LabVIEW™

ユーザマニュアル

NATIONAL INSTRUMENTS™

2003 年 6 月版
製品番号 321187E-01

インターネットサポート
サポート電子メール：supportjapan@ni.com
電子メール：infojapan@ni.com
FTP サイト：ftp.ni.com
日本語ホームページ：http://www.ni.com/jp

電話サポート（日本）
Tel：03-5472-2981
Fax：03-5472-2977

日本ナショナルインスツルメンツ株式会社
〒 105-0011 東京都港区芝公園 2-4-1 秀和芝パークビル A 館 4F Tel：03-5472-2970

National Instruments Corporation
11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

海外オフィス
イスラエル 972 0 3 6393737、イタリア 39 02 413091、インド 91 80 51190000、英国 44 0 1635 523545、
オーストラリア 1800 300 800、オーストリア 43 0 662 45 79 90 0、オランダ 31 0 348 433 466、
カナダ（オタワ）613 233 5949、カナダ（カルガリー）403 274 9391、カナダ（ケベック）514 694 8521、
カナダ（トロント）905 785 0085、カナダ（バンクーバー）514 685 7530、
カナダ（モントリオール）514 288 5722、韓国 82 02 3451 3400、ギリシャ 30 2 10 42 96 427、
シンガポール 65 6226 5886、スイス 41 56 200 51 51、スウェーデン 46 0 8 587 895 00、
スペイン 34 91 640 0085、スロベニア 386 3 425 4200、タイ 662 992 7519、台湾 886 2 2528 7227、
中国 86 21 6555 7838、チェコ 420 2 2423 5774、デンマーク 45 45 76 26 00、ドイツ 49 0 89 741 31 30、
ニュージーランド 1800 300 800、ノルウェー 47 0 66 90 76 60、フィンランド 385 0 9 725 725 11、
フランス 33 0 1 48 14 24 24、ベルギー 32 0 2 757 00 20、ブラジル 55 11 3262 3599、
ポーランド 48 0 22 3390 150、ポルトガル 351 210 311 210、マレーシア 603 9131 0918、
南アフリカ 27 0 11 805 8197、メキシコ 001 800 010 0793、ロシア 7 095 238 7139

サポート情報の詳細については、付録 D「技術サポートおよびサービス」を参照してください。本書に対するご意見は、techpubs@ni.com まで電子メールでお送りください。

© 1997–2003 National Instruments Corporation. All rights reserved.

必ずお読みください

保証

限定的保証：National Instruments Corporation（以下「NI」という）のハードウェア製品は、NIがお客様に製品を出荷した日（以下「配送日」）から次の一定期間、素材及び製作技術上の欠陥に対して保証されています。すなわちIEEE 488に未対応のハードウェア製品については1年間、IEEE 488対応のハードウェア製品については2年間、ケーブルについては90日間の保証が適用されます。ソフトウェア製品の場合は、該当するNIのライセンス条項に基づき、お客様にライセンスが供与されます。配送日から90日間は、NIのソフトウェア製品（但しNIのハードウェア製品に正しくインストールされている場合）について、(a)付属のマニュアル文書に従い実質的に機能すること、および(b)ソフトウェア製品が記録されている媒体は、通常の利用やサービスにおいて素材及び製作技術上の欠陥を有しないこと、が保証されています。ライセンスが供与されたソフトウェア製品の交換については、当初の保証期間の残存期間または30日間のいずれか長い期間について保証されます。お客様が保証期間中の製品をNIに返却するには、事前にNIから返品確認（Return Material Authorization: RMA）番号を取得してください。また、修理・交換品をお客様からNIへ、NIからお客様あてに返送する送料は、お客様の負担になります。返却された製品を検査、試験した後、同製品には欠陥がないとNIが判断した場合、その旨をお客様に通知します。同製品の返送にかかる費用はお客様に負担いただき、試験にかかった費用については後日請求致します。製品の不具合が事故、乱用、誤用、お客様による不適切なキャリブレーションによって発生した場合や、お客様が当該NIソフトウェアと共に使用することが予定されていない第三者のソフトウェアと共に利用した場合、不適切なハードウェアまたはソフトウェアのキーを利用した場合、独断で保守または修理を行った場合、本書に定める限定的保証は無効となります。

救済方法：上記の限定的保証において、NIの唯一の義務（およびお客様の唯一の救済方法）は、NIの選択により、支払われた料金の返還、または欠陥製品の修理・交換に限定されます。ただし、NIが、当該製品に適用される保証期間内に、こうした欠陥について書面で通知を受け取った場合に限ります。お客様は、訴訟原因の発生から1年を超えて経過した後は、上記の限定的保証に基づく本救済方法を強制するために訴訟を提起することはできません。

返品および解約に関する方針：お客様は、不要な製品については、配送日から30日以内であれば、当該製品を返却することができます。この場合の送料はお客様にご負担いただきます。上記30日間満了後は不要な製品の返品は受け付けません。特殊機器または特殊なサービスが係わる場合、お客様は、進行中の関連作業全てに対して責任を負うものとします。ただし、お客様から書面による解約の通知を受領した場合、NIはただちに損害を軽減するための責任ある対策を講ずるものとします。製品の返却の際は、NIから返品確認番号を取得してください。お客様がNIに対して行った説明・表示等が虚偽または誤解を生じさせるものであった場合には、NIは注文を取り消すことがあります。

本書の内容については万全を期しており、技術的内容に関するチェックも入念に行っております。技術的な誤りまたは乱丁・落丁につきましては、お客様への事前の通告なく、NIにて次の版から修正する権利があるものとします。本書で誤りと思われる個所については、NIにご確認ください。NIは、本書およびその内容により、またはそれに関連して発生した損害に対して一切責任を負いません。

本書に規定する保証を唯一の保証とします。NIは、明示・暗示を問わず、ここに記載された以外の保証は行いません。特に、商品適合性の保証や特定用途に対する適合性についての保証は行いません。NIの過失または不注意により発生した損害に関するお客様の賠償請求権は、お客様が製品に支払われた金額を上限とします。NIは、データの消失、利益の逸失、製品の使用から生じた損失や、付随的または結果的に生じた損害に対して、その損害が発生する可能性を通知されていた場合でも、一切の責任を負いません。かかるNIの限定的責任は、訴訟方式、過失責任を含む契約上の責任または不法行為責任を問わず適用されます。NIに対する訴訟は、訴訟原因の発生から1年以内に提起する必要があります。NIは、NIが合理的に支配可能な範囲を超えた原因により発生した履行遅延に関しては一切の責任を負いません。所有者が、NIの指示通りインストール、操作、保守を実施しないことにより発生した損害、欠陥、誤作動、動作不良について、また、所有者による製品の改変、乱用、誤用、または不注意な行動、さらに停電、電源サージ、火災、洪水、事故、第三者の行為、その他の合理的に支配可能な範囲を超えた事象により発生する損害、欠陥、誤作動、動作不良については本書に定める保証の対象となりません。

著作権

著作権法に基づき、National Instruments Corporationの事前の承諾なく、複写、記録、情報検索システムへの保存および翻訳を含め、本書のすべてまたは一部をいかなる手段によっても複製または転載することを禁止します。

商標

CVI™、DAQPad™、DataSocket™、DIAdem™、IMAQ™、IVI™、LabVIEW™、Measurement Studio™、National Instruments™、NI™、ni.com™、NI-DAQ™、NI Developer Zone™、NI-IMAQ™、NI-VISA™、NI-VXI™、SCXI™ は、National Instruments Corporationの商標です。本書に掲載されている製品および会社名は該当各社の商標または商号です。

特許

National Instruments製品を保護する特許については、ソフトウェアに含まれている特許情報（**ヘルプ→特許情報**）、CDに含まれている`patents.txt`ファイル、または`ni.com/patents`のうち、該当するリソースから参照してください。

National Instrumentsの製品を医療用に使用することに関する警告

（1）National Instruments Corporation（以下「NI」という）の製品は、外科移植もしくはそれに関連する用途、または作動不良により人体に深刻な傷害を及ぼすことが合理的に予期される生命維持装置の重要なコンポーネントとしての用途に適した信頼性のレベルでのコンポーネントや試験を採用して設計されておりません。（2）上記用途を含む、あらゆるアプリケーションにおいて、不利な要因によってソフトウェア製品の操作の信頼性が損なわれる可能性があります。これには、電力供給の変動、コンピュータハードウェアの誤作動、コンピュータ・オペレーティングシステム・ソフトウェアの適応性、アプリケーション開発に利用したコンパイラや開発ソフトウェアの適応性、インストールの間違い、ソフトウェアとハードウェアの互換性の問題、電子監視機器または制御機器の誤作動または故障、電気システム（ハードウェア及び/又はソフトウェア）の一時的な障害、予期せぬ使用または誤用、ユーザまたはアプリケーション設計者側のミスなどがありますが、これに限定されません(本書においてこのような不利な要因を総称して「システム故障」といいます)。システム故障が財産または人体に危害を及ぼす可能性（身体の損傷および死亡の危険を含む）があるアプリケーションにおいては、システム故障の危険があるため、単独の電気システム方式のみに依存すべきではありません。損害、人体への傷害、または死亡といった事態を避けるため、ユーザまたはアプリケーション設計者は、システム故障から保護するための合理的に慎重な対策を取る必要があります。これには、バックアップメカニズム、または非常停止メカニズムなどがありますが、これに限定されません。各エンドユーザのシステムはカスタマイズされており、NIの試験プラットフォームとは異なること、またユーザやアプリケーション設計者が、NIが評価したことのない方法や、予期しない方法でNI製品を他の製品と組み合わせて使用する可能性があることから、NI製品をシステムまたはアプリケーションに統合する場合は、ユーザまたはアプリケーション設計者が、最終的にNI製品の適合性(かかるシステムまたはアプリケーションの適切な設計、処理、安全レベルが含まれますが、これに限定されません。）の検証および確認における責任を負うものとします。

目次

本書について

本書の構成 xx
表記規則 xx

第 I 部
LabVIEW の概念

第 1 章
LabVIEW の概要

LabVIEW の関連資料 1-1
LabVIEW のサンプル VI およびツール 1-4
LabVIEW のテンプレート VI 1-4
LabVIEW のサンプル VI 1-4
LabVIEW のツール 1-4

第 2 章
バーチャルインスツルメンツ（仮想計測器）VI の概要

フロントパネル 2-1
ブロックダイアグラム 2-2
端子 2-3
ノード 2-4
ワイヤ 2-4
ストラクチャ 2-4
アイコンとコネクタペーン 2-4
VI およびサブ VI の使用とカスタマイズ 2-5

第 3 章
LabVIEW 環境

制御器パレット 3-1
関数パレット 3-2
制御器パレットおよび関数パレットを操作する 3-2
ツールパレット 3-3
メニューとツールバー 3-3
メニュー 3-4
ショートカットメニュー 3-4
実行モード時のショートカットメニュー 3-4
ツールバー 3-4
ヘルプウィンドウ 3-5

作業環境をカスタマイズする 3-6
制御器および関数パレットをカスタマイズする 3-6
VI と制御器をユーザおよび計測器ドライバサブパレットに追加する 3-6
カスタムパレットセットの作成と編集 3-7
LabVIEW のパレットセットの格納方法 3-7
ActiveX サブパレットを作成する 3-8
パレットにツールセットとモジュールを表示する 3-8
作業環境オプションを設定する 3-8
LabVIEW のオプションの格納方法 3-8
Windows 3-8
Mac OS 3-9
UNIX 3-9

第 4 章
フロントパネルを作成する

フロントパネルオブジェクトを構成する 4-1
オプション項目を表示または非表示にする 4-2
制御器を表示器に、表示器を制御器に変更する 4-2
フロントパネルのオブジェクトを入れ替える 4-2
フロントパネルを構成する 4-3
制御器にキーボードショートカットを設定する 4-3
キー操作でボタン動作を制御する 4-4
フロントパネルオブジェクトのタブ順序を設定する 4-4
オブジェクトの色を決める 4-5
インポートされたグラフィックを使用する 4-5
オブジェクトを整列および均等に配置する 4-6
オブジェクトをグループ化およびロックする 4-6
オブジェクトをサイズ変更する 4-7
フロントパネルオブジェクトをスケールする 4-7
ウィンドウをサイズ変更せずにフロントパネルにスペースを追加する 4-9
フロントパネルの制御器および表示器 4-9
3 次元および旧バージョンの制御器と表示器 4-9
スライド、ノブ、ダイアル、デジタル表示、およびタイムスタンプ 4-10
スライド制御器および表示器 4-10
回転式制御器および表示器 4-10
デジタル制御器および表示器 4-11
数値形式 4-11
タイムスタンプ制御器および表示器 4-11
カラーボックス 4-12
カラーランプ 4-12
グラフおよびチャート 4-12
ボタン、スイッチ、およびライト 4-12

テキスト入力ボックス、ラベル、およびパス表示....4-13
文字列制御器および表示器....4-13
コンボボックス制御器....4-13
パス制御器および表示器....4-14
無効なパス....4-14
空のパス....4-14
配列とクラスタの制御器と表示器....4-15
リストボックス、ツリー制御器、および表....4-15
リストボックス....4-15
ツリー制御器....4-16
表....4-16
リングおよび列挙体制御器と表示器....4-17
リング制御器....4-17
列挙体制御器....4-18
上級の列挙体制御器および表示器....4-18
コンテナ制御器....4-19
タブ制御器....4-19
サブパネル制御器....4-19
I/O 名制御器および表示器....4-20
波形制御器....4-21
デジタル波形制御器....4-21
デジタルデータ制御器....4-21
データをデジタルデータに変換する....4-23
部分デジタル信号を読み取る....4-24
デジタルサンプルおよび信号を連結する....4-24
デジタルデータを圧縮する....4-24
パターンを検索する....4-25
オブジェクトまたはアプリケーションへのリファレンス....4-25
ダイアログ制御器と表示器....4-26
ラベルを付ける....4-26
キャプション....4-27
テキストの特性....4-27
ユーザインタフェースを設計する....4-29
フロントパネルの制御器および表示器....4-29
ファイルダイアログボックスを設計する....4-30
画面サイズを選択する....4-30

第 5 章
ブロックダイアグラムを作成する

フロントパネルオブジェクトとブロックダイアグラム端子の関係 5-1
ブロックダイアグラムオブジェクト 5-1
ブロックダイアグラム端子 5-1
制御器および表示器のデータタイプ 5-2
定数 5-4
ユニバーサル定数 5-5
ユーザ定義定数 5-5
ブロックダイアグラムのノード 5-5
関数の概要 5-7
数値関数 5-7
ブール関数 5-7
文字列関数 5-7
配列関数 5-8
クラスタ関数 5-8
比較関数 5-8
時間＆ダイアログ関数 5-8
ファイル I/O 関数 5-9
波形関数 5-9
アプリケーション制御関数 5-9
上級関数 5-9
関数に端子を追加する 5-10
ブロックダイアグラムオブジェクトをワイヤで接続する 5-10
オブジェクトを自動配線する 5-12
オブジェクトを手動配線する 5-12
ワイヤの経路を調整する 5-13
ワイヤを選択する 5-14
不良ワイヤを修正する 5-14
強制ドット 5-14
多形性 VI および関数 5-15
多形性 VI 5-15
多形性 VI を作成する 5-16
多形性関数 5-18
Express VI 5-18
サブ VI として Express VI を作成する 5-19
ダイナミックデータタイプ 5-19
ダイナミックデータから変換する 5-20
ダイナミックデータへの変換 5-21
バリアントデータを処理する 5-21
数値単位および厳密タイプチェック 5-22
単位および厳密タイプチェック 5-23

ブロックダイアグラムのデータフロー 5-25
データ依存と人工データ依存 5-26
データ依存が存在しない場合 5-26
データフローとメモリ管理 5-27
ブロックダイアグラムを設計する 5-28

第 6 章
VI の実行とデバッグ

VI を実行する 6-1
VI の実行方法を構成する 6-2
壊れた VI を修正する 6-2
壊れた VI の原因を調べる 6-2
壊れた VI の一般的な原因 6-3
デバッグ方法 6-3
実行のハイライト 6-5
シングルステップ 6-6
プローブツール 6-6
プローブのタイプ 6-6
一般 6-6
表示器を使用してデータを表示する 6-7
指定された 6-7
カスタム 6-8
ブレークポイント 6-8
実行を中断する 6-9
サブ VI の現在のインスタンスを調べる 6-10
ブロックダイアグラムのセクションをコメントアウトする 6-10
デバッグツールを無効にする 6-10
不定データまたは予想外のデータ 6-10
ループのデフォルトデータ 6-11
For ループ 6-11
配列内のデフォルトデータ 6-11
不定データを防ぐ 6-12
エラーチェックとエラー処理 6-12
エラーをチェックする 6-13
エラー処理 6-13
エラークラスタ 6-14
エラー処理に While ループを使用する 6-14
エラー処理にケーストラクチャを使用する 6-14

第 7 章
VI およびサブ VI を作成する

プロジェクトの計画と設計 7-1
複数の開発者とともにプロジェクトを設計する 7-2
VI テンプレート 7-2
VI テンプレートを作成する 7-3
その他のドキュメントタイプ 7-3
組み込み VI および関数を使用する 7-3
計測器制御およびデータ集録 VI および関数を作成する 7-3
他の VI にアクセスする VI を作成する 7-4
他のアプリケーションと通信する VI を作成する 7-4
サブ VI 7-5
共通の操作を監視する 7-5
コネクタペーンを設定する 7-6
必須、推奨、および任意の入出力を設定する 7-8
アイコンを作成する 7-8
アイコンまたは拡張可能なノードとして
サブ VI および Express VI を表示する 7-9
VI の一部からサブ VI を作成する 7-10
サブ VI を設計する 7-11
VI の階層を表示する 7-11
VI を保存する 7-12
VI を個別ファイルとして保存する場合の利点 7-12
VI をライブラリとして保存する場合の利点 7-12
ライブラリ内で VI を管理する 7-13
VI に名前を付ける 7-13
旧バージョンで保存する 7-14
VI を配布する 7-14
スタンドアロンアプリケーションと共有ライブラリを作成する 7-15

第 II 部
VI の作成と編集

第 8 章
ループとストラクチャ

For ループおよび While ループのストラクチャ 8-2
For ループ 8-2
While ループ 8-2
無限 While ループを回避する 8-3
ループに自動指標付けを行う 8-4
自動指標付けで For ループの回数を設定する 8-4
While ループで自動指標付けを行う 8-5
ループを使用して配列を作成する 8-5

ループ内のシフトレジスタおよび
「フィードバックノード（Feedback Node）」 8-6
シフトレジスタ 8-6
スタックシフトレジスタ 8-7
シフトレジスタをトンネルと入れ替える 8-8
トンネルをシフトレジスタに入れ替える 8-8
フィードバックノード（Feedback Node） 8-9
フィードバックノードを初期化する 8-10
シフトレジスタをフィードバックノードに入れ替える 8-10
タイミングを制御する 8-10
ケースストラクチャとシーケンスストラクチャ 8-11
ケースストラクチャ 8-11
ケースセレクタの値とデータタイプ 8-12
入力トンネルと出力トンネル 8-13
エラー処理にケースストラクチャを使用する 8-13
シーケンスストラクチャ 8-13
フラットシーケンスストラクチャ 8-13
スタックシーケンスストラクチャ 8-14
シーケンスストラクチャを使用する 8-14
シーケンスストラクチャの過度な使用を避ける 8-15
シーケンスストラクチャを入れ替える 8-16

第 9 章
イベント駆動型プログラミング

イベントとは 9-1
イベントを使用する利点 9-2
イベントストラクチャのコンポーネント 9-2
通知およびフィルタイベント 9-4
LabVIEW でイベントを使用する 9-5
イベントの静的登録 9-8
イベントの動的登録 9-8
ダイナミックイベントのサンプル 9-11
登録を動的に変更する 9-11
ユーザイベント 9-12
ユーザイベントを作成して登録する 9-13
ユーザイベントを生成する 9-13
ユーザイベントを登録解除する 9-14
ユーザイベントの例 9-14

第 10 章
文字列、配列、およびクラスタを使用してデータをグループ化する

文字列 10-1
フロントパネルの文字列 10-2
文字列表示タイプ 10-2
表 10-2
文字列をプログラム的に編集する 10-3
文字列をフォーマットする 10-4
指定子をフォーマットする 10-4
数値と文字列 10-5
データタイプを XML に変換する 10-5
XML ベースのデータタイプ 10-7
LabVIEW XML スキーマ 10-8
データを配列やクラスタとグループ化する 10-8
配列 10-8
指標 10-8
配列の例 10-8
配列に関する制約 10-10
配列制御器、表示器、および定数を作成する 10-11
配列指標表示 10-11
配列関数 10-12
配列関数を自動的にサイズ変更する 10-12
クラスタ 10-13

第 11 章
ローカル変数とグローバル変数

ローカル変数 11-1
ローカル変数を作成する 11-2
グローバル変数 11-2
グローバル変数を作成する 11-2
読み取り変数と書き込み変数 11-3
ローカル変数とグローバル変数を慎重に使用する 11-4
ローカルおよびグローバル変数を初期化する 11-4
競合状態 11-4
ローカル変数を使用する場合のメモリに関する注意事項 11-5
グローバル変数を使用する場合のメモリに関する注意事項 11-5

第 12 章
グラフおよびチャート

グラフとチャートのタイプ 12-1
グラフおよびチャートのオプション 12-2
グラフおよびチャート上の複数の x スケールと y スケール 12-2
グラフおよびチャートのエイリアス除去ラインプロット 12-2

グラフとチャートの外観をカスタマイズする 12-3
グラフをカスタマイズする 12-3
グラフカーソル 12-4
自動スケール 12-5
波形グラフのスケール凡例 12-5
軸のフォーマット 12-5
グラフを動的にフォーマットする 12-6
スムーズアップデート 12-6
チャートをカスタマイズする 12-7
チャート記録の長さ 12-7
チャートの更新モード 12-7
オーバーレイ対スタックプロット 12-8
波形グラフと XY グラフ 12-9
シングルプロット波形グラフのデータタイプ 12-9
マルチプロット波形グラフ 12-10
シングルプロット XY グラフのデータタイプ 12-11
マルチプロット XY グラフのデータタイプ 12-11
波形チャート 12-12
強度グラフおよびチャート 12-12
カラーマッピング 12-14
強度チャートオプション 12-14
強度グラフオプション 12-15
デジタル波形グラフ 12-15
データをマスクする 12-17
3 次元グラフ 12-18
波形データタイプ 12-19
デジタル波形データタイプ 12-19

第 13 章
グラフィック & サウンド VI

ピクチャ表示器を使用する 13-1
ピクチャプロット VI 13-2
極プロット VI をサブ VI として使用する 13-2
プロット波形 VI および XY プロット VI をサブ VI として使用する 13-2
スミスプロット VI をサブ VI として使用する 13-3
ピクチャ関数 VI 13-4
ピクチャ関数 VI を使用した色の作成と変更 13-5
グラフィック形式 VI 13-6
サウンド VI 13-6

第 14 章
ファイル I/O

ファイル I/O の基本 14-1
ファイル I/O 形式を選択する 14-2
テキストファイルを使用する場合 14-2
バイナリファイルを使用する場合 14-3
データログファイルを使用する場合 14-4
上位ファイル I/O VI を使用する 14-5
下位および上級のファイル I/O および関数を使用する 14-6
ディスクストリーミング 14-7
テキストファイルとスプレッドシートファイルを作成する 14-8
データのフォーマットとファイルへの書き込み 14-9
ファイルからデータをスキャンする 14-9
バイナリファイルを作成する 14-9
データログファイルを作成する 14-10
ファイルに波形を書き込む 14-10
ファイルから波形を読み取る 14-11
フロースルーパラメータ 14-12
構成ファイルを作成する 14-12
構成ファイルを使用する 14-13
Windows 構成ファイルの形式 14-13
フロントパネルのデータを記録する 14-15
自動および対話形式でのフロントパネルのデータロギング 14-15
記録済みのフロントパネルデータを対話形式で表示する 14-16
レコードを削除する 14-16
ログファイル連結を消去する 14-16
ログファイル連結を変更する 14-17
プログラムでフロントパネルデータを取り出す 14-17
サブ VI を使用してフロントパネルデータを取り出す 14-17
レコードを指定する 14-18
ファイル I/O 関数を使用してフロントパネルデータを取り出す 14-18
LabVIEW データディレクトリ 14-19
LabVIEW 計測データファイル 14-20

第 15 章
VI の文書化と印刷

VI を文書化する 15-1
VI のレビジョン履歴を設定する 15-1
レビジョン番号 15-2
VI およびオブジェクトの説明を作成する 15-2

ドキュメントを印刷する 15-3
HTML、RTF、またはテキストファイルに
ドキュメントを保存する 15-3
HTML ファイル用のグラフィック形式を選択する 15-4
グラフィックファイルの命名規約 15-4
独自のヘルプファイルを作成する 15-5
VI を印刷する 15-5
アクティブウィンドウを印刷する 15-5
VI をプログラムで印刷する 15-6
VI 実行後に VI のフロントパネルを印刷する 15-6
サブ VI を使用して上位 VI のデータを印刷する 15-6
レポートを生成および印刷する 15-7
その他の印刷方法 15-7

第 16 章
VI をカスタマイズする

VI の外観と動作を設定する 16-1
メニューをカスタマイズする 16-2
メニューを作成する 16-2
メニュー選択処理 16-3

第 17 章
VI をプログラム的に制御する

VI サーバの機能 17-1
VI サーバアプリケーションを作成する 17-2
アプリケーションリファレンスと VI リファレンス 17-3
アプリケーションと VI の設定値を操作する 17-3
プロパティノード 17-4
プロパティノードの自動リンク 17-4
インボークノード 17-5
アプリケーションクラスのプロパティとメソッドを操作する 17-5
VI クラスのプロパティとメソッドを操作する 17-6
アプリケーションクラスと VI クラスのプロパティ
およびメソッドを操作する 17-6
VI のダイナミックなロードと呼び出し 17-7
リファレンス呼び出しノード（Call By Reference Node）と
厳密に類別化された VI Refnum 17-7
リモートコンピュータ上での VI の編集と実行 17-8
フロントパネルオブジェクトを制御する 17-9
厳密に類別化された制御器 refnum と
非厳密に類別化された制御器 refnum 17-9

第 18 章
LabVIEW のネットワーク動作

ファイル I/O、VI サーバ、ActiveX、およびネットワーク動作から選択する 18-1
ネットワークのクライアントおよびサーバとしての LabVIEW 18-2
DataSocket テクノロジを使用する 18-2
URL を指定する 18-3
DataSocket でサポートされているデータ形式 18-5
フロントパネル上で DataSocket を使用する 18-5
ブロックダイアグラムでライブデータの読み書きを行う 18-6
DataSocket 接続をプログラムで開閉する 18-7
DataSocket データをバッファし処理する 18-8
診断を報告する 18-9
DataSocket とバリアントデータ 18-9
ウェブ上に VI をパブリッシュする 18-11
ウェブサーバのオプション 18-11
HTML ドキュメントを作成する 18-11
フロントパネル画像をパブリッシュする 18-12
フロントパネルの画像形式 18-12
フロントパネルをリモートで参照および制御する 18-12
クライアント用のサーバを構成する 18-13
リモートパネルライセンス 18-13
LabVIEW またはウェブブラウザからフロントパネルを参照および制御する 18-14
フロントパネルを LabVIEW で参照および制御する 18-14
ウェブブラウザからフロントパネルを参照および制御する 18-15
リモートフロントパネルの参照および制御でサポートされていない機能 18-16
VI からデータを電子メールで送信する 18-17
文字セットを選択する 18-18
US-ASCII 文字セット 18-18
ISO Latin-1 文字セット 18-18
Mac OS 文字セット 18-19
字訳 18-19
低レベル通信アプリケーション 18-20
TCP と UDP 18-20
Apple Events および PPC Toolbox (Mac OS) 18-21
Pipe VI（UNIX） 18-21
システムレベルのコマンドを実行する（Windows および UNIX） 18-21

第 19 章
Windows の接続性

.NET 環境 19-2
.NET 関数およびノード 19-3
.NET クライアントとしての LabVIEW 19-4
データタイプマッピング 19-5
.NET アプリケーションを導入する 19-5
実行ファイルを導入する 19-5
VI を導入する 19-6
DLL を導入する 19-6
.NET クライアントアプリケーションを構成する（上級） 19-6
ActiveX のオブジェクト、プロパティ、メソッド、およびイベント 19-6
ActiveX VI、関数、制御器、および表示器 19-7
ActiveX クライアントとしての LabVIEW 19-8
ActiveX 有効のアプリケーションにアクセスする 19-8
フロントパネルに ActiveX オブジェクトを挿入する 19-9
ActiveX オブジェクトの設計モード 19-9
ActiveX プロパティを設定する 19-10
ActiveX プロパティブラウザ 19-10
ActiveX プロパティページ 19-10
プロパティノード 19-10
ActiveX サーバとしての LabVIEW 19-11
カスタム ActiveX オートメーションインタフェースのサポート 19-12
定数を使用して ActiveX VI のパラメータを設定する 19-12
ActiveX イベント 19-13
ActiveX イベントを処理する 19-14

第 20 章
テキストベースのプログラミング言語からのコード呼び出し

ライブラリ関数呼び出しノード 20-1
コードインタフェースノード 20-1

第 21 章
フォーミュラと方程式

LabVIEW で方程式を使用する方法 21-1
フォーミュラノード（Formula Nodes） 21-2
「フォーミュラノード」を使用する 21-2
「フォーミュラノード」の変数 21-3
数式ノード（Expression Nodes） 21-3
「数式ノード」の多形性 21-4
MATLAB スクリプトノード（MATLAB Script Nodes） 21-4
MATLAB スクリプトについてのプログラム上の提案 21-6

付録 A
LabVIEW の構成

LabVIEW のディレクトリストラクチャの構成 A-1
ライブラリ A-1
構造とサポート A-2
実習と手順 A-2
マニュアル A-2
Mac OS A-2
ファイル保存の推奨場所 A-3

付録 B
多形性関数

数値変換 B-1
数値関数の多形性 B-2
ブール関数の多形性 B-4
配列関数の多形性 B-4
文字列関数の多形性 B-5
文字列変換関数の多形性 B-5
その他の文字列→数値変換関数の多形性 B-5
クラスタ関数の多形性 B-5
比較関数の多形性 B-6
対数関数の多形性 B-7

付録 C
比較関数

ブール値を比較する C-1
文字列を比較する C-1
数値を比較する C-1
配列とクラスタを比較する C-2
配列 C-2
要素の比較モード C-2
基礎群の比較モード C-3
クラスタ C-3
要素の比較モード C-3
基礎群の比較モード C-3

付録 D
技術サポートおよびサービス

用語

索引

本書について

本書では、テストや計測、データ集録、計測器制御、データロギング、データ解析、レポート生成などのアプリケーションを LabVIEW で作成するための、LabVIEW のグラフィカルなプログラミング環境とその技術について説明します。

本書では、LabVIEW ユーザインタフェース、プログラミング作業スペース、さらに LabVIEW のパレット、ツール、ダイアログボックスなどの LabVIEW プログラミング機能について説明します。ただし、各パレット、ツール、メニュー、ダイアログボックス、制御器、標準 VI、標準関数などの詳細な情報については説明しません。これらの項目の詳細および LabVIEW の機能を使用して特定のアプリケーションを作成するための詳しい手順については、『LabVIEW ヘルプ』を参照してください。『LabVIEW ヘルプ』の詳細については、第 1 章「LabVIEW の概要」の「LabVIEW の関連資料」のセクションを参照してください。

『LabVIEW ユーザマニュアル』は、PDF でも入手可能です。**完全**インストールオプションを選択すると、LabVIEW はすべての LabVIEW マニュアルの PDF バージョンをインストールします。それらのマニュアルは、LabVIEW の中から、**ヘルプ→ LabVIEW ドキュメントライブラリを検索**を選択して表示することができます。

メモ PDF を表示するには、Adobe Acrobat Reader がインストールされている必要があります。Acrobat Reader をダウンロードするには、アドビシステムズ社のホームページ `www.adobe.com` にアクセスしてください。

『LabVIEW ヘルプ』からも PDF にアクセスできますが、そのためには PDF をインストールする必要があります。『LabVIEW ドキュメントライブラリ』の PDF にアクセスする方法の詳細については、第 1 章「LabVIEW の概要」の「LabVIEW の関連資料」のセクションを参照してください。

本書の構成

『LabVIEW ユーザマニュアル』は 2 部構成になっています。第 I 部、「LabVIEW の概念」では、LabVIEW でアプリケーションを作成するためのプログラミング概念について説明します。第 I 部では、LabVIEW のプログラミング環境について説明し、アプリケーションの作成の計画を立てるのに役立つ情報を提供します。

第 II 部、「VI の作成と編集」では、特定の方法でのアプリケーション操作を可能にするために使用する LabVIEW の機能、VI、および関数について説明します。第 II 部では、LabVIEW の各機能の使用法や、VI および関数の各クラスの概要について説明します。

表記規則

本書では以下の表記規則を使用します。

→ →記号に沿って、入れ子のメニュー項目やダイアログボックスをたどっていくと、最終的に必要な操作を実行することができます。**ファイル→ページ設定→オプション**という順になっている場合、まず**ファイル**メニューをプルダウンし、次に**ページ設定**項目を選択して、最後のダイアログボックスから**オプション**を選択します。

