミュレーションアプリケーションを再ビルドする場合は、変数記述ファイル内の変数パスが廃止または変更されていないか確認する必要があります。

ControlDesk を使用している場合は、特別なマークが付いた計器や、Signal Editor の `Check Mapping` コマンドなどを利用して、不整合のある接続を見つけることができます。AutomationDesk では、変数アクセスは変数エイリアスを介して行われます。そのため、変数記述ファイルの修正を自動認識することはできません。ただし、プロジェクトで変数プールを使用している場合は、これを更新するだけで十分です。

構造体やリファレンスなど、新しい TRC ファイル機能をグラフィカルにサポートするために、ControlDesk と AutomationDesk は新しい Variable Browser を提供しています。

ソフトウエアで自動的に移行できない変更については、次のように手作業で移行する必要があります。

問題	移行手順
式を含むパラメータの変数パスを更新	ControlDesk 内の接続、または MATLAB ワークスペース変数、マスクパラメータ、Simulink Parameter オブジェクトを使用した式を含むテストスクリプトで定義された変数を更新します。 通常、制御に必要な変数アクセスを取得するには、変数パスを、生成されたグローバルパラメータに変更するだけでは不十分です。式のエレメントや結果として生成されるブロックの変数も考慮する必要があります。
Simulink 仮想バスの変数パスの更新	ControlDesk 内の接続および Simulink 仮想バス内の信号にアクセスするテストスクリプトを、直接アクセスされる信号ソースブロックに更新します。または、モデルに Bus Selector ブロックを追加して、ブロックの出力変数を接続します。
非仮想 Simulink バスの変数パスを更新	ControlDesk および非仮想 Simulink バス内の信号にアクセスするテストスクリプト内の接続を、構造体変数の対応するフィールドに更新します。 変数記述ファイルに以前生成されていた計測配列は、構造体エレメントで表示されます。  構造体エレメントの構文は、`Out1{myField.mySubField}` から `Out1.myField.mySubField` に変更されました。 これはドットを含む変数名と競合する可能性があります。
参照先モデルに対する Tunable Parameters グループの変数パスを更新	ControlDesk 内の接続または参照先モデルの調整可能なパラメータを参照する、テストスクリプトで定義された変数を更新します。このような変数の変数パスを、最上位の Tunable Parameters グループに変更する必要があります。
Data Stores グループへのアクセスの更新	ControlDesk および廃止された Data Stores グループの変数を参照するテストスクリプト内の接続を、モデルに挿入された Data Store Read または Write ブロックの変数に更新します。

問題	移行手順
ルックアップテーブルへの接続の更新	■ ControlDesk は TRC ファイル内のすべてのルックアップテーブルを認識するわけではありません。そのため、これらのルックアップテーブルを、たとえば、ControlDesk の Variable Browser でマップや曲線として使用することはできません。 ルックアップテーブルの認識は、次の場合には機能しません。 ■ ルックアップテーブルのテーブルデータが、構造体パラメータに含まれている。 ■ ルックアップテーブルのテーブルデータがマスクパラメータを参照している。 ■ ルックアップテーブルが 3 次元以上である。 このような場合、ルックアップテーブルへの接続を更新するには、ルックアップテーブルの個々の変数を接続します。 ■ tableData パラメータが数値でパラメータ化されている場合、ControlDesk では、ルックアップテーブルの tableData パラメータにマップや曲線を提供しなくなりました。 このようなパラメータへの接続を更新するには、ルックアップテーブルの LookUpTableData 変数を接続します（tableData パラメータの代わりに）。

MATLAB R2015b でシミュレーションアプリケーションを再ビルドし、アプリケーションの変数記述ファイルをリロードした後、ControlDesk は自動的に変数接続を移行します。ただし、R2014a より前の MATLAB Release でビルドされたシミュレーションアプリケーションの変数記述ファイルをリロードした場合、移行された変数接続が失われ、これらの接続を手作業で更新する必要があります。

TRC ファイル変更の詳細については、「TRC ファイルの変更の基礎」(35 ページ)を参照してください。

Python 2.7 ディストリビューションの変更点

目的

dSPACE が提供する Python ディストリビューションでの変更点について説明します。

Python の以前のバージョンから Python 2.7 に移行する場合は、dSPACE Release 2013-B の『*新機能と移行手順*』ドキュメントに記載される移行手順を参照してください。

この情報は、dSPACE の Web サイトでも入手することができます（http://www.dspace.com/ja/jpn/home/support/kb/supapnot/tatan/py27mig.cfm を参照）。

項目の一覧

本章の内容

Python 2.7 の主な変更点

目的

dSPACE Release 2015-B に付属している Python 2.7 ディストリビューションの変更点について説明します。

Python Software Foundation 提供のドキュメント『What's New』

Python のアップデートバージョンに関する『*What's New*』ドキュメントは、Python Software Foundation から入手することができます。

- 『What's New for Python 2.7』（http://docs.python.org/2.7/whatsnew/2.7.html）

dSPACE Python ディストリビューションの主な変更点

目的

dSPACE が提供する Python ディストリビューションには、dSPACE 固有の変更点がいくつか含まれています。

dSPACE Python ディストリビューションのコンポーネント

dSPACE DVD の Python 2.7 ディストリビューションは、次の Python コンポーネントを提供します。

Python コンポーネント	バージョン
Python Core	2.7.9
PyWin32	219.10
Numpy	1.7.1
Matplotlib	1.2.1
WxPython	2.9.4.0
Py2exe	0.6.9
Comtypes	0.6.2
PIL	1.1.7
Python for .NET	2.0p3

Python インストレーションの使用に関する一般情報

目的

コンピュータで Python の両バージョンを使用する場合には、次の内容が当てはまります。

Python 2.5 と Python 2.7 の並列使用

Python の両バージョンはコンピュータで並列使用できますが、次の制限事項があります。

- PY および PYW ファイルのファイル関連付けは、1 つの Python バージョンにのみ設定することができます。通常は、インストールされている中で最も新しい Python バージョンです。

- 環境変数は、Python の両バージョンによって使用されます。PYTHONHOME の値などの環境変数の値は、使用する Python インストレーションに設定する必要があります。Python が設定する環境変数の概要については、http://docs.python.org/2/using/cmdline.html を参照してください。

Python の両バージョンの並列動作による dSPACE テストオートメーションの使用

dSPACE Release 2013-A まで提供されていた dSPACE Python 2.5 セットアップまたは dSPACE Python Extensions セットアップにより配布された dSPACE Python モジュールをテストオートメーションスクリプトで使用し、スクリプトを移行しない場合は、Python の両バージョンを使用する必要があります。

標準の Python 2.7 ディストリビューションの機能拡張

目的

標準 Python 2.7 には dSPACE 固有の機能拡張が一部含まれています。これらは以前と同じ動作を確保するか、既知のバグが解決されています。次の機能拡張は dSPACE Release 2015-B で使用することができます。

Python の既知のバグを解決するための拡張

Python 2.7 の既知のバグを解決するために、次の変更が行われました。

- 以前のバージョンからの PyWin32 パッケージへの変更が採用されています。
- Python for .NET パッケージは、.NET 4.5.2 で実行できるように修正されています。
- Python for .NET パッケージは、WPF ユーザインターフェースで実行できるように修正されています。

Python 2.7 のバグとそれらの解決方法の最新情報については、http://bugs.python.org を参照してください。

dSPACE が変更した PyWin32 ファイルを識別するために、ファイルのバージョン番号を 219.0 から 219.10 に変更しています。

AutomationDesk

AutomationDesk 5.1 の新機能

本章の内容

一般的な機能強化

ユーザインターフェースの強化 AutomationDesk での作業を円滑にするため、ユーザインターフェースが次のように強化されています。

- 新しい Expression Editor を使用して、ASAM General Expression Syntax(GES)に従って条件を指定します(例:XIL API ライブラリで定義されたキャプチャタスクのトリガ条件)。
- Condition Editor は、新しい Expression Editor と同じ外観と操作性になるように更新されました。
- Signal Editor に次のような拡張が行われています。
  - 論理的な時間タグをグラフィカルに指定する新しい Time Tag 機能。このタグをセグメントに接続してトリガ条件を設定することができます。
  - Signal Editor のプロパティが更新されました。
  - 継承されたプロパティは別の色で表示されます。

- エディタのコマンドは、個別のリボンで表示されるようになりました。
- エディタで[Undo]および[Redo]操作がサポートされます。

- Library Favorites Viewer に次のような拡張が行われています。
  - ライブラリフォルダとそれに格納されたブロックやデータオブジェクトを一括してお気に入りリストに追加できるようになりました。
  - XML 形式でお気に入りリストをインポートおよびエクスポートすることができます。
  - エレメントを参照するお気に入りと Library Browser で使用できないお気に入りを検出できるようになり、別のアイコンと赤いテキストで表示されます。
- カスタムライブラリに関連する ADL ファイルをファイルエクスプローラから Library Browser にドラッグして、カスタムライブラリを開くことができます。

トレースファイル生成の機能強化のサポート　変数記述ファイルを介してシミュレーションアプリケーションにアクセスする AutomationDesk ライブラリは、トレースファイル生成機能の強化に伴う新しい機能をサポートしています。詳細については、「TRC ファイル生成の変更」(35 ページ)を参照してください。Platform データオブジェクトと MAPortConfiguration データオブジェクトをパラメータ設定するために使用する Variable Browser は、新しい TRC ファイル機能をサポートするように更新されました。

ライブラリの機能強化

次のライブラリが機能強化されました。

Dialogs ライブラリ　このライブラリは、Exec ブロックで使用できる次の新しいメソッドをサポートしています。

- `EnterGESExpression`

  Expression Editor を開きます。
- `EnterPythonExpression`

  Condition Editor を開きます。

詳細については、「Dialogs」(🕮『AutomationDesk Library Reference』)を参照してください。

XIL API ライブラリ　XIL API ライブラリは、Mapping データオブジェクトを提供するようになりました。これを使用して、シミュレーションアプリケーションで変数を XIL API ベースのフォーマットで提供することができます。XIL API マッピングには、抽象的な識別子としての変数名と、具体的な識別子としての変数パス(エイリアス)が含まれています。XIL API マッピングファイルを作成するには、新しい Mapping Editor を使用することができます。これは、Mapping データオブジェクトをダブルクリックし

たときに開く新しい Mapping Viewer のコンテキストメニューから実行することができます。

XIL API SignalGenerator エレメントを使用したスティミュラス生成が次のように拡張されました。

- MicroLabBox のサポート
- DS1007 PPC Processor Board でマルチコアまたはマルチプロセッサアプリケーションを使用する場合、接続された各アプリケーションプロセッサでモデル変数をスティミュレートすることができます。

詳細については、「XIL API」(🕮『AutomationDesk Library Reference』)を参照してください。

Signal-Based Testing ライブラリ Signal-Based Testing ライブラリのインプリメンテーションブロックは、Signal Editor の新しい時間タグ機能をサポートするように内部的に拡張されました。

詳細については、「Signal-Based Testing Library」(🕮『AutomationDesk Library Reference』)を参照してください。

COM API の機能強化

AutomationDesk COM API は、以下の点が機能強化されています。

- お気に入りライブラリをインポートおよびエクスポートする新しいメソッド
- XIL API マッピングを Mapping データオブジェクトにインポートする新しいメソッド

詳細については、🕮『AutomationDesk API Reference』を参照してください。

今後のバージョンでの廃止予定

次のライブラリ、オートメーションブロック、データオブジェクトは、AutomationDesk の今後のバージョンで廃止される予定です。

- Test Framework ライブラリ

  Test Framework ライブラリに基づいたプロジェクトは、Test Builder ライブラリに移行してください。移行については、http://www.dspace.jp/go/TestBuilderMigration を参照してください。

- Platform Access ライブラリ

  Platform Access ライブラリは、dSPACE Release 2016-A に付属するものが最後となります。Platform Access ライブラリをベースとしたプロジェクトは、XIL API ライブラリまたは XIL API Convenience ライブラリに移行してください。このライブラリは、接続されたプラットフォームの変数の読み取り、書き込み、およびスティミュレーションを行うための MAPort を提供します。

  移行については、http://www.dspace.jp/go/pscta を参照してください。

- ControlDeskNG Access ライブラリ内の欠陥シミュレーションオートメーションブロック

  ControlDesk の Failure Simulation Module は、dSPACE Release 2016-A に付属するものが最後となります。自動化によって電気的欠陥シミュレーションを準備するには、ControlDeskNG Access ライブラリ内の欠陥シミュレーションブロックではなく、XIL API ライブラリまたは XIL API Convenience ライブラリ内の電気的欠陥シミュレーションポート（EESPort）を使用します。

  移行については、http://www.dspace.com/go/pscta を参照してください。

- XIL API ライブラリ内の InitCaptureResultIDFReader および InitCaptureResultIDFWriter オートメーションブロック

  InitCaptureResultIDFReader および InitCaptureResultIDFWriter オートメーションブロックは、dSPACE Release 2016-A に付属するものが最後となります。IDF フォーマットは今後のバージョンで廃止される予定です。そのため、これらのオートメーションブロックは、MDF フォーマットをサポートする CaptureResultReader および CaptureResultWriter データオブジェクトに置き換えてください。詳細については、CaptureResultReader (Data Object) (🕮『AutomationDesk Library Reference』)および CaptureResultWriter (Data Object) (🕮『AutomationDesk Library Reference』)を参照してください。

廃止予定のエレメントは、ライブラリブラウザで特別なマーク付きで表示されます。

Automotive Simulation Model（ASM）

項目の一覧

本章の内容

他章の参照情報

ASM モデルの移行 (🕮『ASM ユーザガイド』)
ASM モデルの移行に関する一般的な説明を記載しています。

ASM Base InCylinder Blockset

項目の一覧

本章の内容

ASM Base InCylinder Blockset 2.1 の新機能

ENGINE_SETUP

ENGINE_SETUP ブロックに、ModelDesk でパラメータ設定するための以下のスイッチが追加されました。

- Const_max_num_PortInjector_PressureDrop
- Sw_Turbo_Stage[1SingleStage|2TwoStage]
- Sw_Turbo_Model [1phys|2LUT]
- Sw_ExhMan [1phys|2simple|3LUT]
- Sw_InMan[1phys|2LUT]

これまでは、CPT 構造体でパラメータ設定されていました。

Sw_Turbo_Model_[1phys|2LUT]と Sw_Turbo_Stage_[1SingleStage|2TwoStage]は新しい出力ポートです。

これらの変更は、turbocharger モデルに Turbo_Adv モデル、Turbo_2Stage モデル、またはその両方が含まれる場合に、エンジンモデルで効果を持ちます。

ASM Base InCylinder Blockset 2.1 への移行

ENGINE_SETUP

新しい パラメータと入力はダミー値に設定されます。移行したモデルでは、元の CPT 変数が引き続きスイッチに使用されます。

ASM Diesel Engine Blockset

項目の一覧

本章の内容

ASM Diesel Engine Blockset 2.2 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI' : DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO' : SCALEXIO ハードウエア
- 'VEOS' : VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

ENGINE_SETUP

ENGINE_SETUP ブロックに、ModelDesk でパラメータ設定するための以下のスイッチが追加されました。

- Const_max_num_PortInjector_PressureDrop
- Sw_Turbo_Stage[1SingleStage|2TwoStage]
- Sw_Turbo_Model [1phys|2LUT]

これまで、このパラメータ設定は、SWITCHES_TURBO ブロックで行われていました。

	Sw_Turbo_Model_[1phys|2LUT]と Sw_Turbo_Stage_[1SingleStage|2TwoStage]は新しい出力ポートです。これらの変更は、turbocharger モデルに Turbo_Adv モデル、Turbo_2Stage モデル、またはその両方が含まれる場合に、エンジンモデルで効果を持ちます。 エンジンモデルに含まれるターボチャージャモデルがコンポーネントを含む場合、ModelDesk の[Engine Setup]ページからターボモデルの切り替えを行うことができます。
COMMON_ENGINE_PARAMETERS	ガス燃料の計算用に、Const_kappa_Fuel パラメータが追加されています。 このために、次のブロックに対応する出力ポートが追加されました。 ■ kappa_Fuel[] ■ cv_Fuel[J|[kgK]] ■ cp_Fuel[J|[kgK]]
INJECTOR	このブロックでは、20 気筒での使用が修正されました。このブロックには、想定された気筒 20 ではなく気筒 19 にインジェクションを割り当てるバグがありました。
UNIT_INJECTOR	このブロックでは、20 気筒での使用が修正されました。このブロックには、想定された気筒 20 ではなく気筒 19 にインジェクションを割り当てるバグがありました。
PORTINJECTOR_TIMING	このブロックでは、インジェクタ燃料供給と吸気マニホールド間の差圧が小さすぎる場合に、燃料量を少なく計上するために、噴射時間を長く計算できるようになりました。 Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。 PORTINJECTOR_TIMING ブロックは、V 型エンジンなど、複数の吸気エンジンセットアップで使用することができます。Map_Inj2Cyl および Const_max_num_PortInjector_PressureDrop パラメータを使用して、吸気マニホールド、インジェクション、および気筒を割り当てます。
テストサイクル	Ftp_75 テストサイクルでのエンジンのスイッチオフフェーズが更新されました。1369 秒の合計時間後、エンジンは 10 分間オフになります。その後、エンジンはウォームリスタートされます。 さらに、3 つのクラス（ 動力対質量比に対応）を持つ新しいテストサイクル WLTC（Worldwide Harmonized Light Vehicle Test Procedure）が実装されました。新しいテストサイクルは、新しいエンジンデモの test cycles フォルダに格納されています。

ASM Diesel Engine Blockset 2.2 への移行

RAIL_CONTROL_ CRANKBASED	Const_disable_FMU_q_Inj、Const_enable_FMU_n_Engine、および Const_enable_FMU_q_Inj パラメータは、これらのしきい値が機能的な影響を与えることがなくなったため、削除されました。この変更による、モデルの動作に対する機能的な影響はありません。 Map_eta_DeltaAngle パラメータの名前が Map_phi_FMU_energized に変更されました。この変更による、モデルの動作に対する機能的な影響はありません。 フィードフォワードテーブルによる制御方式を改善するために、Map_phi_FMU_FF パラメータが追加されました。 ポンプサイクル内のコントロール変数の定義済みポンプ角度更新によってパルス制御モードを改善するために、Sw_phi_FMU_Update パラメータが追加されました。 バルブ操作で遅延を考慮するために、Map_ValveDelay パラメータが追加されました。 phi_FMU_energized 出力ベクトル内の非アクティブエレメントは、I/O 動作と並行して、999 に置き換えられるようになりました。999 のクランク角は、高圧ポンププラントモデルでは無効として解釈されます。その結果、対応するポンプサイクルは、高圧レールに燃料を提供しなくなります。
DPF_REGENERATION	MATLAB/Simulink でのデータ型の整合性チェックを考慮するためにデータ型の変換ブロックが挿入されました。この変更は機能への影響はなく、[Update diagram]時に MATLAB で警告が出力されないようにするだけです。
HPP_CRANKBASED	このブロックは、負の TDC オフセット定義も計上するように変更されました。 FMU 作動のないポンプサイクルを考慮するために、FMU 作動情報を含むベクトルが記録されるようになります。これに基づいて、ポンプブロックは現在のポンプサイクルに属する作動信号のみを考慮します。
ENGINE_SETUP	このブロックのパラメータは、ダミー値で初期化されます。新しい出力ポートは終了されます。
COMMON_ENGINE_ PARAMETERS	このパラメータは、ダミー値で初期化されます。新しい出力ポートは終了されます。

INJECTOR	このブロックでは、20 気筒での使用が修正されました。気筒 20 を想定しているインジェクションは、気筒 20 に正しく割り当てられるようになります。
UNIT_INJECTOR	このブロックでは、20 気筒での使用が修正されました。気筒 20 を想定しているインジェクションは、気筒 20 に正しく割り当てられるようになります。

ASM Diesel Exhaust Blockset

ASM Diesel Exhaust Blockset 2.1.1 への移行

DIESEL_OXIDATION_CATALYST	PT1 項は、圧力降下情報のみでなく出力圧力にも遅延を適用するために移動されました。この変更はシミュレーション動作に影響を及ぼします。そのため、移行時に、リンクが以前のバージョンのブロックに変更されます。

ASM Diesel InCylinder Blockset

項目の一覧

本章の内容

ASM Diesel InCylinder Blockset 2.1 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI'：DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO'：SCALEXIO ハードウエア
- 'VEOS'：VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

テストサイクル

Ftp_75 テストサイクルでのエンジンのスイッチオフフェーズが更新されました。1369 秒の合計時間後、エンジンは 10 分間オフになります。その後、エンジンはウォームリスタートされます。

さらに、3 つのクラス（ 動力対質量比に対応）を持つ新しいテストサイクル WLTC（Worldwide Harmonized Light Vehicle Test Procedure）が実装されました。新しいテストサイクルは、新しいエンジンデモの test cycles フォルダに格納されています。

ASM Diesel InCylinder Blockset 2.1 への移行

DPF_REGENERATION	MATLAB/Simulink でのデータ型の整合性チェックを考慮するために データ型の変換ブロックが挿入されました。変更による機能への影響は ありません。

ASM Drivetrain Basic Blockset

項目の一覧

本章の内容

ASM Drivetrain Basic Blockset 4.1.1 の新機能

CYCLES

CYCLES ブロックは、エンジンのスイッチオフフェーズによるテストサイクルの定義を受け入れるようになりました。エンジンのウォームリスタートをテストするために、テスト中にエンジンをオフにして再始動することができます。テストサイクル定義ファイルに新しい変数（Sw_Engine）を追加することができます。この変数は、エンジンをオフにする値 0 とオンにする値 1 を持ちます。

例として、新しいエンジンデモモデルの Ftp_75 テストサイクルを参照してください。ダイナモメータテストサイクルの実行時に、テストベンチがエンジンをオフにすることがないように、低いエンジン回転数も新しいパラメータとして実装されます。

ASM Drivetrain Basic Blockset 4.1.1 への移行

CYCLES

移行時に、低いエンジン回転数用の新しい Const_n_Engine_Min パラメータが追加されます。以前の動作が変更されないように、このパラメータのデフォルト値は 0 になっています。

ASM Electric Components Blockset

項目の一覧

本章の内容

ASM Electric Components Blockset 3.1 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI' : DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO' : SCALEXIO ハードウエア
- 'VEOS' : VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

ALTERNATOR

このブロックに、界磁電流用の新しい出力ポート : I_Excitation が追加されました。

ASM_EC_STARDELTA

これは、新しく追加されたブロックです。3 相電源変数から 3 相機械変数へ、およびその逆方向への電流（または電圧）変換に使用されます。

PMSM_D_Q_NONLINEAR

マシン構成を切り替えるために、新しいパラメータが追加されました。マシンをスター構成またはデルタ構成のどちらかでモデリングすることができます。

PMSM_MAGNET_SYNCHRONOUS_MACHINE_D_Q	マシン構成を切り替えるために、新しいパラメータが追加されました。マシンをスター構成またはデルタ構成のどちらかでモデリングすることができます。
SQUIRREL_CAGE_ASYNCHRONOUS_MACHINE_D_Q	マシン構成を切り替えるために、新しいパラメータが追加されました。マシンをスター構成またはデルタ構成のどちらかでモデリングすることができます。
PMSM_CONTROLLER	マシンをその構成に従って制御するために、新しいパラメータが追加されました。
PMSM_CONTROLLER_THREE_LEVEL	マシンをその構成に従って制御するために、新しいパラメータが追加されました。
SCIM_CONTROLLER	マシンをその構成に従って制御するために、新しいパラメータが追加されました。

ASM Electric Components Blockset 3.1 への移行

SOFT_ECU_HYBRID_MANAGER	ブロックは、サブコンポーネントブロック BRAKE_CONTROL、DRIVE_MANAGEMENT、TRQ_REQUEST_COORDINATION、KEY_SIGNALS_ICE、STARTER_ICE、CLUTCH_CONTROL および HYBRID_VEHICLE_SWITCH に分割されます。この変更は機能への影響はありませんが、SoftECU Hybrid Manager をより簡単に変更できるようになります。
入力ポートと出力ポートの名前変更	入力ポートと出力ポートの名前が次のように変更されています。 ■ v_Stator[a;b;c][V]は v[a;b;c][V]に変更 ■ i_Stator[a;b;c][V]は i[a;b;c][V]に変更 この変更は、以下のブロックに反映されています。 ■ PMSM_D_Q_NONLINEAR ■ BRUSHLESS_DC_MACHINE_ALPHA_BETA ■ PMSM_MAGNET_SYNCHRONOUS_MACHINE_D_Q ■ SQUIRREL_CAGE_ASYNCHRONOUS_MACHINE_D_Q ■ PMSM_CONTROLLER ■ PMSM_CONTROLLER_BASIC ■ PMSM_CONTROLLER_THREE_LEVEL

- SCIM_CONTROLLER
- SCIM_CONTROLLER_BASIC
- THREE_PHASE_INVERTER
- THREE_LEVEL_THREE_PHASE_INVERTER
- THREE_PHASE_DCM_INVERTER

入力ポートの名前変更

v_Stator[a;b;c][V]入力ポートの名前は v[a;b;c][V]に変更されました。
この変更は、以下のブロックに反映されています。

- SPACE_VECTOR_MODULATOR
- THREE_LEVEL_SPACE_ VECTOR_MODULATOR

i_Stator[a;b;c][V]入力ポートの名前は、i[a;b;c][V]に変更されました。
この変更は、以下のブロックに反映されています。

- BLDC_CONTROLLER
- BLDC_CONTROLLER_BASIC

ASM Environment Blockset

項目の一覧

本章の内容

ASM Environment Blockset 4.3 の新機能

LATERAL_CONTROL1/ LATERAL_CONTROL2	ステアリングホイールの角度は、グローバルリセットでリセットされます。

ASM Environment Blockset 4.3 への移行

LATERAL_CONTROL1/ LATERAL_CONTROL2	ステアリングホイールの角度は、グローバルリセットでリセットされます。
ROAD	距離計算は、動的な数のトラフィックオブジェクトに対して実行されます。そのため、対応する ROAD ブロックの Simulink ポートでは、幅が動的にサイジングされます。 Simulink の診断のために変更が加えられています。
BASIC_ROAD	Simulink の診断のために変更が加えられています。
MANEUVER_SCHEDULER	Simulink の診断のために変更が加えられています。
GEAR_SHIFTER	リセット後の動作が修正されました。

ASM Gasoline Engine Basic Blockset

項目の一覧

本章の内容

ASM Gasoline Engine Basic Blockset 2.0.2 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI' : DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO' : SCALEXIO ハードウエア
- 'VEOS' : VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

WALL_FILM

このブロックの Reset 入力ポートは無効でしたが、正しく機能するように修正されました。

PORTINJECTOR

このブロックでは、燃料供給と吸気マニホールドとの差圧が小さい場合に、少ない燃料量を計算できるようになりました。これは、圧縮天然ガスエンジンに使用されます。

Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。

COMMON_ENGINE_PARAMETERS	ガス燃料の計算用に、Const_kappa_Fuel パラメータが追加されています。 このために、次のブロックに対応する出力ポートが追加されました。 ■ kappa_Fuel[] ■ cv_Fuel[J/[kgK]] ■ cp_Fuel[J/[kgK]]
PORTINJECTOR_TIMING	このブロックでは、インジェクタ燃料供給と吸気マニホールド間の差圧が小さすぎる場合に、燃料量を少なく計上するために、噴射時間を長く計算できるようになりました。 Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。 PORTINJECTOR_TIMING ブロックは、V 型エンジンなど、複数の吸気エンジンセットアップで使用することができます。Map_Inj2Cyl および Const_max_num_PortInjector_PressureDrop パラメータを使用して、吸気マニホールド、インジェクション、および気筒を割り当てます。
テストサイクル	Ftp_75 テストサイクルでのエンジンのスイッチオフフェーズが更新されました。1369 秒の合計時間後、エンジンは 10 分間オフになります。その後、エンジンはウォームリスタートされます。 さらに、3 つのクラス（ 動力対質量比に対応）を持つ新しいテストサイクル WLTC（Worldwide Harmonized Light Vehicle Test Procedure）が実装されました。新しいテストサイクルは、新しいエンジンデモの test cycles フォルダに格納されています。

ASM Gasoline Engine Basic Blockset 2.0.2 への移行

WALL_FILM	リセットが意図した通りに機能するようになりました。
PORTINJECTOR	新しい機能の影響を受けないように、新しいパラメータと入力が設定されます。
ENGINE_SETUP	このパラメータは、ダミー値で初期化されます。新しい出力ポートは終了されます。

COMMON_ENGINE_ PARAMETERS	このパラメータは、ダミー値で初期化されます。新しい出力ポートは終了されます。
PORTINJECTOR_TIMING	新しい機能の影響を受けないように、新しいパラメータと入力が設定されます。

ASM Gasoline Engine Blockset

項目の一覧

本章の内容

ASM Gasoline Engine Blockset 3.2 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI'：DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO'：SCALEXIO ハードウエア
- 'VEOS'：VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

WALL_FILM

このブロックの Reset 入力ポートは無効でしたが、正しく機能するように修正されました。

PORTINJECTOR

このブロックでは、燃料供給と吸気マニホールドとの差圧が小さい場合に、少ない燃料量を計算できるようになりました。これは、圧縮天然ガスエンジンに使用されます。

Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。

DIRECTINJECTOR	このブロックでは、20 気筒での使用が修正されました。気筒 20 を想定したインジェクションが気筒 19 に割り当てられるバグがありました。
ENGINE_SETUP	ENGINE_SETUP ブロックに、ModelDesk でパラメータ設定するための以下のスイッチが追加されました。 ■ Const_max_num_PortInjector_PressureDrop ■ Sw_Turbo_Stage[1SingleStage|2TwoStage] ■ Sw_Turbo_Model [1phys|2LUT] これまで、このパラメータ設定は、SWITCHES_TURBO ブロックで行われていました。 Sw_Turbo_Model_[1phys|2LUT]と Sw_Turbo_Stage_[1SingleStage|2TwoStage]は新しい出力ポートです。これらの変更は、turbocharger モデルに Turbo_Adv モデル、Turbo_2Stage モデル、またはその両方が含まれる場合に、エンジンモデルで効果を持ちます。 エンジンモデルに含まれるターボチャージャモデルがコンポーネントを含む場合、ModelDesk の[Engine Setup]ページからターボモデルの切り替えを行うことができます。
COMMON_ENGINE_ PARAMETERS	ガス燃料の計算用に、Const_kappa_Fuel パラメータが追加されています。 このために、次のブロックに対応する出力ポートが追加されました。 ■ kappa_Fuel[] ■ cv_Fuel[J|[kgK]] ■ cp_Fuel[J|[kgK]]
PORTINJECTOR_TIMING	このブロックでは、インジェクタ燃料供給と吸気マニホールド間の差圧が小さすぎる場合に、燃料量を少なく計上するために、噴射時間を長く計算できるようになりました。 Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。 PORTINJECTOR_TIMING ブロックは、V 型エンジンなど、複数の吸気エンジンセットアップで使用することができます。Map_Inj2Cyl および Const_max_num_PortInjector_PressureDrop パラメータを使用して、吸気マニホールド、インジェクション、および気筒を割り当てます。
新しいブロック	次のブロックが新しく追加されています。 ■ CNG_PRESSURE_REGULATOR ■ CNG_SHUTOFF_VALVE

- CNG_HIGH_PRESSURE_LINE
- CNG_TANK
- CNG_RAIL

テストサイクル

Ftp_75 テストサイクルでのエンジンのスイッチオフフェーズが更新されました。1369 秒の合計時間後、エンジンは 10 分間オフになります。その後、エンジンはウォームリスタートされます。

さらに、3 つのクラス（ 動力対質量比に対応）を持つ新しいテストサイクル WLTC（Worldwide Harmonized Light Vehicle Test Procedure）が実装されました。新しいテストサイクルは、新しいエンジンデモの test cycles フォルダに格納されています。

ASM Gasoline EngineBlockset 3.2 への移行

RAIL_CONTROL_CRANKBASED

Const_disable_FMU_q_Inj、Const_enable_FMU_n_Engine、および Const_enable_FMU_q_Inj パラメータは、これらのしきい値が機能的な影響を与えることがなくなったため、削除されました。この変更による、モデルの動作に対する機能的な影響はありません。

Map_eta_DeltaAngle パラメータの名前が Map_phi_FMU_energized に変更されました。この変更による、モデルの動作に対する機能的な影響はありません。

フィードフォワードテーブルによる制御方式を改善するために、Map_phi_FMU_FF パラメータが追加されました。

ポンプサイクル内のコントロール変数の定義済みポンプ角度更新によってパルス制御モードを改善するために、Sw_phi_FMU_Update パラメータが追加されました。

バルブ操作で遅延を考慮するために、Map_ValveDelay パラメータが追加されました。

phi_FMU_energized 出力ベクトル内の非アクティブエレメントは、I/O 動作と並行して、999 に置き換えられるようになりました。999 のクランク角は、高圧ポンププラントモデルでは無効として解釈されます。その結果、対応するポンプサイクルは、高圧レールに燃料を提供しなくなります。

HPP_CRANKBASED	このブロックは、負の TDC オフセット定義も計上するように変更されました。 FMU 作動のないポンプサイクルを考慮するために、FMU 作動情報を含むベクトルが記録されるようになります。これに基づいて、ポンプブロックは現在のポンプサイクルに属する作動信号のみを考慮します。
PORTINJECTOR_TIMING	新しい機能の影響を受けないように、新しいパラメータと入力が設定されます。
PORTINJECTOR	新しい機能の影響を受けないように、新しいパラメータと入力が設定されます。
関連トピック	基礎 • 「ASM モデルの移行」(📖『ASM ユーザガイド』)

ASM Gasoline InCylinder Blockset

項目の一覧

本章の内容

ASM Gasoline InCylinder Blockset 2.1 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI'：DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO'：SCALEXIO ハードウエア
- 'VEOS'：VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

PORTINJECTOR_TIMING

このブロックでは、インジェクタ燃料供給と吸気マニホールド間の差圧が小さすぎる場合に、燃料量を少なく計上するために、噴射時間を長く計算できるようになりました。

Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。

	PORTINJECTOR_TIMING ブロックは、V 型エンジンなど、複数の吸気エンジンセットアップで使用することができます。Map_Inj2Cyl および Const_max_num_PortInjector_PressureDrop パラメータを使用して、吸気マニホールド、インジェクション、および気筒を割り当てます。
PORTINJECTOR	このブロックでは、燃料供給と吸気マニホールドとの差圧が小さい場合に、少ない燃料量を計算できるようになりました。これは、圧縮天然ガスエンジンに使用されます。 Map_mdot_Fuel_Gain_Red パラメータが追加されました。燃料差圧計算用に、2 つの新しい入力ポート、p_InMan[Pa]と p_Fuel[Pa]が作成されました。
テストサイクル	Ftp_75 テストサイクルでのエンジンのスイッチオフフェーズが更新されました。1369 秒の合計時間後、エンジンは 10 分間オフになります。その後、エンジンはウォームリスタートされます。 さらに、3 つのクラス（ 動力対質量比に対応）を持つ新しいテストサイクル WLTC（Worldwide Harmonized Light Vehicle Test Procedure）が実装されました。新しいテストサイクルは、新しいエンジンデモの test cycles フォルダに格納されています。

ASM Gasoline InCylinder Blockset 2.1 への移行

PORTINJECTOR_TIMING	新しい機能の影響を受けないように、新しいパラメータと入力が設定されます。
PORTINJECTOR	新しい機能の影響を受けないように、新しいパラメータと入力が設定されます。

ASM Optimizer

項目の一覧

本章の内容

ASM Optimizer 1.7 の新機能

データ、実行、後処理の分離

データ、実行、後処理の設定はより明確に分離されます。このため、いくつかの入力が別のページに配置されるようになりました。軸定義は、CalcEOPC スクリプトから後処理の個別の設定ファイルに移動されています。

ModelDesk の General Settings ファイルは再利用することができます。

更新動作の改善

更新プロセスが自動的に実行されます。たとえば、マッピングを変更すると、OPData 構造体の再生成が自動的にトリガされます。

ModelDesk 計測データファイルのインポート

ModelDesk 計測データファイル（*.md）は、平均値計測およびインポートのマッピングのソースとして使用することができます。

再起動動作の改善

傾斜最適化アルゴリズムの結果は、開始条件によって異なります。各タスクの各運転ポイントがロードまたは最適化されるようになりました。これにより、最適化を繰り返し実行しても同じ結果が得られます。

後処理の再設計

後処理ページは、最適化タスク処理と同様に再設計されました。グローバルな後処理設定を維持するために、[Manage Postprocessing]ページが導入されています。各最適化タスクに対して、個別の[Postprocessing Task]ページが作成されます。

最適化後のファンクション

各タスクの後にファンクションを追加することができます。これらのファンクションを使用して、OPData を変更することができます。たとえば、タスク結果をさらにプロットに使用することもできます。

ASM Optimizer Blockset 1.7 への移行

全般	すべての設定は新しい場所に転送されます。EOPV 計算の既存の軸定義を使用して、Settings ファイルが作成されます。既存のポストコマンドでは、Post Optimization Function が生成されます。結果の構造体が変更されるため、最適化が開始されると既存の結果は削除されます。

ASM Traffic Blockset

項目の一覧

本章の内容

ASM Traffic Blockset 3.3 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI'：DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO'：SCALEXIO ハードウエア
- 'VEOS'：VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

FUEL_CONSUMPTION

燃料消費量と二酸化炭素排出量をシミュレートするために、FUEL_CONSUMPTION ブロックが ENGINE サブシステムに追加されました。計算は概算です。デモの焦点はビークルダイナミクスであるためです。燃料消費量はブレーキ固有の燃料消費量マップに基づいて計算されます。これが燃費の計測になります。

