LabVIEW™ 2018アップグレードノート

このアップグレードノートは、Windows、macOS、およびLinux用のLabVIEWをLabVIEW 2018にアップグレードする方法を説明します。アップグレードを行う前に、以下のトピックの情報について、このドキュメントを参照してください。

- LabVIEWをアップグレードする際の推奨プロセス
- LabVIEWの旧バージョンで保存されたVIをロードする前に考慮するべき互換性の問題
- LabVIEW 2018に新たに追加された機能および動作

目次

LabVIEW 2018へのアップグレード

小さなアプリケーションは、LabVIEWの新しいバージョンをインストールしてからVIをロードすることにより新しいバージョンにアップグレードできますが、NIでは、アップグレード時の問題をできるだけ効率的に検出して修正できるよう、より綿密なアップグレードプロセスを推奨しています。

ヒント このプロセスは、重要な操作を制御または監視する、長時間のダウンタイムが許されない、複数のモジュール、ツールキット、ドライバを使用する、またはサポートされないバージョンのLabVIEWで保存された大規模なLabVIEWアプリケーションで特に効果があります。現在主要サポートの対象となっているLabVIEWのバージョ

ンについては、NIウェブサイト(ni.com/jp/info)でInfo Codeに「lifecyclejp」と入力してください。

推奨アップグレードプロセスの概要

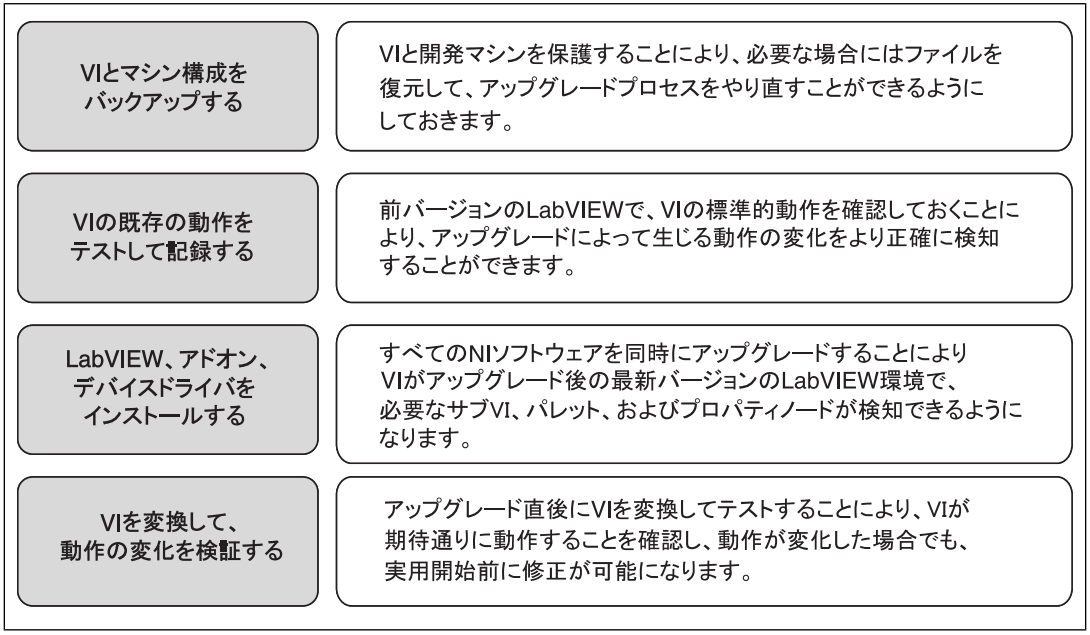

図 1

メモ LabVIEW 5.1以前からアップグレードするには、まずLabVIEWの中間バージョンにアップグレードする必要があります。LabVIEWの特定のレガシーバージョンからのアップグレードについては、NIウェブサイト(ni.com/jp/info)でInfo Codeに「upgradeOldja」と入力してください。

1.VIおよびマシン構成のバックアップを作成する

必要に応じてVIを以前の機能に復元してアップグレードプロセスを再度実行できるように、LabVIEW 2018にアップグレードする前に、VIのコピーおよび開発マシンまたは本稼動マシンの構成 (可能な場合) を保護しておきます。

a. VIのバックアップを作成する

LabVIEWをアップグレードする前にVIをバックアップしておくと、バックアップコピーの状態にすぐ戻すことができます。バックアップコピーがない場合、各VIを旧バージョン用に保存しないと、アップグレードしたVIを旧バージョンのLabVIEWで開けなくなります。

VIセットのバックアップは、以下のいずれかの方法で作成できます。

- **VIをソースコード管理プロバイダにサブミットする**—これにより、VIのアップグレードによる動作の変化に対応できない場合に、VIをこのバージョンに戻すことができます。LabVIEWでソースコード管理プロバイダを使用する方法については、『LabVIEWヘルプ』の**目次**タブで**基本機能→プロジェクトとターゲットを操作する→概念→LabVIEWでソース管理を使用する**トピックを参照してください。