このアイコンは、ユーザへのアドバイスを表しています。

このアイコンは、注意すべき重要な情報があることを示しています。

このアイコンは、人体への損傷、データ損失、システムのクラッシュなどを防止するための注意事項があることを示しています。

太字 太字のテキストは、メニュー項目やダイアログボックスオプションなど、ソフトウェアでユーザが選択またはクリックする必要がある項目を表します。また、フロントパネル上のパラメータ名、制御器やボタン、ダイアログボックスまたはその一部、メニュー名、パレット名も表します。

下線 下線つきのテキストは、変数、強調、相互参照、または重要な概念の説明を示します。また、ユーザが入力する必要がある語または値のプレースホルダを示します。

斜体 このフォントスタイルは変数を示します。または、ユーザが入力する必要がある語または値のプレースホルダを示します。

`monospace`	このフォントのテキストは、キーボードから入力する必要があるテキストや文字、コードの一部、プログラムサンプル、構文例を表します。また、ディスクドライブ名、パス名、ディレクトリ名、プログラム名、サブプログラム名、サブルーチン名、デバイス名、関数名、演算名、変数名、ファイル名と拡張子、引用するコードにも使用します。
`monospace` の太字	このフォントの太字テキストは、画面に自動印刷されるメッセージや応答を示します。また、他のサンプルとは異なるコードラインを強調する場合にも使用します。
`monospace` の下線	このフォントの下線つきのテキストは、ユーザが入力する必要がある語または値のプレースホルダを示します。
プラットフォーム	このフォントのテキストは特定のプラットフォームを示し、そこに記載された説明はそのプラットフォームにのみ適用されます。
右クリック	**(Mac OS)** 右クリックと同じ操作を実行するには、<Command> を押したままクリックします。

第 I 部

LabVIEW の概念

第 I 部では、LabVIEW でアプリケーションを作成するためのプログラミング概念について説明します。各章では LabVIEW プログラミング環境について説明し、アプリケーションの計画に役立つ情報を提供します。

第 I 部「LabVIEW の概念」は以下の章で構成されています。

- 第 1 章 「LabVIEW の概要」では、LabVIEW とその広範なマニュアル、VI の設計および作成に役立つツールについて説明します。
- 第 2 章「バーチャルインスツルメンツ（仮想計測器）VI の概要」では、仮想計測器 (バーチャルインスツルメンツ) または VI の構成要素について説明します。
- 第 3 章「LabVIEW 環境」では、VI のフロントパネルおよびブロックダイアグラムの作成に使用する LabVIEW のパレット、ツール、およびメニューについて説明します。この章では、LabVIEW のパレットをカスタマイズする方法やいくつかの動作環境オプションを設定する方法についても説明します。
- 第 4 章「フロントパネルを作成する」では、VI のフロントパネルの作成方法について説明します。
- 第 5 章「ブロックダイアグラムを作成する」では、VI のブロックダイアグラムの作成方法について説明します。
- 第 6 章「VI の実行とデバッグ」では、VI の実行について構成する方法や、ブロックダイアグラムの構成またはブロックダイアグラムで渡されるデータに関する問題を識別する方法について説明します。
- 第 7 章「VI およびサブ VI を作成する」では、VI およびサブ VI の作成方法、VI の配布方法、およびスタンドアロンアプリケーションと共有ライブラリの作成方法について説明します。

1

LabVIEW の概要

LabVIEW は、テキスト行ではなくアイコンを使用してアプリケーションを作成するグラフィカルなプログラミング言語です。命令でプログラムを実行するテキストベースのプログラミング言語とは異なり、LabVIEW では、データの流れによってプログラムを実行するデータフロープログラミングを使用します。

LabVIEW では、一連のツールやオブジェクトを使用してユーザインタフェースを作成します。ユーザインタフェースをフロントパネルと呼びます。フロントパネルのオブジェクトを制御するには、そしてグラフィカルに表現された関数を使用してコードを追加します。このコードは、ブロックダイアグラムに含まれています。ブロックダイアグラムは、いくつかの点でフローチャートに似ています。

特殊なアプリケーションを開発する場合は、数種類のアドオンソフトウェアツールセットを購入できます。すべてのツールセットはシームレスに LabVIEW に統合されます。これらのツールセットの詳細については、ナショナルインスツルメンツのホームページ `ni.com/jp` を参照してください。

LabVIEW の関連資料

LabVIEW には、初心者と上級者を対象にしたさまざまな資料が用意されています。すべての LabVIEW マニュアルおよびアプリケーションノートは PDF でもご利用いただけます。PDF を表示するには、Adobe Acrobat Reader がインストールされている必要があります。Acrobat Reader をダウンロードするには、アドビシステムズ社のホームページ `www.adobe.com` にアクセスしてください。最新の関連資料については、ナショナルインスツルメンツの製品マニュアルのページ `ni.com/jp/manuals` を参照してください。

- **『LabVIEW ドキュメントライブラリ』**この PDF ファイルを使用すると、すべての LabVIEW マニュアルおよびアプリケーションノート（PDF）を検索できます。『LabVIEW ドキュメントライブラリ』にアクセスするには、**ヘルプ→ LabVIEW ドキュメントライブラリを検索**を選択します。
- **『LabVIEW 入門』**このマニュアルでは、LabVIEW のグラフィカルプログラミング環境と、データ集録および計測制御アプリケーションの作成に使用する LabVIEW の基本機能について説明します。

- **『LabVIEW クイックリファレンスカード』**このカードを使用して、ヘルプリソース、キーボードショートカット、端子データタイプ、編集、実行、およびデバックツールについての情報を参照できます。
- **『LabVIEW ユーザマニュアル』**このマニュアルでは、テスト、計測、データ集録、計測器制御、データロギング、データ解析、およびレポート生成アプリケーションを作成するための LabVIEW プログラミングの概念、テクニック、機能、VI、および関数について説明します。
- **『LabVIEW ヘルプ』**このヘルプファイルでは、LabVIEW のパレット、メニュー、ツール、VI、および関数について説明します。また、『LabVIEW ヘルプ』では、LabVIEW の機能の使用方法を順を追って説明します。『LabVIEW ヘルプ』にアクセスするには、**ヘルプ→オンラインヘルプレファレンス**を選択します。

  『LabVIEW ヘルプ』には、以下のリソースへのリンクが含まれています。

  – 『LabVIEW ドキュメントライブラリ』すべての LabVIEW マニュアルおよびアプリケーションノート（PDF）が含まれています。

  – ナショナルインスツルメンツのホームページには、NI Developer Zone、技術サポートデータベース、製品マニュアルライブラリなどの技術サポートのリソースがあります。

メモ　**（Mac OS および UNIX）**ナショナルインスツルメンツでは、『LabVIEW ヘルプ』を表示する際に、Netscape 6.0 または Internet Explorer 5.0 以降をご使用になることをお勧めします。

- **『LabVIEW Measurements Manual』**このマニュアルでは、LabVIEW を使用したデータ集録および計測制御アプリケーションの作成に関する詳細について説明します。LabVIEW を初めて使用する場合はまず、『LabVIEW 入門』および 『LabVIEW ユーザマニュアル』をお読みください。

- **『LabVIEW アプリケーションビルダユーザガイド』**このマニュアルでは、LabVIEW アプリケーションビルダについて詳しく説明します。アプリケーションビルダは、LabVIEW プロフェッショナル開発システムに含まれていますが、別途でも購入いただけます。このマニュアルでは、アプリケーションビルダのインストール手順、アプリケーションのシステム要件、前のバージョンと現在のバージョンの違いについて説明します。また、アプリケーションまたは共有ライブラリに VI を組み込む際の注意とアドバイスについても説明しています。

- **『LabVIEW Development Guidelines』**このマニュアルでは、容易に理解、使用、修正できる VI の作成方法を説明します。このマニュアルでは、プロジェクトトラッキング、設計、および文書化のテクニックについて説明します。また、このマニュアルには推奨されるスタイルのガイドラインも記載されています。

メモ　『LabVIEW Development Guidelines』マニュアル（印刷版）は、LabVIEW プロフェッショナル開発システムのみを対象としたマニュアルです。PDF は LabVIEW のすべてのパッケージに含まれています。

- **『LabVIEW Analysis Concepts』**このマニュアルでは、LabVIEW で使用される解析の概念について説明します。このマニュアルには、信号生成、高速フーリエ変換（FFT）および離散フーリエ変換（DFT）、スムージングウィンドウ、カーブフィット、線形代数、確率と統計の基本的概念、およびリアルタイム解析におけるポイントバイポイント解析に関する情報が記載されています。

メモ　『LabVIEW Analysis Concepts』は PDF でのみご利用いただけます。

- **『Using External Code in LabVIEW』**このマニュアルでは、コードインタフェースノード（Code Interface Nodes）および外部サブルーチンを使用して、テキストベースのプログラム言語で作成されたコードをインポートする方法を説明します。DLL を呼び出す方法、共有外部サブルーチン、関数ライブラリ、メモリとファイル操作ルーチン、および診断ルーチンについて説明します。

メモ　『Using External Code in LabVIEW』マニュアルは PDF でのみご利用いただけます。

- **『LabVIEW リリースノート』**LabVIEW をインストールおよびアンインストールする方法を説明します。リリースノートには、LabVIEW ソフトウェアのシステム要件と既知の問題点も記載されています。
- **『LabVIEW アップグレードノート』**Windows、Mac OS、および UNIX 対応の LabVIEW を最近バージョンにアップグレードするための情報が記載されています。また、新機能やアップグレードに際して考えられる問題点についても説明されています。
- **『LabVIEW アプリケーションノート』**LabVIEW の上級または特別な概念およびアプリケーションを説明します。最新のアプリケーションノートについては、`ni.com/zone` を参照してください。
- **『LabVIEW VXI VI Reference Manual』**このマニュアルでは、LabVIEW の VXI VI について説明します。このマニュアルは、VXI ハードウェアに付属の『NI-VXI Programmer Reference Manual』

とセットになっています。ナショナルインスツルメンツでは、VXI ハードウェアから成る計測システムを構成、プログラミング、トラブルシューティングする際には VISA のテクノロジーを使用することを推奨します。

メモ　『LabVIEW VXI VI Reference Manual』は PDF でのみご利用いただけます。

LabVIEW のサンプル VI およびツール

LabVIEW のサンプル VI およびツールは、VI の設計と作成に役立ちます。

LabVIEW のテンプレート VI

LabVIEW テンプレート VI には、一般の計測アプリケーションの作成開始に必要なサブ VI、関数、ストラクチャ、およびフロントパネルのオブジェクトが含まれています。テンプレート VI は、保存する必要がある名称未設定の VI として開きます。**ファイル→新規**を選択して、テンプレート**新規**ダイアログボックスを表示します。また、**LabVIEW** ダイアログボックス上の**新規**ボタンをクリックして、**新規**ダイアログボックスを表示することもできます。

LabVIEW のサンプル VI

LabVIEW には数百ものサンプル VI があり、それらを使用したり、独自に VI に組み込むことができます。ユーザのアプリケーションに合うようにサンプルを変更したり、1 つまたは複数のサンプルを独自の VI にコピーして貼り付けることもできます。**ヘルプ→サンプルの検索**を選択して、サンプル VI を参照または検索します。追加のサンプル VI については、NI Developer Zone（`ni.com/zone`）を参照してください。

LabVIEW のツール

LabVIEW には、測定デバイスの迅速な構成に役立つ数多くのツールがあります。これらのツールは、**ツール**メニューからアクセスできます。

- **（Windows）** Measurement & Automation Explorer (MAX) を使用して、ナショナルインスツルメンツのハードウェアおよびソフトウェアを構成します。
- **（Mac OS 9 以前）** NI-DAQ 構成ユーティリティを使用して、ナショナルインスツルメンツの DAQ ハードウェアを構成します。
- **（Mac OS 9 以前）** DAQ チャネルウィザードを使用して、DAQ ハードウェアチャネルに接続されているデバイスのタイプを定義します。チャネルを定義すると、DAQ チャネルウィザードにその設定が保存されます。

- **（Mac OS 9 以前）**DAQ チャネルビューワには、構成されている DAQ チャネルがリストされます。
- **（Mac OS 9 以前）**DAQ ソリューションウィザードを使用すると、一般的な DAQ アプリケーションのソリューションを検索できます。サンプル VI を選択したり、カスタム VI を作成できます。

DAQ アシスタントを使用して、チャネルまたは一般的な測定タスクをグラフィカルに構成します。以下の方法で DAQ アシスタントにアクセスすることができます。

- ブロックダイアグラム上に「DAQ アシスタント（DAQ Assistant）」Express VI を配置します。
- DAQmx グローバルチャネル制御器を右クリックし、ショートカットメニューから**新規チャネル（DAQ アシスタント）**を選択します。DAQmx タスク名制御器を右クリックし、ショートカットメニューから**新規タスク（DAQ アシスタント）**を選択します。DAQmx スケール名制御器を右クリックし、ショートカットメニューから**新規スケール（DAQ アシスタント）**を選択します。
- Measurement & Automation Explorer を起動して、**構成**ツリーから**データ設定**または**スケール**を選択します。**新規作成**ボタンをクリックします。NI-DAQmx チャネル、タスク、またはスケールを構成します。

DAQ アシスタントの詳細については、『LabVIEW Measurements Manual』を参照してください。

2

バーチャルインスツルメンツ（仮想計測器）VI の概要

LabVIEW プログラムは、その外観と操作がオシロスコープやマルチメータなどの実際の計測器に似ているため、バーチャルインスツルメンツ（仮想計測器）、つまり VI と呼ばれれています。各 VI は、ユーザインタフェースや他のソースからの入力を操作して情報を表示したり、他のファイルやコンピュータに情報を移動します。

VI には、以下の 3 つの要素があります。

- **フロントパネル**：ユーザインタフェースとして機能します。
- **ブロックダイアグラム**：VI の機能を定義するグラフィカルなソースコードを含んでいます。
- **アイコンとコネクタペーン**：VI を別の VI の中で使用できるように識別します。VI が別の VI の中にある場合、その VI はサブ VI と呼ばれます。サブ VI は、テキストベースのプログラミング言語のサブルーチンに相当します。

詳細については

VI およびサブ VI の作成の詳細については、『LabVIEW ヘルプ』を参照してください。

フロントパネル

フロントパネルは VI のユーザインタフェースです。図 2-1 はフロントパネルの例を示しています。

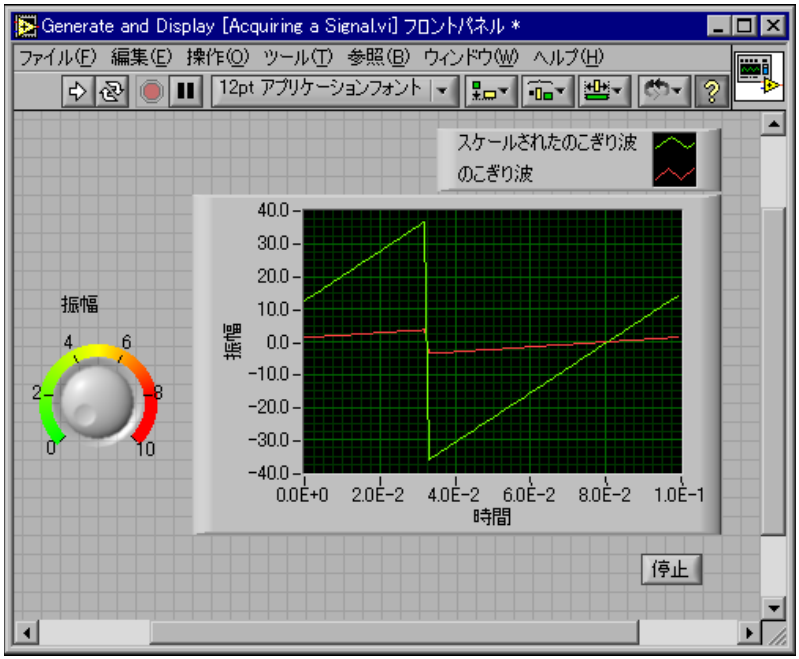


図 2-1　フロントパネルの例

VI の対話形式の入力および出力端子である制御器と表示器を用いて、フロントパネルを作成します。制御器には、ノブ、押しボタン、ダイヤル、その他の入力デバイスがあります。表示器には、グラフ、LED、その他のディスプレイがあります。制御器は計測器入力デバイスをシミュレーションしたもので、VI のブロックダイアグラムにデータを供給します。表示器は計測器出力デバイスをシミュレーションしたもので、ブロックダイアグラムが集録または生成するデータを表示します。フロントパネルの詳細については、第 4 章「フロントパネルを作成する」を参照してください。

ブロックダイアグラム

フロントパネルを作成したら、グラフィカルに表現された関数を使用してコードを追加し、フロントパネルのオブジェクトを制御します。ブロックダイアグラムには、グラフィカルソースコードが含まれています。フロントパネルオブジェクトは、ブロックダイアグラムでは端子として表示されます。ブロックダイアグラムの詳細については、第 5 章「ブロックダイアグラムを作成する」を参照してください。

図 2-2 の VI は、簡単なブロックダイアグラムオブジェクト（端子、関数、ワイヤ）を示しています。

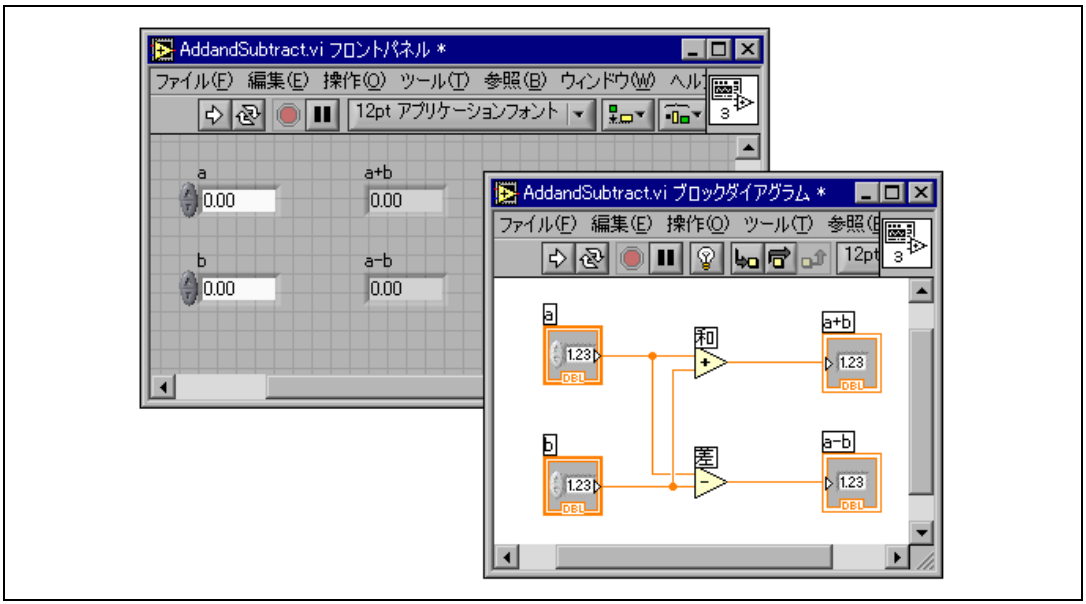


図 2-2　ブロックダイアグラムと対応するフロントパネルの例

端子

端子は制御器または表示器のデータタイプを表します。フロントパネルの制御器または表示器を構成して、アイコンまたはデータタイプ端子としてブロックダイアグラムに表示できます。デフォルトでは、フロントパネルオブジェクトはアイコン端子として表示されます。たとえば、左図に示すノブアイコン端子はフロントパネルのノブを表します。端子の下にある DBL は、倍精度浮動小数点数のデータタイプを表します。左図に示す DBL 端子は、倍精度浮動小数点数の制御器または表示器を表します。LabVIEW のデータタイプとそのグラフィカル表現の詳細については、第 5 章「ブロックダイアグラムを作成する」の「制御器および表示器のデータタイプ」のセクションを参照してください。

端子は、フロントパネルとブロックダイアグラムの間で情報を交換する入出力ポートです。フロントパネルの制御器に入力したデータ（図 2-2 の **a** と **b**）は、制御器端子を介してブロックダイアグラムに入力されます。次に、このデータは「和（Add)」関数および「差（Subtract)」関数に渡されます。「和」関数および「差」関数で内部計算が終了すると新しいデータ値が生成されます。そのデータは表示器端子（図 2-2 の **a+b** と **a-b**）へ渡り、ブロックダイアグラムを出て、フロントパネルにもう一度入り、フロントパネルの表示器に表示されます。

ノード

ノードは、入力端子や出力端子を持ち、VI の実行時に演算を実行するブロックダイアグラム上のオブジェクトです。テキストベースのプログラミング言語におけるステートメント、演算子、関数、およびサブルーチンに似ています。図 2-2 では、「和（Add）」関数および「差（Subtract）」関数がノードです。ノードの詳細については、第 5 章「ブロックダイアグラムを作成する」の「ブロックダイアグラムのノード」のセクションを参照してください。

ワイヤ

ブロックダイアグラムオブジェクト間のデータ転送はワイヤを介して行います。図 2-2 では、制御器および表示器の端子と「和」関数および「差」関数がワイヤで接続されています。各ワイヤのデータソースは 2 つですが、そのデータを読み取る多くの VI および関数に配線できます。ワイヤの色、種類、および太さは、データタイプによって異なります。壊れたワイヤは、真中に赤い x がある黒い破線として表示されます。ワイヤの詳細については、第 5 章「ブロックダイアグラムを作成する」の「ブロックダイアグラムオブジェクトをワイヤで接続する」のセクションを参照してください。

ストラクチャ

ストラクチャは、テキストベースのプログラミング言語のループおよびケースステートメントのグラフィカル表現です。コードのブロックを繰り返したり、条件付きでコードを実行したり、特定の順序でコードを実行するには、ブロックダイアグラムでストラクチャを使用します。ストラクチャの詳細とサンプルについては、第 8 章「ループとストラクチャ」を参照してください。

アイコンとコネクタペーン

VI フロントパネルとブロックダイアグラムを作成した後でアイコンとコネクタペーンを作成すると、その VI をサブ VI として使用できます。各 VI では、フロントパネルおよびブロックダイアグラムウィンドウの右上隅に、左に示すようなアイコンが表示されます。アイコンは VI を図で表現したものです。アイコンはテキスト、画像、またはその両方の組み合わせです。VI をサブ VI として使用する場合、そのアイコンによってその VI のブロックダイアグラム上のサブ VI が識別されます。アイコンは、ダブルクリックすると、カスタマイズまたは編集することができます。アイコンの詳細については、第 7 章「VI およびサブ VI を作成する」の「アイコンを作成する」のセクションを参照してください。

ある VI をサブ VI として使用するには、左図のようなコネクタペーンを作成する必要があります。コネクタペーンは VI の制御器と表示器に対応する一連の端子であり、テキストベースのプログラミング言語の関数呼び出しのパラメータリストに似ています。コネクタペーンは、VI に配線できる入力および出力を定義するため、サブ VI として使用することができます。コネクタペーンは、その入力端子で受け取ったデータをフロントパネルの制御器を介してブロックダイアグラムのコードに渡し、フロントパネルの表示器の結果を出力端子で受け取ります。

コネクタペーンを初めて表示すると、コネクタパターンが現れます。希望に応じて異なるパターンを選択することができます。一般に、コネクタペーンにはフロントパネルの各制御器または各表示器に 1 つの端子があります。コネクタペーンには最大 28 個の端子を割り当てることができます。VI に変更が加えられて新しい入力端子や出力端子が必要となることが予測される場合は、割り当てられていない余分な端子を残しておきます。コネクタペーンのセットアップの詳細については、第 7 章「VI およびサブ VI を作成する」の「コネクタペーンを設定する」のセクションを参照してください。

メモ　1 つの VI に 17 個以上の端子を割り当てないように注意してください。端子が多すぎると、VI が煩雑でわかりにくくなる可能性があります。

VI およびサブ VI の使用とカスタマイズ

VI を作成してそのアイコンおよびコネクタペーンを作成すると、その VI をサブ VI として使用できます。サブ VI の詳細については、第 7 章「VI およびサブ VI を作成する」の「サブ VI」のセクションを参照してください。

VI を個々のファイルとして保存したり、複数の VI をグループ化して VI ライブラリに保存することができます。ライブラリに VI を保存する方法の詳細については、第 7 章「VI およびサブ VI を作成する」の「VI を保存する」のセクションを参照してください。

VI の外観と動作をカスタマイズすることもできます。作成するすべての VI に対してカスタムメニューを作成したり、VI のメニューバーの表示 / 非表示を設定することもできます。VI のカスタマイズの詳細については、第 16 章「VI をカスタマイズする」を参照してください。

3

LabVIEW 環境

LabVIEW のパレット、ツール、メニューを使用して、VI のフロントパネルおよびブロックダイアグラムを作成します。**制御器**パレットと**関数**パレットをカスタマイズしたり、複数の作業環境オプションを設定したりすることもできます。

詳細については

パレット、メニュー、ツールバーの使用と作業環境のカスタマイズの詳細については、『LabVIEW ヘルプ』を参照してください。

制御器パレット

制御器パレットはフロントパネルのみで使用できます。**制御器**パレットには、フロントパネルの作成に使用する制御器や表示器が含まれます。制御器と表示器はそのタイプごとにまとめられたサブパレットにあります。制御器および表示器の詳細については、第 4 章「フロントパネルを作成する」の「スライド制御器および表示器」のセクションを参照してください。**制御器**パレットにある制御器および表示器は、現在選択されているパレットセットによって異なります。パレットセットの詳細については、この章の「カスタムパレットセットの作成と編集」のセクションを参照してください。

制御器パレットを表示するには、**ウィンドウ→制御器パレットを表示**を選択するか、フロントパネルの作業スペースで右クリックします。**制御器**パレットは画面上の任意の場所に配置できます。**制御器**パレットの位置およびサイズは保持されるため、LabVIEW を再起動してもパレットの位置およびサイズは変更されずそのまま表示されます。

制御器パレットの表示方法は変更することができます。**制御器**パレットのカスタマイズの詳細については、この章の「制御器および関数パレットをカスタマイズする」のセクションを参照してください。

関数パレット

関数パレットはブロックダイアグラムのみで使用できます。**関数**パレットには、ブロックダイアグラムを作成するのに使用する VI や関数が含まれています。VI と関数はそのタイプごとにまとめられたサブパレットに入っています。VI と関数のタイプの詳細については、第 5 章「ブロックダイアグラムを作成する」の「多形性 VI を作成する」のセクションを参照してください。**関数**パレットにある制御器および表示器は、現在選択されているパレットセットによって異なります。パレットセットの詳細については、この章の「カスタムパレットセットの作成と編集」のセクションを参照してください。

関数パレットを表示するには、**ウィンドウ→関数パレットを表示**を選択するか、ブロックダイアグラムの作業スペースで右クリックします。**関数**パレットは画面上の任意の場所に配置できます。**関数**パレットの位置とサイズは保持されるため、LabVIEW を再起動してもパレットの位置とサイズは変更されずそのまま表示されます。

関数パレットの表示方法は変更することができます。**関数**パレットのカスタマイズの詳細については、この章の「制御器および関数パレットをカスタマイズする」のセクションを参照してください。

制御器パレットおよび関数パレットを操作する

サブパレットアイコンをクリックすると、パレット全体が、選択したサブパレットに変更されます。パレット上のオブジェクトをクリックしてカーソルでオブジェクトをつかむと、オブジェクトをフロントパネルまたはブロックダイアグラムに配置することができます。また、パレットで VI アイコンを右クリックし、ショートカットメニューから **VI を開く**を選択して VI を開くこともできます。

制御器および**関数**パレットツールバーにある以下のボタンを使用して、パレット内での移動、パレットの構成、および制御器、VI、関数の検索を行うことができます。

- **上**：パレット階層の 1 つ上のレベルに移動します。このボタンをクリックしてマウスボタンを押し続けると、現在のサブパレットへのパスに各サブパレットの一覧を表示するショートカットメニューが表示されます。ショートカットメニューでサブパレット名を選択して、サブパレットへ移動します。

- **検索**：パレットが検索モードになり、テキストベースの検索を実行してパレット上の制御器、VI、または関数を検索できるようになります。パレットが検索モードになっているときは、**戻る**ボタンをクリックすると検索モードが終了してパレットに戻ります。

- **オプション**：**オプション**ダイアログボックスの**制御器／関数パレット**ページを表示します。ここで、パレットセットおよびパレット形式をできます。

- **パレットサイズを復元**：パレットをデフォルトのサイズに戻します。このボタンは、**制御器**または**関数**パレットのサイズを変更した場合にのみ表示されます。

ツールパレット

ツールパレットはフロントパネルおよびブロックダイアグラムで使用できます。ツールとは、マウスカーソルの特殊な操作モードです。カーソルは、パレットで選択されたツールのアイコンに対応します。ツールを使用して、フロントパネルやブロックダイアグラムのオブジェクトを操作したり変更したりします。

ツールパレットを表示するには、**ウィンドウ→ツールパレットを表示**を選択します。**ツール**パレットは画面上の任意の場所に配置できます。**ツール**パレットの位置は保持されるため、LabVIEW を再起動してもパレットは同じ位置に表示されます。

ヒント　**ツール**パレットをカーソル位置に一時的に表示するには、<Shift> キーを押したまま右クリックします。

ツールの自動選択が有効になっているときにフロントパネルまたはブロックダイアグラムのオブジェクト上にカーソルを移動すると、LabVIEW は**ツール**パレットから自動的に対応するツールを選択します。**ツール**パレットにある左図のような**自動ツール選択**ボタンをクリックすると、自動ツール選択を無効にすることができます。自動ツール選択をもう一度有効にするには、<Shift-Tab> キーを押すか、**自動ツール選択**ボタンをクリックします。また、**ツール**パレットにあるツールを手動で選択することによって、自動ツール選択を無効にすることもできます。<Tab> キーを押すか、**ツール**パレットにある**自動ツール選択**ボタンをクリックすれば、自動ツール選択をもう一度有効にできます。

メニューとツールバー

メニューやツールバーの項目を使用して、フロントパネルやブロックダイアグラムのオブジェクトを操作したり変更することができます。VI の実行には、ツールバーボタンを使用します。

メニュー

VI ウィンドウの一番上にあるメニューには、他のアプリケーションに共通の**開く**、**保存**、**コピー**、**貼り付け**などの他に、LabVIEW 固有の項目があります。ショートカットキーも併せて表示されるメニューもあります。

（Mac OS）メニューは画面の一番上に表示されます。

（Windows および UNIX）デフォルトでは、最近使用した項目のみがメニューに表示されます。メニューの下の矢印をクリックしてすべての項目を表示します。**ツール→オプション**を選択して、プルダウンメニューから**その他**を選択すると、デフォルトですべてのメニュー項目を表示できます。

メモ　VI が実行モードのときは使用できないメニュー項目もあります。

ショートカットメニュー

最も頻繁に使用されるメニューはオブジェクトショートカットメニューです。LabVIEW のすべてのオブジェクトと、フロントパネルおよびブロックダイアグラム上の空白部分は、ショートカットメニューに関連付けられています。ショートカットメニュー項目を使用して、フロントパネルおよびブロックダイアグラムオブジェクトの外観や動作を変更します。ショートカットメニューにアクセスするには、オブジェクト、フロントパネル、またはブロックダイアグラムを右クリックします。

実行モード時のショートカットメニュー

VI の実行時または実行モードでは、すべてのフロントパネルオブジェクトにショートカットメニューの短縮版があります。オブジェクトの内容の切り取り、コピー、貼り付けを行ったり、オブジェクトの設定をデフォルトに戻したり、オブジェクトの説明を読み取るには、このショートカットメニューの短縮版を使用します。

追加オプションがついた複雑な制御器もあります。たとえば、配列のショートカットメニューには、値の範囲をコピーしたり、最後の配列要素を表示する項目があります。

ツールバー

ツールバーボタンを使用して、VI を実行したり編集します。VI を実行すると、VI のデバッグに使用できるボタンがツールバーに表示されます。

ヘルプウィンドウ

ヘルプウィンドウには、各オブジェクトの上にカーソルを移動したときに、LabVIEW オブジェクトに関する基本情報が表示されます。ヘルプ情報のあるオブジェクトには、VI、関数、定数、ストラクチャ、パレット、プロパティ、メソッド、イベント、およびダイアログボックスなどがあります。**ヘルプ**ウィンドウを使用して、VI または関数のどこにワイヤを接続するかを指定できます。**ヘルプ**ウィンドウを使用したオブジェクトの配線方法の詳細については、第 5 章「ブロックダイアグラムを作成する」の「オブジェクトを手動配線する」のセクションを参照してください。

ヘルプ→ヘルプを表示を選択して、**ヘルプ**ウィンドウを表示します。また、ツールバーにある左図のような**ヘルプウィンドウを表示**ボタンをクリックするか、<Ctrl-H> キーを押すことによって、**ヘルプ**ウィンドウを表示することもできます。**（Mac OS）**<Command-H> キーを押します。**（UNIX）**<Alt-H> キーを押します。

ヘルプウィンドウは画面上の任意の場所に配置できます。**ヘルプ**ウィンドウは、オブジェクトの説明を表示できるようにサイズが変更されます。**ヘルプ**ウィンドウをサイズ変更して、最大サイズに設定することもできます。**ヘルプ**ウィンドウの位置とサイズは保持されるため、LabVIEW を再起動してもパレットの位置とサイズは変更されずそのまま表示されます。

ヘルプウィンドウの現在の内容をロックすると、他のオブジェクト上にカーソルを移動してもヘルプウィンドウの内容は変更されません。**ヘルプ→ヘルプをロック**を選択して、**ヘルプ**ウィンドウの現在の内容をロックまたはロック解除します。また、**ヘルプ**ウィンドウにある左図のような**ロック**ボタンをクリックするか、<Ctrl-Shift-L> キーを押すことによって、ウィンドウの内容をロックまたはロック解除することもできます。
（Mac OS）<Command-Shift-L> キーを押します。**（UNIX）**<Alt-Shift-L> キーを押し
ます。