このブロックは新しいデモのみに実装され、移行中に自動的に追加されません。このブロックを使用するには、それをライブラリからドラッグして関連するポートに接続します。

ASM Traffic Blockset 3.3 への移行

CUSTOM_SENSOR_ SCOPEZONE	インプリメンテーションが変更されました。Selector ブロックではなく、Demux ブロックが使用されます。
CUSTOM_SENSOR_OUTPUT	インプリメンテーションが変更されました。Selector ブロックではなく、Demux ブロックが使用されます。
TRAFFIC_SCHEDULER	ASMSignalBus が変更されました。詳細については、「Traffic Scheduler」(🕮『ASM Traffic Reference』)を参照してください。
FELLOW_PARAMETERS	PosVec_mainPnt_Fellows[x;y;z][m]出力ポートの幅が、ワークスペース変数によって定義されるようになりました。
SENSOR_POSITION	Simulink の診断のために変更が加えられています。
COORDINATE_ TRANSFORMATION	Simulink の診断のために変更が加えられています。
NEAREST POINT	Simulink の診断のために変更が加えられています。
FELLOW_POSITION	Simulink の診断のために変更が加えられています。
NEAREST SURFACE	Simulink の診断のために変更が加えられています。
RADARSENSOR_3D	Simulink の診断のために変更が加えられています。

ASM Trailer Blockset

項目の一覧

本章の内容

ASM Trailer Blockset 2.4 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI'：DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO'：SCALEXIO ハードウエア
- 'VEOS'：VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

SUSCOMP_RIGID_SYM_xxx

ブロックに新しいパラメータが追加されました：Map_Angle_Gamma_Axle_F_y。これは、左右方向の力を考慮するために、軸回転を行います。

ASM Trailer Blockset 2.4 への移行

SUSCOMP_RIGID_SYM_xxx	新しいパラメータが追加されました：Map_Angle_Gamma_Axle_F_y。
SUSKIN_RIGID_SYM_xxx	テイラー展開の arcsin 計算でのバグ（3 次テイラー級数での符号の間違い）が修正されました。

ASM Truck Blockset

項目の一覧

本章の内容

ASM Truck Blockset 2.3 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI'：DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO'：SCALEXIO ハードウエア
- 'VEOS'：VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

SUSCOMP_RIGID_SYM_xxx

ブロックに新しいパラメータが追加されました：Map_Angle_Gamma_Axle_F_y。これは、左右方向の力を考慮するために、軸回転を行います。

Drivetrain

Drivetrain モデルは、1 つの前輪駆動軸と 2 つの後輪駆動軸を持つ 6x6 構成に変更されました。新しいデモでは、リジッドドライブトレインアプローチが使用されています。2 つのセントラル、1 つのフロント、2 つのリアディファレンシャルで新しいドライブトレイン設定を構築します。このために、ASM ブロックのマルチインスタンス機能が使用されています。

新しい設定は、新しいデモでのみ使用することができます。ただし、ビークルダイナミクスデモモデルの以前の柔軟なドライブトレイン設定は引き続き使用することができます。

ASM Truck Blockset 2.3 への移行

SUSCOMP_RIGID_SYM_xxx	新しいパラメータが追加されました：Map_Angle_Gamma_Axle_F_y。
SUSKIN_RIGID_SYM_xxx	テイラー展開の arcsin 計算でのバグ（3 次テイラー級数での符号の間違い）が修正されました。

ASM Turbocharger Blockset

項目の一覧

本章の内容

ASM Turbocharger Blockset 3.1.1 の新機能

POSTTURBHPMAN

このブロックには、2 つのターボチャージャの使用時に流量割合を一定化するための新しい出力ポートが追加されました。

- Xsi_Fuel_PostTurbHPMan[0_1]
- Xsi_Air_PostTurbHPMan[0_1]
- Xsi_Exh_PostTurbHPMan[0_1]

TURBINE

Sw_TurbineType 入力ポートは削除されました。可変タービンジオメトリ（VTG）の作動によってタービンを制御するかどうかを設定するために、Sw_Ctrl_VTG_On パラメータが追加されました。VTG 作動信号用の入力ポートは、位置信号を待機するようになりました。

タービンを通した逆流を回避するために、パワーバランスが修正されました。

TURBINE_HP

Sw_TurbineType 入力ポートは削除されました。可変タービンジオメトリ（VTG）の作動によって高圧タービンを制御するかどうかを設定する Sw_Ctrl_VTG_On パラメータが追加されました。VTG 作動信号用の入力ポートは、位置信号を待機するようになりました。

タービンを通した逆流を回避するために、パワーバランスが修正されました。

TURBINE_SAEJ922

このブロックは、ASM Gasoline Engine にも使用できるようになりました。異なるタービンデータとの互換性を高めるために、関数が修正されました。

Sw_TurbineType 入力ポートは削除されました。可変タービンジオメトリ（VTG）の作動によって高圧タービンを制御するかどうかを設定する Sw_Ctrl_VTG_On パラメータが追加されました。VTG 作動信号用の入力ポートは、位置信号を待機するようになりました。

	変換スケーリングをカスタマイズするために、Sw_mdot_Conv、Const_ConvScaling_mdot、Sw_omega_TC_Conv、および Const_ConvScaling_omega パラメータが追加されました。 タービンの場合の熱流束を考慮するために、Map_HeatLoss_Turb_Rel パラメータが新たに追加されました。
WASTEGATE_VALVE	Sw_TurbineType 入力ポートは削除されました。 ウェストゲートを制御するかどうかを設定するために、Sw_Ctrl_WGate_On パラメータが追加されました。ウェストゲート作動信号用の入力ポートは、位置信号を待機するようになりました。
WASTEGATE_VALVE_HP	Sw_TurbineType 入力ポートは削除されました。 ウェストゲートを制御するかどうかを設定するために、Sw_Ctrl_WGate_On パラメータが追加されました。ウェストゲート作動信号用の入力ポートは、位置信号を待機するようになりました。
TURBO_BASIC	MAPS_TC ブロックの名前が TURBO_BASIC に変更されました。 Sw_Ctrl_TC 入力ポートが削除され、Sw_Ctrl_TC パラメータとして追加されました。これはウェストゲートか VTG を使用するか、または作動信号を使用しないかを決定するために使用されます。
TURBO_BASIC_2STAGE	MAPS_TC_2STAGE ブロックの名前が TURBO_BASIC_2STAGE に変更されました。 Sw_Ctrl_TC 入力ポートが削除され、Sw_Ctrl_TC パラメータとして追加されました。これはシステムが入力作動信号を使用するかどうかを決定するために使用されます。

ASM Turbocharger Blockset 3.1.1 への移行

以前のバージョン	次のシステムは、移行時に以前のバージョンに移動されます。 ■ POSTTURBHPMAN ■ TURBINE ■ TURBINE_HP ■ TURBINE_SAEJ922 ■ WASTEGATE_VALVE ■ WASTEGATE_VALVE_HP ■ TURBO_BASIC

- TURBO_BASIC_2STAGE
- SWITCHES_TURBO_2_0

ASM Vehicle Dynamics Blockset

項目の一覧

本章の内容

ASM Vehicle Dynamics Blockset 3.2 の新機能

モデルの開始スクリプト go.m ファイル

ASM デモモデルの標準開始スクリプト go.m ファイルは、サポートされるシミュレーションプラットフォームの変更を組込むために調整されました。

修正された go.m ファイルには、次のプラットフォームオプションが含まれます。

- 'RTI' : DS1005、DS1006、DS1007、MicroLabBox
- 'SCALEXIO' : SCALEXIO ハードウエア
- 'VEOS' : VEOS（オフラインシミュレーションプラットフォーム）

go.m ファイルの呼び出し手順の詳細については、対応するファイルヘッダまたは「モデルの初期化」(🕮『ASM ユーザガイド』)を参照してください。

開始スクリプト go.m ファイルには、VEOS ターゲットの変更も組込まれています。Release 2015-B では、VEOS は DSOffSim ターゲットの代わりに DSRT ターゲットを使用します（「VEOS 3.5 への移行」(252 ページ)を参照）。ASM デモプロジェクト内の go.m ファイルは、正しいコンパイラ設定をトリガするように調整されました。

SUSCOMP_RIGID_SYM_xxx

ブロックに新しいパラメータが追加されました : Map_Angle_Gamma_Axle_F_y。これは、左右方向の力を考慮するために、軸回転を行います。

FUEL_CONSUMPTION

燃料消費量と二酸化炭素排出量をシミュレートするために、FUEL_CONSUMPTION ブロックが ENGINE サブシステムに追加されました。計算は概算です。デモの焦点はビークルダイナミクスであるためです。燃料消費量はブレーキ固有の燃料消費量マップに基づいて計算されます。これが燃費の計測になります。

このブロックは新しいデモのみに実装され、移行中に自動的に追加されません。このブロックを使用するには、それをライブラリからドラッグして関連するポートに接続します。

ASM Vehicle Dynamics Blockset 3.2 への移行

VEHICLE_MOVEMENT_INFO_CAR	車両横滑り角の計算で"Not a number avoidance"が削除されました。この計算には Simulink の atan2 関数が使用されます。そのため、ゼロ除算は既に回避されています。
SUSCOMP_RIGID_SYM_xxx	新しいパラメータが追加されました：Map_Angle_Gamma_Axle_F_y。
SUSKIN_RIGID_SYM_xxx	テイラー展開の arcsin 計算でのバグ（3 次テイラー級数での符号の間違い）が修正されました。
SOFT_ECU_POWERSTEERING	"Bus signal treated as vector"警告を回避するために、Bus2Vector ブロックが追加されました。
STEERING_3DOF_VARIABLE_RATIO	"Bus signal treated as vector"警告を回避するために、Bus2Vector ブロックが追加されました。 ステアリングロッドやステアリングコラムでの指数によるバネ摩擦要素の計算が、"Not a Number"の生成を回避するように修正されました。
STEERING	"Bus signal treated as vector"警告を回避するために、Bus2Vector ブロックが追加されました。
STEERING_VARIABLE_RATIO	"Bus signal treated as vector"警告を回避するために、Bus2Vector ブロックが追加されました。

ConfigurationDesk

目的	ConfigurationDesk は、さまざまなシナリオに適用できるツールです。リアルタイムアプリケーションの実装や、RapidPro ハードウエアの設定を行うことができます。

ConfigurationDesk – Implementation

項目の一覧

本章の内容

ConfigurationDesk 5.4（Implementation Version）の新機能

トレースファイル生成の機能強化のサポート

ConfigurationDesk は、MATLAB R2014a で導入された Simulink Coder のパラメータ処理に関する機能をサポートしています。

詳細については、「TRC ファイル生成の変更」(35 ページ)を参照してください。

プリコンパイル済み SIC ファイルのサポート

ConfigurationDesk では、Simulink インプリメンテーションコンテナファイル（SIC ファイル）を ConfigurationDesk アプリケーションに追加することができます。SIC ファイルは、ビヘイビアモデルコードを含むコンテナファイルです。ConfigurationDesk では、SIC ファイルをプリコンパイルするメソッドが追加されました。次のようなパターンが想定されます。

- 読み取り可能なソースファイルのないプリコンパイル済み SIC ファイル

  ConfigurationDesk では、ソースファイルのある SIC ファイルを、ソースファイルはないが IP 保護に適した SCALEXIO 互換のライブラリファイルがある SIC ファイルに変換することができます。

- 元のソースファイルを含むプリコンパイル済み SIC ファイル

  ConfigurationDesk では、元のソースファイルを含めたままプリコンパイル済み SIC ファイルを作成することができます。これは、VEOS Player で使用する場合に役立ちます。

このようなプリコンパイル済み SIC ファイルを使用すると、ビルドプロセスが高速化されます。変換した SIC ファイルを ConfigurationDesk アプリケーションに追加することができます。詳細については、「Creating Precompiled SIC Files」(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

保護された Simulink モデルのサポート

ConfigurationDesk では、Simulink ビヘイビアモデルを含むリアルタイムアプリケーションや、保護された参照先モデルを含む Simulink インプリメンテーションコンテナをビルドすることができます。「Model Interface Package for Simulink 3.1 の機能」(147 ページ)を参照してください。

FMU サポートの新機能

同一データポート名を持つ複数の FMU のサポート
ConfigurationDesk では、FMU のモデル名とそのポート名、および FMU のモデル記述ファイルで定義されたその階層構造を使用して、FMU の各ポートを識別できるようになりました。このため、異なる FMU のポートは、たとえ名前が同じであっても ID が異なります。このため、次のようになります。

- 同じポート名の異なる FMU を、[Model port block: Duplicate ID]コンフリクトを作成せずに ConfigurationDesk アプリケーションに追加することができます。
- FMU をポート名は同じでモデル名が異なる FMU と置き換えると、元の FMU で提供されて ConfigurationDesk アプリケーションで使用されているすべてのポートが未解決になります。

FMU のインポート:列挙のサポート ConfigurationDesk では、FMU で列挙データ型をサポートするようになりました。このため、次のことが可能になります。

- 列挙データ型の FMU 入出力を、ConfigurationDesk でデータポートとして使用することができます。
- 列挙データ型の FMU 変数を、TRC ファイルで使用することができます。

詳細については、「Integrating Functional Mock-up Units in ConfigurationDesk」(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

V-ECU サポートの新機能

サポートされる V-ECU インプリメンテーションコンテナのバージョン 次の表に、V-ECU インプリメンテーションコンテナをエクスポートするツールのバージョンと、関連するコンテナのバージョンを示します。

V-ECU インプリメンテーションを作成した製品	V-ECU インプリメンテーションのバージョン
dSPACE Release 2013-B 以前: ■ SystemDesk 3.x ■ TargetLink 3.5	1.0
dSPACE Release 2014-A: ■ SystemDesk 4.2	2.0

V-ECU インプリメンテーションを作成した製品	V-ECU インプリメンテーションのバージョン
dSPACE Release 2014-B: ■ SystemDesk 4.3 ■ TargetLink 4.0	2.1
dSPACE Release 2015-A: ■ SystemDesk 4.4	2.2
dSPACE Release 2015-B: ■ SystemDesk 4.5 ■ TargetLink 4.1	2.3

同一ポート名を持つ複数の V-ECU インプリメンテーションのサポート
ConfigurationDesk は、V-ECU インプリメンテーション名、ポート名、V-ECU インプリメンテーション内でのポートのパスを使用して、各 V-ECU インプリメンテーションポートを識別するようになりました。そのため、異なる V-ECU インプリメンテーション内のポートは、名前と V-ECU インプリメンテーション内でのパスが同じであっても、異なる ID を持ちます。このため、次のようになります。

- 同じポート名を持つ異なる V-ECU インプリメンテーションを、[Model port block: Duplicate ID]コンフリクトを作成せずに ConfigurationDesk アプリケーションに追加することができます。
- V-ECU インプリメーションを、ポート名が同じで V-ECU インプリメーション名が異なる V-ECU インプリメーションと置き換えると、元の V-ECU インプリメーションで提供されて ConfigurationDesk アプリケーションで使用されているすべてのポートが未解決になります。

シグナルチェーンの拡張時のデータ型変換

デジタルファンクションポートのデータ型変換 [ConfigurationDesk Options]ダイアログの [Configuration]ページにあるグローバルな [Extend signal chain]オプションには、次の事項が適用されます。

[Model port data type]設定を[Inherited]に設定した場合、シグナルチェーン拡張時に、値範囲[0|1]の整数データ型を持つデジタルファンクションポートに、Boolean データ型のモデルポート、たとえばデジタル信号が作成されます。

ユーザ指定の飽和値の範囲 [Extend signal chain]コマンドを使用するときに、独自の飽和値を指定した場合は、次の事項が適用されます。

- 指定した値範囲は、[Extend Signal Chain]コマンドで生成したモデルポートの[Unit]プロパティに書き込まれます。
- 指定した値範囲は、ビルドプロセス中に生成された TRC ファイル内のファンクションブロックの値範囲に書き込まれます。

拡張シグナルチェーンの飽和規則 ConfigurationDesk では、ファンクションポートにユーザ飽和値を指定することができます。[System

min/max values]を使用する場合、必要に応じて、モデルポートとファンクションポート間でデータ型変換が自動的に実行されます。ファンクションポートに独自の[User min/max values]を指定して、[Extend Signal Chain] - [Create Suitable Model Port Block]コマンドを使用した場合、ファンクション入力ポートとファンクション出力ポートの値範囲に特定の飽和規則が適用されます。

「Specifying Global Options for Extending the Signal Chain」(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

拡張されたファンクションブロックタイプ

Current In、Voltage In Current In および Voltage In ファンクションブロックには、次の新しい機能があります。

- 角度位置の取得。角度位置の値は、電流／電圧値の計測と同期して取得されます。
- タイムスタンプおよび角度位置データの取得を有効化または無効化して計算時間を短縮

詳細については、「Configuring the Basic Functionality (Current In)」(🕮『ConfigurationDesk I/O Function Implementation Guide』)または「Configuring the Basic Functionality (Voltage In)」(🕮『ConfigurationDesk I/O Function Implementation Guide』)を参照してください。

Current Signal Capture、Voltage Signal Capture Current Signal Capture および Voltage Signal Capture ファンクションブロックには、次の新しい機能があります。

- モデルステップで取得したすべてのシーケンスのすべてのサンプルに対する角度位置の取得
- 取得した各シーケンス全体に対する開始角度位置の取得
- 角度位置および開始角度位置の取得を有効化または無効化して計算時間を短縮

詳細については、Configuring the Basic Functionality (Current Signal Capture)または「Configuring the Basic Functionality (Voltage Signal Capture)」(🕮『ConfigurationDesk I/O Function Implementation Guide』)を参照してください。

CAN CAN ファンクションブロックは、DS2671 Bus Board の CAN FD モードをサポートするようになりました。CAN FD フレームは、Bosch 社が開発した *Non-ISO CAN FD プロトコル*または 2015 年末にリリース予定の次期 ISO 11898-1:2015 規格に準拠した *ISO CAN FD プロトコル*のいずれかに準拠する必要があります。

詳細については、「CAN」(🕮『ConfigurationDesk I/O Function Implementation Guide』)を参照してください。

Bus Manager の新機能

通信マトリクスの変更 Bus Manager では、通信マトリクスのエレメントにユーザ定義の設定を指定できるようになりました。定義された設定は、アクティブな ConfigurationDesk アプリケーションにのみ適用されます。これにより、通信マトリクスのエレメントを、たとえば、元の通信マトリクスを変更せずに使用できるようになります。詳細については、「Specifying User-Defined Settings for Communication Matrix Elements」(📖『ConfigurationDesk Bus Manager Implementation Guide』)を参照してください。

LIN マスターの LIN Schedule Table ファンクションポート Bus Manager では、バス設定に割り当てられた各 LIN マスターに対して LIN Schedule Table ファンクションポートが提供されるようになりました。LIN Schedule Table ファンクションポートにより、初期のスケジュールテーブルを指定して、実行中にアクティブなスケジュールを切り替えることができます。詳細については、「Working With LIN Schedule Tables」(📖『ConfigurationDesk Bus Manager Implementation Guide』)を参照してください。

新しいフィルタ機能

事前設定された表示セット ConfigurationDesk をインストールすると、次のような表示セット 1 と 2 が事前設定されます。

- Standard

  この表示セットは、ConfigurationDesk のデフォルトの画面配置で、大半の用途および画面解像度にはこれが適しています。

- Small Screen Resolution

  この表示セットは、ノート PC のディスプレイなど、画面解像度が比較的小さいディスプレイの作業領域でシグナルチェーンを設定する場合に最適な画面配置です。

自動化インターフェースの新機能

ConfigurationDesk の自動化インターフェースが拡張され、ConfigurationDesk 機能のサポートが強化されています。たとえば、Bus Manager エレメントに XPath 式を使用してアクセスできるようになりました。XPath 式は、XPath 1.0 に準拠している必要があります。

機能拡張および変更の詳細については、「Changes to the Automation Interface from Release 2015-A」(📖『ConfigurationDesk Automating Tool Handling』)を参照してください。

新しいドキュメント機能

ConfigurationDesk 用語の用語解説 用語解説では、ConfigurationDesk ドキュメントで使用した重要な表現および命名規則を簡単に説明します。「ConfigurationDesk Glossary」を参照してください。

カスタムファンクションブロックのドキュメント　カスタムファンクションブロックの作成に関するすべての情報を、新しい個別のドキュメント 🕮『ConfigurationDesk Custom I/O Function Implementation Guide』で参照することができます。

このドキュメントには、以下の内容が含まれます。

- カスタムファンクションブロックの概要およびカスタムファンクションブロックとそのエレメントの定義の基礎
- シリアル通信および Ethernet 通信でのカスタムファンクションブロックの実装例
- 現時点でのカスタムファンクションブロックの実装に関する制限事項
- カスタムファンクションの XML ファイルで使用できるすべてのエレメントの構造と目的を記述した XML エレメントのリファレンス

ConfigurationDesk 5.4 への移行

TRC ファイル生成の変更

ConfigurationDesk での TRC ファイルの生成に関するいくつかの変更に注意する必要があります。「TRC ファイル生成の変更」(35 ページ)を参照してください。

コンテナ管理

コンテナ管理の新機能

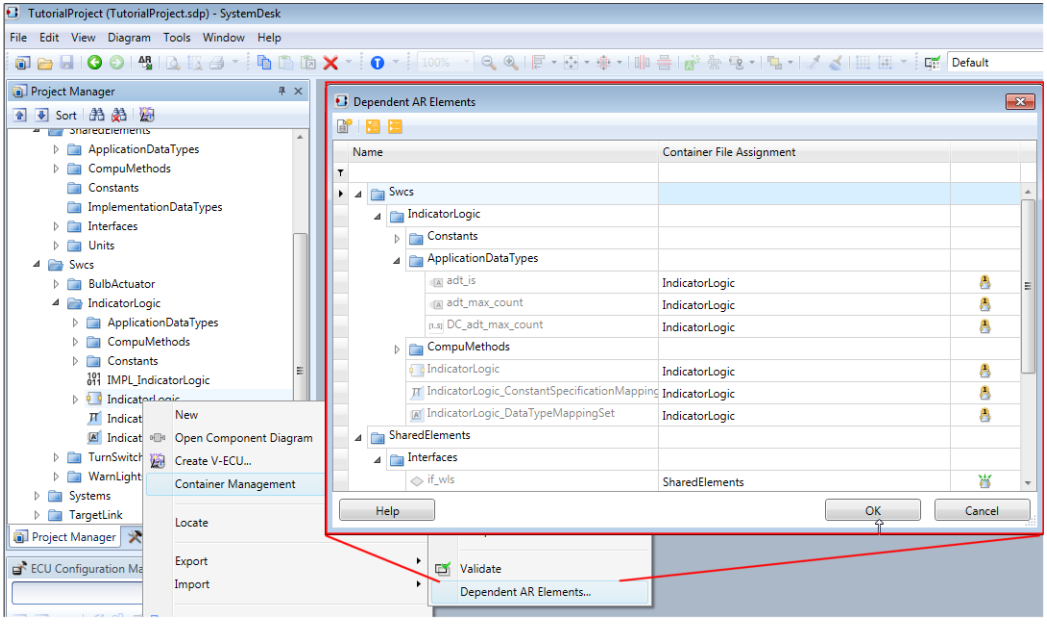
SystemDesk 4.5 でのコンテナファイルへのエレメントの割り当ての改善

このバージョンの Container Manager では、SystemDesk 4.5 での AUTOSAR エレメントのコンテナファイルへの割り当てが改善されました。ファイル割り当ては、SystemDesk と TargetLink の間でソフトウエアコンポーネントをやり取りする際に必要になります。

SystemDesk でアトミックソフトウエアコンポーネントタイプの[Container Management]コンテキストメニューから[Dependent AR elements]コマンドを選択して、エレメントを簡単に割り当てることができます。

このコマンドを選択すると[Dependent AR elements]ダイアログが開き、ここで、選択したソフトウエアコンポーネントタイプによって参照されるすべての AUTOSAR エレメントについて、ファイル割り当てを管理することができます。このコマンドにより、まだ割り当てられていないすべてのエレメントについて、割り当てが提示されます。新しいファイル割り当ての作成や、各エレメントまたはパッケージ全体に対するファイル割り当ての変更を行うことができます。ダイアログには、エレメントが選択したソフトウエアコンポーネントタイプのみで使用されているか( )、または複数のソフトウエアコンポーネントタイプで使用されているか( )が表示されます。

次の図は、SystemDesk 4.5 での例を示しています。



補足

SystemDesk 4.5 でのエレメントのコンテナファイルへの割り当ての詳細については、「How to Assign AR Elements to Container Files」(🕮『Container Management Document』)を参照してください。

ControlDesk Next Generation

項目の一覧

本章の内容

他章の参照情報

🕮 ControlDesk Next Generation 移行ガイド ControlDesk 3.x、CalDesk、および ControlDesk Next Generation の以前のバージョンから ControlDesk 5.5 への移行について説明します。
🕮 ControlDesk Next Generation Migration of ControlDesk 3.x Automation Explains migration from ControlDesk 3.x automation to ControlDesk Next Generation automation.

ControlDesk Next Generation の新機能（ControlDesk 5.5）

項目の一覧

本章の内容

プラットフォーム管理およびプラットフォーム／デバイスの新機能（ControlDesk 5.5）

本章の内容

新しい DCI-CAN2 のサポート

ControlDesk は新しい DCI-CAN2 をサポートしています。DCI-CAN2 により、ホスト PC を CAN FD または CAN ネットワークに接続することができます。

新しい DCI-CAN2 は、次の ControlDesk デバイスで使用することができます。

- CAN バスモニタリング
- CCP
- XCP on CAN

新しい DCI-CAN2 の詳細については、🕮『DCI-CAN2 Feature Reference』を参照してください。

CAN FD のサポート

次のデバイスは、新しい DCI-CAN2 に関連して CAN FD をサポートするようになりました。

- CAN バスモニタリング
- CCP
- XCP on CAN

現在のバージョンの dSPACE CAN API は、CAN FD をサポートしていません。

CAN FD 固有のデバイス設定の詳細については、「CAN Settings Properties」(🕮『ControlDesk Next Generation Reference』)を参照してください。

CAN バスモニタリングデバイス：FIBEX および AUTOSAR システムデスクリプションファイルのサポート

CAN バスモニタリングデバイスで、DBC に加えて、以下の変数記述ファイルフォーマットもサポートされます。

- FIBEX：
  - バージョン 4.1.0、4.1.1
  - バージョン 3.1.0、3.1.1
  - バージョン 3.0.0
- AUTOSAR システムテンプレートに準拠した AUTOSAR システムデスクリプションファイル
  - バージョン 4.2.1
  - バージョン 4.1.1 ...4.1.3
  - バージョン 4.0.3
  - バージョン 3.2.1 ...3.2.3
  - バージョン 3.1.4

ControlDesk 5.4 以前のバージョンで作成された、CAN バスモニタリングデバイスを含むエクスペリメントを再利用する場合は、移行上の注意点に注意してください。「ControlDesk Next Generation への移行（ControlDesk 5.5）」(116 ページ)を参照してください。

FlexRay バスモニタリングデバイス：AUTOSAR システムデスクリプションファイルのサポート

FlexRay バスモニタリングデバイスは、AUTOSAR システムテンプレートに準拠した次の AUTOSAR システムデスクリプションファイルもサポートするようになりました。

- バージョン 4.2.1
- バージョン 4.1.1 ...4.1.3
- バージョン 4.0.3
- バージョン 3.2.1 ...3.2.3
- バージョン 3.1.4

自動再接続をサポートするプラットフォームの追加

ControlDesk は、次のプラットフォームでも自動再接続機能 (→「自動再接続」(🕮『ControlDesk Next Generation ベーシックガイド』)) をサポートするようになりました。

- DS1007 PPC Processor Board
- DS1202 MicroLabBox
- SCALEXIO
- XIL API MAPort

*自動再接続*機能の詳細については、「プラットフォーム／デバイスのハードウエアへの自動再接続」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

シミュレーション時間グループのマスターの指定

シミュレーション時間グループのマスターを指定できるようになりました (→「シミュレーション時間グループ」(🕮『ControlDesk Next Generation ベーシックガイド』))。これは、シミュレーション時間グループのメンバーのいずれかが低いレイテンシと高い同期精度を持つ場合に役立ちます。

「Configure Simulation Time Group」(🕮『ControlDesk Next Generation Reference』)を参照してください。

アプリケーションプロセス情報の表示

マルチプロセッサ／マルチコアシステムのメンバーに割り当てられたハードウエアに関する情報、およびハードウエアに現在ロードされているアプリケーションプロセスに関する状態情報を表示できるようになりました。

「Online Details Properties」(🕮『ControlDesk Next Generation Reference』)を参照してください。

仮想検証シナリオの改善

複数のプラットフォーム／デバイスに対する変数記述ファイルのリロード 仮想検証シナリオでは、アクティブなアプリケーションや、VEOS や SCALEXIO 上で実行される仮想検証シナリオを参照するすべてのプラットフォーム／デバイスの変数記述ファイルを ControlDesk により簡単にリロードすることができます。

「Reload System」(🕮『ControlDesk Next Generation Reference』)を参照してください。

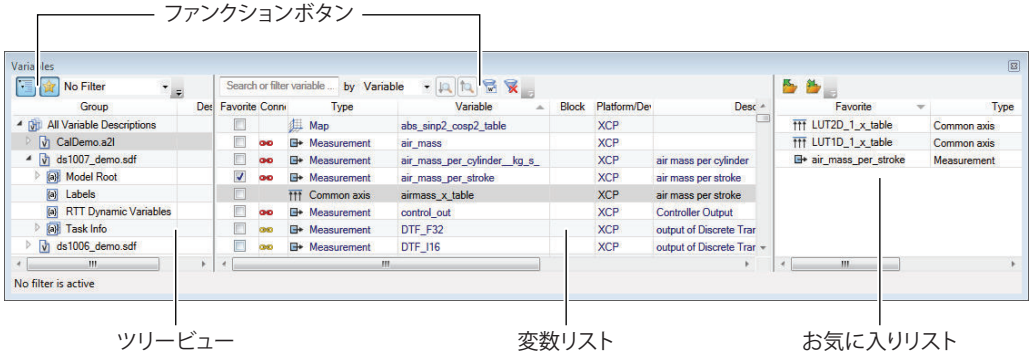
自動化：変数記述ファイルではなくアプリケーションを追加
ControlDesk の自動化インターフェースにより、エクスペリメントの VEOS または SCALEXIO プラットフォームに変数記述ファイルではなくアプリケーションファイルを追加できるようになりました。これは、*環境モデルなしでアプリケーションを使用する場合に便利です。*

「変数記述ファイルではなくアプリケーションを追加」(111 ページ)を参照してください。

変数管理の新機能（ControlDesk 5.5）

新しい Variable Browser

Variable Browser が一新されました。



詳細については、「Variable Browser の基礎」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

構造体と構造体配列のサポート

ControlDesk の Variable Browser は、構造体と構造体配列の表示をサポートするようになりました。

	アイコン	説明
構造体		さまざまなデータ型を持つことができる変数の構造化されたリスト
構造体配列		均一構造体の配列

構造体をバリアブルアレイにドラッグすると、含まれる配列と構造体配列を除き、それに含まれるすべての変数が接続されます。含まれる変数を別の計器に接続する場合は、接続割り当てをカスタマイズします。「計器への変数の接続割り当てのカスタマイズ」(105 ページ)を参照してください。

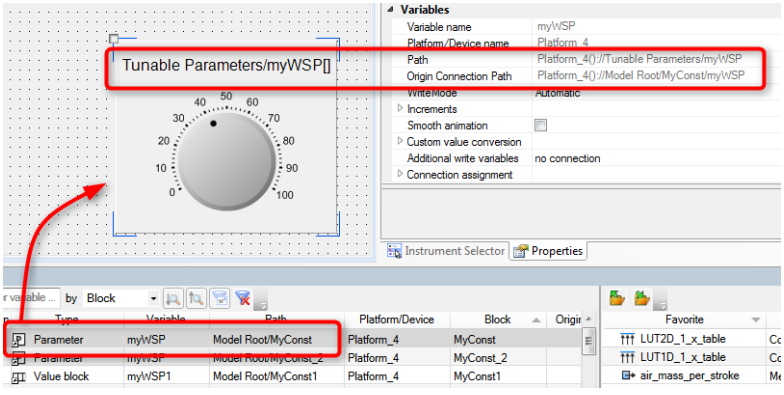
元の接続パスの表示

計器に変数を配置するときに、ControlDesk では、その計器に接続されていた元の変数のパスも表示するようになりました。これにより、接続されていた元の変数が、それが参照する変数に置き換えられた場合に、元の変数を識別することができます。

参照される変数　変数リスト内のサブノードの下に表示される変数のアイコンに、シンボルを追加することができます。矢印は、この変数が、変数記述ファイルの別のノードに存在する変数へのリファレンスであることを示します。たとえば、A2L ファイル内の変数は常にルートノードの下に存在します。

参照を計器にドラッグする場合、リファレンス自体ではなく、参照される変数が接続されます。

パスと元の接続パス　接続される変数のパスとドラッグした変数の*元の接続パス*が、計器プロパティに表示されます。



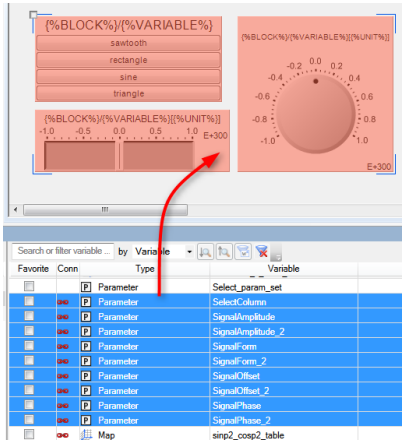
詳細については、「レイアウトでの変数の配置の基礎」(📖『ControlDesk Next Generation ベーシックガイド』)を参照してください。

新しいビジュアル表示および計器機能（ControlDesk 5.5）

計器への変数の接続割り当てのカスタマイズ

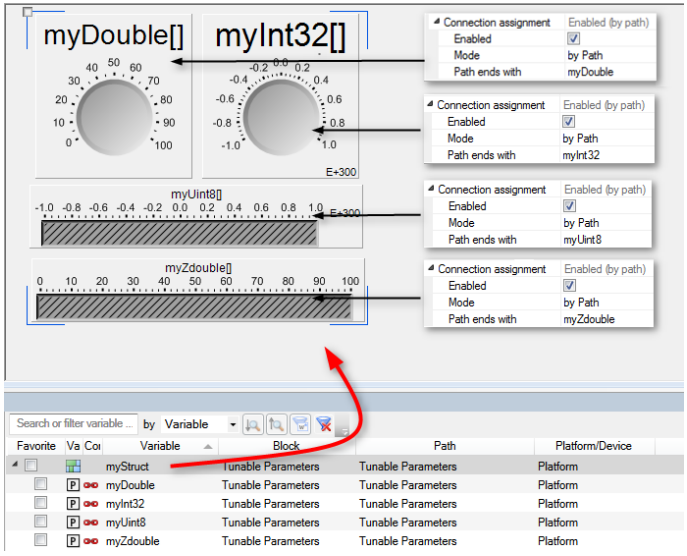
ControlDesk で、接続割り当てをカスタマイズできるようになりました。これにより、たとえば、変数のグループを複数の計器にワンステップで接続するなど、変数と計器の特別な接続を作成することができます。

下の図は、複数の変数をさまざまな計器に一度で接続している状態を示しています。



接続割り当ては、構造体変数の個々のメンバーを接続するのに特に役立ちます。ControlDesk で接続割り当てを実行するには、テキスト文字列を指定し、ControlDesk が構造体の中からこれに一致する変数を検索します。

次の図は、構造体のメンバーをさまざまな計器に一度で接続している状態を示しています。



詳細については、「複数の計器への変数の接続割り当てをカスタマイズする方法」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

Time Plotter と Index Plotter の機能拡張

Time Plotter データを新しい計測として保存(Time Plotter のみ)
Time Plotter のデータを新しい計測データファイルに保存できるようになりました。

詳細については、「プロッタまたは時間プロッタのデータを新しい記録として保存する方法」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

凡例の拡張 Time Plotter の凡例で、次のようないくつかの機能拡張が行われています。

- x Delta などの新しい凡例列
- 複数選択のサポート
- 凡例内の検索とフィルタリング

x 軸の同期 Time Plotter と Index Plotter の x 軸を連続ビジュアル表示モードで同期できるようになりました。この場合、同期化されたすべての Time Plotter は同じ時間範囲を共有し、すべての Index Plotter は同じインデックス範囲を共有します。そのため、ズームと移動動作は、同期化されたすべてのプロッタに同じように影響を与えます。

詳細については、「チャート（時間プロッタ）のズームおよび移動」(🕮『ControlDesk Next Generation ベーシックガイド』)および「チャート（インデックスプロッタ）のズームおよび移動」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

新しい線スタイル　Index Plotter と Time Plotter は、次の線スタイルもサポートするようになりました。

線スタイル	説明
Area	データポイントは、ダイレクトエリアラインによって接続されます。ベースラインとエリアラインの間のエリアは、信号の色で塗りつぶされます。
Area Staircase	データポイントは、階段エリアラインによって接続されます。ベースラインとエリアラインの間のエリアは、信号の色で塗りつぶされます。
Staircase	データポイントは階段ラインによって接続されます。

詳細については、「Axes and Signal Properties (Time Plotter/Index Plotter)」(🕮『ControlDesk Next Generation Reference』)を参照してください。

切り替え可能なラベル方向　y 軸ラベルを水平または垂直に表示できるようになりました。

詳細については、「View Properties (Time Plotter/Index Plotter)」(🕮『ControlDesk Next Generation Reference』)を参照してください。

マルチスイッチの拡張

現在選択しているエレメントを点線フレームで表示するかどうかを指定できるようになりました。

詳細については、「Multiswitch Properties」(🕮『ControlDesk Next Generation Reference』)を参照してください。