- **VIのコピーを作成する**—VIの構成方法に従って、VIのコピーを作成します。
  - プロジェクトとして保存されている—プロジェクトを開き、**別名で保存**を選択して、`.lvproj`ファイルおよびすべてのプロジェクト内容を複製します。プロジェクトが依存するファイルのコピーも維持するために、**すべての依存項目を含む**を選択してください。
  - LLBまたはディレクトリ内のVIとして保存されている—オペレーティングシステムのファイルエクプローラでLLBまたはディレクトリのコピーを作成し、元の場所とは別の場所に保管します。名前が競合する可能性を回避するために、コピーを同じハードドライブに保管しないようにしてください。

b. マシン構成のバックアップを作成する

新しいバージョンのLabVIEWをインストールすると共有ファイルが更新されるため、旧バージョンのVIの動作も影響を受けることがあります。しかし、それらの共有ファイルを更新後に前のバージョンに戻すことは非常に困難です。このため、サポートされていないバージョンのLabVIEWからアップグレードする場合、またはアプリケーションのダウンタイムコストが高い場合には特に、以下のいずれかの方法で、NIソフトウェアの構成をバックアップすることを検討してください。

- **マシン構成のバックアップイメージを作成する**—アップグレードする前に、インストールされているソフトウェア、ユーザ設定、ファイルなどのマシンのディスク状態を、ディスクイメージングソフトウェアを使用して保存します。アップグレードした後でマシンを元の構成に戻すには、バックアップしたディスクイメージをデプロイします。
- **テストマシンでアップグレードプロセスをテストする**—テストマシンでアップグレードを行うとバックアップイメージを作成するよりも時間はかかりますが、NIでは、実際の工程を制御または監視するマシンのダウンタイムを回避または最小化する必要がある場合にはこのアプローチを強く推奨しています。アップグレードに起因するすべての問題をテストマシンで解決した後、テストマシンを本稼動マシンに置き換えるか、本稼動マシンでこのアップグレードプロセスを再実行します。

  ヒント アップグレードしたVIがテストマシンと開発マシンで異なる動作をする可能性を最小化するため、開発マシンのCPU、RAM、オペレーティングシステム、ソフトウェアのバージョンなどの装備とできるだけ一致したテストマシンを使用してください。

2.VIの既存の動作をテストして記録する

VIをアップグレードすると、LabVIEWの旧バージョンとLabVIEW 2018の間の違いによってVIの動作が変化する場合があります。両方のバージョンでVIをテストしてその結果を比較することで、アップグレードに起因する動作の変化を検知できます。このため、以下のいずれかのテストの現在の結果が手元にあることを確認してください。

- 一括コンパイルログ—旧バージョンのLabVIEWのVIを一括コンパイルすると、壊れているVIの詳細なログが生成されます。この情報は、複数の人がそのVIの開発に関わっていたり、それらのVIの中に最近コンパイルされていないVIがある可能性がある場合に特に有益です。この一括コンパイルログを生成するには、**一括コンパイル**ダイアログボックスで**結果をログ**チェックボックスをオンにします。VIの一括コンパイルの詳細については、『LabVIEWヘルプ』の**目次**タブで**基本機能→VIおよびサブVIを作成する→操作手順→VIを保存する→VIを一括コンパイルする**トピックを参照してください。
- 各VIが意図した機能を正しく実行しているかどうかを検証するユニットテスト
- プロジェクトまたはサブVIのグループがまとまって予期した動作をしているかどうかを検証する統合テスト

- VIがデスクトップやFPGAターゲットなどのターゲットにデプロイされたときに予想どおりに動作するかどうかを検証するデプロイメントテスト
- CPU使用率、メモリ使用、コード実行速度のベンチマーク測定する性能テスト。**パフォーマンスおよびメモリをプロファイル**ウィンドウを使用すると、VIの平均実行速度を計算できます。
- VIが予期しないデータを正しく処理できるかどうかを検証するストレステスト

VIテストの詳細については、『LabVIEWヘルプ』の**目次**タブの**基本機能→アプリケーション開発と設計ガイドライン→概念→大規模アプリケーションを開発する→開発モデルの段階→アプリケーションをテストする**トピックを参照してください。

メモ　テスト結果をもとにVIを変更した場合は、この先に進む前に新しいバージョンのVIをバックアップします。

3.LabVIEW、アドオン、およびデバイスドライバをインストールする

a. LabVIEW (モジュール、ツールキット、およびドライバを含む) をインストールする

新しいバージョンのLabVIEWにアップグレードした場合、新しい開発システムだけでなく、その新しいバージョンと互換性のあるモジュール、ツールキット、およびドライバをインストールする必要があります。

b. user.libファイルをコピーする

旧バージョンのLabVIEWで作成したカスタム制御器とVIをLabVIEW 2018で使用するには、旧バージョンのLabVIEWの`labview¥user.lib`ディレクトリのファイルをLabVIEW 2018の`labview¥user.lib`ディレクトリにコピーします。