ヘルプウィンドウで、左図のような**オプションの端子と完全パスを表示**ボタンをクリックして、コネクタペーンのオプションの端子と VI の完全パスを表示します。オプションの端子の詳細については、第 7 章「VI およびサブ VI を作成する」の「必須、推奨、および任意の入出力を設定する」のセクションを参照してください。

ヘルプウィンドウに記述されているオブジェクトに対応する『LabVIEW ヘルプ』トピックがある場合、**ヘルプ**ウィンドウにヘルプを表示するには、ここをクリックが青いリンクで表示されます。また、**ヘルプ**ウィンドウにある左図のような**詳細なヘルプ**ボタンが有効になります。リンクまたはボタンをクリックして、オブジェクトの詳細についての LabVIEW ヘルプを表示します。

説明を作成して**ヘルプ**ウィンドウに表示する方法の詳細については、第 15 章「VI の文書化と印刷」の「VI およびオブジェクトの説明を作成する」のセクションを参照してください。

作業環境をカスタマイズする

制御器および関数パレットをカスタマイズしたり、オプションダイアログボックスを使用してパレットセットを選択したり他の作業環境オプションを設定することができます。

制御器および関数パレットをカスタマイズする

制御器および**関数**パレットは、以下の方法でカスタマイズできます。

- パレットに VI および制御器を追加する。
- ユーザごとに異なるセットを設定する。たとえば、LabVIEW を使用しやすくするために、あるユーザには一部の VI と関数しか表示せず、他のユーザにはパレットをすべて表示する。
- 頻繁に使用する VI と関数にアクセスしやすくするために、標準パレットを並び替える。
- 一連の ActiveX 制御器をカスタム制御器に変換し、パレットに追加する。
- ツールセットをパレットに追加する。

注意　LabVIEW をアップグレードしたり再インストールしたときに上書きされてしまいますので、`vi.lib` ディレクトリにユーザ独自の VI および制御器を保存しないでください。独自の VI と制御器を**関数**および**制御器**パレットに追加する場合は、`user.lib` ディレクトリに保存してください。

VI と制御器をユーザおよび計測器ドライバサブパレットに追加する

関数および**制御器**パレットに VI や制御器を追加するには、`labview¥user.lib` ディレクトリに保存するのが最も簡単な方法です。LabVIEW を再起動すると、**ユーザライブラリ**および**ユーザ制御器**パレットには各ディレクトリのサブパレット、VI ライブラリ（`.llb`）、または `labview¥user.lib` のメニューファイル（`.mnu`）、および `labview¥user.lib` の各ファイルのアイコンが含まれています。特定のディレクトリにファイルを追加または削除した後に LabVIEW を再起動すると、パレットが自動的に更新されます。

計測器ドライバパレットは、`labview¥instr.lib` ディレクトリに対応します。**関数**に計測器ドライバを追加するには、このディレクトリに保存します。

この方法で**関数**および**制御器**パレットに VI や制御器を追加する場合、各サブパレットの名前や、パレット上での VI または制御器の正確な位置を設定することはできません。

カスタムパレットセットの作成と編集

制御器パレットおよび**関数**パレットに追加する各サブパレットの名前、制御器および VI の位置を正確に制御するには、カスタムパレットセットを作成する必要があります。LabVIEW には 2 つの標準パレットセット、Express および上級があります。**ツール→上級→パレットセットを編集**を選択して、カスタムパレットセットを作成したり編集することができます。

メモ　標準パレットセットを編集することはできません。

制御器および**関数**パレットの情報は、`labview¥menus` ディレクトリに格納されています。`menus` ディレクトリには、ユーザが作成またはインストールした各セットに対応するディレクトリがあります。ネットワーク上で LabVIEW を実行する場合は、ユーザごとに各 `menus` ディレクトリを定義できます。これにより他のユーザに簡単にパレットセットを転送できます。

パレットの新しいセットを作成する際、LabVIEW は元の標準パレットセットのコピーを使用します。ユーザはそのコピーに変更を加えます。LabVIEW は、ユーザが変更を加える前に、`labview¥menus` ディレクトリにある元の標準パレットをコピーします。標準パレットは保護されているので、元のセットを破損することなくパレットの変更を試すことができます。

LabVIEW のパレットセットの格納方法

`.mnu` ファイルおよび `.llb` ファイルには、**制御器**パレットと**関数**パレットが 1 つずつ含まれています。また、各ファイルには**制御器**および**関数**パレットのアイコンが含まれています。作成したサブパレットは、別々の `.mnu` ファイルに保存する必要があります。

セットを選択すると、LabVIEW はそのセットに対応するディレクトリを `menus` ディレクトリ内で検索します。LabVIEW は、セットを作成するたびに自動的に作成されるディレクトリ内の `root.mnu` ファイルから、最上位の**制御器**および**関数**パレットとサブパレットを作成します。

LabVIEW は、各 VI または制御器に対し、パレット上にアイコンを作成します。各サブディレクトリ、`.mnu` ファイル、および `.llb` ファイルに対しては、パレット上にサブパレットを作成します。

ActiveX サブパレットを作成する

フロントパネルで ActiveX 制御器を使用する場合は、**ツール→上級→ ActiveX コントロールをインポート**を選択して ActiveX 制御器をカスタム制御器に変換し、**制御器**パレットに追加します。`user.lib` 内のファイルとディレクトリはすべて自動的にパレットに表示されるため、LabVIEW はデフォルトで制御器を `user.lib` ディレクトリに保存します。

パレットにツールセットとモジュールを表示する

LabVIEW を再起動すると、`vi.lib¥addons` に入っている制御器および VI のあるツールセットおよびモジュールが**制御器**および**関数**パレットに表示されます。標準の Express パレットセットでは、ツールセットおよびモジュールによりサブパレットが**すべての制御器**および**すべての関数**サブパレットにインストールされます。標準の上級パレットセットでは、サブパレットはツールセットおよびモジュールにより**制御器**および**関数**パレットの最上位にインストールされます。

ツールセットまたはモジュール制御器および VI を `vi.lib¥addons` ディレクトリの外にインストールした場合、その制御器や VI を `vi.lib¥addons` ディレクトリに移動してパレットに追加することができます。

作業環境オプションを設定する

LabVIEW をカスタマイズするには**ツール→オプション**を選択します。**オプション**ダイアログボックスを使用して、フロントパネル、ブロックダイアグラム、パス、パフォーマンスおよびディスク、アライメントグリッド、パレット、取り消し回数、デバックツール、色、フォント、印刷、**履歴**ウィンドウ、その他の LabVIEW 機能のオプションを設定します。

オプションダイアログボックスの一番上のプルダウンメニューを使用して、異なるカテゴリのオプションの中から選択します。

LabVIEW のオプションの格納方法

オプションダイアログボックスを使用するので、オプションを手動で編集したり、その正確な形式を知っておく必要はありません。LabVIEW は各プラットフォームごとに異なる方法でオプションを格納します。

Windows

LabVIEW は、LabVIEW ディレクトリの `labview.ini` ファイルにオプションを格納します。ファイル形式は他の `.ini` ファイルと同じです。LabVIEW セクションマーカで始まり、`offscreenUpdates=True` のようにオプション名と値がその後に続きます。

別のオプションファイルを使用する場合は、LabVIEW の起動に使用するショートカット内にそのファイルを指定します。たとえば、labview.ini の代わりにユーザのコンピュータ上の lvrc という名前のオプションファイルを使用するには、デスクトップ上の LabVIEW のアイコンを右クリックして**プロパティ**を選択します。**ショートカット**タブをクリックして**リンク先**テキストボックス内に「labview -pref lvrc」と入力します。

Mac OS

LabVIEW は**システム→初期設定**フォルダ内の LabVIEW Preferences テキストファイルにオプションを格納します。

別のオプションファイルを使用する場合は、LabVIEW Preferences ファイルを **LabVIEW** フォルダにコピーして、**Options** ダイアログボックス内でオプションを変更します。LabVIEW を起動すると、LabVIEW はまず **LabVIEW** フォルダ内でオプションファイルを探します。ファイルが見つからないと、**システム**フォルダを探します。そのフォルダでもファイルが見つからないと、**システム**フォルダ内に新規ファイルを作成します。LabVIEW は、**Options** ダイアログボックス内でのすべての変更内容を、最初に見つけた LabVIEW Preferences ファイルに書き込みます。

UNIX

LabVIEW はユーザのホームディレクトリ内の .labviewrc ファイルにオプションを格納します。**Options** ダイアログボックス内でオプションを変更した場合、LabVIEW は変更内容を .labviewrc ファイルに書き込みます。プログラムディレクトリ内に labviewrc ファイルを作成して、VI 検索パスなど、すべてのユーザに共通のオプションを格納できます。ホームディレクトリ内の .labviewrc ファイルのエントリは、プログラムディレクトリ内の競合するエントリよりも優先されるため、フォントや色の設定など、ユーザごとに異なるオプションを格納するには、.labviewrc ファイルを使用します。

たとえば、¥opt¥labview 内に LabVIEW ファイルをインストールした場合、LabVIEW は最初に ¥opt¥labview¥labviewrc からオプションを読み取ります。**Options** ダイアログボックス内でアプリケーションフォントなどのオプションを変更した場合、LabVIEW は変更内容を .labviewrc ファイルに書き込みます。次に LabVIEW を起動したとき、LabVIEW は、¥opt¥labview¥labviewrc に定義されているデフォルトのアプリケーションフォントの代わりに、.labviewrc ファイル内のアプリケーションフォントオプションを使用します。

オプションエントリは、オプション名、コロン (:)、値の順に構成されます。オプション名は、実行可能形式の後にピリオド (.) とオプションが続きます。LabVIEW は、オプション名を検索するときに大文字と小文字を区別します。オプション値は、二重引用符または単一引用符で囲むことが

できます。たとえば、デフォルトの倍精度を使用するには、以下のエントリをホームディレクトリ内の .labviewrc ファイルに追加します。

```
labview.defPrecision : double
```

別のオプションファイルを使用する場合は、LabVIEW を起動するときにコマンドライン上にそのファイルを指定します。たとえば、.labviewrc の代わりに test ディレクトリ内の lvrc という名前のファイルを使用するには、「labview -pref/test/lvrc」と入力します。LabVIEW は **Options** ダイアログボックス内でのすべての変更内容を lvrc オプションファイルに書き込みます。コマンドライン上でオプションファイルを指定すると、LabVIEW はプログラムディレクトリ内の labviewrc ファイルを読み取りますが、コマンドライン上に指定されたオプションファイルは、プログラムディレクトリ内の競合するエントリよりも優先されます。

4

フロントパネルを作成する

フロントパネルは、VI のユーザインタフェースです。通常、最初にフロントパネルを設計し、次にブロックダイアグラムを設計して、フロントパネル上に作成した入出力のタスクを実行します。ブロックダイアグラムの詳細については、第 5 章「ブロックダイアグラムを作成する」を参照してください。

VI の対話形式の入力および出力である、制御器と表示器を用いてフロントパネルを作成します。制御器とは、ノブ、押しボタン、ダイアル、その他の入力デバイスです。表示器とは、グラフ、LED、その他のディスプレイです。制御器は計測器入力デバイスをシミュレーションするもので、VI のブロックダイアグラムにデータを供給します。表示器は計測器出力デバイスをシミュレーションするもので、ブロックダイアグラムが集録または生成するデータを表示します。

ウィンドウ→制御器パレットを表示を選択して、**制御器**パレットを表示し、次に**制御器**パレットから制御器と表示器を選択して、フロントパネル上に配置します。

詳細については

フロントパネルの設計および構築方法の詳細については、『LabVIEW ヘルプ』を参照してください。

フロントパネルオブジェクトを構成する

プロパティダイアログボックスまたはショートカットメニューを使用して、フロントパネル上での制御器や表示器の表示形式および動作方法を構成します。ヘルプを含むダイアログボックスからフロントパネル制御器や表示器を構成する場合や、オブジェクトの複数のプロパティを一度に設定する場合には、プロパティダイアログボックスを使用します。ショートカットメニューを使用すると、制御器や表示器の一般的なプロパティをすばやく構成することができます。プロパティダイアログボックスとショートカットメニューのオプションは、フロントパネルオブジェクトによって異なります。ショートカットメニューを使用して設定したオプションは、プロパティダイアログボックスを使用して設定したオプションよりも優先されます。カスタム制御器、表示器、およびタイプ定義の作成や使用の詳

細については、『LabVIEW Custom Controls, Indicators, and Type Definitions』アプリケーションノートを参照してください。

フロントパネルで制御器または表示器を右クリックして、ショートカットメニューから**プロパティ**を選択すると、そのオブジェクトのプロパティダイアログボックスが表示されます。制御器や表示器のプロパティダイアログボックスは、VI の実行中は表示することができません。

オプション項目を表示または非表示にする

フロントパネルの制御器と表示器には、ユーザが表示、非表示を決めることができるオプション項目があります。フロントパネルオブジェクトのプロパティダイアログボックスの**外観**タブで、制御器と表示器の表示項目を設定します。オブジェクトを右クリックし、ショートカットメニューから**表示項目**を選択して、表示項目を設定することもできます。ほとんどのオブジェクトにはラベルとキャプションがあります。ラベルとキャプションの詳細については、この章の「ラベルを付ける」のセクションを参照してください。

制御器を表示器に、表示器を制御器に変更する

LabVIEW は、**制御器**パレット内のオブジェクトを、最初は一般的な用途に基づいて制御器または表示器として構成します。たとえば、トグルスイッチを選択すると、トグルスイッチは通常入力デバイスであるため、フロントパネル上に制御器として表示されます。LED を選択すると、LED は通常出力デバイスであるため、フロントパネル上に表示器として表示されます。

一部のパレットには同じタイプまたはクラスのオブジェクト用の制御器と表示器が含まれています。たとえば、**数値**パレットにはデジタル制御器とデジタル表示器が含まれています。

オブジェクトを右クリックしてショートカットメニューから**表示器に変更**を選択すると、制御器を表示器に変更することができます。同様に、オブジェクトを右クリックしてショートカットメニューから**制御器に変更**を選択すると、表示器を制御器に変更することができます。

フロントパネルのオブジェクトを入れ替える

フロントパネルオブジェクトを別の制御器または表示器に入れ替えることができます。オブジェクトを右クリックしてショートカットメニューから**入替え**を選択すると、**制御器**パレットが既に開いている場合でも、一時的に**制御器**パレットが表示されます。一時的な**制御器**パレットから制御器または表示器を選択して、フロントパネル上の現在のオブジェクトと入れ替えます。

ショートカットメニューから**入替え**を選択すると、名前、説明、デフォルトデータ、データフローの方向（制御器または表示器）、色、サイズなど、元のオブジェクトに関する可能な限りの情報が保持されます。ただし、新しいオブジェクトは独自のデータタイプを保持します。オブジェクトの端子またはローカル変数からのワイヤはブロックダイアグラム上に残りますが、壊れている可能性があります。たとえば、数値の端子を文字列の端子に入れ替えると、元のワイヤはブロックダイアグラム上に残りますが、破線になっています。

新しいオブジェクトが入れ替える前のオブジェクトに似ているほど、元の特性を多く保持できます。たとえば、スライドを別のスタイルのスライドと入れ替えると、新しいスライドの高さ、スケール、値、名前、説明は同じになります。これに対し、スライドを文字列制御器に入れ替えると、LabVIEW は名前、説明、およびデータフローの方向だけを保持します。スライドは文字列制御器との共通点が少ないからです。

既存のフロントパネル制御器および表示器を入れ替える場合は、クリップボードからオブジェクトを貼り付けることもできます。この方法では古いオブジェクトの特性は何も保持されませんが、ワイヤはオブジェクトに配線されたままになります。

フロントパネルを構成する

フロントパネルオブジェクトのタブ順序を設定したり、インポートしたグラフィックを使用したり、ウィンドウサイズの変更時にフロントパネルオブジェクトを自動でサイズ変更するように設定することによって、フロントパネルをカスタマイズすることができます。

制御器にキーボードショートカットを設定する

制御器にキーボードショートカットを割り当てると、マウスを使用せずにフロントパネルを操作できます。制御器を右クリックしてショートカットメニューから**上級→キー操作**を選択し、**キー操作**ダイアログボックスを表示します。

メモ　LabVIEW は非表示の制御器のキーボードショートカットには反応しません。

VI の実行中にキーボードショートカットを入力すると、関連付けられている制御器にフォーカスが移動します。制御器がテキスト制御器またはデジタル制御器の場合、LabVIEW はテキストをハイライトして編集可能にします。制御器がブールの場合、スペースバーまたは <Enter> キーを押して値を変更します。

表示器にはデータを入力できないため、**上級→キー操作**ショートカットメニューは淡色表示になっています。

メモ　「キーダウン」イベントを使用して、キーボードの特定のキーが押されたときにイベントの発生をトリガすることもできます。

ユーザインタフェースでキーボードショートカットを設定する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Key Navigation」のセクションを参照してください。

キー操作でボタン動作を制御する

フロントパネルの動作を制御するさまざまなボタンにファンクションキーを関連付けることができます。VI のボタンがダイアログボックスボタンと同様に動作するように構成できます。たとえば、<Enter> キーを押すことがボタンをクリックすることと同じになるように定義できます。

<Enter> キーをダイアログボックスのボタンに関連付けると、LabVIEW は自動的にそのボタンの周りを太い枠で囲みます。

<Enter> キーを制御器に関連付けると、そのフロントパネル上の文字列制御器は復帰文字を受け入れることができなくなります。したがって、そのフロントパネル上のすべての文字列は 1 行に入力されます。長い文字列を表示するには、スクロールバーを使用できます。

ブール制御器を選択して <Enter> キーを押すと、他の制御器が <Enter> キーをキーボードショートカットとして使用している場合でも、そのブール制御器は変更されます。割り当てられた <Enter> キーボードショートカットは、ブール制御器が選択されていない場合のみに適用されます。

フロントパネルオブジェクトのタブ順序を設定する

フロントパネル上の制御器および表示器には、タブ順序と呼ばれる順序がありますが、これはフロントパネル上の制御器と表示器の位置とは無関係です。フロントパネル上に最初に作成した制御器および表示器は、要素 0、2 番目は要素 1、というようになります。制御器または表示器を削除すると、タブ順序は自動的に調整されます。

タブ順序は、VI の実行中にユーザが <Tab> キーを押したときに、LabVIEW が制御器および表示器を選択する順序を決定するものです。また、フロントパネルデータをロギングするときに、ユーザが作成するデータログファイルのレコードに表示される制御器と表示器の順序も決定しま

す。データロギングの詳細については、第 14 章「ファイル I/O」の「フロントパネルのデータを記録する」のセクションを参照してください。

フロントパネルオブジェクトのタブ順序を設定するには、**編集→タブ順序を設定**を選択します。

VI の実行中に <Tab> キーを使用して制御器にアクセスできないようにするには、**キー操作**ダイアログボックスで**この制御器をとばす**チェックボックスをオンにします。

オブジェクトの色を決める

ユーザは多くのオブジェクトの色を変更できますが、すべての色を変更できるわけではありません。たとえば、フロントパネルオブジェクトのブロックダイアグラム端子およびワイヤは、転送するデータのタイプと表現を示す特定の色を使用していますので、ユーザはそれらの色を変更できません。

フロントパネルオブジェクトやフロントパネルとブロックダイアグラムの作業スペースの色を変更するには、色付けツールを使用してオブジェクトまたは作業スペースを右クリックします。また、**ツール→オプション**を選択し て、上部のプルダウンメニューから**色**を選択することによって、ほとんどのオブジェクトのデフォルトの色を変更することができます。

色を使用してユーザインタフェースを設計する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Colors」のセクションを参照してください。

インポートされたグラフィックを使用する

他のアプリケーションからグラフィックをインポートして、フロントパネルの背景、リング制御器の項目、その他の制御器および表示器の一部として使用できます。制御器でのグラフィックの使用方法の詳細については、『LabVIEW Custom Controls, Indicators, and Type Definitions』アプリケーションノートを参照してください。

LabVIEW は、BMP、JPEG、動画 GIF、MNG、動画 MNG、および PNG など、ほとんどの標準グラフィック形式をサポートしています。LabVIEW はまた、透明色の動画もサポートします。

グラフィックをインポートするには、クリップボードにコピーしてフロントパネル上に貼り付けます。**編集→ファイルからピクチャをインポート**を選択することもできます。

メモ　**（Windows および Mac OS）** 画像をコピーと貼り付けによってインポートした場合、画像の透明性が失われます。

インポートしたグラフィックを使用した制御器のサンプルについては、`examples¥general¥controls¥custom.llb` を参照してください。グラフィックを使用してユーザインタフェースを設計する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Graphics and Custom Controls」のセクションを参照してください。

オブジェクトを整列および均等に配置する

フロントパネル上でアライメントグリッドを有効にし、オブジェクトを配置したときに整列するようにするには、**操作→パネル上のアライメントグリッドを有効にする**を選択します。アライメントグリッドを無効にし、グリッドを表示してオブジェクトを手動で整列させる場合は、**操作→パネル上のアライメントグリッドを無効にする**を選択します。また、<Ctrl-#> キーを押して、アライメントグリッドを有効または無効にすることもできます。フランス語のキーボードでは、<Ctrl-”> キー、

（Mac OS）では <Command-*> キー、**（Sun）**では <Meta-#> キー、**（Linux）**では <Alt-#> キーを使用します。

また、ブロックダイアグラム上にあるアライメントグリッドを使用することもできます。

ツール→オプションを選択し、上部のプルダウンメニューから**アライメントグリッド**を選択して、グリッドを非表示にしたりカスタマイズすることができます。

配置後にオブジェクトを整列させるには、オブジェクトを選択して、ツールバー上の**オブジェクトの整列**プルダウンメニューを選択します。オブジェクトを等間隔に配置するには、オブジェクトを選択して、ツールバー上の**オブジェクトの間隔**プルダウンメニューを選択します。

オブジェクトをグループ化およびロックする

グループ化してロックするフロントパネルオブジェクトを選択するには、位置決めツールを使用します。ツールバー上の**並べ替え**ボタンをクリックし、プルダウンメニューから**グループ**または**ロック**を選択します。位置決めツールでオブジェクトを移動しサイズを変更する場合、グループ化されたオブジェクトはその相対位置とサイズを維持します。ロックされたオブジェクトはフロントパネル上でその位置を維持し、オブジェクトをロック解除するまで削除できません。オブジェクトのグループ化とロックは、同時に設定することができます。位置決めツール以外のツールは、グループ化またはロックされたオブジェクトに対し通常どおりに動作します。

オブジェクトをサイズ変更する

ほとんどのフロントパネルオブジェクトは、サイズを変更することができます。位置決めツールをサイズ変更可能なオブジェクトに移動すると、オブジェクトをサイズ変更できる部分にサイズ変更ハンドルまたはサイズ変更円が表示されます。オブジェクトをサイズ変更しても、フォントサイズはそのままです。オブジェクトのグループをサイズ変更すると、グループ内のすべてのオブジェクトがサイズ変更されます。

デジタル数値制御器および表示器などの一部のオブジェクトは、サイズを変更すると水平または垂直方向にのみサイズが変わります。ノブのような他のオブジェクトはサイズ変更しても同じ比率を保持します。位置決めカーソルは同じように表示されますが、オブジェクトを囲んでいる破線の枠は一方向にしか動きません。

オブジェクトをサイズ変更するとき、手動でサイズ変更の方向を制限できます。オブジェクトを垂直または平行方向にのみサイズ変更、または現在の垂直と水平の比率を維持するには、<Shift> キーを押したままサイズ変更ハンドルまたは円形をクリックしてドラッグします。中心点の周りでオブジェクトのサイズを変更するには、<Ctrl> キーを押しながら、サイズ変更ハンドル円をクリックして、ドラッグします。

（Mac OS）の場合は、<Option> キーを押します。**（Sun）**では <Meta> キーを押します。**（Linux）**では <Alt> キーを使用します。

複数のオブジェクトを同じサイズに変更するには、オブジェクトを選択して、ツールバー上の**オブジェクトをサイズ変更**プルダウンメニューを選択します。選択したすべてのオブジェクトを、一番大きいまたは一番小さいオブジェクトの幅または高さ、そして特定のサイズ（ピクセル単位）に変更することができます。

フロントパネルオブジェクトをスケールする

フロントパネルウィンドウをサイズ変更するとき、フロントパネルオブジェクトをスケールしたり、ウィンドウサイズに応じて自動的にサイズが変わるように設定できます。フロントパネル上の 1 つのオブジェクトをスケールするように設定することも、フロントパネル上のすべてのオブジェクトをスケールするようにも設定することもできます。ただし、すべてのオブジェクトをスケールするよう設定するか、またはあらかじめオブジェクトをグループ化しないと、フロントパネル上の複数のオブジェクトをスケールするようには設定できません。オブジェクトをスケールするには、オブジェクトを選択して、**編集→オブジェクトをパネルに合わせてスケール**を選択します。

1 つのフロントパネルオブジェクトをスケールするように設定した場合、フロントパネルウィンドウのサイズ変更に応じてオブジェクトは自動的にサイズ変更されます。フロントパネル上の他のオブジェクトはフロントパネル上の以前の配置と一致するように再配置されますが、フロントパネルの新しいウィンドウサイズに合わせてスケールされることはありません。

図 4-1 のように、自動スケールする 1 つのオブジェクトをフロントパネル上で指定すると、フロントパネル上のいくつかの領域が灰色の線で囲まれます。それらの領域は、スケールするオブジェクトを基準にした他のフロントパネルオブジェクトの位置を定義します。フロントパネルウィンドウをサイズ変更すると、自動スケールされるように設定したオブジェクトはサイズ変更され、元の位置を基準に再配置されます。灰色の線は VI を実行すると消えます。

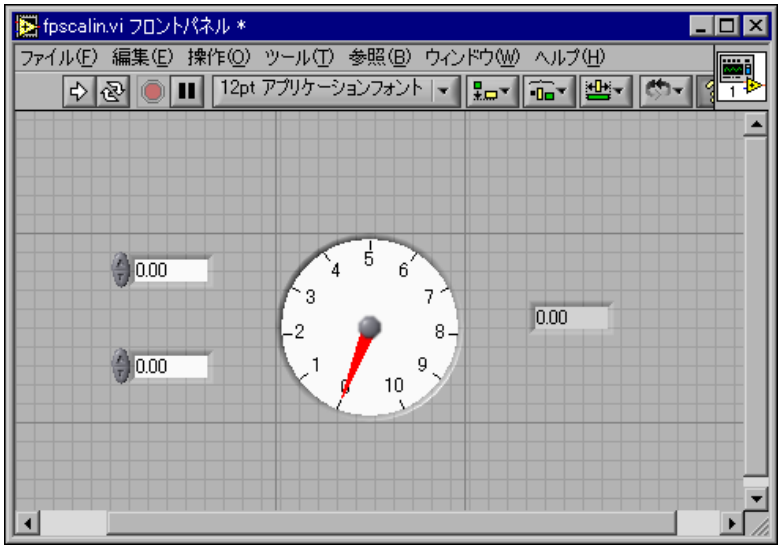


図 4-1　スケールするように設定されたオブジェクトを持つフロントパネル

LabVIEW がオブジェクトを自動的にスケールするときは、ユーザが手動でオブジェクトをサイズ変更したときと同じ規則に従います。たとえば、一部のオブジェクトは水平または垂直方向にのみサイズ変更でき、フォントサイズはオブジェクトがサイズ変更されても同じサイズのままです。

LabVIEW がオブジェクトを自動的にスケールした後、ウィンドウのサイズを変更して元の位置に戻してもオブジェクトが正確に元のサイズに戻らないことがあります。元のフロントパネルウィンドウとオブジェクトのサイズを復元するには、VI を保存する前に**編集→取り消し**を選択します。

配列をスケールするように設定したり、配列内のオブジェクトをスケールするように設定できます。配列をスケールするように設定する場合は、配列内に表示することができる行および列の数を調整します。配列内のオブジェクトをスケールするように設定すると、サイズは異なっても常に同じ数の行および列が配列内に表示されます。

クラスタをスケールするように設定したり、クラスタ内のオブジェクトをスケールするように設定することもできます。クラスタ内のオブジェクトをスケールするように設定すると、クラスタもそれに応じて調整されます。

ウィンドウをサイズ変更せずにフロントパネルにスペースを追加する

ウィンドウのサイズを変更せずにフロントパネルにスペースを追加することができます。込み入った状態で配置またはグループ化されたオブジェクト間のスペースを大きくするには、<Ctrl> キーを押し、位置決めツールを使用してフロントパネルの作業スペースをクリックします。キーを押したまま、挿入するサイズだけ領域をドラッグアウトします。

（Mac OS）の場合は、<Option> キーを押します。**（Sun）**では <Meta> キーを押します。**（Linux）**では <Alt> キーを押します。

破線の枠が付いた四角形により、スペースが挿入される場所が定義されます。キー操作を解除してスペースを追加します。

フロントパネルの制御器および表示器

フロントパネルを作成するには、**制御器**パレットにあるフロントパネルの制御器および表示器を使用します。制御器には、ノブ、押しボタン、ダイアル、その他の入力デバイスがあります。表示器には、グラフ、LED、その他のディスプレイがあります。制御器は計測器入力デバイスをシミュレーションするもので、VI のブロックダイアグラムにデータを供給します。表示器は計測器出力デバイスをシミュレーションするもので、ブロックダイアグラムが集録または生成するデータを表示します。

3 次元および旧バージョンの制御器と表示器

多くのフロントパネルオブジェクトの外観はハイカラーで 3 次元です。オブジェクトの外観を良くするために少なくとも 16 ビットの色を表示するようにモニタを設定します。

3 次元のフロントパネルオブジェクトにも、対応するローカラーの 2 次元オブジェクトがあります。**旧バージョンの制御器**パレットにある 2D の制御器および表示器を使用して、256 色モニタ設定と 16 色モニタ設定の VI を作成します。

ユーザが端子を右クリックしてショートカットメニューから**作成→制御器**または**作成→表示器**を選択したときに LabVIEW が作成する制御器または表示器のスタイルを変更するには、**ファイル→ VI プロパティ**を選択して上部の**カテゴリ**プルダウンメニューから**編集オプション**を選択します。VI の外観と動作の構成の詳細については、第 16 章「VI をカスタマイズする」の「VI の外観と動作を設定する」のセクションを参照してください。プルダウンメニューから**ツール→オプション**と**フロントパネル**を選択し、端子を右クリックしてショートカットメニューから**作成→制御器**または**作成→表示器**を選択すると、新規の VI で作成した制御器または表示器のスタイルを変更できます。

スライド、ノブ、ダイアル、デジタル表示、およびタイムスタンプ

スライド、ノブ、ダイアル、およびデジタル表示をシミュレーションするには、**数値**および**旧バージョンの数値**パレットにある数値制御器および表示器を使用します。このパレットには、色の値を設定するカラーボックスおよびカラーランプも含まれています。数値制御器および表示器を使用して、数値データの入力と表示を行います。

スライド制御器および表示器

スライド制御器および表示器には、垂直と水平のスライド、タンク、および温度計があります。スライド制御器または表示器の値を変更するには、操作ツールを使用してスライダを新しい位置までドラッグするか、スライドオブジェクトの新しいポイントをクリックするか、あるいはオプションのデジタル表示を使用します。スライダを新しい位置にドラッグし、その変更の間 VI が実行されている場合、VI がどのくらいの頻度で制御器を読み取るかによって制御器は VI に中間値を渡します。

スライド制御器または表示器は 2 つ以上の値を表示できます。スライダを追加するには、オブジェクトを右クリックし、ショートカットメニューから**スライダを追加**を選択します。複数のスライダを持つ制御器のデータタイプは各数値を含むクラスタです。クラスタの詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「クラスタ」のセクションを参照してください。

回転式制御器および表示器

回転式制御器および表示器には、ノブ、ダイアル、ゲージ、およびメーターが含まれます。回転式オブジェクトはスライド制御器および表示器と同じように動作します。回転式制御器または表示器の値を変更するには、指針を移動するか、回転式オブジェクトのポイントをクリックするか、またはオプションのデジタル表示を使用します。

回転式制御器または表示器は 2 つ以上の値を表示できます。新しい指針を追加するには、オブジェクトを右クリックし、ショートカットメニューから**指針を追加**を選択します。複数の指針を持つ制御器のデータタイプは各数値を含むクラスタです。クラスタの詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「クラスタ」のセクションを参照してください。

デジタル制御器および表示器

デジタル制御器および表示器は、数値データを入力したり、表示するのに最も簡単な手段です。これらのフロントパネルオブジェクトを水平に拡張して、より多くの桁を表示することができます。以下の方法でデジタル制御器または表示器の値を変更できます。

- 操作ツールまたはラベリングツールを使用してデジタル表示ウィンドウの内側をクリックし、キーボードから数値を入力します。
- 操作ツールを使用して、デジタル制御器の増分矢印ボタンまたは減分矢印ボタンをクリックします。
- 操作ツールまたはラベリングツールを使用して、変更する桁の右側にカーソルを置き、キーボードの上矢印キーまたは下矢印キーを押します。

数値形式

LabVIEW はデフォルトで、数値を計算機のように表示し、保存します。数値制御器または表示器は、自動的に指数表記に切り替わる前に、最大 6 桁を表示します。指数表記に切り替える前に LabVIEW が表示する桁数は、**数値プロパティ**ダイアログボックスの**形式と精度**タブで構成できます。