新しい計測機能および記録機能（ControlDesk 5.5）

新しいデフォルト交換フォーマット ASAM MDF 4.x

ASAM MDF 4.x ファイルフォーマットが、ControlDesk の計測データファイルのデフォルトフォーマットになりました。

DSSIGCONV ツール：ASAM MDF 4.x ファイルフォーマットのサポート
計測データファイルから必要な部分または信号を抽出できる DSSIGCONV ツールは、ASAM MDF 4.x ファイルフォーマットもサポートするようになりました。

計測データファイルから必要な部分または信号を抽出するか、計測データファイルを分割して処理するデータの量を削減することができます。

詳細については、「計測データファイルからデータを抽出する方法」(📖『ControlDesk Next Generation ベーシックガイド』)を参照してください。

多数の計測ラスタの処理

XIL API を介したバイパス処理やプラットフォームアクセスなど、特定のシナリオでは多数の計測ラスタを処理する必要があります。これにより、システムパフォーマンスが低下する場合があります。デフォルトでは、プラットフォーム／デバイスに対して Measurement Configuration で表示される計測ラスタの数は、約 20 ラスタに制限されます。

表示されるラスタの数を必要に応じて変更でき、これによりシステムパフォーマンスを向上させることができます。

詳細については、「多数の計測ラスタの処理」(📖『ControlDesk Next Generation ベーシックガイド』)を参照してください。

Measurement Data API の拡張

ルックアップテーブル（マップと曲線）と共通軸のサポート
Measurement Data API は、ルックアップテーブル（マップと曲線）と共通軸をサポートするようになりました。

詳細については、📖『ControlDesk Next Generation Measurement Data API Reference』を参照してください。

新しい信号を作成する際の変数タイプの指定 Measurement Data API を使用して新しい信号を作成する場合に、信号の変数タイプを指定できるようになりました。

「VariableTypeConstants」(📖『ControlDesk Next Generation Measurement Data API Reference』)を参照してください。

Bus Navigator の新機能（ControlDesk 5.5）

Bus Manager 設定用のバス計器の生成

ControlDesk の Bus Navigator では、ConfigurationDesk の Bus Manager によって作成された EXPSWCFG 設定データファイルに基づいて、SCALEXIO で使用するバス計器を生成できるようになりました。

計器の生成は、次の通信プロトコルでサポートされます。

- CAN
- CAN FD
- LIN

MicroLabBox での CAN バス通信の再生

MicroLabBox と接続して CAN バス通信を再生できるようになりました。

SCALEXIO と DCI-CAN2 での CAN FD のサポート

Bus Navigator は、次の場合に CAN FD（CAN with Flexible Data Rate）メッセージをサポートします。

- SCALEXIO プラットフォームで、CAN バス通信がビヘイビアモデルの RTI CAN MultiMessage Blockset によりモデリングされている場合
- DCI-CAN2 の場合

> SCALEXIO プラットフォームおよび DCI-CAN2 での CAN FD メッセージの再生は、まだサポートされていません。

「CAN に固有の Bus Navigator の機能」(📖『ControlDesk Next Generation アドバンストガイド』)を参照してください。

CAN バスモニタリングデバイス：FIBEX および AUTOSAR システムデスクリプションファイルのサポート

Bus Navigator は、CAN バスモニタリングデバイスに関連して、FIBEX および AUTOSAR システムデスクリプションファイルも変数記述ファイルフォーマットとしてサポートするようになりました。「CAN バスモニタリングデバイス：FIBEX および AUTOSAR システムデスクリプションファイルのサポート」(101 ページ)を参照してください。

FlexRay バスモニタリングデバイス：AUTOSAR システムデスクリプションファイルのサポート

Bus Navigator は、FlexRay バスモニタリングデバイスに関連して、AUTOSAR システムデスクリプションファイルも変数記述ファイルファイルフォーマットとしてサポートするようになりました。「FlexRay バスモニタリングデバイス：AUTOSAR システムデスクリプションファイルのサポート」(102 ページ)を参照してください。

新しいデータセット管理機能（ControlDesk 5.5）

CDFX の新しいデフォルト交換フォーマット

Calibration Data File（CDFX）の ASAM ファイルフォーマット（ASAM Calibration Data Format V2.0）は、ControlDesk のデータセット用デフォルトフォーマットになりました。

Signal Editor の新機能（ControlDesk 5.5）

DS1007 の完全なサポート

ControlDesk の Signal Editor は、DS1007 PPC Processor Board プラットフォーム上でのスティミュラス生成を完全にサポートするようになりました。つまり、DS1007 ベースの MP/MC アプリケーションに対する 1 つのシグナルジェネレータが、他のアプリケーションプロセスの変数をスティミュレートすることができます。

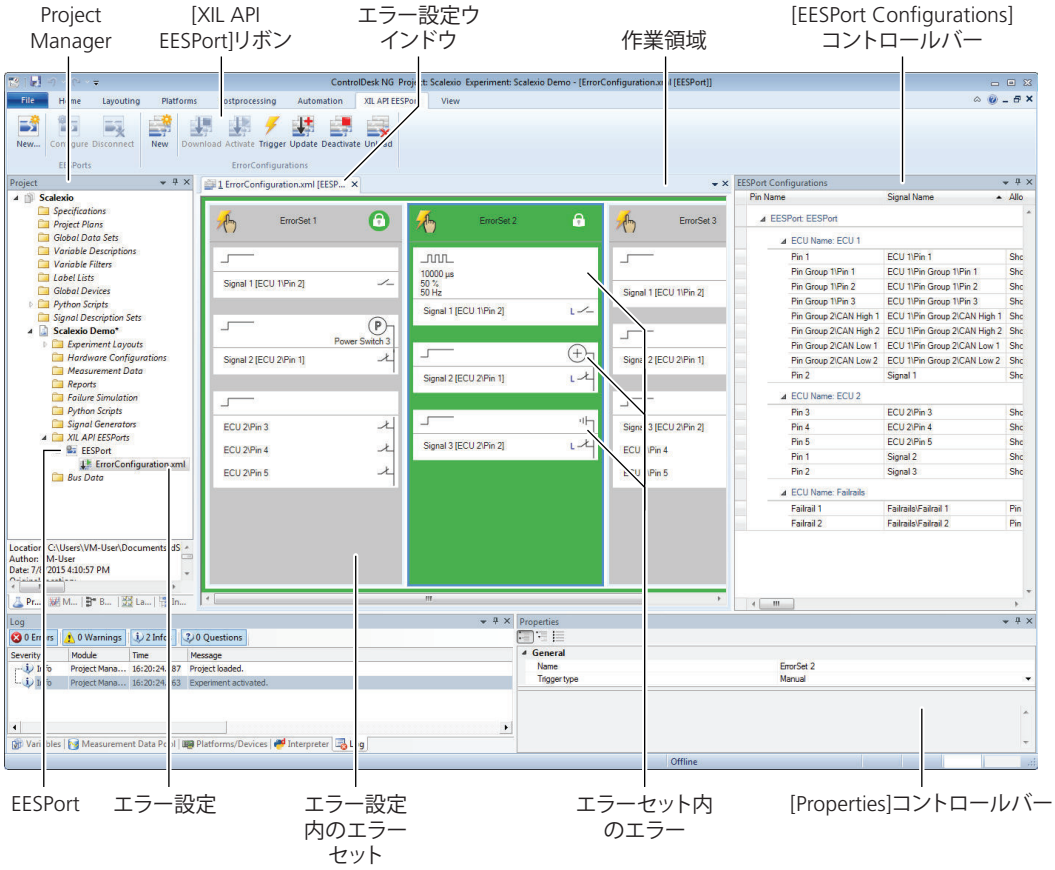
DS1202 MicroLabBox のサポート

ControlDesk の Signal Editor は、DS1202 MicroLabBox プラットフォーム上でのスティミュラス生成もサポートするようになりました。

新しい電気的欠陥シミュレーション機能（ControlDesk 5.5）

新しい XIL API EESPort グラフィカルユーザインターフェース

電気的欠陥シミュレーションを準備するために、ControlDesk で XIL API EESPort グラフィカルユーザインターフェースを利用できるようになりました。



詳細および手順については、「XIL API EESPort による電気的欠陥のシミュレーション」(🕮『ControlDesk Next Generation アドバンストガイド』)を参照してください。

ControlDesk の XIL API EESPort グラフィカルユーザインターフェースは、ControlDesk の Failure Simulation Module の後継です。Failure Simulation Module は、dSPACE Release 2016-A に付属するものが最後となります。

ControlDesk の Failure Simulation Module の電気的なエラー設定は、XIL API EESPort 設定と互換性がないことに注意してください。

移行については、「ControlDesk Next Generation への移行（ControlDesk 5.5）」(116 ページ)を参照してください。

新しい自動化機能（ControlDesk 5.5）

ルックアップテーブルの計測と記録（マップと曲線）

ControlDesk の自動化インターフェースにより、ルックアップテーブルを計測および記録できるようになりました（マップと曲線）。

計測信号リストに信号を追加または信号を削除する際のイベント

ControlDesk の自動化インターフェースは、計測信号リストに信号を追加または信号を削除する際の次のイベントをサポートするようになりました。

- `MeasurementSignalAdded`
- `MeasurementSignalRemoving`

「MeasurementDataManagementEvents / IXaMeasurementDataManagementEvents <<Events>>」(🕮『ControlDesk Next Generation API Reference』)を参照してください。

変数記述ファイルではなくアプリケーションを追加

ControlDesk の自動化インターフェースにより、エクスペリメントの VEOS または SCALEXIO プラットフォームに変数記述ファイルではなくアプリケーションファイルを追加できるようになりました。これは、*環境モデルなしでアプリケーション*を使用する場合に便利です。

「VEOSPlatform / IPmVEOSPlatform <<Interface>>」(🕮『ControlDesk Next Generation API Reference』)および「SCALEXIOPlatform / IPmSCALEXIOPlatform <<Interface>>」(🕮『ControlDesk Next Generation API Reference』)を参照してください。

自動化によるフォルダ場所の取得

Python *拡張スクリプト*を使用して、ControlDesk の機能を拡張し、ControlDesk のリボンをカスタマイズすることができます。拡張スクリプトは、場所に応じて、特定のユーザまたはすべてのユーザに対して実行されます。

たとえば、ControlDesk 5.5 以降、`ApplicationEnvironment / IAeApplicationEnvironment <<Interface>>`インターフェースのプロパティを使用して、`<DocumentsFolder>`の場所を取得することができます。「ApplicationEnvironment / IAeApplicationEnvironment <<Interface>>」(📖『ControlDesk Next Generation API Reference』)を参照してください。

ControlDesk Next Generation への移行（ControlDesk 5.5）

項目の一覧

本章の内容

他章の参照情報

🕮 ControlDesk Next Generation 移行ガイド ControlDesk 3.x、CalDesk、および ControlDesk Next Generation の以前のバージョンから ControlDesk 5.5 への移行について説明します。
🕮 ControlDesk Next Generation Migration of ControlDesk 3.x Automation Explains migration from ControlDesk 3.x automation to ControlDesk Next Generation automation.

ControlDesk での廃止

本章の内容

dSPACE Release 2016-A 以降の ControlDesk での廃止

ControlDesk の ASAP3 インターフェース　ControlDesk の ASAM ASAP3 互換インターフェースは、ControlDesk 5.5（dSPACE Release 2015-B）に付属するものが最後となります。

適合および計測タスクを自動化するには、代わりに以下を使用することができます。

- ControlDesk の自動化インターフェース。「ControlDesk の自動化」(🕮『ControlDesk Next Generation アドバンストガイド』)を参照してください。
- ControlDesk の ASAM MCD-3 互換インターフェース。
  🕮『ControlDesk Next Generation MCD-3 Automation Guide』を参照してください。

CDF のインポート／エクスポート　データセットをインポート／エクスポートするために使用する Calibration Data File（CDF）フォーマットのサポートは、ControlDesk 5.5（dSPACE Release 2015-B）で最後となります。

適合ツール用データを交換するには、CDFX（ASAM Calibration Data File 2.0）、DCM、DSV など、ControlDesk でサポートされる他のファイルフォーマットを使用してください。CDFX フォーマットは、ControlDesk のデータセット用デフォルト交換フォーマットです。

「データセットのエクスポートおよび変換」(📖『ControlDesk Next Generation ベーシックガイド』)を参照してください。

ユーザ定義のデータベース（UDDB）　ユーザ定義のデータベース（UDDB）のサポートは、ControlDesk 5.5（dSPACE Release 2015-B）で最後となります。

そのため、dSPACE リアルタイムハードウエアで CAN コントローラの通信を操作するには、リアルタイムモデルを変更する必要があります。

LDF（フォーマットバージョン 1.2 以前）　LIN Bus Monitoring デバイスでの LDF ファイル（フォーマットバージョン 1.2 およびそれ以前）のサポートは、ControlDesk 5.5（dSPACE Release 2015-B）で最後となります。

MAT ファイル（バージョン 6）のエクスポート　ControlDesk 5.5 以前のバージョンでは、MATLAB バージョン 5（R8）以降にロードできるバージョン 6 の MAT ファイルが作成されます。

dSPACE Release 2016-A 以降、ControlDesk では、MATLAB バージョン 7.3（R2006b）以降にロードできるバージョン 7.3 の MAT ファイルが作成されます。

dSPACE Release 2016-B 以降の ControlDesk での廃止

ControlDesk Failure Simulation Module　ControlDesk の Failure Simulation Module は、dSPACE Release 2016-A に付属するものが最後となります。

- グラフィカルユーザインターフェース（GUI）により電気的欠陥シミュレーションを準備するには、ControlDesk 5.5（dSPACE Release 2015-B）で導入された ControlDesk XIL API EESPort GUI を使用します。
- 自動化により電気的欠陥シミュレーションを準備するには、電気的欠陥シミュレーションポート（EESPort）をサポートする dSPACE XIL API .NET インプリメンテーションを使用します。

移行上の注意点については、「ControlDesk Next Generation への移行（ControlDesk 5.5）」(116 ページ)を参照してください。

プロッタ　Plotter は dSPACE Release 2016-A に付属するものが最後となります。

代わりに次のいずれかの計器を使用してください。

- Index Plotter
- Time Plotter
- XY Plotter

さまざまなプロッタタイプの相違点については、「プロッタ、時間プロッタ、インデックスプロッタ、XY プロッタの違い」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

Table Editor　Table Editor は dSPACE Release 2016-A に付属するものが最後となります。

これは拡張された Table Editor に置き換えられる予定です。

MDF（フォーマットバージョン 2.0 および 3.0）のエクスポート　MDF 計測データファイル（MDF ファイルフォーマットバージョン 2.0 および 3.0）のエクスポートのサポートは、dSPACE Release 2016-A で最後となります。

MDF ファイル（フォーマットバージョン 2.0 および 3.0）の *インポート*のサポートは継続されます。

計測データをエクスポートするには、ControlDesk でサポートされる他のファイルフォーマットのいずれかを使用してください。「記録用のストレージを設定する方法」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。

ControlDesk 3.x エクスペリメントの移行　ControlDesk Next Generation で再利用するための ControlDesk 3.x エクスペリメントの移行のサポートは、dSPACE Release 2016-A で最後となります。

dSPACE Release 2016-B 以降の ControlDesk で ControlDesk 3.x エクスペリメントを再利用するには、以下の手順を実行します。

1. dSPACE Release 2016-A 以前の ControlDesk を使用して、ControlDesk 3.x エクスペリメントを移行します。「Migrating from ControlDesk 3.x to ControlDesk Next Generation」(🕮『ControlDesk Next Generation 移行ガイド』)を参照してください。
2. dSPACE Release 2016-A 以前の ControlDesk から、dSPACE Release 2016-B 以降の ControlDesk にプロジェクトを移行します。「Migrating from Prior Versions of ControlDesk Next Generation」(🕮『ControlDesk Next Generation 移行ガイド』)を参照してください。

CalDesk プロジェクトの移行　ControlDesk Next Generation で再利用するための CalDesk プロジェクトの移行のサポートは、dSPACE Release 2016-A で最後となります。

dSPACE Release 2016-B 以降の ControlDesk で CalDesk プロジェクトを再利用するには、以下の手順を実行します。

1. dSPACE Release 2016-A 以前の ControlDesk を使用して、CalDesk プロジェクトを移行します。「Migrating from CalDesk to ControlDesk Next Generation」(🕮『ControlDesk Next Generation 移行ガイド』)を参照してください。
2. dSPACE Release 2016-A 以前の ControlDesk から、dSPACE Release 2016-B 以降の ControlDesk にプロジェクトを移行します。「Migrating from Prior Versions of ControlDesk Next Generation」(🕮『ControlDesk Next Generation 移行ガイド』)を参照してください。

ControlDesk Next Generation への移行（ControlDesk 5.5）

ControlDesk 5.4 から ControlDesk ControlDesk 5.5 に移行して既存のエクスペリメントを再利用するには、次の移行手順が必要になる場合があります。

5.4 より前のバージョンから ControlDesk 5.5 に移行するには、その間の ControlDesk バージョンの移行手順の実行も必要になる場合があります。

本章の内容

TRC ファイル生成の変更

MATLAB R2015b に関連した、TRC ファイルの生成に関する修正に注意する必要があります。

「TRC ファイルを使用するソフトウエアでの変更の移行」(42 ページ)を参照してください。

Failure Simulation Module：廃止と移行

ControlDesk の Failure Simulation Module は、dSPACE Release 2016-A に付属するものが最後となります。

- グラフィカルユーザインターフェース（GUI）により電気的欠陥シミュレーションを準備するには、ControlDesk 5.5（dSPACE Release 2015-B）で導入された ControlDesk XIL API EESPort GUI を使用します。

  ControlDesk XIL API EESPort GUI を使用するには、XIL API の EESPort をベースとした Failure Simulation Package が必要です。インプリメンテーションのベースは dSPACE XIL API .NET です。

  ControlDesk の Failure Simulation Module の電気的なエラー設定は、XIL API EESPort 設定と互換性がないことに注意してください。

  移行では、`FailureSimulationExportTool` を使用して、既存の ControlDesk 欠陥シミュレーションシステム（FSN）ファイルからの情報を以下のファイルにエクスポートすることができます。

  - ハードウエア依存のポート設定（PORTCONFIG）ファイル

    このファイルを使用して、新しい EESPort を作成することができます。詳細については、「新しい EESPort を作成する方法」(🕮『ControlDesk Next Generation アドバンストガイド』)を参照してください。

- 各欠陥パターンに 1 つのエラー設定 XML ファイル

  このファイルを使用して、電気的エラーを作成および設定することができます。「電気的欠陥を作成および設定する方法」(🕮『ControlDesk Next Generation アドバンストガイド』)を参照してください。

次の表に示すように、使用する `FailureSimulationExportTool` のバージョンは、インストールされている ControlDesk と dSPACE XIL API .NET のバージョンによって異なります。

インストールされている ControlDesk バージョン	インストールされている dSPACE XIL API .NET バージョン	必要な FailureSimulationExportTool バージョン
5.3	2.0	2014-B
5.4	2015-A	2015-A
5.5	2015-B	2015-B

`FailureSimulationExportTool` とユーザドキュメントを含む ReadMe ファイルは、*ControlDesk Next Generation 製品サポートセンター* (http://www.dspace.jp/goto.cfm/cdng_psc) からダウンロードすることができます。

- 自動化により電気的欠陥シミュレーションを準備するには、電気的欠陥シミュレーションポート(EESPort)をサポートする dSPACE XIL API .NET インプリメンテーションを使用します。

CAN Bus Monitoring デバイス:DBC インポートの変更

バージョン 5.5 以降、ControlDesk での DBC ファイルのインポートは、CAN バスモニタリングデバイスに関連して、FIBEX および AUTOSAR システムデスクリプションファイルをサポートするように変更されました。そのため、DBC ファイル内の変数へのパスは、DBC ファイルを ControlDesk 5.4 以前と ControlDesk 5.5 以降のどちらでインポートしたかによって異なります。

ControlDesk 5.4 以前のバージョンで作成されたエクスペリメントを再利用する場合は、インポート元の DBC ファイルに基づいて、デバイスやレイアウト／計器を引き続き通常どおり使用することができます。

これには、次の制限事項があります。

- インポート元の DBC ファイルの置換およびリロードは阻止されます。
- CAN バスモニタリングデバイスに新しいバージョンの DBC ファイルを追加したときに、ControlDesk はこの DBC 変数記述ファイルを有効化して、元の変数接続を復元しようとします。ところが、DBC ファイルのインポートが変更されたことで、新しく追加された DBC ファイル内の変数へのパスが異なるため、同じ DBC ファイルを追加した場合でも、ControlDesk は変数接続を復元することができません。

CAN バスモニタリングデバイスに新しいバージョンの DBC ファイルを追加後、新しく追加した DBC ファイルの変数パスに基づいて新しいレイアウトを生成してください。

互換性のない SCALEXIO アプリケーションのダウンロード時のデフォルト動作の変更

リアルタイムアプリケーションを SCALEXIO システムにダウンロードする場合、SCALEXIO システム I/O とリアルタイムアプリケーションに必要な I/O の相違など、ControlDesk で互換性がないことが検出される場合があります。

このような非互換性を検出した場合の ControlDesk のデフォルト動作は、次のように変更されました。

- ControlDesk 5.4 以前のバージョンでは、ControlDesk はデフォルトで、異なる I/O チャンネルへのアクセスをシミュレートしていました。
- ControlDesk 5.5 以降、ControlDesk はデフォルトで、異なる I/O チャンネルへのアクセスを有効化します。

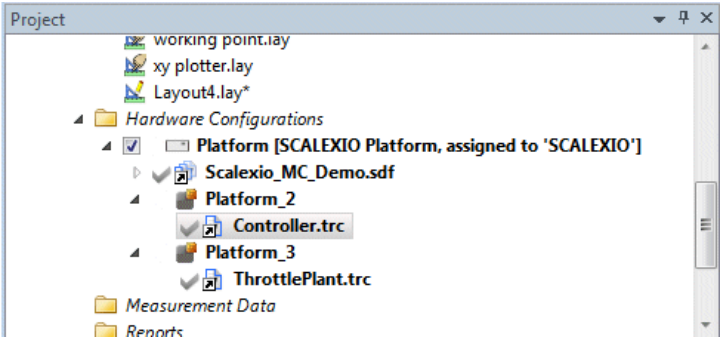
  ControlDesk で異なる I/O チャンネルへのアクセスをシミュレートできるようにするには、デフォルト動作を明示的に選択解除する必要があります。

デフォルト動作が変更されたのは、ControlDesk のグラフィカルユーザインターフェースからアプリケーションをダウンロードした場合です。ControlDesk の自動化インターフェースからアプリケーションをダウンロードした場合のデフォルト動作は、*変更されていません*。

Path プロパティの変更

ControlDesk 5.5 では、マルチプロセッサ／マルチコアリアルタイムアプリケーションのビジュアル表示される変数の[Path]プロパティが[Properties]コントロールバーに表示される方法が変更されています。

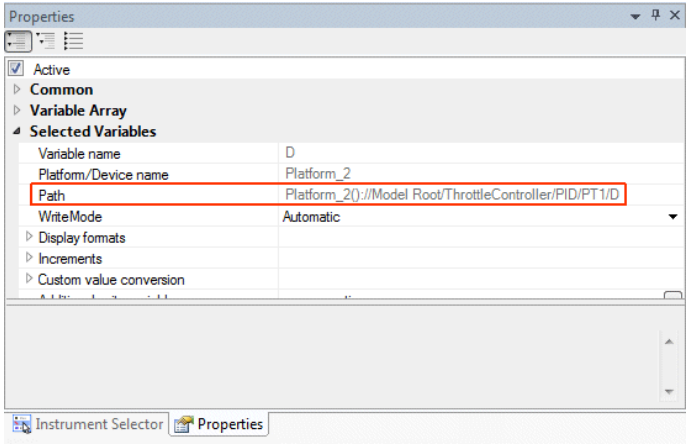
下の図は、マルチコアアプリケーションに関連する変数記述ファイルを含む SCALEXIO プラットフォームの例を示しています。



- ControlDesk 5.4 以前でのパス:

  `[PlatformName()://ModelRoot/...]`

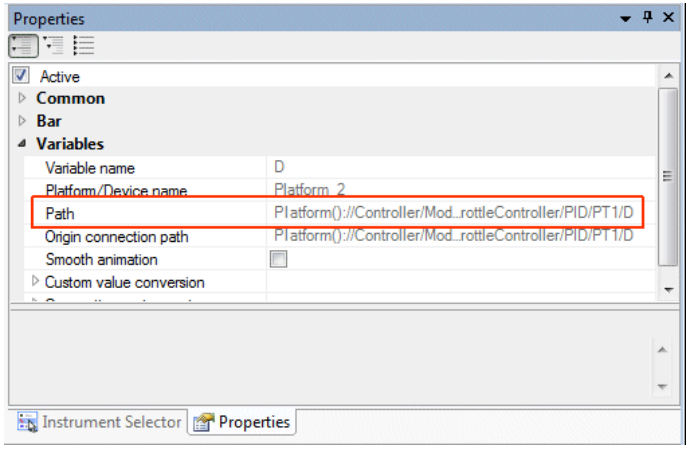
次の図に、マルチコアアプリケーションの変数へのパスの[Properties]コントロールバー（ControlDesk 5.4 以前）での表示例を示します。



■ ControlDesk 5.5 以降でのパス：

```
[MasterPlatform()://SubPlatform/ModelRoot/...]
```

下の図に、同じ変数へのパスの[Properties]コントロールバー（ControlDesk 5.5 以降）での表示例を示します。



この変更には手作業での移行手順は必要ありません。

参照情報については、「Variables Properties」(🕮『ControlDesk Next Generation Reference』)を参照してください。

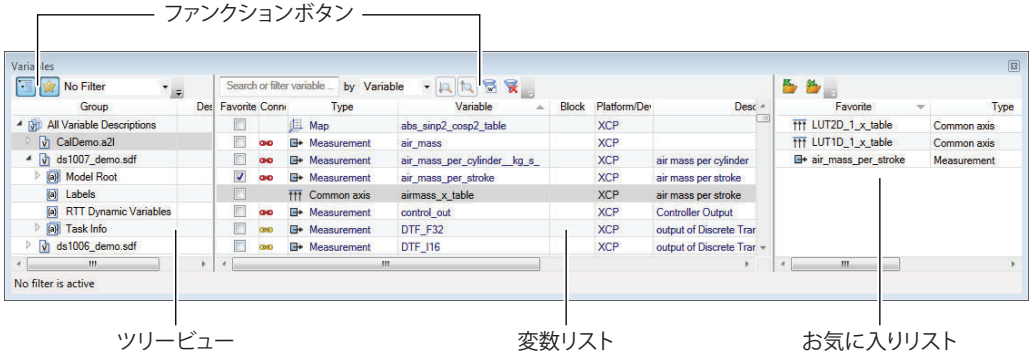
マルチスケーリング変数によるトリガ記録の評価の変更

ControlDesk 5.5 以降、マルチスケーリングテーブルを使用する変数をトリガ規則の記録の中で使用する場合、トリガ評価には*変数のソース値*が使用されます。

ControlDesk 5.4 では、*変数の変換値*がトリガ評価に使用されていました。

チェックされる変数とラベルリストの処理の変更

Variable Browser は、ControlDesk 5.5 で一新されました。Variable Browser の新しいお気に入りリストでは、チェック済み変数リストとラベルリストが統合されています。



Variable Browser の概要については、「Variable Browser の基礎」(🕮『ControlDesk Next Generation ベーシックガイド』)を参照してください。お気に入りリストの詳細については、「Favorites List」(🕮『ControlDesk Next Generation Reference』)を参照してください。

ツール自動化の変更

変数管理自動化インターフェースの変更　バージョン 5.5 以降、ControlDesk の Variable Browser の機能が強化されました。

このため、VariablesManagement / IXaVariablesManagement <<Interface>>の次のプロパティが不要になり、自動化インターフェースから削除されています。

- `Application.VariablesManagement.OperationButtonsVisible`
- `Application.VariablesManagement.SubsystemFirstEnabled`

VideoDisplayStyleConstants の列挙値の変更　ControlDesk 5.5 では、「VideoDisplayStyleConstants <<Enumeration>>」(🕮『ControlDesk Next Generation API Reference』)の列挙値の `Strech` 値が `Stretch` に修正されました。

列挙値 `Stretch` の数値は変更されていません。

記録した多次元配列の保存に関する変更　ControlDesk 5.4 以前のバージョンでは、記録した多次元配列は、配列としてではなくネストされたベクトルとして保存されていました。

ControlDesk 5.5 以降、記録した多次元配列は配列として保存されます。そのため、これに応じて自動化スクリプトの調整が必要になる場合があります。

Python を使用して ControlDesk を自動化している場合は、スクリプトの調整は不要です。

Measurement Data API の変更

- `FileDescription` インターフェースの `Size` 属性のデータ型が、`Int` から `Float` に変更されました。

  「Class Description (FileDescription)」(🕮『ControlDesk Next Generation Measurement Data API Reference』)を参照してください。

- `Signal` インターフェースの `BitOffset` および `NumberOfBits` 属性は*設定*できなくなりました。これらの属性値は*取得*のみ可能です。

  「Class Description (Signal)」(🕮『ControlDesk Next Generation Measurement Data API Reference』)を参照してください。

以前のバージョンの ControlDesk Next Generation からの移行

以前のバージョンの ControlDesk Next Generation から移行して既存のエクスペリメントを再利用するには、追加の移行手順が必要な場合があります。移行手順の詳細については、「Migrating to ControlDesk Next Generation」(🕮『ControlDesk Next Generation 移行ガイド』)を参照してください。

DCI Configuration Tool

DCI Configuration Tool 3.5 の新機能

A2L ファイル適合の改善

DCI Configuration Tool には、DCI-GSI2 での使用に対応した既存の A2L ファイルの調整に関する改善が加えられています。

「[A2L File]ページ」(🕮『DCI の設定』)を参照してください。

dSPACE FlexRay Configuration Package

dSPACE FlexRay Configuration Package 3.6 の新機能

FlexRay Configuration Tool

AUTOSAR System Template 4.2.1 のサポート FlexRay Configuration Tool では、FlexRay ネットワークの記述に、AUTOSAR Release 4.2.1 に準拠した AUTOSAR System Template がサポートされます。

不明確なバイトオーダー形式のサポート FlexRay Configuration Tool では、不明確なバイトオーダーを使用した信号がサポートされます。これは AUTOSAR システムデスクリプションファイルで定義することができます。不明確なバイトオーダー形式を使用した信号インスタンスのデータは、動的な uint8[n]配列として解釈されます。ここで、n は信号の長さに依存します。データはエンディアン変換せずに送信されます。開始ビットはバイトアラインする必要があります。

詳細については、「Handling Configuration Projects」(🕮『FlexRay Configuration Tool Guide』)を参照してください。

FlexRay Replay Script Generator の廃止

FlexRay Replay Script Generator は、dSPACE Release 2015-B に付属しているものが最後となります。

FlexRay Replay Script Generator は、ログ記録された FlexRay 信号（MAT ファイルに保存）を、再生モデルの TX 信号（TRC ファイルに含まれる）にマッピングするために使用されます。マッピングされた信号から Python スクリプトを生成することができます。その後、Python インタープリタを介して Real-Time Testing でスクリプトを再生することができます。

RLS2016-A 以降でも、引き続き Python インタープリタを FlexRay タイムテーブルタスクに統合することができます。これにより、以前と同様にユーザが作成した Python スクリプトを FlexRay バスと同期させて再生することができます。

dSPACE HIL API .NET

dSPACE HIL API .NET 2.0 の新機能

トレースファイル生成の機能強化のサポート

dSPACE HIL API .NET は、トレースファイル生成機能の強化に伴う新しい機能をサポートしています。「TRC ファイル生成の変更」(35 ページ)も参照してください。

テストスクリプト内の変数パスが無効になる場合があります。詳細については、「TRC ファイルを使用するソフトウエアでの変更の移行」(42 ページ)を参照してください。

スティミュラスのサポート

DS1007 での MAPort スティミュラスのサポートが強化されました。DS1007 をマルチコアまたはマルチプロセッサシステムとして使用する場合、シグナルジェネレータを実行している別のサブアプリケーションに含まれる変数もスティミュレートすることができます。

また、MAPort スティミュラスは MicroLabBox もサポートするようになりました。

新しいデフォルトの変数パスフォーマット

MAPort プロパティ `VariableNames` は、ControlDesk Next Generation フォーマットで変数パスを返すようになりました。ControlDesk 3.x の変数パスフォーマットは引き続きサポートされます。

次の表に、例をいくつか示します。

タイプ	ControlDesk 3.x の変数パスのフォーマット	ControlDesk Next Generation の変数パスのフォーマット
スカラー	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'[1)]
ベクトル	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[1,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[2]'[2)]

タイプ	ControlDesk 3.x の変数パスのフォーマット	ControlDesk Next Generation の変数パスのフォーマット
行列	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[2,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[1][2]'
構造体 [3]	–	'Platform()://[MP_SubApplicationName/]Model Root/Structures/Struct.SubStruct.DoubleArray[0]'

[1] 'Platform'は関連するプラットフォーム名（例：'ds1006'）に置換されます。プラットフォーム名はモデルパス自体には関係ありません。マルチプロセッサモデルを使用している場合、サブアプリケーションの名前をモデルパスで使用することができます。
[2] ControlDesk Next Generation フォーマットでは、配列インデックスは 0 から始まり、ControlDesk 3.x フォーマットでは 1 から始まります。
[3] dSPACE Release 2015-B で導入されました。

dSPACE Python Extensions

dSPACE Python Extensions 2.0 の新機能

トレースファイル生成の機能強化のサポート

dSPACE HIL API Python および rtplib2 は、トレースファイル生成機能の強化に伴う新しい機能をサポートしています。「TRC ファイル生成の変更」(35 ページ)も参照してください。

テストスクリプト内の変数パスが無効になる場合があります。詳細については、「TRC ファイルを使用するソフトウエアでの変更の移行」(42 ページ)を参照してください。

スティミュラスのサポート

DS1007 での MAPort スティミュラスのサポートが強化されました。DS1007 をマルチコアまたはマルチプロセッサシステムとして使用する場合、シグナルジェネレータを実行している別のサブアプリケーションに含まれる変数もスティミュレートすることができます。

また、MAPort スティミュラスは MicroLabBox もサポートするようになりました。

新しいデフォルトの変数パスフォーマット

MAPort プロパティ `VariableNames` は、ControlDesk Next Generation フォーマットで変数パスを返すようになりました。ControlDesk 3.x の変数パスフォーマットは引き続きサポートされます。

次の表に、例をいくつか示します。

タイプ	ControlDesk 3.x の変数パスのフォーマット	ControlDesk Next Generation の変数パスのフォーマット
スカラー	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'[1)]
ベクトル	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[1,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[2]'[2)]

タイプ	ControlDesk 3.x の変数パスのフォーマット	ControlDesk Next Generation の変数パスのフォーマット
行列	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[2,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[1][2]'
構造体 [3]	–	'Platform()://[MP_SubApplicationName/]Model Root/Structures/Struct.SubStruct.DoubleArray[0]'

[1] 'Platform'は関連するプラットフォーム名(例:'ds1006')に置換されます。プラットフォーム名はモデルパス自体には関係ありません。マルチプロセッサモデルを使用している場合、サブアプリケーションの名前をモデルパスで使用することができます。
[2] ControlDesk Next Generation フォーマットでは、配列インデックスは 0 から始まり、ControlDesk 3.x フォーマットでは 1 から始まります。
[3] dSPACE Release 2015-B で導入されました。

Test Automation Python モジュール

`rtplib2` Python モジュールでは次のようにメソッドが変更されています。

- `GetVarNames` メソッドは、変数パスを ControlDesk Next Generation フォーマットで返すようになりました。ControlDesk 3.x の変数パスフォーマットは引き続きサポートされます。例については、上記を参照してください。

`matlablib2` Python モジュールでは、新しいプロパティの追加およびメソッドの拡張が行われています。

新しいプロパティを以下に示します。

- `WatchdogMethod`

  MATLAB プロセスを確認するためのメソッドを取得または設定します。

- `Visible`

  MATLAB ユーザインターフェースを可視化または設定します。

- `ProcessArchitecture`

  接続された MATLAB インスタンスのプロセスアーキテクチャ(32 ビットまたは 64 ビット)を取得します。

- `ProcessID`

  接続された MATLAB インスタンスのプロセス ID を取得します。

- `ExecutablePath`

  接続された MATLAB インスタンスの実行ファイルへのパスを取得します。

- `Version`

  接続された MATLAB インスタンスのバージョンを取得します。

- `IsMUMatlabOpen`

  接続された MATLAB インスタンスを複数で使用するように開いているかどうかを示すフラグを取得します。

新しいメソッドを以下に示します。

- `MaximizeCommandWindow`

  MATLAB コマンドウインドウを最大化します。

拡張されたメソッドを以下に示します。

- MATLAB インスタンス:`Open`

  新しいオプションの `MLInstallDir` パラメータにより、指定したインストールフォルダから MATLAB インスタンスを起動するように設定することができます。

- MATLAB インスタンス:`Close`

  新しいオプションの `DisconnectOnly` パラメータにより、インスタンスからの自動アクセスを切断したときに MATLAB インスタンスを閉じるかどうかを設定することができます。

- MATFile インスタンス:`Open`

  `Mode` パラメータは `w7.3` オプションをサポートします。これにより、2 GB を超えるオブジェクトを格納可能な HDF5 ベースの形式で MAT ファイルを作成することができます。

dSPACE XIL API

dSPACE XIL API 2015-B の新機能

トレースファイル生成の機能強化のサポート

dSPACE XIL API は、トレースファイル生成の機能強化に伴う新しい機能をサポートしています。「TRC ファイル生成の変更」(35 ページ)も参照してください。

テストスクリプト内の変数パスが無効になる場合があります。詳細については、「TRC ファイルを使用するソフトウエアでの変更の移行」(42 ページ)を参照してください。

スティミュラスのサポート

DS1007 での MAPort スティミュラスのサポートが強化されました。DS1007 をマルチコアまたはマルチプロセッサシステムとして使用する場合、シグナルジェネレータを実行している別のサブアプリケーションに含まれる変数もスティミュレートすることができます。

