LabVIEW™ 2017アップグレードノート

このアップグレードノートは、Windows、OS X、およびLinux用のLabVIEWをLabVIEW 2017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 2017に新たに追加された機能および動作

目次

LabVIEW 2017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することもできますが、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するため、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「lifecyclejp」と入力してください。

NATIONAL INSTRUMENTS

推奨アップグレードプロセスの概要

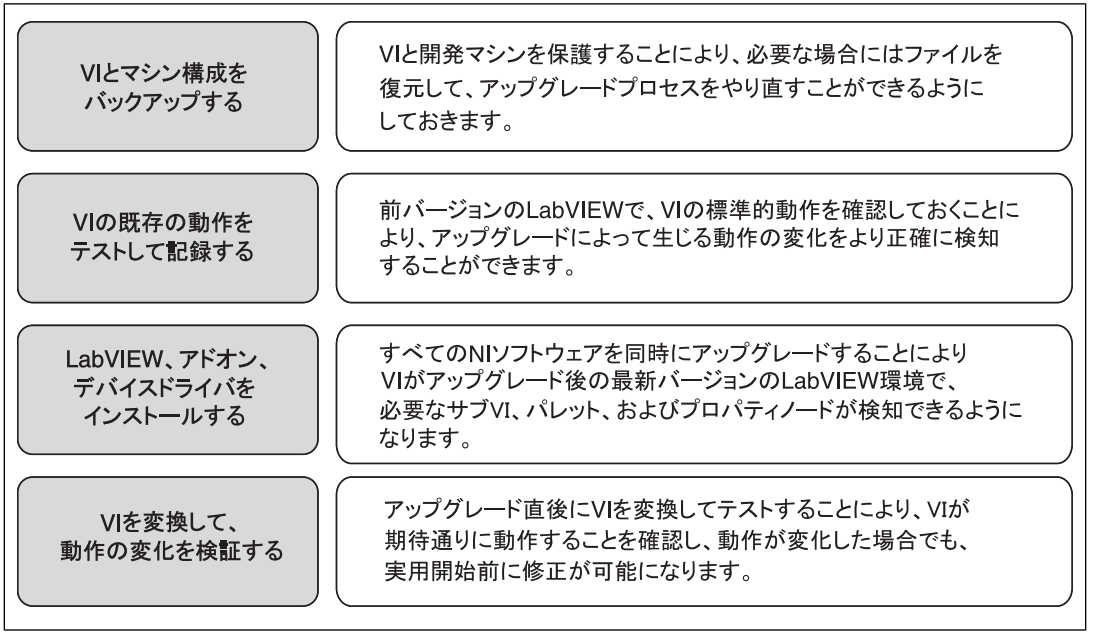

図 1

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「`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 2017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 2017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の一括コンパイル**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 2017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 2017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 2017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。

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 2017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 2017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 2017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。

- **アップグレードVIアナライザテスト**—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. 旧バージョンのLabVIEW以降のすべてのバージョンのLabVIEWのアップグレードVIアナライザテスト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 2014アップグレードVIアナライザテストで、変化する可能性のある動作が見つかった場合は、そのトピックの「LabVIEW 2013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**
  - 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 2017既知の問題リスト—LabVIEW 2017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「`lv2017ki`」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 2017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**—旧バージョンのLabVIEWで行ったのと同じテストをLabVIEW 2017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 2017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 2017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 2017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 2012以前からアップグレードする

LabVIEW 2012以前からLabVIEW 2017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「`upnote13jp`」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 2013からアップグレードする

LabVIEW 2013からLabVIEW 2017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 2014からアップグレードする」、「LabVIEW 2015からアップグレードする」、および「LabVIEW 2016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。

「文字列をパスに変換」関数の動作変更

LabVIEW 2014以降では、「文字列をパスに変換」関数は、`<Not A Path>`文字列の何らかのバリエーションを読み取った時に大/小文字を区別せず、常に<無効パス>定数を返します。たとえば、**文字列**入力には`<not a path>`または`<Not A Path>`と指定でき、いずれの場合も、この関数は<無効パス>という値の定数を返します。旧バージョンのLabVIEWでの「文字列をパスに変換」関数の動作については、下表を参照してください。

LabVIEW 2012および2013	LabVIEW 2011以前
「文字列をパスに変換」関数は、大/小文字に関係なく、<無効パス>定数を返しません。<Not A Path>のあらゆるバリエーションが指定でき、この関数は、<無効パス>定数を返すのではなく、<Not A Path>という名前のディレクトリのパスを返します。	LabVIEW 2014以降と同様、「文字列をパスに変換」関数は、大/小文字を区別せず、<Not A Path>文字列の何らかのバリエーションが指定されたときには、<無効パス>定数を出力します。<not a path>と指定された場合も、<Not a Path>と指定された場合も、この関数は、<無効パス>定数を返します。

