dSPACE Release

新機能と移行手順

Release 2016-B – 2016 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。

緊急の場合はお電話にて dSPACE にお問い合わせください。+49 5251 1638-941 (General Technical Support)

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。ソフトウェ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

内容

本書では、Release 2016-B に含まれるすべての dSPACE ソフトウェ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ェ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

本書で使用される表記規則

警告記号

本書では次の警告記号を使用します。

警告記号	説明
⚠ 危険	回避しないと死亡または重度の人身傷害につながる危険な状況を示します。
⚠ 警告	回避しないと死亡または重度の人身傷害につながる可能性がある危険な状況を示します。
⚠ 注意	回避しないと小規模または軽度の人身傷害につながる可能性がある危険な状況を示します。

警告記号	説明
注意	物的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回避しないと、物的損傷を招く可能性があります。
注記	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。
ヒント	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。

リンク先が用語解説のエントリを参照していることを示します。

特別なフォルダ

いくつかのソフトウェア製品では、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>
```

または

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

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 ソフトウェ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ェア製品およびそれにサポートされるハードウェ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 ソフトウェアの[Help]ボタンをクリックします。

注記

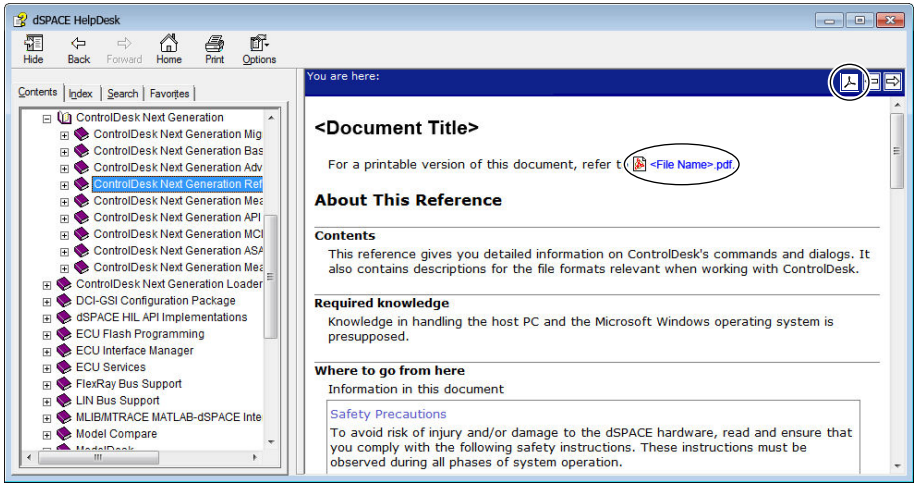
いくつかのソフトウェア製品では、状況依存ヘルプを使用することができません。

dSPACE ソフトウェアの[Help]メニュー　メニューバーから[Help] – [Contents]または[Help] – [Search]（すべてのソフトウェ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ェア製品およびそれにサポートされるハードウェ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 ドキュメントの先頭にある PDF リンクまたはトピックペインヘッダーの をクリックします。



dSPACE Release 2016-B の概要
Release 2016-B

目的	Release 2016-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。
64 ビット Python ディストリビューション	dSPACE Release 2016-B 以降、すべての製品は 64 ビットバージョンのみをサポートします。その結果、dSPACE Release 2016-B 以降では、64 ビットバージョンの Python ディストリビューションのみがサポートされます。

dSPACE Python ディストリビューションのコンポーネント Python 2.7 の 32 ビットバージョンと 64 ビットバージョンのコンポーネントバージョンは異なり、いくつかの新しいコンポーネントが使用可能です。

Python コンポーネント	32 ビットバージョン	64 ビットバージョン
Python Core	2.7.10	2.7.10
PyWin32	219.10	220.10
Numpy	1.7.1	1.11.0
Matplotlib	1.2.1	1.5.1
WxPython	2.9.4.0	3.0.2.0
Py2exe	0.6.9	0.6.9
Comtypes	0.6.2	1.1.2
PIL	1.1.7	–
Python for .NET	2.0p3	2.1.0
Cycler	–	0.10.0
Pillow	–	3.2.0
Pip	–	8.1.1
Pyparsing	–	2.1.1
Python_dateutil	–	2.5.3
Pytz	–	2016.4
Six	–	1.10.0

32 ビット Python と 64 ビット Python 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、PYC および PYW ファイルの Windows スタートメニューでのショートカットと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 64 ビットバージョンから 32 ビットバージョンに切り替えるには、以下の Python コンポーネントの **[Repair]** コマンドを実行する必要があります。
  - Python Core

    `python-2.7.10.msi` ファイルのコンテキストメニューで **[Repair]** をクリックします。

Control Panel\Programs\Programs and Features を使用して、Python 2.7.10 エントリの Repair を開始することもできます。

- PyWin32

  pywin32-219.10.win32-py2.7.exe を実行します。

dSPACE Release 2016-A 以前の dSPACE DVD の Disc1\Products\Common\Python2.7 に両方のファイルがあります。

- Python の 32 ビットバージョンから 64 ビットバージョンに切り替えるには、以下の Python コンポーネントの **[Repair]**コマンドを実行する必要があります。
  - Python Core

    python-2.7.10.amd64.msi ファイルのコンテキストメニューで **[Repair]**をクリックします。

    Control Panel\Programs\Programs and Features を使用して、Python 2.7.10 エントリの Repair を開始することもできます。

  - PyWin32

    pywin32-220.10.win-amd64-py2.7.exe を実行します。

  dSPACE Release 2016-B の dSPACE DVD の Disc1\Products\Common\Python2.7_x64 に両方のファイルがあります。

詳細については、「dSPACE 製品の 64 ビットの互換性に関する注意事項 」(277 ページ)を参照してください。

DVD の内容

dSPACE ソフトウェアは 2 枚の DVD で提供されます。DVD には、以下の dSPACE ソフトウェ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 5.3
  - ControlDesk 6.0
  - TargetLink 4.2
  - Model Compare 2.7

    注記

    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 4.7

- VEOS 3.7
- dSPACE ソフトウェ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 ソフトウェア製品が含まれるソフトウェアパッケージを指す総称です。

ヒント

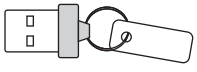
ディスク 2 には、その他の dSPACE ソフトウェア製品は収録されていません。

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ェアドングルが WibuKey ドングルから CmDongle 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

CmDongle



dSPACE Release 2014-B では、新しい CmDongle は、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 2016-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。ドライバソフトウェ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。

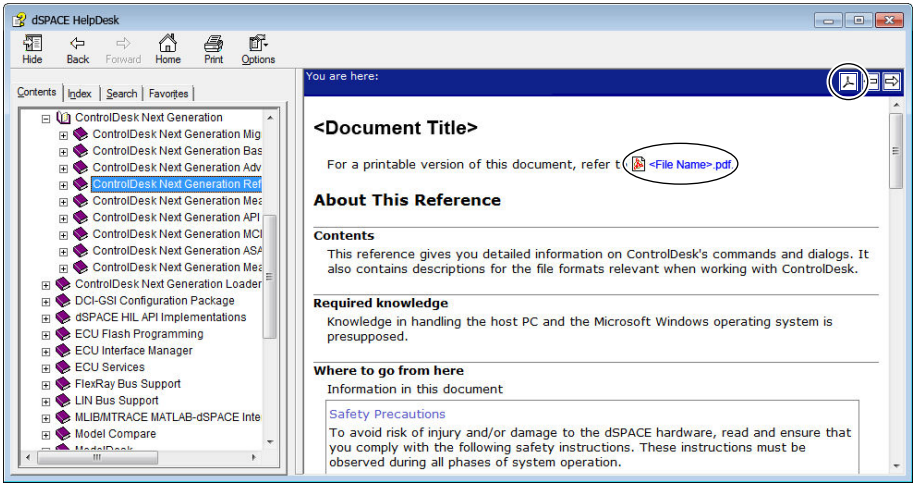
■ 新しい CmDongle で dSPACE Release 2014-A 以前のバージョンを使用する場合は、ご使用のホスト PC に dSPACE Installation Manager 3.8（以降）をインストールする必要があります。このバージョンには、新しいドングルのドライバが含まれています。dSPACE Installation Manager の最新のバージョンは http://www.dspace.jp/goto.cfm/IMUpdate_jp からダウンロードすることができます。

■ dSPACE Release 6.3 以前のバージョンでは、新しい CmDongle 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。

ユーザマニュアルの改善

dSPACE HelpDesk のドキュメントも PDF 形式で利用することができます。dSPACE Release 2016-A 以降では、これらの PDF ファイルにさらに迅速にアクセスできるようになりました。現在開いているトピックに関連する PDF がある場合は、トピックペインのヘッダーにある をクリックすると、そのファイルを開くことができます。

新たに公開されていないトピックを探す場合は、オンラインブックのフロントページに移動する必要があります。このページに関連する PDF ファイルへのハイパーリンクがあります。



dSPACE HelpDesk 使用時の制限事項

dSPACE HelpDesk は、リリースごとのフォルダにインストールされています。

dSPACE ソフトウェア製品に従って（現在、すべての製品が 64 ビットアプリケーションとして実装されています）、dSPACE Release 2016-B のユーザマニュアルが `C:\Program Files` にインストールされます。dSPACE Release 2016-A までは、ユーザマニュアルは `C:\Program Files(x86)` にインストールされます。

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 2016-B のセットアップごとに、該当するユーザマニュアルが dSPACE HelpDesk 2016-B にインストールされます。

  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 2016-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。

注記

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。

ソフトウェアサポートの廃止

プラットフォームアクセス用のテストオートメーションソフトウェアの廃止

プラットフォームアクセス用の以下のテストオートメーションソフトウェアは、dSPACE Release 2016-B で廃止されました。

- Test Automation Python Modules では、`rtplib2` Python モジュールが利用できなくなりました。

  `matlablib2` および `rs232lib2` Python モジュールは、利用することができます。

- HIL API MAPort の dSPACE Python インプリメンテーションおよび dSPACE .NET インプリメンテーションは利用できなくなりました。

テストオートメーションプロジェクトは、HIL API の後継として ASAM XIL API に移行することができます。HIL API .NET から XIL API .NET への移行には、アプリケーションのわずかな変更だけで済みます。「Migrating HIL API Applications to XIL API Applications」(🕮『dSPACE XIL API

Implementation Guide』)を参照してください。HIL API Python および rtplib2 から XIL API .NET への移行の場合は、Python スクリプトを使用することができます。移行の詳細については、テストオートメーションツールサポートセンター(http://www.dspace.jp/go/pscta)にお問い合わせください。

dSPACE Release 2016-B 以降でのプラットフォーム管理自動化 API バージョン 1.0 の廃止　プラットフォーム管理自動化 API バージョン 1.0 のサポートは終了しました。API バージョン 2.0 または 3.0 に移行する必要があります。このバージョンは、dSPACE XIL API .NET セットアップで利用することができます。

詳細については、「Basics on the Platform Management API」(📖『dSPACE Platform Management API Reference』)を参照してください。

AutomationDesk での DTS V7 のサポート　Remote Diagnostics (COM)ライブラリは、DTS V7 をサポートしません。移行については、「General Aspects on Migrating Discontinued Libraries in AutomationDesk」(📖『AutomationDesk Guide』)を参照してください。

AutomationDesk のその他の廃止項目については、「AutomationDesk 5.3 への移行」(52 ページ)を参照してください。

メッセージ処理方法の廃止　dSPACE Release 2016-B では、すべての dSPACE 製品でエラーや警告などのメッセージの取り扱いに改善された方法が使用されます。

その結果、メッセージは `dSPACE.log` ファイルに書き込めなくなりました。つまり、メッセージを ASCII テキストとして利用することはできません。

診断情報を収集し、dSPACE サポートに送信するには、dSPACE Installation Manager を使用する必要があります。

dSPACE ソフトウェアの廃止予定

VEOS および SystemDesk での PIL(Processor-in-the-Loop)シミュレーションの廃止　dSPACE Release 2017-A 以降では、VEOS および SystemDesk で PIL(Processor-in-the-Loop)シミュレーションがサポートされなくなります。これには、PIL シミュレーション向けの V-ECU や、評価ボードでのこれらの V-ECU のシミュレーションも含まれます。

ConfigurationDesk - Configuration Version での MPC5554 サポートの廃止　dSPACE Release 2017-A 以降、ConfigurationDesk - Configuration Version では MPC5554 マイクロコントローラはサポートされません。

MPC5554 は、スタンドアロンプロトタイピング ECU として使用される RapidPro システムの中核的なコンポーネントです。

dSPACE ハードウェアの廃止

DS1005 PPC Board　この製品は、2016 年 12 月で廃止されます。dSPACE ソフトウェアの新しいリリースでは、DS1005 のサポートを少なくとも 2019 年末までは継続します。

ただし、新規プロジェクトには、後継の dSPACE DS1007 PPC Processor Board を使用することをお勧めします。

DS1103 PPC Controller Board　この製品は、2016 年 12 月で廃止されます。dSPACE ソフトウェアの新しいリリースでは、DS1103 のサポートを少なくとも 2018 年末までは継続します。

ただし、新規プロジェクトには、後継の dSPACE MicroLabBox を使用することをお勧めします。

MicroAutoBox II 1401/1511/1512 および MicroAutoBox II 1401/1512/1513　DS1512 I/O Board を搭載した MicroAutoBox バージョンは、2016 年 12 月で廃止されます。dSPACE ソフトウェアの新しいリリースでは、これらの MicroAutoBox バージョンのサポートを少なくとも 2019 年末までは継続します。ただし、新規プロジェクトには、後継の MicroAutoBox II 1401/1511/1514 または MicroAutoBox II 1401/1513/1514 を使用することをお勧めします。

MicroAutoBox　MicroAutoBox（ボードリビジョン DS1401-19 以前）は、2013 年 12 月に廃止されました。ソフトウェアサポートは、dSPACE Release 2016-A で廃止されました。ただし、MicroAutoBox で使用された I/O ボードは、MicroAutoBox II で現在もサポートされています。

dSPACE ソフトウェアの新しいリリースでは、以下の MicroAutoBox II バージョンのサポートを少なくとも 2018 年末までは継続します。

- MicroAutoBox II 1401/1501
- MicroAutoBox II 1401/1504
- MicroAutoBox II 1401/1505/1507

新しいプロジェクトでは、後継の I/O ボードである DS1511、DS1513、および DS1514 を搭載した MicroAutoBox II バージョンを使用することをお勧めします。MicroAutoBox II 1401/1507 は現在も使用することができます。

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			
	2015-A	2015-B	2016-A	2016-B
AutomationDesk	5.0	5.1	5.2	5.3 「AutomationDesk」(49 ページ)を参照してください。
Automotive Simulation Model	8.0	8.1	8.2	8.3 「Automotive Simulation Model（ASM）」(57 ページ)を参照してください。
Bus Manager（スタンドアロン）	–	–	5.5	5.6 「Bus Manager（スタンドアロン）」(115 ページ)を参照してください。
ConfigurationDesk	5.3	5.4	5.5	5.6 「ConfigurationDesk」(117 ページ)を参照してください。
Container Manager	4.3	4.4	4.4	4.5
ControlDesk	5.4	5.5	5.6	6.0 「ControlDesk」(123 ページ)を参照してください。
DCI Configuration Tool	3.4	3.5	3.6	3.7 「DCI Configuration Tool」(147 ページ)を参照してください。
dSPACE CAN API Package	2.7.1	2.7.4	2.7.5	3.0 「dSPACE CAN API Package」(149 ページ)を参照してください。
dSPACE ECU Flash Programming Tool	2.2.6	2.2.6	2.3	2.3.1 「dSPACE ECU Flash Programming Tool」(151 ページ)を参照してください。
dSPACE FlexRay Configuration Package	3.5	3.6	3.7	3.8 「dSPACE FlexRay Configuration Package」(153 ページ)を参照してください。
dSPACE HIL API .NET	1.8	2.0	2.1	-
dSPACE Python Extensions	1.8	2.0	2.1	2.2 「dSPACE Python Extensions」(155 ページ)を参照してください。

製品名	dSPACE Release			
	2015-A	2015-B	2016-A	2016-B
dSPACE XIL API .NET	2015-A	2015-B	2016-A	2016-B 「dSPACE XIL API .NET」(157 ページ)を参照してください。
ECU Interface Manager	1.6	1.7	1.8	2.0 「ECU Interface Manager」(161 ページ)を参照してください。
Firmware Manager	1.3	2.0	2.1	2.2 「Firmware Manager」(163 ページ)を参照してください。
Model Compare	2.5	2.6	2.6	2.7 「Model Compare」(165 ページ)を参照してください。
ModelDesk	4.1	4.2	4.3	4.4 「ModelDesk」(167 ページ)を参照してください。
Model Interface Package for Simulink	3.0	3.1	3.2	3.3 「Model Interface Package for Simulink」(169 ページ)を参照してください。
MotionDesk	3.6	3.7	3.8	3.9 「MotionDesk」(173 ページ)を参照してください。
MotionDesk Blockset	2.3.2	2.4	2.4.1	2.5 「MotionDesk」(173 ページ)を参照してください。
Real-Time Testing	2.5	2.6	3.0	3.1 「Real-Time Testing」(175 ページ)を参照してください。
RTI[1)]	7.4	7.5	7.6	7.7 「RTI/RTI-MP および RTLib」(177 ページ)を参照してください。
RTI-MP[2)]	7.4	7.5	7.6	7.7 「RTI/RTI-MP および RTLib」(177 ページ)を参照してください。
RTI Bypass Blockset	3.4	3.5	3.6	3.7 「RTI Bypass Blockset」(181 ページ)を参照してください。
RTI CAN Blockset	3.4	3.4.1	3.4.2	3.4.3

製品名	dSPACE Release			
	2015-A	2015-B	2016-A	2016-B
RTI CAN MultiMessage Blockset	4.1	4.2	4.3	4.4 「RTI CAN MultiMessage Blockset」(183 ページ)を参照してください。
RTI Electric Motor Control Blockset	1.1	1.2	1.3	1.3.1
RTI Ethernet Blockset	1.2	1.2	1.2	1.2.1
RTI Ethernet (UDP) Blockset	1.4	1.4	1.4	1.4.1
RTI FPGA Programming Blockset	2.9	3.0	3.1	3.2 「RTI FPGA Programming Blockset」(185 ページ)を参照してください。
RTI LIN MultiMessage Blockset	2.5	2.5.1	2.6	2.7 「RTI LIN MultiMessage Blockset」(189 ページ)を参照してください。
RTI RapidPro Control Unit Blockset	2.2.1	2.2.1	2.2.1	2.2.2
RTI USB Flight Recorder Blockset	1.2	1.2	1.2	1.2.1
RTI Watchdog Blockset	1.0	1.0	1.0	2.0 「RTI Watchdog Blockset」(191 ページ)を参照してください。
SCALEXIO Firmware	3.2	3.3	3.4	3.5 「SCALEXIO Firmware」(193 ページ)を参照してください。
SYNECT Server	1.4.1	1.4.1	1.4.1	1.4.1
SystemDesk	4.4	4.5	4.6	4.7 「SystemDesk」(195 ページ)を参照してください。
TargetLink/TargetLink Data Dictionary	4.0	4.1	4.1	4.2 「TargetLink」(203 ページ)を参照してください。
Variable Editor	2.1	2.2	2.3	2.3[3)]
VEOS	3.4	3.5	3.6	3.7 「VEOS」(263 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) Variable Editor は、dSPACE Release DVD に含まれなくなりました。http://www.dspace.jp/go/requestreleasedownload で入手することができます。

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。

- ソフトウェアによる EESPort エラーをトリガする XIL API Convenience ライブラリの新しいオートメーションブロック
- ASAM MDF 4.1 ファイルから Signal データオブジェクトを作成する Evaluation ライブラリの新しいオートメーションブロック

	■ XML 形式で生成された結果ファイルのパスの設定などの COM API の拡張 ■ ユーザマニュアルの拡張 新機能の詳細については、「AutomationDesk 5.3 の新機能」(49 ページ)を参照してください。
Bus Manager（スタンドアロン）	Bus Manager（スタンドアロン）の主な新機能は次のとおりです。 ■ SAE J2602 規格のサポート ■ FIBEX 4.1.1 ファイルのサポート 詳細については、「Bus Manager（スタンドアロン）5.6 の機能」(115 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。 ■ バスシミュレーションコンテナファイル（BSC ファイル）を ConfigurationDesk アプリケーションに追加 ■ ECU インターフェースのサポート ■ ファンクションブロックの拡張：SENT In、SENT Out、CAN ■ Bus Manager： 　■ LIN 通信に関する SAE J2602 規格のサポート 　■ 通信マトリクスとして FIBEX 4.1.1 に基づく FIBEX ファイルのサポート 詳細については、「ConfigurationDesk – Implementation」(118 ページ)を参照してください。
ControlDesk	ControlDesk 6.0 の主な新機能は次のとおりです。 ■ 一般的な機能強化： 　■ 64 ビットアプリケーション 　■ ControlDesk の新しいパッケージ 　新機能の詳細については、「新しい一般機能（ControlDesk 6.0）」(124 ページ)を参照してください。 ■ プロジェクト／エクスペリメント管理の拡張： 　■ フォルダおよびフォルダグループのソート 　新機能の詳細については、「プロジェクトおよびエクスペリメントの新機能（ControlDesk 6.0）」(125 ページ)を参照してください。

- プラットフォーム／デバイスの拡張:
  - 新しい DCI-CAN/LIN1 のサポート
  - SCALEXIO プラットフォーム:Ethernet および PCI/PCIe カードの表示
  - SCALEXIO プラットフォーム:接続された未登録のシステムの表示
  - プラットフォーム／デバイスを追加する場合の重要な追加情報の表示

  新機能の詳細については、「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 6.0)」(125 ページ)を参照してください。
- 計器の拡張:
  - テーブルエディタの拡張
  - Instrument Selector の拡張
  - 新しい STRUCT 計器キャプションマクロ

  新機能の詳細については、「新しい計器機能(ControlDesk 6.0)」(127 ページ)を参照してください。
- 計測および記録の拡張:
  - 大きい ASAM MDF 4.1 ファイルの処理の改善
    - 信号のロードの改善
    - リダクションデータのサポート
  - 計測データファイルから変数を削除する

  新機能の詳細については、「新しい計測機能および記録機能(ControlDesk 6.0)」(131 ページ)を参照してください。
- メッセージ処理の拡張:
  - 新しい Message Viewer
  - dSPACE ログの表示

  新機能の詳細については、「メッセージ処理の新機能(ControlDesk 6.0)」(132 ページ)を参照してください。
- Bus Navigator の拡張:
  - バス計器(FlexRay 用の TX タイプ):最小遅延時間のサポート
  - バス信号のロード／アンロード

  新機能の詳細については、「Bus Navigator の新機能(ControlDesk 6.0)」(132 ページ)を参照してください。

- ECU 診断の拡張:
  - DoIP(Diagnostics over Internet Protocol)のサポート
  - デフォルトの COMPARAM 値が使用される場合の情報

  新機能の詳細については、「ECU 診断の新機能(ControlDesk 6.0)」(133 ページ)を参照してください。
- 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:
  - ソフトウェアトリガのサポート
  - 自動電位マッピング
  - 自動シグナルマッピング

  新機能の詳細については、「新しい電気的欠陥シミュレーション機能(ControlDesk 6.0)」(134 ページ)を参照してください。
- Signal Editor の拡張:
  - eLINEAR 補間メソッドのサポート
  - ダウンロード中の Data File セグメント表示の自動更新

  新機能の詳細については、「Signal Editor の新機能(ControlDesk 6.0)」(135 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
- ECU メモリアクセス

新機能の詳細については、「DCI Configuration Tool 3.7 の新機能」(147 ページ)を参照してください。

dSPACE CAN API Package

dSPACE CAN API Package の主な新機能は、次のとおりです。

- CAN FD(新しい dSPACE CAN API 2.0 でのみ提供)のサポート
- DCI-CAN/LIN1 インターフェース(dSPACE CAN API 1.0 および新しい dSPACE CAN API 2.0 で提供)のサポート

新機能の詳細については、「dSPACE CAN API Package 3.0 の新機能」(149 ページ)を参照してください。

dSPACE ECU Flash Programming Tool

dSPACE ECU Flash Programming Tool の主な新機能は、次のとおりです。

- DCI-CAN/LIN1 インターフェースのサポート
- バージョン 2.3 より前のツールからそのまま XCP on CAN フラッシュプロジェクトをロード可能

新機能の詳細については、「dSPACE ECU Flash Programming Tool 2.3.1 の新機能」(151 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。 ■ フレームワークインプリメンテーションの初期サポート ■ ソフトウェアによるエラーのトリガのための EESPort の機能拡張 新機能の詳細については、「dSPACE XIL API .NET 2016-B の新機能」(157 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。 ■ バックステージビューのヘルプリボンからの『新機能と移行手順』ドキュメントおよび PDF ファイルへの直接アクセス 新機能の詳細については、「Firmware Manager 2.2 の新機能」(163 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。 ■ Windows 10 の正規フルサポート ■ 共通祖先モデルによる三元分析 ■ バージョン管理システムとの連携の改善 新機能の詳細については、「Model Compare 2.7 の新機能」(165 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。 ■ 運転操作およびドライビングサイクルの開始、停止、およびリセット ■ 新しい時間プロッタと XY プロッタ ■ 新しいテーブルエディタ ■ 道路を OpenDRIVE 形式でエクスポート可能 新機能の詳細については、「ModelDesk 4.4 の新機能」(167 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。 ■ リアルタイムのレンズ歪みのシミュレーション ■ 道路生成の精度の指定 ■ 鮮明なテクスチャの設定 新機能の詳細については、「MotionDesk 3.9 の新機能」(173 ページ)を参照してください。

Python Extensions	Python Extensions に含まれる製品に新機能はありませんが、以下の廃止項目があります。 ■ dSPACE HIL API Python Implementation ■ Test Automation Python モジュールの rtplib2 移行の詳細については、「dSPACE Python Extensions 2.2 への移行」(156 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。 ■ Python 2.7.11 に基づくリアルタイム Python インタープリタ ■ `rttlib.hostcalls` モジュールは、Real-Time Testing でサポートされているすべてのプラットフォームをサポートします。 ■ `rttlib.datastream` モジュールは、再生用の新しいモードをサポートします。 新機能の詳細については、「Real-Time Testing 3.1 の新機能」(175 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 R2016b のサポート ■ M ファイルでのタスクコンフィギュレーションをコーディングするための API 新機能の詳細については、「RTI/RTI-MP および RTLib の新機能」(177 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。 ■ オンターゲットバイパス処理の TargetLink によるサポート 新機能の詳細については、「RTI Bypass Blockset 3.7 の新機能」(181 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。 ■ FIBEX 4.1.1 のサポート 新機能の詳細については、「RTI CAN MultiMessage Blockset 4.4 の新機能」(183 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®ソフトウェアのサポートの拡張 ■ DS2655 FPGA Base Board および I/O モジュール向けの FPGA フレームワークの拡張 ■ MicroLabBox 用 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 3.2 の新機能」(185 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。 ■ FIBEX 4.1.1 のサポート 新機能の詳細については、「RTI LIN MultiMessage Blockset 2.7 の新機能」(189 ページ)を参照してください。
RTI Watchdog Blockset	RTI Watchdog Blockset は、Challenge-Response Monitoring ブロックセットにより拡張されています。 新機能の詳細については、「RTI Watchdog Blockset 2.0 の新機能」(191 ページ)を参照してください。
SCALEXIO Firmware	SCALEXIO ファームウェアは、新しい SCALEXIO_RTLIB_MC8 および SCALEXIO_FIU_500 ライセンスをサポートしています。 新機能の詳細については、「SCALEXIO Firmware 3.5 の新機能」(193 ページ)を参照してください。
SystemDesk	SystemDesk 4.7 の主な新機能は次のとおりです。 ■ SystemDesk では、すべてのベンダーの基本ソフトウェアでの ECU 設定に使用できる *ECU コンフィギュレーションフレームワーク*を提供します。 新機能の詳細については、「新しい一般機能」(196 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。 ■ オペレーティングシステム ■ Windows 10 の正規フルサポート AUTOSAR ■ Revision 4.2.2 のサポート ■ 非同期クライアントサーバ通信 ■ クライアントサーバ通信でのデータトランスフォーマ ■ 非スカラーインターランナブルバリアブル ■ SystemDesk でのラウンドトリップの改善 ■ ARXML インポートおよびエクスポートの改善

- Stateflow
  - Superstep セマンティクスのサポート
  - 複数の出力データをもつ（グラフィカル）関数のサポート
  - 状態アクティビティ監視（Stateflow Active State データ）のサポートの改善
- ブロック固有の改善
  - エレメント選択および Data Store Memory ブロックでのバス信号のサポート
  - Vector Concatenate ブロックのサポート
  - モデルルートレベルでの Enable ブロックのサポート（参照先モデルの実行の制御）
- Data Dictionary
  - Windows コマンドプロンプトから起動する DD Comparison ペインを使用した Data Dictionary の比較のサポート
  - カスタムコマンド定義および呼び出しのサポート
- オンターゲットバイパス処理用のフルコード生成を含む、dSPACE の RTI Bypass ブロックを使用したオンターゲットバイパス処理
- 生成されたコードでの Simulink 列挙型（Stateflow を含む）のサポートの改善
- カスタムコードファイルでのユーザ定義文字列のインスタンス固有の置換
- 生成された量産コードおよび固定小数点ライブラリでの MISRA C 準拠の改善

すべての新機能の詳細については、「TargetLink 4.2 および TargetLink Data Dictionary 4.2 の新機能」(204 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 4.2 および TargetLink Data Dictionary 4.2 への移行」(229 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。

- VPU ポートの階層表示
- 構造エレメントのドラッグによる VPU ポートの接続
- FMU 内部メッセージのログの有効化／無効化
- 自動化を介した VEOS Player のバージョン情報の取得

新機能の詳細については、「VEOS 3.7 の新機能」(263 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。

dSPACE Release 2016-B への移行 Release 2016-B

目的	Release 2016-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 2016-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 2015-B 以前のリリースからの移行	dSPACE Release 2015-B 以前のリリースから Release 2016-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 2016-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\Common Files\dSPACE\HelpDesk 2016-B\Print\PreviousReleases` を参照してください。
- dSPACE DVD の、`\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 ソフトウェ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。

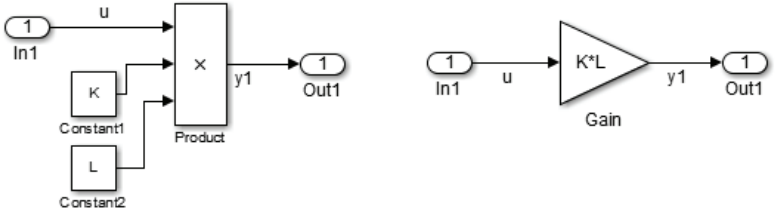
- パフォーマンスが大幅に向上

  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 以降：[Default parameter behavior]を[Inlined]）に設定した使用に制限されていました。MATLAB R2015b 以降を使用する場合、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 などの dSPACE ソフトウェ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 以降を併用することにより可能になります。dSPACE Release 2015-B 以降を以前の MATLAB バージョンで使用する場合、コード生成は以前の dSPACE リリースとほぼ同じです。

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。

最新の dSPACE リリースを使用する場合の動作の違いについては、次の表を参照してください。

dSPACE Release 2016-B で使用する MATLAB リリース			
R2015a	**R2015b**	**R2016a**	**R2016b**
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。 その他のすべてのブロックパラメータの定義は、以前の MATLAB リリース R2013b または R2014a の場合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		
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 以降では、この設定は[Default parameter behavior]オプションを `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、次の変更が導入されました。

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ェアでの変更の移行」(43 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ェアでの変更の移行」(43 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ェアでの変更の移行」(43 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ェアでの変更の移行」(43 ページ)を参照してください。

n-D Lookup Table の処理　dSPACE Release 2015-B 以降では、4x3x2 行列のような 2 次元を超える Look-Up 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ェアでの変更の移行」(43 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ェアでの変更の移行」(43 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]`

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。

注記

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ェ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ェアでの変更の移行

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 R2015a SP1 と dSPACE Release 2016-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 R2015b と dSPACE Release 2016-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ェア、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。
MATLAB R2016a と dSPACE Release 2016-B を使用	注意事項は MATLAB R2015b の場合と同じです。
MATLAB R2016b と dSPACE Release 2016-B を使用	注意事項は MATLAB R2015b の場合と同じです。
TRC ファイルを生成するソフトウェ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 以降を使用する必要はありません。それ以前の dSPACE Release 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ェアでの変更の移行

目的

ControlDesk 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、ソフトウェ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、ソフトウェ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ェ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 以降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。

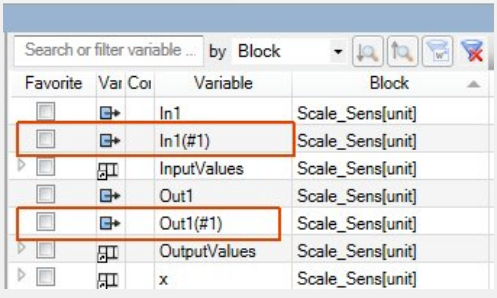
ControlDesk を使用している場合は、特別なマークが付いた計器や、Signal Editor の `Check Mapping` コマンドなどを利用して、不整合のある接続を見つけることができます。AutomationDesk では、変数アクセスは変数エイリアスを介して行われます。そのため、変数記述ファイルの修正を自動認識することはできません。ただし、プロジェクトで変数プールを使用している場合は、これを更新するだけで十分です。

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。

ソフトウェアの特定の変更を自動的に移行できない場合は、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。

問題	移行手順
非仮想 Simulink バスの変数パスの更新	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。 **注記** 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 Tunable Parameters グループなどのテーブルデータを参照している。この場合、ControlDesk はこのテーブルデータを参照する最初のルックアップテーブルのみを認識します。その他のすべてのルックアップテーブルは認識されません。 ■ 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）。 **注記** ■ ControlDesk は、テーブルデータまたはルックアップテーブルの軸によって参照される変数の *複製変数* を作成します。元の変数の名前に対して、複製変数の名前には `(#1)` が付加されています。これらの複製変数を ControlDesk 計器に接続しないでください。 下図に複製変数の例を示します。  ■ Tunable Parameters グループの値ブロックがテーブルデータまたはルックアップテーブルの軸によって参照されている場合、ControlDesk はこの値ブロックの間違った変数タイプを使用します。*value block* 変数タイプの代わりに、次の変数タイプのいずれかが使用されます。 ■ Map ■ Curve ■ Common axis

注記

- MATLAB R2015b 以降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、R2015b 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。
- 変更された TRC ファイル生成に関する制限事項については、「SDF ファイルの制限事項」(🕮『ControlDesk 変数管理』)を参照してください。

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」(37 ページ)を参照してください。

AutomationDesk

項目の一覧

本章の内容

AutomationDesk 5.3 の新機能

本章の内容

一般的な機能強化

ユーザマニュアルの改善 『*AutomationDesk チュートリアル*』には、Signal-Based Testing ライブラリの使用方法に関する新しいレッスンが含まれています。

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。

XIL API Convenience ライブラリ XIL API Convenience ライブラリでは、ライブラリの Software-Triggered Errors フォルダで、以下の新しいオートメーションブロックを利用することができます。

- PrepareMultiPinErrorWithCondition

  このブロックは、複数のピンに影響を及ぼし、指定された条件でトリガされるエラーセットを作成するために使用されます。

- PrepareMultiPinErrorWithDuration

  このブロックは、複数のピンに影響を及ぼし、指定された期間の後でトリガされるエラーセットを作成するために使用されます。

- PrepareSinglePinErrorWithCondition

  このブロックは、単一のピンに影響を及ぼし、指定された条件でトリガされるエラーセットを作成するために使用されます。

- PrepareSinglePinErrorWithDuration

  このブロックは、単一のピンに影響を及ぼし、指定された期間の後でトリガされるエラーセットを作成するために使用されます。

- WaitForTrigger

  このブロックは、次のエラーセットが指定された EES ポートに対してトリガされるまで待機するために使用されます。

オートメーションブロック PrepareSinglePinError、PrepareMultiPinError および Trigger は、Manually Triggered Errors ライブラリフォルダに移動しました。

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

評価ライブラリ Evaluation ライブラリでは、Converters ライブラリフォルダで新しいオートメーションブロックを利用することができます。

- CreateSignalFromMDFFile

  このブロックは、ASAM MDF 4.1(MF4)ファイルに含まれる指定されたチャンネルから評価信号を取得するために使用されます。

詳細については、「Evaluation」(🕮『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。

- 結果の *ReportPool.xml* ファイルのパスを保持するために Result1 オブジェクトに Path プロパティが追加されています。

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

プロジェクト操作の強化

AutomationDesk のプロジェクト操作では、以下の拡張を利用することができます。

- Found Items Viewer では、Tuple、List および Dictionary データオブジェクトにある一致する表現を置き換えることができます。
- [Find]コマンドのダイアログでは、以下の設定の名前が変更されています。
  - [Found in projects, folders, sequences, and blocks]は、[Found elements except data objects]に置き換えられています。
  - [Source code of Exec, ExecFile, and Eval blocks]は、[Python source code]に置き換えられています。
- [Find Inconsistencies]コマンドでは、`self` を介して参照されるオブジェクトは考慮されません。データオブジェクトは常に`_AD_`を介して参照されます。
- `_INFO_`ネームスペースでは、現在操作している例外のメッセージ文字列を使用することができます。詳細については、「Getting Element Information」(📖『AutomationDesk Guide』)を参照してください。

ライブラリおよびブロックの廃止

AutomationDesk 5.3 では、次のライブラリが廃止されました。

- Platform Access

  Platform Access ライブラリブロックを実行する場合、dSPACE Release 2016-B 以降で `rtplib2` Python モジュールが使用できないため、エラーメッセージが表示されます。
- Test Framework

AutomationDesk 5.3 では、次のブロックが廃止されました。

- ControlDesk Access ライブラリの欠陥シミュレーション用のすべてのオートメーションブロック

  これらの欠陥シミュレーションブロックの 1 つを実行する場合、dSPACE Release 2016-B 以降で ControlDesk の対応する機能が使用できないため、エラーメッセージが表示されます。
- 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

移行方法については、「AutomationDesk 5.3 への移行」(52 ページ)を参照してください。

廃止された一般機能

Platform Management ライブラリ　プラットフォーム管理自動化 API バージョン 1.0 のサポートは終了しました。詳細については、「GetPlatformManagement」(📖『AutomationDesk Library Reference』)を参照してください。

Remote Calibration（COM）ライブラリ　INCA または CANape などの MCD 3MC サーバの一部では、内部データストレージの使用時にデータ転送のパフォーマンスを向上させるために、FetchResults および FetchMeasurement オートメーションブロックに対してマーシャリングが使用されます。マーシャリングでは、クライアントおよびサーバが同じビットアーキテクチャを使用している必要があります。AutomationDesk 5.3 では、マーシャラも 64 ビットプロセスで実行されている場合のみ、マーシャリングがサポートされます。

32 ビット MCD 3MC サーバを AutomationDesk で使用するには、マーシャリングを必要としない `eST_FILE` をストレージタイプとして設定する必要があります。

XIL API ライブラリおよび XIL API Convenience ライブラリ　XIL API ライブラリおよび XIL API Convenience ライブラリでは、HIL API 1.0.2 規格に基づくサーバはサポートされません。

サードパーティ製 Python モジュール　32 ビットのサードパーティ製 Python モジュールは、AutomationDesk 5.3 の 64 ビット Python インタープリタではサポートされません。

AutomationDesk 5.3 への移行

移行に関する一般的な注意点

新しいバージョンの AutomationDesk で AutomationDesk プロジェクトを開くと、移行が必要かどうかがソフトウェアによって自動的に検出されます。メッセージダイアログで[OK]をクリックすると、移行が開始されます。AutomationDesk のバージョンには下位互換性がないため、古いプロジェクトで作業を引き続き行う場合には、移行後のプロジェクトで古いプロジェクトを上書きしないようにする必要があります。移行後のプロジェクトは別のパスや名前で保存してください。

注記

AutomationDesk の新しいバージョンで以前のプロジェクトを開く場合、あらかじめ以下の条件を満たしておく必要があります。

- プロジェクトおよびリンクされたカスタムライブラリのバックアップを作成しておく必要があります。
- AutomationDesk が正常に実行されている必要があります。Log Viewer にエラーメッセージが 1 つも表示されない状態になっている必要があります。
- 組込みライブラリ、必要なカスタムライブラリ、およびその他のパッケージが正常にロードされている必要があります。
- 以前のプロジェクトを新しい AutomationDesk バージョンにインポートするには、エクスポートされたプロジェクトまたはカスタムライブラリが ZIP 形式で使用可能である必要があります。自動移行では、XML 形式はサポートされません。

バージョン管理システムを使用している場合、移行を正常に行うにはいくつかの前提条件を満たしている必要があります(「How to Migrate Projects Under Version Control」(🕮『AutomationDesk Guide』)を参照)。

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

ライブラリ

ControlDesk NG Access ライブラリ ControlDesk NG Access ライブラリの名前が、ControlDesk Access に変更されています。AutomationDesk プロジェクトは、変更された名前に自動的に調整されます。

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

XIL API Convenience ライブラリ 電気的欠陥シミュレーション用のオートメーションブロックは、Manually-Triggered Errors フォルダに移動しました。この場合、既存のプロジェクトには影響しません。

Remote Diagnostics(COM)ライブラリ *DTS V7* の診断システム用インターフェースとしての使用は廃止されました。ControlDesk を診断システムとして使用する必要がある場合は、ControlDeskNG.D3System202 を System データオブジェクトの Interface パラメータとして指定して ControlDesk の ECU 診断デバイスへのアクセスを実現することができます。

別の方法としては、AutomationDesk で ControlDesk Access ライブラリを介して ECU 診断デバイスをリモート制御することができます。詳細については、「ControlDesk Access」(🕮『AutomationDesk Library Reference』)を参照してください。

ControlDesk の ECU 診断デバイスの詳細については、🕮『ControlDesk ECU 診断』を参照してください。。

ControlDesk への移行時に問題が発生する場合は、dSPACE サポートにお問い合わせください。

ライブラリおよびブロックの廃止

AutomationDesk 5.3 で廃止されたエレメントを含むプロジェクトを開くと、プロジェクトの自動更新時に、廃止されたデータオブジェクトは Discontinued data object データオブジェクトに置き換えられ、廃止されたブロックは Discontinued block ブロックに置き換えられます。これにより、プロジェクトをロードして、移行するブロックとデータオブジェクトを検索することができます。AutomationDesk または API スクリプトを使用して、廃止されたライブラリのエレメントを含むプロジェクトを実行すると、例外により実行は停止します。

この変更は、次のブロックおよびデータオブジェクトが影響を受けます。

Test Framework　ライブラリ全体が廃止されました。後継バージョンの Test Builder ライブラリを使用するには、プロジェクトを移行する必要があります。

この移行には、dSPACE サポートにより提供されるツールを使用することができます。詳細については、http://www.dspace.jp/go/TestBuilderMigration を参照してください。

Platform Access　ライブラリ全体が廃止されました。XIL API Convenience ライブラリを使用するにはプロジェクトを移行する必要があります。

Platform Access ライブラリに固有のデータオブジェクトのタイプは、現在も継続するデータオブジェクトに自動的に変換されます。

Platform Access ライブラリのオートメーションブロックは、XIL API Convenience ライブラリブロックと一対一で置き換えることができません。移行には、各ブロックが関与する作業を考慮する必要があります。詳細については。「Migrating Blocks of the Platform Access Library」(🕮『AutomationDesk Guide』)を参照してください。

ControlDesk Access　ControlDesk Access ライブラリの欠陥シミュレーション機能は廃止されました。プロジェクトまたはカスタムライブラリに、このライブラリの Convenience - FailureSimulation および Basic Functions - Application - FailureSimulation フォルダのエレメントが含まれている場合は、それらの要素を置き換える必要があります。

AutomationDesk 5.0 では、ASAM XIL API 規格の電気的欠陥シミュレーションポート(EESPort)が、XIL API ライブラリと XIL API Convenience

ライブラリによってサポートされています。XIL API Convenience ライブラリの EESPort ブロックにより、欠陥シミュレーションハードウェアに将来にわたってアクセスすることができます。XIL API 電気的欠陥シミュレーションの詳細については、「Simulating Electrical Errors via the XIL API Convenience Library」(🕮『AutomationDesk Guide』)を参照してください。

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

評価ライブラリ　Platform Access ライブラリのブロックを介してキャプチャされる結果の変換は廃止されました。

影響を受けるブロックは、次のとおりです。

- GetSignalFromCaptureResult

CaptureResult データオブジェクトは、Dictionary データオブジェクトに置き換えられました。移行したキャプチャ結果から信号を取得するには、Exec ブロックで以下のスクリプトを使用することができます。