また、MAPort スティミュラスは MicroLabBox もサポートするようになりました。

新しいデフォルトの変数パスフォーマット

MAPort プロパティ `VariableNames` は、ControlDesk Next Generation フォーマットで変数パスを返すようになりました。ControlDesk 3.x の変数パスフォーマットは引き続きサポートされます。

次の表に、例をいくつか示します。

タイプ	ControlDesk 3.x の変数パスのフォーマット	ControlDesk Next Generation の変数パスのフォーマット
スカラー	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'[1)]
ベクトル	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[1,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[2]'[2)]

タイプ	ControlDesk 3.x の変数パスのフォーマット	ControlDesk Next Generation の変数パスのフォーマット
行列	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[2,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[1][2]'
構造体 [3]	–	'Platform()://[MP_SubApplicationName/]Model Root/Structures/Struct.SubStruct.DoubleArray[0]'

[1] 'Platform'は関連するプラットフォーム名（例：'ds1006'）に置換されます。プラットフォーム名はモデルパス自体には関係ありません。マルチプロセッサモデルを使用している場合、サブアプリケーションの名前をモデルパスで使用することができます。
[2] ControlDesk Next Generation フォーマットでは、配列インデックスは 0 から始まり、ControlDesk 3.x フォーマットでは 1 から始まります。
[3] dSPACE Release 2015-B で導入されました。

ECU Interface Manager

項目の一覧

本章の内容

ECU Interface Manager 1.7 の新機能

DPMEM POD 用に設定された組込みの dSPACE Calibration and Bypassing Service のサポート

ECU Interface Manager では、dSPACE Calibration and Bypassing Service が ECU アプリケーションに組込まれて、dSPACE DPMEM プラグオンデバイス(POD)で使用するように設定されている場合、これもサポートするようになりました。ECU Interface Manager では、このような組込みの dSPACE Calibration and Bypassing Service に、他のサービス関数呼び出しを追加することができます。これにより、DPMEM POD を使用してサービスベースのバイパス処理を準備することができます。

「バイパス処理用のコードアイテムの準備」(🕮『ECU Interface Manager ガイド』)を参照してください。

実行制御の挿入が失敗した場合の動作の指定

実行制御を挿入するには、追加のメモリ空間が必要です。

*コードアイテムのすべてのインスタンスに*実行制御を挿入しようとしており、少なくとも 1 つの実行制御を挿入するためのメモリ空間が十分に空いていない場合は、ECU Interface Manager のダイアログが開き、次の操作のいずれかを選択することができます。

- 問題のあるインスタンスに対してのみ挿入をスキップ
- すべてのインスタンスの実行を*恒久的に*無効化
- すべてのインスタンスに対して挿入をスキップ

「コードアイテムの無効化の基礎」(🕮『ECU Interface Manager ガイド』)を参照してください。

ECU Interface Manager 1.7 への移行

前のバージョンの ECU Interface Manager で最後に保存したプロジェクトの移行

ECU Interface Manager 1.7 では、前のバージョンの ECU Interface Manager で最後に保存したプロジェクトを再利用することができます。

このようなプロジェクトを初めて開くと、プロジェクトをアップデートするかどうかを確認するメッセージが表示されます。

- アップデートを開始した場合、ECU Interface Manager 1.7 でプロジェクトを引き続き使用することができます。
- アップデートを見送った場合、アプリケーションのエクスポート以外のアクションはブロックされます。プロジェクトは後からアップデートすることができます。
- プロジェクトを保存する際には、以前のプロジェクトファイルを上書きするかどうかを確認するメッセージが表示されます。
  - 以前のプロジェクトを上書きした場合、前のバージョンの ECU Interface Manager でそのプロジェクトを使用できなくなります。
  - 以前のプロジェクトを上書きしない場合は、プロジェクトファイルの場所や名前を指定する必要があります。これにより、前のバージョンの ECU Interface Manager で使用できるプロジェクトのバージョンを維持することができます。

新しいソフトウエアモジュールデスクリプションファイルスキーマ

ECU Interface Manager 1.6 では、ECU サプライヤが汎用スキーマを使用してソフトウエアモジュールデスクリプションファイル(→「ソフトウエアモジュールデスクリプションファイル」(🕮『ECU Interface Manager ガイド』))を作成できるようになりました。

また、ECU Interface Manager 1.0 で最初に導入された、dSPACE 固有のスキーマに基づくソフトウエアモジュールデスクリプションファイルをインポートすることもできます。

- dSPACE 固有のスキーマは、下位互換性の理由でのみサポートされます。次回の dSPACE リリースで汎用スキーマに置き換えられる予定です。
- dSPACE 固有のスキーマでは、マルチコアのサポートやその他の拡張機能を利用することはできません。

  代わりに*汎用スキーマ*を使用してください。

汎用スキーマの詳細については、「Generic Schema of Software Module Description Files」(🕮『ECU Interface Manager Reference』)を参照してください。

Firmware Manager

Firmware Manager 2.0 の新機能

プラットフォームサポートの強化	Firmware Manager で、SCALEXIO システムがサポートされます。 このサポートは、dSPACE Release 2015-B でインストールされたファームウエアアーカイブに限定されます。SCALEXIO システム上のファームウエアバージョンを以前のバージョンに更新するには、以前の dSPACE リリースの ControlDesk Next Generation または ConfigurationDesk を使用する必要があります。
ユーザビリティの改善	[Firmware Management]ペインには、使用可能なファームウエアアーカイブバージョンと追加情報のリストを表示する補足ペインがあります。 詳細については、「Firmware Manager Reference」(🕮『Firmware Manager Document』)を参照してください。
アーカイブ形式の変更	DS1007 と MicroLabBox のアーカイブ形式は、dSPACE Release 2015-B に含まれる Firmware Archives 2.0 で変更されました。以前のバージョンのアーカイブは、Firmware Manager 2.0 ではロードすることができません。

Model Compare

この製品は米国での使用が禁止されています

米国では Model Compare を使用することはできません。この製品を米国内で使用することも第三者に使用させることも米国の法律に違反します。

項目の一覧

本章の内容

Model Compare 2.6 の新機能

フックによるダンプファイルのカスタマイズ

Model Compare では、ダンプファイルをカスタマイズできるフック機能を提供しています。フックは、どのブロックおよびどのブロック詳細を、どの形式でダンプするかを詳細に定義します。サブシステムに他のサブシステムが含まれる場合は、含まれるサブシステムを省略することができます。ユーザ独自のフックを書き込むことができます。デモフックを使用することができます。

関連ドキュメント

- 「XML ダンプファイルの基礎」(🕮『Model Compare ガイド』)
- 「フックを使用して XML ダンプをカスタマイズするには」(🕮『Model Compare ガイド』)

リーフモデルエレメントをすばやく強調表示

Model Navigator でモデルエレメントのコンテキストメニューから強調表示オプションを選択する代わりに、リーフモデルエレメントをダブルクリックするだけで、Simulink ですばやく強調表示できるようになりました。

関連ドキュメント

- 「MATLAB でモデルの相違点をビジュアル表示する方法」(📖『Model Compare ガイド』)

検索の強化

このバージョンの Model Compare では、プロパティ値の文字列を検索することができます。さらに、Model Compare では、たとえば正規表現などの高度な検索オプションも使用することができます。

関連ドキュメント

- 「General Page」(📖『Model Compare Reference』)
- 「Find Bar」(📖『Model Compare Reference』)

モデルの保存

モデルが保存されているときに作成されたすべてのバックアップファイルを保持するように設定することができます。さらに、モデルのコピーを、別の名前とパスを使用して保存できるようになりました。

関連ドキュメント

- 「Merge Settings Page」(📖『Model Compare Reference』)
- 「Save Model Copy」(📖『Model Compare Reference』)

プロパティの名前と値のコピー

[Property Inspector]でプロパティをダブルクリックすると個別のウィンドウが開き、そこでプロパティ名とプロパティ値をコピーすることができます。

関連ドキュメント

- 「Property Inspector」(📖『Model Compare Reference』)

新しいクイックガイドとデモモデル

Model Compare では、Model Compare の最も重要な機能をまとめた新しい Model Compare Quick Guide を利用できるようになりました。このドキュメントには、[Help]メニューからアクセスすることができます。さらに、Model Compare にはデモモデルが付属しており、現在の作業フォルダに展開することができます。

関連ドキュメント

- 「Quick Guide」(📖『Model Compare Reference』)
- 「Extract Demos」(📖『Model Compare Reference』)

Model Compare 2.6 への移行

調整不要	Model Compare 2.5 から Model Compare 2.6 へは、何も調整せずに移行することができます。

ModelDesk

ModelDesk 4.2 の新機能

新しくサポートされるプラットフォーム	ModelDesk では、以下のシミュレーションプラットフォームが新たにサポートされます。 ■ DS1007 PPC Processor Board ■ MicroLabBox
処理	**自動化** ツール自動化により、次のような重要な処理ステップを自動化することができます。 ■ 計測タイプの作成と編集 ■ 計測データの作成と編集 ■ パラメータの処理プロパティへのアクセス **自動化されたマッピング** 生データの変数と計測タイプの変数とのマッピングが改善されました。自動マップ機能を使用すると、同じ名前を持つすべての変数がマッピングされます。
プロット	**自動化** ツール自動化により、次のようなプロットを自動化することができます。 ■ レイアウトをプロットするためなどのファイルの管理 ■ レイアウト内での信号の追加および削除 ■ 信号の値の保存 ■ プロットの開始と停止

パラメータ設定

Automotive Simulation Models　このリリースでは、Automotive Simulation Models のパラメータを設定することができます。Automotive Simulation Models の詳細については、「Automotive Simulation Model（ASM）」(53 ページ)を参照してください。

Model Interface Package for Simulink

Model Interface Package for Simulink 3.1 の機能

トレースファイル生成の機能強化のサポート

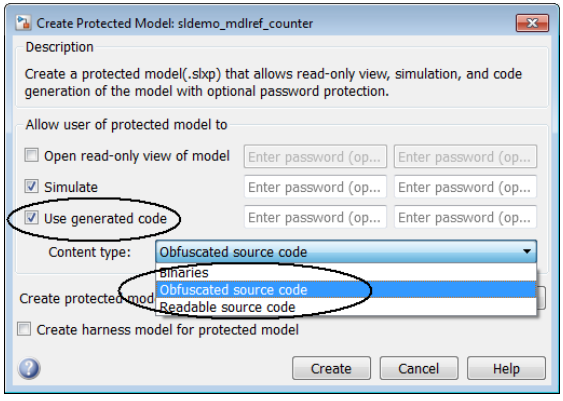
Model Interface Package for Simulink は、Simulink インプリメンテーションコンテナファイルの生成のために、MATLAB R2014a で導入されたパラメータ処理に関する Simulink Coder の機能をサポートしています。

詳細については、「TRC ファイル生成の変更」(35 ページ)を参照してください。

保護された Simulink モデルのサポート

Model Interface Package for Simulink は、Simulink ビヘイビアモデルや保護された参照先モデルを含む Simulink インプリメンテーションコンテナをサポートしています。保護された参照先モデルを作成するには、Simulink モデルの[Create protected models]ダイアログで [Use generated code]チェックボックスを選択する必要があります。Model Interface Package for Simulink は、[Content type]リスト内の次のオプションをサポートしています(パスワード保護を含む)。

- Obfuscated source code
- Readable source code



詳細については、Simulink® Coder™ のドキュメントを参照してください。

[Copy with Identity]/[Paste with Identity]コマンドへの新しいアクセス方法

モデルポートブロックのコンテキストメニューから[Copy with Identity]および[Paste with Identity]コマンドを選択できるようになりました。さらに、Model Interface Package では、これらのコマンドの代わりに次のキーボードショートカットを使用できます。

- [Copy with Identity]:`Ctrl+Alt+C`
- [Paste with Identity]:`Ctrl+Alt+V`

MotionDesk

項目の一覧

本章の内容

MotionDesk 3.7 の新機能

新しくサポートされるプラットフォーム

MotionDesk では、以下のシミュレーションプラットフォームが新たにサポートされます。

- MicroLabBox

計器

計器パネル　MotionDesk には計器パネルがあります。これは、複数の計器を 1 つのパネルにグループ化して、割り当てられたすべての計器を一括して処理できるようにするツールです。これを使用して、たとえば、グループ化した計器の最小化／最大化や、1 つまたは複数のオブザーバへの接続を行うことができます。また、シーンにダッシュボードをビジュアル表示するために、計器パネルを使用することができます。

計器とオブザーバ　計器プロパティの指定または Scene Navigator でのマウス操作により、計器を複数のオブザーバに割り当てることができます。そのため、異なる複数のオブザーバに同じ計器が表示される場合があります。

Scene Navigator では、オブザーバの下に、それに割り当てられているすべての計器がリストされます。

コピーと貼り付け　計器を Windows のクリップボードにコピーして貼り付けることで、計器の複製を作成することができます。

ツール自動化

MotionDesk のツール自動化が拡張されています。次のオブザーバ処理がツール自動化で可能になりました。

- オブザーバの作成
- オブザーバの削除
- デフォルトオブザーバの作成
- オブザーバの設定

MotionDesk 3.7 への移行

MotionDesk 2.2.1 以前からの移行

MotionDesk 2.2.1 以前のバージョンでは、MotionDesk は 3D オブジェクトを VRML フォーマットで使用します。これらのバージョンの MotionDesk で使用していたシーンやカスタムの 3D オブジェクトを使用するには、MotionDesk 3.7 で使用できるように移行する必要があります。詳細については、「Migrating from MotionDesk 2.2.1 and Lower」(🕮『MotionDesk Guide』)を参照してください。

MDX ファイルフォーマットの MotionDesk エクスペリメントの移行

MotionDesk 3.7 では、MDX ファイルフォーマットの古い MotionDesk エクスペリメントを読み取れなくなりました。そのため、2.2 より前のバージョンの MotionDesk エクスペリメントを移行することはできません。

このような古いエクスペリメントを移行するには、MotionDesk 3.0 から MotionDesk 3.6 までのバージョンを使用してください。

Real-Time Testing

Real-Time Testing 2.6 の新機能

新しくサポートされるプラットフォーム	Real-Time Testing は MicroLabBox をサポートしています。 次の Real-Time Testing モジュールは、MicroLabBox ではサポートされません。 ■ rttlib.rs232lib（RS232 インターフェース経由のデータの送受信） ■ rttlib.hostcall
DS1007 マルチプロセッサシステム	ノード上で実行している RTT シーケンスは、DS1007 マルチプロセッサシステムのリモートノードの TRC 変数にアクセスできるようになりました。
データ再生用 MAT ファイル	Real-Time Testing では、dSPACE で再配布される MATLAB DLL を使用できるため、インストールされている MATLAB のバージョンに依存しなくなりました。
新しいクラス	`rttlib.utilities` モジュールには `SequenceProperties` クラスがあり、これを使用して RTT シーケンスのプロパティ（名前、説明、ファイル名、優先度）の読み取り、および SequenceChannel プロパティの読み取り／書き込みを行うことができます。

RTI/RTI-MP および RTLib

項目の一覧

本章の内容

RTI/RTI-MP および RTLib の新機能

トレースファイル生成の機能強化のサポート

RTI および RTI-MP は、MATLAB R2014a で導入されたパラメータ処理に関する Simulink Coder 機能をサポートしています。

詳細については、「TRC ファイル生成の変更」(35 ページ)を参照してください。

MicroLabBox

ソフトウエアサポートの拡張　次の機能がサポートされるようになりました。

- 不揮発性データの処理(NVDATA)

  リアルタイムアプリケーションからボードの不揮発性メモリにデータを格納することができます。このデータは、ハードウエアのシャットダウン後でも再度使用することができます。RTLib や RTI を介して、またはボードの Web インターフェースを介して、64 KB までのデータを管理することができます。

  詳細については、「Nonvolatile Data Handling (NVDATA)」(📖『MicroLabBox Features』)を参照してください。

■ RTI サポートの拡張

FPGA I/O Type 1 ライブラリに、次の新しい ブロックを含む Extras ライブラリが新たに追加されています。

■ LED_BLx

カスタマイズ可能な LED を制御します。

■ Buzzer

ボードのブザーを制御します。

詳細については、「Basic Functions」(📖『MicroLabBox RTI Reference』)を参照してください。

■ DAC_CL1_BLx ブロックでの設定の拡張

特定のチャンネルに Termination mode を設定することができます。この設定を有効にすると、各チャンネルに個別に終了値を設定することができます。この設定を無効にすると、指定したすべてのチャンネルは終了時にハイインピーダンスに設定されます。

詳細については、「DAC_CLASS1_BLx」(📖『MicroLabBox RTI Reference』)を参照してください。

■ モータ制御のサポートの強化

詳細については、「RTI Electric Motor Control Blockset 1.2 の新機能」(165 ページ)を参照してください。

■ Real-Time Testing のサポート

[RTI simulation options]ページの[Enable real-time testing]オプションは、デフォルトでセットされており、変更することができません。

ボードの機能の詳細については、📖『MicroLabBox Features』を参照してください。

DS1007

ソフトウエアサポートの拡張　次の機能がサポートされるようになりました。

■ 不揮発性データの処理(NVDATA)

リアルタイムアプリケーションからボードの不揮発性メモリにデータを格納することができます。このデータは、ハードウエアのシャットダウン後でも再度使用することができます。RTLib や RTI を介して、またはボードの Web インターフェースを介して、64 KB までのデータを管理することができます。

詳細については、「Nonvolatile Data Handling (NVDATA)」(📖『DS1007 Features』)を参照してください。

ボードの機能の詳細については、📖『DS1007 Features』を参照してください。

MicroAutoBox	**パルス幅計測**　DS1511 I/O Board または DS1513 I/O Board を搭載した MicroAutoBox II バージョンで、RTLib および RTI を介したパルス幅計測（PW2D）がサポートされます。接続された信号の High パルスまたは Low パルスを 3.3 µs から 53.6 s の範囲で計測することができます。 詳細については、「パルス幅計測（PW2D）」（🕮『MicroAutoBox 機能解説書』）を参照してください。 **MicroAutoBox I/O ボードの廃止**　次の MicroAutoBox II は、2015 年末で販売終了となります。 ■ MicroAutoBox II 1401/1501 ■ MicroAutoBox II 1401/1504 ■ MicroAutoBox II 1401/1505/1507 ソフトウエアサポートは、少なくとも 2018 年末まで継続されます。DS1511、DS1512、DS1513、および DS1514 の I/O ボードを搭載した MicroAutoBox II の後継バージョンを使用することをお勧めします。MicroAutoBox II 1401/1507 バージョンは引き続きご利用いただけます。 ボードの機能の詳細については、🕮『MicroAutoBox 機能解説書』を参照してください。
MATLAB R2015b のサポート対象外の新機能	ルックアップテーブルに[Evenly spaced breakpoints]オプションを使用する場合は、関連するブロックのパラメータは変数記述ファイルに生成されません。実行時に ControlDesk Next Generation の Table Editor などを使用して、これらのパラメータにアクセスすることはできません。
MATLAB R2015b を使用する場合の制限事項	MATLAB R2015b で RTI/RTI-MP を使用する場合は、下記の制限事項に注意してください。 ■ Triggered subsystem における絶対時間のサポート 絶対時間のスケーリングに誤りがある場合があります。この動作の理由はまだ不明ですが、MathWorks 社ではバージョン R2015a および R2015b の Simulink に対してパッチを発行する予定です。
廃止されたバッチファイル	`build.bat` バッチファイルは、マルチプロセッサアプリケーションをビルドする `rtimp_build` コマンドの代わりに使用されていました。`build.bat` バッチファイルは廃止されました。

RTI/RTI-MP および RTLib の移行上の注意点

TRC ファイル生成の変更	RTI および RTI-MP での TRC ファイル生成に関するいくつかの修正事項に注意する必要があります。「TRC ファイル生成の変更」(35 ページ)を参照してください。
MATLAB R2015b で修正された機能	Simulink の[Configuration Parameters]ダイアログで次の変更が行われました。 ■ [Hardware Implementation]ページの設定が変更されています。量産ハードウエアを若干異なる方法で指定する必要があります。 関連する RTI 設定も調整されています。そのため、以前に古いバージョンの RTI に接続していたすべての MATLAB インストレーションで Simulink Preference をリセットすることをお勧めします。 ■ [Optimization - Signals and Parameters]ページの[Inline parameters]設定が[Default parameter behavior]設定に変更されています。
Real-Time Testing のサポートが DS1007 で常に有効	[RTI simulation options]ページの[Enable real-time testing]オプションは、DS1007 PPC Processor Board では常に有効になります。これは、新しい dSPACE プラットフォーム DS1007 および MicroLabBox でのデフォルトの動作です。既存の設定は、RTI および RTI-MP によって自動的に更新されます。

RTI Bypass Blockset

項目の一覧

本章の内容

RTI Bypass Blockset 3.5 の新機能

RTI Bypass Blockset

XCP 1.3 のサポート　RTI Bypass Blockset は、XCP 1.3 規格に基づいた XCP 固有の `IF_DATA` エントリを含む A2L ファイルをサポートしています。

CAN FD のサポート　RTI Bypass Blockset は、XCP トランスポートレイヤとして、従来の CAN プロトコルに加えて CAN FD（CAN with Flexible Data Rate）もサポートするようになりました。つまり、ECU とプロトタイピングツール間の通信を、CAN や CAN FD 経由で行うことができます。従来の CAN プロトコルと比較して、CAN FD プロトコルでは 1 MBit/s を超えるデータレートと、メッセージ当たり 64 バイトまでのペイロードを使用することができます。

現時点で、市場には 2 つの CAN FD プロトコル、*Non-ISO CAN FD プロトコル*（Bosch 社が開発したオリジナルの CAN FD プロトコル）と *ISO CAN FD プロトコル*（2015 末にリリース予定の次期 ISO 11898-1:2015 規格に準拠した CAN FD プロトコル）があり、これらはお互いに互換性がありません。RTI Bypass Blockset は両方の CAN FD プロトコルをサポートしています。

XCP on CAN FD を介したサービスベースのバイパス処理の場合、RTI Bypass Blockset は既に確立されている XCP on CAN バイパスインターフェースタイプを使用します。CAN FD フォーマットの記述を含むデータベースファイルのインポート後、CAN FD サポートを有効にして、Setup ブロックで CAN FD 固有の設定を行う必要があります。

「[Options]ページ(XCP on CAN の RTIBYPASS_SETUP_BLx)」(🕮『RTI Bypass Blockset リファレンス』)を参照してください。

ブロックセットは、DS4342 CAN FD Interface Modules を搭載した dSPACE プラットフォームに対してのみ、XCP on CAN FD を介したサービスベースのバイパス処理をサポートしています。

RTI Bypass Blockset の MATLAB API

RTI Bypass Blockset の機能拡張のサポート RTI Bypass Blockset の MATLAB API は、RTI Bypass Blockset の機能拡張をサポートしています。

🕮『RTI Bypass Blockset MATLAB API Reference』を参照してください。

RTI Bypass Blockset 3.5 への移行

以前の RTI Bypass Blockset バージョン 3.x および 2.x のモデルの使用

最新のリリースには、以前のブロックセットバージョン 3.x および 2.x と互換性のある RTI Bypass Blockset 3.5 が含まれています。ただし、いくつかの注意事項があります。

■ *RTI Bypass Blockset 2.5 以前のモデルを使用する場合*

以前の RTI Bypass Blockset バージョンと比較して、データ管理が変更されています。RTI Bypass Blockset 2.5 以前でビルドした Simulink モデルを RTI Bypass Blockset 3.4 で開くと、古い Data Dictionary ファイル(ファイル名の拡張子.dd)は、Setup ブロックに格納されている情報を使用して新しい Data Dictionary ファイル(.vdb)に置き換えられます。これは、Setup ブロックダイアログを開いて[OK]をクリックして閉じるか、または Read/Write/Upload/Download ブロックダイアログを開いて[Variables]ページの[Fill Variable Selector]ボタンをクリックするとすぐに行われます。

RTI Bypass Blockset 3.4 で保存したモデルを RTI Bypass Blockset 2.5 以前のバージョンで使用する場合、バージョン 2.5 およびそれ以前のブロックセットに必要なモデルの Data Dictionary ファイル(ファイル名の拡張子.dd)が作成されます。これは、Setup ブロックで A2L ファイルを更新するか、または Read/Write/Upload/Download ブロックを開いて[Variables]ページの[Fill Variable Selector]ボタンをクリック

するとすぐに行われます。RTI Bypass Blockset 3.5 で作成された Data Dictionary ファイル（*.vdb）は、ディスク上にそのまま残ります。

RTI Bypass Blockset で Data Dictionary を再作成するには、Setup ブロックで指定されたデータベースファイルが指定された場所からアクセス可能で、これらのファイルが変更されていないことが必須条件となります。

■ *RTI Bypass Blockset 2.6 から 3.4 までのモデルを使用する場合*

RTI Bypass Blockset 2.6 から RTI Bypass Blockset 3.4 までででビルドされた Simulink モデルを RTI Bypass Blockset 3.5 で開くと、古い Data Dictionary ファイルが新しい Data Dictionary ファイルに置き換えられます。ただし、新しい Data Dictionary ファイルは以前のバージョンの RTI Bypass Blockset では使用することができません。RTI Bypass Blockset 2.6 から RTI Bypass Blockset 3.4 までででビルドされたモデルを再利用するには、Setup ブロックで指定されているデータベースファイル（A2L ファイル）を再インポートして、以前のバージョンの RTI Bypass Blockset で適切なデータベースを作成する必要があります。

RTI CAN MultiMessage Blockset

項目の一覧

本章の内容

RTI CAN MultiMessage Blockset 4.2 の新機能

CAN FD に関連した拡張

RTI CAN MultiMessage Blockset は、CAN FD メッセージの使用に関連して、次のように拡張されています。

SCALEXIO システムでの CAN FD のサポート　このブロックセットは、DS2671 Bus Board を搭載した SCALEXIO システムでの CAN FD メッセージの使用をサポートするようになりました。

「CAN FD の使用の基礎」(🕮『RTI CAN MultiMessage Blockset Reference』)を参照してください。

ISO CAN FD プロトコルのサポート　このブロックセットでは、既にサポートされている Bosch 社開発のオリジナルの CAN FD プロトコルに加えて、ISO CAN FD プロトコル(2015 年末にリリース予定の次期 ISO 11898-1:2015 規格に準拠)もサポートされます。RTICANMM ControllerSetup ブロックで使用する CAN FD プロトコルを選択することができます。

「CAN FD の使用の基礎」(🕮『RTI CAN MultiMessage Blockset Reference』)を参照してください。

アービトレーションフェーズとデータフェーズのサンプリングポイント
CAN FD メッセージの使用を有効にする場合、CAN FD メッセージ送信のアービトレーションフェーズとデータフェーズのサンプリングポイントを CAN ビット周期のパーセンテージとして指定することができます。サンプリングポイントは、同期のために使用されます。

「Setup Page (RTICANMM ControllerSetup)」(🕮『RTI CAN MultiMessage Blockset Reference』)を参照してください。

不明確なバイトオーダー形式のサポート

RTI CAN MultiMessage Blockset は、AUTOSAR システムデスクリプションファイルで定義できる、不明確なバイトオーダーの信号もサポートするようになりました。

RTI CAN MultiMessage Blockset 4.2 への移行

RTI CAN MultiMessage Blockset の以前のバージョンで作成したモデルの使用

RTI CAN MultiMessage Blockset の以前のバージョンで作成されたモデルを再利用するには、CAN の設定に変更を加える前に、すべての RTICANMM ブロックの S-function を更新して保存する必要があります。

モデル内のすべての RTICANMM ブロックに対して新しい S-function を一度に作成するには、モデルを開いた後で次のいずれかを実行します。

- MATLAB コマンドウインドウに `rtimmsu_update('System', gcs)`と入力します。

  このコマンドおよびオプションの詳細を確認するには、MATLAB コマンドウインドウに `help rtimmsu_update` と入力します。
- RTICANMM GeneralSetup ブロックの[Options]メニューから[Create S-Function for all CAN Blocks]コマンドを選択します。

詳細については、「RTICANMM に関する制限事項」(🕮『RTI CAN MultiMessage Blockset Reference』)を参照してください。

バージョン 4.0 より前の RTI CAN MultiMessage Blockset で生成されたコードを使用した場合のコンパイラメッセージ

バージョン 4.0 より前の RTI CAN MultiMessage Blockset で生成されたコードを使用すると、シミュレーションモデルのビルドプロセス中に、`<<argument of type "can_tp1_canChannel *" is incompatible with parameter of type "DsTCanCh">>`というフレーズを含む複数のコンパイラ警告メッセージが表示されます。これはデータ型が変更されたためです。これらの警告は無視してかまいません。最新バージョンのブロックセットを使用して RTICANMM コードを再生成すると、表示されなくなります。

既存のチェックサムアルゴリズムの使用

CAN メッセージを含むアプリケーション用に本来開発されたチェックサムアルゴリズムは、CAN FD メッセージを含むアプリケーションで再利用することはできません。これは、CAN FD に新しいメッセージタイプが含まれているか、データフィールドが長いためです。既存のチェックサムアルゴリズムは、標準的な CAN メッセージのみ含むアプリケーションでは引き続き使用することができます。CAN FD アプリケーションの場合は、チェックサムアルゴリズムを適合させる必要があります。

RTI Electric Motor Control Blockset

RTI Electric Motor Control Blockset 1.2 の新機能

新しいブロック

RTI Electric Motor Control Blockset では、次の新しいブロックを利用することができます。

- EMC_ENDAT_BLx: EnDat インターフェースに接続された絶対エンコーダをモータ制御の入力センサとして使用します。

詳細については、📖『RTI Electric Motor Control Blockset Reference』を参照してください。

RTI FPGA Programming Blockset

項目の一覧

本章の内容

RTI FPGA Programming Blockset 3.0 の新機能

Xilinx®のサポート

RTI FPGA Programming Blockset で、Xilinx 設計ツールの以下の製品とバージョンがサポートされるようになりかした。

Xilinx 設計ツールのバージョン	オペレーティングシステム	MATLAB バージョン
Vivado 2015.2 （64 ビットバージョン）	Windows 7 Business、Ultimate、Enterprise SP1（64 ビット版）	64 ビットバージョン： ■ MATLAB R2014a ■ MATLAB R2014b ■ MATLAB R2015a

サポートされる dSPACE プラットフォーム

次の dSPACE プラットフォームは、RTI FPGA Programming Blockset 3.0 でサポートされます。

- MicroAutoBox（MicroAutoBox II 1401/1511/1514 および MicroAutoBox II 1401/1513/1514）
- MicroLabBox
- モジュール型システム（DS5203（7K325）および DS5203（7K410））
- SCALEXIO（DS2655）

次のハードウエアは Xilinx Vivado でサポートされません。そのため、RTI FPGA Programming Blockset 3.0 は、既存の FPGA モデル INI ファイル用のプロセッサインターフェースのビルドのみをサポートしています。

- DS5203 FPGA Board（SX95）
- DS5203 FPGA Board（LX50）
- MicroAutoBox II 1401/1511/1512
- MicroAutoBox II 1401/1512/1513

バージョン 2.9 以前の RTI FPGA Programming Blockset のみ、DS5203（SX95）および DS5203（LX50）ボードと、DS1512 I/O Board（MicroAutoBox II 1401/1511/1512 および MicroAutoBox II 1401/1512/1513）の Xilinx ISE および FPGA モデリングをサポートしています。Vivado の導入により、MATLAB Release R2013b 以降、Xilinx では Xilinx System Generator for DSP と ISE Design Suite の組み合わせがサポートされなくなりました。

DS2655 FPGA Base Board フレームワークの機能拡張

DS2655 FPGA Base Board のフレームワークは、次のように拡張されています。

複数のクロック周期　最大 10 個の個別のクロック周期を使用して、FPGA 設計の特定の部分をモデリングすることができます。クロック周期の範囲は 1.6e-7 s ... 6.25e-10 s（6.25 MHz ... 1600 MHz）です。

FPGA_SETUP_BL ブロックの[Subsystems Clocks]ページでは、個別のクロック周期を持つサブシステムに対してクロック周期を設定することができます。

DS2655M1 Multi-I/O Module フレームワークの拡張

DS2655M1 Multi-I/O Module のフレームワークは、次のように拡張されています。

- FPGA_IO_READ_BLx ブロックの Digital In ファンクションと FPGA_IO_Write_BLx ブロックの Digital InOut ファンクションに新しいポートが追加されています。Threshold Ack ポートでは、I/O チャンネルでそのしきい値電圧が現在更新されているかどうか、または新しいしきい値を設定できるかどうかを示すフラグが出力されます。

新しい DS2655M2 Digital I/O Module のサポート

RTI FPGA Programming Blockset では、DS2655M2 Digital I/O Module の新しい *DS2655M2 I/O Module* フレームワークが追加されました。

DS2655M2 Digital I/O Module には、ビット単位データやシリアル通信を処理できる 32 個の多用途デジタル I/O チャンネルがあります。このフレームワークの主な機能は、1 つ以上のデジタルチャンネルを使用して特定の I/O 機能を実装する I/O ファンクションです。

次の I/O ファンクションを使用することができます。

- Digital In

  ビット単位のアクセスを提供する最大 32 個のデジタル入力ファンクション

- Digital Out

  ビット単位のアクセスを提供する最大 32 個のデジタル出力ファンクション

- Digital Out-Z

  ビット単位のアクセスとハイインピーダンス出力状態（トライステート）を提供する最大 16 個のデジタル出力ファンクション

- RS232 Rx

  RS232 ネットワークからデータ値を受信する最大 8 個のシリアルファンクション

- RS232 Tx

  データ値を RS232 ネットワークに転送する最大 8 個のシリアルファンクション

- RS485 Rx

  シンプレックスモードで RS485 ネットワークからデータ値を受信する最大 8 個のシリアルファンクション

- RS485 Rx/Tx

  半二重モードで RS485 ネットワークとデータ値を交換する最大 8 個のシリアルファンクション

- RS485 Tx

  シンプレックスモードで RS485 ネットワークへデータ値を送信する最大 8 個のシリアルファンクション

Multi-I/O フレームワーク搭載の FPGA1401Tp1 の拡張

MicroAutoBox のフレームワークは、次のように拡張されています。

複数のクロック周期　最大 10 個の個別のクロック周期を使用して、FPGA 設計の特定の部分をモデリングすることができます。クロック周期の範囲は 1.6e-7 s ...6.25e-10 s（6.25 MHz ...1600 MHz）です。

FPGA_SETUP_BL ブロックの[Subsystems Clocks]ページでは、個別のクロック周期を持つサブシステムに対してクロック周期を設定することができます。

関連トピック

基礎

- 「RTI FPGA Programming Blockset 3.0 への移行」(170 ページ)

RTI FPGA Programming Blockset 3.0 への移行

目的

既存のモデルの移行方法は、使用するブロックセットのバージョンによって異なります。

RTI FPGA Programming Blockset 1.0 から 3.0 への移行

RTI FPGA Programming Blockset 1.0（dSPACE Release 6.4 で提供）は完全に実装されたものではなかったため、これを使用して実装したモデルは手動で移行する必要があります。最新の dSPACE RTI 環境に準拠したモデルのモデル化、ビルド、および実行を行うために、RTI FPGA Programming Blockset の各ブロックを新しいブロックに置き換える必要があります。

スクリプトインターフェースの更新機能は、RTI FPGA Programming Blockset 1.0 をサポートしていません。

RTI FPGA Programming Blockset 1.1 以降から 3.0 への移行

バージョン 1.1 以降の RTI FPGA Programming Blockset を使用して FPGA アプリケーションを実装した場合、これを RTI FPGA Programming Blockset 3.0 で使用するには、FPGA フレームワークを更新する必要があります。この場合に、スクリプトインターフェースを使用することができます（「スクリプトインターフェースを使用した FPGA フレームワークの更新」(170 ページ)を参照）。

MATLAB R2008b 以前のバージョンから MATLAB R2011b 以降のバージョンに更新した場合は、フレームワークも更新する必要があります。

スクリプトインターフェースを使用した FPGA フレームワークの更新

移行を開始する前に、モデルのバックアップを作成することをお勧めします。

スクリプトインターフェースには、フレームワークを更新するための `FPGAFrameworkUpdate` メソッドが用意されています。ブロックパラメータをその初期値に設定するか、変更せずにそのまま使用するかを指定することができます。

ブロックパラメータの値を変更せずに FPGA フレームワークを更新する場合

```
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 <SimulinkHandle>)
```

このスクリプトでは、Simulink Handle で指定されているモデル／サブシステム内のすべてのサブシステムが処理されます。最新のフレームワークバージョンにアップデートしても、ブロックのパラメータは変更されません。

例：次のスクリプトは、*MyProcModel* という名前のプロセッサモデルの中にあるすべての FPGA サブシステムに対して FPGA フレームワークを更新します。ブロックパラメータの指定された値は変更されません。

```
ProcModelHandle = get_param('MyProcModel','handle')
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 ProcModelHandle)
```

FPGA フレームワークを更新して、ブロックパラメータの値をその初期値にリセットする場合

```
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 <SimulinkHandle>, 'ReInit')
```

このスクリプトでは、Simulink Handle で指定されているモデル／サブシステム内のすべてのサブシステムが処理されます。最新のフレームワークバージョンに更新すると、ブロックのパラメータはそれぞれの初期値にリセットされます。

```
ProcModelHandle = get_param('MyProcModel','handle')
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 ProcModelHandle,'ReInit')
```

dSPACE Release 2015-B と互換性のない ConfigurationDesk カスタムファンクション

DS2655 FPGA Base Board および DS2655M1 Multi-I/O Module を搭載した SCALEXIO システム対象

dSPACE Release 2013-A の RTI FPGA Programming Blockset 2.5 を使用して生成されたカスタムファンクションと、カスタムファンクションを含むリアルタイムアプリケーション（*.rta）は、dSPACE Release 2015-B と互換性がありません。使用可能なカスタムファンクションを作成するには、dSPACE Release 2015-B の RTI FPGA Blockset 3.0 を使用して FPGA モデルを再ビルドする必要があります。