4.VIを変換し、動作の変化を検証する

LabVIEW 2018でVIを一括コンパイルすると、VIはLabVIEWの最新バージョンに変換され、壊れたVIの特定に役立つエラーログが作成されます。この情報を、このドキュメントの「アップグレードおよび互換性の問題」セクションと併せて使用することにより、LabVIEWの最新バージョンに関連する動作変化を特定して修正できます。

メモ　LabVIEW 2017にアップグレードしたときにVIを一括コンパイルする場合、LabVIEW 2017以降ではLabVIEWランタイムエンジンの下位互換性がサポートされているため、この手順を省略できます。

a. VIを新しいバージョンのLabVIEWで一括コンパイルする

VIを一括コンパイルすると、VIはLabVIEW 2018に変換されて保存されます。しかし、各VIまたはプロジェクトに対して**ファイル→旧バージョン用に保存**を選択しない限り、VIを旧バージョンのLabVIEWで開けなくなります。このため、新しいバージョンのLabVIEWに変換したいVIのみを一括コンパイルしてください。アップグレードにより発生した問題を特定するには、**一括コンパイル**ダイアログボックスの**結果をログ**チェックボックスをオンにして一括コンパイルログを作成します。

メモ　FPGAまたはリアルタイムリソースが含まれているVIを一括コンパイルすると、**一括コンパイル**ダイアログボックスに、これらのVIは実行不可能なVIであるとレポートされる場合があります。エラーを確認するには、FPGAまたはRTターゲットの下にあるVIを、プロジェクト内で必要なFPGAリソースまたはリアルタイムリソースとともに開きます。

VIの一括コンパイルの詳細については、『LabVIEWヘルプ』の**目次**タブで以下のトピックを参照してください。

- **基本機能→VIおよびサブVIを作成する→操作手順→VIを保存する→VIを一括コンパイルする**
- **基本機能→VIおよびサブVIを作成する→操作手順→VIを保存する→一括コンパイルの一般的なステータスメッセージ**

b. 壊れたVIを修正する

旧バージョンのLabVIEWとLabVIEW 2018との違いにより、変更された機能が使用されているVIが壊れる場合があります。壊れたVIをLabVIEW 2018で特定して、修正するには、以下の手順を実行してください。

1. アップグレード中に壊れたVIを特定するには、前のステップで作成した一括コンパイルエラーログと、VIの既存の動作を確認するために作成したログを比較します。
2. 各VIがLabVIEWのどのアップデートが原因で壊れたかを調べるには、このドキュメントの「アップグレードおよび互換性の問題」セクションを参照してください。

c.動作の変化を特定して修正する

NIでは、LabVIEWのバージョン間におけるVIの動作の違いを最小限に抑えるために多大な努力を払っていますが、VIの改善やバグ修正の結果、VIの動作が変化することがあります。LabVIEWの新しいバージョンでVIの動作が変更するかどうかを素早く確認するには、以下のツールを使用します。

- **VI Analyzer Upgrade Testを実行**—大規模なVIセットの場合、これらのテストにより、アップグレードに起因する多くの動作変化を効率的に特定できます。これらのテストを取得して使用するには、以下の手順に従います。
  1. 使用している旧バージョン以降のLabVIEWのすべてのバージョンのVI Analyzer Upgrade Testをダウンロードします。これらのテストをダウンロードするには、NIウェブサイト (`ni.com/jp/info`) でInfo Codeに「`analyzevi`」を入力してください。
  2. **ツール→VIアナライザ→VIを解析**を選択して、新規のVIアナライザタスクを開始することにより、テストを開いて、実行します。プロジェクト全体をまとめて解析するには、個々のVIからではなく、**プロジェクトエクスプローラ**ウィンドウでこのメニュー項目を選択します。
  3. テストに失敗した場合は、テストに対応するLabVIEWのバージョンの「アップグレードおよび互換性の問題」セクションを参照して修正します。たとえば、LabVIEW 2015 VI Analyzer Upgrade testが動作変更の可能性を検出した場合は、そのトピックの「LabVIEW 2014からアップグレードする」セクションを参照します。
- **アップグレードドキュメントを読む**
  - このドキュメントのアップグレードおよび互換性の問題トピック—VIを壊したり、VIの動作に影響を与える可能性のある変更が記載されています。前のバージョンから始めて、LabVIEWの各バージョンのサブセクションを参照します。

    ヒント 「アップグレードおよび互換性の問題」セクションで言及されている廃止されたオブジェクトやその他のオブジェクトを手早く見つけるには、アップグレードしたVIを開き、**編集→検索して置換**を選択します。