```
_AD_.Signal = evaluationlibrary.Signal(
   _AD_.CaptureResult[u'xAxis'],
   _AD_.CaptureResult[_AD_.SignalName])
```

XIL API ライブラリ　IDF ファイルへのアクセスを初期化するブロックは廃止されました。

影響を受けるブロックは、次のとおりです。

- InitCaptureResultIDFReader
- InitCaptureResultIDFWriter

MDF ファイルにアクセスするには、これらのブロックを移行する必要があります。詳細については、以下を参照してください。

移行上の注意点については、「Migrating AutomationDesk」(🕮『AutomationDesk Guide』)を参照してください。

IDF ファイルへのアクセスを初期化する廃止された XIL API ライブラリブロックの移行

XIL API を使用する場合、取得した結果を保存するファイル形式(推奨)には、ASAM Measured Data Format(MDF)を使用します。AutomationDesk では、ファイル拡張子 `MF4` で示される 4.1 フォーマットが使用されます。このため、IDF ファイルにアクセスするリーダーとライターを初期化するためのブロックは廃止されました。

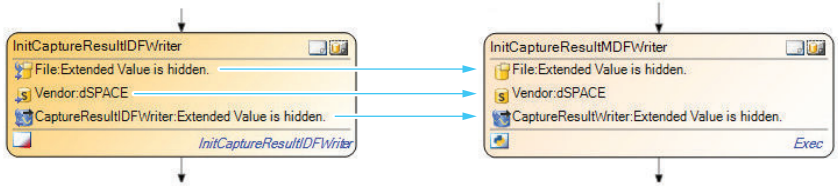
InitCaptureResultIDFWriter ブロックの移行　それぞれの InitCaptureResultIDFWriter ブロックを、File、String、および CaptureResultReader データオブジェクトをパラメータとして提供する Exec ブロックに置き換えます。

文字列データオブジェクト `Vendor` の名前を変更し、値を `dSPACE` に設定するか、またはベンダー情報を含むグローバル文字列データオブジェクトを参照します。

Exec ブロックの Python ソースとして、次のように入力します。

```
_AD_.CaptureResultWriter = xilapi.InitCaptureResultWriter(
    _AD_.Vendor,                    # vendor = "dSPACE"
    "CaptureResultMDFWriter",       # CaptureResultWriterType
    _AD_.File.GetAbsolutePath() )   # the instance-specific
                                    #   capture result MDF file
                                    #   with the file extension MF4
```

InitCaptureResultIDFWriter ブロックと同じように、Exec ブロックのパラメータを設定します。



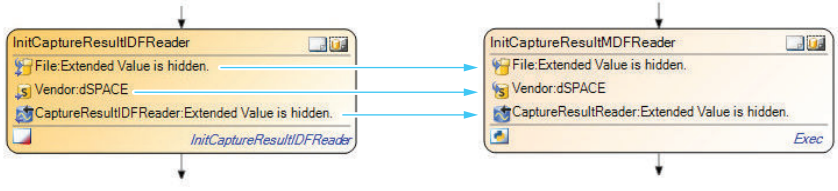
InitCaptureResultIDFReader ブロックの移行　それぞれの InitCaptureResultIDFReader ブロックを、File データオブジェクト、String データオブジェクト、および CaptureResultReader データオブジェクトをパラメータとして提供する Exec ブロックに置き換えます。

文字列データオブジェクト Vendor の名前を変更し、値を dSPACE に設定するか、またはベンダー情報を含むグローバル文字列データオブジェクトを参照します。

Exec ブロックの Python ソースとして、次のように入力します。