RTI LIN MultiMessage Blockset

項目の一覧

本章の内容

RTI LIN MultiMessage Blockset 2.5.1 の新機能

不明確なバイトオーダー形式のサポート

RTI LIN MultiMessage Blockset は、AUTOSAR システムデスクリプションファイルで定義できる、バイトオーダーが不明確な信号もサポートするようになりました。

RTI LIN MultiMessage Blockset 2.5.1 への移行

RTI LIN MultiMessage Blockset の以前のバージョンで作成したモデルの使用

RTI LIN MultiMessage Blockset の以前のバージョンで作成されたモデルを再利用するには、LIN の設定に変更を加える前に、すべての RTILINMM ブロックの S-function を更新して保存する必要があります。

モデル内のすべての RTILINMM ブロックに対して新しい S-function を一度に作成するには、モデルを開いた後で次のいずれかを実行します。

- MATLAB コマンドウインドウに `rtimmsu_update('System', gcs)` と入力します。

  このコマンドおよびオプションの詳細を確認するには、MATLAB コマンドウインドウに `help rtimmsu_update` と入力します。

■ RTILINMM GeneralSetup ブロックの[Options]メニューから[Create S-Function for all LIN Blocks]コマンドを選択します。

詳細については、「RTI LIN MultiMessage Blockset の制限事項」(📖『RTI LIN MultiMessage Blockset リファレンス』)を参照してください。

SCALEXIO Firmware

SCALEXIO Firmware 3.3 の新機能

新しくサポートされるハードウエア	SCALEXIO ファームウエアで、次の SCALEXIO ハードウエアモジュールのサポートが追加されています : DS2655M2 Digital I/O Module

SystemDesk

項目の一覧

本章の内容

SystemDesk 4.5 の新機能

項目の一覧

本章の内容

新しい一般機能

目的

SystemDesk 4.5 には、次の一般機能が新たに追加されています。

SystemDesk 4.5 でサポートされる AUTOSAR リリース

AUTOSAR Release 4.2.1 のモデリングのサポート SystemDesk 4.5 では、AUTOSAR Release 4.2.1 に準拠したソフトウエアおよびシステムアーキテクチャのモデリングがサポートされます。

最近の AUTOSAR リリースとの互換性 SystemDesk では、AUTOSAR Release 4.1.3/4.1.2/4.1.1/4.0.3 の AUTOSAR ファイルをやり取りすることができます。

最新の AUTOSAR リリースとの互換性 SystemDesk 4.5 では、最新の AUTOSAR Release 4.2.2 に準拠した AUTOSAR ファイルのインポートをサポートしています。ただし、この AUTOSAR リリースで導入されたエレメントは、SystemDesk にインポートされません。

SystemDesk から AUTOSAR Release 4.2.2 に準拠した AUTOSAR ファイルをエクスポートすることができます。

ソフトウエアアーキテクチャのモデリング

改善点

このバージョンの SystemDesk では、ソフトウエアアーキテクチャのモデリングが次の点で改善されています。

- SystemDesk で、自動的に作成される定数の名前を通信仕様の初期値に指定できるようになりました。

  この命名の詳細については、「General page (preferences)」(📖『SystemDesk 4.x Reference』)を参照してください。

- SystemDesk に[Start Page]が追加され、ユーザドキュメントやその他の SystemDesk 資料にすばやくアクセスできるようになりました。

- Project Manager などの SystemDesk のコントロールバーで、SystemDesk のメニューから[Navigate back]および[Navigate forward]コマンドを使用して移動できるようになりました。

  コマンドの詳細については、「Navigate Forward」(📖『SystemDesk 4.x Reference』)および「Navigate Backward」(📖『SystemDesk 4.x Reference』)を参照してください。

- Project Manager 検索が改善され、エレメントの AUTOSAR タイプと、使用可能な場合は ECU コンフィギュレーションが表示されます。これにより、エレメントの[Properties]ダイアログにすばやくアクセスすることができます。

  SystemDesk の Project Manager およびエレメントの検索の詳細については、「Project Manager」(📖『SystemDesk 4.x Reference』)を参照してください。

- AUTOSAR テンプレートのインポートが改善され、テンプレートにすばやくアクセスして、標準化された SystemDesk プロジェクトを標準の AUTOSAR タイプと標準化されたパッケージ構造で作成することができます。

  テンプレートのインポートの詳細については、「Import AUTOSAR Templates」(📖『SystemDesk 4.x Reference』)を参照してください。

ダイアグラムの使用

SystemDesk のグラフィカルモデリング用ダイアグラムの改善

このバージョンの SystemDesk では、グラフィカルモデリングが改善されました。SystemDesk のすべてのダイアグラム、つまり、ソフトウエアコンポーネントのモデリングと接続用のコンポジションダイアグラム、アトミックソフトウエアコンポーネントのポートとインターフェースのモデリング用のコンポーネントダイアグラム、およびベーシックソフトウエアとアプリケーションソフトウエアの接続用のサービス接続ダイアグラムが改善されています。

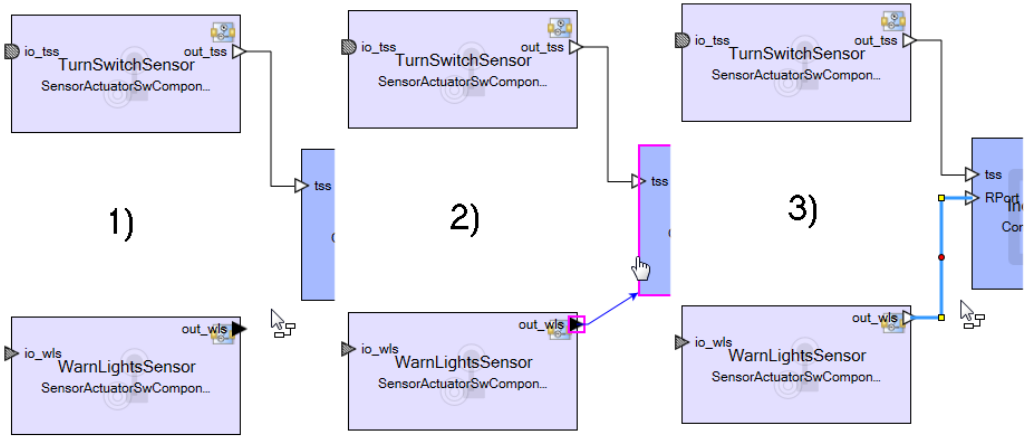
Edit モードと Connection モード

このバージョンの SystemDesk では、コンポジションダイアグラムとサービス接続ダイアグラムでのエレメント処理とエレメント接続を簡易化するために、*Edit* モードと *Connection* モードが導入されています。

ダイアグラムを開いたときに有効になる Edit モードで、エレメントの追加と配置を行うことができます。

Connection モードは特別なカーソル( )で示され、これに切り替えると、ソフトウエアコンポーネントを接続することができます。ソフトウエアコンポーネントの接続は、ドラッグアンドドロップで行うことができます。2 つのポート、1 つのポートと SWC、または 2 つの SWC を接続することができます。SystemDesk はポートを追加し、可能であれば、接続する SWC へのインターフェースを追加します。

下の図は、ポートを別のソフトウエアコンポーネントに割り当てられているインターフェースに接続する例を示しています。SystemDesk は、ターゲットとなるソフトウエアコンポーネントにポートを追加し、追加したポートに開始ポートと同じインターフェースを割り当てます。



その他の改善点

その他の改善点を以下に示します。

- ダイアグラムと接続の自動レイアウトの改善
- Simulink と同じような、親と子のコンポジションダイアグラム間の移動動作
- コンポジションダイアグラムおよびサービスコネクションダイアグラム内のソフトウエアコンポーネント、ポート、およびインターフェースの名前による検索
- 互換性のない接続に関するエラーメッセージの表示
- 利便性が向上したダイアグラムに追加するエレメントの選択
- ダイアグラムでのエレメント表示設定の改善:コンポジションダイアグラムでポートと接続を表示可能など
- SystemDesk の環境設定でのソフトウエアコンポーネント、接続、ポート、およびインターフェースに対するデフォルト外観の指定

補足

SystemDesk のダイアグラムの使用に関する詳細は、「ダイアグラムの使用」(🕮『SystemDesk 4.x Guide』)を参照してください。

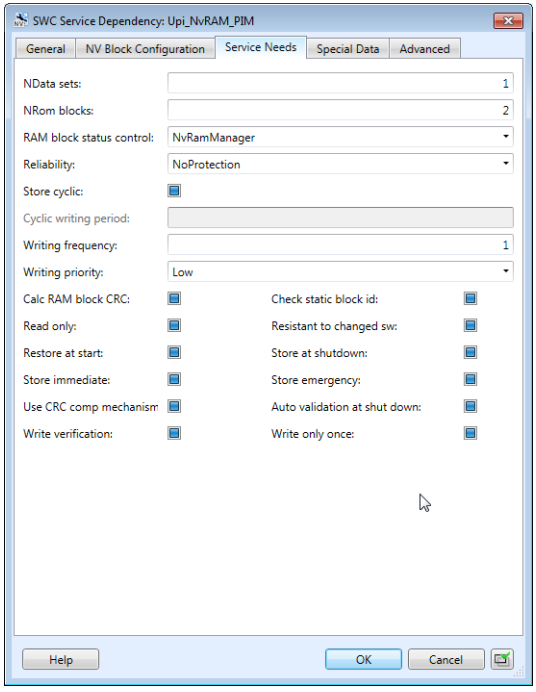
ECU コンフィギュレーション

NVRAM マネージャ

SystemDesk は、NVRAM マネージャのベーシックソフトウエアコンポーネントをサポートするようになりました。

ソフトウエアアーキテクチャのモデリング　NVRAM マネージャの読み取り／書き込み操作を要求するアトミック SWC の NV ブロックサービス要求を簡単にモデリングすることができます。さらに、指定した NV ブロックサービス要求を管理するための NV ブロックソフトウエアコンポーネントをモデリングすることができます。

次の図は、NV ブロックサービス要求を指定する例を示しています。



ECU コンフィギュレーション　NvM ベーシックソフトウエアモジュールを ECU コンフィギュレーションに追加することができます。

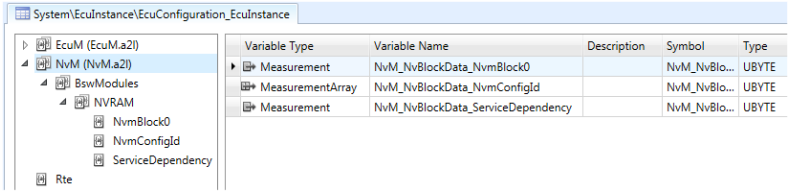
SystemDesk では、次のコマンドを使用して NVRAM マネージャを自動設定および生成することができます。

- 指定したサービス要求に従って NvM モジュールコンフィギュレーションを更新
- NVRAM マネージャのベーシックソフトウエアコンポーネントを生成。これには、ベーシックソフトウエアコンポーネントと、サービスを要求するアプリケーションソフトウエアコンポーネントとの自動接続が含まれます。
- 仮想検証のために NVRAM マネージャをシミュレートするためのコードを生成

システムのシミュレーション　NVRAM マネージャに対して生成された A2L 変数を使用して、試験を実行することができます。

このために、SystemDesk は、それをモニタリングするための NVRAM を記述する配列を生成します。

次の図は、NVRAM マネージャ例に対して生成された A2L 変数を示しています。



ベーシックソフトウエアの記述

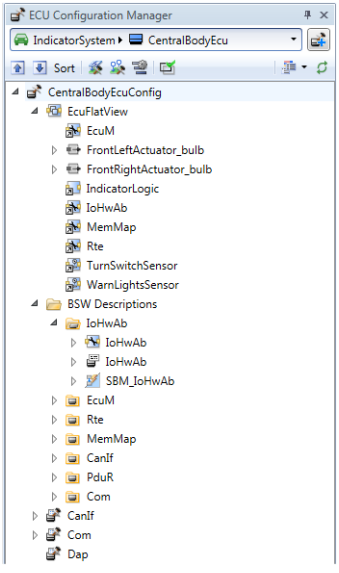
SystemDesk では、ベーシックソフトウエアコンポーネントのエレメントに、AUTOSAR に準拠したベーシックソフトウエアモジュール記述テンプレートを使用するようになりました。そのため、SystemDesk による ECU の設定は、AUTOSAR に完全に準拠しています。これにより、ベーシックソフトウエアモジュールを、AUTOSAR に準拠したサードパーティ製ツールと交換することができます。

次の表では、SystemDesk の ECU Configuration Manager のエレメントについて説明します。

タイプ	説明
ECU コンフィギュレーション	ベーシックソフトウエアと単一 ECU の RTE の全体的な設定
ECU フラットビュー	ECU インスタンスにマッピングされたすべてのアトミックソフトウエアコンポーネントのプロトタイプに加えて、ベーシックソフトウエアと RTE で使用するソフトウエアコンポーネントのプロトタイプも含まれます。 コンポジション SW コンポーネントによってアプリケーションソフトウエアに導入された階層が、コンポジションを削除してアトミックソフトウエアコンポーネントを直接接続することによって解消されるという意味において、ECU フラットビューはフラットです。ただし、SystemDesk はデリゲーションポートも表示します。
SW コンポーネントプロトタイプ	アプリケーションまたはベーシックソフトウエアコンポーネントのプロトタイプ
…	
ポート	コンポジション SW コンポーネントのデリゲーションポート
インターフェース	デリゲーションポートに割り当てられるインターフェース
…	
BSW デスクリプション	以下のエレメント BSW デスクリプションは、SystemDesk によって自動的に管理されます。通常、ユーザはモジュールコンフィギュレーションのパラメータのみを変更し、BSW デスクリプションは設定に合わせて自動的に調整されます。
BSW デスクリプション	

タイプ			説明
		SW コンポーネントタイプ	アプリケーションソフトウエアに対するベーシックソフトウエアモジュールの AUTOSAR インターフェースを定義します。
		BSW モジュールデスクリプション	ベーシックソフトウエアモジュールの標準インターフェースを他のベーシックソフトウエアモジュールに定義します。
		SWC-BSW マッピング	*SW コンポーネントタイプ*のエレメントを *BSW モジュールデスクリプション*のエレメントにマッピングします。
	…		
	モジュールコンフィギュレーション		ベーシックソフトウエアモジュールコードと *BSW デスクリプション*を生成するためのベーシックソフトウエアコンポーネントのコンフィグレーションパラメータを指定します。
	…		

次の図は、ECU Configuration Manager での ECU コンフィギュレーション例を示しています。



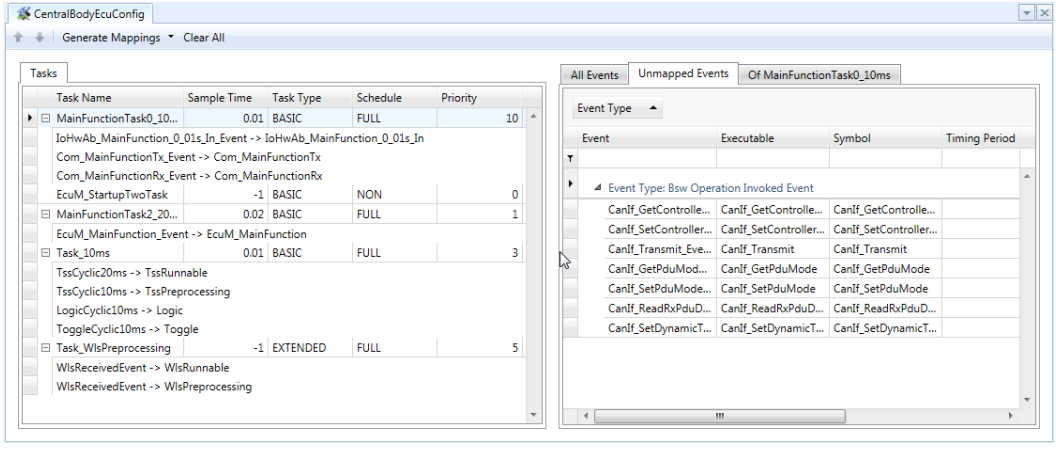
変更されたランナブルエンティティの OS タスクへのマッピング

このバージョンの SystemDesk では、RTE イベントを介して、AUTOSAR に準拠したランナブルと BSW のスケジュール可能なエンティティをマッピングすることができます。

Runnable Mapping Editor は、実行をトリガする、関連するすべての RTE イベントのリスト（つまり、ランナブルエンティティと BSW のスケジュール可能なエンティティ）を提供します。

このエディタで OS タスクを作成し、これをトリガするイベントを介して実行ファイルをマッピングすることができます。各 OS タスクに対して、実行を有効にする順序を指定することができます。

次の図は、ECU コンフィギュレーション例に対する Runnable Mapping Editor を示しています。



Generate Mappings コマンドの改善

このバージョンでは、OS タスクへのランナブルのマッピングを生成する SystemDesk の機能が改善されました。

Runnable Mapping Editor で使用できる[Generate Mappings]コマンドにより、マッピングされていないイベントの OS タスクへのマッピングを生成できるようになりました。このコマンドは、イベントを既存の OS タスクにマッピングするか、必要な場合は新しい OS タスクを作成してそれにマッピングします。

[Generate Mappings]コマンドの詳細については、「Runnable Mapping Editor」(🕮『SystemDesk 4.x Reference』)を参照してください。

RTE 生成の改善

このバージョンの SystemDesk では、RTE API のサポートを向上させるよう、RTE 生成が改善されています。

SystemDesk の RTE API サポートに関する詳細については、dSPACE サポートにお問い合わせください。

RTE インターベンションの改善

デリゲーションポート用 RTE インターベンション　デリゲーションポートのデータエレメントへの読み取り／書き込みアクセスのための RTE インターベンションを作成できるようになりました。これにより、仮想検証のためにシミュレーションでコンポジションソフトウエアコンポーネントをテストすることができます。

RTE インターベンションの設定の簡易化　RTE インターベンションエディタにより、RTE インターベンションをより簡単に作成できるようになりまし

た。このエディタの[Select elements for DAP interventions points]コマンドを使用して、V-ECU の外部からアクセスできる RTE インターベンションを作成します。[Select elements for RTE service ports intervention points]コマンドを使用して、RTE サービスポートを介して V-ECU の内部からアクセスできる RTE インターベンションを作成することができます。

V-ECU ウィザードを使用した V-ECU の迅速な生成

V-ECU ウィザードは SystemDesk の System Wizard に代わるもので、指定したアトミックソフトウエアコンポーネントまたはコンポジションソフトウエアコンポーネントに対して V-ECU を迅速に生成できるよう改善されています。

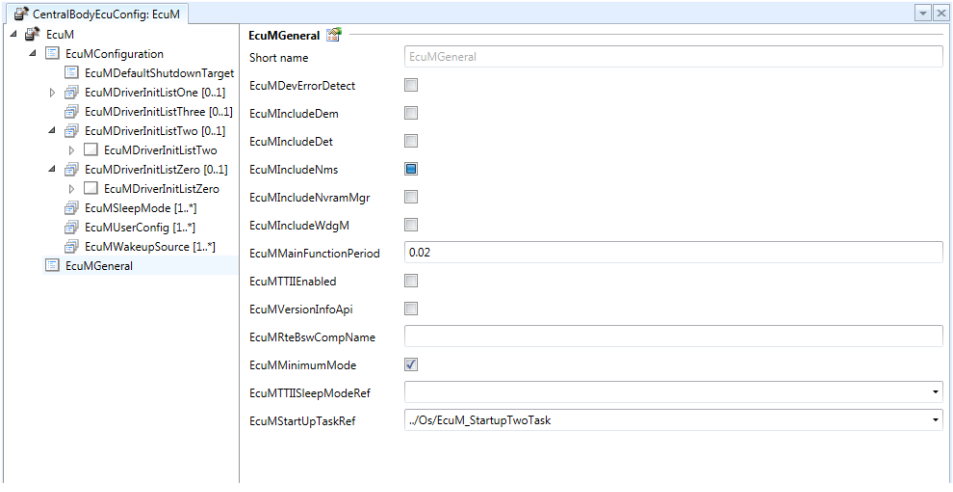
このウィザードを使用して、システムの作成、ECU インスタンスの追加または作成、ECU コンフィギュレーションの作成、およびその自動設定を行うことができます。結果として、SystemDesk は V-ECU インプリメンテーションを生成し、ユーザはそれを VEOS 用に構築することができます。

これにより、ソフトウエアコンポーネントをバーチャルファンクショナルバス(VfB)レベルですばやくシミュレートすることができます。SystemDesk は、選択した SWC を*バーチャルファンクショナルバス*(VFB)として動作する単一の ECU にマッピングします。

ベーシックソフトウエアモジュールを設定するエディタの改善

ベーシックソフトウエアモジュールコンフィギュレーションを設定する BSW Module Editor は再設計され、分かりやすさと操作性が向上しました。

下の図は、新しいレイアウトを示しています。



補足

SystemDesk での ECU の設定と V-ECU インプリメンテーションの生成の詳細については、「Configuring ECUs and Generating V-ECU Implementations」(🕮『SystemDesk 4.x Guide』)を参照してください。

システムのシミュレーション

V-ECU のデバッグの改善

V-ECU を Release 設定と Debug 設定のどちらでビルドするかを指定できるようになりました。V-ECU を Debug 設定でビルドすると、V-ECU のビルド中に必要な情報が生成され、V-ECU をデバッグすることができます。

Debug 設定を使用できるかどうかは、選択したシミュレーションターゲットに依存します。

補足

V-ECU のデバッグの詳細については、「シミュレーションのデバッグおよび分析」(🕮『SystemDesk 4.x Guide』)を参照してください。

SystemDesk の自動化

自動化によるエレメントのリスト作成の改善

SystemDesk の自動化機能は、自動化によってエレメントのリストを作成できるように改善されました。addNew メソッドを持つ各自動化インターフェースに対して、addNewRange メソッドが追加されています。新しく導入された addNewRange メソッドでは、多数のエレメントを作成する際のパフォーマンスが改善されます。

自動化によってパッケージエレメントを作成する次の例について考えます。

```
def AddNewPackages(rootPackage, amount):
    packages = []
    for i in range(0, amount):
        package = rootPackage.ArPackages.AddNew()
        package.ShortName = "package_" + str(i)
        packages.append(package)
    return packages
```

新しく導入された addNewRange メソッドを以下のように使用すると、パフォーマンスが改善されます。

```
def AddNewRangeOfPackages(rootPackage, amount):
    shortNames = []
    for i in range(0, amount):
        shortName = "package_" + str(i)
        shortNames.append(shortName)
    packages = rootPackage.ArPackages.AddNewRange(shortNames)
    return packages
```

補足

SystemDesk の自動化の詳細については、「Programming SystemDesk Automation」(🕮『SystemDesk 4.x Guide』)を参照してください。

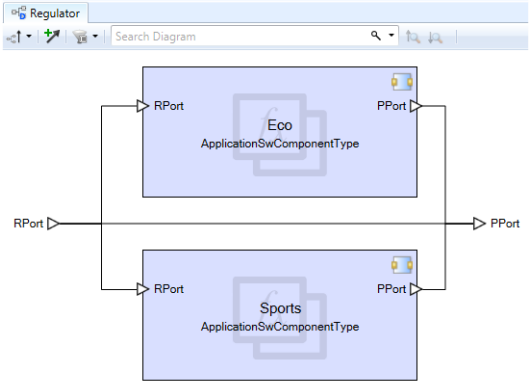
バリアントバインディング

目的

SystemDesk では、多様なバリアントを含むモデルをバインドして、モデル内の多様なバリアントから選択したバリアントを使用することができます。

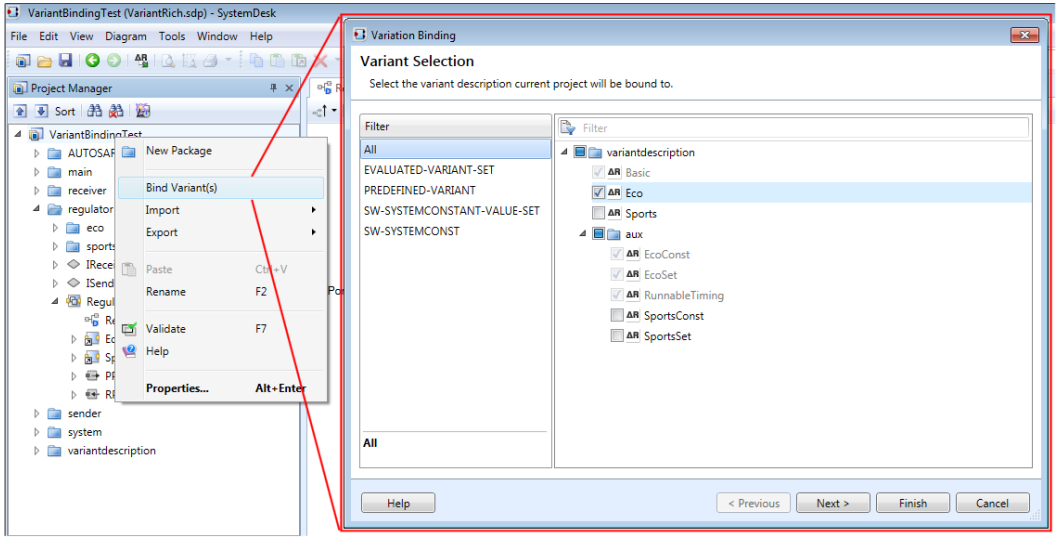
多様なバリアントを含むモデルのインポート

下の図は、SystemDesk にインポートした多様なバリアントを含むモデルのコンポジションダイアグラムを示しています。図に示す Regulator コンポジションは、異なるバリアント用にモデリングされています。



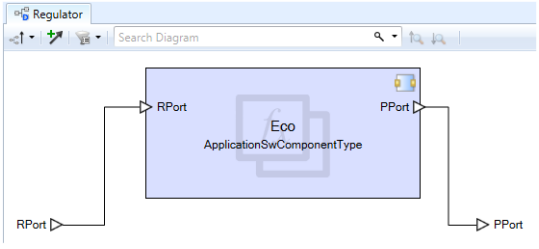
バリアントバインディング

SystemDesk では、インポートした多様なバリアントを含むモデルをバインドすることができます。下の図は、バインドするバリアントの選択を示しています。



結果のバリアントの使用

結果として、SystemDesk は多様なバリアントを含むモデルを、選択したバリアントにバインドします。次の図は、現在 Eco バリアントにバインドされている Regulator コンポジションのコンポジションダイアグラムを示しています。これで SystemDesk で Eco バリアントの使用を開始できるようになります。



補足

SystemDesk でのバリアントバインディングの詳細については、「Variation Binding」(🕮『SystemDesk 4.x Guide』)を参照してください。

SystemDesk 4.5 への移行

SystemDesk 4.5 への移行

SystemDesk 4.5 では、SystemDesk 4.3 および 4.4 の SDP プロジェクトファイルはロード時に自動的に移行されます。

SystemDesk 4.3 または 4.4 の最新のパッチをインストールすることをお勧めします。その後、移行する SDP プロジェクトファイルを保存してから、SystemDesk 4.5 で開きます。

BSW モジュールの移行

このバージョンの SystemDesk では、AUTOSAR に準拠したベーシックソフトウエアモジュールテンプレートを使用してベーシックソフトウエアが記述されます。そのため、SystemDesk 4.3 および 4.4 SDP プロジェクトファイルを SystemDesk 4.5 にロードすると、ベーシックソフトウエアは SystemDesk によって新しい記述に移行されます。V-ECU インプリメンテーションをエクスポートする場合、または移行した SDP ファイルの V-ECU をシミュレートする場合は、各 ECU コンフィギュレーションで*[auto-configure and generate]*を実行し、V-ECU インプリメンテーションをエクスポートするか、必要に応じてそれらをビルドする必要があります。

ただし、dSPACE 仮想検証でサポートされないベーシックソフトウエアを含む SystemDesk プロジェクトを移行する場合は、ベーシックソフトウエアを手作業で移行する必要があります。これを行うには、SystemDesk 4.3 または 4.4 でベーシックソフトウエアを AUTOSAR 形式にエクスポートし、SystemDesk 4.5 で対応するモジュールを作成し、それからエクスポートした AUTOSAR ファイルをインポートします。

TargetLink

項目の一覧

本章の内容

TargetLink 4.1 および TargetLink Data Dictionary 4.1 の新機能

項目の一覧

本章の内容

Simulink または Stateflow でのモデリング

項目の一覧

本章の内容

新しくサポートされる Simulink ブロック

サポートされる Simulink ブロック

TargetLink で、次の Simulink ブロックのサポートが追加されています。

- Signal Conversion ブロック:

  Simulink Signal Conversion ブロックを使用して、たとえばバス信号を再スケーリングすることができます。

- Bus Assignment ブロック:

  Simulink バスを使用してモデリングを単純化する Simulink ブロックです。

関連ドキュメント

- 「Signal Conversion Block」(📖『TargetLink Block and Object Reference』)
- 「Code-Relevant Simulink Blocks」(📖『TargetLink Block and Object Reference』)

カスタムルックアップテーブルの改善

変数が入力信号の次元を継承

`tlscript` コマンドで指定される変数は、入力信号から次元を継承することができます。これにより、たとえば、テーブル検索方法 Local search に、最後のインデックス状態変数を実装することができます。

関連ドキュメント

- 「Basics on Using Custom Look-Up Functions」(📖『TargetLink Preparation and Simulation Guide』)

ベクトルと行列のサポート

カスタムルックアップスクリプト機能で、スカラー信号に加えて、ベクトル／行列入力信号もサポートされます。

関連ドキュメント

- 「廃止された制限事項」(244 ページ)

新しいデモモデル

新しいデモモデル TABLE1D_USR_LOCAL は、カスタムルックアップテーブルによるベクトル／行列入力信号の処理方法および局所検索の実装方法を示します。

関連ドキュメント

- 「ルックアップテーブル、ユーザ記述ルックアップ関数」(📖『TargetLink Demo Models』)
- 「TABLE1D_USR_LOCAL」(📖『TargetLink Demo Models』)

Simulink の簡易モードと IC 構造体のサポート

モデルで指定されていない初期値を決定するために、TargetLink では Simulink の簡易初期化モードをサポートしています。これを選択するには、Simulink の Underspecified initialization detected パラメータを `Simplified` に設定します。

さらに、このモードでは Simulink の初期条件(IC)構造体を使用することで、バス対応のブロック を初期化することができます。このような構造体も、TargetLink 4.1 でサポートされるようになりました。

関連ドキュメント 「Initializing Buses Via Initial Condition Structures」(📖『TargetLink Customization and Optimization Guide』)

Stateflow アクション言語の構文のサポート

Stateflow チャートでのバス信号

TargetLink では、Stateflow 変数に `Simulink.Bus` データ型を使用する Simulink データオブジェクト、および Stateflow チャートの入出力や Stateflow 内部のデータでの Simulink バスをサポートするようになりました。これらの信号にアクセスするために、TargetLink では Stateflow アクション言語の構文をサポートしています。関連するデータオブジェクトは、構造化された Typedef または構造化された DD Variable オブジェクトのどちらかを参照する必要があります。

関連ドキュメント

- 「Basics on the Representation of Buses in the Production Code」(📖『TargetLink Customization and Optimization Guide』)
- 「Basics on the Compatibility of Buses and Predefined Structs」(📖『TargetLink Customization and Optimization Guide』)

コード生成のコア機能

項目の一覧

本章の内容

MISRA-C への準拠

TargetLink の固定小数点ライブラリの改善

TargetLink の固定小数点ライブラリは、MISRA-C に準拠するように複数の改善が行われました。これには次の内容が含まれます。

- 整数型の式の値は、次の場合には、別のデータ型に暗黙的に変換されません。a)それが同じ符号のより幅広い整数型への変換でない場合、b)式が複雑な場合、c)式が定数ではなく、関数の引数である場合、d)式が定数ではなく、戻り式の場合。
- 単項マイナス演算子は、データ型が符号なしのオペランドに適用されません。
- プリプロセス処理の前には、ヌルステートメントのみを単独で一列に配置します。その後にコメントを続けることができます。ただし、ヌルステートメント後の最初の文字は空白文字にする必要があります。
- ゼロ保護により余分な除算は削除されます。ゼロ除算演算を処理するコード保護は、必要に応じてコードジェネレータによって生成されるもので、TargetLink の 固定小数点ライブラリに追加で含まれるわけではありません。

コードジェネレータの改善

MISRA-C に準拠するために、次のようなコードジェネレータの改善が行われています。

- TargetLink は、ルックアップテーブルのコードにビット単位の XOR 演算を生成しなくなりました。
- TargetLink は、Compute Through Overflow(CTO)計算手法を使用した、結果が符号なしの整数になる減算の加算への変換を行わなくなりました。代わりに、減算を量産コードに生成し、可読性の向上と MISRA への準拠を図ります。

TargetLink 4.0 以前	TargetLink 4.1
`U8 = U8 + 255;`	`U8 = U8 - 1;`

- 論理演算の間接参照は、括弧内に記述されるようになりました。

TargetLink 4.0 以前	TargetLink 4.1
`Sa1_OutPort1 = (Int16) (*In2 || *In1);`	`Sa1_OutPort1 = (Int16) ((*In2) || (*In1));`

- *Stateflow 状態 ID の割り当てとビットフィールドへの割り当て*に関するコード変更が行われています。詳細については、「ビットフィールドへの割り当て」(237 ページ)を参照してください。

その他の改善点

MISRA-C に準拠するためのその他の改善点:

- デフォルトで、TargetLink は `Void` typedef ではなく `void` データ型を生成するようになりました。

  「Void typedef の生成なし」(229 ページ)も参照してください。

コード効率性の向上

次元のダウングレード

TargetLink では、スカラー変数によって、アクセスを行列変数の同じインデックス範囲に置き換えることができるようになりました。

さらに、ベクトルと行列変数のエレメントが同じループなし変数式に基づいている場合、これらを置き換えることができます。

TargetLink < 4.1	TargetLink 4.1
`vector2[e-d] = c[e-d]*7;` `g(&vector2[e-d]);`	`Aux_S16_a = c[e-d]*7;` `g(&Aux_S16_a);`

これは、相対オフセットを含むベクトルと行列、およびインデックスアクセス範囲にも適用されます。

関連ドキュメント

- 「Basics on Optimizing Variables」(🕮『TargetLink Customization and Optimization Guide』)
- 「Basics on Eliminating Temporary Variables」(🕮『TargetLink Customization and Optimization Guide』)

コピーの伝播

TargetLink では、より多くのコンテキストで変数を削除できるようになりました。

TargetLink 4.0 以前	TargetLink 4.1
`if (cond) {` `  b = a + 1;` `}` `a = b;`	`if (cond) {` `  a = a + 1;` `}`

TargetLink 4.0 以前	TargetLink 4.1
`if (cond1) {` `  if (cond2) {` `  S = In1;` `  } else {` `  S = X;` `  }` `} else {` `  S = In2;` `}` `X = S;`	`if (cond1) {` `  if (cond2) {` `  X = In1;` `  }` `} else {` `  X = In2;` `}`

関連ドキュメント

■ 「ERASABLE」(🕮『TargetLink Customization and Optimization Guide』)

コードを条件によって実行される分岐の中に移動

TargetLink では、条件によって実行される分岐の中により多くのコードを移動できるようになりました。これは、特にベクトルや行列コードで顕著です。

TargetLink 4.0 以前	TargetLink 4.1
`if (Cond1[0]) {` `  S1[0] = Input1[0];` `  } else {` `  S1[0] = Input2;` `  }` `…` `  if (Cond1[3]) {` `  S1[3] = Input1[3];` `  } else {` `  S1[3] = Input2;` `  }` `  /* Switch: CascSw/S2` `  CascSw/S2: Omitted comparison with constant. */` `  if (Cond2[0]) {` `  Output[0] = (Int16) (-S1[0]);` `  } else {` `  Output[0] = 1;` `  }` `…` `  if (Cond2[3]) {` `  Output[3] = (Int16) (-S1[3]);` `  } else {` `  Output[3] = 4;` ` }	`  /* Switch: CascSw/S2 [0]` `  CascSw/S2: Omitted comparison with constant. */` `  if (Cond2[0]) {` `  if (Cond1[0]) {` `  Output[0] = (Int16) (-Input1[0]);` `  } else {` `  Output[0] = (Int16) (-Input2);` `  }` `  } else {` `  /* # combined # TargetLink outport: CascSw/Output */` `  Output[0] = 1;` `  }` `…` `  /* Switch: CascSw/S2 [3]` `  CascSw/S2: Omitted comparison with constant. */` `  if (Cond2[3]) {` `  if (Cond1[3]) {` `  Output[3] = (Int16) (-Input1[3]);` `  } else {` `  Output[3] = (Int16) (-Input2);` `  }` `  } else {` `  Output[3] = 4;` ` }

関連ドキュメント

■ 「MOVABLE」(🕮『TargetLink Customization and Optimization Guide』)

反復サブシステムでの Assignment ブロックの最適化

反復サブシステムに存在する Assignment ブロックに対して生成される量産コードとその Omit dispensable initializations チェックボックスのオフが改善されました。

TargetLink 4.0 以前

```
for (Sa3_For_Iterator_it = 0; Sa3_For_Iterator_it <=
9; Sa3_For_Iterator_it++) {
   if (Sa3_For_Iterator_it == 0) {
      for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++){
         /* Assignment: … - output initialization */
         Ass_For[Aux_S32] = X_UD_For[Aux_S32];
      }
   }

   /* Assignment: … - output calculation
      # combined # Product: … */
   Ass_For[Sa3_For_Iterator_it] =
                  Op(Sa3_For_Iterator_it);
   for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
      /* Unit delay: … */
      X_UD_For[Aux_S32] = Ass_For[Aux_S32];
    }
}
for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
   /* TargetLink outport: … */
   OutFor[Aux_S32] = Ass_For[Aux_S32];
}
```