選択した精度は値の表示にのみ影響を与えます。内部精度は引き続き表記法に依存します。

タイムスタンプ制御器および表示器

タイムスタンプ制御器および表示器を使用して、時間と日付値をブロックダイアグラムに送信、またはブロックダイアグラムから取得します。以下の方法でタイムスタンプ制御器の値を変更できます。

- 定数を右クリックして、ショートカットメニューから**形式と精度**を選択します。

- 左図のように、**時間／日付参照**ボタンをクリックして、**時間と日付を設定**ダイアログボックスを表示します。
- ショートカットメニューから**データ操作→時間と日付を設定**を選択して、**時間と日付を設定**ダイアログボックスを表示します。また、タイムスタンプ制御器を右クリックし、ショートカットメニューから**データ操作→時間を現在に設定**を選択することもできます。

カラーボックス

カラーボックスは特定の値に相当する色を表示します。たとえば、カラーボックスを使用すると、範囲外の値などのさまざまな状態を示すことができます。カラー値は RRGGBB の形式の 16 進数で表します。最初の 2 桁は赤の値を制御します。次の 2 桁は緑の値を制御します。最後の 2 桁は青の値を制御します。

操作ツールまたは色付けツールでカラーボックスをクリックしてカラーパレットを表示し、カラーボックスの色を設定します。

カラーランプ

カラーランプは色で数値を表示します。それぞれが数値とそれに対応する表示色を持つ、少なくとも 2 つの任意マーカから構成されるカラーランプを構成します。入力値が変更されると、表示色はその値に対応する色に変わります。ゲージが危険な値に達したときの警告範囲などのように、カラーランプはデータの範囲を視覚的に示すのに便利です。たとえば、カラーランプを使用して強度チャートやグラフにカラースケールを設定できます。強度チャートおよびグラフの詳細については、第 12 章「グラフおよびチャート」の「強度グラフおよびチャート」のセクションを参照してください。

外観、サイズ、色、および色数をカスタマイズするには、カラーランプを右クリックしてショートカットメニュー項目を使用します。

また、フロントパネル上のすべてのノブ、ダイアル、ゲージにカラーランプを追加できます。メーターには、デフォルトでカラーランプが表示されます。

グラフおよびチャート

グラフ形式でデータのプロットを表示するにはグラフやチャートを使用します。

LabVIEW でのグラフおよびチャートの使用の詳細については、第 12 章「グラフおよびチャート」を参照してください。

ボタン、スイッチ、およびライト

ボタン、スイッチ、およびライトをシミュレーションするには、ブール制御器および表示器を使用します。ブール制御器および表示器を使用して、ブール値（TRUE/FALSE）を入力し表示します。たとえば、測定温度を監視する場合に、温度が一定のレベルを超えたら警告するように、フロントパネルにブール警告ライトを配置できます。

ブールオブジェクトの外観とクリックしたときの動作をカスタマイズするには、ショートカットメニューを使用します。

テキスト入力ボックス、ラベル、およびパス表示

テキスト入力ボックスとラベルをシミュレーションしたり、ファイルやディレクトリの位置を入力したり返したりするには、文字列＆パス制御器および表示器を使用します。

文字列制御器および表示器

フロントパネルの文字列制御器にテキストを入力したり編集するには、操作ツールやラベリングツールを使用します。デフォルトでは、新たに入力したテキストや変更したテキストは、編集セッションを終了するまでブロックダイアグラムには渡されません。編集セッションを終了するには、パネル上の制御器以外の場所をクリックするか、別のウィンドウを表示するか、ツールバーの**入力**ボタンをクリックするか、または数値キーパッドの <Enter> キーを押します。キーボードの <Enter> キーを押すと復帰文字が入力されます。

文字列制御器および表示器の詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「フロントパネルの文字列」のセクションを参照してください。

コンボボックス制御器

フロントパネルでサイクル可能な文字列のリストを作成するには、コンボボックス制御器を使用します。コンボボックス制御器は、テキストまたはメニューリング制御器に似ています。ただし、値およびデータタイプは、リング制御器では数値なのとは異なり、コンボボックス制御器では文字列です。リング制御器の詳細については、この章の「リングおよび列挙体制御器と表示器」のセクションを参照してください。

たとえば、ケーストラクチャでケースを選択するときにコンボボックス制御器を使用できます。ケーストラクチャの詳細については、第 8 章「ループとストラクチャ」の「ケーストラクチャ」のセクションを参照してください。

コンボボックス制御器を右クリックしてショートカットメニューから**未定義文字列を許可**を選択すると、制御器で選択することができるリストに文字列を追加することができます。**コンボボックスプロパティ**ダイアログボックスの**項目を編集**ページ内の文字列の順序により、制御器内の文字列の順序が決定されます。デフォルトでは、コンボボックス制御器を使用すると、ユーザはその制御器に定義された文字列のリストにない文字列の値を入力することができます。コンボボックス制御器を右クリックしてショート

カットメニューから**項目を編集**を選択すると、チェックマークが外され、ユーザが未定義の文字列値を制御器に入力できないようになります。

実行時にコンボボックス制御器に文字列を入力すると、LabVIEW は入力された文字で始まる最初の最短の文字列を選択します。入力された文字に一致する文字列がなく、制御器が未定義の文字列値を許可しない場合、LabVIEW は制御器に入力された文字を受け入れず、表示もしません。

コンボボックス制御器の文字列のリストを構成する場合、各文字列にカスタムの値を指定することができます。カスタムの値を指定すると、フロントパネルのコンボボックス制御器に表示される文字列とブロックダイアグラムのコンボボックス端子が返す文字列値が異なるようにする場合に便利です。コンボボックス制御器を右クリックしてショートカットメニューから**項目を編集**を選択し、**コンボボックスプロパティ**ダイアログボックスの**項目を編集**ページの**値一致ラベル**チェックボックスをオフにします。このダイアログボックスの表の**値**の列で、制御器の各文字列に対応する値を変更します。

パス制御器および表示器

パス制御器および表示器を使用して、ファイルやディレクトリの位置を入力したりその位置を返します。パス制御器および表示器の動作は文字列制御器および表示器に似ていますが、LabVIEW は使用しているプラットフォーム標準の構文を使用してパスをフォーマットします。

無効なパス

パスを返す関数が失敗すると、その関数は無効なパス値である < 無効パス > を表示器に返します。< 無効パス > 値をパス制御器のデフォルト値として使用すると、ユーザがパスを指定しなかったときに検知して、パスを選択するオプションのあるファイルダイアログボックスを表示できます。ファイルダイアログボックスを表示するには、「ファイルダイアログ（File Dialog）」関数を使用します。

空のパス

パス制御器内の空のパスは、Windows および Mac OS では空の文字列として表示され、UNIX ではスラッシュ（/）として表示されます。パスの指定を促すプロンプトを表示するには、空のパスを使用します。ファイル I/O 関数に空のパスを配線すると、空のパスはコンピュータにマップされたドライブの一覧を参照します。

（Mac OS）　空のパスは、マウントされているボリュームを参照します。
（UNIX）　空のパスはルートディレクトリを参照します。

配列とクラスタの制御器と表示器

配列＆クラスタパレットおよび**旧バージョンの配列＆クラスタ**にある配列およびクラスタの制御器および表示器を使用して、他の制御器および表示器の配列およびクラスタを作成します。配列とクラスタの詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「データを配列やクラスタとグループ化する」のセクションを参照してください。

配列＆クラスタパレットには、標準のエラークラスタ制御器および表示器、バリアント制御器も含まれています。エラークラスタの詳細については、第 6 章「VI の実行とデバッグ」の「エラークラスタ」のセクションを参照してください。バリアント制御器の詳細については、第 5 章「ブロックダイアグラムを作成する」の「バリアントデータを処理する」のセクションを参照してください。

リストボックス、ツリー制御器、および表

リスト & 表パレットおよび**旧バージョンのリスト & 表**パレットにあるリストボックス制御器を使用して、選択項目のリストを表示します。

リストボックス

リストボックスを構成して、1 項目または複数項目の選択を受け入れるようにできます。項目サイズおよび作成日時等、各項目に関する詳細を表示するには、複数列リストボックスを使用します。

実行時にリストボックスに文字を入力すると、LabVIEW は入力された文字で始まるリストボックス内で最初の項目を選択します。入力した文字に一致する前後の項目にジャンプするには、左右の矢印キーを使用します。

ディレクトリとファイルが異なる記号を持つ **VI ライブラリマネージャ**ダイアログボックスのように、リスト項目の隣に記号を追加できます。リスト項目の間に区切り線を挿入することもできます。

「プロパティノード（Property Node）」を使用してリスト項目を修正すると、現在選択されている項目またはユーザがダブルクリックした項目を検出するなど、リスト項目に関する情報を取得することができます。「プロパティノード」の詳細については、第 17 章「VI をプログラム的に制御する」の「プロパティノード」のセクションを参照してください。

ツリー制御器

ツリー制御器を使用して、選択する項目の階層リストを表示します。ツリー制御器に入力した項目を、項目のグループ、つまりノードに整理します。ノードの隣にある展開記号をクリックして、ノードを展開し、そのノード内にあるすべての項目を表示します。また、ノードの隣にある記号をクリックして、ノードを格納することもできます。

メモ　ツリー制御器の作成および編集は、LabVIEW 開発システムおよび LabVIEW プロフェッショナル開発システムのみで実行できます。VI にツリー制御器が含まれている場合、すべての LabVIEW パッケージで VI を実行できますが、ベースパッケージでは制御器を構成できません。

実行時にツリー制御器に文字を入力すると、LabVIEW は入力された文字で始まる文字列内の最初の項目をツリー制御器内で選択します。項目を選択し、ピリオド（.）キーを押して現在の項目をインデントするか、またはコンマ（,）キーを押して現在の項目を左に移動させることによって、ツリー制御器内の項目の階層を変更できます。

ツリー制御器内の項目は、リストボックスと同じ方法で構成します。また、各ノードの隣に表示される記号の種類を変更でき、ツリー制御器内でユーザが項目をドラッグできるかどうかを設定することもできます。

「インボークノード（Invoke Node）」を使用して、ツリー制御器内の項目を修正したり、ユーザがどの項目をダブルクリックしたかなど、項目に関する情報を取得することができます。ツリー制御器内で項目を追加すると、項目固有のタグが作成されます。このタグを使用して、項目を修正したり、プログラム的に情報を取得します。各項目に対して作成されたタグを修正するには、ツリー制御器を右クリックしてショートカットメニューから**項目を編集**選択します。「インボークノード」の詳細については、第 17 章「VI をプログラム的に制御する」の「インボークノード」のセクションを参照してください。

ツリー制御器のサンプルについては、`examples¥general¥controls¥Directory Tree Control.llb` の Directory Hierarchy in Tree Control VI を参照してください。

表

リスト & 表および**旧バージョンのリスト & 表**パレットにある表制御器を使用して、フロントパネルに表を作成します。

表制御器の詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「表」のセクションを参照してください。

リングおよび列挙体制御器と表示器

リング & 列挙体または**旧バージョンのリング & 列挙体**パレットにあるリングおよび列挙体制御器および表示器を使用して、繰り返し表示できる文字列のリストを作成します。

リング制御器

リング制御器は、数値を文字列またはピクチャに関連付ける数値オブジェクトです。リング制御器は、選択肢を繰り返し表示できるプルダウンメニューとして表示されます。

リング制御器は、トリガモードなどの互いに排他的な項目を選択するときに便利です。たとえば、連続、単独、または外部トリガから選択する場合にはリング制御器を使用します。

リング制御器を右クリックしてショートカットメニューから**項目を編集**を選択すると、制御器で選択することができるリストに項目を追加することができます。**リングププロパティ**ダイアログボックスの**項目を編集**ページ内の項目の順序により、制御器内の項目の順序が決定されます。また、リング制御器を右クリックしてショートカットメニューから**未定義の値を許可**を選択することによって、ユーザが制御器に対して定義された項目リストにエントリに関連付けられていない数値を入力できるようにリング制御器を構成することもできます。

実行時にリング制御器に未定義の数値を入力するには、制御器を左クリックしてショートカットメニューから **< その他 >** を選択し、表示されたデジタル表示に数値を入力して、<Enter> キーを押します。未定義の値がリング制御器にかぎ括弧付きで表示されます。LabVIEW は、未定義の値を制御器内で選択できる項目リストに追加しません。

リング制御器の項目リストを構成する場合、各項目に特定の数値を割り当てることができます。項目に特定の数値を割り当てない場合、LabVIEW はリスト内の項目の順序に従って、最初の項目に値 0 を割り当て、以降連続する値を割り当てます。特定の数値を割り当てるには、リング制御器を右クリックしてショートカットメニューから**項目を編集**を選択し、**リングプロパティ**ダイアログボックスの**項目を編集**ページの**後に続く値**チェックボックスからチェックマークを外します。このダイアログボックスの表の**値**セクションで、制御器の各項目に対応する数値を変更します。リング制御器の各項目は、固有の値を持つ必要があります。

列挙体制御器

選択する項目のリストを表示するには列挙体制御器を使用します。列挙体制御器、つまり列挙体はテキストまたはメニューリング制御器に似ています。ただし、列挙体制御器のデータタイプには、制御器内に数値および文字列ラベルの情報が含まれています。リング制御器のデータタイプは数値です。

メモ　ユーザが未定義の値を列挙体制御器に入力するようにすることはできません。また、特定の数値を列挙体制御器内の項目に割り当てることもできません。この場合は、リング制御器を使用する必要があります。リング制御器の詳細については、この章の「リング制御器」のセクションを参照してください。

ケーススストラクチャでケースを選択するときには列挙体制御器を使用します。ケーススストラクチャの詳細については、第 8 章「ループとストラクチャ」の「ケーススストラクチャ」のセクションを参照してください。

列挙体制御器の数値表記法は、8 ビット、16 ビット、32 ビット符号なし整数です。制御器の表記法を変更するには、列挙体制御器を右クリックしてショートカットメニューから**表記法**を選択します。

上級の列挙体制御器および表示器

「インクリメント（Increment）」および「デクリメント（Decrement）」を除くすべての算術関数は、列挙体制御器を符号なし整数と同じように処理します。「インクリメント」は最後の列挙値を最初の値まで増分し、デクリメントは最初の列挙値を最後の値まで減分します。符号付き整数を列挙体に強制的に変換する場合、負の値は最初の列挙値に変換され、範囲外の正の値は最後の列挙値に変換されます。範囲外の符号なし整数は常に最後の列挙値に変換されます。

浮動小数点数を列挙体表示器に配線する場合、浮動小数点数は列挙体表示器内の最も近い数値に強制的に変換されます。範囲外の数値は前述の説明と同様に処理されます。列挙体制御器を任意の数値に配線すると、列挙体の値は数値に強制的に変換されます。列挙体制御器を列挙体表示器に配線するには、表示器内の項目が制御器内の項目に一致している必要があります。ただし、表示器は制御器よりも多くの項目を持つことができます。

コンテナ制御器

コンテナおよび**旧バージョンのコンテナ**パレットにあるコンテナ制御器を使用して、制御器と表示器をグループ化、現在の VI のフロントパネルに他の VI のフロントパネルを表示、または**（Windows）**ActiveX のオブジェクトをフロントパネルに表示します。ActiveX の使用方法の詳細については、第 19 章「Windows の接続性」を参照してください。

タブ制御器

タブ制御器を使用すると、フロントパネル制御器および表示器を狭い領域内で重ねることができます。タブ制御器はページとタブで構成されています。タブ制御器の各ページにフロントパネルオブジェクトを配置し、タブをセレクタとして使用して複数のページを表示します。

一緒に使用するフロントパネルオブジェクトが複数ある場合や操作の特定の段階で、タブ制御器は役に立ちます。たとえば、テストを開始する前にまずユーザがいくつかの設定を構成する必要があり、その後テストの進行中にその内容を修正できるようになり、最後に関連データのみを表示して保存できる VI を使用している場合などです。

ブロックダイアグラムでは、タブ制御器はデフォルトで列挙体制御器です。タブ制御器上の制御器および表示器の端子は、他のブロックダイアグラム端子として表示されます。列挙体制御器の詳細については、この章の「列挙体制御器」のセクションを参照してください。

サブパネル制御器

サブ制御器を使用して、現在の VI のフロントパネル上に他の VI のフロントパネルを表示します。たとえば、サブパネル制御器を使用して、ウィザードのように動作するユーザインタフェースを設計できます。**戻る**ボタンおよび**次へ**ボタンを最上位フロントパネルに配置して、サブパネル制御器を使用してウィザードの各ステップの異なるフロントパネルをロードします。

メモ　サブパネル制御器の作成および編集は、LabVIEW 開発システムおよび LabVIEW プロフェッショナル開発システムのみで実行できます。VI にサブパネル制御器が含まれている場合、すべての LabVIEW パッケージで VI を実行できますが、ベースパッケージでは制御器を構成できません。

フロントパネル上にサブパネル制御器を配置した場合、LabVIEW はブロックダイアグラム上でその制御器に対するフロントパネル端子を作成しません。代わりに、選択された VI に挿入メソッドを使用して、「インボークノード（Invoke Node）」を作成します。サブパネル制御器で VI をロードするには、その VI へのリファレンスを「インボークノード」に配

線します。VI レファレンスと「インボークノード」の詳細については、第 17 章「VI をプログラム的に制御する」を参照してください。

メモ　サブパネル制御器は端子を持たないため、サブパネル制御器の配列、およびサブパネルのタイプ定義を作成することはできません。サブパネル制御器をクラスタに配置して、サブパネル制御器を他の制御器とともにグループ化できますが、クラスタにはサブパネル制御器のみを含むことはできません。

ロードする VI のフロントパネルが開いている場合や、同じフロントパネル上の他のサブパネル制御器をロードした場合、LabVIEW はエラーを返し、サブパネル制御器にフロントパネルをロードできなくなります。また、LabVIEW のリモートインスタンスで VI のフロントパネルをロードすることも、フロントパネルを再帰的にロードすることもできません。

キーボードのショートカットを使用して、サブパネル制御器内のフロントパネルに移動したり操作したりすることはできません。

実行されていない VI をロードすると、サブパネルの VI は実行モードでロードされます。

LabVIEW は、サブパネル制御器にロードした VI のフロントパネルの可視領域のみを表示します。サブパネル制御器を含む VI を停止した後、サブパネル制御器のフロントパネルは消去されます。また、「VI を削除」メソッドを使用して、サブパネル制御器で VI をアンロードすることもできます。

サブパネル制御器のサンプルについては、`examples¥general¥controls¥subpanel.llb` を参照してください。

I/O 名制御器および表示器

構成する DAQ チャネル名、VISA リソース名、および IVI 論理名を I/O VI に渡し、計測器や DAQ デバイスと通信するには、I/O 名制御器および表示器を使用します。

I/O 名定数は**関数**パレットにあります。

メモ　すべての I/O 名制御器または定数はすべてのプラットフォームで使用することができます。これにより、プラットフォーム固有のデバイスと通信可能なあらゆるプラットフォーム上で I/O VI を開発することができます。ただし、プラットフォーム固有の I/O 制御器を持つ VI をそのデバイスをサポートしないプラットフォームで実行すると、エラーが発生します。

(Windows) DAQ チャネル名、VISA リソース文字列、および IVI 論理名を構成するには、**ツール**メニューからアクセスできる Measurement & Automation Explorer を使用します。

(Mac OS) ナショナルインスツルメンツの DAQ ハードウェアを構成するには、**ツール**メニューからアクセスできる NI-DAQ 構成ユーティリティを使用します。、DAQ チャネル名を構成するには、**ツール**メニューからアクセスできる DAQ チャネルウィザードを使用します。

(Mac OS および UNIX) VISA リソース名と IVI 論理名を構成するには、計測器の構成ユーティリティを使用します。構成ユーティリティの詳細については計測器のドキュメントを参照してください。

IMAQ セッション制御器は、ハードウェアへの接続を示す固有の識別子です。

波形制御器

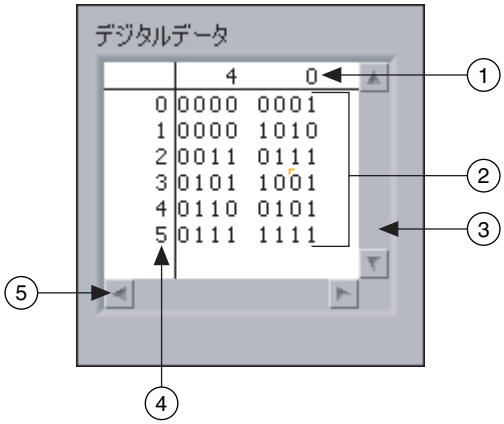
波形の個々のデータ要素を操作するには、波形制御器を使用します。波形データタイプの詳細については、第 12 章「グラフおよびチャート」の「波形データタイプ」のセクションを参照してください。

デジタル波形制御器

デジタル波形制御器を使用して、デジタル波形の個々のデータ要素を操作します。デジタル波形制御器の詳細については、第 12 章「グラフおよびチャート」の「デジタル波形データタイプ」のセクションを参照してください。

デジタルデータ制御器

デジタルデータ制御器には、行と列に配列されたデジタルデータが含まれます。デジタルデータタイプを使用してデジタル波形を生成、またはデジタル波形から抽出されたデジタルデータを表示します。デジタル波形データタイプをデジタルデータ表示器に配線し、デジタル波形のサンプルおよび信号を表示します。図 4-2 のデジタルデータ制御器は、それぞれ 8 つの信号を持つ 5 つのサンプルを示しています。



1　信号	3　垂直スクロールバー	5　水平スクロールバー
2　データ	4　サンプル	

図 4-2　デジタルデータ制御器

デジタルデータ制御器で、行と列を挿入したり削除することができます。行を挿入するには、サンプル列内のサンプルを右クリックしてショートカットメニューから**前に行を挿入**を選択します。行を削除するには、サンプル列内のサンプルを右クリックしてショートカットメニューから**行を削除**を選択します。列を挿入するには、信号列内の信号を右クリックしてショートカットメニューから**前に列を挿入**を選択します。列を削除するには、信号列内の信号を右クリックしてショートカットメニューから**列を削除**を選択します。

制御器内でデジタルデータを切り取り、コピー、貼り付けすることもできます。データを切り取るには、切り取る行または列を選択して右クリックし、ショートカットメニューから**データ操作→データを切り取る**を選択します。データの行または列全体のみを切り取ることができます。切り取ったデジタルデータで新規行および列を作成できません。データをコピーするには、コピーする範囲を選択して右クリックし、ショートカットメニューから**データ操作→データをコピー**を選択します。デジタルデータを貼り付けるには、コピーする範囲を選択してショートカットメニューから**データ操作→データを貼り付け**を選択します。デジタルデータを貼り付ける範囲は、データを切り取った範囲またはコピーした範囲と同じ次元である必要があります。たとえば、4 ビットのデータを 1 行からコピーした場合、同じ行または他の行に貼り付けるには、4 つの既存ビットを選択する必要があります。4 ビットデータを 2 行 × 2 列からなる範囲からコピーした場合、2 行 × 2 列からなる範囲にデータを貼り付ける必要があります。

デジタルデータ制御器およびデジタル波形制御器は、0、1、L、H、Z、X、T、および V の値を受け入れます。デジタルデータ制御器では、バイナリ、16 進数、8 進数、および 10 進数形式でデータを表示します。デジタルの状態、L、H、Z、X、T、および V は、一部の計測デバイスが使用する状態で、16 進数、8 進数、または 10 進数で値を表示すると、疑問符で表示されます。制御器のデータ形式を選択するには、制御器を右クリックしてショートカットメニューから**データ形式**を選択します。

データをデジタルデータに変換する

通常、集録した信号は未処理データとして返されます。デジタル波形グラフで信号をグラフで描写するには、集録した未処理データをデジタルデータタイプまたはデジタル波形データタイプに変換する必要があります。データをデジタル波形データタイプに変換するには、「アナログ－デジタル波形変換（Analog to Digital Waveform）」VI を使用します。デジタル波形データタイプからデジタルデータを抽出するには、「波形コンポーネント取得（Get Waveform Components）」関数を使用します。

図 4-3 のブロックダイアグラムは、振幅 5 の正弦波の集録をシミュレーションしたものです。これは正弦波が -5 から 5 までの範囲の値を含むことができることを意味しています。このブロックダイアグラムの「アナログ－デジタル波形変換」VI は、各値を 8 ビットで表します。8 ビットは、最小値 -5 と最大値 5 を表すことができます。デジタル波形プローブは、結果値の一部をバイナリ形式で表示します。

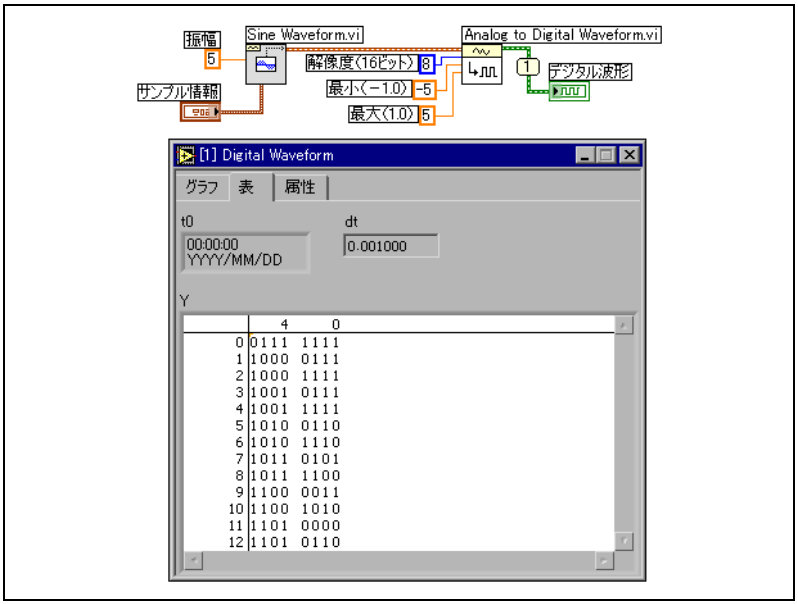


図 4-3　アナログ波形をデジタル波形に変換する

部分デジタル信号を読み取る

「部分デジタル信号（Digital Signal Subset）」VI を使用して、デジタル波形の個々の信号を抽出します。図 4-4 のブロックダイアグラムは、抽出した個々の信号と他のデジタルデータを組み合わせて、デジタル波形を新規作成する方法を示します。

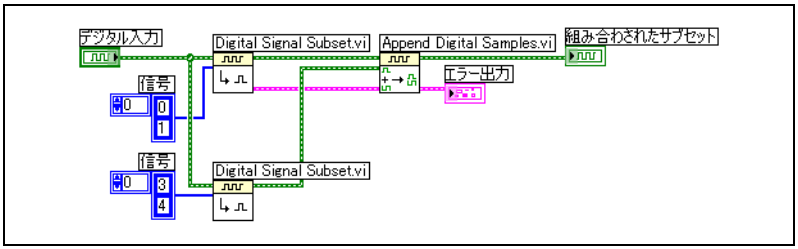


図 4-4　デジタル波形を連結する

上の「部分デジタル信号（Digital Signal Subset）」VI は、デジタル波形から最初の信号と 2 番目の信号を抽出します。下の「部分デジタル信号（Digital Signal Subset）」VI は、4 番目と 5 番目の信号を抽出します。「デジタル波形連結（Append Digital Samples）」VI は、最初の信号と 4 番目の信号を連結し、2 番目の信号と 5 番目の信号を連結し、結果として得られた 2 つの信号をデジタル波形グラフにプロットします。

デジタルサンプルおよび信号を連結する

「デジタル信号連結（Append Digital Signals）」VI を使用して、ある 1 つのデジタル信号内のサンプルを他のデジタル信号のサンプルの最後に連結します。信号を同じ数のサンプルまたは異なる数のサンプルに連結することができます。たとえば、2 つのデジタル信号がそれぞれ 8 ビットの 2 行から構成されている場合、結果として得られるデジタルデータは 16 ビットの 2 行から構成されます。8 ビットの 2 行で構成される信号と、8 ビットの 1 行で構成される 2 つの信号を組み合わせた場合、結果として得られるデジタルデータは 16 ビットの 2 行から構成されます。「デジタル信号連結（Append Digital Signals）」VI は、2 番目のサンプルの残りの列を、**デフォルト値**入力で選択した値でパディングします。

デジタルデータを圧縮する

「デジタル圧縮（Compress Digital）」VI を使用して、同じビット数の複数のシリアルデジタル信号を同じ行に表示してデータをよりわかりやすく表示するために、デジタルデータを圧縮します。たとえば、10 つのデジタル波形サンプルを集録し、10 番目の波形が他の 9 つと異なる場合、デジタルデータを圧縮すると、どの波形が異なるかを簡単に検出できます。

デジタルデータの圧縮は、メモリリソースの節約にもなります。「デジタル解凍（Uncompress Digital）」VI を使用して、圧縮したデジタルデータを解凍します。

パターンを検索する

「デジタルパターン検索（Search for Digital Pattern）」VI を使用して、検索するデジタルパターンを指定します。たとえば、大きなデジタル波形を集録し、そのデジタル波形のどの部分が特定のパターンと一致するかを調べたい場合、そのパターンを「デジタルパターン検索」VI の**デジタルパターン**入力に配線して一致があるかどうかを検出します。

オブジェクトまたはアプリケーションへのリファレンス

ファイル、ディレクトリ、デバイス、およびネットワーク接続を処理するには、**Refnum** および**旧バージョンの Refnum** パレットにあるリファレンス番号制御器および表示器を使用します。サブ VI にフロントパネルオブジェクト情報を渡すには制御器 refnum を使用します。制御器 Refnum の詳細については、第 17 章「VI をプログラム的に制御する」の「フロントパネルオブジェクトを制御する」のセクションを参照してください。

リファレンス番号、つまり refnum は、ファイル、デバイス、ネットワーク接続などのオブジェクトに固有の識別子です。ファイル、デバイス、またはネットワーク接続を開くと、それらに関連付けられた refnum が作成されます。開いているファイル、デバイス、またはネットワーク接続で行うすべての操作で、各オブジェクトを識別する refnum が使用されます。VI との間で refnum の受け渡しを行うには、refnum 制御器または表示器を使用します。たとえば、ファイルを閉じて開く操作を行わずに refnum が参照しているファイルの内容を変更するには、refnum 制御器または表示器を使用します。

refnum は開いているオブジェクトを指す一時的なポインタであるため、オブジェクトが開いているときのみ有効です。オブジェクトを閉じると、refnum とオブジェクトの関連付けは解除され、refnum は使用されなくなります。もう一度オブジェクトを開くと、最初の refnum とは異なる新規 refnum が作成されます。

オブジェクトから読み書きを行った現在の位置、ユーザのアクセス回数など、各 refnum に関連付けられた情報は保存されるので、1 つのオブジェクトに対して複数の操作を平行して実行できます。VI が何回もオブジェクトを開くと、そのたびに異なる refnum が返されます。

ダイアログ制御器と表示器

ダイアログ制御器は、作成したダイアログボックスで使用します。ダイアログ制御器および表示器は、ダイアログボックスで使用するために特別に設計されており、リング制御器、スピン制御器、数値スライドおよび進行状況バー、リストボックス、表、文字列およびパス制御器、タブ制御器、ツリー制御器、ボタン、チェックボックス、ラジオボタン、親の背景色と自動的に一致する不透明ラベルなどがあります。これらの制御器の外観はフロントパネルに表示される制御器の外観とは異なります。これらの制御器はシステムで設定された色で表示されます。

ダイアログ制御器の外観は VI を実行するプラットフォームによって異なるので、作成する VI の制御器の外観はすべての LabVIEW のプラットフォーム上で互換性があります。別のプラットフォームで VI を実行する場合、ダイアログ制御器の色と外観はそのプラットフォームの標準ダイアログボックス制御器に合わせて変わります。

ファイル→ VI プロパティを選択し、**カテゴリ**プルダウンメニューから**ウィンドウの外観**を選択して、メニューバーおよびスクロールバーを非表示にしたり、、それぞれのプラットフォームの標準ダイアログボックスのように表示され動作する VI を作成します。端子を右クリックしてショートカットメニューから**作成→制御器**または**作成→表示器**を選択したときに LabVIEW が作成する制御器または表示器のスタイルを変更するには、**カテゴリ**プルダウンメニューから**編集オプション**を選択します。端子を右クリックしてショートカットメニューから**作成→制御器**または**作成→表示器**を選択したときに LabVIEW が新規 VI で作成する制御器または表示器のスタイルを変更するには、**ツール→オプション**を選択して上部のプルダウンメニューから**フロントパネル**を選択します。VI の外観と動作の構成の詳細については、第 16 章「VI をカスタマイズする」の「VI の外観と動作を設定する」のセクションを参照してください。

ラベルを付ける

フロントパネルとブロックダイアグラムのオブジェクトを識別するにはラベルを使用します。

LabVIEW では、所有ラベルおよびフリーラベルの 2 種類のラベルを使用することができます。所有ラベルは特定のオブジェクトに属するもので、そのオブジェクトとともに移動し、そのオブジェクトだけに注釈を付けるものです。所有ラベルは個別に移動できますが、ラベルを所有するオブジェクトを移動すると、ラベルもそのオブジェクトとともに移動します。所有ラベルを隠すことはできますが、ラベル単独ではコピーまたは削除できません。ショートカットメニューから**表示項目→単位ラベル**を選択すると、数値制御器や数値表示器の単位ラベルを表示することもできます。