タイプ定義を確認して更新する

LabVIEW 2013以前で表示される**タイプ定義から更新**ショートカットメニュー項目は、**確認してタイプ定義から更新**ショートカットメニュー 項目に変更されました。

廃止されたVI、関数、およびノード

LabVIEW 2014以降は、以下のVI、関数、ノードをサポートしていません。

AppleイベントVI

（OS X）LabVIEW 2014以降は、AppleイベントVIをサポートしていません。LabVIEW外部のOS Xアプリケーションと通信するには、ライブラリ&実行可能ファイルパレットの「Run AppleScript Code」VIを使用してください。以下のいずれかのAppleイベントVIを含むVIをロードしようとすると、LabVIEWでエラーが発生し、VIが実行できない可能性があります。

- AESend Do Script
- AESend Finder Open
- AESend Open
- AESend Open Document
- AESend Print Document
- AESend Quit Application
- Get Target ID
- AESend Abort
- AESend Close
- AESend Open、Run、Close
- AESend Run
- AESend VI Active?
- AECreate Comp Descriptor
- AECreate Descriptor List
- AECreate Logical Descriptor
- AECreate Object Specifier
- AECreate Range Descriptor
- AECreate Record
- AESend
- Make Alias

アクターフレームワークVI

LabVIEW 2014以降は、「アクター:アクターを起動」VIをサポートしていません。代わりに、「アクター:ルートアクターを起動」VIまたは「アクター:ネストされたアクターを起動」VIを使用してください。

「In Port」VIおよび「Out Port」VI

LabVIEW 2014以降は、「In Port」VIおよび「Out Port」VIをサポートしていません。

廃止されたプロパティ、メソッド、およびイベント

LabVIEW 2014以降は、アプリケーションクラスの古いヘルプ情報メソッドをサポートしていません。指定されたVIの**VIプロパティ**ダイアログボックスの**ドキュメント**ページからヘルプ情報を返すには、代わりにVIを取得:ヘルプ情報メソッドを使用してください。

LabVIEW 2014からアップグレードする

LabVIEW 2014からLabVIEW 2017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」および「LabVIEW 2016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 2017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」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 2017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。

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。

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として「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 2017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。

VIのロード時間およびコンパイル時間の削減

（Windows）LabVIEW 2017では、LabVIEW開発環境およびLabVIEWランタイムエンジンのビルド用コンパイラがより高速なものにアップグレードされました。このアップグレードにより、VIのロード時間とコンパイル時間の合計が削減されました。

オブジェクト移動時にワイヤ接続を維持

LabVIEW 2017では、ブロックダイアグラム上でオブジェクトをストラクチャに出し入れした際にワイヤの接続が自動的に維持されます。ストラクチャに出し入れするオブジェクトがそのストラクチャ内のオブジェクトに接続されている場合、LabVIEWはトンネルを作成または削除してワイヤの接続を維持します。<W>キーを押すと、オブジェクトを移動した際の自動ワイヤ接続をトグルできます。

順応性VI

LabVIEW 2017には、発呼者VIにインライン化され、対応する入力データタイプに各端子を適応できる順応性VI（.vim）が追加されました。順応性VIを使用すると、使用可能なすべてのデータタイプに対して同じ操作を実行するVIを作成できるため、データタイプごとに別々のVIを保存する必要はありません。

順応性VIは、多態性VIと似ていますが、使用可能なデータタイプをより柔軟に特定できます。多態性VIでは、使用できるデータタイプがあらかじめ定義されています。順応性VIは、データタイプが実装可能かどうか計算します。

順応性VIのファイル拡張子は.vimです。順応性VIは、**ファイル→新規**を選択し、**新規**ダイアログボックスから**順応性VI**を選択すると作成できます。既存のVIのファイル拡張子を.vimに変更すると順応性VIに変換されます。

メモ 順応性VIに変換できるのは、標準のVIのみです。多態性VI、グローバルVI、またはXControl機能は順応性VIに変換できません。

組込順応性VI

LabVIEWには、アプリケーションに使用可能な以下の順応性VIが用意されています。組込順応性VIのアイコンはオレンジ色です。

- 配列パレット
  - 配列要素を減分—1D配列内の指定された要素から1を減算します。配列がタイムスタンプの配列の場合、このVIは要素から1秒を減算します。
  - 配列要素を増分—1D配列内の指定された要素に1を加算します。配列がタイムスタンプの配列の場合、このVIは要素に1秒を加算します。
  - 1D配列シャッフル—1D配列の要素を擬似順序で並べ替えます。
  - 2D配列シャッフル—2D配列の要素を擬似順序で並べ替えます。

  - 2D配列ソート—指定された行または列を昇順にソートして2D配列の行または列を並べ替えます。
- 比較パレット
  - 値が変更された?—これがこのVIの初めての呼び出しの場合、または入力値がこのVIが最後に呼び出されたときの値と異なる場合にTRUEを返します。
- 変換パレット
  - 数値を列挙体に変換—指定された数値に一致する列挙体値を検索し、該当する列挙体項目を返します。
- タイミングパレット
  - データフローを一時停止—指定された期間、ワイヤのデータフローを遅延させます。