```
_AD_.CaptureResultReader = xilapi.InitCaptureResultReader(
    _AD_.Vendor,                    # vendor = "dSPACE"
    "CaptureResultMDFReader",       # CaptureResultReaderType
    _AD_.File.GetAbsolutePath() )   # the instance-specific
                                    #   capture result MDF file
                                    #   with the file extension MF4
```

InitCaptureResultIDFReader ブロックと同じように、Exec ブロックのパラメータを設定します。



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。

すべての ASM ブロックセット

すべての ASM ブロックセットの移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準の Simulink Lookup Table (n-D)ブロックに更新されています。

各 ASM ブロックセットの更新されたブロックのリストは、それぞれの ASM ブロックセットの移行情報を参照してください。

ルックアップテーブル設定の移行は、MATLAB での推奨移行手順と同時に実行されます。ただし、MATLAB R2011a のリリースノートに記載されているように、ブレークポイントが反復する場合には特別な処理が必要です。ブレークポイントの反復は、以前の Look-Up Table ブロックでは禁止されていませんでした。ASM の移行では、提供される ASM デモモデルのパラメータセットについてのみ、ブレークポイント反復の問題が処理されます。これらのブロックでは特別な移行処理が行われます。ブレークポイントが反復するカスタムパラメータ設定は、この移行手順では処理されません。

これらの場合でも期待されるシミュレーション動作を保証するには、手作業での移行が必要となります。ControlDesk 接続の再確立に役立つ Python スクリプトに関する詳細は、以下を参照してください。
http://www.dspace.jp/go/ASM-LUT-CD-migration-RLS2016-B

ASM_UTILS ライセンスの廃止

ASM_UTILS ライセンスは廃止されました。ライセンスチェックは、その他の ASM ライセンスで使用することができます。各 ASM ブロックセットの更新されたブロックのリストは、それぞれの ASM ブロックセットの移行情報を参照してください。

ASM Base InCylinder Blockset

項目の一覧

本章の内容

ASM Base InCylinder Blockset 2.3 の新機能

ENGINE_TORQUE_SET_INTERVENTION

この新しいブロックは、ギアシフトプロセス中のエンジントルクの介入を容易にします。

ASM Base InCylinder Blockset 2.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- DIRECTINJECTOR_TIMING
- DIRECTINJECTOR
- EGR_COOLER、EGR_COOLER_1_0
- EGR_RATE_CONTROL
- EGR_VALVE
- ENGINE_TORQUE_SET
- EXHAUST_MANIFOLD
- EXHAUST_VALVE
- FRICTION_TORQUE
- HIGH_PRESSURE_PUMP
- INJECTION_QUANTITY

	■ INTAKE_MANIFOLD ■ INTAKE_MANIFOLD_BOOST ■ INTAKE_VALVE ■ INTERCOOLER、INTERCOOLER_1_0 ■ PRESSURE_CONTROL_VALVE ■ RAIL ■ RAIL_CONTROL ■ THROTTLE_CONTROL ■ THROTTLE_MECHANICAL ■ THROTTLE_VALVE、THROTTLE_VALVE_2_0 ■ VVT_SETPOINT ■ WALL_HEAT
ASM Utils ブロックのライセンスチェック	ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。 このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ ANALYTIC_THERMODYNAMIC_ANALYSIS ■ APU_EVENT ■ EGR_COOLER、EGR_COOLER_1_0 ■ ENGINE_SETUP ■ EXHAUST_MANIFOLD ■ INTAKE_MANIFOLD ■ INTERCOOLER、INTERCOOLER_1_0 ■ MASS_FRACTION_BURNED ■ RAIL ■ SINGLEZONE_CYLINDER、SINGLEZONE_CYLINDER_3_0
SOFTECU_SETUP	Const_CylinderOffset 入力ポートの単位が[]から[deg]に修正されています。 他の入力ポートの名前に含まれていたアンダースコアは削除されました。
IDLE_SPEED_CONTROL	移行時に、Trq_Engine_Ind_Max[Nm]出力ポートは終了します。

ASM Brake Hydraulics Blockset

ASM Brake Hydraulics Blockset 1.6.1 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- BRAKE_CYLINDER_FL
- BRAKE_CYLINDER_FR
- BRAKE_CYLINDER_RL
- BRAKE_CYLINDER_RR
- CONTINUOUS_CTRL_CHANGE_OVER_VALVE_1
- CONTINUOUS_CTRL_CHANGE_OVER_VALVE_2
- CONTINUOUS_CTRL_INLET_VALVE_FL
- CONTINUOUS_CTRL_INLET_VALVE_FR
- CONTINUOUS_CTRL_INLET_VALVE_RL
- CONTINUOUS_CTRL_INLET_VALVE_RR
- CONTINUOUS_CTRL_OUTLET_VALVE_FL
- CONTINUOUS_CTRL_OUTLET_VALVE_FR
- CONTINUOUS_CTRL_OUTLET_VALVE_RL
- CONTINUOUS_CTRL_OUTLET_VALVE_RR
- CONTINUOUS_CTRL_PRE_CHARGE_VALVE_1
- CONTINUOUS_CTRL_PRE_CHARGE_VALVE_2
- CONTINUOUS_VALVE_CONTROL
- CHANGE_OVER_VALVE_1_TABLE
- CHANGE_OVER_VALVE_2_TABLE
- INLET_VALVE_FL_TABLE
- INLET_VALVE_FR_TABLE

- INLET_VALVE_RL_TABLE
- INLET_VALVE_RR_TABLE
- MASTER_BRAKE_CYLINDER
- OUTLET_VALVE_FL_TABLE
- OUTLET_VALVE_FR_TABLE
- OUTLET_VALVE_RL_TABLE
- OUTLET_VALVE_RR_TABLE
- PRE_CHARGE_VALVE_1_TABLE
- PRE_CHARGE_VALVE_2_TABLE
- PUMP_1
- PUMP_2

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- BRAKE_CYLINDER_FL
- BRAKE_CYLINDER_FR
- BRAKE_CYLINDER_RL
- BRAKE_CYLINDER_RR
- CHANGE_OVER_VALVE_1
- CHANGE_OVER_VALVE_2
- CONNECTION_CHAMBER_1
- CONNECTION_CHAMBER_2
- DAMPER_CHAMBER_1
- DAMPER_CHAMBER_2
- INLET_VALVE_FL
- INLET_VALVE_FR
- INLET_VALVE_RL
- INLET_VALVE_RR
- NON_RETURN_VALVE_FL
- NON_RETURN_VALVE_FR
- NON_RETURN_VALVE_RES_1
- NON_RETURN_VALVE_RES_2

	■ NON_RETURN_VALVE_RL ■ NON_RETURN_VALVE_RR ■ OUTLET_VALVE_FL ■ OUTLET_VALVE_FR ■ OUTLET_VALVE_RL ■ OUTLET_VALVE_RR ■ PRE_CHARGE_VALVE_1 ■ PRE_CHARGE_VALVE_2 ■ RESERVOIR_1 ■ RESERVOIR_2
BRAKE_HYDRAULICS_ SWITCHES_VALVE_ CONTROL_1 and _2	バルブ制御モードの名前がデジタルから線形に、連続から非線形に変更されています。制御モードの機能は変更されていません。

ASM Diesel Engine Blockset

項目の一覧

本章の内容

ASM Diesel Engine Blockset 2.4 の新機能

ENGINE_SETUP

シミュレーションモデルの応答時間を改善するため、Const_Inj_Matrix パラメータが削除されました。さらに、このパラメータの対応する出力ポートが新しい信号 active_Inj_Direct[]で使用されます。

信号 Sw_Turbo_Stage[:1SingleStage|2TwoStage]の名前が、Sw_Turbo_Stage[1SingleStage|2TwoStage]に変更されています。いくつかの出力ポートの名前も変更されました。

DIRECTINJECTOR

このブロックは、INJECTOR ブロックの改良版です。

シミュレーションモデルの応答時間を改善するため、Const_Inj_Matrix パラメータ（ENGINE_SETUP ブロックから GoToFrom 経由で接続）は削除されました。このブロックには、最新リリース（バージョン 5.0）の ENGINE_SETUP ブロックが必要です。

DIRECTINJECTOR ブロックには、噴射モードを平均値からパルスに切り替える Sw_InjMode パラメータが含まれています。これまで、これは SWITCHES_INJ_MODE ブロックに配置されていました。

UNIT_INJECTOR

シミュレーションモデルの応答時間を改善するため、Const_Inj_Matrix パラメータ（ENGINE_SETUP ブロックから GoToFrom 経由で接続）は削除されました。このブロックには、最新リリース（バージョン 5.0）の ENGINE_SETUP が必要です。

SOFTECU_SETUP

Const_CylinderOffset 信号の単位が[]から[deg]に修正されています。他の入力ポートの名前に含まれていたアンダースコアは削除されました。

RAIL_CONTROL_CRANKBASED

バブルソートは削除されました。

HPP_CRANKBASED	StepSize_Correction は、レール圧を安定化させるために導入されています。安定化は、高いエンジン回転数でパルス制御高圧ポンプを使用する場合に必要となります。DeliveryLength_Correction は、UpdateState_Correction によって拡張されています。この信号は、平均有効圧方式で必要となります。 Sw_FMU_Energized[0Off|1On]信号は、パルスモードでの体積流量の計算時の反復処理によって遅延します。
ENGINE_TORQUE_SET_INTERVENTION	ENGINE_TORQUE_SET_INTERVENTION ブロックが導入され、ギアシフトプロセス中のエンジントルクの介入を容易にします。

ASM Diesel Engine デモモデルの変更点

レール制御	モデルで使用される高圧ポンプに応じて、電流ベースまたはクランク角ベースのレール圧コントローラがあります。
エンジンの MDL_DISP	このデモモデルでは、排気システムのわずかな基本信号だけが、スコープでのビジュアル表示のために準備されます。その他の信号は、ASM Diesel Exhaust Library にあります。エンジンモデルの MDL_DISP には、ASM Diesel Exhaust Library での信号の位置へのリンクが含まれています。
インジェクタ	使用されないインジェクタ（ユニットインジェクションまたは直噴）は無効化されています。この変更により、モデルの処理時間が短縮されます。
ターボのスイッチ	GoToFrom 接続を介して AirPath モデルに ENGINE_SETUP ブロックの信号バス全体を経路指定する代わりに、ターボチャージャモデルのスイッチのみが経路指定されます。この変更により、モデルの処理時間が大幅に短縮されます。
ギアシフト中のトルク介入	エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

環境信号ルーティング	環境信号のルーティングが改訂され再構成されています。MDLUserInterface から他のデモパーツへの信号ルーティングの追跡が簡単で分かりやすくなっています。さらに、ソフト ECU モデルに提供される信号が改訂され、そのいくつかは ASMSignalBus からのセンサ信号に置き換えられています。
運転操作の制御	新しい運転操作の制御が新しいデモに統合されています。新しいインプリメンテーションでは、ModelDesk のみを使用して dSPACE プラットフォーム上でのシミュレーション中に運転操作を開始、停止およびリセットすることができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

ASM Diesel Engine Blockset 2.4 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- AIRFILTER
- CATALYST
- COMBUSTION_TORQUE_CI
- COOLER
- CRANKCASE
- CYLINDER_INLET
- DIESEL_OXIDATION_CATALYST_1_0、DIESEL_OXIDATION_CATALYST
- DIESEL_PARTICULATE_FILTER、DIESEL_PARTICULATE_FILTER_1_0、DIESEL_PARTICULATE_FILTER
- DIRECTINJECTOR_TIMING
- DPF_REGENERATION
- EGR_RATE_CONTROL
- EGR_VALVE、EGR_VALVE_1_0
- EGRCOOLER
- ENGINE_SETUP

- ENGINE_TORQUE_SET
- EXHAUST_MANIFOLD
- EXHAUSTTHROTTLE
- FRICTION_TORQUE
- HIGH_PRESSURE_PUMP
- HPP_CRANKBASED
- HPP_CRANKBASED_1_0
- INJECTION_QUANTITY
- INJECTOR, INJECTOR_1_0、INJECTOR_2_0
- INTAKE_MANIFOLD
- INTERCOOLER
- LP_EGR_VALVE
- LP_EGRCOOLER
- LP_INTAKE_MANIFOLD
- LP_INTAKE_MANIFOLD_4_0
- PRESSURE_CONTROL_VALVE
- PUMP_TORQUE
- RAIL
- RAIL_CONTROL
- RAIL_CONTROL_CRANKBASED
- SMOKE_LIMITATION
- SOFT_ECU_DIESEL_1_0、SOFT_ECU_DIESEL_11_0、SOFT_ECU_DIESEL_3_0
- THROTTLE_MECHANICAL
- THROTTLE_VALVE
- UNIT_INJECTOR、UNIT_INJECTOR_1_0、UNIT_INJECTOR_7_0

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- EGR_VALVE、EGR_VALVE_1_0
- EXHAUST_MANIFOLD
- EXHAUSTTHROTTLE
- LP_EGR_VALVE

	■ LP_INTAKE_MANIFOLD、LP_INTAKE_MANIFOLD_4_0 ■ THROTTLE_MECHANICAL
INJECTOR_9_0	このブロックは、INJECTOR ブロックの以前のバージョンで、廃止されました。移行時に、INJECTOR ブロックのライブラリリンクは以前のバージョンに変更されます。 ブロックの新しい機能には、DIRECT_INJECTOR ブロックを ENGINE_SETUP ブロックとともに使用します。
IDLE_SPEED_CONTROL	移行時に、Trq_Engine_Ind_Max[Nm]出力ポートは終了します。
EGR_VALVE	Ctrl2Pos_EGR パラメータの名前が Map_Ctrl2Pos に変更されています。 Map_A_red パラメータの名前が Map_A_Red に変更されています。
LP_EGR_VALVE	Ctrl2Pos_EGR パラメータの名前が Map_Ctrl2Pos に変更されました。 Map_A_red パラメータの名前が Map_A_Red に変更されました。
SWITCHES_INJ_MODE_1_0	このブロックは SWITCHES_INJ_MODE ブロックの以前のバージョンで、廃止されました。移行時に、SWITCHES_INJ_MODE ブロックのライブラリリンクは SWITCHES_INJ_MODE_1_0 に変更されます。スイッチは、DIRECTINJECTOR ブロックに移動しました。
ENGINE_SETUP_4_0	このブロックは、ENGINE_SETUP ブロックの以前のバージョンです。移行時に、ENGINE_SETUP ブロックのライブラリリンクは ENGINE_SETUP_4_0 に変更されます。このブロックの新機能には、ENGINE_SETUP ブロックを、ENGINE_SETUP とのペアとして機能する最新バージョンのインジェクタブロック（DIRECTINJECTOR、UNIT_INJECTOR）とともに使用します。
UNIT_INJECTOR_7_0	このブロックは、UNIT_INJECTOR ブロックの以前のバージョンです。移行時に、UNIT_INJECTOR ブロックのライブラリリンクは UNIT_INJECTOR_7_0 に変更されます。ブロックの新しい機能には、UNIT_INJECTOR ブロックを ENGINE_SETUP ブロックとともに使用します。
COMMON_ENGINE_ PARAMETERS	dSPACE Release 2015-B および Release 2016-A では、COMMON_ENGINE_PARAMETERS ブロックの cpFuel および cvFuel の内部計算は、誤った出力ポートに接続されています。このエラーは、ブロックのマスクにより修正されています。

ただし、シミュレーション結果を再現可能な状態に維持するため、たとえば、cp_Fuel[J|[kgK]]が cv_Fuel Goto タグに接続されるように、出力ポート cp_Fuel[J|[kgK]]と cv_Fuel[J|[kgK]]への接続は、移行時に入れ替わります。これを手動で無効にして、cp_Fuel[J|[kgK]]が cp_Fuel に接続されるようにすることができます

注記

これは、ASM Gasoline Engine Blockset の CNG システムが使用される場合にのみ必要となります。

ASM Diesel Exhaust Blockset

ASM Diesel Exhaust Blockset 2.1.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- ADBLUE_PRESSURE_REGULATION_VALVE
- ADBLUE_PUMP
- AIR_NON_RETURN_VALVE
- AIR_REGULATION_VALVE
- ATOMIZER
- DIESEL_OXIDATION_CATALYST、DIESEL_OXIDATION_CATALYST_10_0、DIESEL_OXIDATION_CATALYST_4_0
- DIESEL_PARTICULATE_FILTER、DIESEL_PARTICULATE_FILTER_3_0
- INJECTION_VALVE
- MIXING_CHAMBER
- MUFFLER、MUFFLER_2_0
- PUMP_HOSE
- RAW_EXHAUST_COMPOSITION
- SCR_CATALYST、SCR_CATALYST_5_0
- THROTTLE
- UREA_DECOMPOSITION
- VENT_VALVE

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- SCR_CATALYST_5_0

ASM Diesel InCylinder Blockset

項目の一覧

本章の内容

ASM Diesel InCylinder デモモデルの変更

ギアシフト中のトルク介入

エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

環境信号ルーティング

環境信号のルーティングが改訂され再構成されています。MDLUserInterface から他のデモパーツへの信号ルーティングの追跡が簡単で分かりやすくなっています。さらに、ソフト ECU モデルに提供される信号が改訂され、そのいくつかは ASMSignalBus からのセンサ信号に置き換えられています。

運転操作の制御

新しい運転操作の制御が新しいデモに統合されています。新しいインプリメンテーションでは、ModelDesk のみを使用して dSPACE プラットフォーム上でのシミュレーション中に運転操作を開始、停止およびリセットすることができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

ASM Diesel InCylinder Blockset 2.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- DPF_REGENERATION
- HEAT_RELEASE_ARRHENIUS、HEAT_RELEASE_ARRHENIUS_4_0
- HEAT_RELEASE_CHMELA、HEAT_RELEASE_CHMELA_4_0
- IGNITION_DELAY
- SMOKE_LIMITATION
- SOFT_ECU_INCYLINDER_DIESEL_4_0

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- IGNITION_DELAY

ASM Drivetrain Basic Blockset

項目の一覧

本章の内容

ASM Drivetrain Basic Blockset 4.3 の新機能

SOFT_ECU_TRANSMISSION_BASIC

SOFT_ECU_TRANMISSION_BASIC ブロックがモジュール化されています。以前の機能と制御ストラテジは、複数のモジュール型ブロックに分割されました。

必要に応じてコントローラ構造を変更し、拡張することができます。新しいインプリメンテーションは、サードパーティ製コントローラのストラテジを統合できる機能も提供します。

以下のブロックがライブラリに追加されています。

- CLUTCH_ENGAGEMENT_CONTROL
- LOCKUP_CLUTCH_CONTROL
- SHIFT_LOCK_CONTROL
- SHIFT_STRATEGY
- SOFTECU_TRANSMISSION_SETUP
- TORQUE_INTERVENTION_CONTROL
- TIP_SHIFT_CONTROL

構造が完全に新しくなったため、以前の SOFT_ECU_TRANSMISISON_BASIC ブロックは、新しいインプリメンテーションには移行されません。代わりに、以前のブロックは FormerVersions に保存され、移行時にこのブロックのリンクが変更されます。

MANEUVER_CONTROL

新しいブロック MANEUVER_CONTROL が導入され、エンジンデモに関する運転操作制御を中心的に記述します。このブロックは、運転操作時間の開始、一時停止および停止や運転操作のステータスを制御します。

これまでは、運転操作機能は CYCLES と他の非ライブラリブロックに分割されていました。リンクされたライブラリブロックを使用することにより、すべてのエンジンデモに対して統一されたインプリメンテーションを提供します。このインプリメンテーションは拡張が可能で、将来的にも新しい機能を利用することができます。

さらに、新しいブロックでは、ModelDesk のみを使用して dSPACE プラットフォーム上でのシミュレーション中に運転操作を開始、停止およびリセットすることができます。

LONGITUDINAL_CONTROLLER_HYBRID

このブロックには、次のいくつかの新機能があります。

- 両方のペダルの作動速度を制限することができます。
- 異なるペダルを切り替えるのに必要な最小時間は、非ゼロ値に設定することができます。
- ドライバは一定の速度の差を許容することができます。
- プレビュー基準速度に関する新しいコントローラパラメータが、制御アルゴリズム改善の準備として追加されています。

ASM Drivetrain Basic Blockset 4.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- CRANKSHAFT
- CYCLES_7_0
- GEARBOX_AT、GEARBOX_AT_1_0、GEARBOX_AT_8_0
- GEARBOX_MT、GEARBOX_MT_1_0、GEARBOX_MT_8_0
- GEAR_SHIFTER_2_0、GEAR_SHIFTER_6_0
- LONGITUDINAL_CONTROL
- SOFT_ECU_TRANSMISSION_1_0
- SOFT_ECU_TRANSMISSION_BASIC_7_0
- TORQUE_CONTROLLER_3_0
- TORQUE_CONVERTER、TORQUE_CONVERTER_4_0

ASM Utils ブロックのライセンスチェック	ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。 このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ GEAR_SHIFTER_6_0 ■ SOFT_ECU_TRANSMISSION_BASIC_7_0
DRIVING_RESISTANCES	タイヤと路面の間の摩擦は、Const_Fric_Coeff パラメータを使用して記述することができます。このパラメータを使用して、駆動力および制動力を制限することができます。
ENGINE	エンジントルクパラメータのサイズが、[30x30]から[31x30]に拡張されています。以前のパラメータは、`asmmigratepost` で外挿され、初期化されます。
TORQUE_CONTROLLER	逆エンジントルクパラメータのサイズが、[40x40]から[41x40]に拡張されています。以前のパラメータは、`asmmigratepost` で外挿され、初期化されます。
LONGITUDINAL_CONTROLLER	逆エンジントルクパラメータのサイズが、[40x40]から[41x40]に拡張されています。以前のパラメータは、`asmmigratepost` で外挿され、初期化されます。 ブロック内の連続積分器が離散積分器に置き換えられています。
CRANK_SHAFT	ブロック内の連続積分器が離散積分器に置き換えられています。
SHAFT_RIGID	新しいパラメータが、回転抑制を記述するために追加されています。 新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
GEAR_SHIFTER	このブロックには、エンジンが動作していないときにクラッチペダルを作動させる新しいパラメータ（Sw_ClutchPedal_EngineOff）が追加されています。 現在のギアがニュートラルの場合にクラッチを解除すべきではない緊急ブレーキ状況での動作が改善されています。
CYCLES	このブロックは、FormerVersions に移動し、廃止されました。移行時に、リンクが `FormerVerions/CYCLES_7_0` に変更されます。

SOFT_ECU_TRANSMISSION_9_0	このブロックは、自動的に移行することはできません。そのため、移行時に、ブロックへのリンクは FormerVersions/SOFT_ECU_TRANSMISSION_9_0 にある従来のインプリメンテーションに変更されます。 新しいインプリメンテーションを使用するには、付属するデモを新しいブロックを使用するためのテンプレートとして使用することができます。
LONGITUDINAL_CONTROLLER_HYBRID	移行時に、次の新しいパラメータが追加されます。 ■ Const_Rate_Max_AccPedal および Const_Rate_Max_BrakePedal：ペダルの最大変更レート ■ Cont_t_min_Pedals：異なるペダルを切り替えるのに必要な最小時間 ■ Map_v_Dead_Driver および Sw_v_Dead_Driver：許容される速度差 ■ Const_Preview_vCtrl_Pedals：プレビュー基準速度に関するコントローラパラメータ これらのパラメータは、asmmigratepost で初期化され、以前の動作が変更されずに維持されます。

ASM Electric Components Blockset

項目の一覧

本章の内容

ASM Electric Components Blockset 3.3 の新機能

BATTERY_MULTICELL

この熱電池モデルは、個々のセルと隣接するセル同士の熱的相互作用をシミュレートするために拡張されています。また、冷却プレートは、6 つのバッテリパック表面をシミュレートすることもできます。

このモデルには、鉛酸、ニッケル水素（NiMH）、およびリチウムイオン（NMC、LFP、LTO）バッテリなどの各種バッテリタイプやテクノロジに関するパラメータ設定の例が含まれています。

THREE_PHASE_RECTIFIER

AC 電源を DC 電源に変換します。このモデルは、3 相電圧、DC 電圧および電源パラメータに基づき、導通および整流状態での直流（DC）電流や線電流をシミュレートします。

BATTERY

このモデルには、鉛酸、ニッケル水素（NiMH）、およびリチウムイオン（NMC、LFP、LTO）バッテリなどの各種バッテリタイプやテクノロジに関するパラメータ設定の例が含まれています。

ASM Electric Components デモモデルの変更

Electric Drive 閉ループかご型誘導モーター DQ

電源は、THREE_PHASE_POWERSUPPLY、THREE_PHASE_RECTIFIER および DC_LINK ブロックで構成されます。

ASM Electric Components Blockset 3.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- ALTERNATOR、ALTERNATOR_2_0
- BATTERY、BATTERY_4_0
- BATTERY_CELL
- BATTERY_MULTICELL、BATTERY_MULTICELL_2_0
- BATTERY_THERMAL
- BRUSHLESS_DC_MACHINE_ALPHA_BETA
- COMPRESSOR
- CURRENT_SINK
- DC_MACHINE、DC_MACHINE_1_0
- DRIVE_MANAGEMENT
- FAN
- PERMANENT_MAGNET_SYNCHRONOUS_MACHINE_D_Q
- POTENTIOMETER
- PTC_GRID_DEFROSTER
- PTC_HEATER
- SOFT_ECU_HYBRID_MANAGER_1_0
- STARTER、STARTER_2_0
- TRQ_REQUEST_COORDINATION

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

	このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ BATTERY_CELL ■ BATTERY_MULTICELL、BATTERY_MULTICELL_2_0 ■ PMSM_D_Q_NONLINEAR
BATTERY_THERMAL	このブロックには、次の 2 つの新しい出力ポートがあります。Capacity_Thermal[J|(kgK)]および P_MainAndLossReaction[W]
THREE_PHASE_ POWERSUPPLY	このブロックには、次の新しい入力ポートがあります。Init_Angle[Rad]および Reset[0|1] このブロックには、次の新しい出力ポートがあります。Const_R_AC[Ohm]および Const_L_AC[H] 次のポートの名前は変更されています。v_Mains[V]は V_phase_to_phase_rms[V]に、v_Mains[a;b;c][V]は v [a;b;c][V]に変更されました。
DC_LINK	このブロックには、次の 2 つの新しい入力ポートがあります。I_DCLink[A]および Sw_Source[1Current|2voltage] このブロックには、次の新しい出力ポートがあります。Const_R_DC[Ohm] 出力ポート V_Out[V]の名前は、V_DCLink[V]に変更されました。

ASM Environment Blockset

項目の一覧

本章の内容

ASM Environment Blockset 4.5 の新機能

LONGITUDINAL_CONTROLLER_HYBRID

このブロックには、次のいくつかの新機能があります。

- 両方のペダルの作動速度を制限することができます。
- 異なるペダルを切り替えるのに必要な最小時間は、非ゼロ値に設定することができます。
- ドライバは一定の速度の差を許容することができます。
- プレビュー基準速度に関する新しいコントローラパラメータが、制御アルゴリズム改善の準備として追加されています。

ASM Environment Blockset 4.5 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- BRAKE_HYDRAULICS_VARIANT
- BRAKE_PNEUMATICS_VARIANT
- CONTROLLER
- GEAR_SHIFTER_9_0

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

	このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ BASIC_ROADS ■ GEAR_SHIFTER_9_0 ■ LATERAL_CONTROL1 ■ MANEUVER_SCHEDULER ■ ROAD
GEAR_SHIFTER	このブロックには、エンジンが動作していないときにクラッチペダルを作動させる新しいパラメータ（Sw_ClutchPedal_EngineOff）が追加されています。 現在のギアがニュートラルの場合にクラッチを解除すべきではない緊急ブレーキ状況での動作が改善されています。
LONGITUDINAL_ CONTROLLER_HYBRID	移行時に、次の新しいパラメータが追加されます。 ■ Const_Rate_Max_AccPedal および Const_Rate_Max_BrakePedal：ペダルの最大変更レート ■ Cont_t_min_Pedals：異なるペダルを切り替えるのに必要な最小時間 ■ Map_v_Dead_Driver および Sw_v_Dead_Driver：許容される速度差 ■ Const_Preview_vCtrl_Pedals：プレビュー基準速度に関するコントローラパラメータ これらのパラメータは、`asmmigratepost` で初期化され、以前の動作が変更されずに維持されます。

ASM Gasoline Engine Basic Blockset

項目の一覧

本章の内容

ASM Gasoline Engine Basic Blockset 2.1 の新機能

ENGINE_SETUP

シミュレーションモデルの応答時間を改善するため、Const_Inj_Matrix パラメータが削除されました。さらに、このパラメータの対応する出力ポートが新しい信号 active_Inj_Direct[]で使用されます。

信号 Sw_Turbo_Stage[:1SingleStage|2TwoStage]の名前が、Sw_Turbo_Stage[1SingleStage|2TwoStage]に変更されています。いくつかの出力ポートの名前も変更されました。

ENGINE_TORQUE_SET_INTERVENTION

ENGINE_TORQUE_SET_INTERVENTION ブロックが導入され、ギアシフトプロセス中のエンジントルクの介入を容易にします。

SOFTECU_SETUP

Const_CylinderOffset 信号の単位が[]から[deg]に修正されています。他の入力ポートの名前に含まれていたアンダースコアは削除されました。

PORTINJECTOR

このブロックには、次の新しい信号があります。ASMSignalBus の q_mean_inj[mm3|cyc]

ASM Engine Gasoline Basic デモモデルの変更

ギアシフト中のトルク介入

エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

環境信号ルーティング	環境信号のルーティングが改訂され再構成されています。MDLUserInterface から他のデモパーツへの信号ルーティングの追跡が簡単で分かりやすくなっています。さらに、ソフト ECU モデルに提供される信号が改訂され、そのいくつかは ASMSignalBus からのセンサ信号に置き換えられています。
運転操作の制御	新しい運転操作の制御が新しいデモに統合されています。新しいインプリメンテーションでは、ModelDesk のみを使用して dSPACE プラットフォーム上でのシミュレーション中に運転操作を開始、停止およびリセットすることができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

ASM Gasoline Engine Basic Blockset 2.1 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- COMBUSTION_TORQUE_SI
- COOLER
- CYLINDER_INLET
- ENGINE_TORQUE_SET
- FRICTION_TORQUE、FRICTION_TORQUE_3_0
- IGNITION_SET
- INJECTOR_SI_4_0
- LAMBDA_CONTROL
- MAPS_TC
- PORTINJECTOR
- PORTINJECTOR_TIMING
- REL_AIRMASS_MAPBASED
- SOFT_ECU_GASOLINE
- SOFT_ECU_GASOLINEBASIC_7_0
- THROTTLE_CONTROL

	■ THROTTLE_MECHANICAL ■ THROTTLE_VALVE ■ TURBO_CONTROL ■ WALL_FILM
ASM Utils ブロックのライセンスチェック	ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。 このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ ENGINE_SETUP ■ INTAKE_MANIFOLD ■ MAPS_TC ■ THROTTLE_MECHANICAL
IDLE_SPEED_CONTROL	移行時に、Trq_Engine_Ind_Max[Nm]出力ポートは終了します。
ENGINE_SETUP_4_0	このブロックは、ENGINE_SETUP ブロックの以前のバージョンです。移行時に、ENGINE_SETUP ブロックのライブラリリンクは ENGINE_SETUP_4_0 に変更されます。このブロックの新機能には、ENGINE_SETUP ブロックを、ENGINE_SETUP とのペアとして機能する最新バージョンのインジェクタブロック（DIRECTINJECTOR、UNIT_INJECTOR）とともに使用します。
COMMON_ENGINE_PARAMETERS	dSPACE Release 2015-B および Release 2016-A では、COMMON_ENGINE_PARAMETERS ブロックの cpFuel および cvFuel の内部計算は、誤った出力ポートに接続されています。このエラーは、ブロックのマスクにより修正されています。 ただし、シミュレーション結果を再現可能な状態に維持するため、たとえば、cp_Fuel[J[kgK]]が cv_Fuel Goto タグに接続されるように、出力ポート cp_Fuel[J[kgK]]と cv_Fuel[J[kgK]]への接続は、移行時に入れ替わります。これを手動で無効にして、cp_Fuel[J[kgK]]が cp_Fuel に接続されるようにすることができます **注記** これは、ASM Gasoline Engine Blockset の CNG システムが使用される場合にのみ必要となります。

ASM Gasoline Engine Blockset

項目の一覧

本章の内容

ASM Gasoline Engine Blockset 3.4 の新機能

ENGINE_SETUP

シミュレーションモデルの応答時間を改善するため、Const_Inj_Matrix パラメータが削除されました。さらに、このパラメータの対応する出力ポートが新しい信号 active_Inj_Direct[]で使用されます。

信号 Sw_Turbo_Stage[:1SingleStage|2TwoStage]の名前が、Sw_Turbo_Stage[1SingleStage|2TwoStage]に変更されています。いくつかの出力ポートの名前も変更されました。

DIRECTINJECTOR

シミュレーションモデルの応答時間を改善するため、Const_Inj_Matrix パラメータ（ENGINE_SETUP ブロックから GoToFrom 経由で接続）は削除されました。このブロックには、最新リリース（バージョン 5.0）の ENGINE_SETUP ブロックが必要です。

DIRECTINJECTOR ブロックには、噴射モードを平均値からパルスに切り替える Sw_InjMode パラメータが含まれています。これまで、これは SWITCHES_INJ_MODE ブロックに配置されていました。

SOFTECU_SETUP

Const_CylinderOffset 信号の単位が[]から[deg]に修正されています。他の入力ポートの名前に含まれていたアンダースコアは削除されました。

RAIL_CONTROL_CRANKBASED

バブルソートは削除されました。

INJECTOR_MODE

Sw_Mode_Strat パラメータがブロックに追加され、成層燃焼モードを簡単に無効にすることができます。

CATALYST

酸素貯蔵と熱平衡を考慮した新しい触媒モデルが実装されています。

LAMBDA_SENSOR	新しいラムダセンサは、センサの加熱を考慮するために開発されました。
HPP_CRANKBASED	StepSize_Correction は、レール圧を安定化させるために導入されています。安定化は、高いエンジン回転数でパルス制御高圧ポンプを使用する場合に必要となります。DeliveryLength_Correction は、UpdateState_Correction によって拡張されています。この信号は、平均有効圧方式で必要となります。 Sw_FMU_Energized[0Off|1On]信号は、パルスモードでの体積流量の計算時の反復処理によって遅延します。
ENGINE_TORQUE_SET_INTERVENTION	ENGINE_TORQUE_SET_INTERVENTION ブロックが導入され、ギアシフトプロセス中のエンジントルクの介入を容易にします。

ASM Engine Gasoline デモモデルの変更

レール制御	モデルで使用される高圧ポンプに応じて、電流ベースまたはクランク角ベースのレール圧コントローラがあります。
排気システム	CATALYST および LAMBDA_SENSOR ブロックが新しくなったため、ExhaustSystem モデルの構造が変更されています。
燃料システム	ポートインジェクタと直噴インジェクタの両方を同時にアクティブにできるようになりました。シリンダに噴射される燃料の量は、両方のインジェクタの合計です。
ターボのスイッチ	GoToFrom 接続を介して AirPath モデルに ENGINE_SETUP ブロックの信号バス全体を経路指定する代わりに、ターボチャージャモデルのスイッチのみが経路指定されます。この変更により、モデルの処理時間が大幅に短縮されます。
ギアシフト中のトルク介入	エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

環境信号ルーティング	環境信号のルーティングが改訂され再構成されています。MDLUserInterface から他のデモパーツへの信号ルーティングの追跡が簡単で分かりやすくなっています。さらに、ソフト ECU モデルに提供される信号が改訂され、そのいくつかは ASMSignalBus からのセンサ信号に置き換えられています。
運転操作の制御	新しい運転操作の制御が新しいデモに統合されています。新しいインプリメンテーションでは、ModelDesk のみを使用して dSPACE プラットフォーム上でのシミュレーション中に運転操作を開始、停止およびリセットすることができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

ASM Gasoline Engine Basic Blockset 3.4 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- AIRFILTER
- CNG_SHUTOFF_VALVE
- COOLER
- CNG_PRESSURE_REGULATOR
- COMBUSTION_TORQUE_CI
- COMBUSTION_TORQUE_SI
- CRANKCASE
- CYLINDER_INLET
- DIRECTINJECTOR
- DIRECTINJECTOR_TIMING
- EGRCOOLER
- EGR_RATE_CONTROL
- EGR_VALVE
- ENGINE_TORQUE_SET
- EXHAUST_MANIFOLD

- FRICTION_TORQUE、FRICTION_TORQUE_3_0
- HIGH_PRESSURE_PUMP
- HPP_CRANKBASED、HPP_CRANKBASED_1_0
- IGNITION_SET
- INJECTION_QUANTITY
- INJECTOR_4_0
- INJECTOR_MODE
- INTERCOOLER
- LAMBDA_CONTROL
- PORTINJECTOR
- PORTINJECTOR_TIMING
- PRESSURE_CONTROL_VALVE
- PUMP_TORQUE
- RAIL
- RAIL_CONTROL
- RAIL_CONTROL_CRANKBASED
- REL_AIRMASS_MAPBASED
- SOFT_ECU_GASOLINE_12_0
- THROTTLE_MECHANICAL
- THROTTLE_VALVE
- WALL_FILM

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- EGR_VALVE
- ENGINE_SETUP
- EXHAUST_MANIFOLD
- HPP_CRANKBASED
- INTAKE_MANIFOLD
- RAIL_CONTROL_CRANKBASED
- THROTTLE_MECHANICAL

EGR_VALVE	Ctrl2Pos_EGR パラメータの名前が Map_Ctrl2Pos に変更されています。 Map_A_red パラメータの名前が Map_A_Red に変更されています。
SWITCHES_INJ_MODE_1_0	このブロックは SWITCHES_INJ_MODE ブロックの以前のバージョンで、廃止されました。移行時に、SWITCHES_INJ_MODE ブロックのライブラリリンクは SWITCHES_INJ_MODE_1_0 に変更されます。スイッチは、DIRECTINJECTOR ブロックに移動しました。
CATALYST_4_0	このブロックは、CATALYST ブロックの以前のバージョンです。移行時に、ブロックのライブラリリンクは CATALYST_4_0 に変更されます。新しい機能には CATALYST ブロックを使用します。
SELECTFUELINJECTION_1_0	このブロックは、SELECTFUELINJECTION ブロックの以前のバージョンで、廃止されました。移行時に、ブロックのライブラリリンクは SELECTFUELINJECTION_1_0 に変更されます。現行のリリースでは、ポートインジェクタと直噴インジェクタは両者とも同時にアクティブにすることができます。シリンダに噴射される燃料の量は、両方のインジェクタの合計です。
IDLE_SPEED_CONTROL	移行時に、Trq_Engine_Ind_Max[Nm]出力ポートは終了します。
ENGINE_SETUP_4_0	このブロックは、ENGINE_SETUP ブロックの以前のバージョンです。移行時に、ENGINE_SETUP ブロックのライブラリリンクは ENGINE_SETUP_4_0 に変更されます。このブロックの新機能には、ENGINE_SETUP ブロックを、ENGINE_SETUP とのペアとして機能する最新バージョンのインジェクタブロック（DIRECTINJECTOR、UNIT_INJECTOR）とともに使用します。
DIRECT_INJECTOR_8_0	このブロックは、DIRECT_INJECTOR ブロックの以前のバージョンです。移行時に、ブロックのライブラリリンクは DIRECT_INJECTOR_8_0 に変更されます。新しい機能には DIRECT_INJECTOR ブロックを使用します。このブロックは、ENGINE_SETUP ブロックとのペアとして機能します。
COMMON_ENGINE_ PARAMETERS	dSPACE Release 2015-B および Release 2016-A では、COMMON_ENGINE_PARAMETERS ブロックの cpFuel および cvFuel の内部計算は、誤った出力ポートに接続されています。このエラーは、ブロックのマスクにより修正されています。 ただし、シミュレーション結果を再現可能な状態に維持するため、たとえば、cp_Fuel[J][kgK]]が cv_Fuel Goto タグに接続されるように、出力ポート cp_Fuel[J][kgK]]と cv_Fuel[J][kgK]]への接続は、移行時に入れ替わります。これを手動で無効にして、cp_Fuel[J][kgK]]が cp_Fuel に接続されるようにすることができます

注記

これは、ASM Gasoline Engine Blockset の CNG システムが使用される場合にのみ必要となります。

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 デモモデルの変更

ギアシフト中のトルク介入

エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

環境信号ルーティング

環境信号のルーティングが改訂され再構成されています。MDLUserInterface から他のデモパーツへの信号ルーティングの追跡が簡単で分かりやすくなっています。さらに、ソフト ECU モデルに提供される信号が改訂され、そのいくつかは ASMSignalBus からのセンサ信号に置き換えられています。

運転操作の制御

新しい運転操作の制御が新しいデモに統合されています。新しいインプリメンテーションでは、ModelDesk のみを使用して dSPACE プラットフォーム上でのシミュレーション中に運転操作を開始、停止およびリセットすることができます。「ASM Drivetrain Basic Blockset 4.3 の新機能」(75 ページ)を参照してください。

ASM Gasoline InCylinder Blockset 2.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

	ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。 ■ CATALYST ■ HEAT_RELEASE_VIBE ■ IGNITION_SET ■ INJECTOR_MODE ■ LAMBDA_CONTROL ■ PORTINJECTOR ■ SOFT_ECU_INCYLINDER_GASOLINE_6_0
INJECTOR_MODE	このブロックには、成層燃焼モードを簡単に無効にできる新しい Add Parameter Sw_Mode_Strat パラメータがあります。

ASM Optimizer

ASM Optimizer 1.8 の新機能

ModelDesk 計測データファイル

ModelDesk 計測データファイル（*.md）は、InCylinder 計測用のソースファイルとして使用することができます。

またこのファイルには、ASM Diesel InCylinder および ASM Gasoline InCylinder Blockset のデモプロジェクトも含まれています。

ASM Pneumatics Blockset

ASM Pneumatics Blockset 2.0.4 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- FOOT_BRAKE_MODULE

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- AIR_SUSPENSION_FORCES_FRONT
- AIR_SUSPENSION_FORCES_REAR
- AIR_SUSPENSION_FORCES_REAR_2ND
- AIR_SUSPENSION_FORCES_REAR_3RD
- AIR_SUSPENSION_FORCES_REAR_4TH
- TRL_AIR_SUSPENSION_FORCES_FRONT
- TRL_AIR_SUSPENSION_FORCES_REAR
- TRL_AIR_SUSPENSION_FORCES_REAR_2ND

ASM Traffic Blockset

項目の一覧

本章の内容

ASM Traffic Blockset 3.5 の新機能

LINE_SENSOR

LINE_SENSOR ブロックは、交差点のラインシェイプや連続シェイプも検出することができます。

ASM Traffic デモモデルの変更

ギアシフト中のトルク介入

エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Vehicle Dynamics Blockset 3.4 の新機能」(109 ページ)を参照してください。

ASM Traffic Blockset 3.5 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- SENSOR_SELECTOR

ASM Utils ブロックのライセンスチェック	ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。 このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ CUSTOM_SENSOR_OUTPUT ■ CUSTOM_SENSOR_SCOPEZONE_CALCULATION ■ FELLOW_PARAMETERS ■ LINE_SENSOR ■ OBJECT_POSITION ■ OBJECT_PROPERTIES_BOX ■ OBJECT_SENSOR_2D_CALCULATION ■ OBJECT_SENSOR_3D_CALCULATION ■ RADARSENSOR_3D ■ TRAFFIC_OBJECTS ■ TRAFFIC_SCHEDULER ■ TRAFFIC_SIGN_SENSOR_CALCULATION
SOFT_ECU_ACC	このブロックには、次の 2 つの新しい出力ポートがあります。Enable_Trq_Brake_ACC[0|1]および Enable_Trq_Engine_ACC[0|1]
OUTPUT_INTERFACE_ACC	このブロックには、次の 2 つの新しい入力ポートがあり、トルク要求値を無効にすることができます。Enable_Trq_Brake_ACC[0|1]および Enable_Trq_Engine_ACC[0|1]
USER_INTERFACE_ACC	クルーズコントロール設定速度を設定する機能が修正されています。

ASM Trailer Blockset

項目の一覧

本章の内容

ASM Trailer Blockset 2.6 の新機能

TIRE_MODEL_MAGIC_ FORMULA_TRAILER_*	無負荷時のタイヤ半径の円周に関する新しい信号が、信号バスに追加されています。
SUSKIN_RIGID_TRUCK_ TRAILER_*	ボール循環式ステアリングシステムを備えたトラックのリジッドアクスルなどの新しいサスペンション運動学があります。
CABIN_VELOCITIES	この新しいブロックは、キャビンの重心での速度を計算します。
ARTICULATED_BODY_ CABIN	この新しいブロックは、キャビンボディの連結された質量マトリクスや連結された一般化力を計算します。
ABA_ACC_CABIN	この新しいブロックは、キャビンボディの加速度を計算します。
PIVOT_POS_CABIN	この新しいブロックは、キャビンボディの回転軸の位置を計算します。
PIVOT_VEL_CABIN	この新しいブロックは、キャビンボディの回転軸の速度を計算します。
PIVOT_FORCE_TORQUE_ CABIN	この新しいブロックは、キャビンボディの自由度に関するスプリングダンパー要素を計算します。

ASM Trailer デモモデルの変更

ギアシフト中のトルク介入

エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Vehicle Dynamics Blockset 3.4 の新機能」(109 ページ)を参照してください。

ASM Trailer Blockset 2.6 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- AERODYNAMICS_TRAILER
- HITCH_TORQUE_BALL_JOINT
- HITCH_TORQUE_BALL_JOINT_DOLLY
- MASTER_BRAKE_CYL_TRAILER
- OMEGA_WHEEL
- OVERRUN_BRAKE_TRAILER
- SUSCOMP_OPP_TRAILER_FRONT
- SUSCOMP_OPP_TRAILER_REAR
- SUSCOMP_OPP_TRAILER_REAR_2ND
- SUSCOMP_OPP_TRAILER_REAR_3RD
- SUSCOMP_RIGID_SYM_TRAILER_FRONT
- SUSCOMP_RIGID_SYM_TRAILER_REAR_2ND
- SUSCOMP_RIGID_SYM_TRAILER_REAR
- SUSCOMP_RIGID_SYM_TRAILER_REAR_3RD
- SUSCOMP_TRAILER_FRONT
- SUSCOMP_TRAILER_REAR
- SUSCOMP_TRAILER_REAR_2ND

- SUSCOMP_TRAILER_REAR_3RD
- TIRE_MODEL_MAGIC_FORMULA_TRAILER_***
- TIRE_MODEL_TMEASY_TRAILER_***

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- DRUM_BRAKE_TRAILER_FRONT
- DRUM_BRAKE_TRAILER_REAR
- DRUM_BRAKE_TRAILER_REAR_2ND
- DRUM_BRAKE_TRAILER_REAR_3RD
- SUSFORCES_ACTIVE_TRAILER_FRONT
- SUSFORCES_ACTIVE_TRAILER_REAR
- SUSFORCES_ACTIVE_TRAILER_REAR_2ND
- SUSFORCES_ACTIVE_TRAILER_REAR_3RD
- SUSFORCES_TRAILER_FRONT_ANC_SPRING
- SUSFORCES_TRAILER_FRONT
- SUSFORCES_TRAILER_REAR
- SUSFORCES_TRAILER_REAR_ANC_SPRING
- SUSFORCES_TRAILER_REAR_2ND
- SUSFORCES_TRAILER_REAR_2ND_ANC_SPRING
- SUSFORCES_TRAILER_REAR_3RD
- SUSFORCES_TRAILER_REAR_3RD_ANC_SPRING
- SUSKIN_RIGID_SYM_TRAILER_FRONT
- SUSKIN_TRAILER_FRONT_ASYM_3DOF
- SUSKIN_TRAILER_FRONT_ASYM_ANC_SPRING_3DOF
- SUSKIN_TRAILER_FRONT_SYM_3DOF
- SUSKIN_TRAILER_FRONT_SYM_ANC_SPRING_3DOF
- SUSKIN_RIGID_SYM_TRAILER_REAR
- SUSKIN_RIGID_SYM_TRAILER_REAR_2ND
- SUSKIN_TRAILER_REAR_ASYM_3DOF
- SUSKIN_TRAILER_REAR_ASYM_ANC_SPRING_3DOF
- SUSKIN_TRAILER_REAR_SYM_3DOF

	■ SUSKIN_TRAILER_REAR_SYM_ANC_SPRING_3DOF ■ SUSKIN_TRAILER_REAR_2ND_ASYM_3DOF ■ SUSKIN_TRAILER_REAR_2ND_ASYM_ANC_SPRING_3DOF ■ SUSKIN_TRAILER_REAR_2ND_SYM_3DOF ■ SUSKIN_TRAILER_REAR_2ND_SYM_ANC_SPRING_3DOF ■ SUSKIN_RIGID_SYM_TRAILER_REAR_3RD ■ SUSKIN_TRAILER_REAR_3RD_ASYM_3DOF ■ SUSKIN_TRAILER_REAR_3RD_ASYM_ANC_SPRING_3DOF ■ SUSKIN_TRAILER_REAR_3RD_SYM_3DOF ■ SUSKIN_TRAILER_REAR_3RD_SYM_ANC_SPRING_3DOF
TIRE_MODEL_TMEASY_TRAILER* **	摩擦係数がゼロの場合は、ゼロ除算が回避されます。

ASM Truck Blockset

項目の一覧

本章の内容

ASM Truck Blockset 3.0 の新機能

TIRE_MODEL_MAGIC_FORMULA_* **	無負荷時のタイヤ半径の円周に関する新しい信号が、信号バスに追加されています。
SUSKIN_RIGID_TRUCK_* **	ボール循環式ステアリングシステムを備えたトラックのリジッドアクスルなどの新しいサスペンション運動学。
CABIN_MASS	2 自由度(垂直方向の変位と y 軸周りの回転)をもつ追加のキャビンボディをシミュレートするための新しいブロックキャビン質量。

ASM Truck デモモデルの変更

ギアシフト中のトルク介入	エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Vehicle Dynamics Blockset 3.4 の新機能」(109 ページ)を参照してください。

ASM Truck Blockset 3.0 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- OMEGA_WHEEL
- SUSCOMP_RIGID_SYM_REAR_2ND