数値端子の詳細については、第 5 章「ブロックダイアグラムを作成する」の「数値単位および厳密タイプチェック」のセクションを参照してください。

フリーラベルはどのオブジェクトにも連結されていないので、個別に作成、移動、回転、または削除が可能です。フリーラベルを使用してフロントパネルおよびブロックダイアグラムに注釈を付けることができます。

フリーラベルは、ブロックダイアグラム上でコードを文書化したり、フロントパネル上に使用方法を表示するのに役に立ちます。フリーラベルを作成したり、いずれかのタイプのラベルを編集するには、空き領域をダブルクリックするか、ラベリングツールを使用します。

ユーザインタフェースにわかりやすいラベルを作成する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Labels」のセクションを参照してください。

キャプション

フロントパネルオブジェクトには、キャプションを付けることもできます。オブジェクトを右クリックしてショートカットメニューから**表示項目→キャプション**を選択し、キャプションを表示します。ラベルとは異なり、キャプションはオブジェクトの名前に影響を与えず、オブジェクトラベルの補足説明として使用できます。キャプションはブロックダイアグラム上には表示されません。

コネクタペーン端子にオブジェクトを指定する場合、ブロックダイアグラム上の端子をカーソルで移動するのに配線ツールを使用するとき、キャプションが現れます。また**ヘルプ**ウィンドウ中の端子の隣にもキャプションが表示されます。コネクタペーン端子の詳細については、第 7 章「VI およびサブ VI を作成する」の「コネクタペーンを設定する」のセクションを参照してください。

テキストの特性

LabVIEW は、使用しているコンピュータにインストールされているフォントを使用します。テキストの属性を変更するには、ツールバーの**テキスト設定**プルダウンメニューを使用します。**テキスト設定**プルダウンメニューから選択する前にオブジェクトまたはテキストを選択した場合、その変更は選択したすべてのオブジェクトまたはテキストに適用されます。何も選択しないと、デフォルトのフォントが適用されます。デフォルトのフォントが変わっても、既存のラベルのフォントは変わりません。それ以降に作成するラベルにのみ影響します。

フロントパネルの**テキスト設定**プルダウンメニューから**フォントダイアログ**を選択して、選択したテキストに特定のフォントスタイルを適用します。どのテキストも選択しなかった場合は、**パネルのデフォルト**オプションのチェックマークがオンになります。どのオブジェクトも選択せずにブロックダイアグラムから**テキスト設定→フォントダイアログ**を選択すると、**ダイアグラムのデフォルトフォント**オプションにチェックマークが表示されます。フロントパネルとブロックダイアグラムには、別々のフォントを設定できます。たとえば、ブロックダイアグラムには小さなフォントを表示し、フロントパネルには大きなフォントを表示できます。

テキスト設定プルダウンメニューには、以下の標準フォントが含まれています。

- **アプリケーションフォント**：**制御器**と**関数**パレット、および新しい制御器のテキストに使用されるデフォルトのフォント。
- **システムフォント**：メニューに使用されるフォント。
- **ダイアログフォント**：ダイアログボックスのテキストに使用されるフォント。

これらの標準フォントのいずれかを含む VI を他のプラットフォームに転送する場合、最も近いフォントが使用されます。

テキスト設定プルダウンメニューには、**サイズ**、**スタイル**、**調整**、および**色**サブメニュー項目があります。

これらのサブメニューから選択したフォントは、選択されているオブジェクトに適用されます。たとえば、ノブとグラフを選択して別のフォントを選択すると、ラベル、スケール、およびデジタル表示がすべてそのフォントに変更されます。

LabVIEW は、変更時にできるだけ多くのフォント属性を維持します。たとえば、いくつかのオブジェクトを Courier フォントに変更した場合、そのオブジェクトは、可能であればそのサイズとスタイルを保持します。**フォント**ダイアログボックスを使用すると、選択したオブジェクトが選択したテキスト特性に変更されます。標準のフォントのいずれかまたは現在使用中のフォントを選択すると、選択したオブジェクトは、そのフォントに関連付けられているフォントとサイズに変更されます。

スライドのように複数の部分にテキストが含まれるオブジェクトを操作する場合、フォントを変更すると現在選択しているオブジェクトまたはテキストが影響を受けます。たとえば、スライド全体を選択して、**テキスト設定**プルダウンメニューから**スタイル→太字**を選択すると、スケール、デジタル表示、およびラベルはすべて太字フォントに変更されます。ラベルだけを選択して**太字**を選択すると、ラベルのみが太字フォントに変更されます。スケールマーカからテキストを選択して**太字**を選択すると、すべてのマーカが太字フォントに変更されます。

フォントとテキスト特性を使用してユーザインタフェースを設計する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Fonts and Text Characteristics」のセクションを参照してください。

ユーザインタフェースを設計する

VI がユーザインタフェースまたはダイアログボックスとして機能する場合、フロントパネルの外観とレイアウトは重要です。フロントパネルは、実行するアクションを容易に識別できるように設計します。フロントパネルは計測器や他のデバイスと同様に設計できます。

ユーザインタフェースの設計については、『LabVIEW Development Guidelines』マニュアルを参照してください。

イベントを使用してユーザインタフェースの機能を拡張する方法については、第 8 章「ループとストラクチャ」の「ケースストラクチャとシーケンスストラクチャ」のセクションを参照してください。

フロントパネルの制御器および表示器

制御器と表示器はフロントパネルの主要なコンポーネントです。フロントパネルを設計する場合は、ユーザがどのように VI と対話するかを考慮に入れて、制御器と表示器を論理的にグループ化します。複数の制御器が関連している場合は、それらの制御器の周囲に装飾フレームを追加したり、それらをクラスタに入れます。**装飾体**パレットにある装飾を使用して、フロントパネルのオブジェクト（ボックス、線または矢印）をグループ化したり分割することができます。これらのオブジェクトは装飾体専用であり、データを表示しません。

フロントパネルオブジェクトの間隔が狭くならないように配置してください。フロントパネルを読みやすくするため、ある程度の間隔を空けます。また、間隔を空けることによって、関係のない制御器やボタンを誤ってクリックするのを防ぎます。

一般的な用語を使用して、ボタンに特定の名前を割り当てます。OK ではなく開始、停止、保存などの名前を使用します。特定の名前を付けた方が、VI が使用しやすくなります。

デフォルトの LabVIEW フォントおよび色を使用します。LabVIEW は、標準のフォントを、異なるプラットフォームの類似したフォントグループに置き換えます。選択した別のフォントがコンピュータで使用できない場合は、最も近いフォントに置き換えます。LabVIEW は色もフォントと同様に処理します。コンピュータで使用できない色は、最も近い色に置き換

えます。システムカラーを使用して、フロントパネルの外観を、VI を実行しているコンピュータのシステムカラーに合わせることもできます。

オブジェクト上に他のオブジェクトを配置しないでください。ラベルまたは他のオブジェクトを制御器あるいは表示器の全部または一部の上に置くと、画面更新が遅くなり、制御器または表示器がちらつく可能性があります。

レイアウト、フォント、および色を使用してユーザインタフェースを設計する方法については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」を参照してください。

ファイルダイアログボックスを設計する

VI に連続するダイアログボックスが含まれていて、それが画面上の同じ位置に表示される場合は、最初のダイアログボックスのボタンが、次のダイアログボックスのボタンと同じ位置にならないようにダイアログボックスを構成します。最初のダイアログボックスでボタンをダブルクリックしたときに、知らずに次のダイアログボックスのボタンをクリックしてしまっていることがあります。

ダイアログ制御器の詳細については、この章の「ダイアログ制御器と表示器」のセクションを参照してください。ユーザインタフェースでダイアログボックスを使用する方法については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Dialog Boxes」のセクションを参照してください。

画面サイズを選択する

VI の設計時には、画面解像度が異なるコンピュータでフロントパネルを表示できるかどうかを考慮に入れます。**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**ウィンドウサイズ**を選択します。**様々なモニタの解像度でウィンドウ比率を維持**チェックボックスをオンにして、画面解像度に対するフロントパネルウィンドウ比率を維持します。

ユーザインタフェースの画面サイズの選択については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Sizing and Positioning」のセクションを参照してください。

5

ブロックダイアグラムを作成する

フロントパネルを作成したら、グラフィカルに表現された関数を使用してコードを追加し、フロントパネルのオブジェクトを制御します。ブロックダイアグラムにはグラフィカルソースコードが含まれています。

詳細については

ブロックダイアグラムの設計と構成の詳細については、『LabVIEW ヘルプ』を参照してください。

フロントパネルオブジェクトとブロックダイアグラム端子の関係

フロントパネルオブジェクトは、ブロックダイアグラム上では端子として表示されます。ブロックダイアグラム端子をダブルクリックすると、フロントパネル上の対応する制御器または表示器がハイライトされます。

端子は、フロントパネルとブロックダイアグラムの間で情報を交換する入出力ポートです。フロントパネルの制御器に入力したデータは、制御器端子を介してブロックダイアグラムに入力されます。実行中に、出力データは表示器端子に移動します。この出力データはブロックダイアグラムを出て、再度フロントパネルに入り、フロントパネル表示器に表示されます。

ブロックダイアグラムオブジェクト

ブロックダイアグラム上のオブジェクトには、端子、ノード、および関数が含まれています。ブロックダイアグラムは、ワイヤでオブジェクトを接続して作成します。

ブロックダイアグラム端子

フロントパネルの制御器または表示器を構成して、アイコンまたはデータタイプ端子としてブロックダイアグラムに表示できます。デフォルトでは、フロントパネルオブジェクトはアイコン端子として表示されます。たとえば、左図に示すノブアイコン端子はフロントパネルのノブを表します。端子の下にある DBL は、倍精度浮動小数点数のデータタイプを表し

ます。左図に示す DBL 端子は、倍精度浮動小数点数の制御器または表示器を表します。チェックマークを削除して端子のデータタイプを表示するには、端子を右クリックして、ショートカットメニューから**アイコンとして表示**を選択します。フロントパネルオブジェクトのデータタイプに加えて、ブロックダイアグラムにフロントパネルオブジェクトのタイプも表示するには、アイコン端子を使用します。データタイプ端子を使用すると、ブロックダイアグラムのスペースを節約できます。

メモ　アイコン端子はデータタイプ端子よりも大きいので、データタイプ端子をアイコン端子に変換した際、無意識に他のブロックダイアグラムオブジェクトを隠してしまうことがあります。

端子とは、別のワイヤ以外でワイヤを接続できる任意のポイントです。LabVIEW には、制御器および表示器端子、ノード端子、定数、およびフォーミュラノード上の入出力端子などのストラクチャ専用端子があります。ワイヤを使用して端子を接続し、データを他の端子に渡します。端子を表示するには、ブロックダイアグラム上のオブジェクトを右クリックし、ショートカットメニューから**表示項目→端子**を選択します。端子を隠すには、オブジェクトを右クリックし、もう一度**表示項目→端子**を選択します。このショートカットメニュー項目は、拡張可能（ドラッグして端子の数を変更できる）VI および関数では使用できません。

制御器および表示器のデータタイプ

表 5-1 は、さまざまなタイプの制御器端子および表示器端子の記号を示してます。各端子の色と記号は制御器や表示器のデータタイプを示します。制御器端子の枠は表示器端子より太くなっています。また、フロントパネル端子に矢印が表示されて、その端子が制御器か表示器かを示します。端子が制御器の場合は右側に、端子が表示器の場合は左側に矢印が表示されます。

表 5-1　制御器端子および表示器端子

制御器	表示器	データタイプ	色	デフォルト値
SGL	SGL	単精度浮動小数点数	オレンジ	0.0
DBL	DBL	倍精度浮動小数点数	オレンジ	0.0
EXT	EXT	拡張精度浮動小数点数	オレンジ	0.0
CSG	CSG	複素単精度浮動小数点数	オレンジ	0.0 + i0.0
CDB	CDB	複素倍精度浮動小数点数	オレンジ	0.0 + i0.0
CXT	CXT	複素拡張精度浮動小数点数	オレンジ	0.0 + i0.0
I8	I8	8 ビット符号付き整数	青	0

表 5-1　制御器端子および表示器端子（続き）

制御器	表示器	データタイプ	色	デフォルト値
I16	I16	16 ビット符号付き整数	青	0
I32	I32	32 ビット符号付き整数	青	0
U8	U8	8 ビット符号なし整数	青	0
U16	U16	16 ビット符号なし整数	青	0
U32	U32	32 ビット符号なし整数	青	0
		<64.64> ビットタイムスタンプ	茶色	日付と時間（ローカル）
		列挙型	青	—
TF	TF	ブール	緑	FALSE
abc	abc	文字列	ピンク	空の文字列
[ ]	[ ]	配列：要素のデータタイプを角括弧で囲み、そのデータタイプの色で表現します。	データタイプによって異なる	—
		クラスタ：複数のデータタイプを囲みます。クラスタデータタイプは、クラスタのすべての要素が数値の場合は茶色、クラスタの要素が異なるタイプの場合はピンクです。	茶色またはピンク	—
		パス	水色	< 無効パス >
		ダイナミック	青	—
		波形：波形のデータ、開始時間、および Δt を含む要素のクラスタ。波形データタイプの詳細については、第 12 章「グラフおよびチャート」の「波形データタイプ」のセクションを参照してください。	茶色	—
		デジタル波形	深緑	—
0101	0101	デジタルデータ	深緑	—
		リファレンス番号 (Refnum)	水色	—

表 5-1　制御器端子および表示器端子（続き）

制御器	表示器	データタイプ	色	デフォルト値
		バリアント：制御器または表示器の名前、変換したデータタイプの情報、およびデータそのものが含まれています。バリアントデータタイプの詳細については、この章の「バリアントデータを処理する」のセクションを参照してください。	紫	—
I/O	I/O	I/O 名：構成する DAQ チャネル名、VISA リソース名、および IVI 論理名を I/O VI に渡し、計測器や DAQ デバイスと通信します。I/O 名データタイプの詳細については、第 4 章「フロントパネルを作成する」の「I/O 名制御器および表示器」のセクションを参照してください。	紫	—
		ピクチャ：線、円、テキスト、その他の画像形状を含むことができるピクチャを表示します。ピクチャデータタイプの詳細については、第 13 章「グラフィック & サウンド VI」の「ピクチャ表示器を使用する」のセクションを参照してください。	青	—

多くのデータタイプは、データを処理できる一連の関数に対応しています。各データタイプでどの関数を使用するかについては、この章の「関数の概要」のセクションを参照してください。

データタイプを選択してメモリの使用状況を最適化する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Memory and Speed Optimization」のセクションを参照してください。

定数

定数は、ブロックダイアグラムに固定データ値を与えるブロックダイアグラム上の端子です。ユニバーサル定数は、pi (π) や無限 (∞) などの固定値を持つ定数です。ユーザ定義定数は、VI の実行前にユーザが定義して編集する定数です。

定数にラベルを付けるには、定数を右クリックし、ショートカットメニューから**項目を表示→ラベル**を選択します。ユニバーサル定数はあらかじめ決まったラベルが付けられていますが、操作ツールまたはラベリングツールを使用して編集できます。

ほとんどの定数はパレットの一番下か一番上にあります。

ユニバーサル定数

ユニバーサル定数は、数値計算や、文字列またはパスのフォーマットに使用します。LabVIEW には以下のタイプのユニバーサル定数があります。

- **ユニバーサル数値定数**：自然数や光速など、一般的に使用される高精度の数学的および物理的な値のセットです。ユニバーサル数値定数は、**その他の数値定数**パレットにあります。
- **ユニバーサル文字列定数**：改行文字や復帰文字など、一般的に使用される非表示文字列のセットです。ユニバーサル文字列定数は、**文字列**パレットにあります。
- **ユニバーサルファイル定数**：無効パス、Not a Refnum、デフォルトディレクトリなど、一般的に使用されるファイルパス値のセットです。ユニバーサルファイル定数は、**ファイル定数**パレットにあります。

ユーザ定義定数

関数パレットには、ブール値、数値、リング、列挙型、カラーボックス、文字列、配列、クラスタ、パス定数などのタイプによって構成された定数があります。

VI または関数の入力端子を右クリックして、ショートカットメニューから**定数を作成**を選択し、ユーザ定義定数を作成します。VI の実行時には、ユーザ定義定数の値の変更はできません。

また、定数は、フロントパネル制御器をブロックダイアグラムにドラッグして作成することもできます。作成された定数には、フロントパネル制御器をブロックダイアグラムにドラッグしたときの制御器の値が含まれています。フロントパネル制御器はフロントパネル上に残ります。制御器の値を変更しても、定数の値には影響しません。逆に、定数の値を変更しても制御器の値には影響がありません。

操作ツールかラベリングツールを使用して定数をクリックし、値を編集します。自動ツール選択機能が有効になっている場合は、定数をダブルクリックしてラベリングツールに切り替え、値を編集します。

ブロックダイアグラムのノード

ノードとは、入力端子や出力端子を持ち、VI の実行時に演算を実行するブロックダイアグラム上のオブジェクトです。テキストベースのプログラミング言語におけるステートメント、演算子、関数、およびサブルーチンに似ています。LabVIEW には以下のタイプのノードがあります。

- **関数**：ビルトイン実行要素。演算子、関数、またはステートメントに相当します。LabVIEW で使用可能な関数の詳細については、この章の「関数の概要」のセクションを参照してください。

- **サブ VI**：他の VI のブロックダイアグラムで使用される VI。サブルーチンに相当します。ブロックダイアグラムでのサブ VI の使用方法については、第 7 章「VI およびサブ VI を作成する」の「サブ VI」のセクションを参照してください。
- **ストラクチャ**：シーケンスストラクチャ、ケースストラクチャ、For ループ、While ループなどのプロセス制御要素。ストラクチャの使用方法については、第 8 章「ループとストラクチャ」を参照してください。
- **フォーミュラノード（Formula Nodes）**：ブロックダイアグラムに方程式を直接入力するためのサイズ変更可能なストラクチャ。フォーミュラノードの使用方法については、第 21 章「フォーミュラと方程式」の「フォーミュラノード（Formula Nodes）」のセクションを参照してください。
- **数式ノード（Expression Nodes）**：変数を 1 つ含む数式や方程式を計算するストラクチャ。数式ノードの使用方法については、第 21 章「フォーミュラと方程式」の「数式ノード（Expression Nodes）」のセクションを参照してください。
- **プロパティノード（Property Nodes）**：クラスのプロパティを設定または検索するストラクチャ。プロパティノードの使用方法については、第 17 章「VI をプログラム的に制御する」の「プロパティノード」のセクションを参照してください。
- **インボークノード（Invoke Nodes）**：クラスのメソッドを実行するストラクチャ。インボークノードの使用方法については、第 17 章「VI をプログラム的に制御する」の「インボークノード」のセクションを参照してください。
- **コードインタフェースノード（CIN）**：テキストベースのプログラミング言語からコードを呼び出すストラクチャ。テキストベースのプログラミング言語からコードを呼び出す方法の詳細については、第 20 章「テキストベースのプログラミング言語からのコード呼び出し」の「コードインタフェースノード」のセクションを参照してください。
- **リファレンス呼び出しノード （Call By Reference Node）**：ダイナミックにロードした ＶＩ を呼び出すストラクチャ。リファレンス呼び出しノード の使用方法については、第 17 章「VI をプログラム的に制御する」の「リファレンス呼び出しノード（Call By Reference Node）と厳密に類別化された VI Refnum」のセクションを参照してください。
- **ライブラリ関数呼び出しノード（Call Library Nodes）**：最も標準的な共有ライブラリまたは DLL を呼び出すストラクチャ。ライブラリ関数呼び出しノードの使用方法については、第 20 章「テキストベースのプログラミング言語からのコード呼び出し」の「ライブラリ関数呼び出しノード」のセクションを参照してください。

関数の概要

関数は LabVIEW の重要な操作要素です。**関数**パレットにある関数アイコンは、背景色が淡い黄色で、前景色が黒です。関数にはフロントパネルやブロックダイアグラムはありませんが、コネクタペーンがあります。関数を開いたり編集したりすることはできません。

また、**関数**パレットには LabVIEW に標準で付属されている VI も含まれています。これらの VI は、データ集録、計測器制御、通信、およびその他の VI を作成する場合にサブ VI として使用します。標準 VI の使用方法については、第 7 章「VI およびサブ VI を作成する」の「組み込み VI および関数を使用する」のセクションを参照してください。

数値関数

数値関数を使用して、数値について算術演算、三角関数、対数関数、および複素演算を作成または実行したり、数値のデータタイプを変換します。

ブール関数

ブール関数を使用して、1 つのブール値またはブール値の配列について論理演算を実行します。たとえば以下のようなタスクを実行します。

- TRUE 値を FALSE 値に変更したり、FALSE 値を TRUE 値に変更する。
- 複数のブール値を受け取った場合に、どのブール値を返すかを決める。
- ブール値を 1 または 0 の数値に変換する。
- 複数のブール値で複合演算を実行する。

文字列関数

以下のタスクを実行するには、文字列関数を使用します。

- 複数の文字列を連結する。
- 文字列から文字列の一部を抽出する。
- 文字列の文字または部分文字列を検索および置き換える。
- 数値データを文字列に変換する。
- ワープロや表計算アプリケーションで使用するために文字列をフォーマットする。

文字列関数の使用方法については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「文字列」のセクションを参照してください。

配列関数

以下のタスクように複数の配列を作成するには、配列関数を使用します。

- 配列から個々のデータ要素を抽出する。
- 配列に個々のデータ要素を追加する。
- 配列を分割する。

配列関数の使用方法については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「配列」のセクションを参照してください。

クラスタ関数

以下のタスクのように複数のクラスタを作成して操作するには、クラスタ関数を使用します。

- クラスタから個々のデータ要素を抽出する。
- クラスタに個々のデータ要素を追加する。
- クラスタを個々のデータ要素に分割する。

クラスタ関数の使用方法については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「クラスタ」のセクションを参照してください。

比較関数

比較関数を使用して、ブール値、文字列、数値、配列、およびクラスタを比較します。

比較関数の使用方法については、付録 C「比較関数」を参照してください。

時間＆ダイアログ関数

以下のタスクを実行するには、時間＆ダイアログ関数を使用します。

- 演算の実行速度を操作する。
- コンピュータの時計から時刻と日付の情報を取り出す。
- ユーザに指示を与えるダイアログボックスを作成する。

時間＆ダイアログパレットにはエラー処理 VI も含まれています。エラー処理 VI の使用方法については、第 6 章「VI の実行とデバッグ」の「エラーチェックとエラー処理」のセクションを参照してください。

ファイル I/O 関数

以下のタスクを実行するには、ファイル I/O 関数を使用します。

- ファイルを開いたり閉じる。
- ファイルから読み取ったり、ファイルに書き込む。
- パス制御器で指定するディレクトリとファイルを作成する。
- ディレクトリ情報を取り出す。
- 文字列、数値、配列、およびクラスタをファイルに書き込む。

また、**ファイル I/O** パレットには、一般的なファイル I/O タスクを実行する VI も含まれています。ファイル I/O VI および関数の使用方法の詳細について は、第 14 章「ファイル I/O」を参照してください。

波形関数

以下のタスクを実行するには、波形関数を使用します。

- 波形値、チャネル情報、およびタイミング情報を含む波形を作成する。
- 波形から個々のデータ要素を抽出する。
- 波形の個々のデータ要素を編集する。

VI での波形の作成および使用方法については、『LabVIEW Measurements Manual』の Chapter 5「Creating a Typical Measurement Application」を参照してください。

アプリケーション制御関数

アプリケーション制御関数を使用して、ローカルコンピュータ上またはネットワークを介して、プログラム的に VI および LabVIEW アプリケーションを制御します。アプ リケーション制御関数の使用方法については、第 17 章「VI をプログラム的に制御する」を参照してください。

上級関数

ダイナミックリンクライブラリ (DLL) などのライブラリからのコードの呼び出し、他のアプリケーションで使用するための LabVIEW データの操作、Windows レジストリキーの作成と操作、およびテキストベースのプログラミング言語から一部のコードの呼び出しを行うには、上級関数を使用します。上級関数の使用方法については、『Using External Code in LabVIEW』マニュアルを参照してください。

関数に端子を追加する

関数には端子の数を変更できるものがあります。たとえば、10 個の要素を持つ配列を作成するには、「配列連結追加（Build Array）」関数に 10 個の端子を追加する必要があります。

拡張可能な VI および関数に端子を追加するには、位置決めツールを使用してその関数の上枠または下枠をそれぞれ上下にドラッグします。また、位置決めツールを使用して拡張可能な VI および関数から端子を削除することもできますが、既に配線されている端子は削除できません。端子を削除するには、まず既存のワイヤを削除する必要があります。

また、アイコン上のいずれかの端子を右クリックし、ショートカットメニューから**入力端子を追加**、**出力端子を追加**、**入力端子を削除**、または**出力消去**を選択すると、端子を追加または削除できます。関数によっては、入力、出力、または Refnum 制御器の端子を追加できます。**入力端子を追加**および**出力端子を追加**ショートカットメニュー項目を選択すると、右クリックした端子のすぐ後に新しい端子が追加されます。**入力端子を削除**および**出力消去**ショートカットメニュー項目を選択すると、右クリックした端子が削除されます。ショートカットメニュー項目を使用して、配線された端子を削除すると、端子が削除され、ワイヤが切断されます。

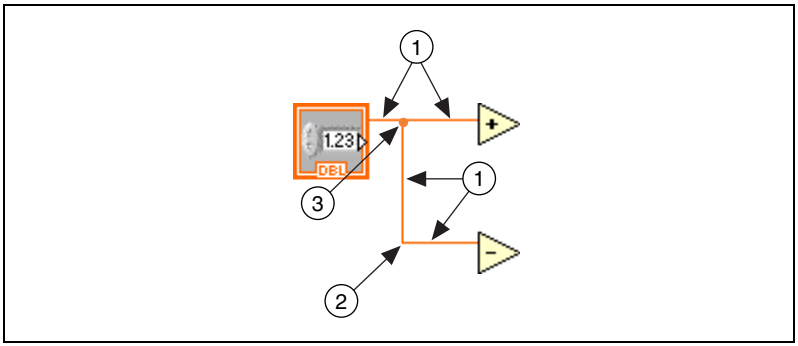
ブロックダイアグラムオブジェクトをワイヤで接続する

ブロックダイアグラムオブジェクト間のデータ転送はワイヤを介して行います。各ワイヤのデータソースは 1 つですが、そのデータを読み取る多くの VI および関数にこのデータソースを配線できます。必要なすべてのブロックダイアグラム端子を配線する必要があります。上記のような手順を行わないと、VI は壊れた状態となり、実行できません。ブロックダイアグラムノードに必要な端子を確認するには、**ヘルプ**ウィンドウを表示してください。必要な端子のラベルは、**ヘルプ**ウィンドウ内に太字で表示されます。必要な端子の詳細については、第 7 章「VI およびサブ VI を作成する」の「必須、推奨、および任意の入出力を設定する」のセクションを参照してください。壊れた VI の詳細については、第 6 章「VI の実行とデバッグ」の「壊れた VI を修正する」のセクションを参照してください。

ワイヤの色、スタイル、および太さは、データタイプによって異なります。不良ワイヤは、真中に赤い x がある黒い破線として表示されます。赤い x のどちらかについた矢印はデータフローの方向を示し、矢印の色はワイヤを流れるデータのデータタイプを示します。ワイヤのデータタイプの詳細については、『LabVIEW クイックリファレンスカード』を参照してください。

ワイヤスタブは、配線ツールを VI または関数ノード上で移動したときに未配線の端子の隣に現れる短いワイヤです。ワイヤスタブは各端子のデータタイプを示します。端子の名前を記述するヒントラベルも表示されます。端子を配線すると、ノード上を配線ツールが移動しても、その端子のワイヤスタブは表示されません。

ワイヤセグメントは水平方向または垂直方向の 1 本のワイヤです。ワイヤの屈折点は 2 つのセグメントの結合点です。2 本以上のワイヤセグメントの結合点を接点といいます。ある接点から別の接点までのセグメント、ある端子から次の接点までのセグメント、端子間に接点がない場合は端子から端子までのセグメントはすべて、1 つのワイヤブランチに含まれます。図 5-1 は、ワイヤセグメント、屈折点、および接点を示しています。



1　セグメント	2　屈折点	3　接点

図 5-1　ワイヤセグメント、屈折点、および接点

端子を配線する際にカーソルを水平方向または垂直方向に移動すると、一度だけワイヤを直角に曲げることができます。ワイヤを複数の方向に曲げるには、マウスボタンをクリックしてワイヤを固定した後、さらに新しい方向にカーソルを移動します。ワイヤを固定して新しい方向に移動するという操作は繰り返し行うことができます。

最後にワイヤを設定したポイントを取り消すには、<Shift> キーを押しながら、ブロックダイアグラム上の任意の場所をクリックします。

（Mac OS）の場合は、<Option> キーを押したままクリックします。
（UNIX および Linux）の場合は、中央マウスボタンをクリックします。

ワイヤを交差させると、最初に描いたワイヤに小さなすき間が表示されて、最初のワイヤが 2 本目のワイヤの下にあることを示します。

注意　ワイヤを交差させると、ブロックダイアグラムが煩雑になり、デバッグが難しくなる場合があります。

配線方法およびそのヒントの詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Wiring Techniques」のセクションを参照してください。

オブジェクトを自動配線する

LabVIEW では、オブジェクトはブロックダイアグラム上に配置したとおりに自動的に配線されます。また、すでにブロックダイアグラム上にあるオブジェクトを自動的に配線することもできます。LabVIEW では、最適な端子が接続され、不適切な端子は接続されません。

選択したオブジェクトをブロックダイアグラム上の別のオブジェクトの近くに移動すると、有効な接続を示すワイヤが一時的に描かれます。マウスボタンを離してブロックダイアグラム上にオブジェクトを配置すると、ワイヤが自動的に接続されます。

位置決めツールを使用してオブジェクトを移動する場合は、スペースバーを押すことによって自動配線機能を切り替えることができます。

関数パレットからオブジェクトを選択したり、既にブロックダイアグラム上にあるオブジェクトを <Ctrl> キーを押したままドラッグしてコピーすると、デフォルトで自動配線機能が有効になります。位置決めツールを使用してすでにブロックダイアグラム上にあるオブジェクトを移動すると、デフォルトで自動配線は無効になります。

（Mac OS）の場合は、<Option> キーを押します。**（Sun）**では <Meta> キーを押します。**（Linux）**では <Alt> キーを使用します。

自動配線を無効にするには、**ツール→オプション**を選択して、プルダウンメニューの上部から**ブロックダイアグラム**を選択します。

オブジェクトを手動配線する

ブロックダイアグラムノードの端子を他のブロックダイアグラムノードの端子に手動で接続するには、配線ツールを使用します。ツールのカーソルポイントは、糸巻きから引き出されたワイヤの先端です。端子上に配線ツールを移動すると、その端子が点滅します。VI または関数の端子上に配線ツールを移動すると、その端子の名前がヒントラベルが表示されます。端子に配線するとワイヤが壊れる可能性がある場合、カーソルが警告マーク付きの糸巻きに変わります。不良ワイヤを配線することもできますが、不良ワイヤを修正しなければ VI は実行できません。不良ワイヤを修正する方法の詳細については、この章の「不良ワイヤを修正する」のセクションを参照してください。

正確にワイヤの接続場所を確認するには、**ヘルプ**ウィンドウを参照してください。VI または関数上にカーソルを移動すると、**ヘルプ**ウィンドウでは VI または関数の各端子のリストが表示されます。**ヘルプ**ウィンドウは、「配列作成（Build Array）」関数などの拡張可能な VI および関数の端子は表示しません。コネクタペーンのオプションの端子を表示するには、**ヘルプ**ウィンドウで**オプションの端子と完全パスを表示**ボタンをクリックします。オプションの端子の詳細については、第 7 章「VI およびサブ VI を作成する」の「必須、推奨、および任意の入出力を設定する」のセクションを参照してください。

ワイヤの経路を調整する

LabVIEW は配線すると自動的にワイヤの経路を見つけます。LabVIEW は、ループやストラクチャなど、ブロックダイアグラム上の既存のオブジェクトにワイヤを配線します。また、配線が複雑にならないように、できる限り直線的にワイヤを配線します。さらに可能であれば、制御器端子から自動ルートされたワイヤは端子の右側から出て、表示器端子に自動ルートされたワイヤは端子の左側に入るようになっています。

ワイヤ上で右クリックし、ショートカットメニューから**配線を整理**を選択することで、自動的に既存ワイヤの経路を調整できます。

一時的に自動ワイヤルーティングを無効にし、手動配線で経路を決定するには、配線を開始した後に <A> キーを押します。再び <A> キーを押すと、自動ワイヤルーティングを有効にすることができます。配線が終了すると、LabVIEW では再び自動ワイヤルーティングが有効になります。配線を開始した後でも、マウスボタンを押しながら他の端子に配線したりポイントを設定してからマウスボタンを放すと、自動ワイヤルーティングを一時的に無効にできます。マウスボタンを放すと、LabVIEW では自動ワイヤルーティングが再び有効になります。

すべての新規ワイヤで自動ワイヤルーティングを無効にするには、**ツール→オプション**を選択して、プルダウンメニューの上部から**ブロックダイアグラム**を選択します。

自動ワイヤルーティングを無効にすると、最初に配線ツールを移動した方向によって、端子を垂直方向または水平方向に配線できます。端子のどこをクリックしても、ワイヤは端子の中心に接続されます。端子をクリックした後に、スペースバーを押すと水平方向と垂直方向が切り替わります。

自動ワイヤルーティングが有効な場合にも、スペースバーを押して水平方向と垂直方向を切り替えることができます。LabVIEW は、新しい方向にワイヤの経路を見つけると、その方向にワイヤの方向を変更します。