順応性VIの詳細については、『LabVIEWヘルプ』の**目次**タブから**基本機能→VIおよびサブVIを作成する→概念→モジュールコードを作成する→順応性VI**トピックを選択して参照してください。

順応性VIを使用したサンプルについては、`labview¥examples¥Malleable VIs¥Basics¥Malleable VIs Basics.lvproj`を参照してください。

(NIディスカッションフォーラムメンバーDanyAllardの提案による)

追加および変更されたVIおよび関数

LabVIEW 2017では、以下のVIおよび関数が新たに追加または変更されました。

データ値リファレンスの読み取り専用アクセス

In Place要素ストラクチャのデータ値リファレンス要素読み取り/書き込み境界ノードは、データ値リファレンスへの読み取り専用アクセスを許可します。ストラクチャの右側にある境界ノードを右クリックし、**並列読み取り専用アクセスを許可**を選択します。右側の境界ノードが未配線の場合、複数の読み取り専用操作が同時に実行することができ、データ値リファレンスは変更されません。

新規チャンネルテンプレート

LabVIEW 2017にはイベントメッセンジャーチャンネルテンプレートが追加されました。このチャンネルは、複数の書き込みから1つまたは複数のイベントストラクチャにデータを送信するために使用します。チャンネルへの各書き込み操作は、イベントをトリガします。イベントメッセンジャーチャンネルでは、チャンネルの構文とユーザインタフェースと生成されたイベントを制御する構文の両方を使用できます。イベントメッセンジャーチャンネルを使用したサンプルについては、`labview¥examples¥Channels¥Event Messenger¥Channel - Event Messenger.lvproj`を参照してください。

追加および変更されたクラス、プロパティ、メソッド、およびイベント

LabVIEW 2017では、VIの依存項目を取得（名前&パス）メソッドが変更されました。**Express VIを保持する?**パラメータは、**Express VIと順応性VIを保持する?**という名前に変更されました。**Express VIと順応性VIを保持する?**がFALSE（デフォルト）の場合、LabVIEWはExpress VIおよび順応性VIのベースとなっている非表示のインスタンスVIの名前を返します。TRUEの場合、LabVIEWはExpress VIおよび順応性VIを依存項目として返します。編集時の依存項目を取得するには、**Express VIと順応性VIを保持する?**をTRUEに設定します。実行時の依存項目を取得するには、**Express VIと順応性VIを保持する?**をFALSEに設定します。この設定がいずれであっても、インスタンスVIのサブVIは、リファレンスVIの依存項目として含まれます。

アプリケーションビルダの改善点

LabVIEW 2017には、LabVIEWアプリケーションビルダおよびビルド仕様に関して以下の改善点が含まれています。

LabVIEWランタイムエンジンの下位互換性

前のバージョンのLabVIEWでは、再コンパイルをせずに、古いバージョンのLabVIEWでビルドされたバイナリやVIをロードしたり、実行したりすることはできません。2017からは、LabVIEWはLabVIEWランタイムエンジンの下位互換性をサポートするようになりました。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ります。ビルド仕様を特定のバージョンのLabVIEWに関連付けるには、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。

LabVIEWおよび他の言語との間の呼び出しの改善

LabVIEW 2017では、LabVIEWおよび他の言語からLabVIEWでビルドされた共有ライブラリ（DLL）を呼び出した場合のパフォーマンスおよび安定性が大幅に改善されました。たとえば、LabVIEWでビルドされた共有ライブラリは、C言語アプリケーションから呼び出されるとマルチスレッド実行システムで実行します。この改善により、LabVIEWからLabVIEWでビルドされたDLLを呼び出した場合も、デッドロックや原子性違反などを一部回避できるようになりました。

この機能を使用するには、**共有ライブラリプロパティ**ダイアログボックスの**上級**ページの**プライベート実行システムでVIを実行**チェックボックスをオンにします。このオプションは、新規ビルド仕様ではデフォルトで有効になっています。LabVIEW 2016以前から移植されたビルド仕様では、意図しない動作の変更を防ぐため、このオプションは無効になっています。たとえば、このオプションを無効にすることにより、LabVIEWでビルドされたシングルスレッドに依存する共有ライブラリが、LabVIEW以外のアプリケーションから呼び出された場合にマルチスレッド実行システムで実行するのを防ぐことができます。**（NI Linuxリアルタイム）**パフォーマンスジッタが発生する可能性があるため、このオプションはLinux RTターゲットではデフォルトで無効になっています。

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 2013アップグレードノートーupnote13jp
- LabVIEW 2014アップグレードノートーupnote14jp
- LabVIEW 2015アップグレードノートーupnote15jp
- LabVIEW 2016アップグレードノートーupnote16jp

371780N-0112 2017 年 03 月