- SUSCOMP_RIGID_SYM_REAR_3RD
- SUSPENSION_COMPLIANCE_OPP_REAR_2ND
- SUSPENSION_COMPLIANCE_OPP_REAR_3RD
- SUSPENSION_COMPLIANCE_REAR_2ND
- SUSPENSION_COMPLIANCE_REAR_3RD
- TIRE_MODEL_MAGIC_FORMULA_***
- TIRE_MODEL_TMEASY_***

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- DRUM_BRAKE_REAR_2ND
- DRUM_BRAKE_REAR_3RD
- PIVOT_TORQUE_X_VEHICLE_2ND
- SUSFORCES_REAR_2ND_ANC_SPRING
- SUSFORCES_REAR_3RD_ANC_SPRING
- SUSKIN_REAR_2ND_ASYM_3DOF
- SUSKIN_REAR_2ND_ASYM_ANC_SPRING_3DOF
- SUSKIN_REAR_2ND_SYM_3DOF
- SUSKIN_REAR_2ND_SYM_ANC_SPRING_3DOF
- SUSKIN_REAR_3RD_ASYM_3DOF
- SUSKIN_REAR_3RD_ASYM_ANC_SPRING_3DOF
- SUSKIN_REAR_3RD_SYM_3DOF
- SUSKIN_REAR_3RD_SYM_ANC_SPRING_3DOF

	■ SUSKIN_RIGID_SYM_REAR_2ND ■ SUSKIN_RIGID_SYM_REAR_3RD ■ SUSPENSION_FORCES_ACTIVE_REAR_3RD ■ SUSPENSION_FORCES_ACTIVE_REAR_2ND ■ SUSPENSION_FORCES_REAR_2ND ■ SUSPENSION_FORCES_REAR_3RD
TIRE_MODEL_TMEASY_*	摩擦係数がゼロの場合は、ゼロ除算が回避されます。

ASM Turbocharger Blockset

項目の一覧

本章の内容

ASM Turbocharger Blockset 3.1.3 の新機能

COMPRESSOR、COMPRESSOR_HP

cp_air 入力ポートの名前が cp_In_Comp に、kappa_Air 入力ポートの名前が kappa_In_Comp に変更されました。

新しい入力ポート Conv_mdot_Comp および Conv_omega_TC が、カスタム値変換用ブロックに追加されています。

TURBINE、TURBINE_HP、TURBINE_SAEJ922

新しい入力ポート Conv_mdot_Turb および Conv_omega_TC が、カスタム値変換用ブロックに追加されています。

ASM Turbocharger Blockset 3.1.3 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- COMPRESSOR
- COMPRESSOR_HP
- MAPS_TC_2_0、MAPS_TC_6_0
- SHAFT_TC、SHAFT_TC_5_0
- SHAFT_TC_HP
- TURBINE、TURBINE_11_0
- TURBINE_HP、TURBINE_HP_6_0

- TURBINE_SAEJ922、TURBINE_SAEJ922_6_0
- TURBO_BASIC、TURBO_BASIC_8_0
- TURBO_BASIC_2STAGE、TURBO_BASIC_2STAGE_3_0
- TURBO_CONTROL
- WASTEGATE_VALVE、WASTEGATE_VALVE_6_0
- WASTEGATE_VALVE_HP、WASTEGATE_VALVE_HP_3_0

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- MAPS_TC_2_0、MAPS_TC_6_0
- POSTTURBHPMAN
- SHAFT_TC、SHAFT_TC_2_0、SHAFT_TC_5_0
- SHAFT_TC_HP
- TURBINE、TURBINE_11_0
- TURBINE_HP、TURBINE_HP_6_0
- TURBO_BASIC、TURBO_BASIC_8_0
- TURBO_BASIC_2STAGE、TURBO_BASIC_2STAGE_3_0
- TURBINE_SAEJ922

ASM Utils

ASM Utils の移行

ルックアップテーブルの移行	ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。
ASM Utils ブロックのライセンスチェック	ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。 このライブラリ内の以下のブロックで、Utils ブロックが更新されています。 ■ Forward Euler ■ asm_1d_integral_recip_lut ■ asm_1d_lookup ■ asm_2d_lookup ■ asm_shifttable_lookup ■ compare_value これらのブロックが（別の ASM ブロック内ではなく）モデルで使用される場合、移行プロセスは最も近い ASM ブロックを探し、関連するライブラリをライセンスチェックに使用するように設定します。
LabelInterface Bus2Vector ブロック	Vector2Datatype および Datatype2Vector ブロックを個別に生成する Bus2Vector ブロックのオプションは廃止されました。この変換は、Bus2Vector および Vector2Bus ブロックに常に含まれるようになりました。 Bus2Vector ブロックが更新された場合は、関連する Vector2Datatype および Datatype2Vector ブロックを手作業で削除する必要があります。

ASM Vehicle Dynamics Blockset

項目の一覧

本章の内容

ASM Vehicle Dynamics Blockset 3.4 の新機能

TIRE_MODEL_MAGIC_FORMULA_*

無負荷時のタイヤ半径の円周に関する新しい信号が、信号バスのブロックに追加されています。

SUSKIN_RIGID_TRUCK_*

このブロックには、ボール循環式ステアリングシステムを備えたトラックのリジッドアクスルなどの新しいサスペンション運動学モデルが含まれています。

SOFT_ECU_TRANSMISSION

SOFT_ECU_TRANMISSION ブロックがモジュール化されています。以前の機能と制御ストラテジは、複数のモジュール型ブロックに分割されました。

必要に応じてコントローラ構造を変更し、拡張することができます。新しいインプリメンテーションは、サードパーティ製コントローラのストラテジを統合できる機能も提供します。

以下のブロックがライブラリに追加されています。

- CLUTCH_ENGAGEMENT_CONTROL
- LOCKUP_CLUTCH_CONTROL
- SHIFT_LOCK_CONTROL
- SHIFT_STRATEGY
- SOFTECU_TRANSMISSION_SETUP
- TORQUE_INTERVENTION_CONTROL
- TIP_SHIFT_CONTROL

構造が完全に新しくなったため、以前の SOFT_ECU_TRANSMISISON ブロックは、新しいインプリメンテーションには移行されません。代わりに、以前のブロックは FormerVersions に保存され、移行時にこのブロックのリンクが変更されます。

ASM Vehicle Dynamics デモモデルの変更

ギアシフト中のトルク介入

エンジンおよびトランスミッションソフト ECU は相互接続されているため、ギアシフトプロセス中にエンジントルクの介入を実現することができます。介入トルクの量は、SoftECU_Transmission の TORQUE_INTERVENTION_CONTROL ブロックでパラメータ化することができます。「ASM Vehicle Dynamics Blockset 3.4 の新機能」(109 ページ)を参照してください。

ASM Vehicle Dynamics Blockset 3.4 への移行

ルックアップテーブルの移行

ASM ライブラリブロック内の廃止された Simulink ブロック Lookup および Lookup2D は、新しい標準 Simulink Lookup Table (n-D)ブロックに更新されています。「すべての ASM ブロックセットの移行」(59 ページ)を参照してください。

ルックアップテーブルは、このライブラリ内の以下のブロックに更新されています。

- AERODYNAMICS
- BRAKE_BOOSTER
- CENTRAL_DIFFERENTIAL
- CLUTCH
- CLUTCH_4WD
- ENGINE
- ENGINE_BASIC_3_0
- ESP_TORQUE_INTERVENTION_SLOW_3_0
- FRONT_DIFFERENTIAL
- GEARBOX_AT
- GEARBOX_MT
- MANIFOLD_PRESSURE

- REAR_DIFFERENTIAL
- SOFT_ECU_TRANSMISSION_8_0
- STEERING
- STEERING_VARIABLE_RATIO
- STEERING_3DOF_VARIABLE_RATIO
- SUBFRAME
- SUSCOMP_RIGID_SYM_FRONT
- SUSCOMP_RIGID_SYM_REAR
- SUSPENSION_COMPLIANCE_FRONT
- SUSPENSION_COMPLIANCE_OPP_FRONT
- SUSPENSION_COMPLIANCE_OPP_REAR
- SUSPENSION_COMPLIANCE_REAR
- TIRE_MODEL_MAGIC_FORMULA_***
- TIRE_MODEL_TMEASY_***
- TORQUE_CONVERTER
- TORQUE_CONVERTER_RIGID
- TORQUE_INTERVENTION_SLOW_ENGINE_BASIC
- WHEEL_SPEED

ASM Utils ブロックのライセンスチェック

ASM_UTILS ライセンスは廃止されました。ASM Utils ブロックが、使用されている ASM ブロックセットのライセンスをチェックします。

このライブラリ内の以下のブロックで、Utils ブロックが更新されています。

- CLUTCH
- DRUM_BRAKE_FRONT
- DRUM_BRAKE_REAR
- RIGID_AXLE
- SOFT_ECU_TRANSMISSION
- STEERING_VARIABLE_RATIO
- STEERING_3DOF_VARIABLE_RATIO
- SUSKIN_FRONT_ASYM_3DOF
- SUSKIN_FRONT_ASYM_ANC_SPRING
- SUSKIN_FRONT_ASYM_ANC_SPRING_3DOF
- SUSKIN_FRONT_SYM_3DOF

	■ SUSKIN_FRONT_SYM_ANC_SPRING ■ SUSKIN_FRONT_SYM_ANC_SPRING_3DOF ■ SUSKIN_RIGID_SYM_FRONT ■ SUSKIN_REAR_ASYM_3DOF ■ SUSKIN_REAR_ASYM_ANC_SPRING ■ SUSKIN_REAR_ASYM_ANC_SPRING_3DOF ■ SUSKIN_REAR_SYM_3DOF ■ SUSKIN_REAR_SYM_ANC_SPRING ■ SUSKIN_REAR_SYM_ANC_SPRING_3DOF ■ SUSKIN_RIGID_SYM_REAR ■ SUSFORCES_FRONT_ANC_SPRING ■ SUSFORCES_REAR_ANC_SPRING ■ SUSPENSION_FORCES_ACTIVE_FRONT ■ SUSPENSION_FORCES_ACTIVE_REAR ■ SUSPENSION_FORCES_FRONT, SUSPENSION_FORCES_FRONT_1_0 ■ SUSPENSION_FORCES_REAR, SUSPENSION_FORCES_REAR_1_0 ■ SUSPENSION_KINEMATICS_FRONT_ASYMMETRIC ■ SUSPENSION_KINEMATICS_FRONT_SYMMETRIC ■ SUSPENSION_KINEMATICS_FRONT_1_0 ■ SUSPENSION_KINEMATICS_REAR_ASYMMETRIC ■ SUSPENSION_KINEMATICS_REAR_SYMMETRIC ■ SUSPENSION_KINEMATICS_REAR_1_0
TIRE_MODEL_TMEASY_*	摩擦係数がゼロの場合は、ゼロ除算が回避されます。
ENGINE	エンジントルクパラメータのサイズが、[30x30]から[31x30]に拡張されています。以前のパラメータは、`asmmigratepost` で外挿され、初期化されます。 エンジントルクのルックアップテーブル Map_EffectiveEngineTorque の名前は、Map_Trq_MeanEff_Engine[Nm]に変更されています。
CRANK_SHAFT	新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
SHAFT_*	入力ポートおよび出力ポートの名前はモジュール構成で作成されています。 ASMSignalBus にその他の信号が追加されました。

CRANK_SHAFT_RIGID	新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
SHAFT_RIGID	新しいパラメータが、回転抑制を記述するために追加されています。 新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
CLUTCH_4WD	このブロックは、GoToFrom ブロックの代わりに入力ポートを介してリアディファレンシャルの主減速比を取得します。 新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
REAR_DIFFERENTIAL	CLUTCH_4WD ブロックでの使用を目的とする主減速比のグローバルな GoToFrom 接続は削除されました。 新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
DRIVETRAIN_VARIANT_SWITCHES	FF4WD バリアントのドライブトレインモードのコメントが、[1FourWheel|2FrontWheel|3RearWheel]から[1AWD|2FWD|3RWD|4FF-4WD|5FF]に拡張されています。
SOFT_ECU_TRANSMISSION_10_0	このブロックは、自動的に移行することはできません。そのため、移行時に、ブロックへのリンクは以前のバージョン: `FormerVersions/SOFT_ECU_TRANSMISSION_10_0` に変更されます。
LOCKUP_CLUTCH	被駆動プレートのイナーシャパラメータは、マスクパラメータに転送されます。 新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
CLUTCH	新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
CENTRAL_DIFFERENTIAL	新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。
REAR_DIFFERENTIAL	新しい入力ポートが、エンジン回転数積分器の外部初期化のために追加されています。

Bus Manager（スタンドアロン）

Bus Manager（スタンドアロン）5.6 の機能

SAE J2602 のサポート	Bus Manager（スタンドアロン）は、SAE J2602 規格に準拠した LIN 通信をサポートします。SAE J2602 に準拠した通信マトリクスをインポートし、J2602 に準拠した ECU をシミュレートすることができます。ただし、J2602 に準拠した LIN 通信には、いくつかの制限事項が適用されます。詳細については、「Limitations for LIN Communication」(🕮『Bus Manager (Stand-Alone) Implementation Guide』)を参照してください。
FIBEX 4.1.1 のサポート	Bus Manager（スタンドアロン）は、通信マトリクスとして FIBEX 4.1.1 に基づく FIBEX ファイルをサポートします。

ConfigurationDesk

目的

ConfigurationDesk では、それぞれの用途に適した 2 つのバージョンを利用することができます。リアルタイムアプリケーションを実装するには、ConfigurationDesk - Implementation Version を使用することができます。dSPACE RapidPro ハードウェアを設定するには、ConfigurationDesk - Configuration Version を使用することができます。

ConfigurationDesk – Implementation

項目の一覧

本章の内容

ConfigurationDesk 5.6（Implementation Version）の新機能

BSC ファイルのサポート

ConfigurationDesk では、バスシミュレーションコンテナファイル（BSC ファイル）を ConfigurationDesk アプリケーションに追加することができるようになりました。BSC ファイルには、アプリケーションプロセスの設定済みのバス通信が含まれています。これらのファイルは、Bus Manager で生成されます。「Working With Bus Simulation Containers」(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

[Add model]ダイアログの改善

ConfigurationDesk の[Add model]ダイアログが改善されました。機能拡張は、次のとおりです。

- ConfigurationDesk アプリケーションに追加される複数のモデルを同時に選択することができます。
- MATLAB で開いている Simulink モデルは、リストに表示されます。これらのモデルは、パスを参照することなく ConfigurationDesk アプリケーションに直接追加することができます。

「Add Model」(🕮『ConfigurationDesk Real-Time Implementation Reference』)を参照してください。

タスクの拡張

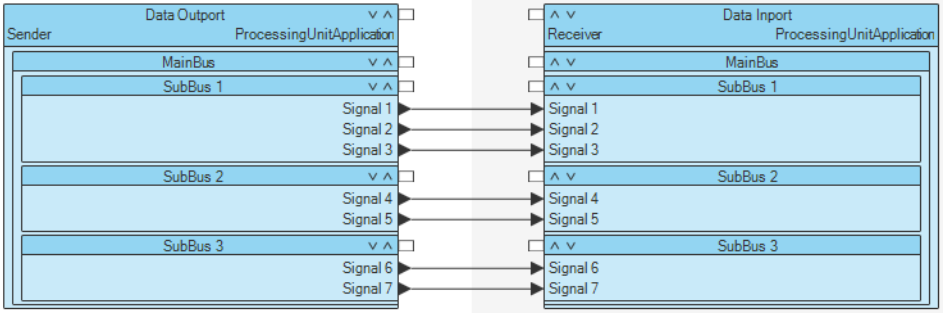
No jitter, low latency　タスクでは、タスクをポーリングモードで実行できる[Jitter and latency optimization]プロパティを利用することができます。これにより、リアルタイムアプリケーションのサンプル時間を短くすることができます。タスクを[No jitter, low latency]として設定するには、そのタスクがアプリケーションプロセスでの唯一のタスクであり、タイマーイベントによってトリガされる必要があります。タスクを[No jitter, low latency]として設定する前に、この機能の制限事項を熟知しておく必要があります。「How to Configure Tasks in ConfigurationDesk」

(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

タスクのトリガ　ConfigurationDesk を使用して、複数の I/O イベントによってタスクをトリガすることができます。「Basics on Modeling Asynchronous Tasks」(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

モデル通信のパフォーマンスの改善

2 つのモデルポートブロックが同じ構造で、それらのポートが下図に示すように一対一で接続されていると仮定します。



この場合、ConfigurationDesk はビルドプロセス時に最適化されたコードを生成し、これによりリアルタイムアプリケーションのパフォーマンスが改善します。

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、サポートされる 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 2016-B: ■ SystemDesk 4.7 ■ TargetLink 4.2	2.4.1
dSPACE Release 2016-A: ■ SystemDesk 4.6	2.4
dSPACE Release 2015-B: ■ SystemDesk 4.5 ■ TargetLink 4.1	2.3
dSPACE Release 2015-A: ■ SystemDesk 4.4	2.2
dSPACE Release 2014-B: ■ SystemDesk 4.3 ■ TargetLink 4.0	2.1

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 3.2 ■ TargetLink 3.5	1.0

アップデートされた QNX コンパイラのサポート

ConfigurationDesk は、64 ビット Windows に対応した QNX コンパイラバージョン 5.2.0 をサポートします。これにより、非常に大きな Simulink モデルでも Out-of-Memory エラーを発生せずにコンパイルすることができます。

ECU インターフェースのサポート

SCALEXIO システムとの ECU インターフェースを実行することができます。

このためには、ECU インターフェースコンテナ(EIC)ファイル(ECU Interface Manager で生成)を ConfigurationDesk アプリケーションにインポートする必要があります。EIC ファイルは、ECU インターフェースのために準備された ECU アプリケーションを記述します。ConfigurationDesk では、ECU アプリケーションの準備された部分をシグナルチェーンに統合し、SCALEXIO システムに対応したリアルタイムアプリケーションをビルドすることができます。

サポートされている ECU インターフェース SCALEXIO との ECU インターフェースを実行するには、ターゲット ECU が、以下の ECU インターフェースのいずれかを介して SCALEXIO システムの Ethernet アダプタに接続されている必要があります。

- DCI-GSI2
- XCP on Ethernet

詳細については、「ECU Interfacing with SCALEXIO Systems」(🕮『ConfigurationDesk Real-Time Implementation Guide』)を参照してください。

新しいファンクションブロックタイプ

Ethernet のセットアップ Ethernet Setup ファンクションブロックでは、SCALEXIO ハードウェアの Ethernet アダプタへのアクセスを設定し、初期化することができます。

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

ECU Interface の設定 ECU Interface Configuration ファンクションブロックは、リアルタイムアプリケーション(SCALEXIO 上で実行)と ECU アプリケーション(ECU 上で実行)との間のデータのやり取りを設定し、ECU インターフェースを可能にします。

詳細については、「ECU Interface Configuration」(🕮『ConfigurationDesk I/O Function 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

SENT In, SENT Out SENT In および SENT Out ファンクションブロックでは、以下の拡張を利用することができます。

- ユーザ定義のプロトコルファイルを作成およびインポートし、これに代わる SENT プロトコルをサポートします。「Creating User-Defined Protocol Files」(📖『ConfigurationDesk I/O Function Implementation Guide』)を参照してください。
- SAE J2716 APR2016 SENT 規格に基づく、新たに事前定義されたプロトコルは、温度センサや位置／レシオセンサなどのより多くのセンサをサポートします。あらかじめ定義されたすべてのプロトコルの概要については、「Using Application-Specific Protocols」(📖『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が、CAN 通信に関する ISO 11898-6 に準拠したパーシャルネットワーキングをサポートするようになりました。割り当てられたハードウェアリソースの CAN トランシーバは、有効化されると低電力モードと同じモード(スタンバイ、スリープ、サイレント、ノーマル)で動作することができます。低電力モードに比べ、CAN トランシーバは有効なウェイクアップフレームを受信した場合にのみウェイクアップします。CAN ファンクションブロックのプロパティを介して、有効なウェイクアップフレームを指定することができます。

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

Bus Manager の新機能は、次のとおりです。

- SAE J2602 のサポート

  Bus Manager は、SAE J2602 規格に準拠した LIN 通信をサポートします。SAE J2602 に準拠した通信マトリクスをインポートし、J2602 に準拠した ECU をシミュレートすることができます。ただし、J2602 に準拠した LIN 通信には、いくつかの制限事項が適用されます。詳細については、「Limitations for LIN Communication」(📖『ConfigurationDesk Bus Manager Implementation Guide』)を参照してください。

- FIBEX 4.1.1 のサポート

  Bus Manager は、通信マトリクスとして FIBEX 4.1.1 に基づく FIBEX ファイルをサポートします。

ConfigurationDesk 5.6 への移行

dSPACE Release 2016-B 以降でのプラットフォーム管理自動化 API バージョン 1.0 の廃止	プラットフォーム管理自動化 API バージョン 1.0 のサポートは、dSPACE Release 2016-A の ConfigurationDesk 5.5 で最後となります。「Automating Platform Management」(🕮『ConfigurationDesk Automating Tool Handling』)を参照してください。
Ethernet アダプタの不整合	プロジェクトを dSPACE Release 2016-B に移行する場合、移行プロセスで SCALEXIO Real-Time PC の Ethernet アダプタが移行するプロジェクトのハードウェアトポロジに追加されます。追加される Ethernet アダプタのデフォルト名は、"[MAC address 00:00:00:00:00:00] no name assigned"です。この名前は、アクセス可能なプラットフォームの Ethernet アダプタの名前と一致しません。したがって、ステータスバーに"No matching platform connected"というステータスが表示されます。このステータスがあると、ビルド後のリアルタイムアプリケーションをただちにダウンロードすることができません。 この不整合を解決するには、Hardware Resource Browser および Platform Manager で、SCALEXIO Real-Time PC の Ethernet アダプタと同一の名前を指定します。

ControlDesk

項目の一覧

本章の内容

ControlDesk の新機能（ControlDesk 6.0）

項目の一覧

本章の内容

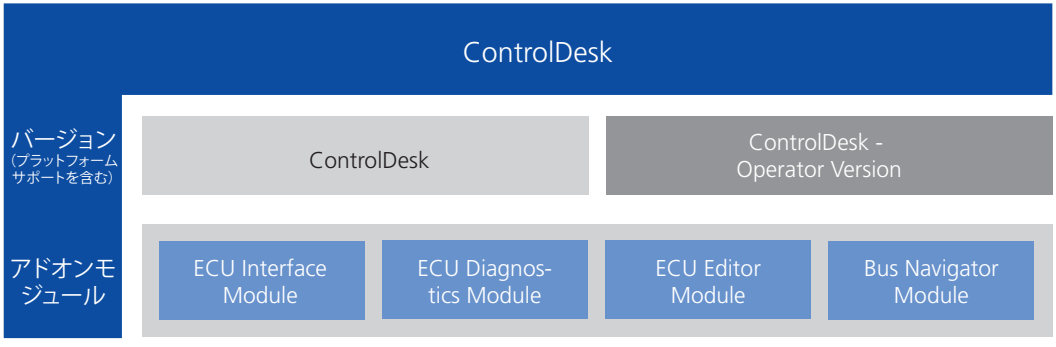
新しい一般機能（ControlDesk 6.0）

64 ビットアプリケーション

ControlDesk は、64 ビットアプリケーションとして実装されています。

ControlDesk の新しいパッケージ

ControlDesk のパッケージが変更されています。次の図に、使用可能なバージョンとモジュールの概要を示します。



「Software Versions and Modules」(🕮『ControlDesk Introduction and Overview』)を参照してください。

プロジェクトおよびエクスペリメントの新機能（ControlDesk 6.0）

フォルダおよびフォルダグループのソート

ControlDesk を使用して、Project Manager でアルファベットの昇順または降順にフォルダおよびフォルダグループをソートすることができます。

下記を参照してください。

- 「Sort - Ascending」(🕮『ControlDesk Project and Experiment Management』)
- 「Sort - Descending」(🕮『ControlDesk Project and Experiment Management』)

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 6.0）

本章の内容

新しい DCI-CAN/LIN1 のサポート

ControlDesk は、新しい DCI-CAN/LIN1 をサポートしています。

CAN サポート　DCI-CAN/LIN1 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-CAN/LIN1 は、ControlDesk の次の CAN ベース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
- ECU 診断

LIN のサポート　DCI-CAN/LIN1 により、ホスト PC を LIN ネットワークに接続することができます。

新しい DCI-CAN/LIN1 は、ControlDesk の次の LIN ベースデバイスで使用することができます。

- LIN バスモニタリング

新しい DCI-CAN/LIN1 の詳細については、🕮『DCI-CAN/LIN1 Feature Reference』を参照してください。

SCALEXIO プラットフォーム：Ethernet および PCI/PCIe カードの表示

ControlDesk の Platform/Device Manager は、SCALEXIO システムに接続されている Ethernet カードやサポートされている PCI/PCIe カードを表示します。

SCALEXIO プラットフォーム：接続された未登録のシステムの表示

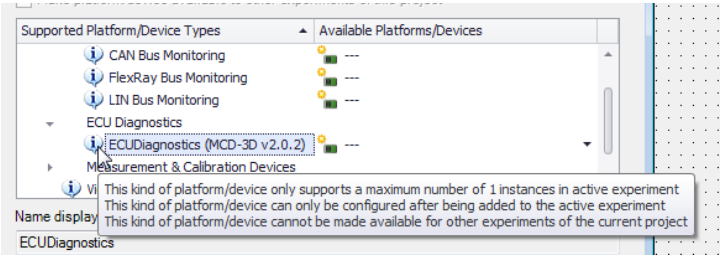
ControlDesk の Platform/Device Manager は、接続されている未登録の SCALEXIO システムを表示します。

「Uplink and Downlink Connections Properties」(🕮『ControlDesk Platform Management』)を参照してください。

プラットフォーム／デバイスを追加する場合の重要な追加情報の表示

特定のプラットフォーム／デバイスタイプのプラットフォーム／デバイスを追加するときに、重要な追加情報が存在する場合（デバイスの設定はアクティブなエクスペリメントに追加された後にのみ可能など）は、ControlDesk はプラットフォーム／デバイスタイプの横に記号を表示します。マウスポインタを記号に合わせると、詳細情報がツールチップで表示されます。

下の図に例を示します。

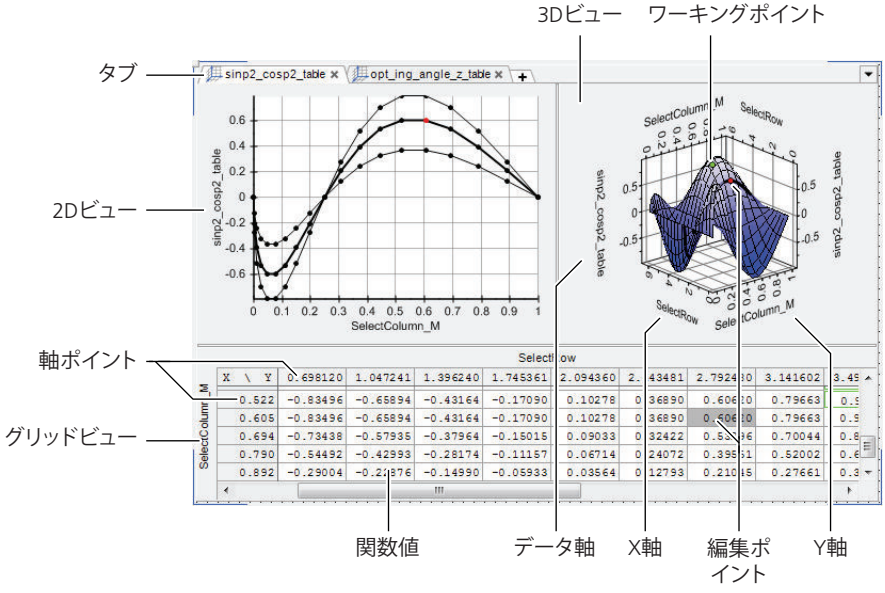


「Insert Platform / Add Platform/Device」(🕮『ControlDesk Platform Management』)を参照してください。

新しい計器機能（ControlDesk 6.0）

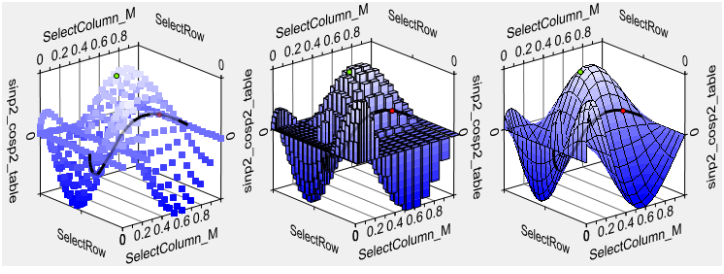
Table Editor の拡張

ControlDesk では、拡張された Table Editor を利用することができます。



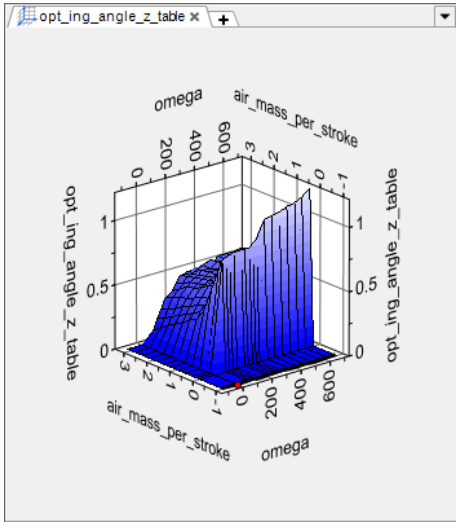
最も重要な拡張は次のとおりです。

3D ビューで選択可能なチャートタイプ　3D ビューの異なるチャートタイプ（スキャッタプロット、棒グラフ、サーフェス）を選択できます。

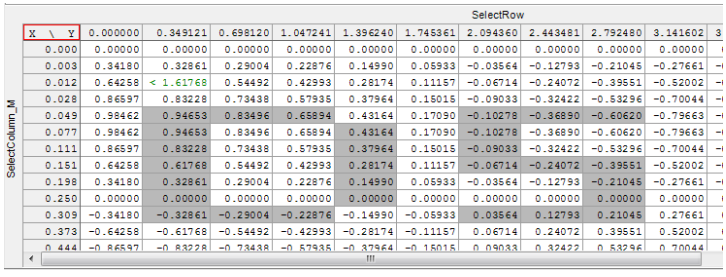


ビュー表示の有効化／無効化　Table Editor を使用して、ビューの Visible プロパティを指定することにより、各ビュー（3D ビュー、2D ビュー、グリッドビュー）を個別に有効化または無効化することができます。

次の図は、2D およびグリッドビューの表示を無効にした Table Editor を示します。



編集ポイントの複数選択の拡張　Table Editor で任意の編集ポイントを複数選択することができます。下の図を参照してください。



2D および 3D ビューの設定の拡張　2D および 3D ビューの設定がさらに便利になりました。

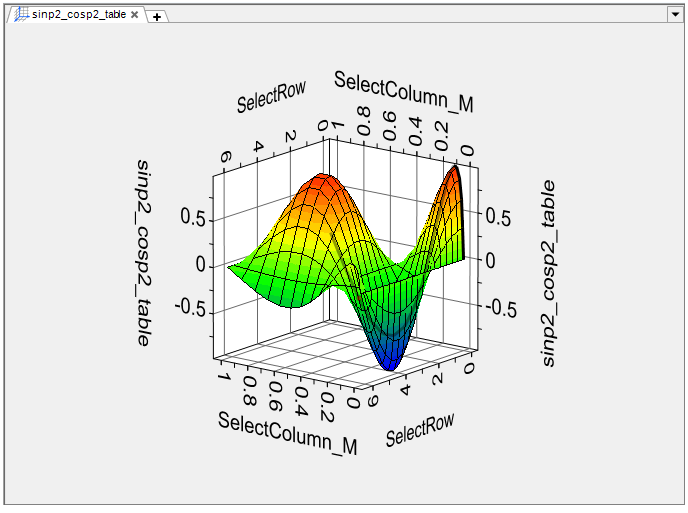
たとえば、次のような操作が可能です。

- マウスホイールを使ってビューを拡大
- マウスの左ボタンを使って 3D ビューを回転
- マウスの左ボタンを使って 2D ビューを移動
- ダブルクリックでビューを再スケーリング

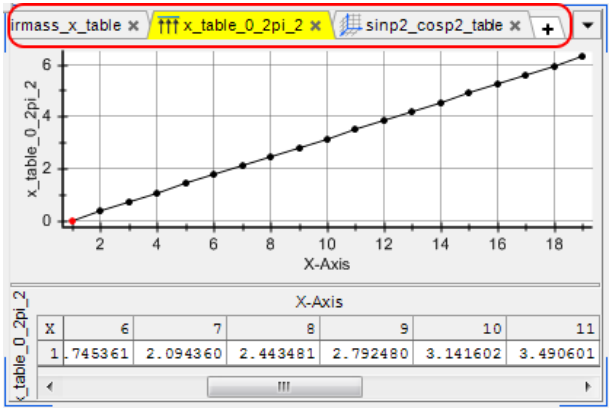
詳細については、「Table Editor のビューをセットアップする方法」(🕮『ControlDesk 計器の操作』)を参照してください。

カラーサポートの拡張　Table Editor では、カラーサポートが拡張されています。

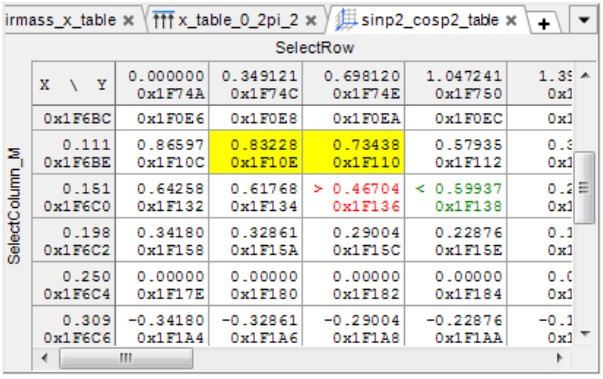
- 3D ビューで表面の色を指定することができます。下の図を参照してください。



- 各ビューで個々の背景色と背景画像を個別に指定することができます。
- Table Editor の各タブで個々の背景色とテキスト色を指定することができます。下の図を参照してください。



■ グリッドビューで編集ポイントの背景色を指定することができます。下の図を参照してください。



「Table Editor の操作の基礎」(🕮『ControlDesk 計器の操作』)を参照してください。

移行上の注意点については、「Migrating from ControlDesk 5.6 to 6.0」(🕮『ControlDesk Introduction and Overview』)を参照してください。

Instrument Selector の拡張

ControlDesk の Instrument Selector では、次の機能が強化されています。

■ Instrument Selector に対する計器カテゴリの追加／削除が可能です。これにより、必要に応じて Instrument Selector を構築することができます。

■ Instrument Selector でカスタム計器を元のカスタム計器にドラッグすることにより、簡単に変更を適用することができます。ControlDesk は、元のカスタム計器を上書きするかどうかを確認します。

■ カスタム計器を*任意*の計器カテゴリに Instrument Selector で追加することができます。

■ 関連する計器カテゴリをエクスポートすることにより、カスタム計器を交換することができます。

「Instrument Selector を設定する方法」(🕮『ControlDesk 計器の操作』)を参照してください。

新しい STRUCT 計器キャプションマクロ

ControlDesk では、%STRUCT%計器キャプションマクロを利用することができます。このマクロを使用して、構造体の該当する部分とともに(変数が構造体の一部である場合)変数の名前を表示することができます。構造体の各部分はドットで区切られています。たとえば、次のようになります。MyStruct.MySubstruct.MyParameter

「Captions/Operating Elements Properties」(🕮『ControlDesk 計器の操作』)を参照してください。

新しい計測機能および記録機能（ControlDesk 6.0）

大規模な ASAM MDF 4.1 ファイルの処理の改善

ControlDesk では、大規模な ASAM MDF 4.1（MF4）ファイルの処理が改善されています。

信号のロードの改善　ControlDesk は、計器が現在必要とするデータのみをロードします。つまり、ControlDesk は MF4 ファイルに保存されているすべての信号をロードするのではなく、たとえば、時間プロッタにビジュアル表示される信号部分のみをロードします。これによりロード時間が短縮され、必要なメモリが少なくて済みます。

リダクションデータのサポート　ControlDesk では、ASAM MDF 4.1 ファイルの *リダクションデータ* がサポートされています。

リダクションデータとは、ビジュアル表示の解像度に応じて MF4 ファイルデータをビジュアル表示することを可能にする、MF4 ファイルの追加コンテンツのことです。したがって、リダクションデータは計測データのビジュアル表示と後処理を改善します。

- 時間プロッタなどの ControlDesk の計器は、リダクションデータを含む MF4 ファイルをサポートしています。
- ControlDesk には、計測データが ASAM MDF 4.1 ファイル形式に保存された場合の *リダクションデータ* が含まれています。

計測データファイルから変数を削除する

ControlDesk を使用して、Measurement Data Pool で開いた計測データファイルから 1 つまたは複数の変数を削除することができます。

「Remove Variables」(🕮『ControlDesk 計測および記録』)を参照してください。

新たに追加された dSPACE プラットフォームでの連続計測（新しいデフォルト設定）

ControlDesk 6.0 以降では、*連続（トリガなし）計測* の実行が、エクスペリメントに新たに追加された dSPACE プラットフォームのデフォルト動作です。ControlDesk 5.6 以前では、*トリガ計測* の実行が新たに追加された dSPACE プラットフォームのデフォルト動作でした。

新たに追加された dSPACE プラットフォームのデフォルト設定は、Measurement Configuration Page で変更することができます。

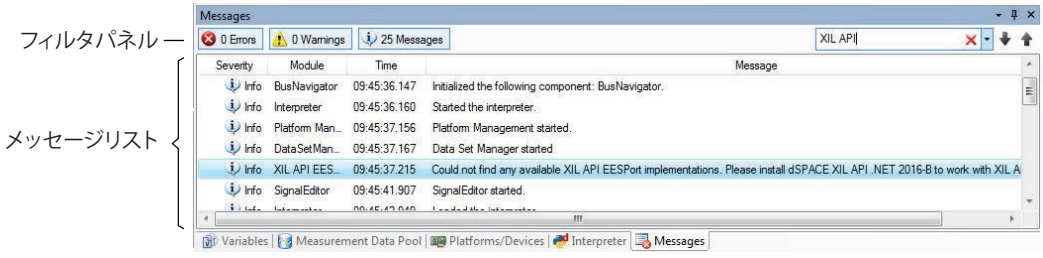
「Measurement Configuration Page」(🕮『ControlDesk 計測および記録』)を参照してください。

メッセージ処理の新機能（ControlDesk 6.0）

新しい Message Viewer

ControlDesk は新しい Message Viewer をサポートしています。Message Viewer を使用して、製品の使用中に発生した情報メッセージ、助言メッセージ、エラーメッセージ、警告メッセージ、質問のすべての履歴を表示することができます。システムの状態をチェックするのに役立ちます。

Message Viewer は、次のようになります。



「Messages」(🕮『ControlDesk メッセージ処理』)を参照してください。

dSPACE ログの表示

ControlDesk を使用して dSPACE ログを表示することができます。

dSPACE ログは、すべての dSPACE 製品および接続されたシステムによって発行されたエラーや警告、情報、質問、および助言を複数回のセッションにわたって記録したものです。

dSPACE ログを dSPACE サポートに転送するには、dSPACE Installation Manager を使用します。

「dSPACE Log」(🕮『ControlDesk メッセージ処理』)を参照してください。

Bus Navigator の新機能（ControlDesk 6.0）

バス計器（FlexRay 用の TX タイプ）：最小遅延時間のサポート

計器を使用して*最小遅延時間*を表示することができます。AUTOSAR システムデスクリプションファイルで定義されている PDU の最小遅延時間は、新しい PDU データの継続する送信間の最小遅延時間を指定します。

バス計器（FlexRay 用の TX タイプ）については、「Bus Instrument (TX Type for FlexRay)」(🕮『ControlDesk Bus Navigator』)を参照してください。

バス信号のロード／アンロード

ControlDesk の Bus Navigator では、バス信号をロード／アンロードできるようになりました。この機能は、メモリ例外を起こす可能性のある多くのバス信号を含むバス設定に役に立ちます。

ControlDesk を使用して、Bus Navigator Page にバス信号をロードする場合のデフォルトの起動動作を指定することができます。

「Bus Navigator Page」(🕮『ControlDesk Bus Navigator』)を参照してください。

Bus Navigator 内のバス信号をロード／アンロードするには、以下の新しいコマンドを使用します。

- 「Load All Bus Signals」(🕮『ControlDesk Bus Navigator』)
- 「Unload all Bus Signals」(🕮『ControlDesk Bus Navigator』)

ECU 診断の新機能（ControlDesk 6.0）

DoIP（Diagnostics over Internet Protocol）のサポート

ControlDesk の ECU 診断デバイスでは、DoIP（Diagnostics over Internet Protocol）もサポートされます。DoIP は、UDS 用の転送プロトコルです。

DoIP プロトコルでの Ethernet ベースの UDS では、ControlDesk はホスト PC の使用可能な Ethernet インターフェースを使用して ECU にアクセスします。その他のインターフェースモジュールは必要ありません。

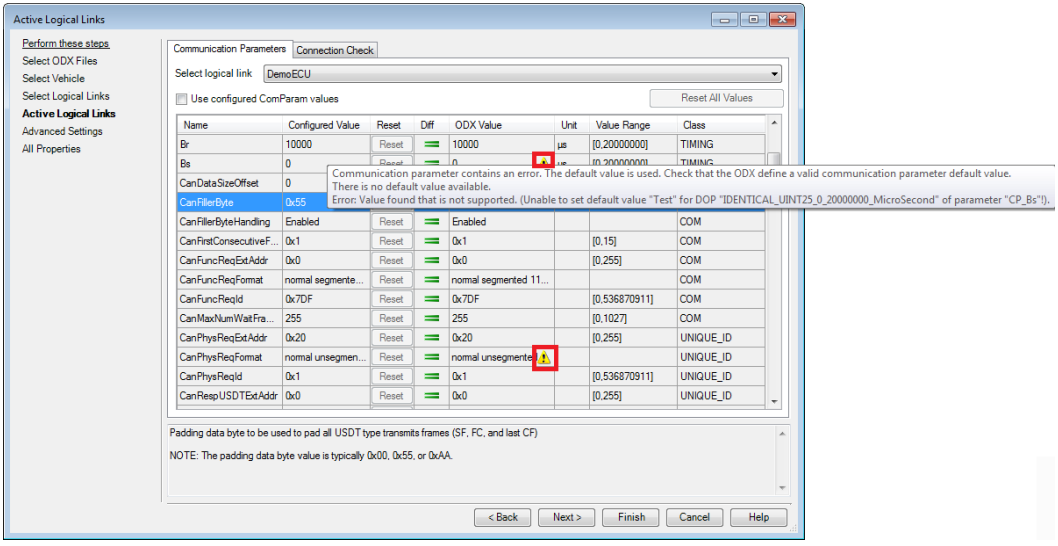
ECU 診断デバイスとの接続に DoIP プロトコルで UDS を使用する場合の詳細については、「ODX データベースとの接続規約」(🕮『ControlDesk ECU 診断』)を参照してください。

デフォルトの COMPARAM 値が使用される場合の情報

ControlDesk のバージョン 5.6 以前では、ODX データベースで定義された COMPARAM 値にエラーが含まれている場合、ControlDesk は暗黙的にデフォルトの通信パラメータ（COMPARAM）を使用します。ただし、ControlDesk はこの動作に関する情報を表示しませんでした。

ControlDesk のバージョン 6.0 以降では、間違った COMPARAM 値が⚠警告記号とともに[Active Logical Links - Communication Parameters]ページに表示されます。さらに、ControlDesk では、間違ったパラメータの代わりにデフォルトの通信パラメータが使用されたことを示すツールチップが表示されます。

下の図に例を示します。



「Configure Platform/Device」(🕮『ControlDesk Platform Management』)を参照してください。

新しい電気的欠陥シミュレーション機能(ControlDesk 6.0)

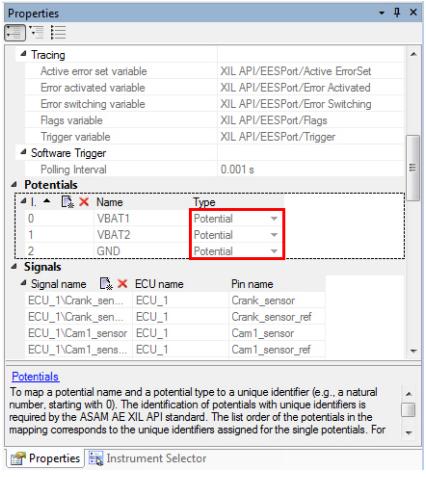
ソフトウェアトリガのサポート

ControlDesk では、ソフトウェアトリガを介してエラーセットをアクティブ化できるようになりました。これにより、定義されたトリガ条件または持続時間に応じて、エラーセットをアクティブ化することができます。ControlDesk を使用して、ASAM General Expression Syntax(GES)に従ってトリガ条件を設定することができます。

「Basics on Software Triggers」(🕮『ControlDesk Electrical Error Simulation via XIL API EESPort』)および「How to Configure a Software Trigger」(🕮『ControlDesk Electrical Error Simulation via XIL API EESPort』)を参照してください。

自動電位マッピング

EESPort を作成する場合、ControlDesk は、選択された欠陥シミュレーションハードウェアに提供される電位名に基づいて、あらかじめ設定された電位マッピングを自動的に作成します。デフォルトで、すべての電位の[Type]は[Potential]に設定されます。次の図を参照してください。



[Properties]コントロールバーを使用して、あらかじめ設定された電位マッピングを後で変更することができます。

「Basics on Potential Mapping」(🕮『ControlDesk Electrical Error Simulation via XIL API EESPort』)を参照してください。

自動シグナルマッピング

EESPort を新規作成する場合、dSPACE シミュレータに指定されている ECU 名とピン名に基づいてデフォルトのシグナルマッピングを ControlDesk に生成させることができます。

「Basics on Signal Mapping」(🕮『ControlDesk Electrical Error Simulation via XIL API EESPort』)を参照してください。

変数のドラッグアンドドロップのサポート

ControlDesk の電気的欠陥シミュレーションで変数のドラッグアンドドロップがサポートされます。ソフトウェアトリガのトリガ条件にトリガ変数を指定する場合、Variable Browser から[Properties]コントロールバーなどに変数をドラッグすることができます。

Signal Editor の新機能（ControlDesk 6.0）

eLINEAR 補間メソッドのサポート

ControlDesk の Signal Editor は、Data File セグメントおよび Signal Value セグメントでの eLINEAR 補間メソッドをサポートします。

	「Interpolation Property」(🕮『ControlDesk Signal Editor』)を参照してください。
ダウンロード中の Data File セグメント表示の自動更新	ControlDesk は、関連するシグナルジェネレータをダウンロードしたときに Data File セグメントの画面表示を自動的に更新します。 「Data File (Segment)」(🕮『ControlDesk Signal Editor』)を参照してください。

ControlDesk の移行（ControlDesk 6.0）

項目の一覧

本章の内容

ControlDesk での廃止

本章の内容

ControlDesk 6.0 での廃止項目

ControlDesk Failure Simulation Module　ControlDesk の Failure Simulation Module は、dSPACE Release 2016-A の ControlDesk 5.6 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 の移行（ControlDesk 6.0）」(139 ページ)を参照してください。

プラットフォーム管理自動化 API バージョン 1.0　プラットフォーム管理自動化 API バージョン 1.0 のサポートは、dSPACE Release 2016-A の ControlDesk 5.6 で最後となります。

API バージョン 2.0 への移行（dSPACE Release 2014-A の ControlDesk 5.2 から導入）については、「Migrating from ControlDesk 5.1 to 5.2」(🕮『ControlDesk Introduction and Overview』)を参照してください。

プロッタ　プロッタは dSPACE Release 2016-A の ControlDesk 5.6 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。

- インデックスプロッタ
- 時間プロッタ
- XY プロッタ

さまざまなプロッタタイプの相違点については、「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 計器の操作』)を参照してください。

移行上の注意点については、「Migrating from ControlDesk 5.6 to 6.0」(🕮『ControlDesk Introduction and Overview』)を参照してください。

統合された Variable Editor　ControlDesk では、統合コンポーネントとしての Variable Editor の提供を終了しました。

ただし、Variable Editor をスタンドアロンツールとして使用することはできます。スタンドアロンの Variable Editor は、http://www.dspace.jp/go/requestreleasedownload で入手することができます。

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
ControlDesk 6.0 以降では、MDF 計測データファイル（MDF ファイル形式バージョン 2.0 および 3.0）のエクスポートがサポートされません。

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 計測および記録』)を参照してください。

メッセージ処理の方法　dSPACE Release 2016-B では、すべての dSPACE 製品でエラーや警告などのメッセージの取り扱いに改善された方法が使用されます。

その結果、次のような利点があります。

- メッセージが dSPACE.log ファイルに書き込まれなくなり、メッセージをプレーンテキストとして利用できなくなりました。

  ログメッセージを含む診断情報を収集し、dSPACE サポートに送信するには、*dSPACE Installation Manager を使用します*。
- ControlDesk 6.0 では、dSPACE Message Service で記録されたログメッセージをモニタリングできる dSPACE Message Monitor が削除されました。
- ControlDesk 6.0 では、メッセージログファイルのフルパス名を取得する Log / ILoLog <<Interface>>の LogFilePath プロパティが削除されました。

メッセージ処理方法の改善については、🕮『ControlDesk メッセージ処理』を参照してください。

ControlDesk 3.x エクスペリメントの移行　dSPACE Release 2016-A の ControlDesk 5.6 は、ControlDesk 3.x のエクスペリメントを ControlDesk で再利用するための移行をサポートする最後のバージョンとなります。このことは、ControlDesk 3.x のレイアウトのインポートにも適用されます。

CalDesk プロジェクトの移行　dSPACE Release 2016-A の ControlDesk 5.6 は、CalDesk のプロジェクトを ControlDesk で再利用するための移行をサポートする最後のバージョンとなります。このことは、CalDesk のレイアウトのインポートにも適用されます。

dSPACE Release 2017-A 以降の ControlDesk で廃止予定の項目

PAR ファイルのインポート　dSPACE Release 2017-A 以降では、ControlDesk では ControlDesk 3.x で作成された PAR ファイルのインポートがサポートされなくなります。

ControlDesk の移行（ControlDesk 6.0）

ControlDesk 5.6 から ControlDesk ControlDesk 6.0 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。

注記

5.6 より前のバージョンから ControlDesk 6.0 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

Failure Simulation Module：廃止項目および移行

ControlDesk の Failure Simulation Module は、dSPACE Release 2016-A の ControlDesk 5.6 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。

  - ハードウェ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、「How to Create a New EESPort」(🕮『ControlDesk Electrical Error Simulation via XIL API EESPort』)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、電気的欠陥を作成および設定することができます。「How to Create and Configure an Electrical Error」(🕮『ControlDesk Electrical Error Simulation via XIL API EESPort』)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
5.6	2016-A	2016-A

`FailureSimulationExportTool` とユーザドキュメントを含む ReadMe ファイルを入手するには、dSPACE サポートにお問い合わせください。

- 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。

プロッタ:廃止項目および移行

プロッタは dSPACE Release 2016-A の ControlDesk 5.6 に付属するものが最後となります。

- ControlDesk 6.0 以降を使用して、エクスペリメントをプロッタとともに開くことができます。プロパティと変数接続を含むプロッタ計器は、エクスペリメントの移行時に自動的に時間プロッタ(「時間プロッタの使用」(🕮『ControlDesk 計器の操作』)を参照)計器に移行します。
- 時間プロッタは、時間ベースのラスタで計測される信号を表示するためのプロッタの後継バージョンで、新しい自動化コンセプトを提供します。その結果、プロッタを自動化するために開発された自動化スクリプトは、時間プロッタの自動化には再利用することができません。

Table Editor

ControlDesk 6.0 には、拡張された Table Editor が付属しています。

- ControlDesk 6.0 以降を使用して、ControlDesk 5.6 以前で作成された Table Editor を使用するエクスペリメントを開くことができます。これらの計器では、エクスペリメントの移行時にプロパティおよび変数接続が維持されます。

■ ControlDesk 5.6 以前の Table Editor の自動化のために開発された自動化スクリプトは、ControlDesk 6.0 以降でも再利用することができます。ただし、Table Editor *配列*の自動化に関連して移行上の注意点があります。「Table Editor：Table Editor 配列作成の変更」(144 ページ)を参照してください。

dSPACE プラットフォーム上での計測のデフォルト動作の変更

ControlDesk 5.6 以前では、「Measurement Configuration Page」(📖『ControlDesk 計測および記録』)の[Measure continuously on newly added dSPACE platforms]オプションはインストール後には非アクティブで、新たに追加された dSPACE プラットフォームの計測は、デフォルトで*[triggered]*に設定されます。

ControlDesk 6.0 以降では、[Measure continuously on newly added dSPACE platforms]オプションはインストール後にアクティブで、新たに追加された dSPACE プラットフォームの計測は、デフォルトで*[continuous]*に設定されます。

A2L ファイルのインポートの変更

ControlDesk 6.0 での A2L ファイルのインポートは、次の点が変更されています。

■ バージョン 5.6 以前の ControlDesk では、計測配列の LAYOUT フラグで COLUMN_DIR 設定はサポートされません。インポートする A2L ファイルに計測配列などが含まれる場合、ControlDesk では、計測配列の LAYOUT フラグで代わりに ROW_DIR 設定をサポートします。

■ バージョン 6.0 以降の ControlDesk では、計測配列の LAYOUT フラグで COLUMN_DIR 設定がサポートされています。インポートする A2L ファイルに計測配列が含まれる場合、ControlDesk は COLUMN_DIR 設定を使用します。

ECU 診断：ECU 診断ジョブで呼び出される DLL

診断ジョブは、Seed&Key DLL などの DLL を呼び出し、ECU パラメータ適合のためのセキュリティアクセスを取得することができます。

バージョン 6.0 以降では、ControlDesk は ECU 診断ジョブで呼び出された 64 ビット DLL のみをサポートします。ECU 診断ジョブで呼び出された 32 ビット DLL はサポートされません。したがって、これらの DLL は再コンパイルする必要があります。

Signal Editor：eBACKWARD 補間方法のインプリメンテーションの変更

Data File セグメントおよび Signal Value セグメントでの `eBACKWARD` 補間方法に使用されるインプリメンテーションが、次のように変更されています。

■ ControlDesk 5.6 および Real-Time Testing 3.0（dSPACE Release 2016-A）以前では、Real-Time Testing の `RM_SAMPLED` データストリーミングモードが使用されます。

- ControlDesk 6.0 および Real-Time Testing 3.1（dSPACE Release 2016-B）以降では、Real-Time Testing の RM_BACKWARD データストリーミングモードが使用されています。このモードは、ASAM AE XIL API 規格で定義された eBACKWARD 補間方法の仕様に準拠しています。

2 つの補間タイプの相違点については、「MatFile Class Description」(📖『Real-Time Testing Library Reference』)を参照してください。

ツール自動化の変更

Failure Simulation Module：廃止項目および移行 ControlDesk の Failure Simulation Module は、dSPACE Release 2016-A の ControlDesk 5.6 に付属するものが最後となります。

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。

プラットフォーム管理自動化 API バージョン 1.0 の廃止 プラットフォーム管理自動化 API バージョン 1.0 のサポートは、dSPACE Release 2016-A の ControlDesk 5.6 で最後となります。

API バージョン 2.0 への移行（dSPACE Release 2014-A の ControlDesk 5.2 から導入）については、「Migrating from ControlDesk 5.1 to 5.2」(📖『ControlDesk Introduction and Overview』)を参照してください。

32 ビットのサードパーティ製 Python 拡張モジュールのサポートなし
バージョン 6.0 では、ControlDesk の内部インタープリタは 64 ビットのサードパーティ製 Python 拡張モジュールのみをサポートします。32 ビットのサードパーティ製 Python 拡張モジュールはサポートされません。

IXaMeasurementDataManagement インターフェースへの変更
ControlDesk 6.0 以降では、MDF 計測データファイル（MDF ファイル形式バージョン 2.0 および 3.0）のエクスポートがサポートされません。

したがって、MeasurementDataManagement / IXaMeasurementDataManagement <<Interface>>インターフェースの以下のプロパティは、ControlDesk 6.0 では削除されています。

- MDFFormat
- VariableNameStorage
- MDFXAxisDataType

MeasurementDataManagement / IXaMeasurementDataManagement <<Interface>>を参照してください。

IXaECUDiagnostics202ServerSettings インターフェースへの変更
ControlDesk 6.0 以降では、診断サーバは常に ControlDesk プロセス内から開始されます。

したがって、ECUDiagnostics202ServerSettings / IXaECUDiagnostics202ServerSettings <<Interface>>インターフェースの以下のプロパティは、ControlDesk 6.0 では削除されています。

- ExecuteDiagnosticsServerInProcessOfCDNGEnabled

  これは常に有効で、変更することはできません。

- UseServerJVMForDatabaseOptimizationEnabled

  これは常に有効で、変更することはできません。

「ECUDiagnostics202ServerSettings / IXaECUDiagnostics202ServerSettings <<Interface>>」(📖『ControlDesk Automation』)を参照してください。

Table Editor:Table Editor 配列作成の変更 ControlDesk 6.0 には、拡張された Table Editor が付属しています。ControlDesk 5.6 以前の Table Editor の自動化のために開発された自動化スクリプトは、ControlDesk 6.0 以降でも再利用することができます。

ただし、Table Editor Array 計器タイプは使用できなくなりました。代わりに、Table Editor 計器タイプを使用する必要があります。Table Editor Array 計器タイプとは異なり、Table Editor 計器タイプには既にデフォルトで 1 つのサブ計器が含まれています。

以下のリストでは、ControlDesk 5.6 以前で 2 つのサブ計器を使用して Table Editor を作成する方法を示します。

ControlDesk 5.6 以前で 2 つのサブ計器を使用して Table Editor を作成