ワイヤを選択する

ワイヤを選択するには、位置決めツールを使用してワイヤを 1 回、2 回、または 3 回クリックします。ワイヤを 1 回クリックするとワイヤの 1 つのセグメントが選択されます。ダブルクリックするとブランチが選択されます。3 回クリックすると、ワイヤ全体が選択されます。

不良ワイヤを修正する

不良ワイヤは、真中に赤い x がある黒い破線として表示されます。ワイヤは、データタイプに互換性がない 2 つのオブジェクトを配線しようとした場合など、さまざまな理由で壊れます。不良ワイヤ上に配線ツールを移動すると、ワイヤが壊れた理由を示すヒントラベルが表示されます。また、不良ワイヤ上に配線ツールを移動すると、この情報が**ヘルプ**ウィンドウにも表示されます。**エラーリスト**ウィンドウを表示するには、ワイヤを右クリックしてショートカットメニューから**エラーリスト**を選択します。ワイヤが壊れた理由の詳細については、**ヘルプ**ボタンをクリックしてください。

不良ワイヤを削除するには、位置決めツールでワイヤを 3 回クリックし、<Delete> キーを押します。また、ワイヤを右クリックしてショートカットメニューから**分岐ワイヤを削除**、**分岐ワイヤを作成**、**未接続の配線を削除**、**ワイヤを消去**、**制御器に変更**、**表示器に変更**、**ソースで指標付け使用**、および**ソースで指標付け不使用**などのショートカットメニューのオプションを選択することもできます。これらのオプションは、ワイヤが壊れている理由によって変わります。

不良ワイヤをすべて削除するには、**編集→不良ワイヤを削除**を選択するかまたは <Ctrl-B> キーを押します。**（Mac OS）**の場合は、<Command-B> キーを押します。**（UNIX）**の場合は、<Meta-B> キーを押します。

注意　不良ワイヤをすべて削除するときは注意してください。ブロックダイアグラムの配線が完了していないためにワイヤが壊れているように見えることがあります。

強制ドット

異なる 2 つの数値データタイプを配線すると、警告を示す強制ドットがブロックダイアグラムノードに表示されます。ドットは、ノードに渡された値が異なる表記法に変換されたことを示しています。たとえば、「和（Add）」関数に 2 つの倍精度浮動小数点数を入力するとします。この入力のうちの 1 つを整数に変更すると、強制ドットが「和」関数に表示されます。

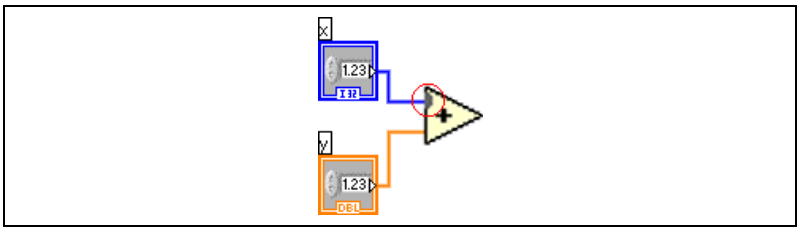


図 5-2　強制ドット

ブロックダイアグラムでは変換が行われた端子の枠に強制ドットが表示されて、自動数値変換が行われたことを示します。VI および関数は多くの端子を持つことができるため、ある端子と別の端子を配線した場合、強制ドットがアイコンの内部に表示されることがあります。

任意のデータタイプをバリアント端子に配線すると、2 つのバリアントを一緒に配線しない限り、端子上に強制ドットが表示されます。バリアントデータタイプの詳細については、この章の「バリアントデータを処理する」のセクションを参照してください。

強制ドットによって、VI ではより多くのメモリが使用されるようになり、実行時間が長くなります。VI では、データタイプを統一するように注意してください。

多形性 VI および関数

多形性 VI および関数は、データタイプの異なる入力に適応します。ほとんどの LabVIEW ストラクチャが多形性であるように、VI および関数の中にも多形性のものがあります。

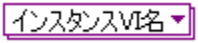
多形性 VI

多形性 VI では、1 つの入力端子または出力端子に異なるデータタイプを受け入れます。多形性 VI は、同じコネクタペーンのパターンを持つサブ VI の集合体です。各サブ VI は、多形性 VI のインスタンスです。

たとえば、「キー読み取り（Read Key）」VI は多形性 VI です。その**デフォルト値**の端子はブール値、倍精度浮動小数点の数値、符号付き 32 ビット整数の数値、パス、文字列、または 32 ビット符号なし整数の数値のデータを受け入れます。

たいていの多形性 VI では、多形性 VI の入力に配線したデータタイプが使用するインスタンスを決定します。そのデータタイプと互換性のあるサブ VI が多形性 VI に含まれていない場合、不良ワイヤが表示されます。多形性 VI の入力に配線したデータタイプによって、使用するインスタンスが

決定されない場合は、インスタンスを手動で選択する必要があります。多形性 VI のインスタンスを手動で選択すると、VI は選択したインスタンスのデータタイプのみを受け入れたり返したりするので、その VI は多形性 VI としては動作しなくなります。



手動でインスタンスを選択するには、多形性 VI を右クリックし、ショートカットメニューから**タイプを選択**して、使用するインスタンスを選択します。操作ツールを使用して、左図に示すような多形性 VI セレクタをクリックし、ショートカットメニューからインスタンスを選択します。ブロックダイアグラムの多形性 VI を右クリックし、ショートカットメニューから**表示項目→多形性 VI セレクタ**を選択して、使用するインスタンスを選択します。処理されたすべてのデータタイプを受け入れるように再度多形性 VI を変更するには、多形性 VI を右クリックしてショートカットメニューから**タイプを選択→自動**を選択するか、操作ツールで多形性 VI セレクタをクリックして、ショートカットメニューから**自動**を選択します。

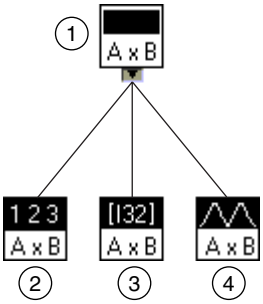
多形性 VI を作成する

異なるデータタイプで同じ操作を実行する場合は、独自の多形性 VI を作成します。

メモ　多形性 VI の作成と編集は、LabVIEW プロフェッショナル開発システムのみで可能です。

たとえば、単精度浮動小数点数、数値の配列、または波形で同じ数値演算を実行する場合は、数値演算、配列演算、および波形演算の 3 つの独立した VI を作成できます。演算を実行する必要がある場合は、入力として使用するデータタイプに応じて、これらの VI のいずれかをブロックダイアグラムに配置します。

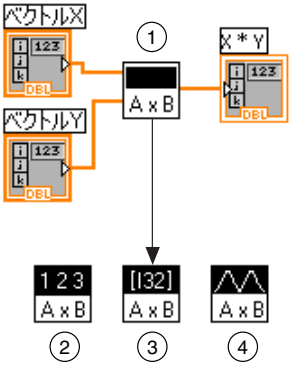
あるバージョンの VI をブロックダイアグラム上に手動で配置する代わりに、1 つの多形性 VI を作成して使用できます。図 5-3 のように、多形性の Compute VI には VI の 3 つのインスタンスが含まれています。



1	演算	3	配列演算
2	数値演算	4	波形演算

図 5-3　多形性 VI の例

Compute VI は、図 5-4 のように、ブロックダイアグラム上で演算サブ VI に配線したデータタイプに基づいて、VI の正しいインスタンスをスタティックにリンクします。



1	演算	3	配列演算
2	数値演算	4	波形演算

図 5-4　サブ VI を呼び出す多形性 VI

多形性 VI は他のほとんどの VI と異なり、ブロックダイアグラムやフロントパネルがありません。

多形性 VI を作成する場合は以下の問題を考慮に入れてください。

- 多形性 VI のコネクタペーンとその多形性 VI の作成に使用する VI のコネクタペーンは一致している必要があるため、多形性 VI に含めるすべての VI は、同じコネクタペーンパターンを持つ必要があります。
- VI の各インスタンスのコネクタペーン上にある入力端子と出力端子は、多形性 VI のコネクタペーン上にある入力端子と出力端子に対応している必要があります。
- 多形性 VI の作成に使用する VI は、同じサブ VI および関数からなっていなくてもかまいません。
- VI の各フロントパネルのオブジェクト数は同じである必要はありません。ただし、各フロントパネルは、多形性 VI のコネクタペーンを構成するのと同じ数以上の制御器および表示器を持つ必要があります。
- 多形性 VI のアイコンを作成できます。
- 多形性 VI を他の多形性 VI の中で使用することはできません。

多形性のサブ VI を含む VI の完全なドキュメントを作成すると、多形性 VI とそのインスタンスがドキュメントのサブ VI リストのセクションに表示されます。

多形性関数

どの入力も多形性でない関数、入力の一部が多形性である関数、または入力のすべてが多形性である関数など、関数の多形性の度合いはさまざまです。関数の入力には、数値またはブール値を受け入れるものがあります。数値や文字列を受け入れるものもあります。また、スカラ数値だけでなく、数値配列、数値クラスタ、数値クラスタの配列などを受け入れるものもあります。また、1 次元配列しか受け入れないものもありますが、配列要素はどのタイプでもかまいません。複素数値を含むすべてのタイプのデータを受け入れる関数もあります。多形性関数の詳細については、付録 B 「多形性関数」を参照してください。

Express VI

Express VI は、一般的な計測タスクに使用します。Express VI は、ダイアログボックスを使用して構成するため、最小限の配線しか必要としない関数ノードです。Express VI の入力および出力は、その VI の構成方法によって異なります。Express VI は、青色のフィールドで囲まれたアイコン付きの拡張可能ノードとして、ブロックダイアグラム上に表示されます。

拡張可能ノードの詳細については、第 7 章「VI およびサブ VI を作成する」の「アイコンまたは拡張可能なノードとしてサブ VI および Express VI を表示する」のセクションを参照してください。Express VI の詳細については、『LabVIEW 入門』を参照してください。

サブ VI として Express VI を作成する

Express VI 構成からサブ VI を作成することができます。たとえば、Express VI を毎回構成し直すのではなく、「LabVIEW 計測ファイル書き込み（Write LabVIEW Measurement File）」Express VI の構成を保存して、構築する他の VI のサブ VI として使用できるようにすることがあります。Express VI を右クリックし、ショートカットメニューから**フロントパネルを開く**を選択して、Express VI からサブ VI を作成します。

Express VI からサブ VI を作成すると、サブ VI のフロントパネルが表示されます。次に、その VI を編集して保存します。各制御器に入力された値を保存するには、**操作→現在のすべての設定をデフォルト設定にする**を選択するか、各制御器を右クリックして、ショートカットメニューから**現在の設定をデフォルト設定にする**を選択します。ブロックダイアグラムには、Express VI の代わりに新しいサブ VI が拡張可能ノードとして表示されます。

Express VI から VI を作成した後に、そのサブ VI を Express VI に戻すことはできません。

ダイナミックデータタイプ

ダイナミックデータタイプは、左図のように、紺色の端子として表示されます。ほとんどの Express VI は、ダイナミックデータタイプを受け取ったり返したりします。ダイナミックデータタイプは、数値、波形、あるいはブール値を受け取る任意の表示器または入力に接続することができます。データを最良な形で表示する表示器へダイナミックデータタイプを書き込んでください。表示器には、グラフ、チャート、または数値表示器などがあります。

ダイナミックデータタイプは Express で使用するものです。LabVIEW に標準で付属されている他のほとんどの VI および関数では、このデータタイプを受け取りません。標準の VI または関数を使用して、ダイナミックデータタイプを含むデータを解析または処理するには、ダイナミックデータタイプを変換する必要があります。ダイナミックデータタイプの変換方法については、この章の「ダイナミックデータから変換する」のセクションを参照してください。

ダイナミックデータタイプには、信号データのほかに、信号名またはデータが集録された日付と時間など、信号についての情報を提供する属性が含まれています。属性はグラフまたはチャートに信号を表示する方法を指定します。たとえば、「DAQ アシスタント（DAQ Assistant）」Express VI を使用し、信号を集録したりグラフに信号をプロットする場合、グラフのプロット凡例に信号名が表示されたり、x 軸を調整して、タイミングの情報が表示されます。このタイミングの情報は、信号属性に基づいた相対または絶対時間単位で信号が関連付けられています。「スペクトル計測（Spectral Measurements）」Express VI を使用して、信号で FFT 解析を

実行したりグラフに結果値をプロットしたりする場合は、x 軸が自動的に調整され、信号の属性に基づいて周波数領域に信号をプロットします。データを数値表示器に表示するには、ブロックダイアグラム上にある VI または関数のダイナミックデータタイプの出力端子を右クリックし、**作成→グラフ表示器**を選択して、データをグラフに表示するか、**作成→数値表示器**を選択します。

表 5-2 は、ダイナミックデータタイプやダイナミックデータタイプに入力できるデータのタイプを受け取る表示器のリストと、表示器のデータ処理方法を示しています。

表 5-2　ダイナミックデータタイプ表示器

ダイナミックデータタイプのデータ	表示器	結果
単精度数値	グラフ	タイムスタンプおよび属性を含む、1 つの値をプロットする
単一チャネル		タイムスタンプおよび属性を含む、波形全体をプロットする
複数チャネル		タイムスタンプおよび属性を含む、すべてのデータをプロットする
単精度数値	数値表示器	1 つの値を表示する
単一チャネル		チャネルデータの最後の値を表示する
複数チャネル		最初のチャネルデータの最後の値を表示する
単精度数値	ブール表示器	数値が .5 以上の場合に TRUE 値を表示する
単一チャネル		チャネルデータの最後の値が .5 以上の場合に TRUE 値を表示する
複数チャネル		最初のチャネルデータの最後の値が .5 以上の場合に TRUE 値を表示する

ダイナミックデータから変換する

「ダイナミックデータからの変換（Convert from Dynamic Data）」Express VI を使用して、ダイナミックデータタイプを数値、波形、および配列データタイプに変換し、他の VI や関数とともに使用できるようにします。「ダイナミックデータからの変換」Express VI をブロックダイアグラムに配置すると、**ダイナミックデータからの変換構成**ダイアログボックスが表示されます。**ダイナミックデータからの変換構成**ダイアログボックスには、「ダイナミックデータからの変換」Express VI が返すデータのフォーマット方法を指定するオプションが表示されます。

たとえば、データ集録デバイスから正弦波を集録する場合、**ダイナミックデータからの変換構成**ダイアログボックスの**シングル波形**オプションを選択します。「ダイナミックデータからの変換」Express VI の**波形**出力を、データタイプを受け入れる関数または VI に配線します。DAQ デバイスを使用してさまざまなチャネルから温度を集めたものを集録する場合は、**1D スカラ配列 - 最新値**および**浮動小数点数（倍精度）**オプションを選択します。「ダイナミックデータからの変換」Express VI の**配列**出力を、入力として数値配列を受け入れる関数または VI に配線します。

ダイナミックデータタイプを配列表示器に配線すると、LabVIEW ではブロックダイアグラム上に自動的に「ダイナミックデータからの変換」Express VI が配置されます。「ダイナミックデータからの変換」Express VI をダブルクリックし、**ダイナミックデータからの変換構成**ダイアログボックスを開いて、配列でのデータの表示方法を制御します。

ダイナミックデータへの変換

「ダイナミックデータへの変換（Convert to Dynamic Data）」Express VI を使用して、数値、波形、および配列のデータタイプを Express VI で使用するダイナミックデータタイプに変換します。ブロックダイアグラム上に「ダイナミックデータへの変換」Express VI を配置すると、**ダイナミックデータへの変換構成**ダイアログボックスが表示されます。このダイアログボックスを使用して、変換するデータの種類を選択し、ダイナミックデータタイプを変換します。

たとえば、従来型 NI-DAQ のアナログ入力 VI を使用して正弦波を集録し、「信号解析（Signal Analysis）」Express VI を使用して信号を解析する場合は、**ダイナミックデータへの変換構成**ダイアログボックスにあるシングル波形オプションを選択します。次に、**ダイナミックデータタイプ**出力を、入力としてダイナミックデータタイプを受け入れる Express VI に配線します。

バリアントデータを処理する

バリアントデータは特定のデータタイプだけに適合するものではなく、また属性を含むこともできます。LabVIEW はバリアントデータタイプによってバリアントデータを表現します。バリアントデータタイプは、制御器名または表示器名、変換する前のデータタイプの情報、およびデータその ものを含んでいるという点で他のデータタイプとは異なります。そのため、LabVIEW ではバリアントデータタイプを希望のデータタイプへ正確に変換することができます。たとえば、文字列データタイプをバリアントデータタイプに変換すると、そのバリアントデータタイプはテキストを格納し、そのテキストが文字列であることを示します。

バリアント関数を使用して、バリアントデータの作成や操作を実行します。任意の LabVIEW のデータタイプをバリアントデータタイプに変換して、バリアントデータを他の LabVIEW の VI や関数で使用できます。いくつかの多形性関数がバリアントデータタイプを返します。

データの転送や格納、不明なデバイスへの読み書き、スタック、キュー、またはノーティファイアでの情報の格納、あるいは異なる制御器のセットでの操作実行など、データタイプのデータを個別に操作することが重要な場合はバリアントデータタイプを使用します。

「文字列に平坦化（Flatten to String）」関数を使用して、データタイプを文字列データタイプに変換し、タイプとは無関係にデータを表すことができます。プロトコルが文字列のみを理解するので、TCP/IP を使用してデータを転送する場合にはデータを文字列に平坦化すると便利です。

ただし、変換するデータタイプに元のデータタイプが一致しない場合、LabVIEW では平坦化データを強制できないので、平坦化データの使用には限界があります。また、平坦化された整数を拡張精度浮動小数点数として非平坦化しようとするとうまくいきません。データの平坦化および非平坦化の詳細については、『LabVIEW Data Storage』アプリケーションノートの「Flattened Data」セクションを参照してください。

バリアントデータを使用するもう 1 つの利点は、データの属性を格納できる機能です。属性とは、バリアントデータタイプに格納されるデータに関する情報です。たとえば、データの 1 つが作成された時間を知りたい場合は、バリアントデータとしてデータを格納し、**時間**という名前で属性を追加して、時間文字列を格納できます。属性データは任意のタイプを入力できます。特定の属性でデータをソートしたり、デバイスやデータを生成したアプリケーションを識別したり、または特定の属性でそのバリアントのみのデータをフィルタ処理する場合には、バリアント属性を使用します。

数値単位および厳密タイプチェック

浮動小数点表記法を持つ数値制御器または表示器には、メートルまたはキロメートル / 秒などの物理的な測定単位を関連付けることができます。

制御器の単位は、単位ラベルと呼ばれる独立した所有ラベルに表示されます。単位ラベルを表示するには、制御器を右クリックし、ショートカットメニューから**表示項目→単位ラベル**を選択します。単位ラベルを編集するには、単位ラベルを右クリックして、ショートカットメニューから**単位文字列の作成**を選択します。

LabVIEW で単位ラベルを表示する場合、メートルは `m`、フィートは `ft`、秒は `s` など、標準の略号を使用して単位を入力できます。

メモ　「フォーミュラノード（Formula Node）」では単位を使用できません。

単位および厳密タイプチェック

オブジェクトに単位を関連付けた場合は、その単位と互換性のある単位を持つオブジェクトにのみ配線できます。LabVIEW では、厳密タイプチェックを使用して単位間に互換性があるかどうかを確認します。単位に互換性がない 2 つのオブジェクトを配線すると、エラーが返されます。たとえば、単位タイプがマイルのオブジェクトを、単位タイプがリットルのオブジェクトに配線するとエラーが返されます。これは、マイルが距離の単位、リットルが量の単位であるためです。

図 5-5 は、単位に互換性があるオブジェクトの配線を示しています。この図では表示器の単位がキロメートルであるため、LabVIEW は**距離**表示器のスケールを自動的に変更して、メートルではなくキロメートルを表示します。

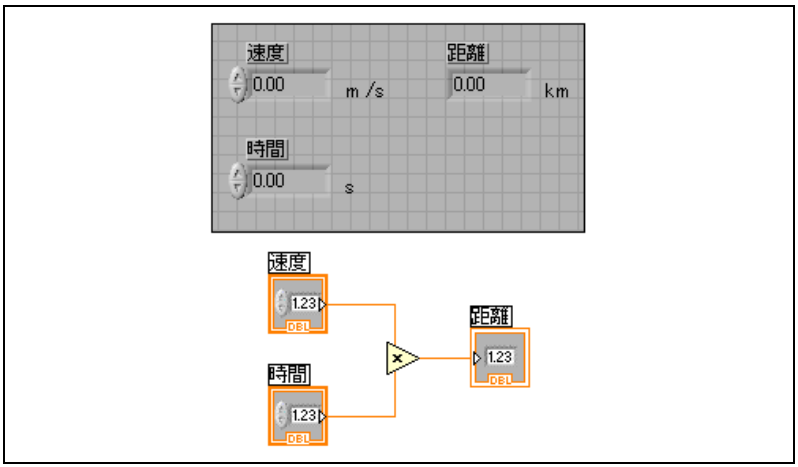


図 5-5　単位に互換性があるオブジェクトの配線

図 5-6 では、**距離**表示器の単位タイプが秒であるため、エラーが発生しています。エラーを修正するには、図 5-5 のように、秒をキロメートルなどの距離の単位に変更します。

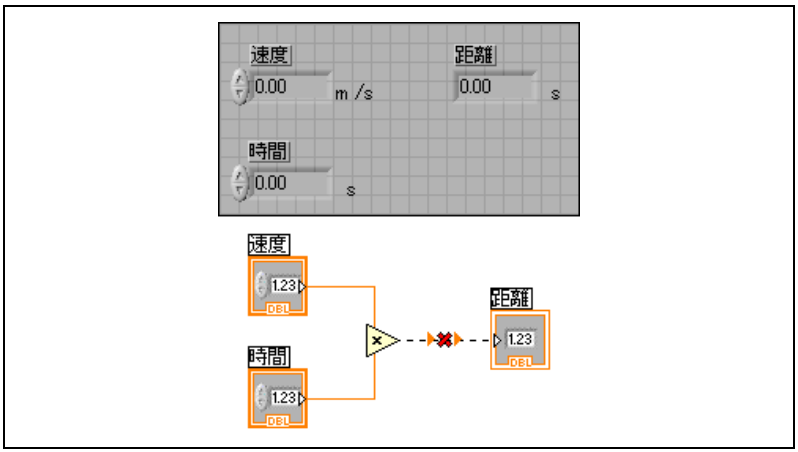


図 5-6　単位に互換性がないオブジェクトの配線による不良ワイヤ

VI および関数には単位に対してあいまいなものもあります。これらの VI および関数は、単位を持つ他の端子では使用できません。たとえば、「インクリメント（Increment）」関数は単位に対してあいまいです。距離単位を使用する場合、「インクリメント」関数は、1 メートル、1 キロメートル、あるいは 1 フィート加算するのか区別できません。このあいまいさのため、「インクリメント」関数や単位に関連付けられたデータを持つ値を増減する他の関数を使用できません。

この例のようなあいまいさを避けるには、図 5-7 のように、適切な単位を持つ数値定数と「和（Add）」関数を使用して、独自のインクリメント単位関数を作成します。

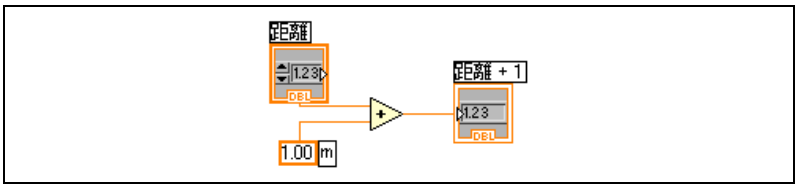


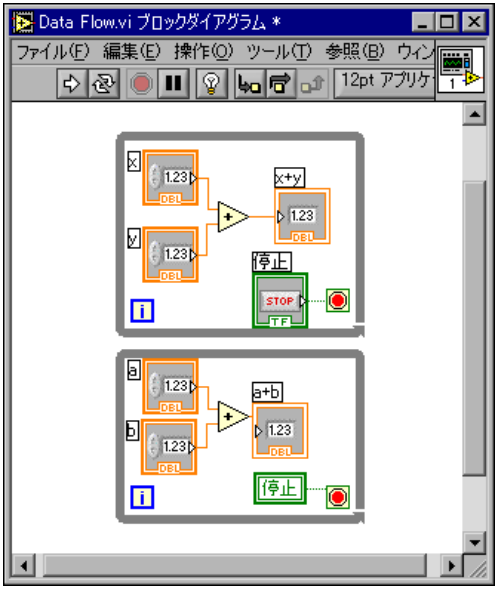
図 5-7　単位を持つ「インクリメント」関数の作成

ブロックダイアグラムのデータフロー

LabVIEW は、データフローモデルに従って VI を実行します。ブロックダイアグラムノードは、そのすべての入力が使用可能なときに実行されます。ノードは実行を終了すると、データをその出力端子に渡し、その出力データをデータフローパスの次のノードに渡します。

Visual Basic、C++、JAVA、その他のほとんどのテキストベースのプログラミング言語は、プログラム実行の制御フローモデルに従います。制御フローでは、プログラム要素の順序によってプログラムの実行順序が決まります。

LabVIEW では、コマンドの順序ではなく、データフローによってブロックダイアグラム内の要素の実行順序が決まるため、並列処理されるブロックダイアグラムを作成できます。たとえば、同時に 2 つの While ループを実行して、フロントパネルにその結果を表示することができます。



LabVIEW はマルチタスクおよびマルチスレッドシステムであるため、複数の実行スレッドと複数の VI を同時に実行します。LabVIEW でタスクを同時に実行する方法の詳細については、『Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability』アプリケーションノートを参照してください。

データ依存と人工データ依存

実行の制御フローモデルは命令駆動です。データフローの実行はデータ駆動、つまりデータに依存しています。他のノードからデータを受け取るノードは、常に他のノードの実行が終了した後で実行します。

ワイヤで接続されていないブロックダイアグラムのノードは、任意の順序で実行できます。『LabVIEW Development Guidelines』マニュアルでは、左から右、上から下にレイアウトすることを推奨していますが、ノードは必ずしも左から右、上から下に実行するわけではありません。

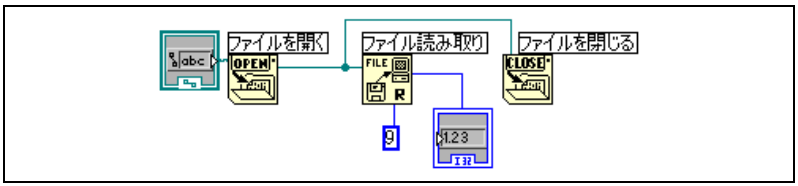
シーケンスストラクチャを使用して、自然なデータ依存が存在しない場合の実行順序を制御します。シーケンスストラクチャの詳細については、第 8 章「ループとストラクチャ」の「シーケンスストラクチャ」のセクションを参照してください。フロースルーパラメータを使用して実行順序を制御することもできます。フロースルーパラメータの詳細については、第 14 章「ファイル I/O」の「フロースルーパラメータ」のセクションを参照してください。

データを受診したノードが実際には受信したデータを使用していない人工データ依存も作成できます。その代わり、受信側ノードはデータの到着によってその実行をトリガします。人工データ依存の使用例については、`examples¥general¥structs.llb` 内の Timing Template (data dep) VI を参照してください。

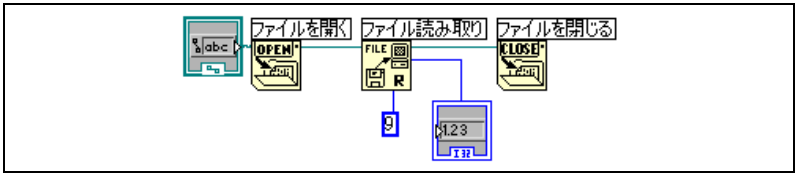
データ依存が存在しない場合

データ依存が存在しない場合は、左から右または上から下へ実行されるとは限りません。必要なときは必ず、データフローを配線することによってイベントのシーケンスを明示的に定義してください。

次の例では、「ファイル読み取り（Read File）」関数が「ファイルを閉じる（Close File）」関数に配線されていないため、「ファイル読み取り」関数と「ファイルを閉じる」関数との間にはデータ依存がありません。この例では、最初に実行される関数が決定できないため、期待どおりに機能しない場合があります。「ファイルを閉じる」関数が最初に実行された場合、「ファイル読み取り」関数は機能しません。



以下のブロックダイアグラムでは、「ファイル読み取り」関数の出力を「ファイルを閉じる」関数に配線することによってデータ依存を確立します。「ファイルを閉じる」関数は、「ファイル読み取り」関数の出力を受け取るまで実行されません。



データフローとメモリ管理

データフロー実行モデルでは、制御フロー実行モデルよりも簡単にメモリを管理できます。LabVIEW では、変数を割り当てたり変数に値を割り当てることはありません。その代わりに、データの移動を表すワイヤを使用してブロックダイアグラムを作成します。

データを生成する VI と関数がそのデータにメモリを自動的に割り当てます。VI や関数がそのデータを使用しなくなると、LabVIEW は関連付けられているメモリの割り当てを解除します。新しいデータを配列や文字列に追加すると、LabVIEW は新しいデータを管理するために十分なメモリを割り当てます。

LabVIEW ではメモリ管理が自動的に処理されるため、メモリの割り当てや割り当て解除の制御はほとんど必要ありません。ただし、VI が大量のデータセットを処理する場合は、メモリの割り当てがいつ行われるかを認識する必要があります。関連する原理を理解していれば、最小限のメモリ使用量で VI を作成できます。メモリ使用を最小限に抑えると、VI の実行速度が向上します。

メモリ割り当ての詳細については、『LabVIEW Performance and Memory Management』アプリケーションノートを参照してください。開発時にメモリ使用を最適化する方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Memory and Speed Optimization」のセクションを参照してください。

ブロックダイアグラムを設計する

ブロックダイアグラムを設計するには、以下のガイドラインに従ってください。

- 左から右、上から下にレイアウトしてください。ブロックダイアグラム内の要素の位置によって実行順序が決まるわけではありませんが、ブロックダイアグラムをわかりやすく配列するために、右から左には配線しないでください。実行順序は、ワイヤとストラクチャのみによって決まります。
- 複数の画面を占めるブロックダイアグラムは作成しないでください。ブロックダイアグラムが大きく複雑になると、わかりにくく、デバッグが困難になる可能性があります。
- ブロックダイアグラム内のコンポーネントを他の VI で再利用できるかどうか、またはブロックダイアグラムの一部を論理的な要素としてまとめることができるかどうかを判断してください。それが可能な場合は、ブロックダイアグラムを、特定のタスクを実行するサブ VI に分割します。サブ VI を使用すると、変更の管理やブロックダイアグラムのデバッグを速やかに実行できます。サブ VI の詳細については、第 7 章「VI およびサブ VI を作成する」の「サブ VI」のセクションを参照してください。
- ブロックダイアグラム内のエラーを管理するには、エラー処理 VI、関数、およびパラメータを使用してください。エラー処理の詳細については、第 6 章「VI の実行とデバッグ」の「エラーチェックとエラー処理」のセクションを参照してください。
- 配線を効率的に行うと、ブロックダイアグラムの外観が向上します。ワイヤが効率よく配線されていなくても、エラーは発生しないかもしれませんが、ブロックダイアグラムが読みにくく、デバッグが困難になり、実際には VI では実行していないことを実行しているような印象を与えてしまう可能性があります。
- ストラクチャの枠の下や重なり合ったオブジェクトの間には配線しないでください。LabVIEW では、そのようなワイヤの一部のセグメントが表示されない場合があります。
- ワイヤの上にオブジェクトを配置しないでください。ワイヤはクリックしたオブジェクトだけに接続されます。ワイヤの上に端子やアイコンを配置すると、接続されていないのに端子やアイコンが接続されているように見えてしまいます。
- ブロックダイアグラム上にコードを記録するには、フリーラベルを使用します。
- 密集したオブジェクト間の間隔を空けるには、<Ctrl> キーを押したまま、位置決めツールでブロックダイアグラムの作業スペースをク

リックします。キーを押したまま、挿入するサイズだけ領域をドラッグアウトします。

（Mac OS）の場合は、<Option> キーを押します。**（Sun）**では <Meta> キーを押します。**（Linux）**では <Alt> キーを押します。

破線の枠が付いた四角形により、スペースが挿入される場所が定義されます。キー操作を解除してスペースを追加します。

ブロックダイアグラムの設計の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Block Diagram Style」のセクションを参照してください。

6

VI の実行とデバッグ

VI を実行するには、端子に対して適切なデータタイプを持つすべてのサブ VI、関数、およびストラクチャを配線する必要があります。VI は、予想外の方法でデータを生成したり、動作する場合があります。LabVIEW を使用すると、VI の実行方法を構成したり、ブロックダイアグラムの構成やブロックダイアグラムで受け渡されるデータに関する問題を確認できます。

詳細については

VI の実行およびデバッグの詳細については、『LabVIEW ヘルプ』を参照してください。

VI を実行する

VI を実行すると、VI を設計した際に目的とした操作が実行されます。左図のように、ツールバーの**実行**ボタンが白く塗りつぶされた矢印で表示されている場合は VI を実行できます。また、白く塗りつぶされた矢印は、VI のコネクタペーンを作成した場合にその VI をサブ VI として使用できることを示します。左図のように、VI の実行中には**実行**ボタンは濃い色の矢印になり、VI が実行中であることを示します。VI の実行中は、この VI を編集することはできません。