TargetLink 4.1

```
for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++){
   /* Assignment: … - output initialization */
   Ass_For[Aux_S32] = X_UD_For[Aux_S32];
}

for (Sa3_For_Iterator_it = 0; Sa3_For_Iterator_it <= 9;
Sa3_For_Iterator_it++) {
   /* Assignment: … - output calculation
      # combined # Product: Direct_Init/For/Product */
   Ass_For[Sa3_For_Iterator_it] =
                  Op(Sa3_For_Iterator_it);
   for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
      /* Unit delay: …_For */
      X_UD_For[Aux_S32] = Ass_For[Aux_S32];
   }
}
```

```
for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
   /* TargetLink outport: … */
   OutFor[Aux_S32] = Ass_For[Aux_S32];
}
```

これは、次のように単純化することができます。

```
for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
   /* TargetLink outport: …
      # combined # Assignment: … - output calculation
      # combined # Product: … */
   OutFor[Aux_S32] = Op(Aux_S32);
}
```

Rte_IRead()ポインタ戻り変数の最適化

TargetLink では、Rte_IRead() および Rte_IWriteRef()に対して不要な戻り変数を生成しなくなりました。

TargetLink 4.0 以前

```
if (cond1) {
   p_DE = Rte_IRead_Run_DE();
   … /* Use p_DE */

}

…

p_DE_a = Rte_IRead_Run_DE();

… /* Use p_DE_a */
```

TargetLink 4.1

```
p_DE = Rte_IRead_Run_DE();
if (cond1) {
   … /* Use p_DE */
}
…
… /* Use p_DE */
```

コードパターンで使用する補助変数に対する代数の簡略化

TargetLink では、複雑な操作のコードパターンで累積に使用する補助変数に対して、さらに最適化を実行できるようになりました。

TargetLink 4.0 以前	TargetLink 4.1
aux = 42; aux += 0; aux -= 0; aux *= 1; aux /= 1;	aux = 42;

コンポーネントベース開発

関数再利用の改善

インクリメンタルサブシステムと参照先モデルでの関数再利用

この TargetLink バージョンでは、関数の再利用は、単純なアトミックサブシステムのみでなく、インクリメンタルコード生成や参照先モデル（複数のインスタンスを含む）用に設定されたサブシステムでも可能です。

関連ドキュメント

- 「Basics on Function Reuse」(📖『TargetLink Customization and Optimization Guide』)
- 「MULTIPLE_INSTANCES_REFMODEL」(📖『TargetLink Demo Models』)

関数再利用のための変数の伝播

TargetLink では、追加のインターフェース変数を生成しなくても、再利用できるシステム定義の先行および後続ブロックの変数を再利用できるようになりました。

関連ドキュメント

- 「Basics on Reusing Variables of Preceding and Subsequent Blocks」(📖『TargetLink Customization and Optimization Guide』)
- 「FUNCTION_REUSE」(📖『TargetLink Demo Models』)

AUTOSAR

項目の一覧

本章の内容

サポートされている AUTOSAR リリース

サポートされている AUTOSAR リリース

次の AUTOSAR リリースがサポートされます。

AUTOSAR リリース	リビジョン
4.2	4.2.1[1)]
4.1	4.1.3 4.1.2 4.1.1
4.0	4.0.3 4.0.2
3.2	3.2.3 3.2.2 3.2.1
3.1	3.1.5 3.1.4 3.1.2 3.1.0
3.0	3.0.7 3.0.6 3.0.4 3.0.2
2.1	2.1.4

[1)] TargetLink 4.1 で新たに追加されたサポート

NvData 通信のサポート

NvData 通信

TargetLink は、AUTOSAR で規定されている暗黙的な NvData 通信をサポートしています。これには Provide ポートと Require ポートのサポートも含まれます。

Port ブロック、Data Store Memory ブロック、およびブロックパラメータを介して、NVRAM へのアクセスをモデリングすることができます。

TargetLink では、特別なモデリングスタイルを使用して、NVRAM へのアクセスを削減することができます。

関連ドキュメント

- 「Modeling NvData Communication」(🕮『TargetLink AUTOSAR Modeling Guide』)
- 「AR_NVDATA_TRANSFORMER」(🕮『TargetLink Demo Models』)

データ変換

エラーロジックのモデリングとシミュレーション

TargetLink では、AUTOSAR で規定されているデータ変換を使用して、関連するエラーロジックのモデリングやシミュレーションを行うことができます。これは、たとえば、セーフティクリティカルなアプリケーションや車載 Ethernet および SOME/IP に対するエンドツーエンド保護に利用することができます。

関連ドキュメント

- 「Basics on Data Transformation」(🕮『TargetLink AUTOSAR Modeling Guide』)
- 「How to Model and Simulate Transformation Error Logic in Sender-Receiver Communication」(🕮『TargetLink AUTOSAR Modeling Guide』)
- 「tlTransformerError」(🕮『TargetLink API Reference』)
- 「AR_NVDATA_TRANSFORMER」(🕮『TargetLink Demo Models』)

ランナブルの Activation Reason

Activation Reason

TargetLink では、AUTOSAR で規定されているランナブルに Activation Reason を使用することができます。

TargetLink では、Runnable オブジェクトのサブツリーで DD ActivationReason オブジェクトを指定することができます。

Activation Reason をモデリングするには、TargetLink InPort ブロックを使用します。

関連ドキュメント

- 「Basics on Activation Reasons」(📖『TargetLink AUTOSAR Modeling Guide』)
- 「How To Model a Runnable's Activation Reasons」(📖『TargetLink AUTOSAR Modeling Guide』)
- 「AR_POSCONTROL」(📖『TargetLink Demo Models』)

ポートにより定義される引数値

ポートにより定義される引数値

TargetLink は、AUTOSAR で規定されている、ポートにより定義される引数値をサポートしています。

モデルでは、ポートにより定義される引数値を表す DD PortDefinedArgument オブジェクトを、TargetLink InPort ブロックで参照します。

量産コードでは、TargetLink はポートにより定義される引数値を仮引数として、ランナブルファンクションのシグネチャ内に生成します。

関連ドキュメント

- 「Basics on Port-Defined Argument Values」(📖『TargetLink AUTOSAR Modeling Guide』)
- 「How to Model Port-Defined Argument Values」(📖『TargetLink AUTOSAR Modeling Guide』)

その他の AUTOSAR 機能

NvData 通信用の Rte_IWriteRef のサポート

TargetLink は、NvData 通信用の API 関数 `Rte_IWriteRef` をサポートしています。

ComSpec の変換

TargetLink では、データ変換および `TRANSFORMER_HARD_ERROR_EVENT` のための通信仕様をインポートおよびエクスポートすることができます。

パラメータプロトタイプの ImplementationPolicy	TargetLink では、パラメータデータプロトタイプのインプリメンテーションポリシーをインポートおよびエクスポートすることができます。 データは、パラメータプロトタイプの DD ImplementationPolicy プロパティに格納されます。
StepSize プロパティ	TargetLink では、計測および適合ツールに指定されたステップサイズをインポートおよびエクスポートすることができます。 データは、プリミティブアプリケーションデータ型またはデータプロトタイプの StepSize プロパティに格納されます。
ConstrLevel プロパティ	TargetLink では、計測および適合ツールに指定された制約レベルをインポートおよびエクスポートすることができます。 データは、プリミティブアプリケーションデータ型またはデータプロトタイプの ConstrLevel プロパティに格納されます。 インプリメンテーションデータ型の場合は、Constraints サブツリーに格納されます。
LINEAR カテゴリのスケーリング	TargetLink では、カテゴリが `LINEAR` に設定されているスケーリングのデフォルト値をインポートおよびエクスポートすることができます。 データは、DD Scaling オブジェクトの DefaultValue プロパティに格納されます。
DataReceiverComSpec オブジェクトのデータステータス	TargetLink は、DD DataReceiverComSpec オブジェクトの HandleDataStatus プロパティをサポートしています。

ターゲットシミュレーション（PIL）

項目の一覧

本章の内容

ターゲットシミュレーションモジュールの変更

新規および廃止されたコンパイラバージョン

次の表は、TargetLink 4.1 でサポートされるコンパイラバージョンを示しています。新規と変更なしの列を参照してください。サポートが終了したコンパイラバージョンは、廃止の列に示しています。

ターゲット	コンパイラ	新規	変更なし	廃止
ARM CortexM3	Keil	5.1	—	—
C16x	TASKING	—	8.6	8.7
HCS12	Cosmic	—	—	4.8
	Metrowerk	—	—	5.1
M32R	Gaio	—	—	11、9
	Renesas	—	—	5.1
MC56F83	Metrowerk	—	—	8.3
MPC55xx	Diab	—	—	5.9
	GreenHill	—	—	2013
	GNU	—	—	4.1
	Metrowerk	—	—	2.8
MPC55xxVLE	Diab	—	—	5.9
	GreenHill	—	—	2013
	Metrowerk	—	—	2.8
MPC57xxVLE	Diab	5.9	—	—
	GreenHill	2014	—	—
MPC560xVLE	Diab	—	5.9	—
	GreenHill	2014	2012	2013
RH850	GreenHill	2014	—	—
S12X[1)]	Cosmic	—	4.8	—
	Metrowerk	—	5.1	—
SH2	Renesas	—	9.3	9.4
SH2A-FPU	Renesas	—	9.4	—
TriCore17xx	TASKING	5.0	3.2	4.3
	GNU	—	—	3.4
TriCore2xx	TASKING	5.0	—	—
	GNU	4.6	—	—

ターゲット	コンパイラ	新規	変更なし	廃止
V850	GreenHill	2014	—	2013
	NEC	—	—	3.40
XC22xx	TASKING	—	3.0	—

[1] Freescale S12XEVB/S12XEVB_USB は、新しい Freescale EVB9S12XEP100 に置き換えられます。

TargetLink でサポートされている評価用ボードの詳細については、📖『TargetLink Evaluation Board Hardware Reference』を参照してください。

有効なソフトウエア保守サービス（SMS）契約に含まれる PIL サポート対象の組み合わせについては、TargetLink 製品サポートセンターにある dSPACE の TargetLink PIL Support Web サイトを参照してください。

廃止されたボード

サポート終了、販売終了　下記のボードの TargetLink でのサポートと dSPACE による販売は終了しました。

- Freescale 56F83xx
- Freescale HCS12
- Freescale PowerPC MPC5500
- Freescale PowerPC MPC5500VLE
- NEC V850ES
- Renesas M32R

TargetLink 4.1 でサポートされていない評価ボードを使用する場合は、dSPACE サポートにお問い合わせください。

サポート継続、販売終了　下記のボードは TargetLink で引き続きサポートされます。ただし、dSPACE による販売は終了しています。

- Freescale S12XEVB/S12XEVB_USB は、新しい Freescale EVB9S12XEP100 に置き換えられます。

TSM 拡張用フォルダ

API によるフォルダの指定

TargetLink Preferences Editor のみでなく、API でも TSM 拡張用フォルダを指定できるようになりました。たとえば、次のように指定します。

```
TlTsmManager.exe -SetTsmExtensionFolder -Folder:C:\exp
```

関連ドキュメント

- 「How to Clone Target/Compiler Combinations to Outside the TargetLink Installation」(📖『TargetLink Customization and Optimization Guide』)(特に「前提条件」)

Data Dictionary とデータ管理

項目の一覧

本章の内容

Data Dictionary の改善点

さまざまな DD 表示用にあらかじめ設定されたフィルタ規則セット

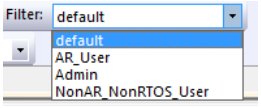
TargetLink には、あらかじめ設定されたされた新しいフィルタ規則セットが付属しており、DD ツリーの特定のオブジェクトやプロパティを非表示にすることができます。これにより、関連するデータをより簡単に表示することができます。フィルタ規則セットは、一般的なユーザグループ向けに設計されています。

使用可能なあらかじめ設定されたフィルタ規則セットは以下のとおりです。

- Admin - 管理者向けのフィルタ規則セット
- AR_User - AUTOSAR ユーザ向けのフィルタ規則セット
- NonAR_NonRTOS_User - AUTOSAR も RTOS も使用しないユーザ向けのフィルタ規則セット

フィルタ規則セットの詳細については、「DD_FILTER」(📖『TargetLink Demo Models』)を参照してください。

フィルタ規則セットは、Data Dictionary Manager のツールバーの[Filter]リストで変更することができます。



関連ドキュメント

- 「Basics on Filter Rule Sets for the Data Model」(🕮『TargetLink Data Dictionary 基本コンセプトガイド』)
- 「How to Create Filter Rule Sets via DD Files」(🕮『TargetLink Data Dictionary 基本コンセプトガイド』)

Object Comparison Navigator のフィルタ表示

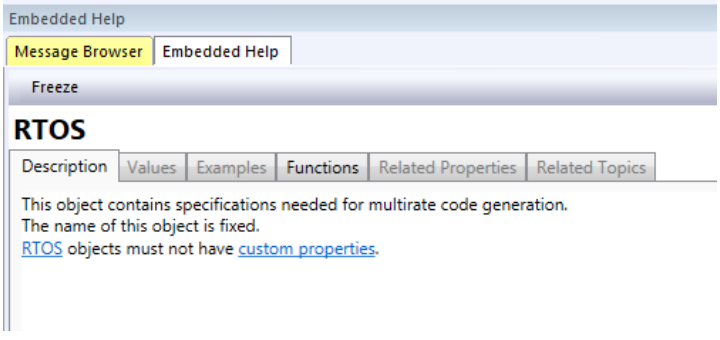
Object Comparison Navigator の[Filter]リストを使用して、関心のない特定のオブジェクトやプロパティを非表示にすることができます。これにより、多数の DD オブジェクトをより簡単に比較することができます。あらかじめ設定されたフィルタ規則セットから選択するか、独自のフィルタを作成することができます。

関連ドキュメント

- 「Basics on Filter Rule Sets for the Data Model」(🕮『TargetLink Data Dictionary 基本コンセプトガイド』)
- 「Data Dictionary オブジェクトの比較およびマージ」(🕮『TargetLink Data Dictionary 基本コンセプトガイド』)

メッセージアラート

メッセージブラウザペインが開いても、アクティブな[Embedded Help]ペインがあるために表示されない場合、Message Browser のタブの色が変わり、新しいメッセージがあることが示されます。



- 新しい情報メッセージ - 青色のタブ
- 新しい 警告メッセージ - 黄色のタブ
- 新しい エラーメッセージ - 赤色のタブ

関連ドキュメント

- 「Overview of the User Interface」(📖『TargetLink Data Dictionary 基本コンセプトガイド』)

新しい DD MATLAB API 関数

新しい DD API 関数

TargetLink で、以下の DD MATLAB API 関数を新たに利用することができます。詳細については、📖『TargetLink Data Dictionary Reference』を参照してください。

GetAutosarVersion

```
[version, errorCode] = dsdd('GetAutosarVersion'[,<DD_identifier>]);
```

指定した DD の/Pool/Autosar/Config.AutosarVersion プロパティで指定した AUTOSAR バージョンを取得します。

GetRenameBaseType

```
[version, errorCode] = dsdd('GetAutosarVersion'[,<DD_identifier>]);
```

基本データ型の名前変更規則を指定する Typedef オブジェクトを取得します。

TargetLink Main Dialog ブロックでの DD CodegenOptions オブジェクトの参照

DD CodegenOptions オプションの参照

処理を一元化し、オプション指定の整合性を簡単に維持するために、TargetLink 4.1 では DD CodegenOptions オブジェクトを TargetLink Main Dialog ブロックで直接参照することができます。

関連ドキュメント 「Basics on Configuring the Code Generator for Production Code Generation」(📖『TargetLink Customization and Optimization Guide』)

Code Generator オプション

新しい Code Generator オプション

新しい Code Generator オプションの概要

TargetLink 4.1 では、次の新しい Code Generator オプションを使用することができます。

- AvoidNestedVariablePropagationPointerAccess

  これらの再利用構造体が、変数の伝播が指定されている変数にアクセスする場合、再利用構造体に追加のポインタを生成します。詳細については、「AvoidNestedVariablePropagationPointerAccess」(📖『TargetLink Block and Object Reference』)を参照してください。

- ReportFailedFunctionReuseVariablePropagation

  変数の伝播中に関数を再利用するシステムで発生した問題を示すオプションのレポートを有効にします。詳細については、「ReportFailedFunctionReuseVariablePropagation」(📖『TargetLink Block and Object Reference』)を参照してください。

- ReplaceUnrolledVectorsAndMatricesByScalar

  アンロールされたベクトルおよび行列アクセスのスカラーとの置換を制御します。詳細については、「ReplaceUnrolledVectorsAndMatricesByScalars」(📖『TargetLink Block and Object Reference』)を参照してください。

すべての Code Generator オプションの詳細については、「Alphabetical List of Code Generator Options」(📖『TargetLink Block and Object Reference』)を参照してください。

Code Generator オプションの移行上の注意点

移行に関しては、以下の点に注意してください。

- 削除された Code Generator オプション
- 変更された Code Generator オプション
- 推奨される互換性設定
- 変更されたデフォルトの基礎

詳細については、「Code Generator オプションに関する移行上の注意点」(224 ページ)を参照してください。

ツールチェーンの統合

項目の一覧

本章の内容

Functional Mock-up Unit のエクスポート

Functional Mock-up Unit

TargetLink では、TargetLink サブシステム用の Functional Mock-up Unit（FMU）を生成することができます。これらの FMU は、FMI 互換のシミュレーション環境で TargetLink コードを実行するために、FMI 2.0 規格をベースにしています。

これには、VEOS、SCALEXIO、および幅広いサードパーティ製の FMI 互換ツールが含まれます（https://www.fmi-standard.org/tools）。

関連ドキュメント

- 「Definition of the FMI Standard and FMUs」（📖『TargetLink Interoperation and Exchange Guide』）
- 「Basics on Exporting FMUs from TargetLink」（📖『TargetLink Interoperation and Exchange Guide』）

Data Dictionary 内の要件情報

要件情報を DD オブジェクトとして格納

TargetLink では、要件情報を Data Dictionary に RequirementInfo オブジェクトとして格納することができます。これらのオブジェクトは、要求管理システムへのプロキシとして機能します。

モデルでは、これらのオブジェクトを TargetLink ブロックで、および Stateflow オブジェクトから参照することができます。これは、生成される量産コードと生成されるドキュメントに要件情報をコメントとして追加するように TargetLink に指示します。

要件情報のブロックデータをプログラム処理するには、API 関数 `tlRequirementInfo()` を使用します。この関数は、このデータに対する `tl_set()` または `tl_get()` の代わりに使用されます。

関連ドキュメント

- 「Basics on Requirement Information in the Generated Code」 (🕮『TargetLink Interoperation and Exchange Guide』)
- 「Basics on Using DD Based Requirement Information」 (🕮『TargetLink Interoperation and Exchange Guide』)
- 「tlRequirementInfo」 (🕮『TargetLink API Reference』)

その他

項目の一覧

本章の内容

一般的な機能拡張および変更

TargetLink のコンテキストメニュー

TargetLink ブロックのコンテキストメニューに、次の 2 つのオプションが新しく追加されました。

- Create reference (incr. subsystem to ref. model)
- Disable reference (ref. model to incr. subsystem)

以前は、これらのオプションは TargetLink Main ブロックのダイアログでのみ使用可能でした。これらのオプションを使用して、インクリメンタルコード生成用に設定されているサブシステムを参照先モデルと交換、またはその逆を実行することができます。

関連ドキュメント

- 「Common TargetLink Context Menu Options」 (🕮『TargetLink Block and Object Reference』)

シミュレーションパフォーマンスの向上

TargetLink のシミュレーションパフォーマンスが向上しました。以下のタイプのモデルは、MIL シミュレーションモードでのシミュレーションパフォーマンスが向上しました。

- 多数の scaling-invariant サブシステムを含むモデル
- 多数のワークスペース変数を使用するモデル

TargetLink モデルのシミュレーションパフォーマンスを最大化するには、次のいずれかの方法を使用してシミュレーションを開始することをお勧めします。

- Simlunk の独自の `sim()` 関数ではなく **`tl_sim(model, parameters)`** (📖『TargetLink API Reference』) API 関数を使用
- TargetLink Main Dialog ブロックを使用
- TargetLink プロットダイアログを使用

関連ドキュメント

- なし

ファンクションシステムシグネチャの同期の改善

ファンクションシステムシグネチャの指定および同期が以下のように改善されています。

- TargetLink Function ブロックで、DD Function オブジェクトに加えて、DD Signature オブジェクトをリンクできるようになりました。DD Signature オブジェクトには、インターフェース指定(ポートデータ)が含まれます。
- DD Signature オブジェクトで指定されたインターフェース(ポートデータ)と Function システムのモデルインターフェースの間での整合性チェックを実行することができます(Check ports)。
- モデルのインターフェースを DD Signature オブジェクトの指定で更新／同期することができます(DD to Model)。
- `/Pool/ModelDesign/Config/`オブジェクトツリー内の SyncSystemSignature オブジェクトは、文字列プロパティを含まなくなりました。代わりに、`Boolean` や `Enum` のような型付きデータ型を含みます。これにより、Data Dictionary Manager でこのようなタイプを設定しやすくなります。
- DD Signature オブジェクトの指定がないが、モデルインターフェースデータを Data Dictionary に転送する場合は、ポートデータを Data Dictionary に同期することもできます(Model to DD)。
- HTML レポートは、ポートをチェックまたは同期するたびに生成されます。レポートには、DD Signature オブジェクトの指定と Function システムのインターフェースの相違に関する詳細情報が含まれます。

関連ドキュメント

- 「Centrally Specifying and Synchronizing Function System Signatures」(📖『TargetLink Customization and Optimization Guide』)

MEX コンパイラと SIL コンパイラの個別の設定

量産コード DLL 用 SIL コンパイラを、シミュレーション S-function 用 MEX コンパイラとは別に設定できるようになりました。このため、量産コードのコンパイルと SIL デバッグに無料の MSVC Express Edition を使用することができます。

関連ドキュメント

- 「How to Set or Change MEX and SIL Compilers」(📖『TargetLink Preparation and Simulation Guide』)
- 「How to Debug in SIL Simulation Mode」(📖『TargetLink Preparation and Simulation Guide』)
- 「tlProductionCodeSILCompiler」(📖『TargetLink API Reference』)

TargetLink デモ

新しいデモ

TargetLink 4.1 には次の新しいデモが付属しています。

Ar_nvdata_transformer　この新しいデモは次の 2 つの機能を示します。

- NVData インターフェースを介した不揮発性 RAM への読み取り／書き込みアクセス
- AUTOSAR のトランスフォーマの使用：セーフティクリティカルなアプリケーションに対するエンドツーエンド通信保護をモデリングするためなど

DD_filter　このデモでは、必要な仕様を含む DD ファイルに基づいた XML フィルタ規則セットを作成する方法を示します。デモには、`tl_example_CreateDDFilterBasedOnDDFile` スクリプトと、一般的なフィルタ用途での仕様を含む 3 つの汎用 DD ファイルが含まれます。このスクリプトは、XML フィルタ規則セットを生成します。

このデモにはモデルは含まれません。

関連ドキュメント

- 「DD_FILTER」(📖『TargetLink Demo Models』)
- 「Basics on Filter Rule Sets for the Data Model」(📖『TargetLink Data Dictionary 基本コンセプトガイド』)

DD_ML_API　このデモでは、DD MATLAB API の 2 つの使用方法例を示します。

- `simple.m` スクリプトでは、dsdd コマンドを使用して、DD 変数オブジェクトとユーザ定義の汎用 DD オブジェクトを作成します。
- `findCalVariables.m` スクリプトでは、DD ファイル内で STATIC_CAL クラスを持つすべての変数オブジェクトを検索する方法を示します。

関連ドキュメント

- 「DD_ML_API」(🕮『TargetLink Demo Models』)

DD_ML_ImportExport　このデモには、XLS および XML ファイルから Data Dictionary にオブジェクトをインポートする方法を示すさまざまな M スクリプトが含まれています。

関連ドキュメント

- 「DD_ML_IMPORTEXPORT」(🕮『TargetLink Demo Models』)

Function_reuse　このデモでは、マスクパラメータを定義して、インスタンス固有のパラメータ初期値を含むサブシステムに関数再利用を適用する機能を示します。

関連ドキュメント

- 「FUNCTION_REUSE」(🕮『TargetLink Demo Models』)
- 「Basics on Function Reuse」(🕮『TargetLink Customization and Optimization Guide』)

Multiple_instances_refmodel　このデモでは、モデル引数を定義して、インスタンス固有のパラメータ初期値を含む参照先モデルに関数再利用を適用する機能を示します。

関連ドキュメント

- 「MULTIPLE_INSTANCES_REFMODEL」(🕮『TargetLink Demo Models』)
- 「Basics on Function Reuse」(🕮『TargetLink Customization and Optimization Guide』)

Table1d_usr_local　このデモでは、カスタムルックアップ関数により、ローカル検索アルゴリズムを使用する TargetLink ルックアップテーブルのコードを、非スカラー入力に置き換える方法を示します。

関連ドキュメント

- 「TABLE1D_USR_LOCAL」(🕮『TargetLink Demo Models』)
- 「Basics on Using Custom Look-Up Functions」(🕮『TargetLink Preparation and Simulation Guide』)

Variable_vector_width　このデモモデルでは、幅バリアントを持つベクトル変数の使用方法を示します。ベクトル幅に対してプリプロセッサマクロを使用すると、同じモデルと生成された量産コードをすべての幅に対して使用することができます。サブシステム全体のコードは、コードのコンパイル時に幅を変更できるようになります。

関連ドキュメント

- 「VARIABLE_VECTOR_WIDTH」(🕮『TargetLink Demo Models』)
- 「Introduction to Variable Vector Widths」(🕮『TargetLink Customization and Optimization Guide』)

改良されたデモ

次のデモには新機能のデモが含まれます。

Ar_poscontrol　このデモでは、さまざまな RTE イベントをトリガする Stateflow チャートを使用して、Activation Reason をシミュレートします。信号は、DD で指定された値に合わせて自動的に調整されます。

関連ドキュメント

- 「AR_POSCONTROL」(🕮『TargetLink Demo Models』)
- 「Basics on Activation Reasons」(🕮『TargetLink AUTOSAR Modeling Guide』)

Poscontrol　このデモでは、TargetLink ファンクションブロックと、Data Dictionary 内の DD Function および DD Signature オブジェクトの接続を示し、ファンクションとモデリングされるサブシステムとの間のインターフェースの整合性を確認します。

関連ドキュメント

- 「POSCONTROL」(🕮『TargetLink Demo Models』)
- 「Centrally Specifying and Synchronizing Function System Signatures」(🕮『TargetLink Customization and Optimization Guide』)

API 関数とフック関数

項目の一覧

本章の内容

新しい API 関数

FMU のエクスポート

`tl_generate_fmu(propertyName, propertyValue, ...)` API 関数により、Functional Mock-up Units（FMU）をエクスポートして、FMI 互換のツールで使用することができます。

関連ドキュメント

- 「Definition of the FMI Standard and FMUs」（📖『TargetLink Interoperation and Exchange Guide』）
- 「Basics on Exporting FMUs from TargetLink」（📖『TargetLink Interoperation and Exchange Guide』）
- 「How to Generate an FMU to use in an FMI-Compliant Tool」（📖『TargetLink Interoperation and Exchange Guide』）
- 「TargetLink FMU Export」（📖『TargetLink Tool and Utility Reference』）

SIL コンパイラの切り替え

`tlProductionCodeSILCompiler` API 関数により、SIL コンパイラを設定または変更することができます。

関連ドキュメント

- 「How to Set or Change MEX and SIL Compilers」（📖『TargetLink Preparation and Simulation Guide』）

ブロックでの要件情報の設定

`tlRequirementInfo` API 関数により、TargetLink ブロックと Stateflow オブジェクトで DD ベースの要件情報を管理することができます。このデータには、`tl_set()` または `tl_get()` の代わりに `tlRequirementInfo` が使用されます。

関連ドキュメント

- 「Basics on Using DD Based Requirement Information」（📖『TargetLink Interoperation and Exchange Guide』）

変換エラーロジックのシミュレーションの準備

`tlTransformerError` API 関数により、シミュレーション用に変換エラーロジックを含む AUTOSAR モデルを準備することができます。

関連ドキュメント

- 「Basics on Data Transformation」（📖『TargetLink AUTOSAR Modeling Guide』）
- 「How to Model and Simulate Transformation Error Logic in Sender-Receiver Communication」（📖『TargetLink AUTOSAR Modeling Guide』）

新しいフック関数

システムシグネチャの生成と同期

TargetLink に、DD からのサブシステム生成をカスタマイズするための次の新しいフック関数が追加されました。

- **`tl_pre_add_ddsignatureport_hook`** このフック関数は、新しい DD SignaturePort オブジェクトが指定した DD Signature オブジェクトに追加される前に呼び出されます。
- **`tl_post_add_ddsignatureport_hook`** このフック関数は、新しい DD SignaturePort オブジェクトが指定した DD Signature オブジェクトに追加された後に呼び出されます。
- **`tl_pre_sync_systemsignatureport_hook`** このフック関数は、既存の Port ブロックを対応する DD Signature オブジェクトと同期する前に呼び出されます。
- **`tl_post_sync_systemsignatureport_hook`** このフック関数は、既存の Port ブロックを対応する DD Signature オブジェクトと同期した後に呼び出されます。

関連ドキュメント 「Basics on Using Hook Functions」(🕮『TargetLink Customization and Optimization Guide』)

「Centrally Specifying and Synchronizing Function System Signatures」(🕮『TargetLink Customization and Optimization Guide』)

DD Variable オブジェクトを介したバスの初期化

TargetLink では、次のカスタマイズファイルを使用して、Simulink IC 構造体を介してバスの初期化をカスタマイズすることができます。

- **`tlGetBusStructMapping`** このカスタマイズファイルは、バス構造体のマッピング(DD Variable オブジェクトを Simulink.Bus オブジェクトに)を取得します。

このカスタマイズファイルを使用して、DD ベースの構造体変数を Simulink.Bus オブジェクトにマッピングすることができます。

関連ドキュメント 「How to Manually Create a Mapping Between a DD Variable and a Simulink Bus」(🕮『TargetLink Customization and Optimization Guide』)

TargetLink 4.1 および TargetLink Data Dictionary 4.1 への移行

アップグレードプロセス

新しい TargetLink バージョンにアップグレードするには、以下を調整する必要があります。

- Data Dictionary
- モデル
- スクリプトとフック関数

TargetLink 4.0 より前のバージョンからライブラリ／モデルを移行するには、その間の TargetLink バージョンの移行手順も実行する必要があります。詳細については、DVD で提供されている以前の『TargetLink Migration Guide』を参照してください。

アップグレードを手作業で実行するには、`tlUpgrade` API 関数を使用します。詳細については、「API を使用してライブラリとモデルを手作業でアップグレードする方法」(223 ページ)を参照してください。

次の情報すべてをよくお読みいただき、適宜ツールチェーンを変更してください。

項目の一覧

本章の内容

モデル、ライブラリ、Data Dictionary のアップグレード

項目の一覧

本章の内容

TargetLink 4.1 への移行

TargetLink 2.x からの間接アップグレード

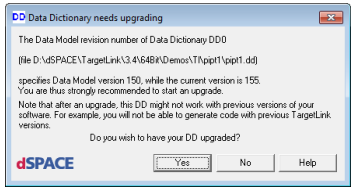
TargetLink 3.1 より前の TargetLink バージョンのライブラリ、モデル、DD ファイルを直接アップグレードすることはできません。

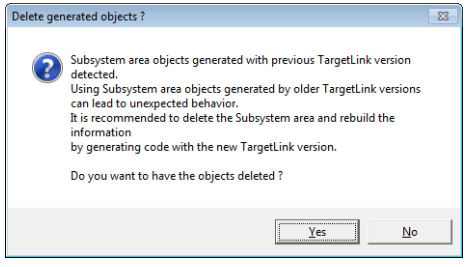
ただし、間接アップグレードを実行することができます。最初に、古いライブラリ、モデル、DD ファイルを TargetLink 3.5 に移行します。その後、それを TargetLink 4.1 にアップグレードすることができます。

TargetLink 3.1 以降からの直接アップグレード

TargetLink 4.1 では、TargetLink 3.1 以降で作成したモデル、ライブラリ、Data Dictionary が自動的にアップグレードされます。

Data Dictionary ファイルのアップグレード時には、次のダイアログが表示されます。





ユーザによるインタラクティブな操作　次の場合は、ユーザによるインタラクティブな操作が必要になります。

- 「TargetLink 向けに準備されていない古いライブラリ」(220 ページ)
- 「TargetLink 32 ビットバージョンから TargetLink 64 ビットバージョンへのアップグレード」(220 ページ)
- 「インクルードされる部分的な DD ファイルを含む DD ファイル」(220 ページ)
- 「API を使用してライブラリとモデルを手作業でアップグレードする方法」(223 ページ)

TargetLink 向けに準備されていない古いライブラリ

`tl_prepare_system(propertyName, propertyValue, ...)` (📖『TargetLink API Reference』) API 関数を使用して準備されていない TargetLink 3.x または 4.0 で作成したライブラリは、TargetLink 4.1 で自動的にアップグレードすることはできません。

解決策

1. 以前の TargetLink バージョンでライブラリを開き、`tl_prepare_system(propertyName, propertyValue, ...)` (📖『TargetLink API Reference』)を使用してアップグレード用に準備します。
2. ライブラリを保存します。
3. TargetLink 4.1 でライブラリを開きます。

関連ドキュメント

- 「TargetLink ユーザライブラリをアップグレード可能にする方法」(📖『TargetLink Orientation and Overview Guide』)

TargetLink 32 ビットバージョンから TargetLink 64 ビットバージョンへのアップグレード

32 ビットバージョンの TargetLink でビルドしたカスタムコード S-function は、64 ビットバージョンの TargetLink では使用することができません。これは、逆の場合も同様です。

解決策　`tlUpgrade('Model',<MyModel>,'CheckModel','FixIssues')` (「tlUpgrade(propertyName, propertyValue, ...)」(📖『TargetLink API Reference』)を参照) API 関数を使用して、すべてのカスタムコード S-function の再ビルドを行います。

インクルードされる部分的な DD ファイルを含む DD ファイル

インクルードされる部分的な DD ファイルを含む DD ファイルをアップグレードする場合は、「インクルード DD ファイルのある Data Dictionary をアップグレードする方法」(221 ページ)を参照してください。

インクルード DD ファイルのある Data Dictionary をアップグレードする方法

目的

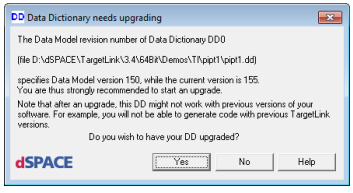
アップグレードしていない古い Data Dictionary ファイルを含む TargetLink モデルを開く場合は、Data Dictionary ファイルをアップグレードする必要があります。

操作手順

インクルード DD ファイルのある Data Dictionary をアップグレードするには

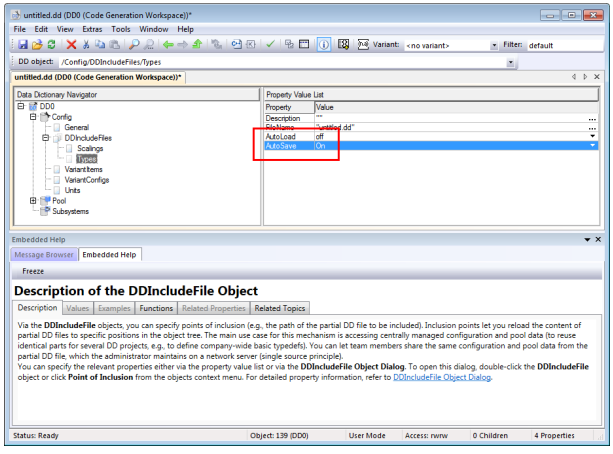
1 モデルおよび参照する TargetLink Data Dictionary を開くか、MATLAB コマンドウインドウで `dsdd('Open',<DDFile>)`と入力します。

古いバージョンの DD が使用されている場合は、[Data Dictionary needs upgrading]ダイアログが自動的に開きます。



2 アップグレードダイアログで[No]を選択します。

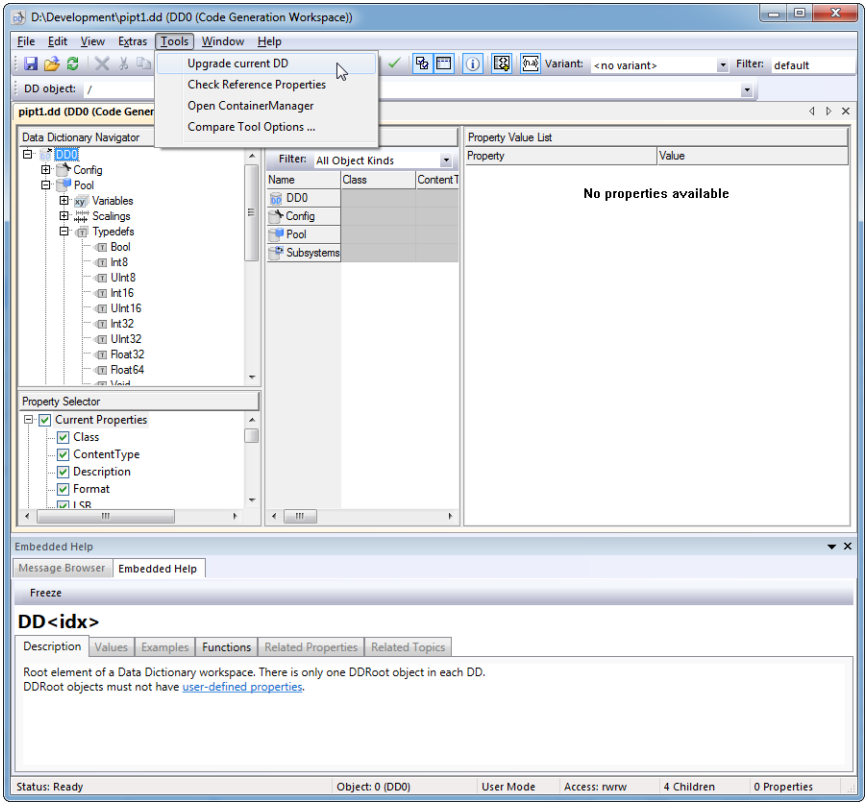
3 /Config/DDIncludeFiles で、下の画面のように各インクルード DD ファイルの AutoLoad および AutoSave プロパティを設定します。