```
TableEditor = ControlDeskApplication.LayoutManagement.ActiveLayout.Instruments.Add(
            "Table Editor Array",
            TableEditorInstrumentName,
            0,
            0,
            500,
            500)

        # Add sub instrument.
        TableEditor.SubInstruments.Add()
        (...)
        # Add sub instrument.
        TableEditor.SubInstruments.Add()
```

以下のリストでは、ControlDesk 6.0 以降で 2 つのサブ計器を使用して Table Editor を作成する方法を示します。

ControlDesk 6.0 以降で 2 つのサブ計器を使用して Table Editor を作成

```
TableEditor = ControlDeskApplication.LayoutManagement.ActiveLayout.Instruments.Add(
            "Table Editor",
            TableEditorInstrumentName,
            0,
            0,
            500,
            500)

        # Add sub instrument.
        TableEditor.SubInstruments.Add()
```

dSPACE Message Monitor の廃止　ControlDesk 6.0 では、dSPACE Message Service で記録されたログメッセージをモニタリングできる dSPACE Message Monitor が削除されました。

ILoLog インターフェースへの変更　ControlDesk 6.0 では、メッセージログファイルのフルパス名を取得する `Log / ILoLog <<Interface>>` の `LogFilePath` プロパティが削除されました。

Log / ILoLog <<Interface>>を参照してください。

Measurement Data API の変更

MDFFormatOption2 インターフェースの利用不可　ControlDesk 6.0 以降では、MDF 計測データファイル（MDF ファイル形式バージョン 2.0 および 3.0）のエクスポートがサポートされません。

その結果、`MDFFormatOption2` インターフェースは利用できなくなりました。

📖『ControlDesk Measurement Data API』を参照してください。

以前の ControlDesk バージョンからの移行

以前の ControlDesk バージョン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 from Prior Versions of ControlDesk」（📖『ControlDesk Introduction and Overview』）を参照してください。

関連トピック

基礎

- 「Basics on Migrating from Prior Versions of ControlDesk」（📖『ControlDesk Introduction and Overview』）

DCI Configuration Tool

DCI Configuration Tool 3.7 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。
ECU メモリアクセス	DCI Configuration Tool には、新しい[ECU Memory Access]ダイアログページが付属し、これを使用して特定の ECU メモリアドレスの読み書きを手作業でチェックすることができます。 「[ECU Memory Access]ページ」(🕮『DCI の設定』)を参照してください。
DCI-GSI1 および DCI-GSI2 インターフェースのファームウェアバージョン	DCI-GSI1 および DCI-GSI2 インターフェース用の以下のファームウェアバージョンが、DCI Configuration Tool 3.7 で提供されます。 ■ DCI-GSI1 ファームウェアバージョン 1.6.7 ■ DCI-GSI2 ファームウェアバージョン 1.4.4 **注記** DCI Configuration Tool で提供されるファームウェアバージョンは、必ずしも使用可能な最新のファームウェアバージョンではありません。問題が発生する場合は、新しいファームウェアバージョンが使用可能かどうかを dSPACE サポートにお問い合わせください。

dSPACE CAN API Package

dSPACE CAN API Package 3.0 の新機能

dSPACE CAN API 1.0 と dSPACE CAN API 2.0 の比較	dSPACE CAN API Package 3.0 では、以下の API バージョンを利用することができます。 ■ dSPACE CAN API 1.0 ■ dSPACE CAN API 2.0 dSPACE Release 2016-B では、dSPACE CAN API 2.0 が導入されています。これは dSPACE CAN API 1.0 の後継で、以前の機能をすべて含み、CAN FD のサポートが追加されています。dSPACE CAN API 1.0 とは異なり、dSPACE CAN API 2.0 は今後も開発が継続されます。
CAN FD のサポート	新しい dSPACE CAN API 2.0 は CAN FD をサポートします。 「Basics on CAN FD」(🕮『dSPACE CAN API 2.0 C Reference』)を参照してください。
DCI-CAN/LIN1 インターフェースのサポート	dSPACE CAN API 1.0 と新しい dSPACE CAN API 2.0 は、DCI-CAN/LIN1 インターフェースをサポートしています。 🕮『dSPACE CAN API 1.0 C Reference』および🕮『dSPACE CAN API 2.0 C Reference』を参照してください。

dSPACE ECU Flash Programming Tool

dSPACE ECU Flash Programming Tool 2.3.1 の新機能

DCI-CAN/LIN1 インターフェースのサポート	dSPACE ECU Flash Programming Tool は、CAN および CAN FD に対応した dSPACE の DCI-CAN/LIN1 インターフェースをサポートしています。 「サポートされる ECU インターフェースのタイプ」(🕮『ECU フラッシュプログラミング』)を参照してください。
バージョン 2.3 より前のツールからそのまま XCP on CAN フラッシュプロジェクトをロード可能	dSPACE ECU Flash Programming Tool 2.3 では、CAN および CAN FD が XCP トランスポートレイヤとしてサポートされます。これに関連して、Configure Flash Project ウィザードにコンフィギュレーション設定が追加されました。 dSPACE ECU Flash Programming Tool 2.3.1 では、XCP on CAN フラッシュプロジェクトを 2.3 より前のバージョンからロードプロセス時に自動的に移行します。

dSPACE FlexRay Configuration Package

dSPACE FlexRay Configuration Package 3.8 の新機能

今後のバージョンでの廃止予定

dSPACE Release 2017-A 以降の信号ベースのモデリングの廃止予定　RTI FlexRay Configuration Blockset を使用した信号ベースのモデリングのサポートは、dSPACE Release 2016-B で最後となります。dSPACE Release 2017-A 以降では、RTI FlexRay Configuration Blockset は、PDU ベースのモデリングのみをサポートするようになります。

そのため、信号ベースのモデリングコンセプトから、RTI FlexRay Configuration Blockset による PDU ベースのモデリングに切り替える必要があります。FlexRay Configuration Tool で作成された Simulink 設定データから、PDU ベースのモデリング用の設定済みの RTI FlexRay ブロックを生成する必要があります。自動的に生成された FlexRay モデルに含まれる RTI ブロックでは、設定済みの各 PDU は複数の信号で構成されています。つまり、PDU ベースのモデリングでは、複数の信号が 1 つの Simulink ブロックで処理されます。PDU ブロックを介して単一の信号にアクセスすることができます。

RTI FlexRay Configuration Blockset による PDU ベースのモデリングでは、信号ベースのモデリングよりも多くの機能を利用することができます。

dSPACE Python Extensions

項目の一覧

本章の内容

dSPACE Python Extensions 2.2 の新機能

64 ビットソフトウェアサポート

dSPACE Python Extensions は、64 ビットソフトウェアとして実装されています。

これにより、次のような影響があります。

- インストールパスは、`C:\Program Files` です。
  `C:\Program Files (x86)` ではありません。
- 64 ビットクライアントアプリケーションのみがサポートされます。
- `matlablib2` および `rs232lib2` Python モジュールは、64 ビット Python インタープリタでのみ使用可能です。

dSPACE Python Extensions 2.2 への移行

dSPACE Python Extensions に含まれるソフトウェアの廃止

dSPACE Release 2016-B では、dSPACE Python Extensions には以下が付属しません。

- dSPACE HIL API Python Implementation
- `rtplib2`

テストオートメーションプロジェクトは、HIL API の後継として ASAM XIL API に移行することができます。

HIL API Python または `rtplib2` からの XIL API .NET への移行については、テストオートメーションツールサポートセンター（http://www.dspace.jp/go/pscta）にお問い合わせください。

dSPACE XIL API .NET

項目の一覧

本章の内容

dSPACE XIL API .NET 2016-B の新機能

64 ビットソフトウェアサポート

dSPACE XIL API .NET は、64 ビットソフトウェアとして実装されています。

これにより、次のような影響があります。

- インストールパスは、`C:\Program Files` です。
  `C:\Program Files (x86)` ではありません。
- dSPACE XIL API .NET 2016-B の製品コンポーネントは、64 ビットバージョンとしてのみ使用することができます。

  その結果、次の点に注意する必要があります。

  - C#などの XIL API .NET MAPort または EESPort を使用する.NET プログラミング言語は、ターゲットアーキテクチャとして 64 ビットを使用してコンパイルする必要があります。
  - XIL API .NET MAPort または EESPort を使用する Python スクリプトは、64 ビット Python インタープリタでのみ使用でき、32 ビット Python インタープリタでは使用することができません。
  - XIL API .NET MAPort を使用する M で書かれたスクリプトは、64 ビット MATLAB でのみ実行することができます。

Platform Management API

Platform Management API は、Python Extensions セットアップから dSPACE XIL API セットアップに移動しました。以前と同様に、この API はライセンスなしでインストールすることができます。

dSPACE Release 2016-B 以降でのプラットフォーム管理自動化 API バージョン 1.0 の廃止 プラットフォーム管理自動化 API バージョン 1.0 のサポートは終了しました。API バージョン 2.0 を使用する必要があります。

詳細については、「Basics on the Platform Management API」(🕮『dSPACE Platform Management API Reference』)を参照してください。

XIL API Framework

ASAM XIL API Framework を使用して、異なるベンダーの複数の Testbench ポートを初期化し、変数オブジェクトを介して変数にアクセスすることができます。変数オブジェクトの使用は、リアルタイムアプリケーションでのモデル変数の具体的なモデルパスが、フレームワークの抽象的な識別子にマッピングされることを意味します。これらのマッピングは、固有の XML ベースのファイル形式で保存されます。このファイル形式は、ControlDesk や AutomationDesk でもサポートされています。

XIL API Framework は、単位変換もサポートします。テスト環境で使用される単位は、リアルタイムアプリケーションで使用される単位に変換することができます。たとえば、km/h から miles/h に変換することができます。

詳細については、「Basics on the Framework」(🕮『dSPACE XIL API Implementation Guide』)を参照してください。

MAPort 機能の拡張

MAPort インプリメンテーションには、以下の拡張が含まれます。

- MAPort スティミュレーションは、DataFileSegment および SignalValueSegment タイプのセグメントでの `eLINEAR` 補間方法をサポートします。この方法では、信号値はセグメントによって与えられるデータポイント間の線形補間によって計算されます。
- `eBACKWARD` 補間方法は、ASAM XIL API 規格で指定された動作を満たすように変更されています。詳細については、「dSPACE XIL API .NET 2016-B への移行」(159 ページ)を参照してください。
- dSPACE XIL API 2016-B では、`eFORWARD` 補間方法の使用は例外を発生します。dSPACE XIL API 2016-A 以前では、`eFORWARD` 補間方法は `eBACKWARD` 補間方法に暗黙的にマッピングされていました。
- ASAM MDF 4.1 (MF4) ファイルの Reduction Data 機能がサポートされました。

EESPort 機能の拡張

ソフトウェアトリガ　エラーセットがソフトウェアによってトリガできるようになりました。エラーセットを有効にする条件または持続時間のいずれかを指定することができます。

詳細については、「Basic Information on Configuring Errors」(🕮『dSPACE XIL API Implementation Guide』)を参照してください。

dSPACE XIL API .NET 2016-B への移行

dSPACE HIL API .NET から dSPACE XIL API .NET へのアプリケーションの移行

dSPACE HIL API .NET は、dSPACE Release 2016-B で廃止されます。必要な移行手順については、「Migrating HIL API Applications to XIL API Applications」(🕮『dSPACE XIL API Implementation Guide』)を参照してください。

eBACKWARD 補間の動作の変更

DataFileSegment および SignalValueSegment タイプのセグメントでの eBACKWARD 補間方法に使用されるインプリメンテーションが変更されています。

- XIL API 2016-A および Real-Time Testing 3.0 以前では、Real-Time Testing の RM_SAMPLED データストリーミングモードが使用されていました。
- XIL API 2016-B および Real-Time Testing 3.1 以降では、Real-Time Testing の RM_BACKWARD データストリーミングモードが使用されます。このモードは、ASAM AE XIL API 規格で定義された eBACKWARD 補間方法の仕様に準拠しています。

2 つの補間タイプの相違点については、「MatFile Class Description」(🕮『Real-Time Testing Library Reference』)を参照してください。

ホスト PC 上での信号評価の変更

XIL API 2016-B では、ホスト PC 上での信号の評価(SignalDescription の CreateSignalValue メソッドを実行する場合)が変更されました。

- XIL API 2016-A 以前では、セグメント境界を 2 つのサンプルステップ間に設定でき、連続するセグメントをサンプルステップ間で開始することが可能でした。
- XIL API 2016-B 移行では、セグメントのサンプリング値は常にサンプルステップのタイムスタンプから開始します。

EESPortConfiguration API の新バージョン

EESPortConfiguration API のバージョンは、2.2.0.0 から 3.0.0.0 に変更されます。

アプリケーションで関連するアセンブリを参照する場合は、以下のバージョンが必要です。

- dSPACE XIL API .NET 2016-A の使用:

  ```
  dSPACE.XIL.Testbench.EESPort.Interfaces.Extended.dll
  (Version 2.2.0.0, PublicKeyToken=f9604847d8afbfbb)
  ```

- dSPACE XIL API .NET 2016-B の使用:

  ```
  dSPACE.XIL.Testbench.EESPort.Interfaces.Extended.dll
  (Version 3.0.0.0, PublicKeyToken=f9604847d8afbfbb)
  ```

ECU Interface Manager

ECU Interface Manager 2.0 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 2.0 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 2.0 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。

新しいソフトウェ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ェアモジュールデスクリプションファイルを作成することができます。

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ェ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。

注記

- 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(「Using the Generic Schema to Create Software Module Description Files」(📖『ECU Interface Manager Reference』)を参照)*を使用してください。

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

Firmware Manager

Firmware Manager 2.2 の新機能

ユーザビリティの改善

ヘルプリボンの拡張　バックステージビューの[Help]リボンでは、『新機能と移行手順』ドキュメントや、dSPACE ソフトウェアとともにインストールされた PDF ファイルへの直接アクセスを利用することができます。

詳細については、「Basics on Ribbons」(🕮『Firmware Manager Document』)を参照してください。

新しい Message Viewer　Firmware Manager では、新しい Message Viewer をサポートします。Message Viewer を使用して、製品の使用中に発生した情報メッセージ、助言メッセージ、エラーメッセージ、警告メッセージ、質問のすべての履歴を表示することができます。システムの状態をチェックするのに役立ちます。

Model Compare

注記

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 2.7 の新機能

三元分析

Model Compare では、参照モデルと比較モデルを参照する第 3 のモデル（*共通祖先モデルなど*）を指定して、より精度の高い比較分析を行うことができます。この分析は、モデル部分とプロパティの値で構成されます。

関連ドキュメント

- 「How to Create Comparison Sessions」（📖『Model Compare Guide』）
- 「tlSimulinkBusObject」（📖『TargetLink API Reference』）

バージョン管理システムとの連携の改善

Model Compare は、外部のバージョン管理システムによって維持されるバージョン管理をサポートします。コマンドライン（API）から文字列または識別子を比較モデル、参照モデル、および共通祖先モデルにそれぞれ割り当てることができ、これによりバージョン管理システムが必要とするバージョンラベルの提供が可能となります。

関連ドキュメント

- 「Basics on the Interaction with Version Control Systems」（🕮『Model Compare Guide』）

新しい自動ハイライトモード

Model Compare の階層で選択されているブロックやサブシステムは、同時に MATLAB/Simulink でもハイライト表示されます。

関連ドキュメント

- 「How to Browse Differences」（🕮『Model Compare Guide』）

HTML/PDF レポートのレイアウトのカスタマイズ

レイアウトや言語をカスタマイズするため、Model Compare では、PDF および HTML レポート生成のためのユーザ固有のスタイルシートを指定することができます。

関連ドキュメント

- 「How to Generate Difference Reports」（🕮『Model Compare Guide』）

Model Compare 2.7 への移行

調整不要

Model Compare 2.6 から Model Compare 2.7 へは、何も調整せずに移行することができます。

ModelDesk

項目の一覧

本章の内容

ModelDesk 4.4 の新機能

運転操作

運転操作やドライビングサイクルを ModelDesk から直接、開始、停止、およびリセットすることができます。

Road Generator

OpenDRIVE のエクスポート Road Generator は、OpenDRIVE フォーマット 1.4H で道路をエクスポート／インポートすることができます。

Table Editor Road Generator には、高さポイントや横方向の傾斜ポイント、カスタム高さおよび摩擦マップを指定するための新しい Table Editor が付属しています。

プロット

新しいプロッタ計器 ModelDesk には次の新しいプロッタが付属しています。時間プロッタと XY プロッタ

Signal Selector アップデートされた Signal Selector では、ModelDesk の他のツリーリストと同様、信号検索や複数選択などの機能を利用することができます。

プロットとダウンロード プロットとダウンロードを同時に実行できるようになりました。

処理	**生データファイル**　処理実行時に MATLAB で生データファイルに関する情報を利用することができます。 **処理実行時のメッセージ**　処理実行時に、情報、エラー、警告などの各種メッセージを表示することができます。メッセージはそれぞれのタイプに応じて構造化されています。 **パラメータページの検索**　[Parameter Configuration]ペインでパラメータページの名前を検索することができます。
パラメータ管理	**Table Editor**　ModelDesk には、多次元パラメータを指定できる新しい Table Editor が付属しています。

ModelDesk 4.4 への移行

プロットでのツールオートメーション	ModelDesk のプロッタが新しくなったことにより、プロットでのツールオートメーションが変更されています。プロット用のスクリプトを再利用するには、これらのスクリプトを調整する必要があります。

Model Interface Package for Simulink

項目の一覧

本章の内容

Model Interface Package for Simulink 3.3 の新機能

ビヘイビアモデルの通信インターフェース生成の簡略化

dsmpb_createreversedmodelportblock() API コマンドが拡張されています。作成された逆方向のモデルポートブロックの保存先パスを指定することができます。「dsmpb_createreversedmodelportblock」(🕮『Model Interface Package for Simulink Reference』)を参照してください。

SIC ファイルでの生成された S-function のサポートの改善

以前は、dsrt_addnonbuildfiles() API コマンドを使用して、生成された S-function のソースコードディレクトリを SIC ファイルに追加する必要がありました。dSPACE Release 2016-B 以降では、Model Interface Package for Simulink が、生成された S-function をモデルが含んでいるかどうかを検出します。S-function が使用され、その名前が"_sf"(S-functions 生成時のデフォルト名の接尾辞)で終わる場合、Model Interface Package for Simulink は対応するソースコードディレクトリを SIC ファイルに自動的に追加します。

MATLAB R2016b を使用する場合の制限事項

MATLAB R2016b で Model Interface Package for Simulink を使用する場合は、下記の制限事項に注意してください。

- 再帰的関数のコード生成

  再帰的関数のコード生成は Simulink Coder でサポートされますが、リアルタイムハードウェア上で実行されるリアルタイムアプリケーションの生成には使用しないでください。

- Upgrade Advisor API の使用

  新しい MATLAB バージョンへのアップグレードを行う場合は、`upgradeadvisor` 関数を使用して、モデルに必要な移行手順を分析し実行することができます。RTI ブロックセットなど、使用するサードパーティ製ブロックセットは、この関数を使用してアップグレードしないでください。`SkipBlocksets` 引数のデフォルト値を `true` から `false` に変更しないでください。

- 標準 Math ライブラリの変更

  標準 Math ライブラリ設定および `TargetLangStandard` パラメータのデフォルト値は `C99 (ISO)` です。C99 規格は、すべてのコンパイラではサポートされていません。したがって、dSPACE プラットフォームが選択されている場合、この設定は `C89/90 (ANSI)` に自動的に切り替えられます。

- 新しい Simulink ブロック

  以下の新しい Simulink ブロックはサポートされていません。

  - Initialize Function ブロック
  - Terminate Function ブロック

Model Interface Package for Simulink の移行上の注意点

MATLAB R2016b で修正された機能

起動動作の変更　MATLAB では起動スクリプトの実行順序が変更されています。

- MATLAB R2016a 以前では、`startup.m` スクリプトが最初に実行され、dSPACE 固有の初期化スクリプト `dsstartup.m` および `dspoststartup.m` が後に続きます。
- MATLAB R2016a 以降では、`startup.m` スクリプトは、dSPACE 固有の初期化スクリプトの後に最後に実行されます。

この変更によって起動動作が変化する場合があります。

新しい MATLAB バージョンへの切り替え　新しい MATLAB バージョンをインストールする場合、既にインストールされている MATLAB バージョンから設定の一部が引き継がれます。新しい MATLAB バージョンまたは dSPACE Release に切り替える場合、Simulink モデルの予期しない動作を防ぐために、作業を開始する前に MATLAB および Simulink のプリファレンスを必ずデフォルト状態にリセットしてください。

MotionDesk

項目の一覧

本章の内容

MotionDesk 3.9 の新機能

リアルタイムのレンズ歪み

MotionDesk は、実際のレンズの歪みをシミュレートすることができます。カメラベースの認識アルゴリズムのテストに MotionDesk を使用する場合は、通常、MotionDesk ディスプレイの正面にレンズを含むカメラが配置されます。MotionDesk はレンズの歪みをシミュレートできるため、ECU テストで実際のレンズを使用する必要はありません。
MotionDesk は、ビネット効果、たる形歪曲、ピンクッション歪み、色収差などの複数の歪み効果をサポートします。

道路の生成

生成される道路形状の精度を指定することができます。道路の曲率に基づく精度、または道路に使用されるセグメントタイプに基づく精度を指定することができます。

選択可能な鮮明なテクスチャ

鮮明なテクスチャを使用して風景のディテールを表現し、シーンをリアルに見せることができます。シーンを引き立たせるエレメントを選択することができます。地面の花や、路面の擦り切れた場所、ランダムな亀裂やパターンをビジュアル表示することができます。

Scene Editor	Library Browser から 3D オブジェクトをシーンにドラッグすると、シーン編集モードが自動的に有効になります。この機能は、[MotionDesk Properties]ダイアログの[Visualization]ページで無効にすることができます。 dSPACE オブジェクトライブラリには、テクスチャが圧縮された 3D オブジェクトが含まれます。そのため、実行時に 3D オブジェクトを圧縮する必要はありません。
MotionDesk Blockset	MD_Object ブロックは、カルダン角の解釈を使用する場合、以下の 2 つの異なる回転順序(規則)を使用して変換マトリクスを計算します。 ■ Z-Y'-X''規則:オブジェクトは、z 軸、y 軸周りを回転してから実際の x 軸周りを回転します。 ■ Z-X'-Y''規則:オブジェクトは、z 軸、x 軸周りを回転してから実際の y 軸周りを回転します。

MotionDesk 3.9 への移行

MicroAutoBox のサポートの廃止	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 ソフトウェアは、MicroAutoBox II バージョン 1401/1501、1401/1504、1401/1505/1507、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。
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 3.9 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 3.9 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

項目の一覧

本章の内容

Real-Time Testing 3.1 の新機能

新しい Python バージョン

シミュレーションプラットフォームで実行される Python インタープリタは、Python 2.7.11 に基づいています。

ホスト呼び出し

`rttlib.hostcall` モジュールが、Real-Time Testing でサポートされているすべてのプラットフォームをサポートします。これには特に、SCALEXIO システムや DS1007、MicroLabBox、VEOS が含まれます。

データストリーミング

`rttlib.datastreaming` モジュールには、再生用の新しいモードがあります。MAT ファイルタイムベクトルのタイムスタンプがモデルステップに適合しない場合は、それぞれのモードに応じて異なる値が再生されます。

- `datastream.RM_LINEAR`: 再生に使用される値は、モデルステップの前と後の MAT ファイルタイムベクトルの値の間の線形補間です。
- `datastream.RM_BACKWARD`: 再生に使用される値は、モデルステップの直前のタイムスタンプをもつ MAT ファイルタイムベクトルの値です。

Real-Time Testing 3.1 への移行

互換性のない BCG ファイル

Real-Time Testing 2.6 以前で生成された BCG ファイルは、Real-Time Testing 3.1 では使用することができません。RTT シーケンスの BCG ファイルをもう一度作成する必要があります。「Creating and Starting RTT Sequences in Python Scripts」(🕮『Real-Time Testing Guide』)を参照してください。

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

MicroAutoBox

安全機能の改善　MicroAutoBox でサポートされる RTI Watchdog Blockset は、Challenge-Response Monitoring サブライブラリを提供します。

詳細については、「RTI Watchdog Blockset 2.0 の新機能」(191 ページ) を参照してください。

RTI ライセンスパッケージの拡張

RTI 基本製品のライセンスには、以下のライセンスが含まれます。

- RTI USB Flight Recorder Blockset
- RTI DS1552 I/O Extension Blockset

MATLAB R2016b の新機能

MATLAB R2016b で導入された以下の新機能は、RTI/RTI-MP でサポートされます。

- [Hardware Implementation]ダイアログでの新しい設定

  このダイアログでは、size_t および ptrdiff_t 設定を利用することができます。どちらの設定も、すべての RTI プラットフォームで 32 ビットに設定する必要があります。

MATLAB R2016b を使用する場合の制限事項

MATLAB R2016b 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。

- Upgrade Advisor API の使用

  新しい MATLAB バージョンへのアップグレードを行う場合は、`upgradeadvisor` 関数を使用して、モデルに必要な移行手順を分析し実行することができます。RTI ブロックセットなど、使用するサードパーティ製ブロックセットは、この関数を使用してアップグレードしないでください。`SkipBlocksets` 引数のデフォルト値を `true` から `false` に変更しないでください。

- Property Inspector の使用

  Property Inspector を使用して、モデルエレメントのパラメータやプロパティを編集することができます。これは、標準的な Simulink マスクインターフェースを使用してパラメータが設定されたブロックのみをサポートします。独自のダイアログを提供するほとんどの RTI ブロックはサポートされません。

- 再帰的関数のコード生成

  再帰的関数のコード生成は Simulink Coder でサポートされますが、リアルタイムハードウェア上で実行されるアプリケーションの生成には使用しないでください。

- 標準 Math ライブラリの変更

  標準 Math ライブラリ設定および `TargetLangStandard` パラメータのデフォルト値は `C99 (ISO)` です。C99 規格は、すべてのコンパイラではサポートされていません。したがって、dSPACE プラットフォームが選択されている場合、この設定は `C89/90 (ANSI)` に自動的に切り替えられます。

- 新しい Simulink ブロック

  以下の新しい Simulink ブロックはサポートされていません。

  - Initialize Function ブロック
  - Terminate Function ブロック

RTI/RTI-MP の新機能

RTI/RTI-MP の以下の機能が、dSPACE Release 2016-B で導入されています。

- プログラムタスク設定

  RTI および RTI-MP のタスク設定には、プログラム API を介してアクセスすることができます。詳細については、「RTI and RTI-MP Task Configuration API Reference」(🕮『RTI and RTI-MP Implementation Reference』)を参照してください。

- MATLAB R2016a 以降では、[Simulink Start]ページの Simulink テンプレートを選択することによりモデルを新規作成できます。dSPACE Release 2016-B 以降では、RTI テンプレートファイルは、[Simulink Start]ページの[My Templates]カテゴリにあります。

  MATLAB R2016b を使用する場合、テンプレートは、`new_system` によって作成されたモデルにも使用することができます。RTI プラットフォームサポートを切り替えることにより、あらかじめ設定されたプラットフォーム関連のテンプレートがデフォルトとして設定されます。

  以下の制限事項に注意する必要があります。

  - カスタムモデルテンプレートでの RTI ブロックの使用はサポートされません。
  - 以前の dSPACE Release のモデルテンプレートの使用はサポートされません。
- `rti_build` コマンドに代わる新しい `rti_build2` コマンドは、ビルドプロセスでの不整合のリスクを減少させます。

  詳細については、「rti_build2」(📖『RTI and RTI-MP Implementation Reference』)および「rtimp_build2」(📖『RTI and RTI-MP Implementation Reference』)を参照してください。

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 ファイル生成には、いくつかの変更があります。「TRC ファイル生成の変更」(37 ページ)を参照してください。

MATLAB R2016b で修正された機能

起動動作の変更 MATLAB では起動スクリプトの実行順序が変更されています。

- MATLAB R2016a 以前では、`startup.m` スクリプトが最初に実行され、dSPACE 固有の初期化スクリプト `dsstartup.m` および `dspoststartup.m` が後に続きます。
- MATLAB R2016a 以降では、`startup.m` スクリプトは、dSPACE 固有の初期化スクリプトの後に最後に実行されます。

この変更によって起動動作が変化する場合があります。

ビルド時の入力パラメータの変更 MATLAB R2016b では、RTI コマンド `rti_build` および `rtimp_build` は、`MakeOptions` 入力パラメータをサポートしません。`MakeOptions` パラメータによって開始されるビルドプロセスは、エラーメッセージを表示して中断されます。

新しい MATLAB バージョンへの切り替え　新しい MATLAB バージョンをインストールする場合、既にインストールされている MATLAB バージョンから設定の一部が引き継がれます。新しい MATLAB バージョンまたは dSPACE Release に切り替える場合、Simulink モデルの予期しない動作を防ぐために、作業を開始する前に MATLAB および Simulink のプリファレンスを必ずデフォルト状態にリセットしてください。

RTI Bypass Blockset

項目の一覧

本章の内容

RTI Bypass Blockset 3.7 の新機能

RTI Bypass Blockset

オンターゲットバイパス処理:TargetLink のサポート RTI Bypass Blockset では、オンターゲット(内部)バイパス処理の実行時のモデルのコード生成用に、Simulink® Coder™ のみでなく TargetLink Code Generator もサポートされます。オンターゲットバイパス部分を含む Simulink モデルを使用する場合、使用するコードジェネレータを選択することができます。

さらに、RTI Bypass Blockset は、モデリングツールとして TargetLink をサポートしています。これにより、RTI Bypass Blockset のブロックを TargetLink モデルで使用し、TargetLink モデルを既存の ECU アプリケーションに容易に統合することができます。

🕮『RTI Bypass Blockset リファレンス』を参照してください。

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 3.7 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 3.7 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 3.7 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]をクリックするとすぐに行われます。

RTI Bypass Blockset 3.7 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]をクリックするとすぐに行われます。RTI Bypass Blockset 3.7 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 3.6 までのモデルを使用する場合*

RTI Bypass Blockset 2.6 から RTI Bypass Blockset 3.6 までででビルドされた Simulink モデルを RTI Bypass Blockset 3.7 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 3.6 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 4.4 の新機能

FIBEX 4.1.1 のサポート

RTI CAN MultiMessage Blockset では、データベースファイルとして FIBEX 4.1.1 ファイルをサポートしています。

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

RTI CAN MultiMessage Blockset 4.4 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、「Limitations with RTICANMM」(📖『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 FPGA Programming Blockset

項目の一覧

本章の内容

RTI FPGA Programming Blockset 3.2 の新機能

Xilinx®のサポートの拡張

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ました。

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 [1]
Vivado 2016.2 （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 R2015a SP1 ■ MATLAB R2015b

[1] RTI FPGA Programming Blockset の Processor Interface サブライブラリは、MATLAB R2016a および MATLAB R2016b もサポートしています。

タイミング分析の拡張

FPGA モデルのタイミング分析に加え、FPGA アプリケーション全体のタイミング分析を使用することができます。FPGA アプリケーション全体には、FPGA モデルと使用されるプラットフォーム用のフレームワークが含まれます。ビルドプロセスは、FPGA アプリケーション全体のタイミング分析を自動的に開始し、MATLAB コマンドウインドウで結果へのリンクを作成します。

DS2655 FPGA Base Board のフレームワークの機能拡張	DS2655 FPGA Base Board と I/O モジュールのフレームワークは、次のように拡張されています。 **ConfigurationDesk での電気的インターフェースの設定** ConfigurationDesk で FPGA I/O インターフェースの電気的特性を変更することができます。これにより、同じ FPGA カスタムファンクションブロックを複数の ConfigurationDesk プロジェクトで再利用でき、バリアントの数を少なくすることができます。 **FPGA 信号のトレース**　Simulink の Bus Creator および Bus Selector ブロックのすべての出力信号を試験ソフトウェアを使用して直接トレースできるように指定することができます。FPGA_SETUP_BL ブロックを使用して、FPGA 信号をトレース可能にすることができます。 トレース可能な信号の数を少なくするため、サブシステムオミッションタグ（DsVdOmit）を使用してサブシステムの信号をトレースから除外することができます。サブシステムの[Block Properties]ダイアログで、DsVdOmit タグを入力します。
MicroLabBox のフレームワークの拡張	**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 ... 1.25e-9 s（6.25 MHz ... 800 MHz）です。 The FPGA_SETUP_BL block provides the Subsystems Clocks page to configure the clock period for subsystems with an individual clock period. **ブザー**　MicroLabBox のブザーを使用して、音声信号を生成することができます。
関連トピック	基礎 • 「RTI FPGA Programming Blockset 3.2 への移行」(186 ページ)

RTI FPGA Programming Blockset 3.2 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 1.1 以降から 3.2 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、これを RTI FPGA Programming Blockset 3.2 で使用する場合、フレームワークは自動的に最新のフレームワークバージョンに更新されます。

この更新では、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。

dSPACE Release 2016-B と互換性のない ConfigurationDesk カスタムファンクション

注記

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 FPGA カスタムファンクションブロックと、FPGA カスタムファンクションブロックを含むリアルタイムアプリケーション（*.rta）は、dSPACE Release 2016-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 2016-B の RTI FPGA Blockset 3.2 を使用して FPGA モデルを再ビルドする必要があります。

異なる dSPACE ハードウェアの使用

FPGA モデルを異なる dSPACE ハードウェアで使用するには、モデルの一部修正が必要です。「Migrating to Another dSPACE Hardware」(🕮『RTI FPGA Programming Blockset Guide』)を参照してください。

RTI LIN MultiMessage Blockset

項目の一覧

本章の内容

RTI LIN MultiMessage Blockset 2.7 の新機能

FIBEX 4.1.1 のサポート

RTI LIN MultiMessage Blockset では、データベースファイルとして FIBEX 4.1.1 ファイルをサポートしています。

「General Settings Page (RTILINMM MainSetup)」(🕮『RTI LIN MultiMessage Blockset Reference』)を参照してください。

RTI LIN MultiMessage Blockset 2.7 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 CAN Blocks]コマンドを選択します。

詳細については、「Limitations of RTI LIN MultiMessage Blockset」(📖『RTI LIN MultiMessage Blockset Reference』)を参照してください。

RTI Watchdog Blockset

RTI Watchdog Blockset 2.0 の新機能

Challenge-response monitoring

RTI Watchdog Blockset は、周期的タスクまたはサブシステムなどの任意のソフトウェアエンティティの監視に使用されるチャレンジレスポンス方法をサポートする Challenge-Response Monitoring サブライブラリによって拡張されています。

新しい Challenge-Response Monitoring ブロックでは、以下の 2 つの機能のため、現在の Watchdog Handling ブロックよりも上位レベルの安全機能を実装することができます。

- 欠陥は、指定されたタイムアウト期間内に応答が得られない場合に検出されるだけでなく、返された応答が無効な値をもつ場合にも検出されます。
- Challenge-response monitoring ユニットは、リアルタイムアプリケーションで使用されないハードウェアコンポーネント上で実行されます。そのため、監視されるエンティティの実行は、監視自体から分離されています。

Watchdog ブロックは、Watchdog Handling サブライブラリに移動しました。

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

SCALEXIO Firmware

SCALEXIO Firmware 3.5 の新機能

新しいライセンス	SCALEXIO ファームウェアは、新しいライセンスをサポートします。 ■ 新しい SCALEXIO_RTLIB_MC8 ライセンスでは、プロセッサユニットに対して 4 より多いアプリケーションのダウンロードが可能になります。 ■ 新しい SCALEXIO_FIU_500 ライセンスでは、ConfigurationDesk でリアルタイムアプリケーションに最大 500 までのファンクションブロックを実装した欠陥シミュレーションが可能になります。
アップデートされた QNX コンパイラのサポート	SCALEXIO ファームウェアは、64 ビット Windows に対応した QNX コンパイラバージョン 5.2.0 をサポートします。これにより、非常に大きな Simulink モデルでも Out-of-Memory エラーを発生せずにコンパイルすることができます。

SystemDesk

項目の一覧

本章の内容

SystemDesk 4.7 の新機能

項目の一覧

本章の内容

新しい一般機能

目的

SystemDesk 4.7 には、次の一般機能が新たに追加されています。

コンピュメソッド'BITFIELD TEXTTABLE'に対応したソフトウェアアーキテクチャモデリングと RTE コード生成

SystemDesk では、カテゴリ BITFIELD TEXTTABLE のコンピュメソッドのサポートが改善され、これにより内部値の範囲のテクスチャ表現が可能になりました。

プロパティダイアログを介してこのカテゴリのコンピュメソッドを指定し、これらのコンピュメソッドを含むマッピングされたソフトウェアコンポーネントを使用する ECU コンフィギュレーションの RTE コードを生成することができます。

妥当性確認の改善

このバージョンの SystemDesk には、以下の妥当性確認規則が追加されています。

- AUTOSAR の制約の妥当性確認の改善は、ソフトウェアアーキテクチャのモデリングや ECU 設定に役立ちます。
- TargetLink 互換性の妥当性確認の改善は、TargetLink と SystemDesk 間のソフトウェアコンポーネントのやり取りや実装に役立ちます。

ECU コンフィギュレーション

目的

SystemDesk では、すべてのベンダーの RTE および基本ソフトウェア（BSW）モジュールでの ECU 設定に使用できる *ECU コンフィギュレーションフレームワーク*を提供します。

ECU コンフィギュレーションフレームワーク

ECU コンフィギュレーションフレームワークには、以下のコンポーネントが含まれています。

- システムデスクリプションからの BSW モジュールコンフィギュレーションのパラメータの導出や、BSW コード生成などのタスクを実行する、BSW モジュールコンフィギュレーション向けのプラグインコマンドインターフェース。

  SystemDesk は、AUTOSAR ビルドアクションマニフェストをサポートし、プラグインコマンドの実行を定義します。各モジュールコンフィギュレーションは、関連するビルドアクションマニフェストを参照し、コンテキストメニューで定義されたプラグインコマンドを提供します。「プラグインコマンド」(198 ページ)を参照してください。

- ECU コンフィギュレーションのモジュールを自動設定しそのコードを生成する、プラグインコマンドのグループの実行を可能にするビルドツール。

  SystemDesk は、関連情報を提供するビルドアクションマニフェストを介して、プラグインコマンド間の相互依存性を分析します。

- IronPython スクリプトを使用して BSW モジュールコンフィギュレーションのプラグインコマンドを開発するためのサポート。

  関連するビルドアクションマニフェストを使用して、IronPython スクリプトを生成することができます。「BSW スクリプトの開発」(198 ページ)を参照してください。

- データを損失することなく ECU コンフィギュレーションパラメータ定義の間でモジュールコンフィギュレーションを変換するスクリプトベースのメカニズム。

  これに使用する IronPython スクリプトを生成することができます。「BSW モジュールコンフィギュレーションの変換」(199 ページ)を参照してください。

- ランナブルエンティティの実行をスケジュールする Runnable Mapping エディタおよび Generate Mappings コマンド。アブストラクションを介してすべての OS および RTE モジュールでこの機能を使用することができます。「ランナブルのマッピングアブストラクション」(199 ページ)を参照してください。

- BSW モジュールコンフィギュレーションのパラメータを設定する BSW Module エディタ。

- すべてのベンダーの ECU の AUTOSAR モデルおよび BSW モジュールコンフィギュレーションパラメータのすべてのエレメントにアクセス可能な SystemDesk の自動化機能。

IronPython の使用　ECU コンフィギュレーションフレームワークは、IronPython インタープリタでプラグインコマンドや BSW モジュール変換用の BSW スクリプトを実行します。

Visual Studio 用 Python ツールや IronPython 開発パッケージを使用する場合は、BSW スクリプトを開発し、Visual Studio のデバッグ機能を使用することができます。

プラグインコマンド

BSW モジュールコンフィギュレーションのためのプラグインコマンドを実行し、システムデスクリプションからの BSW モジュールコンフィギュレーションのパラメータの導出や、BSW コード生成などのタスクを実行します。

SystemDesk は、AUTOSAR ビルドアクションマニフェストをサポートし、プラグインコマンドの実行を定義します。各モジュールコンフィギュレーションは、関連するビルドアクションマニフェストを参照し、コンテキストメニューで定義されたプラグインコマンドを提供します。

プラグインコマンドを Microsoft .NET ライブラリや IronPython スクリプト、または実行形式などのさまざまな環境に実装することができます。SystemDesk は、IronPython を使用してプラグインコマンドを開発するためのサポートを提供します。

プラグインコマンドの実行は、次のタスクで非常に役に立ちます。

- システムデスクリプションからの BSW モジュールコンフィギュレーションパラメータの導出。AUTOSAR では、このための上流のマッピングルールが記述されています。
- BSW モジュールのためのソフトウェアコンポーネントデスクリプションの生成。このファイルは、アプリケーションソフトウェアをベーシックソフトウェアに接続し、RTE コードを生成するために必要となります。
- BSW モジュールコンフィギュレーションパラメータに従った BSW モジュールのコード生成。

「Basics on Build Action Manifests」(🕮『SystemDesk Guide』)を参照してください。

BSW スクリプトの開発

ECU を設定するには、基礎となる ECU エクストラクトを分析し、それに従って BSW パラメータを設定する必要があります。BSW パラメータに基づいて BSW コードと BSW コンポーネントを生成することができます。

SystemDesk では、これらのタスクに使用する IronPython スクリプト開発のための次のサポートを提供します。

- ECU 設定のためのテンプレート BSW モジュールスクリプトを生成します。

  BSW モジュールコンフィギュレーションのすべてのコンテナおよびパラメータを設定するためのスタブ関数を使用して、テンプレートスクリプトを生成することができます。

- BSW スクリプト用のビルドアクションマニフェストを関連する BSW モジュールに追加します。

  関連する BSW モジュールのコンテキストメニューから、BSW スクリプトを実行することができます。

- BSW スクリプトで ECU ソースシステムおよび BSW モジュールコンフィギュレーションの AUTOSAR エレメントにアクセスします。

  SystemDesk からパラメータとして BSW スクリプトに渡される自動化インターフェースを介して、関連情報を取得することができます。

「Basics on Developing BSW Module Scripts」(🕮『SystemDesk Guide』)を参照してください。

BSW モジュールコンフィギュレーションの変換

SystemDesk では、Python スクリプトに基づいて BSW モジュールコンフィギュレーションを変換することができます。任意の RTE または BSW モジュール向けの変換スクリプトを開発することができます。そのため、データを損失することなく BSW モジュールコンフィギュレーションをインポート／エクスポートすることができます。

SystemDesk では、BSW モジュール変換スクリプトを開発するための次のサポートを提供します。

- BSW モジュールコンフィギュレーションパラメータ用のスタブ関数を使用して、変換スクリプト(両方向)を生成します。
- 変換スクリプトで変換できないデータは SystemDesk で維持されます。データは逆方向の変換で復元することができます。

また、SystemDesk では、BSW モジュールコンフィギュレーションのコンテナとパラメータの名前照合を介した単純な変換も利用することができます。

「Basics on Converting Basic Software Module Configurations」(🕮『SystemDesk Guide』)を参照してください。

ランナブルのマッピングアブストラクション

Runnable Mapping エディタや Generate Mappings コマンドをすべてのベンダーの OS および RTE モジュールとともに使用することができます。

ECU コンフィギュレーションフレームワークは、ランナブルマッピングアブストラクションとして IronPython スクリプトを提供し、これにより AUTOSAR 4 準拠の大多数の RTE および OS モジュールを使用することができます。ただし、Runnable Mapping エディタを使用してエラーが発生する場合は、ランナブルマッピングアブストラクションを調整する必要があります。このような場合は、ベンダー固有の RTE および OS モジュールでの使用にスクリプトを調整するため、dSPACE サポートにお問い合わせください。

ランナブルマッピングアブストラクションの使用事例　ランナブルマッピングアブストラクションを使用すると、システムコンフィギュレーション段階でアプリケーションソフトウェアコンポーネントのランナブルマッピングを実行し、ECU コンフィギュレーション段階の開発を担当するパートナーとのマッピングのやり取りが可能になります。

システムコンフィギュレーション段階でのランナブルの OS タスクへのマッピング　一般的に、ランナブルの OS タスクへのマッピングは、ECU コンフィギュレーション段階で行われます。ただし、アプリケーションソフトウェアコンポーネントを ECU にマッピングする場合は、システムコンフィギュレーション段階でアプリケーションソフトウェアコンポーネントのランナブルマッピングが必要となる場合があります。これは、サプライヤと OEM メーカーとの協業などで、ECU コンフィギュレーションとシステムコンフィギュレーションが別のパートナーによって実施される場合に役に立ちます。

ランナブルマッピングのやり取り　ECU エクストラクトを介して、システムコンフィギュレーションと ECU コンフィギュレーションの間で必要な情報をやり取りすることができます。ただし、この目的のランナブルマッピングは、RTE および OS コンフィギュレーションの一部であり、ここには含まれていません。BSW コンフィギュレーションおよびソフトウェアアーキテクチャツールを使用するサプライヤと OEM メーカーとの協業でこの情報をやり取りするには、RTE および OS コンフィギュレーションもやり取りする必要があります。

システムコンフィギュレーション段階で SystemDesk を使用する場合は、ランナブルマッピングアブストラクションを使用して、ベンダー固有の RTE および OS モジュールでの OS タスクのマッピングにランナブルを実行することができます。サプライヤと OEM メーカーとの協業でのその他のパートナーは、初期設定として RTE および OS コンフィギュレーションをランナブルマッピングとともに使用するか、または既存のコンフィギュレーションとマージすることができます。そのために追加の変換手順を実行する必要はありません。

「Basics on the Runnable Mapping Abstraction」(📖『SystemDesk Guide』)を参照してください。

仮想検証で使用するシミュレーションシステムの作成

SWC テンプレートファイルの生成

RteGeneration/RteGenerateSwcTemplateFiles モジュールのコンフィギュレーションパラメータを使用して、RTE 生成を設定することができます。このオプションを使用して、ソフトウェアコンポーネントのテンプレートファイルを生成することができます。

生成されたテンプレートファイルは、以下の目的に使用することができます。

- すべてのソフトウェアコンポーネントを実装する前に、シミュレーションシステムのビルドをテストします。これにより仮想検証を準備し、妥当性確認プロセスをセットアップするのに役立ちます。

  このためには、テンプレートファイルを作業フォルダにコピーし、このファイルを個々の SWC インプリメンテーションから参照します。

- 手書きのソフトウェアコンポーネントの実装を開始します。テンプレートは、データアクセスのためにランナブルシグネチャと関連する RTE API 関数を提供します。

  このためには、テンプレートファイルを作業フォルダにコピーし、ソフトウェアコンポーネントの動作に関するコードを追加します。

下の図にテンプレートファイルの例の一部を示します。