ブロックダイアグラムのツールバーの**実行**ボタン、**連続実行**ボタン、またはシングルステップボタンをクリックすると VI が実行されます。**実行**ボタンをクリックすると、VI は一度実行されます。VI は、そのデータフローを完了すると停止します。左図のような**連続実行**ボタンをクリックすれば、手動で停止するまで VI を継続して実行できます。シングルステップボタンをクリックすると、VI は 1 ステップずつ実行されます。シングルステップボタンを使用して VI をデバッグする方法の詳細については、この章の「シングルステップ」のセクションを参照してください。

メモ **実行停止**ボタン を使用して VI を停止することは避けてください。VI のデータフローを完了するか、プログラムで VI を停止する方法を設計してください。そうすることにより VI が既知の状態となります。たとえば、VI を停止するボタンをフロントパネルに配置します。

VI の実行方法を構成する

VI の実行方法を構成するには、**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから実行を選択します。たとえば、VI を開くとすぐに実行するように構成したり、サブ VI として呼び出されると一時停止するように構成することができます。また、さまざまな優先順位で実行するように VI を構成できます。たとえば、他の操作が完了するのを待たずに VI を実行する必要がある場合、時間重視（最高）の優先順位で実行するように VI を構成します。マルチスレッド VI の作成については、『Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability』アプリケーションノートを参照してください。VI の実行方法の構成の詳細につていは、第 16 章の「VI をカスタマイズする」を参照してください。

メモ　異なる優先順位で実行するように VI を誤って構成すると、VI が予想外の動作をすることがあります。異なる優先順位で実行するのに、ほとんどの VI では構成を必要としません。

壊れた VI を修正する

VI が実行されない場合、その VI は壊れています。つまり、実行不可能な VI です。作成または編集した VI にエラーがある場合、左図のように**実行**ボタンが壊れた状態で表示されることがよくあります。ブロックダイアグラムの配線を終了してもこのボタンが壊れた状態である場合は、その VI は壊れているため実行できません。

壊れた VI の原因を調べる

VI が壊れている原因を調べるには、壊れた**実行**ボタンをクリックするか、**ウィンドウ→エラーリストを表示**を選択します。**エラーリスト**ウィンドウにすべてのエラーがリストされます。**VI リスト**セクションには、エラーが発生したメモリ内のすべての VI の名前がリストされます。**エラーと警告**セクションには、**VI リスト**セクションで選択した VI に関するエラーと警告がリストされます。**詳細**セクションにエラーが記述され、場合によってはエラーを修正したり、エラーの詳細情報を調べるための推奨方法が示されます。LabVIEW エラーとそれらの説明をリストするオンラインヘルプファイルを開くには、**ヘルプ**を選択します。

エラーを表示ボタンをクリックするか、またはエラーの説明をダブルクリックすると、関連するブロックダイアグラムまたはフロントパネルが表示され、エラーを含むオブジェクトがハイライトされます。

VI に警告が含まれていて、**エラーリスト**ウィンドウ内の**警告を表示**チェックボックスがオンになっている場合、ツールバーに左図のような**警告**ボタンが表示されます。

常に**エラーリスト**ウィンドウに警告を表示するように LabVIEW を構成するには、**ツール→オプション**を選択し、一番上のプルダウンメニューから**デバッグ**を選択して、**デフォルトとして警告をエラーボックス内に表示する**チェックボックスをオンにします。**エラーリスト**ウィンドウを開いたままこの変更を行うと、この変更をすぐに表示できます。

警告が表示されても、VI の実行は妨げられません。これらの警告は、VI 内の潜在的な問題を避ける目的で設けられています。

壊れた VI の一般的な原因

VI の編集時に VI が壊れる一般的な理由を以下にリストします。

- ブロックダイアグラムに、データタイプの不一致や未接続配線による壊れたワイヤが含まれている。ブロックダイアグラムオブジェクトの配線方法の詳細については、第 5 章「ブロックダイアグラムを作成する」の「ブロックダイアグラムオブジェクトをワイヤで接続する」のセクションを参照してください。
- 必要なブロックダイアグラムの端子が配線されていない。必要な端子の詳細については、第 7 章「VI およびサブ VI を作成する」の「必須、推奨、および任意の入出力を設定する」のセクションを参照してください。
- サブ VI が壊れているか、あるいは VI のブロックダイアグラム上にそのアイコンを配置した後でそのコネクタペーンを編集した。サブ VI の詳細については、第 7 章の「VI およびサブ VI を作成する」の「サブ VI」のセクションを参照してください。

デバッグ方法

VI は壊れていないのに、予想外のデータが得られた場合、以下の方法で VI またはブロックダイアグラムのデータフローの問題を確認し、修正できます。

- ほとんどの標準 VI や標準関数の一番下にあるエラー入力およびエラー出力パラメータを配線します。これらのパラメータはブロックダイアグラム上の各ノードで発生したエラーを検出し、エラーが発生したがどうかとその発生場所を示します。作成した VI 内でこれらのパラメータを使用することもできます。これらのパラメータの使用方法の詳細については、この章の「エラー処理」のセクションを参照してください。

- すべての VI の警告を削除し、**ウィンドウ→エラーリストを表示**を選択し、**警告を表示**チェックボックスをオンにして、VI のすべての警告を表示します。
- 操作ツールでワイヤを 3 回クリックしてそのパス全体をハイライトし、ワイヤが適切な端子に接続されていることを確認します。
- **ヘルプ**ウィンドウを使用して、ブロックダイアグラムにある各関数およびサブ VI の初期値をチェックします。推奨またはオプションの入力が未接続であると、VI および関数がデフォルト値を渡します。たとえば、未配線の場合、ブールの入力が TRUE に設定されていることがあります。
- **検索**ダイアログボックスを使用してサブ VI、文字列、および他のオブジェクトを検索し、アプリケーション全体を修正します。
- **参照→ VI 階層を表示**を選択し、未配線のサブ VI を検索します。未配線の関数とは異なり、未配線の VI は入力を必須としてそれぞれ構成していない限り、必ずしもエラーは発生しません。ブロックダイアグラム上に未配線のサブ VI を誤って配置すると、ダイアグラム実行時にそのサブ VI も実行されます。その結果、アプリケーションが余分な動作を実行してしまう可能性があります。
- ブロックダイアグラム内でのデータの移動を監視するには、実行のハイライトを使用します。
- ブロックダイアグラム上で VI の各動作を表示するには、VI をシングルステップにします。
- プローブツールを使用して、データの中間値を調べ、VI や関数 (特に I/O を実行する VI や関数) のエラー出力をチェックします。
- シングルステップを実行したり、プローブを挿入するために実行を一時停止するには、ブレークポイントを使用します。
- 制御器や表示器の値を編集したり、実行回数を制御したり、サブ VI の実行の開始時点に戻るには、サブ VI の実行を中断します。
- ブロックダイアグラムのある部分がない方が VI のパフォーマンスが向上するかどうかを調べるには、そのセクションをコメントアウトします。
- ある関数またはサブ VI から渡しているデータが不定かどうかを調べます。これは数値の場合に頻繁に発生します。たとえば、VI のあるポイントで数字をゼロで割った場合、後続の関数やサブ VI では数値が返されることを予期していたのにもかかわらず **`Inf`**(無限) が返されます。
- VI の実行速度が予想よりも遅い場合は、サブ VI の実行のハイライトがオフになっているかどうかを確認します。また、ウィンドウを開いておくと実行速度に影響を及ぼすので、使用していないサブ VI のフロントパネルとブロックダイアグラムは閉じておきます。

- 制御器や表示器の表記法をチェックし、浮動小数点数を整数に変換したり、整数をそれより小さい整数に変換することによってオーバーフローが返されていないかを確認します。たとえば、8 ビット整数しか受け入れない関数に 16 ビット整数を配線したとします。これによって関数は 16 ビット整数を 8 ビットの表記法に変換しますが、潜在的にデータの損失が発生します。
- For ループが誤って反復回数 0 で実行されたために空の配列が生成されていないかどうかを確認します。
- ループのある実行から別の実行までデータを保存するようにシフトレジスタを設定していない場合は、シフトレジスタを正しく初期化したかどうかを確認します。
- 接続元および接続先のクラスタ要素の順番を確認します。LabVIEW では、編集時にデータタイプおよびクラスタサイズの不一致が検出されますが、同じタイプの要素の不一致は検出されません。
- ノードの実行順序を確認します。
- 隠れたサブ VI がないことを確認します。あるサブ VI を別のノードの上に直接ドロップしたり、サブ VI を表示しないでストラクチャのサイズを縮小すると、サブ VI を誤って隠してしまうことがあります。
- さらに、VI で使用しているサブ VI のインベントリを、**参照→この VI のサブ VI** および**参照→開かれていないサブ VI** の結果と比較して、余分なサブ VI がないかどうかを調べます。また、**階層**ウィンドウを開いて VI のサブ VI を調べます。VI に対する入力を必須として指定すると、隠れた VI による不正な結果を避けることができます。

実行のハイライト

ブロックダイアグラムの実行を動画表示するには、左図に示す**実行のハイライト**ボタンをクリックします。実行のハイライトは、ワイヤに沿って移動するバブルを使用して、ブロックダイアグラム上のノードからノードへのデータ移動を示します。VI 全体でノードからノードにデータがどのように移動するかを調べるには、シングルステップとともに実行のハイライトを使用します。

メモ　実行のハイライトを使用すると、VI の実行速度が大幅に低下します。

エラーアウトクラスタがエラーをレポートすると、エラー値が赤で**エラーアウト**の横に表示されます。エラーが発生していない場合は、**エラーアウト**の横に OK が緑で表示されます。エラークラスタの詳細については、この章の「エラークラスタ」のセクションを参照してください。

シングルステップ

VI 実行時にブロックダイアグラム上の VI の各動作を表示するには、VI をシングルステップで実行します。左図に示すシングルステップボタンが機能するのは、シングルステップモードでの VI またはサブ VI 内の実行に限られます。シングルステップモードに切り替えるには、ブロックダイアグラムのツールバーにある**飛び越える**または**中に入る**ボタンをクリックします。**飛び越える**、**中に入る**、または**外に出る**ボタン上にカーソルを置くと、そのボタンをクリックした場合の次のステップを記述したヒントラベルが表示されます。サブ VI では、シングルステップで実行するか通常どおりに実行できます。

実行のハイライトをオンにした状態で VI をシングルステップで実行する場合は、左図のような実行グリフが現在実行中のサブ VI のアイコン上に表示されます。

プローブツール

VI 実行時にワイヤ上の中間値を確認するには、左図に示すプローブツールを使用します。一連の操作が含まれている複雑なブロックダイアグラムを作成し、そのいずれかの操作が誤ったデータを返す可能性がある場合にプローブツールを使用します。実行のハイライト、シングルステップ、ブレークポイントとともにプローブツールを使用して、データに誤りがないか、またその場所を確認します。データが利用可能な場合、シングルステップの実行時またはブレークポイントで一時停止したときにプローブはすぐに更新されます。シングルステップまたはブレークポイントのために実行がノードで一時停止している場合は、実行直後のワイヤにプローブを挿入して、そのワイヤを通った値を確認することもできます。

プローブのタイプ

VI の実行時にワイヤ上の中間値をチェックするには、一般プローブを使用するか、**制御**パレットにある表示器を使用してデータを表示するか、指定されたプローブを使用するか、指定されたカスタムプローブを使用するか、あるいは新規プローブを作成します。

一般

一般プローブを使用して、ワイヤを介して渡されたデータを表示します。一般プローブを使用するには、ワイヤを右クリックして、ショートカットメニューから**カスタムプローブ→一般プローブ**を選択します。

一般プローブはデータを表示します。データに応答するように一般プローブを構成することはできません。

データタイプをカスタムまたは指定されたプローブにしない限り、LabVIEW ではワイヤを右クリックして**プローブ**を選択すると一般プローブが表示されます。

VI と同じようにしてカスタムプローブをデバッグすることができます。ただし、プローブはそのプローブが表示されているブロックダイアグラムやそのサブ VI のブロックダイアグラムを調査することはできません。プローブをデバッグする場合は、一般プローブを使用してください。

表示器を使用してデータを表示する

表示器を使用して、ワイヤを介して渡されるデータを表示することもできます。たとえば、数値データを表示する場合、プローブ内でチャートを使用してデータを表示できます。ワイヤを右クリックして、ショートカットメニューから**カスタムプローブ→制御器**を選択し、使用する表示器を選択します。**制御器**パレット上の**制御器を選択**をクリックし、コンピュータまたはサーバ上の共有ディレクトリに保存されたカスタム制御器またはタイプ定義を選択します。LabVIEW は、プローブを挿入したデータを表示するために使用されたタイプ定義をカスタム制御器として扱います。

選択した表示器のデータタイプが、右クリックしたワイヤのデータタイプと一致しない場合、表示器はワイヤ上に配置されません。

指定された

指定されたプローブは、ワイヤを介して渡されるデータに関する総合的な情報を表示する VI です。たとえば、VI Refnum プローブは、リファレンスの VI 名、VI パス、および 16 進数値に関する情報を返します。また、指定されたプローブを使用すると、ワイヤを通過するデータに基づいて応答することができます。たとえば、エラークラスタ上のエラープローブを使用して、エラーの状態、コード、ソース、および説明を受け取り、エラーまたは警告が発生した場合に条件付きブレークポイントを設定するかどうかを指定することができます。

指定されたプローブは、**カスタムプローブ**ショートカットメニューの一番上に表示されます。ワイヤを右クリックして、ショートカットメニューから**カスタムプローブ**を選択し、指定されたプローブを選択します。右クリックしたワイヤのデータタイプと一致するプローブのみがショートカットメニューに表示されます。

指定されたプローブのサンプルについては、`examples¥general¥probes.llb` の Using Supplied Probes VI を参照してください。

カスタム

カスタムプローブウィザードを使用して、既存プローブに基づいてプローブを作成するか、新規プローブを作成します。カスタムプローブウィザードを表示するには、ワイヤを右クリックして、ショートカットメニューから**カスタムプローブ→新規**を選択します。ワイヤを通過するデータのプローブ方法に対してより自由に制御したい場合に、プローブを作成します。新規プローブを作成すると、プローブのデータタイプと右クリックしたワイヤのデータタイプが一致します。作成したプローブを編集する場合、保存したディレクトリからプローブを開く必要があります。

カスタムプローブショートカットメニューからプローブを選択するか、**制御器を選択**パレットオプションを使用してそのプローブに移動するか、あるいはカスタムプローブウィザードを使用して新規プローブを作成した後、そのプローブがそのデータタイプのデフォルトプローブとなり、ワイヤを右クリックしてショートカットメニューから**プローブ**を選択すると、そのプローブがロードされます。LabVIEW では右クリックしたワイヤのデータタイプと完全に一致するプローブのみをロードします。つまり、データの変換はできますが、倍精度小数点数プローブは符号なし 32 ビット整数ワイヤをプローブすることはできません。

メモ　特定のデータタイプに対してカスタムプローブをデフォルトのプローブにしたい場合は、`user.lib¥_probes¥default` ディレクトリにプローブを保存します。`vi.lib¥_probes` ディレクトリにはプローブを保存しないでください。ファイルをアップグレードまたは再インストールすると上書きされます。

カスタムプローブを使用したり作成したりする際の注意事項については、『LabVIEW ヘルプ』を参照してください。

ブレークポイント

ブロックダイアグラム上の VI、ノード、またはワイヤにブレークポイントを配置し、その位置で実行を一時停止するには、左図に示すブレークポイントツールを使用します。ワイヤ上にブレークポイントを設定すると、そのワイヤからデータが渡された後で実行が一時停止します。ブロックダイアグラム上のすべてのノードの実行後に一時停止するようにするには、ブロックダイアグラムにブレークポイントを配置します。

VI がブレークポイントで一時停止すると、LabVIEW はブロックダイアグラムを表示し、ブレークポイントを含むノードまたはワイヤをマーキーでハイライトします。ブレークポイント上にカーソルを移動すると、ブレークポイントツールのカーソルの黒色部分が白色で表示されます。

実行中にブレークポイントに到達すると VI は一時停止し、**一時停止**ボタンは赤になります。以下のことを実行できます。

- シングルステップボタンを使用して実行をシングルステップする。
- ワイヤにプローブを置いて中間値をチェックする。
- フロントパネルの制御器の値を変更する。
- **一時停止**ボタンをクリックして、次のブレークポイントまで、または VI が実行を停止するまで実行を続ける。

LabVIEW は VI とともにブレークポイントを保存しますが、VI を実行する場合にのみブレークポイントはアクティブになります。**参照→ブレークポイント**を選択して、すべてのブレークポイントを表示することができます。

実行を中断する

制御器や表示器の値を編集したり、呼び出し側 VI に返すまでの実行回数を制御したり、サブ VI の実行開始時点に戻るには、サブ VI の実行を中断します。実行を中断した状態でサブ VI へのすべての呼び出しを開始したり、サブ VI への特定の呼び出しを中断することができます。

サブ VI へのすべての呼び出しを中断するには、サブ VI を開き、**操作→呼び出されたら中断する**を選択します。他の VI がその VI を呼び出すと、サブ VI は自動的に中断されます。シングルステップの実行中にこのメニュー項目を選択した場合は、サブ VI はすぐには中断されません。サブ VI は呼び出されると中断されます。

特定のサブ VI の呼び出しを中断するには、ブロックダイアグラム上のサブ VI ノードを右クリックし、ショートカットメニューから**サブ VI ノード設定**を選択します。サブ VI のそのインスタンスだけで実行を中断するには、**呼び出されたら中断する**チェックボックスをオンにします。

参照→ VI 階層を表示を選択することによって表示される**階層**ウィンドウは、VI が一時停止または中断されているかどうかを示します。矢印グリフは、通常どおりに実行している VI またはシングルステップで実行している VI を示します。一時停止グリフは、一時停止または中断されている VI を示します。緑の一時停止グリフ、または白黒の場合中抜きのグリフは、呼び出されたら一時停止する VI を示します。赤の一時停止グリフ、または白黒の場合塗りつぶされたグリフは、現在一時停止している VI を示します。感嘆符グリフはサブ VI が中断されていることを示します。VI を同時に中断または一時停止できます。

サブ VI の現在のインスタンスを調べる

サブ VI を一時停止すると、ツールバーの**呼び出しリスト**プルダウンメニューが、最上位 VI からサブ VI に至るまでの呼び出し側 VI のチェーンをリストします。このリストは、**参照→この VI の発呼者**を選択すると表示されるリスト（VI が現在実行中であるかどうかにかかわらず、すべての呼び出し側 VI をリストで表示）とは異なります。ブロックダイアグラムに複数のインスタンスが含まれている場合にサブ VI の現在のインスタンスを調べるには、**呼び出しリスト**メニューを使用します。**呼び出しリスト**メニューから VI を選択すると、そのブロックダイアグラムが開き、現在のサブ VI のインスタンスがハイライトされます。

ブロックダイアグラムのセクションをコメントアウトする

ブロックダイアグラムのセクションを無効にした状態で、VI を実行することができます。これは、テキストベースのプログラミング言語でコードのセクションをコメントアウトすることに似ています。ある部分がない方が VI のパフォーマンスが向上するかどうかを調べるには、ブロックダイアグラムのそのセクションを無効にします。

無効にするセクションをケースストラクチャ内に入れ、ブール定数を使用して両方のケースを実行します。ケースストラクチャの詳細については、第 8 章「ループとストラクチャ」の「ケースストラクチャ」のセクションを参照してください。VI のコピーを作成し、コピーからブロックダイアグラムのそのセクションを削除することもできます。使用しない方の VI は破棄します。

デバッグツールを無効にする

デバッグツールを無効にすると、メモリの必要条件が低減されるためパフォーマンスがわずかに向上します。コネクタペーンを右クリックし、**VI プロパティ**を選択します。**カテゴリ**プルダウンメニューの**実行**を選択し、**デバッギングを許可**チェックボックスをオフにします。

不定データまたは予想外のデータ

不定データ、すなわち `NaN`（数値以外）または `Inf`（無限大）は、後続のすべての操作を無効にします。浮動小数点演算は、誤った計算や意味のない結果を示す以下の 2 つの記号値を返します。

- `NaN`（数値以外）は、負の数値の平方根を求めるようにな無効な演算が生成する浮動小数点値を表します。
- `Inf`（無限大）は、0 で数値を除算するといった演算が生成する浮動小数点値を表します。

LabVIEW は、整数値がオーバーフローまたはアンダーフローの状態になっているかどうかを確認しません。浮動小数点数のオーバーフローおよびアンダーフローについては、IEEE 754 の「Standard for Binary Floating-Point Arithmetic」を参照してください。

浮動少数点演算は、NaN および Inf を忠実に伝達します。NaN や Inf を明示的または暗示的に整数またはブール値に変換すると、値は意味のない値になります。たとえば、1 を 0 で割ると Inf になります。Inf を 16 ビット整数に変換すると、32,767 となり、通常の値のように見えます。数値の変換方法の詳細については、付録 B「多形性関数」の「数値変換」のセクションを参照してください。

データを整数データタイプに変換する前に、中間の浮動小数点値の有効性をチェックするには、プローブツールを使用します。「数値 / パス /Refnum ではない？（Not A Number/Path/Refnum?）」比較関数を、有効性が疑わしい値に配線することによって、NaN がないかをチェックします。

ループのデフォルトデータ

For ループは、繰り返し回数が 0 のとき、デフォルトデータを返します。

データタイプのデフォルト値の詳細については、第 5 章「ブロックダイアグラムを作成する」の「制御器および表示器のデータタイプ」のセクションを参照してください。

For ループ

For ループのカウント端子に 0 を配線した場合や、自動指標付けが有効になった状態で入力として空の配列を For ループに配線した場合、For ループはデフォルトデータを生成します。ループは実行せず、自動指標付けが無効になった状態の出力

For ループ、自動指標付け、およびシフトレジスタの詳細については、第 8 章「ループとストラクチャ」の「For ループおよび While ループのストラクチャ」のセクションを参照してください。

配列内のデフォルトデータ

配列の境界を越えて指標付けすると、配列要素パラメータのデフォルト値が生成されます。配列のサイズを調べるには、「配列サイズ（Array Size）」関数を使用します。配列の詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「配列」のセクションを参照してください。指標付けの詳細については、第 8 章「ループとストラクチャ」の「ループに自動指標付けを行う」のセクションを参照してください。While ループを使用して最後の要素を超えて配列を指標付けし

たり、「インデックス配列（Index Array）」関数の**指標**入力に大きすぎる値を入力したり、「インデックス配列」関数に空の配列を指定すると、誤って配列の範囲を超えて指標付けする可能性があります。

不定データを防ぐ

VI が不定データを生成するかどうかを調べるのに、`NaN`、`Inf`、空の配列などの特別な値ばかりを使用してないでください。その代わりに、不定データが生成されそうな状況に直面したときに VI にエラーを報告させることによって、VI が確定データを生成することを確認します。

たとえば、入力配列を使用して For ループを自動的に指標付けする VI を作成する場合は、入力配列が空のときに VI がどのように動作するかを決めます。出力エラーコードを生成するか、あるいはループが生成するデータを確定データに置き換えます。

エラーチェックとエラー処理

各エラーには数値によるコードとそれに対応するエラーメッセージがあります。デフォルトでは、LabVIEW は VI の実行中にエラーが発生すると実行を中断し、エラーの発生したサブ VI または関数をハイライトして、**エラー**ダイアログボックスを表示します。

特定の VI の自動エラー処理を無効にするには、**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**実行**を選択します。**ツール→オプション**を選択し、一番上のプルダウンメニューから**ブロックダイアグラム**を選択して、新規に作成した VI の自動エラー処理を無効にします。

サブ VI または関数で発生したエラーを無視するには、サブ VI または関数の**エラー出力**出力を「エラークリア（Clear Errors）」VI の**エラー入力**入力に配線します。サブ VI または関数の自動エラー処理を無効にするには、**エラー出力**パラメータを、他のサブ VI または関数の**エラー入力**パラメータに配線するか、**エラー出力**表示器に配線します。

エラーを管理するには、LabVIEW エラー処理 VI、関数、およびパラメータを使用します。たとえば、LabVIEW がエラーを検出した場合は、ダイアログボックスにエラーメッセージを表示できます。デバッグツールとともにエラー処理機能を使用して、エラーを検出し、管理します。ナショナルインスツルメンツでは、エラー処理を行うことを強くお勧めします。

エラーをチェックする

作成した VI に信頼性がある場合でも、直面するすべての問題を予測することはできません。エラーをチェックするメカニズムがないと、VI が正しく機能しないということしかわかりません。エラーチェック機能があると、エラーが発生した理由と場所がわかります。

どのような入出力（I/O）動作を行う場合も、エラーが発生する可能性を考慮する必要があります。通常、I/O 関数はエラー情報を返します。特に I/O 動作（ファイル、シリアル、計測器、データ集録、および通信）に対して、VI 内にエラーチェック機能を組み込み、エラーを正しく処理するメカニズムを装備してください。

VI 内のエラーをチェックすると、以下の問題を確認できます。

- 通信を間違えて初期化したか、あるいは外部機器に不適切なデータを書き込んだ場合。
- 外部機器の電源が入っていないか、故障しているか、あるいは正しく機能していない場合。
- オペレーティングシステムのソフトウェアをアップグレードした際に、ファイルへのパスや、VI やライブラリの機能が変更された場合。VI またはシステムプログラムの問題に気付くはずです。

エラー処理

デフォルトでは、LabVIEW は実行を中断し、自動的にエラーを処理します。ただし他のエラー処理方法を選択することもできます。たとえば、ブロックダイアグラム内の I/O VI がタイムアウトになっても、アプリケーション全体を停止させたくない場合があります。また、一定時間、再実行したい場合もあります。LabVIEW では、このようなエラー処理の判断をブロックダイアグラム内で行えます。

VI と関数は、数値によるエラーコードかエラークラスタのいずれかの方法でエラーを返します。関数は通常数値によるエラーコードを使用し、VI は通常エラー入出力とともにエラークラスタを使用します。エラークラスタの詳細については、この章の「エラークラスタ」のセクションを参照してください。

LabVIEW 内のエラー処理はデータフローモデルに従います。データが VI 内を移動する場合と同様に、エラー情報も VI 全体を移動できます。エラー情報は VI の最初から最後まで配線します。エラーが発生せずに VI が実行されたかどうかを調べるには、VI の最後にエラー処理 VI を組み込みます。使用または作成する各 VI 内で**エラー入力**と**エラー出力**クラスタを使用して、VI 内でエラー情報を渡します。

VI の実行時、LabVIEW は各実行ノードでエラーがないかをテストします。エラーが検出されなかった場合、ノードは正常に実行されます。エラーが検出された場合、コード部分は実行されずにノードは次のノードにエラーを渡します。次のノードは同じ動作を繰り返します。実行フローの最後にエラーが報告されます。

エラークラスタ

エラー入力クラスタおよび**エラー出力**クラスタには以下の情報のコンポーネントが含まれています。

- **ステータス**はブール値であり、エラーが発生した場合に TRUE を報告します。ほとんどの VI、関数、およびブールデータを受け取るストラクチャは、このパラメータを認識します。たとえば、エラークラスタを「停止（Stop）」、「LabVIEW 終了（Quit LabVIEW）」、または「セレクト（Select）」関数のブール入力に配線することができます。エラーが発生した場合、エラークラスタは関数に TRUE の値を渡します。
- **コード**は符号付き 32 ビット整数で、番号でエラーを識別します。**ステータス**の FALSE を伴う 0 以外のエラーコードは、致命的なエラーでなく警告であることを知らせます。
- **ソース**はエラーが発生した場所を識別する文字列です。

エラー処理に While ループを使用する

エラークラスタを While ループの条件端子に配線すると、While ループの繰り返しを停止できます。エラークラスタを条件端子に配線すると、エラークラスタの**ステータス**パラメータの TRUE または FALSE の値だけが端子に渡されます。エラーが発生すると、While ループは停止します。

条件端子にエラークラスタを配線すると、ショートカットメニュー項目の **True の場合停止**および **True の場合、継続**は、**エラーで停止**および**エラーの場合、継続**に変わります。

エラー処理にケースストラクチャを使用する

ケースストラクチャのセレクタ端子にエラークラスタを配線すると、ケースセレクタのラベルには 2 つのケース、すなわちエラーとエラーなしが表示されます。ケースストラクチャの枠の色は、エラーの場合は赤にかわり、エラーなしの場合は緑に変わります。エラーが発生すると、ケースストラクチャはエラーサブダイアグラムを実行します。ケースストラクチャの使用の詳細については、第 8 章「ループとストラクチャ」の「ケースストラクチャ」のセクションを参照してください。

7

VI およびサブ VI を作成する

フロントパネルとブロックダイアグラムの作成方法を習得すると、独自の VI やサブ VI を作成したり、VI を配布したり、スタンドアロンアプリケーションおよび共有ライブラリを作成できます。

開発時の一般的な落とし穴やプロジェクトの開発に使用できるツールの情報など、プロジェクトの計画の詳細については、『LabVIEW Development Guidelines』マニュアルを参照してください。

詳細については

サブ VI の作成および使用方法、VI の保存方法、スタンドアロンアプリケーションおよび共有ライブラリの作成方法の詳細については、『LabVIEW ヘルプ』を参照してください。

プロジェクトの計画と設計

独自の VI を開発する前に、ユーザが実行する必要があるタスクのリストを作成します。ユーザインタフェースのコンポーネントと、データ解析や解析結果の表示に必要な制御器と表示器の数およびタイプを決めます。アプリケーションを使用するユーザやプロジェクトチームの他のメンバーとともに、ユーザが関数や機能にアクセスする必要が生じる状況とその方法を考察し検討します。アプリケーションを使用するユーザやプロジェクトチームのメンバに提示するためのサンプルフロントパネルを作成し、ユーザがそのフロントパネルを使用してタスクを実行できるかどうかを検討します。このような対話式のプロセスを経て、必要に応じユーザインタフェースを改良していきます。

アプリケーションを適切な位置で、管理可能なサイズの論理的断片に分割します。まず、アプリケーションのメインコンポーネントが含まれている上位ブロックダイアグラムから分割します。たとえば、ブロックダイアグラムには、構成のためのブロック、データ集録のためのブロック、集録データを解析するためのブロック、解析結果を表示するためのブロック、データをディスクに保存するためのブロック、エラーを処理するためのブロックなどが含まれます。

上位ブロックダイアグラムを設計したら、入出力を定義します。次に、上位ブロックダイアグラムのメインコンポーネントを構成するサブ VI を設計します。サブ VI を使用すると、上位ブロックダイアグラムの読み取り、デバッグ、理解、および管理が容易になります。また、一般的な操作または頻繁に行われる操作のために再利用可能なサブ VI を作成できます。
サブ VI は、作成しながらテストを行います。より上位のテストルーチンを作成できますが、小さいモジュールでエラーを検出する方が複数の VI の階層をテストするよりも簡単です。上位ブロックダイアグラムの初期設計に不完全な部分が見つかる可能性もあります。下位のタスクを実行するためにサブ VI を使用すると、アプリケーションの変更や再構成も簡単に行えます。サブ VI の詳細については、この章の「サブ VI」のセクションを参照してください。

ブロックダイアグラムとサブ VI のサンプルについては、**ヘルプ→サンプルの検索**を選択してください。

複数の開発者とともにプロジェクトを設計する

複数の開発者が同じプロジェクトに取り組む場合は、開発プロセスとアプリケーションが確実に機能するように、プログラミングの責任範囲、インタフェース、およびコーディング基準をあらかじめ定義しておきます。
コード基準の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」を参照してください。

1 台のコンピュータにプロジェクト VI のマスタコピーを保存し、ソースコードの管理方針を定めます。ファイルの共有を容易にするソースコード管理ツールが装備されている、LabVIEW プロフェッショナル開発システムの使用を検討してください。このツールには、VI を比較し、VI のバージョン間で行われた変更を表示するユーティリティも用意されています。
ソースコード管理の使用方法の詳細については、、『LabVIEW Development Guidelines』マニュアルの Chapter 2「Incorporating Quality into the Development Process」の「Source Code Control」のセクションを参照してください。

VI テンプレート

LabVIEW テンプレートには、一般の計測アプリケーションの作成開始に必要なサブ VI、関数、ストラクチャ、およびフロントパネルのオブジェクトが含まれています。VI テンプレートは、保存する必要がある名称未設定の VI として開きます。**ファイル→新規**を選択して、LabVIEW VI のテンプレートを含む**新規**ダイアログボックスを表示します。また、**LabVIEW** ダイアログボックスの**新規**プルダウンメニューから**新規**を選択して、**新規**ダイアログボックスを表示することもできます。

VI テンプレートをサブ VI として使用する場合、LabVIEW はその VI を閉じる前にテンプレートを VI として保存するようにプロンプトします。

VI テンプレートを作成する

カスタム VI テンプレートを作成して、似たような操作を実行するたびにフロントパネルまたはブロックダイアグラムに同じコンポーネントが配置されないようにします。VI を作成し、それをテンプレートとして保存することで、カスタム VI テンプレートを作成します。

その他のドキュメントタイプ

新規ダイアログボックスの**新規作成**リストで**その他のドキュメントタイプ**見出しの下にある項目を選択して、グローバル変数、カスタム制御器、ランタイムメニュー、多形性 VI、またはグローバル変数と制御器のテンプレートの作成を開始します。

組み込み VI および関数を使用する

LabVIEW には、データ集録 VI および関数、他の VI にアクセスする VI、他のアプリケーションと通信する VI など、特定のアプリケーションの作成に役立つ VI および関数が用意されています。これらの VI をアプリケーション内でサブ VI として使用すると、開発時間を短縮できます。サブ VI の詳細については、この章の「サブ VI」のセクションを参照してください。