  - LabVIEW 2018既知の問題リスト—LabVIEW 2018のリリース前およびリリース中に発見されたバグが記載されています。このリストを参照するには、NIウェブサイト (`ni.com/jp/info`) でInfo Codeに「`lv2018ki`」を入力してください。「アップグレード—動作の変化」と「アップグレード—移行」セクションがある場合は、アッ

プグレードしたVIの動作に影響するバグの回避方法についてそれらを参照してください。

- モジュールおよびツールキットのドキュメント—LabVIEW FPGAおよびLabVIEW Real-Timeモジュールなど、一部のモジュールおよびツールキットに特有のアップグレード問題が記載されています。
- ドライバのReadmeファイル—各ドライバに特有のアップグレード問題が記載されています。各Readmeを見つけるには、そのドライバのインストールメディアを参照してください。

ヒント 動作が変化した原因がLabVIEWのアップデートではなく、ドライバのアップデートであることを確認するには、LabVIEW 2018をインストールした後に旧バージョンのLabVIEWでそのVIをテストしてください。

- **ユーザ独自のテストを実行する**—旧バージョンのLabVIEWで行ったものと同じテストをLabVIEW 2018のVIに対して行い、結果を比較します。新しい動作を特定した場合は、その変化の原因を診断するために、アップグレードドキュメントを参照します。

アップグレードに関する一般的な問題のトラブルシューティング

アップグレードに関する以下の問題の解決方法については、『LabVIEWヘルプ』の**目次**の**LabVIEW 2018へのアップグレード→アップグレードに関する一般的な問題のトラブルシューティング**トピックを参照してください。

- 見つからないモジュールまたはツールキットの機能を探す
- 見つからないサブVI、パレット、プロパティノードを探す
- 旧バージョンのLabVIEWで作成したVIがLabVIEW 2018で開けない理由を特定する
- インストールされているNIソフトウェアのバージョンを特定する
- VIを旧バージョンのLabVIEWに復元する

アップグレードおよび互換性の問題

VIを壊したり、動作を変更したりする可能性のある、LabVIEWのバージョンごとの変更については、以下のセクションを参照してください。

新しいバージョンのLabVIEWの既知の問題、その他の互換性に関する問題、LabVIEW 2018の最新の追加機能の詳細は、`labview`ディレクトリの`readme.html`を参照してください。

LabVIEW 2013以前からアップグレードする

LabVIEW 2013以前からLabVIEW 2018へのアップグレード時に発生する可能性があるアップグレードまたは互換性の問題に関する情報を参照するには、NIウェブサイト(`ni.com/jp/info`)で「`upnote14jp`」と入力してください。また、その他のアップグレードに関する問題については、このドキュメントの「LabVIEW *x*からアップグレードする」セクションを参照してください。

LabVIEW 2014からアップグレードする

LabVIEW 2014からLabVIEW 2018へアップグレードすると、以下の互換性問題が発生する可能性があります。発生する可能性のあるその他のアップグレード関連の問題については、このドキュメントの「LabVIEW 2015からアップグレードする」、「LabVIEW 2016からアップグレードする」、および「LabVIEW 2017からアップグレードする」セクションを参照してください。

LabVIEWアプリケーションにおけるバッファ割り当てを識別する

LabVIEW 2014 Service Pack 1以降には、LabVIEWアプリケーションにおけるバッファ割り当てを特定および解析するための**バッファ割り当てをプロファイル**ウィンドウが追加されました。このウィンドウを表示するには、**ツール→プロファイル→バッファ割り当てをプロファイル**を選択します。

フリーラベル内のハイパーリンク

LabVIEW 2015以降では、LabVIEWがフリーラベル内のURLを検出し、下線付きの青いテキストのハイパーリンクに変換します。LabVIEW 2014以前からアップグレードした場合、フリーラベル内のURLは自動的にはハイパーリンクに変換されません。フロントパネルのラベルのハイパーリンクを有効にするには、フリーラベルを右クリックし、ショートカットメニューで**ハイパーリンクを有効化**を選択します。ブロックダイアグラムのラベルでハイパーリンクを無効にすることはできません。

廃止されたVI、関数、およびノード

LabVIEW 2015以降は、以下のVI、関数、およびノードをサポートしていません。

- 「**スプレッドシートファイルから読み取る**」—代わりに「区切られたスプレッドシートを読み取る」VIを使用します。
- 「**スプレッドシートファイルに書き込む**」—代わりに「区切られたスプレッドシートに書き込む」VIを使用します。

LabVIEW 2015からアップグレードする

LabVIEW 2015からLabVIEW 2018へアップグレードすると、以下の互換性問題が発生する可能性があります。発生する可能性のあるその他のアップグレード関連の問題については、このドキュメントの「LabVIEW 2016からアップグレードする」および「LabVIEW 2017からアップグレードする」セクションを参照してください。

LabVIEW 2016以降では、**クイックドロップ構成**ダイアログボックスに、フロントパネルオブジェクトとブロックダイアグラムオブジェクトのショートカットのデフォルトリストが含まれています。LabVIEW 2015以前で作成したショートカットは、LabVIEW 2016以降のショートカットリストに自動的に統合されません。