```
/**********************************************************************************************\
 ***  FUNCTION:
 ***      Linearization
 ***
 ***  DESCRIPTION:
 ***
 ***  PARAMETERS:
 ***      Type               Name               Description
 ***      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 ***
 ***  RETURNS:
 ***      void
 ***
 ***  SETTINGS:
 ***
\**********************************************************************************************/
FUNC(void, RTE_CODE) Linearization(void)
{
    {
      VAR(sint16, AUTOMATIC) PosSensor = Rte_InitValue_PosSensor_PosSensor;

      /* Performs an implicit read on a sender-receiver communication data item with data semantics.
       */
      PosSensor = Rte_IRead_Linearization_PosSensor_PosSensor();
    }
    {
      VAR(sint16, AUTOMATIC) LinPos = 0;

      /* Performs an implicit write on an interrunnable variable. */
      Rte_IrvIWrite_Linearization_LinPos(LinPos);
    }
```

V-ECU インプリメンテーションのインポート

V-ECU インプリメンテーションをシミュレーションシステムにインポートすることができます。インポート時に、SystemDesk がコードベースの V-ECU を作成し、V-ECU インプリメンテーションファイル（C/H、ARXML、A2L）をインポート／参照します。

これにより、次の作業を実行することができます。

- 同じ SystemDesk プロジェクトに関連する AUTOSAR モデルがなくても、V-ECU をシミュレーションシステムに追加することができます。
- V-ECU インプリメンテーションを簡単に交換して、SystemDesk でビルドやデバッグを行うことができます。

SystemDesk 4.7 への移行

SystemDesk 4.7 への移行

SystemDesk 4.7 では、SystemDesk 4.5 および 4.6 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。

注記

SystemDesk 4.5 または 4.6 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 4.7 で開きます。

TargetLink

項目の一覧

本章の内容

TargetLink 4.2 および TargetLink Data Dictionary 4.2 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(コード関連)

TargetLink では、TargetLink のコード生成に影響を及ぼす次の Simulink ブロックのサポートが追加されています。

- Vector Concatenate

  このブロックは、Matrix Concatenate ブロックに類似したブロックです。これらの 2 つのブロックでは、Mode 設定(Vector と Multidimensional array)のみが異なります。

関連ドキュメント

- 「Code-Relevant Simulink Blocks」(🕮『TargetLink Block and Object Reference』)
- 「Basics on Reusing Variables of Preceding and Subsequent Blocks」(🕮『TargetLink Customization and Optimization Guide』)

サポートされる Simulink ブロック(非コード関連)

TargetLink では、TargetLink のコード生成に影響を及ぼさない次の Simulink ブロックのサポートが追加されています。

- Data Type Duplicate
- Data Type Propagation
- Floating Scope
- XY Graph
- Stop Simulation
- Timed-Based Linearization
- Trigger-Based Linearization

関連ドキュメント

- 「Code-Irrelevant Simulink Blocks」(🕮『TargetLink Block and Object Reference』)

Data Store ブロックおよびバスサポートの改善

要素の選択およびバス対応のデータストア

この TargetLink バージョンでは、Data Store Memory、Data Store Read、および Data Store Write ブロックでバス信号がサポートされています。バス全体を、あらかじめ設定された構造タイプ定義やあらかじめ設定された構造変数にマッピングすることができます。これは、バスが多数のバスエレメントで構成されている場合や、1 つまたは複数のモデル内でタイプや変数が複数回使用される場合に特に役立ちます。

廃止された制限事項 Data Store ブロックに関するいくつかの制限事項がなくなりました。詳細については、「廃止された制限事項」(259 ページ)を参照してください。

- Data Store Read および Data Store Write ブロックでは、複数の入力および出力ポートを使用することができます。
- データストアに対する部分的な読み取り／書き込みも、TargetLink によってサポートされます。

関連ドキュメント

- 「Basics on Modeling Buses via Data Store Blocks」(📖『TargetLink Customization and Optimization Guide』)
- 「Basics on the Representation of Buses in the Production Code」(📖『TargetLink Customization and Optimization Guide』)

バス関連の API コマンド

構造化された DD オブジェクトから Simulink.Bus オブジェクトを作成することができます。逆に、Simulink.Bus オブジェクトから DD Variable オブジェクトを作成することもできます。

関連ドキュメント

- 「Basics on Modeling Buses via Data Store Blocks」(📖『TargetLink Customization and Optimization Guide』)
- 「tlSimulinkBusObject」(📖『TargetLink API Reference』)

Stateflow コード生成の改善

Superstep セマンティクスのサポート

ステートマシンで Superstep セマンティクスを使用することができます。これにより、すべての可能な状態遷移を同時に実行することができます。

関連ドキュメント

- 「Basics on Working with Superstep Semantics」(📖『TargetLink Preparation and Simulation Guide』)

複数の出力データをもつ関数のサポート

TargetLink では、複数の出力データをもつ Stateflow 関数（グラフィカル関数など）をサポートします。

関連ドキュメント

- 「Working with Stateflow」（📖『TargetLink Preparation and Simulation Guide』）

状態アクティビティ監視のサポートの改善

Stateflow Active State データを介して状態アクティビティを監視するために、次の 3 つのアクティビティモードがあります。`Self Activity`、`Child Activity` および `Leaf State Activity`。

TargetLink では、`Self Activity` および `Child Activity` アクティビティモードがサポートされ、状態のコード化のための C コード列挙型や C プリプロセッサマクロを含む、最適化された量産コードを生成します。ただし、`Leaf State Activity` はサポートされていません。

関連ドキュメント

- 「Basics on Working with Active State Data」（📖『TargetLink Preparation and Simulation Guide』）

モデルリファレンスの改善

モデルのルートレベルの Enable ブロック

この TargetLink バージョンでは、Trigger ブロックだけでなく Enable ブロックを介しても、参照先モデルの実行を制御することができます。

関連ドキュメント

- 「Basics on Model Referencing」（📖『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 準拠

MISRA C の改善

個々の MISRA C 規則への準拠を改善するために、TargetLink の固定小数点ライブラリとコードジェネレータ本体のさまざまな点が改善されています。これには次の内容が含まれます。

- 関数の戻り経路は 1 つのみです（MISRA C:2012 - 15.5）。
- 変数は常に最小の有効スコープで作成されます（固定小数点ライブラリ）。
- 到達不可能コードが減少されています（MISRA C:2012 - 2.1, 2.2, 14.3）。
- 戻り値または比較のためのキャストを追加することにより、MISRA C:2012 の変換規則のサポートが改善されています。
- TargetLink では、Bool の実装で C99 Bool タイプがサポートされます。「一般的な機能拡張および変更」(223 ページ)を参照してください。

関連ドキュメント

- なし

コード安定性の向上

アトミックサブシステムでの計算シーケンスの変更

アトミックサブシステムでの量産コードの計算シーケンスは、ブロックの実行順序が制御およびデータフローによって決定されていない場合は変更することができます。

理由　量産コードのブロックスケジューリングをモデルの変更に対してより柔軟に行います（Addfile または Unit Delay ブロックの追加など）。

移行の問題　以前の TargetLink バージョンでは、量産コードのソートを可能にするには、特定の状況および使用事例に応じて、一時的な回避策を適用する必要がありました。TargetLink については dSPACE 製品サポートにお問い合わせください。

コード効率性の向上

Assignment ブロックでの Init セクションの省略

Assignment ブロックのコードは、以下のコンテキストで変更されています。

- ブロックが AUTOSAR NvData 通信での NVRAM への再書き込み低減をモデル化するために使用されている場合。
- ブロックが複数の後継ブロックを持っていない場合。
- ブロックが反復するサブシステム内に配置されていない場合。

この場合、以下のコード例に示すように、TargetLink は output initialization を生成しません。

TargetLink 4.1 以前

```
p_Err = Rte_IRead_Run_PR_NV_Err_Err();
…
SCtrl2_Logical_Operator = SCtrl2_Relational_Operator &&
SCtrl2_Relational_Operator1;

for (Aux_ = 0; Aux_ < 10; Aux_++)
   {
      /* Assignment: pidcontroller/Efficient_Write_To_NVRAM - output
initialization */
      SCtrl1_Efficient_Write_To_NVRAM[Aux_] = p_Err[Aux_];
   }
   SCtrl1_Efficient_Write_To_NVRAM[0] = SCtrl2_Logical_Operator;

   p_Err = Rte_IRead_Run_PR_NV_Err_Err();
   if (p_Err[0] != SCtrl1_Efficient_Write_To_NVRAM[0]) {
      p_Err_a = Rte_IWriteRef_Run_PR_NV_Err_Err();
      p_Err_a[0] = SCtrl1_Efficient_Write_To_NVRAM[0];
   }
```

TargetLink 4.2

```
p_Err = Rte_IRead_Run_PR_NV_Err_Err();
…
SCtrl2_Logical_Operator = SCtrl2_Relational_Operator && SCtrl2_Relational_Operator
1;
SCtrl1_Efficient_Write_To_NVRAM[0] = SCtrl2_Logical_Operator;
   p_Err = Rte_IRead_Run_PR_NV_Err_Err();
if (p_Err[0] != SCtrl1_Efficient_Write_To_NVRAM[0]) {
   p_Err_a = Rte_IWriteRef_Run_PR_NV_Err_Err();
   p_Err_a[0] = SCtrl1_Efficient_Write_To_NVRAM[0];
}
```

利用効果　TargetLink は SCtrl1_Efficient_Write_To_NVRAM を完全に最適化でき、これによって効率性が向上します。

不要なキャストの省略

TargetLink は、以下のコンテキストで使用される場合、(Type) <Boolean operation, variable, macro>のような表現からキャストを削除します。

- if-、while-、do … while-、for-、および?:条件式
- 論理演算のオペランドとして

これはまた、上記のコンテキストで発生する 0 または 1 との比較の場合も成り立ちます。

さらに、TargetLink は、定数をキャストし値を変更しない switch 条件式でのキャストを省略します。通常、これによって、switch 文を case 分岐または default 分岐の 1 つのコードによって置き換えるなどのさらなる最適化が可能となります。

TargetLink 4.1 以前	TargetLink 4.2
`switch((Type) <constant>)`	`switch(<constant cast to T>)`

ローカルスコープをもつ変数の定義

TargetLink は、ローカルスコープの変数がループ複合文の開始部分で直接に定義されることがないようにします。これらの変数は、常に一貫して関数の冒頭で定義されます。最適化は、オプション ReduceScopeOfVariablesOnlyDownToFunctionLevel に応じて、関数ローカル変数をループ内のネストされた複合文の最小限のブロックスコープに制限することに注意してください。

TargetLink 4.1 以前	TargetLink 4.2

`void Subsystem(void)`
`{`
`  for (Aux__b = 0; Aux__b < Width; Aux__b++)`
`  {`
`  UInt8 TlAuxVar;`
`  /* … */`
`  }}`
`}`
	`void Subsystem(void)`
	`{`
	`  UInt8 TlAuxVar;`
	`  for (Aux__b = 0; Aux__b < Width; Aux__b++)`
	`  {`
	`  /* … */`
	`  }`
	`}`

関数ローカル変数およびマクロのすべての識別子は、関数範囲のスコープしか持たず、あたかも個々の変数またはマクロが関数の開始部分で定義されているかのように扱われます。これは主として、最適化されていないコードでの補助変数の識別子に影響を及ぼします。

TargetLink 4.1 以前

```
for (Aux__b = 0; Aux__b < BlockWidth; Aux__b++)
  {
    /* SLLutLocal: Default storage class for local variables | Width: 8 */
    UInt8 TlAuxVar; /* Index saturation for Direct Look-Up Table block */

TlAuxVar = C__U8SATI16_SATb(Sa1_In[Aux__b], 9, 0);

    /* Subsystem/TableAsParam_BlockVVW */
    BlockVar4[Aux__b] = Sa1_TableAsParam_BlockVVW_table[TlAuxVar];
  }
…
  for (Aux__d = 0; Aux__d < BlockWidth; Aux__d++)
  {
    /* SLLutLocal: Default storage class for local variables | Width: 8 */
    UInt8 TlAuxVar; /* Index saturation for Direct Look-Up Table block */

TlAuxVar = C__U8SATI16_SATb(Sa1_In[Aux__d], 9, 0);

    /* Subsystem/TableAsInput_TableAndBlockVVW */
    BlockVar3[Aux__d] = TableVar1[TlAuxVar];
  }
```

TargetLink 4.2

```
for (Aux__b = 0; Aux__b < BlockWidth; Aux__b++)
   {
      /* SLLutLocal: Default storage class for local variables | Width: 8 */
      UInt8 TlAuxVar_a; /* Index saturation for Direct Look-Up Table block */

TlAuxVar_a = C__U8SATI16_SATb(Sa1_In[Aux__b], 9, 0);

      /* Subsystem/TableAsParam_BlockVVW */
      BlockVar4[Aux__b] = Sa1_TableAsParam_BlockVVW_table[TlAuxVar_a];
   }
…

   for (Aux__d = 0; Aux__d < BlockWidth; Aux__d++)
   {
      /* SLLutLocal: Default storage class for local variables | Width: 8 */
      UInt8 TlAuxVar_b; /* Index saturation for Direct Look-Up Table block */

TlAuxVar_b = C__U8SATI16_SATb(Sa1_In[Aux__d], 9, 0);

      /* Subsystem/TableAsInput_TableAndBlockVVW */
      BlockVar3[Aux__d] = TableVar1[TlAuxVar_b];
   }
```

理由　異なるブロックの異なる補助変数識別子を含む、コードパターン全体での補助変数の一貫した命名を確立します。

論理表現の最適化

TargetLink は、ゼロ以外の定数を含む論理表現を最適化することができます。

TargetLink 4.1 以前	TargetLink 4.2
論理 AND	
`1 && boolexpr => boolexpr`	**<non-zero const>** `&& boolexpr => boolexpr`
論理 OR	
`1 || boolexpr => optmized to: 1`	**<non-zero const>** `|| boolexpr => optmized to: 1`

浮動小数点モジュロおよび剰余

Math ブロック向けに生成されたコードは、その入力または出力の 1 つが浮動小数点タイプである場合は変更することができます。

TargetLink 4.1 以前　出力が Float64、入力が Float32 および Int16 となるモジュロ関数を含む Math ブロックの場合は、次のようになります。

```
if (I16DenomIn != 0.F) {
      F64_F32I16_ = F32NumIn -
 (((Float64) I16DenomIn) * ((Float64) (Int32) (F32NumIn / ((Float64) I16DenomIn)))
);
...
```

出力が Int16 で飽和し、入力が Float32 となるモジュロ関数を含む Math ブロックの場合は、次のようになります。

```
if (F32DenomInInt != 0.F) {
...
} else {
    I16_F32F32_ = (Int16) F32NumIn;
}
```

TargetLink 4.2 出力が Float64、入力が Float32 および Int16 となるモジュロ関数を含む Math ブロックの場合は、次のようになります。

```
if (I16DenomIn != 0) {
      F64_F32I16_ = ((Float64) F32NumIn) -
 (((Float64) I16DenomIn) * ((Float64) (Int32) (((Float64) F32NumIn) / ((Float64) I
16DenomIn))));
```

出力が Int16 で飽和し、入力が Float32 となるモジュロ関数を含む Math ブロックの場合は、次のようになります。

```
if (F32DenomIn != 0.F) {
...
} else {
      if (F32NumIn > 32767.F) {
         I16_F32F32_ = 32767;
      }
      else {
         if (F32NumInInt < -32768.F) {
            I16_F32F32_ = -32768;
         }
         else {
            I16_F32F32_ = (Int16) F32NumIn;
         }
      }
   }
```

詳細 TargetLink の動作が次のようになるため、コードを変更することができます。

- すべてのオペランドを演算が計算されると想定されるタイプに明示的にキャストします。
- 演算結果またはそのオペランドの 1 つが Float64 である場合、Float64 を使用してすべての中間演算を計算します。
- 条件制御フローの else 分岐でコードを飽和します。

理由 精度を向上させ、MISRA C に準拠するために暗黙のキャストを抑制します。

列挙型サポートの向上

生成されたコードの列挙型

TargetLink は、モデル（Stateflow を含む）や生成されたコードのインプリメンテーションでの Simulink 列挙型の使用をサポートしています。Simulink 列挙型は、C *列挙型*、*マクロ*、または整数のいずれかとして実装されます。したがって、シミュレーション（MIL および SIL/PIL）することが可能です。

以下は、ENUM_TYPES デモモデルからのコードの例を示します。

```
typedef enum eBlinkMode_tag {
   eBlinkMode_NONE = 0,
   eBlinkMode_BLINKER_ON = 1,
   eBlinkMode_BLINKER_OFF = 2
} eBlinkMode;

   switch (Ca1_ChartMode) {
      case CHARTMODE_DO_BLINK: {
         if (!(Sa1_ENABLE)) {
            Ca1_BlinkState = eBlinkMode_NONE;
            Ca1_ChartMode = CHARTMODE_OFF;
         }
         else {
            if (Ca1_BlinkState == eBlinkMode_BLINKER_ON) {
               Ca1_BlinkState = eBlinkMode_BLINKER_OFF;
            }
            else {
               Ca1_BlinkState = eBlinkMode_BLINKER_ON;
            }
         }
```

関連ドキュメント

- 「Applying Simulink Enumeration Data Types in TargetLink」(🕮『TargetLink Preparation and Simulation Guide』)
- 「ENUM_TYPES」(🕮『TargetLink Demo Models』)

TargetLink Custom Code ブロックを使用する場合の柔軟性の向上

Custom Code テンプレートファイルでの文字列の代替

TargetLink Custom Code ブロックは、ブロックインスタンスごとに Custom Code テンプレートファイルで任意の文字列のプレースホルダの使用をサポートします。プレースホルダは、コード生成時にユーザが提供する任意の文字列に置き換えることができ、これにより Custom Code テンプレートファイルの柔軟性が向上します。

関連ドキュメント

- 「Basics on Custom Code and Custom Code Files」(🕮『TargetLink Preparation and Simulation Guide』)
- 「Code and Logging Page (Custom Code Block)」(🕮『TargetLink Block and Object Reference』)
- 「VARIABLE_INITIALIZATION」(🕮『TargetLink Demo Models』)

コードセクションからのブレースの削除

Custom Code セクションを囲むブレース(波括弧{})は、このコードセクションの始めに `nobraces` キーワードを使用することにより、削除することができます。

関連ドキュメント 「Basics on Custom Code and Custom Code Files」
(📖『TargetLink Preparation and Simulation Guide』)

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	4.2.2[1)] 4.2.1
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

AUTOSAR リリース	リビジョン
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 4.2 の新機能（TargetLink 4.1 のパッチ 1 以降でもサポート）

非同期クライアントサーバ通信

非同期クライアントサーバ通信をモデル化し、量産コードに `Rte_Call` および `Rte_Result` API 関数の呼び出しを生成することができます。

関連ドキュメント

- 「Modeling Client-Server Communication」(📖『TargetLink AUTOSAR Modeling Guide』)
- 「TargetLink と TargetLink Data Dictionary API 関数の変更」(238 ページ)
- 「AR_COLLISION_DETECTION」(📖『TargetLink Demo Models』)

クライアントサーバ通信でのデータトランスフォーマ

クライアントサーバ通信での変換エラーロジックをモデル化することができます。

関連ドキュメント

- 「Modeling Transformer Error Logic」(📖『TargetLink AUTOSAR Modeling Guide』)
- 「AR_COLLISION_DETECTION」(📖『TargetLink Demo Models』)

非スカラーインターランナブルバリアブル

非スカラーインターランナブルバリアブルをモデル化できるようになりました。

関連ドキュメント

- 「Modeling Interrunnable Communication」(📖『TargetLink AUTOSAR Modeling Guide』)
- 「インターランナブルバリアブル仕様の移行」(240 ページ)

SystemDesk およびその他の AUTOSAR ツールでのラウンドトリップの改善

TargetLink と SystemDesk 間のラウンドトリップが改善されています。DD ExportRule オブジェクトを介してエクスポート時に暗黙的に生成される AUTOSAR エレメントに関するパッケージ情報を指定することができます。ソフトウェアコンポーネントのインターナルビヘイビアは、個別のファイルにエクスポートすることができます。さらに、単位も個別のファイルやパッケージにエクスポートすることができます。

関連ドキュメント

- 「Basics on Packages」(📖『TargetLink Interoperation and Exchange Guide』)
- 「ExportRule」(📖『TargetLink Data Dictionary Reference』)

その他の AUTOSAR 機能

インポートおよびエクスポートの改善

TargetLink では、カテゴリが `BITFIELD_TEXTTABLE` に設定された CompuMethod エレメン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。

関連ドキュメント

- なし

ランナブルサブシステム外部の Data Store Memory ブロック

Data Store Memory ブロックを使用して、*ランナブルサブシステム*(📖『TargetLink Glossary』)の外部に NvData 通信をモデル化することができます。

関連ドキュメント
■ なし

シミュレーション向け Data Store ブロック

Data Store Read および Data Store Write ブロックをランナブルサブシステムの外部に配置して、NvData またはインターランナブル通信をモデル化する Data Store Memory ブロックにアクセスすることができます。これにより、シミュレーション目的でシミュレーションフレームの通信変数に直接アクセスすることができます。

ターゲットシミュレーション（PIL）

項目の一覧

本章の内容

PIL シミュレーション向けの高速ボーレート

仮想 COM ポート

TargetLink では、PIL シミュレーション向けの TargetLink 評価ボード（EVB）に／からデータを転送するための従来より高速なボーレートがサポートされます。仮想 COM ポートは、115,200 の 2 倍、4 倍、および 8 倍高速な通信ボーレートをサポートしますが、実際の通信ボーレートはハードウェアに依存します。また、ダウンロード時のボーレートには影響しないため、その最大限度は従来の 115,200 と変わりません（EVB によってはこれより低くなる場合があります）。

関連ドキュメント 「Topic Settings」（🕮『TargetLink Tool and Utility Reference』）

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 4.2 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。

マイクロコントローラファミリ	コンパイラ	新規	変更なし	廃止
ARM CortexM3	Keil	5.2	—	5.1
C16x	TASKING	—	8.6	—
MPC57xxVLE	Diab	—	5.9	—
	GreenHill	—	2014	—
MPC560xVLE	Diab	—	5.9	—
	GreenHill	—	2012, 2014	—
RH850	GreenHill	2015	—	2014
S12X	Cosmic	—	4.8	—
	Metrowerk	—	5.1	—
SH2	Renesas	—	9.3	—
SH2A-FPU	Renesas	—	9.4	—
TriCore17xx	TASKING	6.0	3.2	5.0
TriCore2xx	TASKING	6.0	—	5.0
	GCC	—	4.6	—
V850	GreenHill	2015	—	2014
XC22xx	TASKING	—	3.0	—

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、「Combinations of Evaluation Boards, Microcontrollers, and Compilers」(🕮『TargetLink Evaluation Board Hardware Reference』)を参照してください。

注記

有効なソフトウェア保守サービス(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。

Data Dictionary とデータ管理

項目の一覧

本章の内容

Data Dictionary の改善点

Windows コマンドプロンプトを介した DD Comparison ペイン(GUI)

Windows コマンドプロンプトを介して 2 つの DD ファイルをグラフィカルに比較することができます。Data Dictionary Manager とその DD Comparison ペインを、バージョン管理システム内などから、MATLAB 環境を使用せずにスタンドアロンで開始することができます。

関連ドキュメント

- 「Basics on Comparing and Merging DD Workspaces in the DD Comparison Pane」(🕮『TargetLink Data Dictionary Basic Concepts Guide』)

カスタムコマンドの呼び出し

Data Dictionary 内からユーザ独自のコマンドを呼び出し、任意の数のパラメータを渡すことができます。カスタムコマンドには、外部プログラムやユーザ固有の M スクリプト、その他の呼び出す必要がある準備用スクリプトまたは関数を使用することができます。

関連ドキュメント

- 「How to Specify Custom Command Calls」(🕮『TargetLink Data Dictionary Basic Concepts Guide』)
- 「DD_CUSTOM_COMMAND」(🕮『TargetLink Demo Models』)

組込みヘルプ

組込みヘルプが、Data Dictionary Manager のいくつかのオブジェクトやプロパティに追加されています。

非整数タイプの処理の改善

Plain Variable ダイアログを使用して、Data Dictionary で非整数タイプにローカルスケーリングを指定することができます。

さらに、Adjust to Typedef コンテキストメニューコマンドも非整数タイプに使用することができます。

関連ドキュメント

- Adjust to Typedef

Data Dictionary のあらかじめ設定された Code Generator オプションセット

DD オプションセットを介したコード生成の調整

TargetLink Data Dictionary（dsdd_master_advanced.dd などのあらかじめ設定されたシステムテンプレートに基づく）は、MISRA C への高度な準拠や速度対コードサイズなどの特定の目的のために、量産コード生成を容易に調整できる各種オプションセットを提供します。

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 4.2 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。

- ForceBooleanOperandsInBooleanOperations

  追加の *ゼロ以外*（`x != 0`）の比較を生成されたコードに挿入し、非ブール型の論理操作や制御フロー表現にブール型コンテキストを強制し、そのようなモデリング状況でも MISRA C への準拠を達成します。

関連ドキュメント

- 「ForceBooleanOperandsInBooleanOperations」（📖『TargetLink Block and Object Reference』）。

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」(236 ページ)を参照してください。

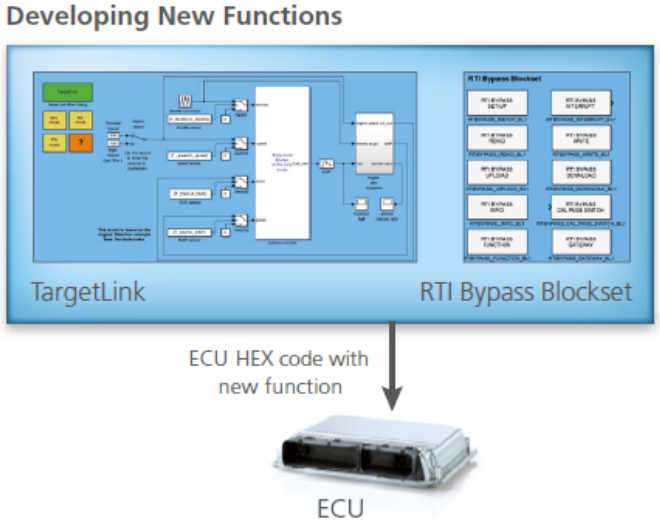
ツールチェーンの統合

項目の一覧

本章の内容

RTI Bypass ブロックによるモデリングとオンターゲットバイパス処理用のコード生成

TargetLink では、RTI Bypass Blockset を使用してモデリングを行うことができます。RTI Bypass Blockset を使用して、ECU インターフェースを設定し、バイパス処理やテストのための新しい ECU 機能を実装することができます。さらに、TargetLink のコードジェネレータをコード生成に使用し、ターゲットハードウェア上で直接(オンターゲットバイパス処理)、新しい制御機能の追加や既存の制御機能の置き換えを行うことができます。これは、ターゲット ECU の空きリソースを使用して実行されます。また、オンターゲットバイパス処理は、しばしば*オンターゲットプロトタイピング*とも呼ばれます。これは、追加の(外部)ハードウェアを使用せずに ECU 機能を開発し、置き換えることができるためです。



関連ドキュメント

- 「Basics on Modeling with RTI Bypass Blocks and Generating Code for On-Target Bypassing」(🕮『TargetLink Interoperation and Exchange Guide』)
- 次の RTI Bypass Blockset のドキュメントも参照してください:「RTI Bypass Blockset に関する一般情報」(🕮『RTI Bypass Blockset リファレンス』)

関連トピック

リファレンス

- 「RTI Bypass Blockset に関する一般情報」(🕮『RTI Bypass Blockset リファレンス』)

Functional Mock-up Unit(FMU)用のバイナリファイルのビルド

TargetLink サブシステムから生成されエクスポートされた Functional Mock-up Unit(FMU)に 32 ビットおよび 64 ビット用の事前にビルドされた Windows DLL ファイルを含めることができます。これにより、FMU をビルドできない FMI 準拠ツールでこれらの FMU を使用することができます。

関連ドキュメント

- 「Basics on Exporting FMUs from TargetLink」(🕮『TargetLink Interoperation and Exchange Guide』)
- 「TargetLink FMU Export」(🕮『TargetLink Tool and Utility 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

ドキュメント生成の改善

HTML のサポート TargetLink は、生成されるドキュメントへの HTML ファイルの追加をサポートします。

ドキュメントのインクリメンタル生成 TargetLink はドキュメントのインクリメンタル生成をサポートします。これにより、ドキュメント生成時に既に生成されたドキュメントの一部を統合モデルなどに再利用することができます。

選択可能なベクトルグラフィックのサポート TargetLink は、生成されるドキュメントでの SVG をサポートします。SVG はデフォルト形式です。PNG も使用することができます。`tldoc` コマンドの `ImageFormatType` プロパティでファイル形式を設定することができます。ただし、EPS のサポートは終了しました。「廃止された TargetLink の機能 」(258 ページ)も参照してください。

組込みコマンドのユーザビリティの向上 Autodoc Customization ブロックダイアログで、組込みコマンドのあらかじめ定義された設定を選択することができます(設定を入力する必要はありません)。TargetLink が、選択されたコマンドを正しい構文を使用して編集フィールドに入力します。

ブロックの分類とフィルタリング　Autodoc Customization ブロックを分類してフィルタリングできるため、ドキュメントの生成時に TargetLink によってどのブロックが考慮され、どのブロックが無視されるかを簡単に制御することができます。

フックスクリプトによるカスタマイズの拡張　新しいフックスクリプトを使用して Autodoc Customization ブロックを管理することができます。たとえば、Autodoc Customization ブロックが生成される順序を設定することができます。フックスクリプトは、ブロックレベルで指定された競合する設定を常に上書きします。

関連ドキュメント

- 「Basics on Documentation Generation」(🕮『TargetLink Interoperation and Exchange Guide』)

ダイアログを開く速度の向上

TargetLink は、Main Dialog やその他のブロックダイアログを TargetLink 4.1 の 2 倍の速さで開きます。

関連ドキュメント

- なし

ddv IC 構造の Simulink データタイプの定義

TargetLink を使用して、IC 構造(ddv)の生成に使用される構造化された DDVariable オブジェクトのコンポーネントから Simulink データタイプを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 Object and a Simulink Bus」(🕮『TargetLink Customization and Optimization Guide』)

`TargetConfig.xml` でサポートされる C99 _Bool の新しいコードタイプ

C99 `_Bool` データタイプを使用でき、TargetLink の `Bool` typedef が `_Bool` にマッピングされます。

このコードタイプを導入することにより、一部の MISRA C チェッカーで量産コードの管理が容易になります。現在、TargetLink は、MISRA に準拠した独自の `Bool` data typedef を提供しています。ただし、一部の MISRA チェッカーはこのデータタイプを許容しないため、違反が誤検出される原因となります。C99 `_Bool` データタイプは、MISRA 準拠チェック時に常に有効な `boolean` データタイプとして認識されます。これにより、MISRA 規則違反の誤検出の数を減少させることができます。

ヒント

`TargetConfig.xml` 設定を調整するため(特に PIL シミュレーションで)、ターゲットを新規作成するか、または既存のターゲットを複製することができます。

新しい汎用 C99 コード生成ターゲット　さらに、新しい Generic C99 コード生成ターゲットを TargetLink Main Dialog で使用することができます。これには、C99 `_Bool` データタイプがデフォルトで使用されます。

注記

`_Bool` の使用は、C コンパイラにも依存します。詳細については、使用するターゲットコンパイラのマニュアルを参照してください。

SIL コンパイルでは、サポートされている MEX/SIL コンパイラは、`_Bool` データタイプをデフォルトのコンパイラオプションでサポートします。

関連ドキュメント

- 「Example of Controlling the Generation of the File Containing TargetLink Base Data Types」(🕮『TargetLink Customization and Optimization Guide』)
- 「How to Clone Target/Compiler Combinations to Outside the TargetLink Installation」(🕮『TargetLink Customization and Optimization Guide』)

Windows-1252 文字セットのサポート

TargetLink は、Windows-1252（CP1252、西ヨーロッパ）文字セットもサポートします。

関連ドキュメント

- 「Basics on the Code Generation Report」(🕮『TargetLink Preparation and Simulation Guide』)
- 「How to Specify the Used Character Set」(🕮『TargetLink Interoperation and Exchange Guide』)

構造体変数での信号の挿入とトンネリング

信号の挿入またはトンネリング時に、変数のコンポーネントのみでなく構造体変数全体にアクセスすることができます。

関連する Simulink.Signal オブジェクトは、構造体変数の階層ツリーに対応する階層ツリーをもつ Bus データタイプである必要があります。ノードの名前は考慮されません。

関連ドキュメント

- 「Basics on Injecting or Tunneling Signals During Simulation」(🕮『TargetLink Preparation and Simulation Guide』)
- 「Modeling Buses via Data Store Blocks」(🕮『TargetLink Customization and Optimization Guide』)

TargetLink デモ

新しいデモ

TargetLink には次の新しいデモが付属しています。

AR_COLLISION_DETECTION　この新しいデモは次の機能を示します。

- 非同期クライアントサーバ通信をモデル化し、量産コードに `Rte_Call` および `Rte_Result` API 関数の呼び出しを生成します。
- クライアントサーバ通信での変換エラーロジックをモデル化します。

「AR_COLLISION_DETECTION」(🕮『TargetLink Demo Models』)を参照してください。

AUTODOC_GENERATION　この新しいデモは次の機能を示します。

- 専用の M スクリプトを使用してドキュメント生成を呼び出しドキュメントの一般的構造および内容を制御する方法
- Autodoc Customization ブロックを使用してドキュメント化するファンクションを制御する方法
- ユーザが用意したコンテンツ(テキスト、画像、HTML コマンドなど)をドキュメント生成プロセスに挿入する方法

「AUTODOC_GENERATION」(🕮『TargetLink Demo Models』)を参照してください。

DD_CUSTOM_COMMAND　この新しいデモは次の機能を示します。

- Data Dictionary 内からユーザ独自のコマンドを呼び出すことができます。追加の引数も指定することができます。カスタムコマンドには、外部プログラムやユーザ固有の M スクリプト、その他の呼び出す必要がある準備用スクリプトまたは関数を使用することができます。dd_customcommand デモには、Data Dictionary Manager の Pool 領域での 4 つの異なるカスタムコマンド呼び出しが含まれています。

「DD_CUSTOM_COMMAND」(🕮『TargetLink Demo Models』)を参照してください。

EMBEDDING_EXTERNAL_CODE　この新しいデモは次の機能を示します。

- このデモでは、Function ブロックや Custom Code ブロックなどを介して外部コードを組込むさまざまな方法を示します。

「EMBEDDING_EXTERNAL_CODE」(🕮『TargetLink Demo Models』)を参照してください。

ENUM_TYPES　この新しいデモは次の機能を示します。

- Stateflow Active State データなどによるモデルの列挙型および生成された量産コードの使用

「ENUM_TYPES」(📖『TargetLink Demo Models』)を参照してください。

FUNCTION_VARIANTS　この新しいデモは次の機能を示します。

- モデリング中または量産コード生成前などに、機能バリアントのさまざまなバインディングタイミングを選択することができます。

「FUNCTION_VARIANTS」(📖『TargetLink Demo Models』)を参照してください。

VARIABLE_INITIALIZATION　この新しいデモは次の機能を示します。

- `Main Restart` 関数などを介して変数を初期化するさまざまな方法

「VARIABLE_INITIALIZATION」(📖『TargetLink Demo Models』)を参照してください。

拡張デモ

次のデモには新機能のデモが含まれます。

BUS_STRUCT　この変更されたデモは次の機能も示します。

- Data Store を備えたバスを操作する方法

「BUS_STRUCT」(📖『TargetLink Demo Models』)を参照してください。

API 関数とフックスクリプト

項目の一覧

本章の内容

新しい API 関数

API 関数	目的
`tlEnumDataType`	Simulink 列挙型データタイプを Data Dictionary にインポートします。

API 関数	目的
tlExecuteDDCustomCommand	Data Dictionary の CustomCommand または CustomCommandGroup オブジェクトを実行します。
tlSimulinkBusObject	Simulink.Bus オブジェクトを TargetLink で処理します。
tlStartCallbackWithTimer	タイマー内でコールバックを開始します。

関連ドキュメント:

- 「tlEnumDataType('CreateDDTypedef', propertyName, propertyValue, ...)」(📖『TargetLink API Reference』)
- 「tlExecuteDDCustomCommand(ddCmdObj, propertyName, propertyValue, ...)」(📖『TargetLink API Reference』)
- 「tlSimulinkBusObject」(📖『TargetLink API Reference』)
- 「tlStartCallbackWithTimer(callback)」(📖『TargetLink API Reference』)

新しいフックスクリプト

TargetLink には、次の新しいフックスクリプトが提供されています。

フックスクリプト	目的
tl_post_add_operationresultprovidersubsystem_hook	AUTOSAR フレームモデル生成をカスタマイズすることができます。
tl_post_autodoc_filter_hook	検出された Autodoc Customization ブロックのリストをフィルタリングするためなどに、ユーザ固有のコマンドを入力することができます。

関連ドキュメント

- 「tl_post_add_operationresultprovidersubsystem_hook」(📖『TargetLink File Reference』)
- 「tl_post_autodoc_filter_hook」(📖『TargetLink File Reference』)

TargetLink 4.2 および TargetLink Data Dictionary 4.2 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スクリプト

TargetLink 4.1 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。🕮『TargetLink Migration Guide』を参照してください。一般的および TargetLink 4.2 のリリース固有の移行に関する情報は、両者のドキュメントとも同一です。

モデルの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」(235 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、必要に応じて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

MATLAB 関連の変更

MATLAB R2016b で修正された機能

起動動作の変更　MATLAB では起動スクリプトの実行順序が変更されています。

- MATLAB R2016a 以前では、`startup.m` スクリプトが最初に実行され、dSPACE 固有の初期化スクリプト `dsstartup.m` および `dspoststartup.m` が後に続きます。
- MATLAB R2016a 以降では、`startup.m` スクリプトは、dSPACE 固有の初期化スクリプトの後に最後に実行されます。

この変更によって起動動作が変化する場合があります。

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 4.2 への移行

以前の TargetLink バージョンでの TargetLink モデルの使用

新しい TargetLink バージョンで保存されたモデルは、それ以前の TargetLink バージョンでは開くことができません。

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。

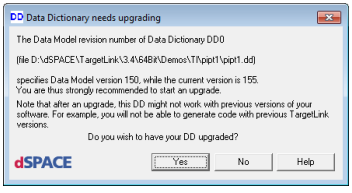
ただし、間接アップグレードを実行することができます。最初に、古いライブラリ、モデル、DD ファイルを TargetLink 3.1 以上のバージョンに移行します。その後、それを TargetLink 4.2 にアップグレードすることができます。

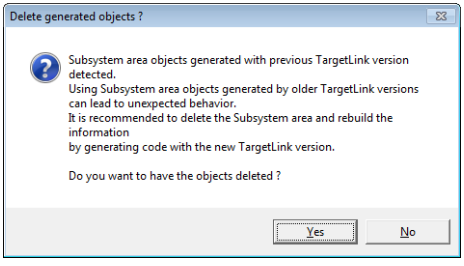
TargetLink 4.2 にアップグレードする前に、TargetLink 3.5 にアップグレードすることをお勧めします。

TargetLink 3.1 以降からの直接アップグレード

TargetLink 4.2 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」(231 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」(232 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」(232 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」(235 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 4.2 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 4.2 でライブラリを開きます。

関連ドキュメント

- 「How to Make TargetLink User Libraries Upgrade-Capable」(📖『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')` 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。詳細については、「tlUpgrade(propertyName, propertyValue, ...)」(📖『TargetLink API Reference』)を参照してください。

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」(232 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

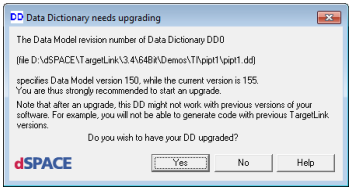
アップグレードしていない古い Data Dictionary ファイルを含む TargetLink モデルを開く場合は、Data Dictionary ファイルをアップグレードする必要があります。

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

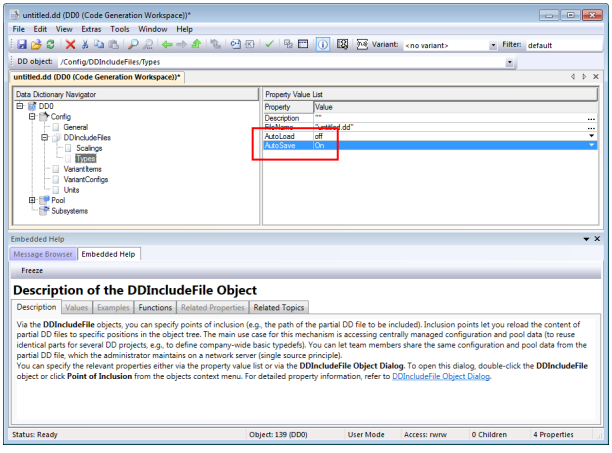
1 モデルおよび参照する TargetLink Data Dictionary を開くか、MATLAB コマンドウインドウで `dsdd('Open',<DDFile>)` と入力します。

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。



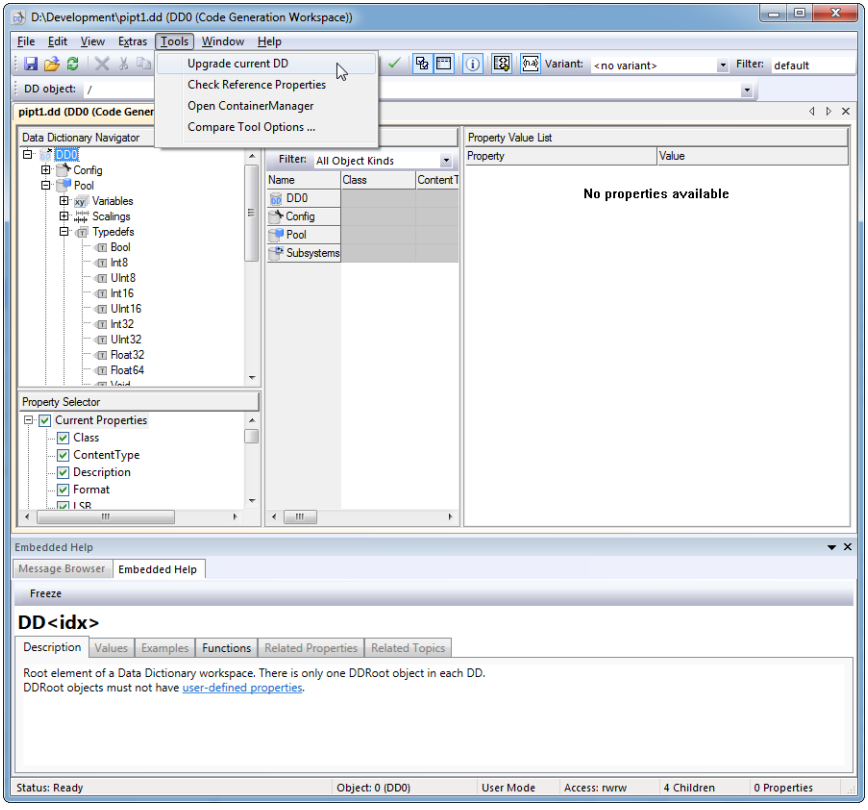
これにより、Data Dictionary とインクルード DD ファイルのアップグレード後、Data Dictionary を保存するときにアップグレードしたインクルード DD ファイルが保存されます。Object Explorer を使用して、複数のインクルード DD ファイルのこれらのプロパティを設定することができます。

ヒント

[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をお勧めします。

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。

注記

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 4.2 への移行」(230 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

次の Code Generator オプションは TargetLink から削除されました。

削除されたオプション
ConsiderFunctionClassesForBlockDiagramBasedSwitchOptimization
ConsiderStateflowAuxiliariesForVariableSharing
EnableVariableVectorWidths
TreatAllStateflowFunctionsAsWeakAtomic

変更された Code Generator オプション

Code Generator オプション	以前のデフォルト	新しいデフォルト
EnableReportGeneration	`off`	`on`
ReportFailedFunctionReuseVariablePropagation	`on`	`off`
ReportMismatchingPortBlockSignalSpecifications	`on`	`off`
ReportVariablesExcludedFromPIM	`on`	`off`

推奨される互換性設定

新しい TargetLink 4.2 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。

- ForceBooleanOperandsInBooleanOperations を `off` に設定します。

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、DD CodegenOptionSet 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）。TL 4.1 以降では、DD CodegenOptionSet 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。また、モデルベースのオプション設定の代わりに使用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)。

注記

$TL_A \Rightarrow TL_B \Rightarrow TL_C$ 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 のバージョン A、B、および C のデフォルト値が 9、11、および 13 で、オプション値がバージョン A の 11 と等しい場合は、バージョン C へのアップグレードにより次のようにオプション値が変更されます。

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」(220 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 関数とフックスクリプト

TargetLink と TargetLink Data Dictionary API 関数の変更

tl_set

移行の問題 AllowLibraryBlockModification プロパティを on に設定することにより、tl_set を使用して、同じ API 呼び出しのマスクコールバックおよび類似のコールバック内からライブラリアイテムのブロックプロパティを変更することができます。