これにより、Data Dictionary とインクルード DD ファイルのアップグレード後、Data Dictionary を保存するときにアップグレードしたインクルード DD ファイルが保存されます。Object Explorer を使用して、複数のインクルード DD ファイルのこれらのプロパティを設定することができます。

[Point of Inclusion]ダイアログを使用してインクルード DD ファイルのプロパティを設定することもできます。

4 DD Manager で[Tools] - [Upgrade Current DD]を使用して Data Dictionary アップグレード（インクルード DD ファイルを含む）を開始するか、MATLAB コマンドウインドウで `dsdd('Upgrade')` と入力します。



5 （関連 DD ファイルへの書き込みを許可して）Data Dictionary を保存します。これで、DD ファイルとインクルードされる部分的な DD ファイルのアップグレードは完了です。

結果

DD ファイルを再度開いたときには、DD ファイルとインクルードされている部分的な DD ファイルが最新であるため、アップグレードダイアログは開きません。ファイルを正常にアップグレードした後で、インクルード DD ファイルを古い設定に戻すことが必要となる場合もあります。

API を使用してライブラリとモデルを手作業でアップグレードする方法

目的

`tlUpgrade(propertyName, propertyValue, ...)` (📖『TargetLink API Reference』) API 関数を使用してライブラリとモデルを手作業でアップグレードし、後でそれを保存して、たとえば複数のユーザとのツールチェーンシナリオでライブラリとモデルの集中アップグレードを準備することができます。

モデルおよびライブラリをアップグレードする場合には、他のライブラリを参照しない（含まれているブロック／サブシステムに他のライブラリへのリンクがない）モデルまたはライブラリを最初にアップグレードします。一番下のライブラリから開始して、順次上のライブラリをアップグレードします。

準備されていないライブラリの詳細については、「*TargetLink 向けに準備されていない古いライブラリ*」(「TargetLink 4.1 への移行」(219 ページ))を参照してください。

操作手順

API を使用してライブラリとモデルを手作業でアップグレードするには

1. MATLAB コマンドウインドウで `dsdd_manage_project('Open','<name>.dd')` と入力して、既にアップグレードされている必要な DD プロジェクトファイルをロードします（DD プロジェクトファイルをアップグレードするには、`dsdd('Upgrade'[,<DD_Identifier>])` コマンドを使用します。`Upgrade` (📖『TargetLink Data Dictionary Reference』)を参照してください）。
2. `tlUpgrade('Model', '<Model|Library>.mdl', 'CheckModel','FixIssues')` と入力してモデルまたはライブラリをアップグレードします。
3. アップグレードしたモデルまたはライブラリファイル（`Library.mdl` など）を保存します。
4. 他のすべてのモデルまたはライブラリに対して、手順 2 と 3 を繰り返します。

結果

モデルとライブラリがアップグレードされます。

Code Generator オプション

Code Generator オプションに関する移行上の注意点

削除された Code Generator オプション

MPC5xx 専用（TOM 専用）の Code Generator オプション EnableLogicalOperationOptimisation は、TargetLink から削除されました。

削除されたオプション	代わりのオプション	互換設定値
EnableLogicalOperationOptimisation	なし	なし

TargetLink 4.1 では、このオプションは MPC5xx のサポートとともに削除されました。

変更された Code Generator オプション

次の Code Generator オプションは、TargetLink 4.1 で変更されています。

- なし

推奨される互換性設定

新しい TargetLink 4.1 Code Generator オプションを次のように設定して、可能なかぎり下位互換性を確保してください。

- なし

変更されたデフォルトの基礎

Code Generator オプションの設定は、モデルとともに保存されます（モデルベースのオプションの保存）。また、DDCodegenOptions オブジェクトにユーザ定義の一連の Code Generator オプションを保存することができます（DD ベースのオプションの保存）。DD CodegenOptions オブジェクトのみを使用して、モデルベースのオプション設定を上書きすることができます。

モデルベースのオプションの値が以前のデフォルト値と等しい場合は、アップグレードの際に新しいデフォルト値に自動的に変更されます。DD ベースのオプションの値が以前のデフォルト値と等しい場合は、アップグレードの際に新しいデフォルト値に変更されず、以前のデフォルト値が維持されます。

オプションの値 = 以前のデフォルト値 Code Generator オプションが TargetLink の以前のバージョンのデフォルト値と等しく、TargetLink の新しいバージョンが変更されたデフォルト値を使用する場合は、以下の点に注意してください。

- モデルベースのオプション:

  以前のデフォルト値を維持する場合は、それらを手作業でリセットする必要があります。

- DD ベースのオプション:

  新しいデフォルト値を使用する場合は、それらを手作業で調整する必要があります。

3 つの任意のオプション値 9、11、13 における TargetLink アップグレード(TL_{Old} から TL_{New} へ)の影響について、次の表に例を示します。表では、2 つの基本的な移行シナリオについて説明します。

- シナリオ#1:新しいデフォルト値 = 以前のデフォルト値

  Code Generator オプションのデフォルト値は新しい TargetLink バージョンで変更されず、デフォルト値は 9 のままとなります。

  オプション値はいずれも変更されません。

- シナリオ#2:新しいデフォルト値 ≠ 以前のデフォルト値

  Code Generator オプションのデフォルト値は新しい TargetLink バージョンで変更され、デフォルト値は 11 に変更されます。

オプションの保存	**オプション値(TL_{Old})**	**オプション値(≤ TL_{New})**	
	デフォルト = 9	**デフォルト = 9 (シナリオ#1)**	**デフォルト = 11 (シナリオ#2)**
モデルベース	9[1]	9[1]	11[2]
	11	11	11[1]
	13	13	13
DD ベース	9	9	9[3]
	11	11	11
	13	13	13

[1] オプション値は、デフォルト値と等しいためこのモデルで保存されません。
[2] 手作業によるリセットが必要な場合があります。
[3] 手作業による調整が必要な場合があります。

オプションの値 = 新しいデフォルト値 Code Generator オプションが TargetLink の以前のバージョンのデフォルト値と等しくない(A)が、TargetLink の新しいバージョンのデフォルト値と等しい場合(B)、TargetLink では、新しいバージョンで意図的にデフォルト値が指定されたと見なされます(C)。TargetLink の次のバージョンでデフォルト値が再び変更された場合も同様です(C)。

$TL_A \Rightarrow TL_B \Rightarrow TL_C$ のアップグレードと $TL_A \Rightarrow TL_C$ のアップグレードにより、異なるオプション値になる可能性があります（次の表を参照）。

TargetLink バージョン A、B、C のデフォルト値が 9、11、13 を読み取るとします。バージョン A でオプション値が 11 と等しい場合は、バージョン C へのアップグレードでは次のようにオプション値が変わります。

アップグレード方法	オプション値 TL_A デフォルト = 9	オプション値 TL_B デフォルト = 11	オプション値 TL_C デフォルト = 13
A ⇒ B ⇒ C	11（≠ デフォルト）	11（= デフォルト）[1]	13（= デフォルト）[1]
A ⇒ C	11（≠ デフォルト）	—	11（≠ デフォルト）

[1] オプション値は、デフォルト値と等しいためこのモデルで保存されません。

新しい Code Generator オプション

新しい Code Generator オプションの詳細については、「新しい Code Generator オプション」(209 ページ)を参照してください。

関連トピック

リファレンス

- 「Code Generator Options」(『TargetLink Block and Object Reference』)

API 関数とフック関数

TargetLink と TargetLink Data Dictionary API 関数の変更

カスタムルックアップ関数

`tlscript` API コマンドでは、入力信号の次元に関連する新しいプロパティを使用することができます。これにより、たとえば、テーブル検索方法 Local search で最後のインデックス状態変数を実装することができます。

- `InheritDimensionFromInput`

関連ドキュメント

- 「Permissible Properties for Variables」(『TargetLink API Reference』)
- 「Basics on Using Custom Look-Up Functions」(『TargetLink Preparation and Simulation Guide』)

AUTOSAR に関する移行上の注意点

AUTOSAR に関する移行上の注意点

ランナブルオブジェクトの名前テンプレートでの名前マクロ	TargetLink 4.1 では、Runnable オブジェクトの NameTemplate プロパティに$D 以外の名前マクロを含めることはできません。
インポート／エクスポートで削除されたオプション	Merge および EnablePackageSupport オプションは廃止されました。 /Pool/Autosar/Config/ImportExport 内の対応するプロパティは削除されました。 TargetLink では常に以下を実行するようになりました。 ■ 1 つのソフトウエアコンポーネントに属するが、別のサブシステムに存在する Data Dictionary エレメントを、1 つのマージされたファイルにマージします（Merge = On）。 ■ AUTOSAR ファイルと Data Dictionary で提供されるパッケージ情報をインポートおよびエクスポートします（EnablePackageSupport = On）。 これに応じて、ユーザスクリプトを適宜修正してください。
Data Dictionary の RTE エラーコードマクロ	/Pool/Variables/AUTOSAR/Std_ReturnType 変数グループおよび事前定義された変数クラス/Pool/VariableClasses/AUTOSAR/RTE/RTE_ERROR_CODE 内の RTE エラーコードマクロを表す変数は、ModuleRef プロパティが Rte_Frame に設定されるようになりました。 これは AUTOSAR 規格に基づくもので、このマクロを Rte.h で定義することが求められています。 各変数の ModuleRef プロパティの新しい値は、Data Dictionary のアップグレード時に自動的に設定されます。 このため、モデル内のブロックでこれらの変数のいずれかを参照している場合、量産コードに#include Rte.h が追加される可能性があります。

コードの変更

コードの変更

Stateflow チャートから生成されたコード

特別な Stateflow セマンティクスにより正確に従うために（たとえば、特定のモデリングスタイルでの MIL と SIL の相違を回避するためなど）、Stateflow チャートから生成されたコードは、次のような Stateflow チャートシナリオでは効率が低下する場合があります。

- 他のチャートからインポートされたグラフィカル関数が呼び出される
- 関数呼び出し出力イベントが発生する

結果として、定数が伝播されないか、使用されていないコードフラグメントが削除される可能性があります。

これを変更するには、SIDE_EFFECT_FREE 最適化フラグを有効にした関数クラスを、インポートされたグラフィカル関数や、関数呼び出し出力イベントによってトリガされるサブシステム／チャートに割り当てます。

関数に実際に副作用がないことを保証する必要があります。たとえば、副作用のない関数は、以下のことを行いません。

- グローバル変数の変更
- 副作用のある関数の呼び出し

反復されるサブシステム内の Assignment ブロック

反復されるサブシステム内に複数の Assignment ブロックが存在する場合、最初の 1 つの反復フラグのみを実装して、コードの効率性を向上させます。その結果、このようなフラグの名前が変更されます。たとえば、次のようになります。

TargetLink 4.0 以前	TargetLink 4.1
{Subsystem_AssignmentBlock}_FirstIter	{Subsystem_IterationBlock}_FirstIter

フラグは、反復作業の開始時に明示的に初期化されます。

```
…_FirstIter = 1
```

フラグは、反復作業の終了時にリセットされます。

```
…_FirstIter = 0
```

ネストされたサブシステム内の Assignment ブロック　アトミックサブシステム内に 1 つ以上の Assignment ブロックが存在し、このアトミックサブシステムが反復サブシステム内に存在する場合、反復変数に次の変数スコープが割り当てられます。

- *Global*：アトミックサブシステムの関数が*インライン化されていない*場合。
- *Local*：アトミックサブシステムの関数が*インライン化されている*場合。

飽和加算または飽和減算のコードパターン

TargetLink 4.1 では、Code Generator オプション ExploitComputeThroughOverflow が NEVER に設定されている場合、飽和加算または飽和減算のコードに Compute Through Overflow（CTO）コードパターンが使用されることはありません。次の例は、加算 `out = in + Const` を `I16Out = I16In +1` で示したものです。

TargetLink 4.0 以前	TargetLink 4.1
`if (I16In > 32766) { /* Max(Result type) - Const */` `  I16Out = 32767;  }` `else {` `  I16Out = (Int16) (( /* CTO */ (UInt16) I16In) + 1);  }`	`if (I16In > 32766) { /* Max(Result type) - Const */` `  I16Out = 32767;  }` `else {` `  I16Out = (Int16) (I16In + ((Int16) 1));  }`

Code Generator オプション ExploitComputeThroughOverflow が Always または Optimized に設定されている場合、量産コードは以前のバージョンと同じになります。

Void typedef の生成なし

MISRA-C への準拠を向上させるために、TargetLink ではデフォルトで tl_basetypes.h 内に Voidtypedef が生成されなくなりました。代わりに標準の C void データ型が使用されます。

ターゲットとコンパイラの組み合わせに属する TargetConfig.xml ファイルを編集することで、これを以前の動作（Void を使用）に戻すことができます。

このファイルは、<TL_InstRoot>\Matlab\Tl\TargetConfiguration\<MicrocontrollerFamily>\<CompilerFamily>内、または TSM 拡張フォルダ内の同じ場所にあります。

再度 Void 基本タイプを tl_basetypes.h に生成するように TargetLink に指示するには、以下のようにします。

1. ターゲットとコンパイラの組み合わせに属する TargetConfig.xml ファイルを開きます。
2. name 属性が Void に設定されている ddObj XML エレメントを見つけます。
3. name 属性が CodedType に設定されている子 ddProperty XML エレメントを見つけます。

4. その値を Use standard C void type から void に変更します。
5. TargetConfig.xml ファイルを保存してコードを生成します。

デフォルトスケーリングのコードコメントの追加

可読性を向上させるために、TargetLink では、変数定義でデフォルトのスケーリングに関するコードコメントが追加されています。

TargetLink 4.0 以前	TargetLink 4.1
`Int16 SX1_OutPort1_FR_Actual;`	`Int16 SX1_OutPort1_FR_Actual /* LSB: 2^0 OFF: 0 MIN/MAX: -32768 .. 32767 */;`

プリプロセッサの#IF ステートメントのカプセル化

TargetLink では、インクルードされるファイルの定義と宣言がすべて#IF 命令でカプセル化されている場合のみ、プリプロセッサ命令#include を #IF 命令によってカプセル化するようになりました。

例として、次の FuncDefModule.h ヘッダーファイルについて考えます。

```
extern GLOBAL Int16 SEnc1_Out1;

#if FLAG
extern void Encapsulated(void);
#endif
```

TargetLink の生成されるコードは次のように変更されます。

TargetLink 4.0 以前	TargetLink 4.1
`#if FLAG` `#include "FuncDefModule.h"` `#endif` `…` `void TL_Root(void)` `{` `  #if FLAG` `  Encapsulated(Sa1_InPort);` `  #endif` `  SEnc1_Out1 = …` `}`	`#include "FuncDefModule.h"` `…` `void TL_Root(void)` `{` `  #if FLAG` `  Encapsulated(Sa1_InPort);` `  #endif` `  SEnc1_Out1 = …` `}`

ベクトルまたは配列変数のインデックスでのコードコメント

可読性を向上させるために、ベクトルまたは行列変数の初期値に関するコードコメントが変更されました。

TargetLink 4.0 以前	TargetLink 4.1
ベクトル	
`Int16 MyVar[3] =` `{` `  /*[0..2]*/ 61, 52, 43` `};`	`Int16 MyVar[3] =` `{` `  /* [0..2] */ 61, 52, 43` `};`

TargetLink 4.0 以前	TargetLink 4.1
行列	
Int16 MyVar[2][3] = { { **/*[0..2]*/** 61, 52, 43 }, { **/*[0..2]*/** 34, 25, 16 } };	Int16 MyVar[2][3] = { { **/* [0][0..2] */** 61, 52, 43 }, { **/* [1][0..2] */** 34, 25, 16 } };

暗黙的に生成される構造体タイプの識別子

暗黙的に生成される構造体タイプに対する TargetLink の識別子は、AUTOSAR 規格に準拠するようになりました。これは次の正規表現に準拠しています。

```
[a-zA-Z]([a-zA-Z0-9]|_[a-zA-Z0-9])*_?
```

$C、$R または$S 名前マクロを使用して指定した typedef 識別子に対して、TargetLink は次のことを行います。

- 暗黙的に生成された typedef の識別子の先頭のアンダースコアを削除します。
- 2 つのアンダースコアを 1 つに置き換えます。

TargetLink 4.0 以前	TargetLink 4.1
/* update(s) for inport Subsystem/Func/In1_with_super_long_name_to_break_limit */ Rte_Pim_ACP_a(instance)->Sa2_In1_with_s__to_break_limit = (sint16) DataElement;	/* update(s) for inport Subsystem/Func/In1_with_super_long_name_to_break_limit */ Rte_Pim_ACP_a(instance)->Sa2_In1_with_s_to_break_limit = (sint16) DataElement;

アンダースコアのみで構成される識別子、または先頭のアンダースコアの後に数字が続く識別子は変更されません。これらの typedef は、自動生成される Per Instance Memory（PIM）には生成されません。

TargetLink 4.0 以前	TargetLink 4.1
/* update(s) for inport Subsystem/Func/In1 */ Rte_Pim_ACP_a(instance)->**_1Var** = (sint16) DataElement;	/* update(s) for inport Subsystem/Func/In1 */ **_1Var** = (sint16) DataElement;

条件によって実行されるシステムの出力ポート用の IF 変数

MIL シミュレーションと SIL シミュレーションの相違の可能性を排除するために、TargetLink では、条件によって実行されるサブシステムの拡張されていない出力ポートに対して、次のブロックのいずれかが前にある場合、追加の変数 IF_<Suffix>を生成するようになりました。

- Data Type Conversion
- (Matrix) Concatenate
- Permute Dimensions
- Reshape
- Selector
- Bus Assignment

- Zero Order Hold
- Rate Transition

最適化に関するコードコメント

可読性を向上させるために、TargetLink の最適化に関するコードコメントが変更されました。チェーン `A replaced by … replaced by … X` は `A replaced by X` に置き換えられます。

TargetLink 4.0 以前	TargetLink 4.1
`/* Gain: foo/Gain` **`Variable 'Sa1_Gain' replaced by 'Aux_f'`** **`Variable 'Aux_f' replaced by 'Aux_F32_e' */`**	`/* Gain: foo/Gain` **`Variable 'Sa1_Gain' replaced by 'Aux_F32_e' */`**

補助変数に対するアクセス関数なし

アクセス関数によって生成される補助変数に対して、アクセス関数テンプレート(AFT)を定義できなくなりました。以前の動作では、必要以上に多くのアクセス関数が作成されることや、コード生成時にほぼ無限のループが発生することもありました。

加算と減算用の新しい CTO 回避マクロ

MISRA-C 適合度を向上させるために、TargetLink の固定小数点ライブラリに、16 ビット以下のオペランドの加算と減算用の新しいマクロが追加されました。これらのマクロは、CTO コードパターンを抑制している場合(Code Generator オプション ExploitComputeThroughOverflow により)のみ生成されます。マクロ名は常に `_PROT` で終了し、`I16` または `U16` を含みます。

TargetLink 4.0 以前	TargetLink 4.1
`C__I32ADDI32U32_PROT((Int32) Sa1_I16InPort, Sa1_U32InPort)`	`C__I32ADDI16U32_PROT(Sa1_I16InPort, Sa1_U32InPort)`

論理演算の間接参照を括弧内に記述

論理演算の間接参照は、MISRA-C に準拠して括弧内に記述されるようになりました。

TargetLink 4.0 以前	TargetLink 4.1
`Sa1_OutPort1 = (Int16) (*In2 || *In1);`	`Sa1_OutPort1 = (Int16) ((*In2) || (*In1));`

MinMax ブロックのコードパターン

2 つ以上の入力、または 2 つ以上の信号を持つ 1 つの入力を含む MinMax ブロックでは、最初の 2 つの入力または信号が出力に適合しない場合にコードパターンが変更されます。つまり、次のいずれかが適用されます。

- 比較の順序が変更されます。
- 出力は、各信号と比較される前に、最小値または最大値で初期化されます。

RTE_Invalidate のステータス

TargetLink は、RTE API 関数 Rte_Invalidate の戻り値を評価できるようになりました。次の表に、Invalidate および Status ポートを有効にした SenderComSpec ブロックに対する Rte_Invalidiate 呼び出しのコードサンプルを示します。

TargetLink 4.0 以前	TargetLink 4.1
if(InvalidateCondition) { Rte_Invalidate_x_y(); **status = 0;** } else { status = Rte_Write_x_y(); }	if(InvalidateCondition) { **status = Rte_Invalidate_x_y();** } else { status = Rte_Write_x_y(); }

インクリメンタルコード生成用のサブシステムの命名変更

参照先モデルから変換された、インクリメンタルコード生成用に設定されたサブシステムのコードは、以前のバージョンとは異なる場所に保存される場合があります。詳細については、「移行に関するその他の注意点」(243 ページ)を参照してください。

ロギング変数に基本タイプを使用

ロギングのために Sink ブロックに補助変数が作成される場合、その補助変数は常に TargetLink 基本タイプになります(AUTOSAR ではプラットフォームタイプ)。ただし、ロギングコードは量産コードではありません。

Abs パターンの変更

可読性を向上させるために、短縮した固定小数点 Abs パターンはサポートされなくなりました。代わりに、if-else 式が使用されます(例:スケーリング設定されていない Int16 Abs とスケーリング設定されていない Int16 入力)。

TargetLink 4.0 以前	TargetLink 4.1
Sa1_OutPort = Sa1_InPort; if (Sa1_InPort < 0) { Sa1_OutPort = -Sa1_OutPort; }	if (Sa1_InPort >= 0) { Sa1_OutPort = Sa1_InPort; } else { Sa1_OutPort = (Int16) (-Sa1_InPort); }

次の変更が適用される場合があります。

- 別の(通常はより良い)範囲情報が継承されるため、後続のコードパターンが変更される場合があります。例:段落に符号が続く場合、その後の負の分岐は省略されます。これは、Abs の出力は常に正の値であるためです。
- 飽和では、KeepSaturationStatements が設定されている場合、追加の 64 ビット演算が生成されます
- FloatOutVar = abs (IntVar) のような Stateflow 式では、別のコードパターンが生成されます(?演算子を使用して)。

非スカラーの AUTOSAR または間接関数の再利用での追加のキャスト

左辺のインデックス付きポインタにアクセスすると必ず、TargetLink の一般的なキャストスタイルへの適合性が向上したキャストが追加されます。

TargetLink 4.0 以前	TargetLink 4.1
`pISV->pISV_SL1_0_tp->pSL1_ImplicitOut1[Aux_S32] = 0;`	`pISV->pISV_SL1_0_tp->pSL1_ImplicitOut1[Aux_S32] = (sint16) 0;`

この特定のケースでは、グラウンド記号が元は右辺にあったためにキャストが作成されます。

オーバーフローなし、単項マイナス

次の例は、Int16 入力と UInt16 出力を含む Abs ブロックのコードを示しています。

TargetLink 4.0 以前	TargetLink 4.1
`if(I16In >= 0) {` `  …` `} else {` `  U16Out = (UInt16)(-I16In)` `}` このパターンには重大な問題があります。値-32768(INT16MIN)で未定義のオーバーフローが発生する可能性があるためです。32768 は 16 ビットのプラットフォーム int に適合せず、int でマイナスが計算されます。	`if(I16In >= 0) {` `  …` `} else {` `  if (I16In == -32768) {` `  U16O7ut = 32768;` `  } else {` `  U16Out = (UInt16)(-I16In)` `  }` `}` これで負の最小値が使用されるようになります。

2 番目の *if-else* は、次のように?演算子に置き換えられる場合があります。

```
… (I16In == -32768) ? 32768 : (UInt16)(-I16In)
```

割り当ての右辺の最後の演算が単項マイナスでない場合は、常にこのようになります。一般的な例として、入力／出力のスケーリングが異なる Abs ブロックがあります。この場合、マイナスが再スケーリング(シフト／除算)のオペランドです。結果として、?演算子が使用されます。

次の式およびブロックが影響を受ける可能性があります。

- 単項マイナスを含む Stateflow 式
- Abs ブロック
- 負のゲイン値を含む Gain ブロック
- 負の定数デフォルトを含む Product ブロック
- 対応する設定を含む Sum ブロック

飽和マクロ：FIT マクロ呼び出しが SAT マクロ呼び出しに置換

TargetLink の固定小数点ライブラリには、次の 2 種類の飽和マクロがあります。

- 範囲制限（実装範囲）で飽和を実行する FIT マクロ
- ユーザ定義の制限で飽和を実行する SAT マクロ（例：Saturation ブロックまたは Stateflow で）

TargetLink 4.1 より前のバージョンでは、まれに、範囲制限での飽和が実行されたにもかかわらず、SAT マクロが呼び出されることがありました。これは、整合性と MISRA-C への準拠の向上のために変更されています。

次の例は、以下の場合の Stateflow での飽和を示しています。

```
UInt16 U16Out ; // LSB = 0.002 and checkmax = 1

U16Out = U16Out + I8In;
```

TargetLink 4.0 以前	TargetLink 4.1
Aux_I32 = … U16Out = C__U16SATI32_SATb(Aux_I32, 65535 /* 131.07 */, 0)	Aux_I32 = … U16Out = C__U16FITI32_SAT(Aux_I32, 65535 /* 131.07 */)

飽和マクロ

Saturation ブロックまたは Stateflow 用に生成された SAT マクロを呼び出すときに、制限の変数クラスがデフォルトと異なる場合、制限のタイプ（パラメータ 2、3）は出力のタイプに一致するように調整されます。

TargetLink 4.0 以前	TargetLink 4.1
GLOBAL Int32 Sb1_Saturation_lower = 0; GLOBAL Int32 Sb1_Saturation_upper = 100; Int16 Aux_S16; Int16 Aux_S16_a; Aux_S16 = (Int16) Sb1_Saturation_upper; Aux_S16_a = (Int16) Sb1_Saturation_lower; Sb1_OutPort = C__U16SATI16_SATb(Sb1_InPort, Aux_S16, Aux_S16_a);	GLOBAL Int32 Sb1_Saturation_lower = 0; GLOBAL Int32 Sb1_Saturation_upper = 100; UInt16 Aux_U16; UInt16 Aux_U16_a; Aux_U16 = (UInt16) Sb1_Saturation_upper; Aux_U16_a = (UInt16) Sb1_Saturation_lower; Sb1_OutPort = C__U16SATI16_SATb(Sb1_InPort,Aux_U16, Aux_U16_a);

ループ内の行列変数へのアクセス

コード効率性を向上させるために、ループ内の行列変数へのアクセスをスカラー変数で置き換えることができます。ループ外部からアクセスされた場合でも、場合によっては、行列およびベクトル変数が 1 つまたは複数のスカラー変数で置き換えられます。

TargetLink 4.0 以前	TargetLink 4.1
`Int16 vec[…];` `vec[0] = …` `  … = v[0];` `loop (i = 1:n) {` `  v[i] = …` `  = v[i];` `}`	`Int16 scalar;` `scalar = …` `  … = scalar;` `loop (i = 1:n) {` `  scalar = …` `  … = scalar;` `}` または、さらに多くのスカラー変数が導入されている場合: `Int16 scalar1;` `Int16 scalar2;` `scalar1 = …` `  … = scalar1;` `loop (i = 1:n) {` `  scalar2 = …` `  … = scalar2;` `}`

ループ外部からの行列変数へのアクセス

ループ外部からの行列変数へのアクセスを、1 つまたは複数のスカラー変数に置き換えることができます。また、ループ内部からのアクセスとループ外部からのアクセスの組み合わせを最適化できるようになりました。さらに、インデックス式のアクセスが同じように構築されている場合、インデックスの不明なアクセスを最適化することができます。

スカラー変数による置換時に、通常はメモリ節約が発生します(累積で)。ただし、状況によっては、メモリ消費を節約することはできません。つまり、新しいスカラー変数は、最初の多次元変数として可能なかぎり多くのメモリを使用します。

これは、完全にアンロールされたコードに対してのみ発生します。正確には、次の場合に発生します。

- (ベクトル変数の場合)`Number of elements < LoopUnrollThreshold`
- (行列変数の場合)`Number of elements < LoopUnrollThreshold`^{2}

アクセス元	TargetLink 4.0 以前	TargetLink 4.1
ループ外、ベクトルのアナログ	`M[0][0] = …;` `…` `… = M[0][0];`	`Aux = …;` `…` `… = Aux;`

アクセス元	TargetLink 4.0 以前	TargetLink 4.1
（部分的にアンロールの）ループ内およびループ外、ベクトルのアナログ	`M[0][0] = …;` `loop {` `  M[1][i] = …` `  M[2][i] = …` `  …` `  … = M[1][i];` `  … = M[2][i];` `}`	`Aux = …;` `loop {` `  Aux_a = …` `  Aux_b = …` `  …` `  … = Aux_a;` `  … = Aux_b;` `}`
ループ外、行列のアナログ	`V[<expr>] = …;` `…` `… = V[<expr>];`	`Aux = …;` `…` `… = Aux;`

タンジェントコードパターン

Stateflow 内からのタンジェント、および TargetLink Trigonometric および Math ブロックのタンジェントのコードパターンが修正されました。

Stateflow 内のタンジェント TargetLink 4.1 までは、飽和が誤って省略されていました（特に飽和式の場合）。

以下の規則が適用されるようになりました。

- タンジェントが `out = tan(expr)` などの割り当てで使用される場合、`out` が固定小数点タイプであればタンジェントは固定小数点で計算されます。
- 複雑な式など、他のすべての場合では、TargetLink はタンジェント計算を浮動小数点で実行します。コード生成中に純粋な固定小数点のコンテキストが検出された場合は、エラーメッセージが出力されます。

Trigonometric および Math ブロックのタンジェント LSB >= 2^{-14} で結果が 32 ビットまたは 16 ビットの場合、不要な飽和はデフォルトで省略されるようになりました。そのため、*省略された飽和*コメントは場合によっては削除されます。

ビットフィールドへの割り当て

Stateflow 状態 ID の割り当て Use bitfields in state machines = On で、多値の状態変数がある場合、Stateflow 状態 ID の `unsigned int` へのキャストは不要になりました。

TargetLink 4.0 以前	TargetLink 4.1
`SIBFS_Chart_a.Ca1_Chart_ns = (unsigned int) Ca5_OFF_id;`	`SIBFS_Chart_a.Ca1_Chart_ns = Ca5_OFF_id;`

また、Stateflow 状態 ID のクラスデフォルト設定が変更されました。Stateflow 状態 ID は、生成されるコード内のグローバルマクロとなり、初期値はキャストされません。

純粋なビットフィールド用のビットフィールドセマンティクス TargetLink は、ビットフィールドが Boolean 型ビットフィールド（UseGlobalBitfieldsForBooleans = On）か純粋なビットフィールド（Base type = Bitfield）かを評価します。TargetLink 4.0 以前のバージョンでは、Bool 型セマンティクスは、Boolean 型ビットフィールドと純粋なビット

フィールドの両方に適用されます。必要に応じて、!= 0 が追加されます。TargetLink バージョン 4.1 以降、ビットフィールドのセマンティクスは純粋なビットフィールドに適用されます。

TargetLink 4.0 以前	TargetLink 4.1
BooleanBitfield = (Int16Var != 0) GenuineBitfield = (Int16Var != 0)	BooleanBitfield = (Int16Var != 0) GenuineBitfield = (Int16Var & 1)

ビットフィールドのセマンティクスは定数にも適用することができます。たとえば、次のように指定します。

```
GenuineBitfield = 3 & 1
```

コードの最適化は、定数が値 0 または 1 の場合のみ実行することができます。

ビットフィールドへの即時割り当て　右辺が次のいずれかの場合、ビットフィールドへの即時割り当てが生成されます。

- 1 つのビットフィールド
- 1 つの Bool 式
- 1 つの& 1（ビット）演算
- 1 つの Stateflow 状態 ID マクロ

浮動小数点やスケーリングオペランドを純粋なビットフィールドに割り当てた場合、TargetLink は!= 0 を生成せずに、エラーを返すようになりました。

再スケーリングのための Z および N 値の向上

Z 値および N 値の計算アルゴリズムが改善されました。これにより、より小さいビット幅に基づいた計算で、精度または効率性のどちらかを向上させるコードになります。

TargetLink 4.0 以前	TargetLink 4.1
Sb1_PRODUCT = (Int16) (((Int32) ((((Int32) Op1) * ((Int32) Op2)) << 2)) / 39);	Sb1_PRODUCT = (Int16) (((Int32) ((((Int32) OP1) * ((Int32) Op2)) << 6)) / 625);

コード効率性を向上させるために、パラメータの許容誤差がより頻繁に使用されるようになりました。つまり、パラメータの許容誤差が 0 より大きい場合、結果としてコードの精度は低くなります。必要に応じて、パラメータの許容誤差を下げることで、この結果を変更することができます。

UInt32 変数と Int32 変数の比較

TargetLink では、UInt32 変数と Int32 変数を比較するときに、符号付き変数が負かどうか、また 64 ビットマクロを回避できるかどうかをチェックするようになりました。

TargetLink 4.0 以前	TargetLink 4.1
C__I64COPYU32(U32Var, I64Var_hi, I64Var_lo); C__I64COPYI32(I32Var, I64Var2_hi, I64Var2_lo); if(C__LE64(I64Var_hi, I64Var_lo, I64Var2_hi, I64Var2_lo))	<=、<、および!=の場合: if(I32Var < 0 || ((UInt32)I32Var <op> U32Var)) >、>=、および ==の場合: if(I32Var >= 0 && ((UInt32)I32Var <op> U32Var))

結果として、その後の最適化により、生成されたコードから不要な論理分岐を除外することができます。

64 ビット変数の比較

内部で計算されるワーストケースのレンジ幅がどちらも 32 ビット以下の 2 つの 64 ビット変数を比較する場合に、TargetLink は上位 32 ビットも考慮するようになりました。たとえば、Int64Var は負の値になる場合があります。

TargetLink 4.0 以前	TargetLink 4.1
if(Int64Var_lo < Int64Var2_lo)	if(C__LT(Int64Var_hi, Int64Var_lo, Int64Var2_hi, Int64Var2_lo))

この変更は、特に Discrete-Time Integrator ブロックに関係します。

中間変数の削除

TargetLink では、後でインデックス選択を行う計算では、完全な幅の計算を実行せず、目的のエレメントの計算のみを実行することで、中間変数の削除を実行できるようになりました。

TargetLink 4.0 以前	TargetLink 4.1
VectorVar[<iteration range>] = expression(<iteration range>) … = VectorVar[cnst]	… = expression(cnst)

次の条件を満たす必要があります。

- VectorVar[cnst] のみが VectorVar[<iteration range>] で使用されます。
- インデックス cnst は繰り返し範囲のサブセットです。これは、<iteration range> がすべてのエレメントをカバーする場合に満たされます。

For Iterator サブシステムの Assignment ブロック

Assignment ブロックが、For Iterator サブシステム内に存在するアトミックサブシステム内にある場合、次のことが適用されます。

ブロックプロパティ[Omit dispensable initializations]が[off]に設定されている場合、Y0 信号による初期化は最初のサンプルステップ中にのみ実行されます。これは Simulink の動作と同じです。

浮動小数点のゼロ除算チェック

TargetLink では、MISRA-C 適合性が向上しています。

不要な再スケーリングの回避 TargetLink では、無関係な再スケーリングが回避されます。

TargetLink 4.0 以前	TargetLink 4.1
`if ((((Float32) Sa1_ScaledDenom) * 0.25F) != 0) {` `  Sa1_OutPort1 = Sa1_F32Num_ / (((Float32) Sa1_ScaledDenom) * 0.25F);` `} else { …`	`if (Sa1_ScaledDenom != 0)  {` `  Sa1_OutPort = Sa1_F32Num / (((Float32) Sa1_ScaledDenom) * 0.25F);` `} else { …`

オフセット用の補助変数 TargetLink は次の補助変数を使用します。

TargetLink 4.0 以前	TargetLink 4.1
`if (((((Float32) Sa1_ScaledDenomWithOffset) * 0.25F) + 42.F) != 0) {` `  Sa1_OutPort1 = Sa1_F32Num_ / ((((Float32) Sa1_ScaledDenomWithOffset) * 0.25F) + 42.F);` `} else { …`	`Aux_F32 = (((Float32) Sa1_ScaledDenomWithOffset) * 0.25F) + 42.F;` `if (Aux_F32 != 0.F) {` `  Sa1_OutPort1 = Sa1_F32Num_ / Aux_F32;` `} else { …`

符号なし減算の結果

TargetLink 4.0 以前のバージョンでは、符号なしの減算は結果として加算になっていました。例:`U8Var = U8Var2 + 255;`

TargetLink 4.1 では可読性を向上させるために、定数が結果のタイプに適合する場合、減算は減算のままになります。例:`U8Var = U8Var2 - 1;` 計算はより小さいタイプで実行される場合があります。

TargetLink 4.0 以前	TargetLink 4.1
`U8Var = U8Var2 + 255;`	`U8Var = U8Var2 - 1;`

あらかじめ設定された AUTOSAR 変数クラス

追加の`#include tl_defines.h` 命令を抑制するために、あらかじめ設定された AUTOSAR 変数が次のように変更されました。既存の DD ファイルを更新する際に、それが DD AccessFunctionTemplate オブジェクトを参照する場合、TargetLink はあらかじめ設定された AUTOSARVariableClass オブジェクトの UseName プロパティを `off` に設定します。

結果として、次の変数クラスを持つ別の宣言または変数の定義になります。

TargetLink 4.0 以前	TargetLink 4.1
`extern` **`EXPLICIT_IRV`** `S16_LinPos Rte_Irv_Controller_LinPos`	`extern S16_LinPos Rte_Irv_Controller_LinPos`