LabVIEW 2016からアップグレードする

LabVIEW 2016からLabVIEW 2018へアップグレードすると、以下の互換性問題が発生する可能性があります。発生する可能性のあるその他のアップグレード関連の問題については、このドキュメントの「LabVIEW 2017からアップグレードする」セクションを参照してください。

アクターフレームワークVIの動作の変更

LabVIEW 2016以前では、ネストされたアクターは、起動前初期化メソッドのエラーが原因で起動できないと、エラーを返し、そのエラーを含んだ最終Ackメッセージを発呼者アクターに送信します。LabVIEW 2017では、ネストされたアクターは、最終Ackメッセージを送信せずにエラーを返します。

LabVIEW 2017からアップグレードする

LabVIEW 2017からLabVIEW 2018へアップグレードすると、以下の互換性問題が発生する可能性があります。

LabVIEWランタイムエンジンの下位互換性

LabVIEW 2017以降では、LabVIEWランタイムエンジンの下位互換性がサポートされています。旧バージョンのLabVIEWで作成されたバイナリおよびVIを、バイナリを再度ビルドしたり、VIを現在のバージョンのLabVIEWで一括コンパイルしたりせずにロードおよび実行できます。

たとえば、LabVIEW 2017以降のバージョンでは、LabVIEW 2017でビルドされたバイナリおよびVIを再コンパイルせずにロードできます。この改善の対象となるのは、スタンドアロンアプリケーション (EXE)、共有ライブラリ (DLL)、およびパックプロジェクトライブラリです。

バイナリで下位互換性を有効にするには、ビルド仕様に基づいて、特定のダイアログボックスの**上級**ページで以下のチェックボックスをオンにします。

ビルド仕様	ダイアログボックス	チェックボックス
スタンドアロンアプリケーション (EXE)	**アプリケーションプロパティ**	**今後のLabVIEWランタイムバージョンでこのアプリケーションの実行を許可する**
パックプロジェクトライブラリ	**パックライブラリプロパティ**	**今後のLabVIEWバージョンでこのパックライブラリのロードを許可する**
共有ライブラリ (DLL)	**共有ライブラリプロパティ**	**今後のLabVIEWバージョンでこの共有ライブラリのロードを許可する**

LabVIEW 2017以降で作成したビルド仕様では、これらのオプションはデフォルトで有効になっています。ビルド仕様を特定のバージョンのLabVIEWに関連付ける場合は、これらのオプションを無効にします。これらのオプションを無効にすると、パフォーマンスプロファイルへの変更を防ぐことができる上、コンパイラのアップグレードによる予期しない問題の発生を回避できます。リアルタイムアプリケーションでは、ダイアログボックスにこれらのオプションは表示されませんが、この機能はデフォルトで有効になっています。

レポート生成VIの動作の変更

LabVIEW 2018では、「レポート生成」VIは、標準レポート形式のレポートの生成をサポートしなくなりました。レポートは、HTML、Word、またはExcel形式でのみ生成できます。この動作変更により、次のVIは廃止予定となりました。

- **簡易VIパネルまたはドキュメントを印刷**—このVIは廃止予定となりました。代わりに、「**VIパネルまたはドキュメントを印刷** 」VIを使用してください。
- **簡易テキストレポート**—このVIは廃止予定となりました。代わりに、「**簡易テキストレポートを作成**」VIを使用してください。
- **レポートタイプ取得**—このVIは廃止予定となりました。代わりに、「**レポートタイプ**」VIを使用してください。
- **新規レポート**—このVIは廃止予定となりました。代わりに、「**レポートを作成**」VIを使用してください。
- **レポートタブ幅設定**—このVIは廃止予定となりました。

廃止予定のVI、関数、およびノード

LabVIEW 2018以降では、「**数値を列挙体に変換**」VIはサポートされません。「タイプに強制変換」関数を使用してください。

LabVIEW 2018の機能および変更点

意見交換アイコンは、ni.comのNI Idea Exchangeディスカッションフォーラム (英語) での製品に関する提案を元に開発された新機能を示しています。NI Idea Exchangeディスカッションフォーラムにアクセスするには、NIウェブサイト`ni.com/jp/info`でInfo Codeに「`ex3gusja`」と入力してください。

既知の問題、修正されたバグの部分的なリスト、その他の互換性に関する問題、LabVIEW 2018における追加機能については、`labview`ディレクトリの`readme.html`ファイルを参照してください。

順応性VIを異なるデータタイプ用にカスタマイズする

「比較」パレットに、「タイプをアサート」サブパレットが新たに追加されました。タイプをアサートVIおよび関数は、順応性VI (.vim) が特定の要件を満たすデータタイプのみを受け入れるように強制します。順応性VIのコードの一部を特定のデータタイプ用にカスタマイズするには、タイプ特化ストラクチャを使用します。