```
tl_set(<block>,…, 'AllowLibraryBlockModification', 'on',…)
```

関連ドキュメント tl_set(「tl_set(hBlock, property_1, value_1, ..., property_n, value_n)」(📖『TargetLink API Reference』)を参照)

tl_generate_swc_model	**移行の問題**　非同期クライアントサーバ通信の導入により、「tl_generate_swc_model」(📖『TargetLink API Reference』) API 関数の AddOperationCallTriggerPort プロパティの名前が AddOperationSubsystemTriggerPort に変更されています。 **解決策**　ユーザスクリプトを調整します。 TargetLink は、Data Dictionary ファイル／コンテンツを最新バージョンにアップグレードすると、/Pool/Autosar/Config/FrameModelGeneration/AddOperationCallTriggerPort プロパティの名前を AddOperationSubsystemTriggerPort に自動的に変更します。 **関連ドキュメント** ■ 「非同期クライアントサーバ通信」(215 ページ) ■ 「Modeling Client-Server Communication」(📖『TargetLink AUTOSAR Modeling Guide』) ■ 「tl_generate_swc_model」(📖『TargetLink API Reference』)
tldoc	RTF ドキュメントの生成はサポートされません。tldoc('Convert')コマンドの CreateRTFFile プロパティは廃止されました。このプロパティを過去に使用した場合は、スクリプトを編集して tldoc('Convert')コマンドからこのプロパティを削除する必要があります。代わりに、HTML または PDF を使用します。 ドキュメントを手作業で後処理するには、適当な HTML エディタまたは PDF エディタを使用します。ただし、手作業による編集手順が不要となるように、ドキュメント生成をカスタマイズすることを検討してください。TargetLink ドキュメント生成プロセスの新機能を利用することができます。「一般的な機能拡張および変更」(223 ページ)を参照してください。 **関連ドキュメント** ■ 「Basics on Customizing the Generated Documentation」(📖『TargetLink Interoperation and Exchange Guide』) ■ 「tldoc('Convert', propertyName, propertyValue, ...)」(📖『TargetLink API Reference』) ■ 「廃止された TargetLink の機能 」(258 ページ)
データモデルフィルタルールファイル	**移行の問題**　TargetLink Data Dictionary のデータモデルが変更されているため、既存のデータモデルフィルタルールファイルには無効なエレメントが含まれている場合があります。TargetLink 4.1 に付属する以下のファイルが影響を受けます。 ■ DD_Filter_Admin.xml ■ DD_Filter_AR_User.xml ■ DD_Filter_NonAR_NonRTOS_User.xml

解決策 MATLAB コマンドウインドウの API を介してフィルタルールファイルをチェックすることができます。

単一ファイルのチェック	フィルタ規則セットのチェック [1]

dsdd_free;
dsdd('ReadFilterRuleSet', 'file', '<myFile>.xml');
ds_error_register(dsdd('GetMessageList'));
ds_msgdlg('update');
	dsdd_free;
	dsdd('ReloadFilterRuleSets');
	ds_error_register(dsdd('GetMessageList'));
	ds_msgdlg('update');

[1] TargetLink Preferences Editor の Data Dictionary - Filter Rules で定義されたディレクトリにすべてのファイルが含まれています。

TargetLink は、TargetLink の Message Browser にエラーに関する情報を表示します。各エラーには以下の情報が含まれているため、適当な XML 対応エディタを使用してエラーを修正することができます。

- ファイル名
- 行番号
- 列番号

AUTOSAR に関する移行上の注意点

項目の一覧 本章の内容

インターランナブルバリアブル仕様の移行

移行の問題

インターランナブルバリアブル(IRV)は、現在、専用の DD InterRunnableVariable オブジェクトを使用してモデリングされ、EXPLICIT_IRV、DISP_EXPLICIT_IRV、IMPLICIT_IRV および DISP_IMPLICIT_IRV(アクセス関数機構)と呼ばれるあらかじめ設定された DD VariableClass オブジェクトの 1 つを参照する DD Variable オブジェクトは使用されません。

以前の Data Dictionary の準備

IRV 仕様を TargetLink 4.2 に移行する前に、以前の Data Dictionary を次のように準備する必要があります。

- IRV を表す DD Variable オブジェクトを含むすべての DD VariableGroup オブジェクトは、その `RelatedVariables/InterrunnableVariablesRef` プロパティを介して *1 つのみの* DD SoftwareComponent オブジェクトによって参照される必要があります。
- 初期化する必要のある IRV ごとに、AUTOSAR 定数仕様が DD Variable オブジェクトの以下のプロパティによって十分に定義されていることを確認してください。
  - Value
  - InitConstantName
  - InitConstantFileName
  - InitConstantPackage
  - InitConstantUuid
- Data Dictionary を保存します。

Data Dictionary のアップグレード

以上の準備が終われば、Data Dictionary をアップグレードすることができます。TargetLink は、DD VariableGroup オブジェクトに含まれる各 DD Variable オブジェクトに対して、次の手順を実行します。DD VariableGroup は、DD SoftwareComponent オブジェクトの InterrunnableVariablesRef プロパティ(RelatedVariables サブツリーに含まれる)を介して参照されています。

- DD InterRunnableVariable オブジェクトを作成し、このオブジェクトをソフトウェアコンポーネントの InterRunnableVariables サブツリーに追加します。
- IRV 固有のプロパティの値とすべてのカスタムプロパティを DD Variable オブジェクトから DD InterRunnableVariable オブジェクトにコピーします。
- DD InterRunnableVariable オブジェクトの UpgradeFromVariableRef カスタムプロパティで、以前の DD Variable オブジェクトを参照します。
- 以前の DD Variable オブジェクトが IRV を初期化する定数を定義している場合は、このオブジェクトが IRV の定数に一致する DD Variable オブジェクトを検索します。
  - このオブジェクトが適切な Variable オブジェクトを検出した場合は、TargetLink は InterRunnableVariable オブジェクトの InitValueRef プロパティでこのオブジェクトを参照します。

- このオブジェクトが適切な Variable オブジェクトを検出しなかった場合は、TargetLink は新しいオブジェクトを作成し、InterRunnableVariable オブジェクトの InitValueRef プロパティでこのオブジェクトを参照します。

- すべてのオブジェクトが処理された後で、以前の DD Variable オブジェクトの Class プロパティを次のように設定します。

以前の変数クラス	新しい変数クラス
EXPLICIT_IRV	default
IMPLICIT_IRV	
DISP_EXPLICIT_IRV	DISP
DISP_IMPLICIT_IRV	

- 以前の DD VariableClass オブジェクトを Data Dictionary から削除します。

注記

- UpgradeFromVariableRef プロパティ、またはこのプロパティを参照する DD Variable オブジェクトを削除または解除しないでください。

  この情報は、モデルのアップグレード時に必要となります。

- AUTOSAR ファイルまたは SWC コンテナからインポートされたデータのみを含む新しい DD ワークスペースを作成する必要がある場合は、必ず、各 DD InterRunnableVariable オブジェクトの UpgradeFromVariableRef プロパティとこのプロパティを参照する DD Variable オブジェクトを新しいワークスペースにマージしてください。

モデルのアップグレード

Data Dictionary をアップグレードした後に、モデルをアップグレードすることができます。モデルのアップグレードは、モデルを新しい TargetLink バージョンで最初に開いたときに自動的に実行されます。付加されている Data Dictionary がアップグレードされない場合は、TargetLink は Data Dictionary をアップグレードするようにプロンプトを表示します。

モデルのアップグレード時に、TargetLink は、ポートブロックの [AUTOSAR]ページと Data Store Memory ブロックの[Output]ページで、DD InterRunnableVariable オブジェクトの UpgradeFromVariableRef プロパティを介して参照される DD Variable オブジェクトの参照を検索し、以下の設定を行います。

Port ブロック	Data Store Memory ブロック
以前の DD Variable オブジェクトを新しい DD InterRunnableVariable オブジェクトへの参照に置き換えます。	以前の Variable オブジェクトへの参照をブロックの[Output]ページから無効にし、ブロックの[AUTOSAR]ページで以下の設定を行います。 ■ AUTOSAR 通信の使用チェックボックスを選択します。 ■ Kind プロパティを `InterRunnable` に設定します。 ■ 新しい DD InterRunnableVariable オブジェクトを参照します。

以前の変数のページ

モデルのアップグレードが成功したことを確認した後に、この DD を以前の DD Variable オブジェクトから除去し、DD InterRunnableVariable オブジェクトの UpgradeFromVariableRef プロパティを削除することができます。

TargetLink で用意されている、このための API は次のとおりです。

```
errorCode = dsdd(
   'DeleteObsoleteInterRunnableVariables',
   <DdSwcObjectIdentifier>);
```

注記

この API 関数の実行は、警告なしに以下の手順を実行します。この操作は取り消すことができません。

- DD InterRunnableVariable オブジェクトの UpgradeFromVariableRef プロパティを介して参照されるすべての DD Variable オブジェクトを削除します。
- DD Variable オブジェクトを含む DD VariableGroup オブジェクトが空の場合は、このオブジェクトを削除します。
- UpgradeFromVariableRef プロパティを DD InterRunnableVariable オブジェクトから削除します。

関連ドキュメント

- 「Modeling Interrunnable Communication」(🕮『TargetLink AUTOSAR Modeling Guide』)
- 「非スカラーインターランナブルバリアブル」(216 ページ)
- 「DeleteObsoleteInterRunnableVariables」(🕮『TargetLink Data Dictionary Reference』)

トランスフォーマエラーコード仕様の移行

移行の問題

AUTOSAR では、トランスフォーマエラーコードがリビジョン 4.2.1 から 4.2.2 に変更されています。

これらの変更は自動更新時に解決されないため、注意が必要です。

解決策

dsdd_master_autosar4.dd [System] DD テンプレートの最新バージョンには、新しいトランスフォーマエラーコードが含まれています。

このテンプレートを新しいワークスペースで開き、[DD Comparison]ペインを使用してエラーコードを DD プロジェクトファイルにマージすることができます。これらのテンプレートは以下の DD VariableGroup にあります。

```
/Pool/Variables/AUTOSAR/TransformerError
```

コードの変更

コードの変更

ブール型コンテキストでの非ブール型オペランド

ブール型コンテキストでの非ブール型オペランドのコードが変更されています。この変更により、クリーンでないモデリング状況での論理演算や制御フロー条件、定数、および列挙型が影響を受けます（非ブール型の信号の供給など）。

TargetLink 4.1 以前	TargetLink 4.2

`Int16Var && BoolVar`
`if(Int16Var) { … }`
`42`
`0`
列挙型定数はサポートされません。
	`(Int16Var != 0) && BoolVar`
	`if (Int16Var != 0) { … }`
	`42 != 0`
	`0 != 0`
	列挙型変数または列挙型定数で、`0` を表す列挙型定数を提供するオペランドでは、この定数は次のように使用されます。
	`enum BasicColors = {RED = 0, GREEN = 1, …}`
	`BasicColors color;`
	`color != RED`
	`GREEN != RED`

理由 MISRA C に準拠するためです。

移行の問題 TargetLink バージョン 4.1 以前で生成されたコードに戻すには、ForceBooleanOperandsInBooleanOperations Code Generator オプションを無効にします。

列挙型コードセクションの変更

列挙型のコードセクションは、include のコードセクションの直後に移動しました。この変更によって、ソースおよびヘッダファイルが影響を受けます。

理由 列挙型は、include セクションの後にあるその他の定義セクションで必要となる場合があります。

インターランナブル通信の補助変数

コードの可読性を改善するため、インターランナブル通信用に作成された補助変数は、DD InterRunnableVariable オブジェクトと同じ名前になります。

TargetLink 4.1 以前	TargetLink 4.2

`sint32 Aux_;`
`  Aux_ = (sint32) Sa2_Rescaler;`
`  Rte_IrvWrite_Run_MyIrv(Aux_);`
	`sint32 MyIrv;`
	`  MyIrv = (sint32) Sa2_Rescaler;`
	`  Rte_IrvWrite_Run_MyIrv(MyIrv);`

ポインタ変数 ポインタである IRV 変数の名前には、p_の接頭辞が付きます。

Math ブロックの新機能

Math ブロックの pow 関数を使用する場合、コーナーケースでの Simulink と TargetLink の量産コード間の計算の相違が削除されました。

TargetLink 4.1 以前	TargetLink 4.2

`if ((Sa1_u_ <= 0.F) && (((Float64) Sa1_v) != floor((Float64) Sa1_v))) {`
`…`
`}`
	`if ((Sa1_u_ < 0.F) && (((Float64) Sa1_v) != floor((Float64) Sa1_v))) {`
	`…`
	`}`

Discrete-Time Integrator ブロック

-inf/inf 以外の Forward Euler および Limits に設定された Discrete-Time Integrator ブロックで生成されるコードは、次のように変更することができます。

TargetLink 4.1 以前	TargetLink 4.2
`UInt16 X_Sa1_DTI_FW_Sat = 10 /* 5. */ /* LSB: 2^-1 OFF: 0 MIN/MAX: 2 .. 730 */;`	`Int16 X_Sa1_DTI_FW_Sat = 10 /* 5. */ /* LSB: 2^-1 OFF: 0 MIN/MAX: -98 .. 630 */;`

理由　範囲計算が修正されたことにより、コード生成時に選択されたデータタイプを適合させます。

External reset で生成されたコードが変更されています。

TargetLink 4.1 以前	TargetLink 4.2

/* Discrete Integrator: Subsystem/DTI: Condition for either edge trigger */
if ((Sa1_Reset >= 0) && (X_Sa1_DTI_TriggerIn <= 0) && ((Sa1_Reset != X_Sa1_DTI_TriggerIn) &&
(Sa1_DTI_LastEvent != 1))) {
Sa1_DTI_LastEvent = 1;
}
else {
if ((Sa1_Reset <= 0) && (X_Sa1_DTI_TriggerIn >= 0) && ((Sa1_Reset != X_Sa1_DTI_TriggerIn) &&
(Sa1_DTI_LastEvent != -1))) {
Sa1_DTI_LastEvent = -1;
}
else {
Sa1_DTI_LastEvent = 0;
}
}
	/* Discrete Integrator: Subsystem/DTI: Condition for either edge trigger */
	if ((Sa1_Reset **>** 0) && (X_Sa1_DTI_TriggerIn <= 0) && ((Sa1_Reset != X_Sa1_DTI_TriggerIn) &&
	(Sa1_DTI_LastEvent != 1))) {
	Sa1_DTI_LastEvent = 1;
	}
	else {
	if ((Sa1_Reset <= 0) && (X_Sa1_DTI_TriggerIn **>** 0) && ((Sa1_Reset != X_Sa1_DTI_TriggerIn) &&
	(Sa1_DTI_LastEvent != -1))) {
	Sa1_DTI_LastEvent = -1;
	}
	else {
	Sa1_DTI_LastEvent = 0;
	}
	}

理由　正の値のみが立ち上がりエッジとして考慮される Simulink の動作に一致させます。

Rate Limiter

Rate Limiter ブロックの RemainderState 変数で生成されたコードは、次のように変更することができます。

TargetLink 4.1 以前	TargetLink 4.2
static **Int8** XRem_Sa1_Rate_Limiter1; static **Int16** XRem_Sa1_Rate_Limiter2;	static **Int16** XRem_Sa1_Rate_Limiter1; static **Int32** XRem_Sa1_Rate_Limiter2;

理由　小さすぎるデータタイプが誤って選択されていたのを修正します。

追加のキャストを剰余計算に挿入することもできます。

TargetLink 4.1 以前	TargetLink 4.2
Aux_S16 = Aux_U16 % 100;	Aux_S16 = (Int16) (Aux_U16 % 100);

理由　MISRA C に準拠するためです。

Stateflow での関数の再利用および pointer-to-struct

Stateflow での関数の再利用および pointer-to-struct に関するコード最適化が改善されています。

TargetLink 4.1 以前	TargetLink 4.2
pISV->comp = 0; pISV->comp = 1;	pISV->comp = 1;

 # combined #コメントの順序

関数呼び出しの上にある# combined #コメントの順序を変更することができます。

TargetLink 4.1 以前	TargetLink 4.2

/* call of function: SimpleCall/Subsystem
  # combined # update(s) for inport SimpleCall/Subsystem/InPort
  # combined # Gain: SimpleCall/Gain
# combined # Gain: SimpleCall/Gain1 	
  # combined # TargetLink outport: SimpleCall/OutPort */
foo(a, h);
	/* call of function: SimpleCall/Subsystem
	  # combined # update(s) for inport SimpleCall/Subsystem/InPort
	# combined # Gain: SimpleCall/Gain1
	# combined # Gain: SimpleCall/Gain
	  # combined # TargetLink outport: SimpleCall/OutPort */
	foo(a, h);

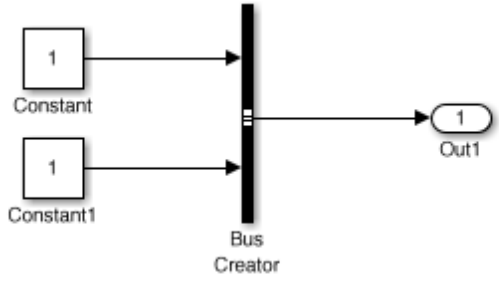
従来の初期化モードでの IF 変数

「*classic initialization mode*」(📖『TargetLink Glossary』)では、拡張されていない Outport ブロックで生成されるコードが変更されています。このブロックは、直前にある Bus Creator ブロックによって作成される非仮想バスからデータを受信します。これにより、次のように IF 変数が違う名前となる場合があります。

TargetLink 4.1 以前	TargetLink 4.2

void Sa2_Subsystem(void)
{
  /* update of variable(s) associated with TL_Root/Subsystem/Constant */
  IF_Sa2_a = 1;
  /* update of variable(s) associated with TL_Root/Subsystem/Constant1 */
  IF_Sa2_b = 1;
}
	static void Sa2_Subsystem(void)
	{
	  /* Outport: TL_Root/Subsystem/Out1 */
	  IF_Sa2_Out1 = 1;
	  IF_Sa2_Out1_a = 1;
	  }

使用モデル　先行するテーブルのコード例は、次のようにモデルから生成されます。



非仮想バスの信号は、a および b と呼ばれます。

Unit Delay Reset Enabled

外部初期値の使用チェックボックスが選択されている Unit Delay Reset Enabled ブロックで生成されるコードが変更されています。このブロックの状態変数を計算する場合、TargetLink は FirstRun 変数の enable condition をチェックしません。

TargetLink 4.1 以前	TargetLink 4.2

/* Subsystem/UDRE: reset and external initial value condition */
if ((Sa3_R != 0) || Sa3_Subsystem_FirstRun) {
  /* Subsystem/UDRE: state initialization */
  X_Sa3_UDRE = Sa3_IV;
}
 	
/* Subsystem/UDRE: output */
Sa3_UDRE = X_Sa3_UDRE;
 	
/* Subsystem/UDRE: **enable condition** */
if ((Sa3_E != 0) && ((!(Sa3_R != 0)) **&& (!(Sa3_Subsystem_FirstRun != 0))**)) {
  /* Subsystem/UDRE: state update */
  X_Sa3_UDRE = Sa3_u;
}
	/* Subsystem/UDRE: reset and external initial value condition */
	if ((Sa3_R != 0) || Sa3_Subsystem_FirstRun) {
	  /* Subsystem/UDRE: state initialization */
	  X_Sa3_UDRE = Sa3_IV;
	}
	
	/* Subsystem/UDRE: output */
	Sa3_UDRE = X_Sa3_UDRE;
	
	/* Subsystem/UDRE: enable condition */
	if ((Sa3_E != 0) && (!(Sa3_R != 0))) {
	  /* Subsystem/UDRE: state update */
	  X_Sa3_UDRE = Sa3_u;
	}

理由　内部および外部初期値でのブロックの動作が、以下の条件が適用される場合に整合性がありません。

- R 入力が値 false を受信した場合。
- E 入力が値 true を受信した場合。

この場合、内部および外部初期値は、MIL および SIL シミュレーションモードで同一の値になります。

コードコメントとしての AUTOSAR バージョン

AUTOSAR モードでコードを生成する場合、TargetLink は、AUTOSAR バージョンをコードファイルの開始ヘッダに配置します。

理由　AUTOSAR バージョンはコード生成に影響を及ぼすため、明確に見える位置に表示する必要があります。

反復変数に代わる補助変数

反復変数は、最適化が有効な場合は、補助変数によって置き換えられる場合があります。

TargetLink 4.1 以前	TargetLink 4.2
Sa3_For_Iterator_it[1]	Aux_S32_a[2]

1) $S_$B_it という名前に設定されます。
2) Aux_TR という名前に設定されます。

理由　コードジェネレータの最適化が変更されています。

変数およびマクロの定義

変数およびマクロの定義が、次のように異なる場合があります。

- enum タイプの変数を使用することができます。
- マクロ定義は関数内でソートされ、変数クラスに応じてコードコメントを受け取ります。
- パラメータ化されたマクロ（マクロアクセス関数など）は、グループ化されてソートされます。

理由

- C 列挙型をサポートします。
- コードの可読性によるものです。

while-サブシステムの反復変数

do-while-サブシステムで生成される反復変数は、*while* ループの末尾で常にインクリメントされます。また、これにより変数がその他の条件によっても影響を受けます。

TargetLink 4.1 以前	TargetLink 4.2

`while (Sa2_While_Iterator_cond && (Sa2_While_Iterator_it <= 4)) {`
`  …`
`Sa2_While_Iterator_cond = 1;`
`Sa2_While_Iterator_it++;`
 	
`  /* Unit delay: TL_SS/For Iterator Subsystem/Unit Delay */`
`  X_Sa2_Unit_Delay = (Int16) Sa2_While_Iterator_it;`
`}`
	`while (Sa2_While_Iterator_cond && (Sa2_While_Iterator_it <= 4)) {`
	`  …`
	`  /* Unit delay: TL_SS/For Iterator Subsystem/Unit Delay */`
	`  X_Sa2_Unit_Delay = (Int16) Sa2_While_Iterator_it;`
	
	`Sa2_While_Iterator_cond = 1;`
	`Sa2_While_Iterator_it++;`
	`}`

理由 Unit Delay ブロックに関連して間違ったコードが生成される可能性がありました。

指数のコード形式

指数のコード形式が、1 または 2 桁で十分な場合は、3 桁から 2 桁に変更されています。

理由 Microsoft の CRT ランタイムライブラリの変更に従い、浮動小数点精度を向上させ、C 標準への準拠を改善します。詳細については、https://msdn.microsoft.com/en-us/library/bb531344.aspx#BK_CRT を参照してください。

テーブルマップの再利用

ルックアップテーブル関数のテーブルマップ変数が、関数の再利用に関して次のように変更されています。

TargetLink 4.1 以前	TargetLink 4.2
テーブルおよび軸は、再利用されるシステムに残っているポインタを介して参照されます。	ポインタによって参照されるテーブルおよび軸は、再利用構造体を含むファイルに移動しました。

理由 コンポーネントベースの開発を容易にします。

SateReset を含む If Action および Switch Case Action サブシステム

StateReset を含む If Action および Switch Case Action サブシステムで生成されるコードが変更されています。現在実行されているシステムを保存する変数では、不要な初期化を行いません。

TargetLink 4.1 以前	TargetLink 4.2

`UInt8 Sa1_If_System = 0;`
`if (Sa1_tlIn1 > 16384 /* 0.5 */) {`
`  Sa1_If_System = 1;`
`...`
`} else {`
`  Sa1_If_System = 2;`
`}`
	`UInt8 Sa1_If_System;`
	`if (Sa1_tlIn1 > 16384 /* 0.5 */) {`
	`  Sa1_If_System = 1;`
	`...`
	`} else {`
	`  Sa1_If_System = 2;`
	`}`
	`LastSystem = Sa1_If_System;`

理由 MISRA C に準拠するためです。

変化検出機能付き状態遷移図

`input`、`Datastore`、または `Machine data` に関する変化検出機能付きの状態遷移図は、`FirstRun` と呼ばれる新しい変数を受信します。

理由 Stateflow と生成された量産コードパターンとの間の間違った相違を削除します。

新しいコードコメント

TargetLink は、以下のコードエレメントに対して追加または変更されたコードコメントを生成します。

- グループ化宣言および変数または関数の定義に対して、追加のコメントが生成されます。
- PreBlockStatements および PostBlockStatements は、常に空行で囲まれます。
- include や typedef などの空でない CodeSection-StatementTable の後は、常に空行が入ります。

理由 コードパターン全体の整合性を向上させます。

代数的簡略化

`UInt16Var > 0` のような式の代数的簡略化について、TargetLink のコードが変更されています。

TargetLink 4.1 以前	TargetLink 4.2
`UInt16Var`	`UInt16Var != 0`

理由 MISRA C 準拠のためにブール型コンテキストを維持し、さらなるコード最適化を容易にします。

浮動小数点オペランドを使用する論理演算

少なくとも 1 つの浮動小数点タイプのオペランドを含む論理演算では、TargetLink はすべてのオペランドを演算内に含まれている最大の浮動小数点タイプにキャストしません。

TargetLink 4.1 以前	TargetLink 4.2[1)]

`Sal_F32F64AND = ((Float64) F32Var) && F64Var;`
`Sal_F32I16OR = ((Float32) I16Var) || F32Var;`
`Sal_F32ScaledAND = F32Var && ((((Float32) ScaledI16Var) * 0.5F) + 2.F); // ScaledI16Var: Lsb = 0.5, Offset = 2.0`
`Sal_F32IntValueAND = F32Var && 1.F;`
`Sal_F32FlpValueOR = 1.25F || F32Var;`
	`Sal_F32F64AND = (F32Var != 0.F) && (F64Var != 0.);`
	`Sal_F32I16OR = (I16Var != 0) || (F32Var != 0.F);`
	`Sal_F32ScaledAND = (F32Var != 0.F) && ((ScaledI16Var + 4) != 0); //`
	`ScaledI16Var: Lsb = 0.5, Offset = 2.0`
	`Sal_F32IntValueAND = (F32Var != 0.F) && (1 != 0);`
	`Sal_F32FlpValueOR = (1.25F != 0.F) || (F32Var != 0.F);`

1) ForceBooleanOperandsInBooleanOperations Code Generator オプションを on.に設定。

理由 MISRA C に準拠するためです。

Stateflow の Scaling-Invariant グラフィカル関数

Stateflow の Scaling-Invariant グラフィカル関数で生成されたコードでは、最適化が実行されない場合があります。

理由 これは、副作用を伴う偽のコード最適化を防止するための安全策です。

解決策 グラフィカル関数には[Optimization]プロパティが `SIDE_EFFECT_FREE` に設定されている関数クラスを使用します。

注記

この関数が動作するすべての変数が関数ローカルで、関数がイベントをトリガしない場合のみ、この解決策を使用してください。

Simulink 列挙型定数

以下のモデリングエレメントでは、Simulink 列挙型定数のコードが変更されています。これらは整数から列挙型データタイプの列挙型定数に変更され、TargetLink によって暗黙的に生成されます。

- Simunlink の Switch-Case ブロックの Case conditions
- Stateflow アクション言語の Simulink 列挙型定数
- TargetLink の Multiport Switch ブロックの Data port indices
- 次のように指定された TargetLink の Switch ブロックの Threshold:
  - DD Variable オブジェクトを参照しない。
  - Class が `default` に設定されている。
- 次のように指定された TargetLink の Constant ブロック:
  - DD Variable オブジェクトを参照しない。
  - [Allow signal specification]チェックボックスが選択されていない。
  - Class が `default` に設定されている。

次のコードが Multiport Switch ブロックから生成されます。

TargetLink 4.1 以前