これにより、`#include tl_defines.h` 命令の欠落が発生する場合があります。

CTO を使用しない加算と減算のコードパターン

効率性と MISRA-C への準拠を向上させるために、次の条件のすべてを満たす場合、加算と減算のコードパターンを変更することができます。

- ExploitComputeThroughOverflowCode Generator オプションが never に設定されている
- 演算のオペランドが符号なし
- 演算のオペランドが 32 ビット未満

TargetLink 4.0 以前	TargetLink 4.1
`UInt8Var = (UInt8)((Int16)UInt8Var + (Int16)UInt8Var)`	`UInt8Var = (UInt8)(UInt8Var + UInt8Var)`
`UInt8Var = (UInt8)((Int16)UInt8Var - (Int16)UInt8Var)`	`UInt8Var = (UInt8)(UInt8Var - UInt8Var)`

Data Store Memory ブロック変数の定義

整合性とユーザ制御を向上させるために、Data Store Memory ブロック変数の定義は常に、Data Store Memory ブロックを含むサブシステムのステップ関数が生成されるモジュールに配置されるようになりました。

これに応じて、次の条件をすべて満たす場合、Data Store Memory ブロック変数の定義は別のヘッダーファイルに配置される場合があります。

- Data Store Memory ブロックは、ネストされたサブシステムも含むサブシステムに配置されます。
- ネストされたサブシステムは、データストアメモリにアクセスする Data Store Read または Data Store Write ブロックを含みます。
- サブシステムのステップ関数は、次のように異なるモジュールに生成されます。
  - Data Store Memory ブロックを含むサブシステムのステップ関数は、モジュール A に生成されます。
  - Data Store Read または Data Store Write ブロックを含むサブシステムのステップ関数は、モジュール B に生成されます。

TargetLink 4.0 以前	TargetLink 4.1
モデリング状況によっては、TargetLink は Data Store Memory ブロックのブロック変数をモジュール B に定義する場合がありました。	TargetLink は、Data Store Memory ブロックのブロック変数を常にモジュール A に定義します。

Float32 比較での倍精度の丸め

TargetLink では、Float32 比較でのリテラル値の丸め動作が変更されています。この変更は次の理由により実施されました。

- 値がモデルでの指定と同じように使用されるため、コードを理解しやすくなります。
- ユーザ制御の強化:
  - モデル内で丸めた値を指定することができます。
  - C コンパイラオプションを使用して丸めを制御し、TargetLink が生成した量産コードとレガシーコードの整合性を保つことができます。

次の例では、データ型が `Float32` の変数と値 `0.1` を比較しています。

```
0.1 > Float32Var
```

次の表に、TargetLink の丸め動作を示します。

TargetLink 4.0 以前	TargetLink 4.1
`Sa1_Relational_Operator = 0.1000000015F > Sa1_F32In;`	`Sa1_Relational_Operator = 0.1F > Sa1_F32In;`

Float32 値の倍精度

モデルと量産コード間のトレーサビリティの改善のため、また、前述した丸め制御のために、TargetLink では `Float32` のリテラル値を倍精度で、`F` サフィックスを付加して生成するようになりました。次の表は、`GainValue = pi` での `Float32` ゲインの例を示しています。

TargetLink 4.0 以前	TargetLink 4.1
`Sa1_OutPort = Sa1_InPort * 3.141592654F;`	`Sa1_OutPort = Sa1_InPort * 3.1415926535897931F;`

TargetLink 4.0 以前のバージョンと同様に、モデルで値が浮動小数点精度または単精度で正確に指定されていない場合、これによって生成されたコードに小数点以下の桁数がより多く表示される場合があります。

コード効率性の向上

TargetLink 4.1 では、コード効率性が向上しています。これに伴い、生成されるコードも以前の TargetLink バージョンと比べて変更される場合があります。詳細については、「コード効率性の向上」(196 ページ)を参照してください。

その他

移行に関するその他の注意点

TargetLink Main Dialog ブロック

[Compare with reference]コマンドは、ブロックの[Code Generation]ページにリストされるコード生成ユニットのコンテキストメニューから削除されました。

参照先モデルの変換

参照先モデルを変換する際の TargetLink の動作が変更されました。

参照先モデルをインクリメンタルサブシステムに変換 参照先モデルをインクリメンタルコード生成用に設定されたサブシステムに変換する場合、作成されたサブシステムは、次の表に示すように名前が付けられます。

参照先モデルの再利用	TargetLink 4.1 より前のバージョン	TargetLink 4.1
あり	-	サブシステム名 = モデルブロック名
なし	サブシステム名 = モデルブロック名	サブシステム名 = モデル名

したがって、インクリメンタルコード生成用に設定されたモデルに対して生成されるコードには、新しい保管場所があります。

変換前の参照先モデル(TL4.0 および TL4.1)	変換後のインクリメンタルサブシステム (TL4.0)	変換後のインクリメンタルサブシステム (TL4.1)
生成されたコードモジュール: ■ <ModelName>.c[1)] ■ <ModelName>.h[1)] DD Subsystems オブジェクトの名前: <ModelName>	生成されたコードモジュール: ■ <ModelBlockName>.c[1)] ■ <ModelBlockName>.h[1)] DD Subsystems オブジェクトの名前: <ModelBlockName>	生成されたコードモジュール: ■ <ModelName>.c[1)] ■ <ModelName>.h[1)] DD Subsystems オブジェクトの名前: <ModelName>

[1)] モデルの Function ブロックでモジュール名を直接指定しなかった場合

インクリメンタルサブシステムから参照先モデルへの変換 TargetLink は、参照先モデルのシミュレーションモードを次の表のように設定します。

TargetLink 4.1 より前のバージョン	TargetLink 4.1
アクセラレータ	TargetLink 4.1 でサブシステムを参照先モデルから変換した場合、再変換された参照先モデルのシミュレーションモードは、元の参照先モデルのシミュレーションモードと同じになります。 別の TargetLink バージョンでサブシステムを変換した場合、または一度も変換しなかった場合、参照先モデルのシミュレーションモードは Normal に設定されます。

関連ドキュメント

- `tl_refmodel_to_subsystem(propertyName, propertyValue, ...)` (🕮『TargetLink API Reference』)
- `tl_subsystem_to_refmodel(propertyName, propertyValue, ...)` (🕮『TargetLink API Reference』)

Float32 と固定小数点を含む明示的にモデリングされた飽和

TargetLink の Saturation ブロックを使用せずに(たとえば、代わりに MinMax ブロックを使用して)、固定小数点または Float32 を含む飽和をモデリングする場合、TargetLink はその意図を推測することができません。このような場合、C コンパイラによる倍精度値から Float32 の単精度への丸めの効果を必ず確認してください。または、必要に応じて、Saturation ブロックを使用するか、モデル内の既に正しく丸められている単精度値を指定します。

TargetLink のブロック出力コードの飽和は影響を受けないことに注意してください。これは、TargetLink は内部で必要に応じて正しく丸められた単精度値を計算するためです。変更は、ユーザ指定の値にも関連します。「Float32 比較での倍精度の丸め」(242 ページ)も参照してください。

廃止事項

廃止された制限事項

TargetLink 4.1 では、以前の TargetLink バージョンの以下の制限事項がなくなりました。

全般的な制限事項

バスの初期化 [1]
バス信号は、次のいずれかの条件を満たしている場合のみ初期化することができます。 ■ すべてのバス信号が同じタイプの列挙である。 ■ すべてのバス信号が非列挙である。 基本的に、同じスカラー初期値がすべてのバス信号に適用されます。

[1] TargetLink 4.1 では Simulink の簡易モードがサポートされるため、この制限事項はなくなりました。

ブロック固有の制限事項

Look-Up Table ブロック（ベクトル信号）
条件に一致するカスタムルックアップスクリプトが有効な場合、テーブルルックアップ関数は、ユーザが作成した関数に置き換えられます。これは、出力がスカラー値の Look-Up Table ブロックのみ適用されます。カスタムルックアップスクリプト機能では、ベクトル／行列入力信号や統一されたエレメントはサポートされません。

Merge ブロック
バス信号を Simulink Initial Condition Structure に渡すブロックの Initial Output パラメータを設定することにより、バス信号エレメントの初期値を指定することはできません。

OutPort ブロック
バス信号を Simulink Initial Condition Structure に渡すブロックの Initial Output パラメータを設定することにより、バス信号エレメントの初期値を指定することはできません。

Rate Transition ブロック
バス信号を Simulink Initial Condition Structure に渡すブロックの Initial Output パラメータを設定することにより、バス信号エレメントの初期値を指定することはできません。

コンポーネントベース開発の制限事項

Function Reuse（関数の再利用）
TargetLink では、1 つのモデル階層内での同じモデルの複数回参照はサポートされていません。そのため、参照先のモデルに対して生成されたコードを再利用することはできません。

Simulink モデルの引数
TargetLink では、参照先のモデルに対する Simulink モデルの引数をサポートしていません。

コード生成の制限事項

インクリメンタルコード生成
インクリメンタル生成されたサブシステムまたは参照先モデルは関数の再利用ができず、再利用関数の中に配置することもできません。

TargetLink の今後のバージョンでの変更予定

項目の一覧

本章の内容

廃止予定の機能

A2L のインポート	A2L のインポートは、今後の TargetLink バージョンで廃止される予定です。
RTF ドキュメントの生成	リッチテキスト形式（RTF）でドキュメントを生成するオプションは、今後の TargetLink バージョンで廃止される予定です。
MISRA-C:2004 Compliance Documentation ドキュメント	MISRA-C:2004 Compliance Documentation ドキュメントは、今後の TargetLink バージョンで廃止される予定です。TargetLink ユーザは、代わりに MISRA-C:2012 Compliance Documentation を使用してください。
Simulink のクラス初期化モード	Simulink のクラス初期化方法（`Classic` に設定された Underspecified initialization detection パラメータ）は今後の TargetLink バージョンで廃止される予定です。
動的コンポーネント	DD Variable オブジェクトに対する動的コンポーネントの指定は、今後の TargetLink バージョンでサポートが廃止される予定です。
特別な OSEK バージョン用のコード生成	OsCan などの特別な OSEK バージョン用のコード生成は、今後の TargetLink バージョンで廃止される予定です。
Signal logging format	Simulink のロギング方法 `ModelDataLogs`（Signal logging format パラメータ）は、今後の TargetLink バージョンでサポートが廃止される予定です。
V-ECU の OSA としての完全なビルド	今後の TargetLink バージョンでは、V-ECU を OSA ファイルとしてビルドできなくなります。TargetLink では、生成された量産コードは V-ECU インプリメンテーション（CTLGZ ファイル）としてのみエクスポートされ、プラットフォーム固有のビルドプロセスは、オフラインシミュレーションの場合は VEOS、リアルタイムシミュレーションの場合は ConfigurationDesk のままになります。

廃止予定の API 関数

廃止された API 関数

次の API 関数は廃止され、今後の TargetLink バージョンで削除される予定です。

関数	廃止されるバージョン	代わりの関数
tl_adapt_dd_references	TargetLink 4.0	**tlMoveDDObject**
tl_extract_subsystem	TargetLink 4.0	**tlExtractSubsystem**
tl_find_dd_references	TargetLink 4.0	**tlFindDDReferences**
tl_get_blockset_mode	TargetLink 4.0	**tlOperationMode**
tl_sim_interface	TargetLink 4.0	**tlSimInterface**
tl_switch_blockset	TargetLink 4.0	**tlOperationMode**
tl_upgrade	TargetLink 4.0	**tlUpgrade**

ユーザスクリプトを適宜調整するには、新しい API 関数に関するヘルプコンテンツを参照してください。

VEOS

項目の一覧

本章の内容

VEOS 3.5 の新機能

本章の内容

より直感的なユーザインターフェース

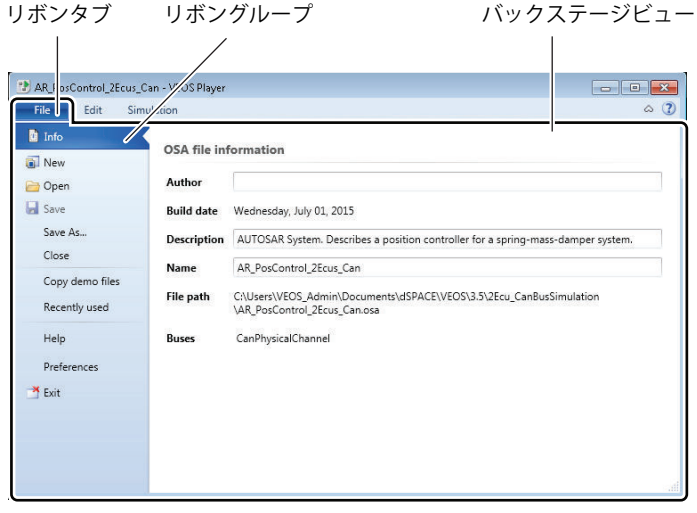
VEOS Player のユーザインターフェースがより直感的になりました。メニューバーとツールバーは、Microsoft Office などで使用される*リボン*と*バックステージビュー*に置き換えられています。

リボン　VEOS Player のリボンは、類似コマンドを整理してグループに分けています。リボンはユーザインターフェースの最上部に配置されています。次の図を参照してください。

各リボンには、関連するコマンド一式を含む*リボングループ*が存在します。

バックステージビュー　VEOS Player の*バックステージビュー*は、OSA ファイルのオープンや保存などの基本的なコマンドを提供します。ここから最近使用した OSA ファイルにすばやくアクセスすることもできます。

例として、[Info]リボングループが表示されているバックステージビューを下の図に示します。



コンテキストに応じたヘルプ　VEOS Player では、*コンテキストに応じた*ヘルプを利用できるようになりました。

VEOS Player で **F1** キーを押すか、[Help]ボタンをクリックすると、現在アクティブなコンテキストのヘルプが表示されます。

詳細については、「リボンの基礎」(🕮『VEOS ガイド』)を参照してください。

デバッグ情報の生成の有効化／無効化(MSVC、GCC)

VEOS 3.4 以前　VEOS 3.4 以前のバージョンでは、ソースコードのデバッグは以下のように処理されていました。

- たとえば、V-ECU インプリメンテーションなどのアイテムを MSVC コンパイラとの組み合わせでインポートする場合、ソースコードのデバッグは*常に有効化*されていました。
- たとえば、V-ECU インプリメンテーションなどのアイテムを GCC コンパイラとの組み合わせでインポートする場合、ソースコードのデバッグは*常に無効化*されていました。

VEOS 3.5 以降　VEOS 3.5 以降、MSVC コンパイラと GCC コンパイラのどちらとの組み合わせでも、*ソースコードのデバッグを有効化および無効化*することができます。

「オフラインシミュレーションでのソースコードのデバッグの基礎」(🕮『VEOS ガイド』)を参照してください。

例外時の呼び出しスタック情報へのアクセス

オフラインシミュレーション中に特定の VPU プロセス(FMU または SIC ベース)のモデルコードで例外が発生した場合、VEOS はエラーメッセージに次の呼び出しスタック情報を出力します。

- 例外が発生した関数名
- その関数を含むソースコードファイルの名前と場所
- 例外に関連する呼び出しスタック

VEOS は次の 2 つの条件を満たす場合のみ、呼び出しスタックを提供します。

- MSVC コンパイラが選択されている
- VPU のビルドプロセスに対してソースコードのデバッグを有効にしている

[Build Output]ダイアログ

たとえば、V-ECU インプリメンテーションをインポートおよびビルドする場合、VEOS Player では個別の[Build Output]ダイアログに次の情報が表示されるようになりました。

- コンパイラメッセージやリンカメッセージなど、ビルドプロセスに関する情報
- ビルドプロセス中に発生したすべてのエラーメッセージと警告メッセージの履歴

[Build Output]ダイアログは、VEOS Player の自動化インターフェースからインポートおよびビルドを実行した場合にも表示されます。

詳細については、「VPU、V-ECU インプリメンテーション、Simulink インプリメンテーション、FMU をインポートする方法」(📖『VEOS ガイド』)を参照してください。

アプリケーションを既に実行している場合のオフラインシミュレーションアプリケーションのロード

オフラインシミュレーションアプリケーションを既に実行しているときにオフラインシミュレーションアプリケーションを VEOS にロードする場合、VEOS では以下のどちらかを選択する画面が表示されます。

- 実行中のオフラインシミュレーションアプリケーションに接続する
- 実行中のオフラインシミュレーションアプリケーションを置き換える

「Load」(📖『VEOS Player リファレンス』)を参照してください。

オフラインシミュレーションアプリケーションのアンロード

VEOS Player では、オフラインシミュレーションアプリケーションを VEOS からアンロードできるようになりました。これにより、ControlDesk Next Generation などの他の製品が別のオフラインシミュレーションアプリケーションをロードできるようになります。

「Unload」(📖『VEOS Player リファレンス』)を参照してください。

FMU のインポート:列挙のサポート

列挙データ型の FMU の入出力がサポートされるようになりました。VEOS Player では、これらの入出力用の VPU ポートが作成されます。

VEOS 3.5 への移行

互換性一覧

VEOS と OSA　次の表は、VEOS 3.5 と OSA ファイルの互換性を示しています。

OSA ファイルを作成した製品	OSA のバージョン
dSPACE Release 2014-A	3.2
dSPACE Release 2014-B	3.3
dSPACE Release 2015-A	3.4
dSPACE Release 2015-B	3.5

- VEOS 3.5 で作成または変更した OSA ファイルを、それより前のバージョンの VEOS でロードすることはできません。
- ASM モデルのシミュレーションに関連する移行の問題があります。「ASM モデルの移行」(📖『VEOS ガイド』)を参照してください。

VEOS と CTLGZ（V-ECU インプリメンテーション） 次の表は、VEOS 3.5 と CTLGZ ファイル（V-ECU インプリメンテーション）の互換性を示しています。

V-ECU インプリメンテーションを作成した製品	V-ECU インプリメンテーションのバージョン
dSPACE Release 2013-B 以前: ■ SystemDesk 3.x ■ TargetLink 3.5	1.0
dSPACE Release 2014-A: ■ SystemDesk 4.2	2.0
dSPACE Release 2014-B: ■ SystemDesk 4.3 ■ TargetLink 4.0	2.1
dSPACE Release 2015-A: ■ SystemDesk 4.4	2.2
dSPACE Release 2015-B: ■ SystemDesk 4.5 ■ TargetLink 4.1	2.3

VEOS と SIC VEOS 3.5 は、dSPACE Release 2015-B（Simulink インプリメンテーションコンテナバージョン 1.0.1）から Model Interface Package for Simulink 3.1 で作成された Simulink インプリメンテーションコンテナ（SIC）ファイルと互換性があります。

ASM モデルのシミュレーションに関連する移行の問題があります。「ASM モデルの移行」(🕮『VEOS ガイド』)を参照してください。

VEOS と FMU FMU は、FMI 2.0 for Co-Simulation 規格に準拠している必要があります。FMI 1.0 規格準拠の FMU は、サポートされていません。

FMI をサポートする dSPACE 製品およびリリースの概要については、

http://www.dspace.jp/go/FMI-Compatibility を参照してください。

dSPACE Target for Offline Simulation の廃止と移行

Model Interface Package for Simulink *dSPACE Target for Offline Simulation* は、dSPACE Release 2015-A に付属のものが最後でした。Simulink®モデルを VEOS でのオフラインシミュレーション用に準備するには、dSPACE Release 2015-A で導入された *Model Interface Package for Simulink* を使用します。

Model Interface Package for Simulink に用意されている `dsrt.tlc` システムターゲットファイルを利用して、Simulink®モデルから Simulink インプリメンテーションコンテナ（SIC）ファイルを生成することができます。SIC ファイルはシミュレーションプラットフォームに依存しないため、同じ SIC ファイルを、VEOS で使用するオフラインシミュレーションアプリケーションと SCALEXIO で使用するリアルタイムアプリケーションに統合することができます。

dSPACE Target for Offline Simulation と *Model Interface Package for Simulink* のワークフローの違いの詳細については、「Differences Between DsOffSim and the Model Interface Package for Simulink」(🕮『Model Interface Package for Simulink - Modeling Guide』)を参照してください。

新しいワークフロー　*Model Interface Package for Simulink* がオフラインシミュレーション用の Simulink®モデルを準備するために提供するワークフローは、*dSPACE Target for Offline Simulation* が提供するワークフローとは異なります。

Simulink®モデルのオフラインシミュレーションを準備するには、次の手順を実行します。

1. Simulink®モデルで、Model Interface Package for Simulink が提供する `dsrt.tlc` システムターゲットファイルを指定します。
2. Simulink®モデル用のコードを生成します。

   dSPACE Target for Offline Simulation と異なり、*Model Interface Package for Simulink* では、特定のシミュレーションプラットフォームに対応したアプリケーションは生成されません。代わりに、Simulink®モデルコードを含む Simulink インプリメンテーションコンテナ（SIC）ファイルが生成されます。

   詳細については、「Generating Simulink Implementation Containers」(🕮『Model Interface Package for Simulink - Modeling Guide』)を参照してください。
3. SIC ファイルを VEOS Player にインポートして、VEOS のオフラインシミュレーションアプリケーションにモデルを統合します。

   VEOS Player は、VEOS シミュレーションプラットフォーム用の SIC ファイルを作成します。

   詳細については、「VPU、V-ECU インプリメンテーション、Simulink インプリメンテーション、FMU をインポートする方法」(🕮『VEOS ガイド』)を参照してください。

モデルインプリメンテーションを VEOS Player にインポートする場合の自動化動作の変更

VEOS 3.5 では、VEOS Player API の `IProject <<Interface>>`インターフェースの `Import` メソッドについて、互換性のない変更が行われています。このメソッドにより、SIC や CTLGZ ファイルなどのモデルインプリメンテーションを VEOS Player プロジェクトにインポートすることができます。

- VEOS 3.4 以前のバージョンでは、`Import` メソッドは、`<System.String[]>`データ型のパラメータとして、すべてのコンパイラメッセージのリストを提供します。
- VEOS 3.5 以降、`Import` メソッドは、`IBuildResult <<Interface>>`を介してビルド情報を提供します。

互換性情報

項目の一覧

本章の内容

サポートしている MATLAB リリース

サポートしている MATLAB リリース

MATLAB のリリース	dSPACE Release 2015-B の各コンポーネントによるサポート				
	RCP and HIL Software	AutomationDesk 5.1 [1]	TargetLink 4.1	Model Compare 2.6	dSPACE Python Extensions 2.0 [2]
R2015b（64 ビット）	✓[3]	✓	✓	✓	✓
R2015b（32 ビット）	–	–	–	–	–
R2015a（64 ビット）	✓	✓	✓	✓	✓
R2015a（32 ビット）	–	–	–	–	–
R2014b（32 ビットと 64 ビット）	✓[3]	✓	✓	✓	✓
R2014a（32 ビットと 64 ビット）	✓[3]	✓	✓	✓	✓

[1] AutomationDesk の MATLAB Access ライブラリには MATLAB が必要です。
[2] dSPACE Python Extensions の matlablib2 には MATLAB が必要です。
[3] R2014a（32 ビット）、R2014b（32 ビット）、R2015b（32 ビット）および R2015b（64 ビット）は、RTI FPGA Programming Blockset – FPGA Interface ではサポートされません。

dSPACE ソフトウエアと組み合わせて使用可能なその他の MATLAB Release の最新情報については、http://www.dspace.jp/goto.cfm/jp_compati_RCP_HIL を参照してください。

オペレーティングシステム

ホスト PC のオペレーティングシステム

Release 2015-B の dSPACE 製品では、次のオペレーティングシステムをサポートしています。

32 ビットバージョン dSPACE ソフトウエア	64 ビットバージョン dSPACE ソフトウエア
■ Windows 7 Professional、Ultimate、Enterprise Service Pack 1（32 ビット版または 64 ビット版） 上記のエディションのみサポートされます。Windows 7 Home および Starter エディションはサポートされません。 ■ ControlDesk Next Generation は、MicroAutoBox Embedded PC にインストールすることもできます。バージョン別のオペレーティングシステムは次のとおりです。 ■ Intel® Atom™ Processor N270 を搭載した MicroAutoBox Embedded PC：Windows 7 Ultimate、32 ビット版 ■ Intel® Core™ i7-3517UE プロセッサを搭載した MicroAutoBox Embedded PC：Windows 7 Professional、Ultimate、Enterprise、64 ビット版	■ Windows 7 Professional、Ultimate、Enterprise Service Pack 1（64 ビット版） 上記のエディションのみサポートされます。Windows 7 Home および Starter エディションはサポートされません。
注意点と制限事項	
■ 「Windows 7 の場合の一般的な制限事項」(263 ページ)を参照してください。 ■ 64 ビットオペレーティングシステムのサポート：32 ビットバージョンの dSPACE ソフトウエアは、Windows 7 の 64 ビット版のみサポートします。その他の 64 ビット版のオペレーティングシステム（Windows XP および Windows Vista）はサポートされません。 32 ビットバージョンの dSPACE ソフトウエアは、Windows オペレーティングシステム（64 ビット版）の WoW64（Windows-on-Windows 64-bit）サブシステムで動作します。WoW64 は、Windows 64 ビット版で Windows 32 ビット版ベースのアプリケーションをシームレスに実行できるようにするための Windows の x86 エミュレータです。これにより、大きなメモリ領域を使用できるようアプリケーションが準備されている場合は、32 ビット版の各プロセスで最大 4 GB の仮想メモリを使用できるようになります。そうでない場合、プロセスの仮想アドレス空間は 2 GB に制限されます。 Windows（64 ビット版）で 32 ビットバージョンの dSPACE ソフトウエアを使用する場合は、制限事項が適用されます。「Windows（64 ビット版）で dSPACE ソフトウエア（32 ビットバージョン）を使用する場合の制限事項」(260 ページ)を参照してください。	■ 「Windows 7 の場合の一般的な制限事項」(263 ページ)を参照してください。

以下の製品の一部の複雑なタスクでは、Windows 7（64 ビット版）の使用が必須であるか推奨されます。

- RTI FPGA Programming Blockset：このブロックセットの FPGA インターフェースを使用する場合は、Windows 7（64 ビット版）が必須です。
- Automotive Simulation Models：ビークルダイナミクス、トレーラー、トラック、およびトラフィックシナリオのシミュレーション用モデルを使用する場合、Windows 7（64 ビット版）の使用をお勧めします。
- ControlDesk Next Generation：ControlDesk Next Generation のビデオキャプチャリングデバイスを使用する場合は、Windows 7（64 ビット版）の使用をお勧めします。
- ConfigurationDesk（Implementation Version）：ConfigurationDesk アプリケーションで 1000 以上のファンクションブロックを使用する場合（または複数のファンクションブロックのファンクションポートの合計数が 1000 を超える場合）は、Windows 7（64 ビット版）が必須です。

ファイアウォールルールを追加して通信を許可

各種 dSPACE ソフトウエア製品のインストール時には、Windows のファイアウォールルールが追加してインストールされます。たとえば、あるルールによって AutoBox などの dSPACE 拡張ボックスとの通信を行い、また他のルールによって MotionDesk でネットワークチャネルからモーションデータを受信します。これらのルールは、次のコマンドで生成されます。

- `netsh advfirewall firewall add rule name="dSPACE Net Service" service=any dir=in action=allow profile=any protocol=icmpv4:0, any description="Allow the dSPACE Net Service to connect to a dSPACE expansion box via network."`
- `netsh advfirewall firewall add rule name="dSPACE MotionDesk" program="%dspace_root%\MotionDesk\Bin\MotionDesk.exe" dir=in action=allow profile=any description="Allow dSPACE MotionDesk to receive motion data via network."`

ホスト PC でサードパーティ製ファイアウォールソフトウエアを実行している場合は、dSPACE ソフトウエアの TCP/IP 通信がブロックされないかどうか確認してください。

dSPACE License Server のオペレーティングシステム

フローティングネットワークライセンスを購入した場合は、ネットワーク接続されている PC の 1 台を dSPACE License Server としてインストールおよび設定する必要があります。

dSPACE License Server のオペレーティングシステムは、次のいずれかである必要があります。

- Windows XP Professional（32 ビット版）Service Pack 3
- Windows Vista Business、Ultimate、または Enterprise（32 ビット版または 64 ビット版）最新のサービスパック
- Windows 7 Professional、Ultimate、または Enterprise（32 または 64 ビット版）最新のサービスパック
- Windows Server 2003（32 ビット版または 64 ビット版）
- Windows Server 2008 R2
- Windows Server 2012、Windows Server 2012 R2

dSPACE License Server は Windows 以外のオペレーティングシステムをサポートしていません。

dSPACE ソフトウエアのランタイム互換性

定義

ランタイム互換性とは、以下のことを意味します。

- 別々のフォルダにインストールされている場合でも、複数の dSPACE 製品の同時使用が可能
- 相互作用なく個別に dSPACE 製品を使用可能

dSPACE Release 2015-B の製品互換性

dSPACE では、同一の dSPACE Release のソフトウエア製品のみ使用することをお勧めしています。これにより、ランタイム互換性を最大限に確保することができます。

次の点に注意してください。

- 異なる dSPACE Release の製品を併用した場合、dSPACE ツールチェーンでランタイム互換性に関連する制限が生じる可能性があります。

  dSPACE 製品が（自動化インターフェースを介して）直接連携する場合や、（A2L のような共通のファイルタイプを介して）間接的に連携する場合は、制限事項が適用されることがあります。詳細な制限事項については、該当する製品のマニュアルを参照してください。主要な制限事項については、次を参照してください。

  まれに、ランタイム互換を実行するために製品に追加のパッチをインストールする必要がある場合があります。パッチに関する情報およびパッチの必要性については、http://www.dspace.jp/go/CompPatch を参照してください。

■ Release 2015-B の RCP and HIL Software 製品は、それより前の dSPACE Release の RCP and HIL Software 製品と併用することはできません。

TargetLink および Model Compare に関する主要な制限事項 ビット互換のある MATLAB バージョン（32 ビットまたは 64 ビット）のみ使用可能であるため、64 ビットバージョンの TargetLink を 32 ビットバージョンの Model Compare と併用することはできません。また、32 ビットバージョンの TargetLink を 64 ビットバージョンの Model Compare と併用することはできません。

SCALEXIO システムの使用に関する主要な制限事項 SCALEXIO システムで使用する製品には互換性が必要です。同一の dSPACE Release で提供される製品でのみ互換性が保証されます。ご不明な点がございましたら、dSPACE にお問い合わせください。

以前のリリースの dSPACE 製品との併用	以前のリリースの複数の製品を併用する場合の詳細と注意事項については、http://www.dspace.jp/goto.cfm/ja_0501 を参照してください。

Windows（64 ビット版）で dSPACE ソフトウエア（32 ビットバージョン）を使用する場合の制限事項

目的	Windows（64 ビット版）で dSPACE ソフトウエア（32 ビットバージョン）を使用する場合は、さらにいくつかの制限事項が適用されます。
デバイスドライバの制限事項	メーカーから 64 ビット版ドライバが提供されている場合のみ、サードパーティ製バスインターフェース（CAN、LIN、または FlexRay）がサポートされます。
TargetLink：ターゲットコンパイラの制限事項	特定のターゲットコンパイラのサポート情報については、当該のコンパイラメーカーにお問い合わせください。
MATLAB	MATLAB の 32 ビットバージョンを Windows 7（64 ビット版）にインストールすると、MATLAB のインストールプログラムにより MATLAB の 64 ビットバージョンが提供されていることを示すメッセージが表示されます。MATLAB の 32 ビットバージョンをインストールするには、[OK]をクリックします。

64 ビット dSPACE DVD セットに含まれる製品とその MATLAB サポート

目的

64 ビット dSPACE DVD セットには、32 ビット dSPACE DVD セットと同じ製品バージョンが含まれています。ただし、一部の製品については、64 ビット DVD セットに 32 ビットバージョンが含まれています。

64 ビット DVD セットを使用する場合は、以下の表およびセクションに記載の制限事項に注意してください。

製品とそれぞれの MATLAB サポート

次の表に、64 ビット dSPACE DVD セットに含まれるすべての dSPACE 製品の一覧と、それぞれの MATLAB サポート状況を示します。

- MATLAB の 64 ビットバージョンをサポートする MATLAB を使用するすべての dSPACE 製品
- MATLAB の 64 ビットバージョンをサポートするすべての 32 ビットバージョンの dSPACE 製品
- MATLAB に関連しないすべての 32 ビットバージョンの dSPACE 製品

dSPACE 製品		MATLAB 64 ビットバージョンをサポートする製品	MATLAB アーキテクチャ（32 ビット／64 ビット）に依存しない製品	32 ビットバージョンが収められた製品
ControlDesk Next Generation		–	✓	✓
SystemDesk		–	✓	✓
AutomationDesk		✓	–	✓
TargetLink		✓	–	–
Model Compare		✓	–	–
VEOS		–	✓	✓
Real-time testing		–	–	✓
Platform API Package	dSPACE Python Extensions	–[1]	✓	✓
	HIL API .NET MAPort	–	✓[2]	✓
	XIL API .NET MAPort	–[3]	✓[3]	✓
Failure Simulation API Package	XIL API .NET EESPort	–	✓	✓

dSPACE 製品		MATLAB 64 ビットバージョンをサポートする製品	MATLAB アーキテクチャ(32 ビット／64 ビット)に依存しない製品	32 ビットバージョンが収められた製品
RCP and HIL Software パッケージ	RTI および RTI-MP	✓	–	–
	RTI Gigalink Blockset	✓	–	–
	RTI CAN Blockset	✓	–	–
	RTI CAN MultiMessage Blockset	✓	–	–
	RTI LIN MultiMessage Blockset	✓	–	–
	RTI FlexRay Configuration Blockset	✓	–	–
	RTI FPGA Programming Blockset	✓	–	–
	RTI Electric Motor Control Blockset	✓	–	–
	RTI Ethernet Blockset	✓	–	–
	RTI Ethernet UDP Blockset	✓	–	–
	RTI XCP on Ethernet Blockset	✓	–	–
	RTI Watchdog Blockset	✓	–	–
	RTI RapidPro Control Unit Blockset	✓	–	–
	RTI Bypass Blockset	✓	–	–
	RTI USB Flight Recorder Blockset	✓	–	–
	ConfigurationDesk	✓	–	✓
	FlexRay Configuration Blockset	✓	–	–
	FlexRay Configuration Tool	–	✓	✓
	ModelDesk	✓	–	✓
	Automotive Simulation Model	✓	–	–
	MotionDesk	✓	–	✓
	MotionDesk Blockset	✓	–	–
	Flight Rec Data Merger	–	✓	✓
	Model Interface Package for Simulink	✓	–	–

dSPACE 製品		MATLAB 64 ビットバージョンをサポートする製品	MATLAB アーキテクチャ(32 ビット／64 ビット)に依存しない製品	32 ビットバージョンが収められた製品
	RCP and HIL Software パッケージのその他の製品	–	✓	✓

[1] dSPACE Python Extensions には、matlablib2 Python ライブラリが含まれています。このライブラリは、64 ビット MATLAB のリモート制御とアクセスをサポートしています。matlablib2 は 64 ビット DVD に 32 ビットバージョンとして収められています。
[2] HIL API .NET MAPort は、MATLAB Interface for .NET を介して 32 ビット MATLAB から使用できますが、64 ビット MATLAB からは使用することができません。
[3] XIL API .NET MAPort は、MATLAB Interface for .NET を介して 32 ビットおよび 64 ビットの MATLAB から使用することができます。

dSPACE 製品と 64 ビット MATLAB バージョンとの互換性については、http://www.dspace.jp/go/matlab64bit を参照してください。

製品固有の制限事項

MAT ファイルのサポートの制限 ControlDesk Next Generation (ControlDesk 5.5)は、ファイルフォーマットバージョン 5.0 の MAT ファイルの読み書きのみサポートしています。このバージョンの MAT ファイルは、MATLAB で `save` コマンドの'-v6'オプションを使用して作成することができます。

RTI-MP `rtimpdiag` コマンドは機能しません。このコマンドは、64 ビットバージョンの MATLAB をサポートしていない dSPACE HIL API .NET をベースとしています。

TargetLink の 64 ビットバージョンの制限事項

A2L ファイルのインポート A2L ファイルは 64 ビットバージョンの TargetLink にインポートすることはできません。ただし、解決策は「Basics of Importing A2L Files」(📖『TargetLink Data Dictionary A2L Import and Export』)に記載されています。

Windows 7 の場合の一般的な制限事項

目的

Windows 7 と dSPACE ソフトウエアを組み合わせて使用する場合には、注意する必要がある事項が存在します。

MATLAB のサポート

MathWorks®社製ソフトウエアのシステム要件については、http://www.mathworks.com/support/sysreq/current_release を参照してください。

ユーザの簡易切り替えのサポートなし

dSPACE ソフトウエアは、Windows のユーザの簡易切り替えをサポートしません。

PC をシャットダウンする前に dSPACE ソフトウエアを閉じる	Windows オペレーティングシステムのシャットダウン手順では、いくつかの必要なプロセスが、dSPACE ソフトウエアによって利用されている状態であっても中断されることがあります。データの損失を回避するため、PC のシャットダウンを実行する前に dSPACE ソフトウエアを手動で終了することをお勧めします。
ユーザアカウント制御	dSPACE ソフトウエアをインストールするときは、Windows のユーザアカウント制御（UAC）を無効にすることをお勧めします。UAC を無効にできない場合は、Windows の次の動作に注意してください: UAC を有効にしていると、セットアッププログラムはユーザのアカウントではなく管理者アカウントで実行されます。そのため、管理者アカウントが必要なドライブ、特に必要なネットワークドライブへのアクセス権を持つことが重要です。
USB デバイス	光絶縁対応ケーブルを使用する dSPACE USB デバイスを初めて PC に接続したときに、デバイスドライバソフトウエアが正常にインストールされていないことを示すメッセージが表示される場合があります。ただし、dSPACE デバイスはその後正常に動作します。

数字

A

C

D

E

F

L

M

R

S

T