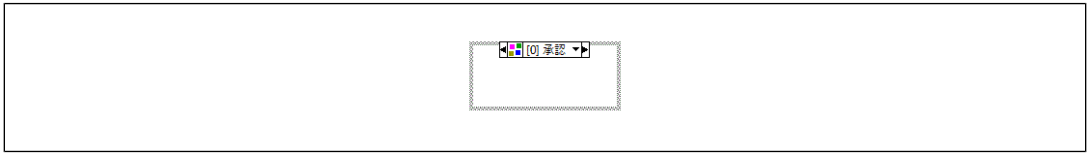

図 2

タイプ特化ストラクチャを使用して順応性VIのコードの一部をカスタマイズする方法については、`labview¥examples¥Malleable VIs¥Type Specialization Structure¥Malleable VIs - Type Specialization Structure.lvproj`を参照してください。

LabVIEW用コマンドラインインターフェイスを使用して操作を実行する

LabVIEW 2018では、LabVIEW用コマンドラインインタフェース (CLI) を使用してコマンドを実行することにより、LabVIEWで操作を実行できます。たとえば、LabVIEW用CLIは、LabVIEWアプリケーションのビルドプロセスの自動化に使用できます。LabVIEW用CLIは、以下の操作をサポートしています。

- MassCompile—指定されたディレクトリのファイルを一括コンパイルします。
- ExecuteBuildSpec—ビルド仕様の設定を使用してアプリケーション、ライブラリ、またはビットファイルをビルドし、出力ファイルのパスを返します。
- RunVI—定義済みのコネクタペーンインタフェースでVIを実行し、出力またはエラー情報を返します。
- CloseLabVIEW—ユーザに通知せずにLabVIEWを閉じます。
- **(VI Analyzerツールキット)** RunVIAnalyzer—LabVIEW VI Analyzerツールキットで指定されたVI Analyzerタスクを実行し、指定された場所にテストレポートを保存します。
- **(Unit Frameworkツールキット)** RunUnitTests—LabVIEW Unit Test Frameworkツールキットで指定されたファイルに対してテストを実行し、指定された場所にJUnitファイルを保存します。

メモ LabVIEWでこの操作を実行するには、JKI VI Package Manager (VIPM) ソフトウェアを使用してUTF Junit Reportライブラリをインストールする必要があります。VIPMソフトウェアの使用方法については、『LabVIEWヘルプ』の**目次**タブにある**ツールキット→VI Package Manager (VIPM) ソフトウェアを使用してLabVIEWアドオンにアクセスする**を参照してください。

LabVIEWで実行するカスタム操作を作成することもできます。

LabVIEW CLIの詳細については、『LabVIEWヘルプ』の**目次**タブで、**基本機能→LabVIEW用コマンドラインインターフェイスを使用して操作を実行する**ブックを参照してください。

LabVIEWからPythonコードを呼び出す

「接続」パレットに、LabVIEWコードからPythonコードを呼び出すために使用できる「Python」サブパレットが新たに追加されました。「Python」パレットには次の関数が含まれています。

- Pythonセッションを開く—特定のバージョンのPythonでPythonセッションを開きます。

- Pythonノード—Python関数を直接呼び出します。
- Pythonセッションを閉じる—Pythonセッションを閉じます。

メモ LabVIEW Python関数を使用するには、 Python 2.7または3.6をインストールする必要があります。サポートされていないバージョンでもLabVIEW Python関数が動作する場合もありますが、NIでは、サポートされているバージョンのPythonのみを使用することを推奨しています。Pythonのインストールの詳細については、`ni.com/jp/info`でInfo Codeに「`python`」と入力してください。

アプリケーションビルダの改善点

LabVIEW 2018では、LabVIEWアプリケーションビルダおよびビルド仕様が以下のように改善されました。

WindowsおよびLinuxリアルタイムターゲットでパッケージを作成する

LabVIEWでパッケージを作成し、NI パッケージマネージャまたはSystemLinkを使用してクライアントにデプロイできます。パッケージマネージャおよびSystemLinkでパッケージを使用すると、ソース配布、パックプロジェクトライブラリ、共有ライブラリ、.NETアセンブリ、および実行ファイルなど、すべてのタイプのファイルを配布できます。

(Windows 64ビット) プロジェクトエクスプローラウィンドウで**ビルド仕様**を右クリックし、**新規→Package**を選択してNIパッケージ(`.nipkg`)を作成します。クライアントはパッケージマネージャまたはSystemLinkを使用してフィードをサブスクライブし、パッケージを検索してインストールできます。

(NI Linux Real-Time) LabVIEW Real-Timeモジュールをインストールすると、NI Linux Real-Timeターゲットでもopkgパッケージ(`.ipk`)を作成できます。クライアントは、SystemLinkまたはNI Linux Real-Timeターゲットのコマンドラインからパッケージをインストールできます。パッケージマネージャは、`.ipk`ファイルをサポートしていません。