```
switch (Sa1_InPort1) {
      case 0: {

         Sa1_Merge = Sa1_InPort2;
         break;
      }
      case 2:
      case 3: {

         Sa1_Merge = Sa1_InPort3;
         break;
      }
      default: {

         Sa1_Merge = Sa1_InPort4;
         break;
      }
   }
```

TargetLink 4.2

```
switch (Sa1_InPort1) {
      case BasicColors_Red: {

         Sa1_Merge = Sa1_InPort2;
         break;
      }
      case BasicColors_Green:
      case BasicColors_Blue: {

         Sa1_Merge = Sa1_InPort3;
         break;
      }
      default: {

         Sa1_Merge = Sa1_InPort4;
         break;

      }
   }
```

TargetLink の列挙型データタイプの定義は、次のようになります。

```
typedef enum BasicColors_tag {
   BasicColors_Red = 0,
   BasicColors_Yellow = 1,
   BasicColors_Green = 2,
   BasicColors_Blue = 3,
   BasicColors_Orange = 4,
   BasicColors_Black = 5,
   BasicColors_Pink = 6,
   BasicColors_Brown = 7,
   BasicColors_Magenta = 8
} BasicColors; /* Description: Enumeration type derived from Simulink type
BasicColors */
```

理由　列挙型データタイプを完全にサポートします。

解決策　バージョン 4.1 以前の TargetLink によって生成されたコードに戻すには、Filter プロパティが ALWAYS に設定され、DD EnumImplementation オブジェクトを参照する DD EnumTemplate オブジェクトを作成します。

さらに、モデル準備のためにマッピングファイルを EnumImplementation に調整することもできます。

再利用される変数およびデータバリアントの制約レンジ

対応する DD Variable オブジェクトが次のように指定されている場合は、関数の再利用およびデータバリアントに関連して変数のコードが変更されています。

- Min プロパティが指定されていない。
- Max プロパティが指定されていない。
- VariableClass オブジェクトの Const プロパティが on に設定されている。

- VariableClass オブジェクトの Info プロパティが none、readonly、または bypassing_readonly に設定されている。

TargetLink は、これらの変数のレンジ幅制限の限界値に従って、そのタイプのレンジを使用します。これにより、コードの効率性が低下する場合があります。

理由 関数の再利用およびデータバリアントに関連して、コードが誤って過度に最適化されるのを防止します。

移行の問題 このような変数で効率性の低いコードが見られる場合は、Min および Max の値を指定することを検討してください。

タイプ定義および変数の宣言に関するコードコメント

次のコードエレメントのコードコメントが変更されています。

- タイプ定義の制限(レンジ、スケーリング)は、次の例のようにタイプ定義でドキュメントされます。

```
+typedef Uint16 ADC; /*
     Unit: m
LSB:   2^-11 OFF:   0 MIN/MAX:   -10 .. 10 */
```

- 宣言に関するコードコメント内の順序が変更されています。スケーリングや記述に関する情報の場所が変更されています。

```
Int16 Sal_InPort3; /*
     <BlockComment>
     Unit: Gigawatt
LSB:   2^-8 OFF:   0 MIN/MAX:   0 .. 0.99
Description: wash dish */
```

さらに、余分な空白が OFF および MIN/MAX から削除されています。

- 変数宣言での;や、構造体初期化子での,は、オプションのコメントの後ではなく、識別子の直後に配置されます。

```
Int16 Sal_InPort3; /* Description: something */

struct my_tiny_struct S {
     5, /* Description: five */
     6
  };
```

これは、数値定数の物理値を記述するコメントには影響を及ぼしません。

```
Int32 MyVal = 37 /* 0.145 */; /* LSB: 2^-8 OFF: 0 MIN/MAX: 0 .. 0.99 */
```

- デフォルトのスケーリング(LSB = 1、Offset = 0、単位なし、すべての寸法に適用)を使用する変数のコメントには、レンジが明示的に指定されている場合を除いて、スケーリングコメントは含まれません。
- レンジが明示的に指定された Float 変数には、MIN/MAX コメントは付きますが、スケーリングコメントは付きません。
- マクロ変数(RDI マクロを除く)には、明示的に指定されたレンジの MIN/MAX コメントのみが付けられ、計算されたレンジのコメントは付きません。

■ Float 変数には、次の例のように Unit コメントが付きます。

```
Float32 Sa1_Out1; /* Unit: <unit> */
```

理由 コードパターン全体の整合性を確立し、可読性を向上させます。

条件分岐内へのコードの移動

TargetLink は、関数内に含まれている、条件によって実行される制御フローのすべての分岐で定義されている変数を、この変数が別の関数内でアクセスされる場合は、同じ関数に含まれている後続の制御フローの条件分岐内に移動しません。

これにより、コードの効率性が低下する場合があります。

理由 最適化の不足による誤ったコード計算を防止します。

ExtendedLifeTimeOptimization

TargetLink は、条件によって実行されるサブシステムの out パラメータで ExtendedLifeTimeOptimization 最適化を実行します。

TargetLink 4.1 以前	TargetLink 4.2

Foo(Int * out)
{
Int B;
B = ... ;
.... //some Code
*out = B;
}
	Foo(Int * out)
	{
	Int B;
	B = ... ;
	*out = B;
	.... //some Code
	}

理由 最適化およびコード効率性を改善します。

この最適化は、AUTOSAR RTE の Rte_IWriteRef および Rte_IrvIWriteRef 関数での参照によって渡される戻り変数へのアクセスにも影響を及ぼします。

Stateflow およびブール型へのキャスト

Stateflow での非ブール型のブール型へのキャストは、TargetLink の Bool 基本タイプのコードタイプに関わりなく、生成される量産コードで効果的に boolean になるよう強制されます。

TargetLink 4.1 以前	TargetLink 4.2
F32Var = (Float32)((**Bool**)**I16Var**);	F32Var = (Float32)(**I16Var != 0**);

理由 MISRA C 準拠を改善し、MIL および SIL シミュレーションでの異なる動作を修正します。

詳細 このコード変更は、ForceBooleanOperandsInBooleanOperations Code Generator オプションによって影響を受けません。

Stateflow 状態のコード化

[Create output for monitoring property]が Self activity に設定されている状態では、TargetLink は常にこの状態のコードアクティビティに、この状態の Active State Data を使用します。

TargetLink がブール型の Active State データを S の 1 つの子に使用する場合は、ブール型の Active State データを S のすべての子に使用します。

理由 コードパターン全体の整合性を向上させます。

複数の出力をもつブロックの継承

ブロックが複数の入力をもち、[Inherit properties]チェックボックスが選択されている場合、以下のブロックのコードが変更されています。

- Switch ブロック
- Multiport Switch ブロック
- MinMax ブロック
- Merge ブロック

これは、以下の条件のときに、Bitwise Operator ブロックで生成されるコードにも影響を与える場合があります。

- ブロックの[Use bit mask]チェックボックスが選択されていない。
- ブロックが DD Variable または ReplaceableDataItem オブジェクトを参照しない。

TargetLink 4.2 の出力データタイプは、次の方法で決定されます。

- すべての入力で有効な信号仕様が同じ最小値／最大値の制約をもつ場合、出力データタイプは、有効な信号仕様をもつ最初のブロック入力のデータタイプになります。
- そうでない場合は、有効な信号仕様が異なる最小値／最大値の制約をもつ少なくとも 1 つの入力がある場合、出力データタイプは、有効な信号仕様をもつ最初のブロック入力の基本データタイプになります。

TargetLink 4.1 以前

```
DataTypeSecondInput Sa1_SwitchScalar /* LSB: 2^0 OFF:  0 MIN/MAX:  -100 .. 100 */;
```

TargetLink 4.2

```
DataTypeFirstInput Sa1_SwitchScalar; /* LSB: 2^0 OFF: 0 MIN/MAX: -100 .. 100 */
```

理由　コードパターン全体の整合性を向上させます。

その他の影響　この変更によるその他の影響の 1 つとして、構造体タイプが複数の入力をもつブロックを通して継承されない場合があります。構造体変数は普通の変数の集まりとは異なる最適化が行われるため、これによりコード最適化が影響を受ける場合があります。

TargetLink 4.1 以前	TargetLink 4.2

`struct StructUDT_tag Sa1_SwitchStruct;`
 	
`/* Switch: Subsystem/SwitchStruct */`
`if (Sa1_CtrlIn >= 0) {`
 	
`/* Switch: Subsystem/SwitchStruct */`
`Sa1_SwitchStruct.a = Sa1_InPortNoUDTWithMinMax;`
`Sa1_SwitchStruct.b = Sa1_InPortInt16;`
`}`
`else {`
 	
`/* Switch: Subsystem/SwitchStruct */`
`Sa1_SwitchStruct = X_Sa1_Unit_Delay;`
`}`
	`Int16 Sa1 SwitchStruct; /* LSB: 2^0 OFF: 0 MIN/MAX: -100 .. 100 */`
	`Int16 Sa1 SwitchStruct_a;`
	
	`/* Switch: Subsystem/SwitchStruct */`
	`if (Sa1_CtrlIn >= 0) {`
	`/* Switch: Subsystem/SwitchStruct */`
	`Sa1_SwitchStruct = Sa1_InPortNoUDTWithMinMax;`
	`Sa1_SwitchStruct_a = Sa1_InPortInt16;`
	`}`
	`else {`
	`/* Switch: Subsystem/SwitchStruct */`
	`Sa1_SwitchStruct = X_Sa1_Unit_Delay.a;`
	`Sa1_SwitchStruct_a = X_Sa1_Unit_Delay.b;`
	`}`

解決策　TargetLink 4.1 で量産コードに構造体変数が必要な場合は、次のいずれかの方法を使用します。

- 複数の入力をもつブロックのすべての入力で対応するバスエレメントが必ず同じデータタイプをもつようにします（最初のバスエレメントがすべての入力で同じデータタイプをもち、2 番目のバスエレメントがすべての入力で同じデータタイプをもつなど）。
- Data Dictionary で明示的に構造体を指定し、モデルの各ブロックでタイプまたは変数を参照します。

その他

移行に関するその他の注意点

要件情報

要件情報は、表示可能なブロックプロパティとして保存されなくなりました。

代わりに、TargetLink ブロックに HasRequirementInfos フラグが含まれます。これにより、以下の API 関数を介して要件情報をもつブロックのリストを取得することができます。

```
hBlockList = tl_find('<System>', 'HasRequirementInfos', true)
```

要件情報を追加または削除するには、tlRequirementInfo API 関数を使用します。

システムクリア時の TargetLink スケーリングプロパティの再マッピング

移行の問題 TargetLink がスケーリングされた変数（LSB != 2^0 または Offset != 0.0）を指定し、Simulink がスケーリングされていないデータタイプ（ビルトインデータタイプの int8、int16、、single、double、boolean などのいずれか）を指定する場合、TargetLink 4.2 は Simulink のデータタイプにスケーリングされたタイプを設定せず、メッセージ E03819 を表示します。

解決策 どの TargetLink スケーリングプロパティが Simulink スケーリングパラメータにマッピングできないかを知りたい場合、[Clear System from TargetLink]ダイアログの[Verbose output]チェックボックスを選択します。

詳細出力モードでは、マッピングできない各プロパティに関するメッセージが TargetLink に表示されます。

次に例を示します。

- Simulink's settings alread match（Simulink の設定が既に一致しています）
- Simulink specifies property inheritance（Simulink がプロパティの継承を指定しています）
- Simulink specifies a bus or enum datatype（Simulink がバスまたは列挙型データタイプを指定しています）

注記

大きなモデルでは、メッセージ数が非常に多くなる場合があります。

廃止事項

項目の一覧

本章の内容

廃止された TargetLink の機能

V-ECU の OSA としての完全なビルド

TargetLink は、ECU を OSA ファイルとしてビルドしません。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 に引き継がれます。

RTF ドキュメントの生成

TargetLink は、リッチテキスト形式(RTF)でドキュメントを生成することができません。RTF ファイルが必要な場合は、HTML 生成機能と Microsoft Word などのサードパーティ製ツールを組み合わせて使用し、RTF ファイルを作成してください。

ドキュメント生成用 EPS グラフィック

EPS ファイルはドキュメント生成でサポートされなくなりました(`ImageFormatTypetldoc`)。HTML や PDF ドキュメントで SVG を使用して、高解像度のスクリーンショットを取得することができます。「一般的な機能拡張および変更」(223 ページ)を参照してください。

A2L のインポート

TargetLink の Data Dictionary では、A2L ファイルの Subsystem 領域へのデフォルトインポートが利用できなくなりました。

廃止された制限事項

TargetLink 4.2 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

sim コマンドによる MIL シミュレーションの開始

MATLAB R2015b（64 ビット）では、Simulink の sim コマンドを使用して関数内から MIL シミュレーションを開始すると、エラーにつながります。通常は、代わりに TargetLink の tl_sim コマンドを使用することをお勧めします。

Windows ユーザ名の非 ASCII 文字

MATLAB R2014a の 64 ビットバージョンの場合、Windows ユーザ名に Müller の ü のような非 ASCII 文字が含まれていると、TargetLink デモのスタートページが正常に表示されません。このような場合、右側ペインに何も表示されません。

ブロック固有の制限事項

Matrix Concatenate ブロック

TargetLink は、Mode ブロックパラメータが Multidimensional array に設定されている場合のみ、このブロックをサポートします。

Data Store ブロック

Data Store Memory/Data Store Read/Data Store Write ブロックでは、次の制限事項が適用されます。

- 入力および出力ポートの数は、1 つに制限されます。
- データストアに対する*部分的な*読み取り／書き込みはサポートされません。

Data Store Memory ブロック

TargetLink では、データタイプがバスを指定する Data Store Memory ブロックはサポートされません。

コード生成の制限事項

LCC コンパイラでの CTC コードカバレッジ計測不可

LCC コンパイラをサードパーティの CTC Testwell 製のツールで使用してコードカバレッジ計測を行うことはできません。ただし、代わりに TargetLink 固有のコードカバレッジ計測を使用することができます。

AUTOSAR の制限事項

インターランナブルバリアブルのデータタイプ

TargetLink で指定されるインターランナブルバリアブルはスカラーである必要があります。

Stateflow

複数のデータ出力を持つグラフィカル関数

複数の出力データを持つグラフィカル関数の定義および呼び出しを行うことはできません。

廃止された API 関数

関数	ステータス	代替プロパティ
get_tloptions	Error[1)]	tl_global_options
tl_build_vecu	Error[1)]	-
tl_compile_vecu	Error[1)]	-
tl_vecu_compiler	Error[1)]	-
dsdd('AddCodegenOptions'…)	Error[1)]	dsdd('AddCodegenOptionSet'…)
dsdd('CreateCodegenOptionSetsTLPredefinedOptionset'…)	Error[1)]	dsdd('CreateTLPredefinedOptionSet'…)

[1)] この関数は TargetLink から削除されました。

互換性の考慮 これに応じて、ユーザスクリプトとツールチェーンを適合させてください。

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

"Clean code"および"Do not log anything"

ロギング対象として選択した変数は、完全には最適化することができません。TargetLink でグローバルロギングオプション Do not log anything または Log according to block data を指定してコードを生成する場合、テストを円滑にするため、コードの完全な最適化は行われません。コードの違いはログマクロに関する部分のみです。この違いは、TargetLink Main Dialog ブロックの[Code Generation]ページにある

Clean code チェックボックスに関するもので、いかなる場合も完全なコード最適化を有効にします。

特別な `Do not log anything` 動作は、今後の TargetLink バージョンで削除される予定です。

Stateflow でのユーザ状態フラグ

Stateflow Active State データが同様の機能を提供し、こちらのほうが使い勝手が良いため、Stateflow での TargetLink 独自のユーザ状態フラグ機能のサポートは、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、TargetLink バージョン 4.2 の後に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「*classic initialization mode*」(📖『TargetLink Glossary』)のサポートは、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。

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`**

注記

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。

廃止された Code Generator オプション

次の Code Generator オプションは廃止され、今後の TargetLink バージョンで削除される予定です。

- 「SideEffectFreeAnalysisThreshold」(📖『TargetLink Block and Object Reference』)
- 「TreatAllForcedAtomicSubsystemsAsWeakAtomic」(📖『TargetLink Block and Object Reference』)
- 「DisableFunctionsAsAnalysisBoundaries」(📖『TargetLink Block and Object Reference』)
- 「CreateRestartFunctions」(📖『TargetLink Block and Object Reference』)

VEOS

項目の一覧

本章の内容

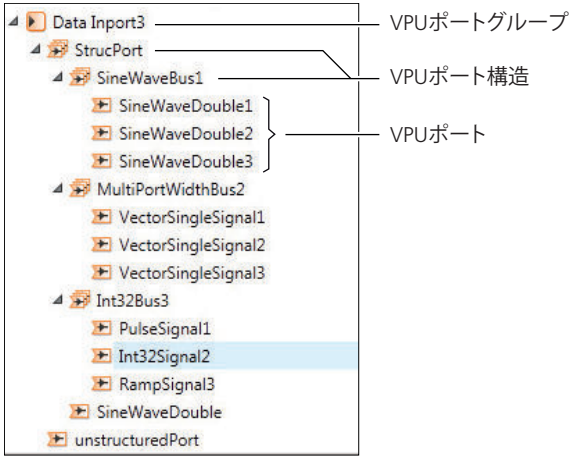
VEOS 3.7 の新機能

本章の内容

VPU ポートの階層表示

VEOS Player は、VPU の VPU ポートを*階層的に*(VPU ポートグループと VPU ポート構造とともに)表示します。VPU ポートグループと VPU ポート構造は、元のモデルのエレメントから決定されます。

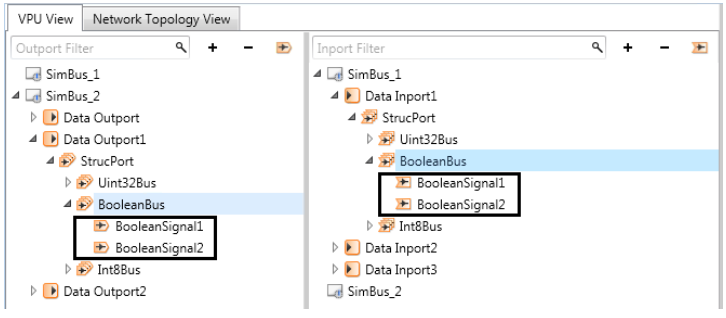
次の図は、オフラインシミュレーションアプリケーションの VPU ポートの階層表示の例を示しています。



構造エレメントのドラッグによる VPU ポートの接続

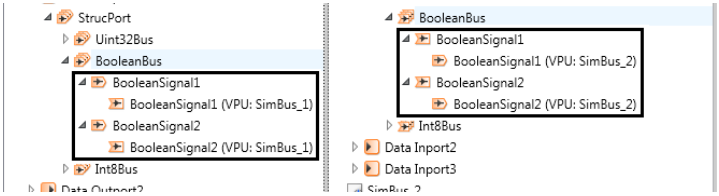
VEOS Player を使用して、1 つの VPU ポート構造を別の VPU ポート構造にドラッグすることにより、VPU ポートを接続することができます。さらにポートの名前が同じ場合は、VEOS Player は、最初の構造のすべての VPU ポートを 2 番目の構造の VPU ポートに自動的に接続します。

下の図に例を示します。



1 つのツリービューから BooleanBus VPU ポート構造を、別のツリービューの BooleanBus VPU ポート構造にドラッグすると、VEOS Player は 2 つの VPU ポートを接続します。

次の図は、ドラッグアンドドロップ後の接続された VPU ポートを示します。



FMU 内部メッセージのログの有効化／無効化

FMU を VEOS Player にインポートする場合、コンパイラオプションを使用して FMU 内部メッセージのログを有効化／無効化することができます。

「Import」(🕮『VEOS Player Reference』)を参照してください。

自動化を介した VEOS Player のバージョン情報の取得

`IVersionInformation` インターフェースを使用して、メジャー、マイナー、およびメンテナンスバージョンを含む VEOS Player のバージョンに関する詳細情報を取得することができます。

注記

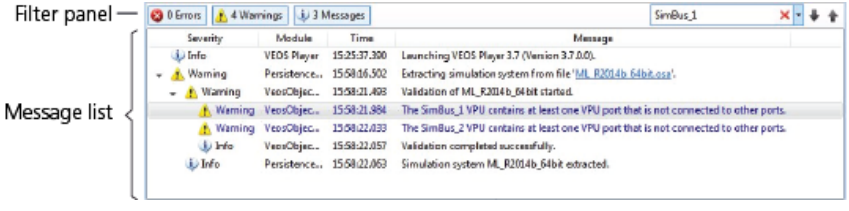
バージョン情報を取得するには、`IApplication` インターフェースの `Version` プロパティを使用しないことをお勧めします。

「Automating the Start of the VEOS Player, and Getting Version Information」(🕮『VEOS Guide』)および「IVersionInformation」(🕮『VEOS Player API Reference』)を参照してください。

Message Viewer と dSPACE Log Viewer

Message Viewer　VEOS Player は新しい Message Viewer をサポートしています。Message Viewer を使用して、製品の使用中に発生した情報メッセージ、助言メッセージ、エラーメッセージ、警告メッセージ、質問のすべての履歴を表示することができます。システムの状態をチェックするのに役立ちます。

Message Viewer は、次のようになります。



「Message Viewer」(🕮『VEOS Player Reference』)を参照してください。

dSPACE Log Viewer　VEOS Player は新しい dSPACE Log Viewer をサポートしています。

dSPACE ログは、すべての dSPACE 製品および接続されたシステムによって発行されたエラーや警告、情報、質問、および助言を複数回のセッションにわたって記録したものです。

dSPACE ログを dSPACE サポートに転送するには、dSPACE Installation Manager を使用します。

「dSPACE Log Viewer」(🕮『VEOS Player Reference』)を参照してください。

VEOS 3.7 の互換性

本章の内容

互換性一覧

一般的な互換性　dSPACE では、同一の dSPACE Release のソフトウェ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。

CTLGZ の互換性　次の表は、VEOS 3.7 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 2016-B: ■ SystemDesk 4.7 ■ TargetLink 4.2	2.4.1
dSPACE Release 2016-A: ■ SystemDesk 4.6	2.4
dSPACE Release 2015-B: ■ SystemDesk 4.5 ■ TargetLink 4.1	2.3
dSPACE Release 2015-A: ■ SystemDesk 4.4	2.2
dSPACE Release 2014-B: ■ SystemDesk 4.3 ■ TargetLink 4.0	2.1

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 3.2 ■ TargetLink 3.5	1.0

SIC の互換性　VEOS 3.7 は、dSPACE Release 2016-B の Model Interface Package for Simulink 3.3 で作成した Simulink インプリメンテーションコンテナ(SIC)ファイル(SIC バージョン 1.2)と互換性があります。

BSC の互換性　VEOS 3.7 は、dSPACE Release 2016-B の Bus Manager で作成されたバスシミュレーションコンテナ(BSC)ファイル(BSC バージョン 1.1)と互換性があります。

FMU の互換性　VEOS では、FMI for Co-Simulation インターフェースのみサポートしており、FMI for Model Exchange インターフェースはサポートしていません。

VEOS の FMI サポートに関する詳細情報および最新の互換性情報については、

http://www.dspace.jp/go/FMI-Compatibility を参照してください。

OSA の互換性　次の表は、VEOS 3.7 とオフラインシミュレーションアプリケーション(OSA)ファイルの互換性を示しています。

注記

1 つの VEOS バージョンから別のバージョンに移行する場合、既存のモデルインプリメンテーションコンテナ(CTLGZ、SIC、BSC、FMU)ファイルのバイナリ OSA ファイルを再ビルドすることをお勧めします。これは、OSA ファイルの互換性の制約によるものです。

OSA ファイルを作成した製品	OSA のバージョン
dSPACE Release 2016-B	3.7
dSPACE Release 2016-A	3.6[1)]
dSPACE Release 2015-B 以前	3.5 以前 [2)]

1) VEOS 3.6 で作成または変更された OSA ファイルは、VEOS 3.7 にロードできますが、すべての VPU プロパティは読み取り専用で変更することができません。ポートとネットワーク接続は編集することができます。

2) VEOS 3.5 以前で作成または変更された OSA ファイルは、それらにバス通信エレメントが含まれていない場合は、VEOS 3.7 にロードしてシミュレートすることができます。VPU プロパティは読み取り専用で、変更することができません。ポートとネットワーク接続は編集することができます。

注記

VEOS 3.7 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。

Real-Time Testing の互換性　VEOS と ControlDesk で RTT を使用する場合は、シミュレーションシステムを実行中の VEOS シミュレータで使用する Real-Time Testing（RTT）バージョンと、ホスト PC でアクティブな RTT バージョンが同じである必要があります。

次の表に、VEOS シミュレータのバージョンと対応する RTT バージョンを示します。

VEOS シミュレータ	Real-Time Testing バージョン
VEOS 3.7 以降	Real-Time Testing Version 3.1
VEOS 3.6 以降	Real-Time Testing Version 3.0
VEOS 3.5 以降	Real-Time Testing Version 2.6
VEOS 3.4 以降	Real-Time Testing Version 2.5
VEOS 3.2 以降	Real-Time Testing Version 2.4
VEOS 3.2 以降	Real-Time Testing Version 2.3
VEOS 3.1 以降	Real-Time Testing Version 2.2
VEOS 3.0 以降	Real-Time Testing Version 2.0

ControlDesk 6.0 では、自動的に VEOS 3.7 以降の VEOS シミュレータを使用します。そのため、RTT 3.1 がホスト PC でアクティブな場合は、VEOS と ControlDesk で RTT を使用することができます。

VEOS 3.7 への移行

配列 VPU ポートの表示と接続性

VEOS 3.6 から VEOS 3.7 へのバージョンアップで、配列 VPU ポートの表示と接続性が変更されています。

VEOS 3.6 以前	VEOS 3.7 以降
表示	
配列 VPU ポートとその要素は、  アイコンとともに VEOS Player に表示されます。	配列 VPU ポート（およびスカラー VPU ポート）は、  アイコンとともに VEOS Player に表示されます。 配列 VPU ポートの個々の要素は、表示されません。代わりに、要素の数が配列 VPU ポートの[Data width]プロパティとして表示されます。 配列 VPU ポートの個々の要素の初期値を指定するには、ポートの[Initial value]プロパティ編集フィールドにコンマ区切りリストを入力します。

VEOS 3.6 以前	VEOS 3.7 以降
接続性	
■ 配列 VPU ポート全体を接続することができます。 ■ 配列 VPU ポートの要素を個別に接続することができます。	■ 配列 VPU ポート全体を接続することができます。 ■ 配列 VPU ポートの要素を個別に接続することはできません。[1]

[1] VEOS 3.7 では、VEOS 3.6 以前で最後に保存したオフラインシミュレーションアプリケーションを開く場合、配列 VPU ポートの個々の要素間の接続が維持されます。

VEOS Player 自動化の変更

VEOS 3.7 では、VEOS Player の自動化インターフェースにある以下の変更を考慮する必要があります。

VEOS 3.6 以前	VEOS 3.7 以降
VPU ポートへのアクセスは、IVpu インターフェースの VpuPorts プロパティを介して可能です。	IVpu インターフェースの VpuPorts プロパティにより、VPU のルート階層にある VPU ポートのみにアクセスすることができます。この方法では、それより深い階層にある VPU にアクセスすることはできません。 その代わり、以下のいずれかの方法を使用することができます。 ■ IVpuPorts および IVpuPortGroups インターフェースは、下位の IVpuPort および IVpuPortGroup インターフェースに戻る Item メソッドを提供します。 ■ 別の方法としては、IVpu および IVpuPortGroup インターフェースは、下位の汎用的な IVpuItem インターフェースに戻る Item メソッドを提供します。 IVpuItem インターフェースの VpuItemType プロパティを介して、VPU ポートまたは VPU ポートグループのいずれかのアイテムタイプを取得することができます。 この変更は、VEOS 3.7 でサポートされた VPU ポートの階層表示によるものです。 「IVpu」(📖『VEOS Player API Reference』)および「IVpuPortGroup」(📖『VEOS Player API Reference』)を参照してください。
IVpuPort インターフェースの SendsTo プロパティのタイプは、System.String[]を使用します。	IVpuPort インターフェースの SendsTo プロパティのタイプは、IVpuPorts を使用します。 この変更は、VEOS 3.7 でサポートされた VPU ポートの階層表示によるものです。
IVpuPort インターフェースの ReceivesFrom プロパティのタイプは、String を使用します。	IVpuPort インターフェースの ReceivesFrom プロパティのタイプは、IVpuPort を使用します。 この変更は、VEOS 3.7 でサポートされた VPU ポートの階層表示によるものです。

VEOS 3.6 以前	VEOS 3.7 以降
配列 VPU ポートの個々の要素へのアクセスは、IVpuPort インターフェースの DataElements プロパティを介して可能です。	配列 VPU ポートの個々の要素にアクセスすることはできません。その結果、IVpuPort インターフェースの DataElements プロパティは利用できなくなりました。代わりに、配列 VPU ポートの要素の数は、IVpuPort インターフェースの DataWidth プロパティを介して利用することができます。
バージョン情報を取得するには、IApplication インターフェースの Version プロパティを使用することができます。	バージョン情報を取得するには、IVersionInformation インターフェースを使用することができます。 **注記** IApplication インターフェースの Version プロパティは互換性の理由からまだサポートされています。ただし、今後廃止される予定です。

ASM モデルの移行

モデルが以前の dSPACE Release で作成された OSA または SIC ファイルに含まれる場合、ASM モデルを VEOS 3.7（dSPACE Release 2016-B）でシミュレートすることはできません。

VEOS 3.7 では、VEOS 3.7 で Release 2016-B より前の dSPACE Release で保存された ASM モデルをシミュレートするには、次の手順を実行します。

1. ASM モデルを dSPACE Release 2016-B に移行します。

   ASM モデルの移行については、「ASM モデルの移行」(🕮『ASM ユーザガイド』)を参照してください。

2. *Model Interface Package for Simulink* を使用して、ASM モデルに基づく Simulink インプリメンテーションコンテナ（SIC）ファイルを生成します。

   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 3.7 の VEOS Player にインポートします。

   詳細については、「How to Import Simulink Implementations」(🕮『VEOS Guide』)を参照してください。

VEOS での廃止

dSPACE Release 2017-A 以降の VEOS での廃止

VEOS および SystemDesk での PIL（Processor-in-the-Loop）シミュレーションの廃止　dSPACE Release 2017-A 以降では、VEOS および SystemDesk で PIL（Processor-in-the-Loop）シミュレーションがサポートされなくなります。これには、PIL シミュレーション向けの V-ECU や、評価ボードでのこれらの V-ECU のシミュレーションも含まれます。

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

MATLAB®

各種 dSPACE 製品を使用するには、MATLAB をインストールしておく必要があります。

ヒント

MathWorks®社製ソフトウェ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。

MATLAB のリリース	dSPACE の各コンポーネントによるサポート Release 2016-B					
	RCP and HIL Software	AutomationDesk 5.3 [1]	TargetLink 4.2	Model Compare 2.7	dSPACE Python Extensions 2.2 [2]	XIL API .NET MAPort 2016-B
R2016b（64 ビット）	✓ [3]	✓	✓	✓	✓	✓
R2016a（64 ビット）	✓ [3]	✓	✓	✓	✓	✓
R2015b（64 ビット）	✓	✓	✓	✓	✓	✓
R2015a SP1（64 ビット）	✓	✓	✓	✓	✓	✓

[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] R2016a と R2016b は、RTI FPGA Programming Blockset - FPGA Interface ではサポートされません。

注記

dSPACE Release 2016-A 以降、dSPACE ソフトウェアは 64 ビット MATLAB バージョンのみをサポートします。MATLAB の 32-bit バージョンはサポートされません。

dSPACE ソフトウェ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/go/MATLABCompatibility を参照してください。

RCP and HIL Software：MEX ファイルのビルド用 C コンパイラ RCP and HIL Software では、MEX 関数のビルドには Microsoft Windows SDK 7.1 のみ使用することができます。

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 も必要です。こちらも Microsoft 社から無料で提供されています。サポートされているコンパイラについては、http://www.mathworks.com/support/compilers/R2016a/index.html を参照してください。

RTI CAN MultiMessage Blockset、RTI LIN MultiMessage Blockset、または Automotive Simulation Models など MEX コンパイラが必要な 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 MATLAB で MEX コンパイラとして設定する必要があります。

MAT ファイルのサポートの制限　ControlDesk（ControlDesk 6.0）の Signal Editor 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。

ModelDesk の制限事項　Simulink シミュレーションに MATLAB R2016a を使用し、シミュレーションを実行する場合は、ダウンロードに少なくとも 1 分必要です。

シミュレーションを停止または一時停止する場合の制限事項はありません。

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

dSPACE Release 2016-B の dSPACE 製品は、以下のオペレーティングシステムをサポートしています。

- 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。

- dSPACE 製品の Windows 10 のサポート:

  dSPACE Release 2016-B では、TargetLink と Model Compare のみが、以下のエディションおよびサービスオプションの Windows 10 をサポートします。

  - Windows 10 Professional、Education、Enterprise（64 ビット版）

    Windows 10 Home および Mobile エディションはサポートされません。

  - Long-Term Servicing Branch：LTSB 2016
  - Current Branch for Business：CBB 1511

  dSPACE Release 2016-B 全体での Windows 10 のサポートは、2017 年半ばに利用可能になるサービスパックによって受けられるようになります。dSPACE Release 2017-A 以降では、dSPACE 製品全体で Windows 10 をサポートする予定です。

Windows と dSPACE ソフトウェ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。「Windows 機能の使用に関する制限事項」(279 ページ)を参照してください。

注記

dSPACE Release 2016-A 以降、dSPACE ソフトウェアは 64 ビット版のオペレーティングシステムのみをサポートします。32 ビット版のオペレーティングシステムはサポートされなくなります。

MicroAutobox Embedded PC をホスト PC として使用

ControlDesk は、Windows 7 Professional、Ultimate、および Enterprise (64 ビット版) が稼働している MicroAutoBox Embedded PC (Intel® Core™ i7-3517UE Processor 搭載) にインストールすることもできます。

ファイアウォールルールを追加して通信を許可

各種 dSPACE ソフトウェ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。その 1 つは、AutoBox などの dSPACE 拡張ボックスとの通信を許可するためのルールです。もう 1 つは、MotionDesk によるネットワークチャンネルからのモーションデータの受信を許可するためのルールです。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ェアを実行している場合は、dSPACE ソフトウェアの 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 Vista Business、Ultimate、または Enterprise（64 ビット版）の最新のサービスパック
- Windows 7 Professional、Ultimate、または Enterprise（64 ビット版）の最新のサービスパック
- Windows Server 2008 R2
- Windows Server 2012、Windows Server 2012 R2

注記

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 製品の 64 ビットの互換性に関する注意事項

注

dSPACE Release 2016-B 以降、すべての製品は 64 ビットバージョンのみをサポートします。その結果、dSPACE Release 2016-B 以降では、以下のソフトウェアの 64 ビットバージョンのみがサポートされます。

- Windows オペレーティングシステム
- MATLAB
- Python

一般的には、64 ビットテクノロジによって 32 ビットテクノロジよりも複雑なモデルやレイアウト、および大量のデータを処理することができます。

ただし、以下の点に注意してください。

- ConfigurationDesk カスタム I/O ファンクションブロックなどの製品拡張が 64 ビットで利用可能である必要があります。
- Python:
  - 32 ビットのサードパーティ製拡張は、dSPACE が提供する 64 ビット Python インストレーションではサポートされません。
  - 64 ビットの dSPACE Python Extensions は、32 ビット Python インストレーションでの並行使用はサポートされません。
- 一部の製品では移行作業が必要となります。詳細については、『*新機能と移行手順*』のそれぞれの製品に関する章や、該当する dSPACE 製品のユーザマニュアルを参照してください。

dSPACE ソフトウェ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

製品互換性:dSPACE Release 2016-B

dSPACE では、同一の dSPACE Release のソフトウェ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 2016-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。

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。

DS1005、DS1006、DS1103、DS1104 または MicroAutoBox プラットフォームにロードされたリアルタイムアプリケーションの互換性　リアルタイムアプリケーションが dSPACE Release 2016-B のソフトウェア製品を使用して関連するプラットフォームのいずれかにロードされた場合、dSPACE Release 2016-A のソフトウェア製品は、ロードされたリアルタイムアプリケーションがホスト PC に保存されたリアルタイムアプリケーションと同一であることを検出しません。

	これと同じことが、dSPACE Release 2016-A のソフトウェア製品を使用してリアルタイムアプリケーションをロードし、dSPACE Release 2016-B のソフトウェア製品を使用して試験などを実行する場合にも適用されます。
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 機能の使用に関する制限事項

目的	dSPACE ソフトウェアを Windows の機能と組み合わせて使用する場合には、いくつかの制限事項が適用されます。
ユーザの簡易切り替えのサポートなし	dSPACE ソフトウェアは、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 ソフトウェアを閉じる	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 ソフトウェ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 ソフトウェ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 ソフトウェ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ェ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。
長いパス名	dSPACE ソフトウェアは、Windows API の長いパス名構文をサポートしていません。260 文字を超えるパス名が直接または間接的に使用された場合の dSPACE ソフトウェアの動作は未確認です。

Windows の 8dot3name 作成オプションの有効化

注記

サードパーティ製ソフトウェア（MATLAB®/Simulink®など）および dSPACE ソフトウェアをインストールする前に、Windows の [8dot3name の作成]オプションをすべてのドライブ（インストールに使用するドライブと作業に使用するドライブ）で有効にすることを強くお勧めします。

ソフトウェアのインストール時にこのオプションが無効な場合、dSPACE ソフトウェアの実行時に、ビルドプロセスの中断など重大なエラーが発生する可能性があります。[8dot3name の作成]オプションが無効な状態で行われたインストールを修復するには、dSPACE ソフトウェアと必要なサードパーティ製ソフトウェアを再インストールする必要があります。dSPACE Maintenance Setup を使用してもこの問題は解決しません。

設定の確認とオプションの有効化の手順については、http://www.dspace.com/faq?346 または Microsoft Windows のマニュアルを参照してください。