パッケージの作成の詳細については、『LabVIEWヘルプ』の**目次**タブで、**基本機能→アプリケーションの作成および配布→ビルド仕様を作成する→配布用のパッケージを作成する**を参照してください。

LabVIEWで作成された.NETアセンブリの下位互換性サポート

下位互換性のサポートにより、.NET interopアセンブリは、作成に使用されたLabVIEWバージョンまたはマシンにインストールされている最新バージョンのLabVIEWランタイムエンジンのいずれかにロードできます。たとえば、LabVIEW 2018で作成された.NET interopアセンブリは、再コンパイルしなくても、2018以降のバージョンのLabVIEWランタイムエンジンにロードして実行できます。

.NETアセンブリの下位互換性サポートを有効にするには、**.NET Interopアセンブリプロパティ**ダイアログボックスの**上級**ページで、**今後のLabVIEWバージョンでこの.NETアセンブリのロードを許可する**チェックボックスをオンにします。

LabVIEW 2018以降で作成したビルド仕様では、このオプションがデフォルトで有効になっています。ビルド仕様を特定のバージョンのLabVIEWに関連付ける場合は、このオプションを無効にします。このオプションを無効にすると、パフォーマンスプロファイルへの変更を防ぐことができる上、コンパイラのアップグレードによる予期しない問題の発生を回避できます。リアルタイムアプリケーションでは、ダイアログボックスにこのオプションは表示されませんが、この機能はデフォルトで有効になっています。

環境の改善点

LabVIEW 2018では、LabVIEW環境が以下のように改善されました。

タイプ定義の作成における改善点

LabVIEW 2018では、カスタム制御器または表示器のすべてのインスタンスを、保存済みのカスタム制御器ファイルまたは表示器ファイルにリンクさせるタイプ定義を作成する新しい方法が追加されました。新規タイプ定義は、以下の方法のいずれかで作成できます。

- **ファイル→新規**を選択し、**他のファイル**から**タイプ定義**を選択します。
- **プロジェクトエクスプローラ**ウィンドウの**マイ コンピュータ**を右クリックし、ショートカットメニューから**新規→タイプ定義**を選択します。

(NIディスカッションフォーラムメンバーMathis_Bの提案による)

テキストのフォーマット用のキーボードショートカット

LabVIEW環境でテキストを編集する場合は、次のキーボードショートカットを使用してフォントスタイルをフォーマットします。

- <Ctrl-B>—太字のテキスト。
- <Ctrl-I>—斜体のテキスト。
- <Ctrl-U>—下線付きのテキスト。

(NIディスカッションフォーラムメンバーvt92の提案による)

ブロックダイアグラムの改善点

LabVIEW 2018では、ブロックダイアグラムおよびそれに関連する機能が以下のように改善されました。

並列Forループでのエラー処理の改善

LabVIEW 2018では、並列反復が有効になっているForループでのエラー処理を簡素化するため、エラーレジスタが追加されました。次のブロックダイアグラムにあるように、エラーレジスタは、並列Forループでエラークラスタのシフトレジスタの役割を果たします。

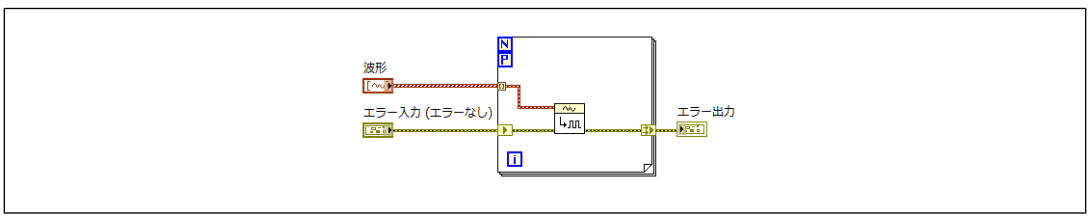

図 3

エラーレジスタは、並列反復からのエラーを自動的に統合します。Forループで並列反復を構成した場合、LabVIEWは、シフトレジスタをエラーレジスタに自動的に変換することにより、シフトレジスタによるエラー転送というベストプラクティスを維持します。

エラーレジスタとシフトレジスタは、実行時の動作が異なります。エラーレジスタの左側の端子は、指標が付いていない入力トンネルと同じように動作し、すべての反復で同じ値を生成します。エラーレジスタの右側の端子は、各反復の値を統合し、一番最初(指標による)の反復からのエラーまたは警告値がエラーレジスタの出力値になります。Forループがゼロ回反復した場合は、左側のトンネルに配線した値が、右側のトンネルの出力に渡されます。

オブジェクトの削除および再配線の改善

選択したブロックダイアグラムオブジェクトを削除して再配線すると、選択四角形内にあるフリーラベルを含むすべての装飾体も削除されます。オブジェクトを削除して再配線するには、ブロックダイアグラムオブジェクトの周りに選択四角形をドラッグして右クリックし、

削除して再配線を選択します。または、オブジェクトを選択した後、**クイックドロップ**キーボードショートカットの<Ctrl-Space>と<Ctrl-R>キーを使用して、オブジェクトを削除して再配線することもできます。

フロントパネルの改善点

NXGスタイルの制御器と表示器

制御器パレットに、**NXGスタイル**のフロントパネル制御器と表示器が新たに追加されました。NXGスタイルの制御器と表示器を使用すると、LabVIEW NXGと同じスタイルのフロントパネルを作成できます。これらの制御器と表示器の外観は、エンドユーザがそのVIを実行するプラットフォームによって異なります。これらの制御器と表示器を使用すると、VIをLabVIEW NXGに移植した場合のフロントパネルの変形を最小限に抑えられます。

新しいVIおよび関数

LabVIEW 2018には、以下のVIおよび関数が新たに追加されました。

- 「比較」パレットに、以下のVIと関数を含む「タイプをアサート」サブパレットが新たに追加されました。
  - 配列次元カウントをアサート
  - 配列次元サイズをアサート
  - 複素数値タイプをアサート
  - エラークラスタタイプをアサート
  - 固定小数点数値タイプをアサート
  - 浮動小数点数値タイプをアサート
  - 小数値タイプをアサート
  - 整数タイプをアサート
  - 浮動小数点実数値タイプをアサート
  - 実数値または実数波形タイプをアサート
  - 実数値タイプをアサート
  - 同一または派生タイプをアサート
  - スカラ数値または波形タイプをアサート
  - スカラ数値タイプをアサート
  - 符号付き整数タイプをアサート
  - ストラクチャタイプの一致をアサート
  - 符号なし整数タイプをアサート
  - タイプ特化ストラクチャ
- 「接続」パレットに、以下の関数を含む「Python」サブパレットが新たに追加されました。
  - Pythonセッションを開く
  - Pythonノード
  - Pythonセッションを閉じる
- 「変換」パレットに、「タイプに強制変換」関数が新たに追加されました。この関数は、データ値を維持して、入力データを互換性のあるデータタイプに変換するために使用します。「型変換」関数とは異なり、この関数は入力データを再解釈しません。この関数は以下の目的で使用します。
  - 強制ドットを除去する
  - タイプ定義のないデータを、互換性のあるタイプ定義に変換する、あるいはその逆
  - ユーザーイベントrefnumなど、ワイヤ上のデータの名前を変更する

(NIディスカッションフォーラムメンバーJackDunawayの提案による)

- 「タイミング」パレットに、「高解像度ポーリング待機」VIが新たに追加されました。このVIは、「待機 (ms)」関数で取得可能なよりも高解像度で指定された秒数間待機するために使用します。

新しいプロパティとメソッドおよび変更点

LabVIEW 2018では、以下のプロパティおよびメソッドが新たに追加または変更されました。

- LeftShiftRegisterクラスに、「エラーレジスタ?」プロパティが新たに追加されました。このプロパティは、シフトレジスタがエラーレジスタかどうかを読み取るために使用します。エラーレジスタは、特殊な形式のシフトレジスタであり、シフトレジスタのデータタイプがエラークラスタである場合に、並列反復が有効になっているForループ上に存在します。
- VIクラスに、「パネルをトップレベル非表示に設定する」メソッドが新たに追加されました。このメソッドは、VIのフロントパネルを非表示にし、オプションで、VIがトップレベルVIとして実行する際にタスクバーで非表示にするために使用します。たとえば、LabVIEWでビルドされたスタンドアロンアプリケーションのスタートアップVIのフロントパネルを非表示にする場合は、このメソッドを使用します。
- 無効ストラクチャクラスに、無効スタイルプロパティが新たに追加されました。このプロパティは、ストラクチャがダイアグラム無効ストラクチャ、条件無効ストラクチャ、またはタイプ特化ストラクチャであるかを読み取るために使用します。
- 無効スタイルを変更(クラス:無効ストラクチャ)メソッドの**無効スタイル**パラメータに、**タイプ特化スタイル**オプションが新たに追加されました。このオプションは、ダイアグラム無効ストラクチャまたは条件無効ストラクチャをタイプ特化ストラクチャに変更するために使用します。

旧バージョンのLabVIEWの機能と変更点

お使いのバージョン以降にリリースされたLabVIEWの各バージョンの新機能を特定するには、それらのバージョンのアップグレードノートを参照してください。これらのドキュメントにアクセスするには、NIウェブサイト (`ni.com/jp/info`) で以下のリストから適切なLabVIEWバージョン用のInfo Codeを入力してください。

- LabVIEW 2014アップグレードノート—`upnote14jp`
- LabVIEW 2015アップグレードノート—`upnote15jp`
- LabVIEW 2016アップグレードノート—`upnote16jp`
- LabVIEW 2017アップグレードノート—`upnote17jp`

371780P-0112

2018 年 03 月