計測器制御およびデータ集録 VI および関数を作成する

LabVIEW には、ユーザが作成する VI に組み込むことのできるサンプル VI が数多く含まれています。サンプル VI をユーザのアプリケーションに合わせて変更したり、1 つまたは複数のサンプルからユーザが作成する VI にコピーして貼り付けたりすることができます。

標準 VI および関数は、オスシロスコープなどの外部計測器を制御したり、熱電対からの読み取りなどのデータ集録に使用できます。

外部計測器を制御するには、計測器 I/O VI および関数を使用します。LabVIEW で計測器を制御するには、ハードウェアを正しくインストールし、電源をオンにして、コンピュータ上で動作させる必要があります。計測器の制御に使用する VI および関数は、ご使用のハードウェアがサポートする計測器の通信プロトコルによって異なります。計測器を制御する VI の作成方法の詳細については、『LabVIEW Measurements Manual』を参照してください。

DAQ デバイスからデータを集録するには、データ集録 VI および関数を使用します。これらの VI を使用するには、NI-DAQ ドライバソフトウェアと DAQ ハードウェアをインストールする必要があります。NI-DAQ ドライバおよび DAQ ハードウェアのインストール方法とデータを集録する VI の作成方法の詳細については、『LabVIEW Measurements Manual』を参照してください。データを集録したら、標準の解析 VI、レポート生成 VI、および数学 VI および関数を使用して、そのデータに対する解析、レポート生成、および数学演算を実行できます。LabVIEW で使用する解析や数学的な概念の詳細については、『LabVIEW Analysis Concepts』マニュアルを参照してください。

他の VI にアクセスする VI を作成する

サブ VI として呼び出された VI やユーザが実行する VI の動作を制御するには、アプリケーション制御 VI および関数を使用します。これらの VI および関数を使用すると、複数の VI を同時に構成できます。また、他の LabVIEW ユーザとネットワーク接続されている場合は、これらの VI と関数を使用して VI にリモートでアクセスして制御できます。VI をリモートで制御する方法の詳細については、第 17 章「VI をプログラム的に制御する」を参照してください。

他のアプリケーションと通信する VI を作成する

Microsoft Excel などの他のアプリケーションとの間でデータを読み書きするには、ファイル I/O VI および関数を使用します。これらの VI および関数を使用すると、レポートを生成したり、他のアプリケーションからのデータを VI に取り込むことができます。他のアプリケーション間でデータをやり取りする方法の詳細については、第 14 章「ファイル I/O」を参照してください。

FTP などの通信プロトコルを使用して LabVIEW データをウェブ上に転送したり、通信プロトコルを使用してクライアントサーバアプリケーションを作成するには、通信 VI および関数を使用します。ネットワークまたはウェブ上で他のアプリケーションと通信する方法の詳細については、第 18 章「LabVIEW のネットワーク動作」を参照してください。

(Windows) ActiveX オブジェクトを VI に追加したり、ActiveX 対応アプリケーションを制御するには、ActiveX 関数を使用します。ActiveX テクノロジの使用方法の詳細については、第 19 章「Windows の接続性」を参照してください。

サブ VI

VI を作成し、そのアイコンおよびコネクタペーンを作成したら、その VI を他の VI 内で使用できます。他の VI のブロックダイアグラムから呼び出された VI をサブ VI といいます。サブ VI は、テキストベースのプログラミング言語のサブルーチンに相当します。サブ VI ノードは、テキストベースのプログラミング言語におけるサブルーチン呼び出しに相当します。プログラム内のサブルーチンコールステートメントがサブルーチンそのものではないのと同様に、ノードもサブ VI そのものではありません。同じサブ VI ノードが複数含まれているブロックダイアグラムでは、同じサブ VI が数回呼び出されます。

サブ VI の制御器と表示器は、呼び出し側 VI のブロックダイアグラムとの間でデータをやり取りします。**関数**パレットで **VI を選択**アイコンをクリックし、VI をダブルクリックしてブロックダイアグラムに VI を配置すると、その VI へのサブ VI 呼び出しが作成されます。

メモ　VI をサブ VI として使用するには、その前にコネクタペーンを設定する必要があります。コネクタペーンの設定方法の詳細については、この章の「コネクタペーンを設定する」のセクションを参照してください。

サブ VI を編集するには、ブロックダイアグラム上のサブ VI を操作ツールまたは位置決めツールでダブルクリックします。サブ VI を保存すると、現在のインスタンスだけでなく、サブ VI に対するすべての呼び出しに変更の影響があります。

通常、LabVIEW がサブ VI を呼び出すと、そのフロントパネルを表示せずにサブ VI が実行されます。呼び出し時に、サブ VI の 1 つのインスタンスのフロントパネルを表示するには、サブ VI アイコンを右クリックし、ショートカットメニューから**サブ VI ノード設定**を選択します。呼び出し時にサブ VI のすべてのインスタンスのフロントパネルを表示するには、**ファイル→VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**ウィンドウの外観**を選択して、**カスタマイズする**ボタンをクリックします。

共通の操作を監視する

VI を作成すると、特定の操作を頻繁に行うことに気付くことがあります。そのような場合はサブ VI またはループを使用してその操作を繰り返し行うことを検討してください。たとえば、図 7-1 のブロックダイアグラムには 2 つの同じ操作が含まれています。

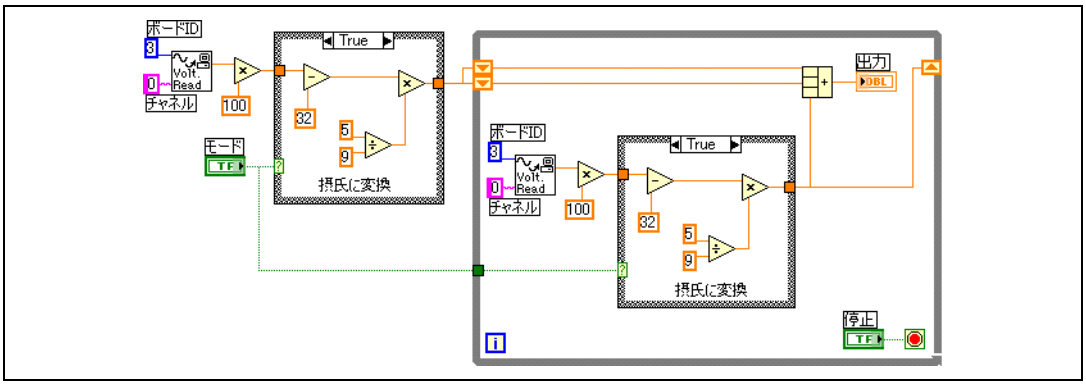


図 7-1　2 つの同じ操作を持つブロックダイアグラム

図 7-2 のブロックダイアグラムのように、その操作を実行するサブ VI を作成し、そのサブ VI を 2 回呼び出すことができます。

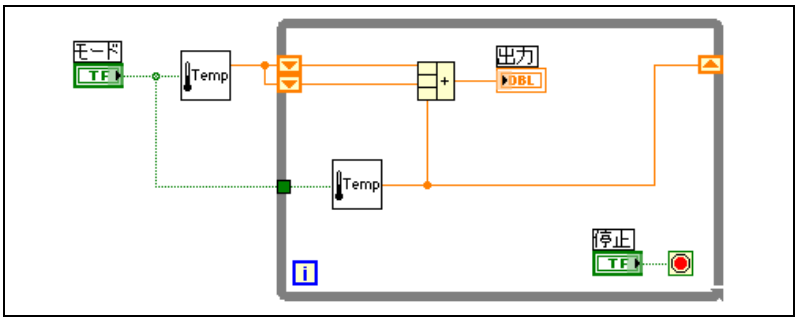


図 7-2　サブ VI を 2 回呼び出す

他の VI でこのサブ VI を再利用することもできます。ループを使用して共通の操作を結合する方法の詳細については、第 8 章「ループとストラクチャ」を参照してください。

コネクタペーンを設定する

VI をサブ VI として使用するには、左図に示すコネクタペーンを作成する必要があります。コネクタペーンは、VI の制御器および表示器に対応する端子のセットで、テキストベースのプログラミング言語での関数呼び出しのパラメータリストに似ています。コネクタペーンは、VI に配線できる入力および出力を定義するため、サブ VI として使用することができます。コネクタペーンの詳細については、第 2 章「バーチャルインスツルメンツ（仮想計測器）VI の概要」の「アイコンとコネクタペーン」のセクションを参照してください。

接続を定義するには、フロントパネル上の制御器または表示器をコネクタペーンの各端子に割り当てます。コネクタペーンを定義するには、フロントパネルウィンドウの右上隅にあるアイコンを右クリックし、ショートカットメニューから**コネクタを表示**を選択してコネクタペーンを表示します。これで、アイコンがコネクタペーンに置き換えられます。コネクタペーン上の各四角形は端子を表します。この四角形を使用して入出力を割り当てます。LabVIEW がコネクタペーン上に表示する端子の数は、フロントパネル上の制御器と表示器の数によって異なります。

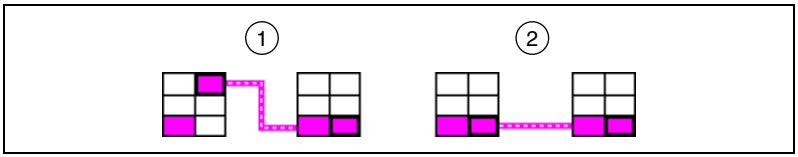
コネクタペーンには、最大 28 個の端子があります。プログラムで使用する制御器および表示器がフロントパネル上に 29 以上ある場合は、制御器および表示器のいくつかを 1 つのクラスタにグループ化して、このクラスタをコネクタペーン上の端子に割り当てます。クラスタを使用してデータをグループ化する方法の詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「クラスタ」のセクションを参照してください。

メモ　1 つの VI に 16 端子より多い数の端子を割り当てると、読みやすさや使いやすさが損なわれることがあります。

VI に対して別の端子パターンを選択するには、コネクタペーンを右クリックし、ショートカットメニューから**パターン**を選択します。追加端子を持つコネクタペーンパターンを選択します。追加端子は、必要になるまで未接続にしておくことができます。このように柔軟性を持たせることによって、VI の階層に対する影響を最小限に抑えた状態で変更を行うことができます。

頻繁に併用するサブ VI のグループを作成する場合は、各入力を配置する位置を思い出せるように、同じ位置に共通の入力端子を持つ統一されたコネクタペーンをサブ VI に指定します。他のサブ VI が入力として使用する出力を生成するサブ VI を作成する場合は、配線パターンをわかりやすくするために入出力接続を揃えます。フロントパネルの左下隅に**エラー入力**クラスタを配置し、右下隅に**エラー出力**クラスタを配置します。

図 7-3 は、不適切に揃えられたエラークラスタの例と適切に揃えられたエラークラスタの例を示しています。



1　不適切な例　　2　適切な例

図 7-3　不適切に揃えられたエラークラスタと適切に揃えられたエラークラスタ

コネクタペーンを設定するときに使用するスタイルの詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Connector Panes」のセクションを参照してください。

必須、推奨、および任意の入出力を設定する

ユーザがサブ VI 接続の配線を忘れないように、必須、推奨、および任意の入出力を指定できます。

コネクタペーンの端子を右クリックし、ショートカットメニューから**この接続は**を選択します。チェックマークは現在の端子の設定を示します。**必須**、**推奨**、または**任意**を選択します。

端子入力の場合、必須とは、その入力が配線されないと、サブ VI が配置されたブロックダイアグラムが壊れることを意味します。必須は、端子出力に使用することはできません。端子の入出力の場合、推奨または任意とは、その端子を配線しなくてもサブ VI を配置したブロックダイアグラムを実行できることを意味します。端子を配線しなくても、VI は警告を表示しません。

`vi.lib` 内の VI の入出力は、**必須**、**推奨**、または**任意**としてすでにチェックマークが付けられています。作成した VI の入出力はデフォルトで**推奨**に設定されます。VI が正しく動作するためにその入力または出力が必要な場合にのみ、端子の設定を必須に設定します。

ヘルプウィンドウに、必須接続は太字、推奨接続は通常のテキスト、任意接続は淡色表示で表示されます。**ヘルプ**ウィンドウで**オプションの端子と完全パスを隠す**ボタンをクリックすると、任意端子のラベルは表示されません。

アイコンを作成する

各 VI では、フロントパネルおよびブロックダイアグラムウィンドウの右上隅に左図のようなアイコンが表示されます。アイコンは VI を図で表現したものです。アイコンはテキスト、画像、またはその両方の組み合わせです。ある VI をサブ VI として使用する場合、アイコンはその VI のブロックダイアグラム上にあるサブ VI を識別します。

デフォルトのアイコンには、LabVIEW の起動後に開いた新規 VI の数を示す数が含まれています。カスタムアイコンを作成してデフォルトアイコンと置き換えるには、フロントパネルまたはブロックダイアグラムの右上隅にあるアイコンを右クリックして、ショートカットメニューから**アイコンを編集**を選択するか、あるいはフロントパネルの右上隅にあるアイコンをダブルクリックします。

ファイルシステムの任意の場所からグラフィックをドラッグして、フロントパネルまたはブロックダイアグラムの右上隅にドロップすることができます。LabVIEW は、グラフィックを 32ξ32 ピクセルのアイコンに変換します。

使用するモニタのタイプによって、モノクロ、16 色、および 256 色のアイコンを個別に設計できます。カラープリンタを使用していない場合は、モノクロのアイコンが印刷に使用されます。

アイコンの設計の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Icons」のセクションを参照してください。

アイコンまたは拡張可能なノードとしてサブ VI および Express VI を表示する

アイコンまたは拡張可能なノードとして VI および Express VI を表示できます。拡張可能なノードはカラーのフィールドで囲まれたアイコンとして表示されます。サブ VI は黄色のフィールド付きで表示され、Express VI は青色のフィールド付きで表示されます。ブロックダイアグラム上でスペースを節約する場合は、アイコンを使用してください。また、配線をより簡単にしたりブロックダイアグラムを記録しやすくするには、拡張可能なノードを使用してください。デフォルトでは、サブ VI はブロックダイアグラムにアイコンとして表示され、Express VI は拡張可能なノードとして表示されます。サブ VI や Express VI を拡張可能なノードとして表示するには、サブ VI または Express VI を右クリックし、ショートカットメニューから**アイコンとして表示**を選択してチェックマークを外します。

メモ　拡張可能なノードとしてサブ VI または Express VI を表示すると、そのノードの端子を表示したり、そのノードに関するデータベースへのアクセスを有効にすることはできません。

拡張可能なサブ VI または Express VI のサイズを変更すると、サブ VI または Express VI の入力および出力端子がアイコンの下に表示されます。任意の端子は灰色の背景付きで表示されます。表示しない推奨（または必須）の入力（または出力）端子は、サブ VI（または Express VI）アイコンを囲むカラーのフィールドで入力または出力の矢印として表示されます。サブ VI または Express VI が拡張される際に任意の端子へ配線すると、サブ VI または Express VI のサイズが変更されて任意の端子は拡張フィールドに表示されなくなり、任意の端子はカラーのフィールドに入力または出力の矢印として表示されます。ただし、任意の端子を未配線にすると、入力または出力の矢印は表示されません。

デフォルトでは、サブ VI または Express VI を拡張すると、入力は出力の上に表示されます。拡張可能なフィールドで端子を右クリックし、ショートカットメニューから**入出力を選択**を選択して、表示する入力または出力を指定します。ショートカットメニューでは、ラインが入力と出力を区別します。拡張可能なサブ VI および Express VI のラベルは、アイコンを囲むカラーのフィールドに表示されます。拡張可能なノードのサイズを変更して、各端子の名前が拡大可能フィールドで 1 行に収まるようにするには、サブ VI または Express VI を右クリックしてショートカットメニューから**テキストにサイズを合わせる**を選択します。

以下の拡張可能なサブ VI は 10 個の入力および出力端子のうち 4 つを表示します。

Express VI の詳細については、『LabVIEW 入門』を参照してください。

VI の一部からサブ VI を作成する

VI の一部をサブ VI に変換するには、位置決めツールを使用して、再利用するブロックダイアグラムの部分を選択し、**編集→選択範囲をサブ VI に変換**を選択します。ブロックダイアグラムの選択された部分がこの新しいサブ VI のアイコンに置き換えられます。LabVIEW は新しいサブ VI の制御器と表示器を作成し、サブ VI を既存のワイヤに配線します。

選択範囲からサブ VI を作成する機能は便利ですが、VI の論理階層を作成する場合には慎重に計画する必要があります。選択範囲にどのオブジェクトを含めるのかを検討し、その結果作成される VI の機能が変わらないようにしてください。

サブ VI を設計する

ユーザがサブ VI のフロントパネルを表示する必要がない場合は、色、フォントなど外観の作成に時間をかける必要はありません。ただし、VI をデバッグするときにフロントパネルを表示しなければならない場合があるので、フロントパネルの構成は重要です。

コネクタペーンに表示されるとおりに制御器と表示器を配置します。フロントパネルの左側に制御器を配置し、右側に表示器を配置します。フロントパネルの左下に**エラー入力**クラスタを配置し、右下に**エラー出力**クラスタを配置します。コネクタペーンの設定方法の詳細については、本章の「コネクタペーンを設定する」のセクションを参照してください。

制御器がデフォルト値を持つ場合は、そのデフォルト値を括弧で囲んで制御器名の一部とします。該当する場合は、制御器の名前に測定単位も指定します。たとえば、制御器が上限温度を設定する場合にデフォルト値が 75 °F のときは、制御器に**上限温度 (75 degF)** という名前を付けます。複数のプラットフォーム上でそのサブ VI を使用する場合は、制御器の名前に特殊文字を使用しないでください。たとえば、**°F** の代わりにを **degF** 使用してください。

ユーザが TRUE 状態での制御器の動作を判断できるように、ブール制御器に名前を付けます。**キャンセル**、**リセット**、**初期化**のように、実行される動作を示す名前を使用します。

VI の階層を表示する

階層ウィンドウには、タイプ定義やグローバル変数など、メモリ内にあるすべての VI の呼び出し側階層のグラフィック表現が表示されます。**階層**ウィンドウを表示するには**参照→ VI 階層を表示**を選択します。このウィンドウを使用して、現在の VI を構成するサブ VI や他のノードを表示します。

階層ウィンドウのオブジェクト上に操作ツールを移動すると、各 VI の名前が表示されます。VI を他の VI 内でサブ VI として使用するには、位置決めツールを使用して**階層**ウィンドウからブロックダイアグラムにその VI をドラッグします。また、1 つまたは複数のノードを選択してクリップボードにコピーし、他のブロックダイアグラム上に貼り付けることもできます。その V のフロントパネルを表示するには、**階層**ウィンドウ内の VI をダブルクリックします。

サブ VI が含まれている VI には、そのボタン枠に矢印ボタンが表示されます。サブ VI を表示または非表示にするには、この矢印ボタンをクリックします。赤い矢印ボタンは、すべてのサブ VI が非表示になっていることを示します。黒い矢印ボタンは、すべてのサブ VI が表示されていることを示します。

VI を保存する

VI を個別のファイルとして保存したり、複数の VI をグループ化して VI ライブラリ内に保存できます。VI ライブラリファイルの拡張子は `.llb` です。ナショナルインスツルメンツでは、VI を個別ファイルとしてディレクトリに整理して保存することをお勧めします。特に複数の開発者が同じプロジェクトで作業する場合は、個別ファイルでの保存をお勧めします。ディレクトリ内での VI の分類方法の詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Organizing VIs in Directories」のセクションを参照してください。

VI を個別ファイルとして保存する場合の利点

VI を個別ファイルとして保存する利点を以下に示します。

- ファイルシステムを使用して個別ファイルを管理できる。
- サブディレクトリを使用できる。
- 同じファイルにプロジェクト全体を保存するよりも、個別ファイルに VI および制御器を保存する方が信頼性が高い。
- プロフェッショナル開発システムの標準ソースコード管理ツールやサードパーティのソースコード管理ツールを使用できる。

VI をライブラリとして保存する場合の利点

VI をライブラリとして保存する利点を以下に示します。

- ファイル名に最大 255 文字を使用できる。

  (Mac OS) Mac OS 9*x* 以前のバージョンでは、ファイル名が 31 文字に制限されていますが、ライブラリ内の VI ファイル名には字数制限はありません。
- 他のプラットフォームへ VI ライブラリを転送する方が、個々の VI を何度か転送するよりも簡単である。また、ユーザがすべての必要なファイルを受け取ったかどうかを確認できる。
- VI ライブラリはファイルを圧縮するため、プロジェクトのサイズを若干小さくできる。
- ライブラリで最上位 VI としてマークを付け、ライブラリを開いたときに LabVIEW が自動的にそのライブラリのすべての最上位 VI を開くようにすることができる。

メモ　LabVIEW では、すべてのプラットフォーム上で記憶位置を統一するため、標準 VI とサンプル VI の多くが VI ライブラリ内に保存されています。

VI ライブラリを使用する場合は、アプリケーションを複数の VI ライブラリに分割することを検討してください。つまり、ある VI ライブラリに上位 VI を配置し、それ以外のライブラリは機能別に VI を含むように設定します。ライブラリの VI に変更を保存する場合、オペレーティングシステムは大きなファイルに変更を書き込まなければならないため、個別の VI に変更を保存する場合より時間がかかります。大きなライブラリに変更を保存すると、メモリ容量が増え、性能が低下する可能性があります。各ライブラリのサイズは約 1 MB までとしてください。

ライブラリ内で VI を管理する

VI ライブラリ内でのファイルのコピー、名前の変更、および削除を簡単に行うには、VI ライブラリマネージャを使用します。このツールにアクセスするには、**ツール→ VI ライブラリマネージャ**を選択します。このツールを使用すると、新しい VI ライブラリとディレクトリを作成し、そのディレクトリ間で VI ライブラリを変換することもできます。ソースコード管理ツールを使用して VI を管理する必要がある場合は、新しい VI ライブラリおよびディレクトリを作成してそのディレクトリ間で VI ライブラリを変換することが重要になります。

VI ライブラリマネージャを使用する前に、すでにメモリ内にある VI に対してファイル操作が行われないように、影響を受ける可能性があるすべての VI を閉じます。

(Windows 2000/XP/Me/98) Windows エクスプローラで `.llb` ファイルをダブルクリックしてライブラリの内容を表示し、ライブラリ内でファイルを開いたり、移動したり、名前を変更したり、削除したりすることができます。

VI に名前を付ける

VI を保存するときは、わかりやすい名前を使用してください。`Temperature Monitor.vi` や `Serial Write & Read.vi` のようなわかりやすい名前を使用すると、VI を簡単に識別にでき、その使用方法が一目でわかります。`VI#1.vi` のようなあいまいな名前を使用すると、特に複数の VI を保存した場合に VI の識別が困難になる可能性があります。

ユーザが他のプラットフォームでその VI を実行する可能性があるかどうかを考慮してください。一部のオペレーティングシステムで特別な目的のために予約されている文字は使用しないでください。予約されている文字の例は、`\`、`:`、`/`、`?`、`*`、`<`、`>`、`#` などです。

Mac OS 9.*x* 以前のバージョンで VI を実行する場合は、VI の名前を 30 文字以下にしてください。

旧バージョンで保存する

VI は、旧バージョンの形式で保存できます。これにより、LabVIEW のアップグレードが便利になり、必要に応じて VI を 2 種類のバージョンの LabVIEW 用に維持しておくことができます。LabVIEW を新しいバージョンにアップグレードしても、旧バージョンの VI に戻すことができます。

旧バージョンの形式で VI を保存する際、LabVIEW はその VI だけでなく、`vi.lib` ファイルを除くその階層内のすべての VI を変換します。

VI には旧バージョンの LabVIEW では提供されていなかった機能が使用されていることがあります。この場合、LabVIEW はできるだけ多くの VI を保存し、変換できなかった VI についてはレポートを生成します。このレポートは生成されるとすぐに**警告**ダイアログボックスに表示されます。これらの警告を確認してダイアログボックスを閉じるには、**OK** ボタンをクリックします。**保存**ボタンをクリックして警告をテキストファイルに保存しておくと、後で確認することができます。

VI を配布する

他のコンピュータや他のユーザに VI を配布する場合は、ユーザが編集可能なブロックダイアグラムのソースコードを含めるか、あるいはブロックダイアグラムを非表示または削除するか、どちらにするかを検討します。

他の LabVIEW ユーザへブロックダイアグラムのソースコードを含む VI を配布する場合は、ブロックダイアグラムにパスワード保護を割り当てることができます。ブロックダイアグラムは使用可能ですが、ユーザがブロックダイアグラムを表示したり編集する場合にパスワードを入力する必要があります。

他のプログラミング言語の開発者へ VI を配布する場合は、スタンドアロンアプリケーションまたは共有ライブラリを作成する方法があります。ユーザが VI を編集する可能性がない場合は、スタンドアロンアプリケーションまたは共有ライブラリの使用が適しています。ユーザはアプリケーションを実行したり共有ライブラリを使用することができますが、ブロックダイアグラムの編集や表示はできません。

メモ　スタンドアロンアプリケーションまたは共有ライブラリは、LabVIEW プロフェッショナル開発システムまたはアプリケーションビルダでのみ作成できます。

VI の配布の際には、他のユーザが VI を編集できないようにブロックダイアグラムのソースコードを削除するというオプションもあります。ファイルサイズを縮小し、ユーザがソースコードを変更できないようにするには、**ファイル→オプション付き保存**を選択して、ブロックダイアグラムな

しで VI を保存します。ブロックダイアグラムなしで VI を保存すると、ユーザは VI を他のプラットフォームに移動したり、LabVIEW の今後のバージョンに VI をアップグレードすることができなくなります。

注意　ブロックダイアグラムなしで VI を保存する場合は、元のバージョンの VI に上書きしないでください。別のディレクトリに保存するか、別名で保存するようにします。

プラットフォーム間での VI のポート、および VI のローカライズの詳細については『LabVIEW VI の移植とローカライズ』アプリケーションノートを参照してください。

スタンドアロンアプリケーションと共有ライブラリを作成する

アプリケーションビルダを使用してスタンドアロンアプリケーションやインストーラまたは VI の共有ライブラリ（DLL）を作成するには、**ツール→アプリケーションまたは共有ライブラリ（DLL）を作成**を選択します。LabWindows™/CVI™、Microsoft Visual C++、Microsoft Visual Basic などのテキストベースのプログラミング言語を使用して共有ライブラリ内の VI を呼び出す場合は、共有ライブラリを使用します。

VI で作成した機能をその他の開発者と共有したい場合に共有ライブラリを使用すると便利です。共有ライブラリを使用することにより、LabVIEW 以外のプログラミング言語でも、LabVIEW で開発したコードにアクセスすることが可能になります。

他の開発者は、スタンドアロンのアプリケーションを実行したり共有ライブラリを使用することはできますが、ブロックダイアグラムの編集や表示はできません。

メモ　LabVIEW プロフェッショナル開発システムにはアプリケーションビルダが含まれています。LabVIEW 基本パッケージまたは LabVIEW 開発システムをご使用の場合は、アプリケーションビルダを別途購入できます。ナショナルインスツルメンツ社のウェブサイト `ni.com/info` にアクセスし、Language の設定を Japanese にして、info コード `rdlv21` を入力してください。作成するアプリケーションまたは共有ライブラリのさまざまな設定を構成するには、**アプリケーションまたは共有ライブラリ（DLL）を作成**ダイアログボックスにあるタブを使用します。これらの設定を行ったら、必要に応じてアプリケーションを簡単に再構築できるように、それらの設定をスクリプトとして保存します。

アプリケーションビルダのインストール方法の詳細については、『LabVIEW アプリケーションビルダリリースノート』を参照してください。

第 II 部

VI の作成と編集

第 II 部では、アプリケーションが特定の方法で動作するようにするための LabVIEW の機能、VI、および関数について説明します。各章では、LabVIEW の各機能の有効性や、VI および関数の各クラスの概要について説明します。

第 II 部「VI の作成と編集」は以下の章で構成されています。

- 第 8 章「ループとストラクチャ」では、ブロックダイアグラムでループおよびストラクチャを使用して、コードのブロックを繰り返したり、条件付きでコードを実行したり、特定の順序でコードを実行する方法を説明します。
- 第 9 章「イベント駆動型プログラミング」では、動的および静的にイベントを登録する方法、独自のイベントの作成方法、および独自のイベントの名前の付け方を説明します。
- 第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」では、文字列、配列、およびクラスタを使用して、データをグループ化する方法を説明します。
- 第 11 章「ローカル変数とグローバル変数」では、ローカル変数とグローバル変数を使用して、ワイヤで配線できないアプリケーション位置の間で情報を受け渡しする方法を説明します。
- 第 12 章「グラフおよびチャート」では、グラフおよびチャートを使用して、グラフィック形式でデータプロットを表示する方法を説明します。
- 第 13 章「グラフィック & サウンド VI」では、VI でグラフィックやサウンドを表示する方法を説明します。
- 第 14 章「ファイル I/O」では、ファイル I/O 操作の実行方法を説明します。
- 第 15 章「VI の文書化と印刷」では、VI を文書化したり印刷する方法を説明します。

- 第 16 章「VI をカスタマイズする」では、アプリケーションのニーズに従って、VI およびサブ VI を構成する方法を説明します。
- 第 17 章「VI をプログラム的に制御する」では、VI が LabVIEW の他のインスタンスと通信する方法を説明します。これにより、VI および LabVIEW をプログラム的に制御できます。
- 第 18 章「LabVIEW のネットワーク動作」では、VI を使用して他のアプリケーションやリモートコンピュータで実行中のものも含めた他のプロセスとの通信またはネットワーク方法を説明します。
- 第 19 章「Windows の接続性」では、他の Windows アプリケーションでアクセス可能なパブリックのオブジェクト、コマンド、および関数のセットの提供方法を説明します。
- 第 20 章「テキストベースのプログラミング言語からのコード呼び出し」では、テキストベースのプログラミング言語からのコードの呼び出し方法および DLL の使用方法について説明します。
- 第 21 章「フォーミュラと方程式」では、VI での方程式の使用方法を説明します。

8

ループとストラクチャ

ストラクチャは、テキストベースのプログラミング言語のループおよびケースステートメントのグラフィカル表現です。コードのブロックを繰り返したり、条件付きでコードを実行したり、特定の順序でコードを実行するには、ブロックダイアグラムでストラクチャを使用します。

他のノードと同様に、ストラクチャには端子があります。これらの端子はストラクチャをブロックダイアグラムの他のノードに接続し、入力データを受け取ると自動的に実行して、実行が終了するとそのデータを出力ワイヤに渡します。

各ストラクチャにはサイズ変更可能な特有の枠があり、ストラクチャの規則に従って実行されるブロックダイアグラムの一部を囲んでいます。ストラクチャの枠で囲まれたブロックダイアグラムの部分をサブダイアグラムと呼びます。ストラクチャにデータを受け渡す端子をトンネルと呼びます。トンネルはストラクチャの枠上にある接続ポイントです。

詳細については

ストラクチャの使用方法の詳細については、『LabVIEW ヘルプ』を参照してください。

ブロックダイアグラムでのプロセスの実行方法を制御するには、**ストラクチャ**パレットにある以下のストラクチャを使用します。

- **For ループ**：設定された回数だけサブダイアグラムを実行します。
- **While ループ**：条件が一致するまでサブダイアグラムを実行します。
- **Case ストラクチャ**：複数のサブダイアグラムが含まれており、ストラクチャに渡された入力値に従って、そのうちの 1 つだけを実行します。
- **シーケンスストラクチャ**：1 つまたは複数のサブダイアグラムが含まれており、順番に実行されます。
- **フォーミュラノード**：入力された数値に基づいて数学演算を実行します。「フォーミュラノード」の使用方法の詳細については、第 21 章「フォーミュラと方程式」の「フォーミュラノード（Formula Nodes）」のセクションを参照してください。
- **イベントストラクチャ**：ユーザの VI の操作方法に応じて実行する 1 つまたは複数のサブダイアグラムが含まれています。

ショートカットメニューを表示するには、ストラクチャの枠を右クリックします。

For ループおよび While ループのストラクチャ

繰り返し操作を制御するには、For ループと While ループを使用します。

For ループ

左図に示す For ループは、設定された回数だけサブダイアグラムを実行します。

左図に示すカウント端子（入力端子）の値は、サブダイアグラムの実行を繰り返す回数を示します。回数を明示的に設定するには、ループの外側からカウント端子の左側または上側に値を配線するか、自動指標付け機能を使用して自動的に回数を設定します。自動的に回数を設定する方法の詳細については、この章の「自動指標付けで For ループの回数を設定する」のセクションを参照してください。

左図に示す繰り返し端子（出力端子）には、実行された繰り返しの回数が表示されます。繰り返しのカウントは常にゼロ（0）から始まります。最初の繰り返しでは、繰り返し端子は **0** を返します。

カウント端子と繰り返し端子はともに 32 ビット符号付き倍長整数です。浮動小数点数をカウント端子に配線すると、LabVIEW は、浮動小数点数を範囲内の数値に強制的に丸めます。カウント端子に 0 または負の数を配線すると、ループは実行されず、出力はデータタイプのデフォルトデータを含みます。

現在の繰り返しから次の繰り返しにデータを渡すには、For ループにシフトレジスタを追加します。シフトレジスタをループに追加する方法の詳細については、この章の「ループ内のシフトレジスタおよび「フィードバックノード（Feedback Node）」」のセクションを参照してください。

While ループ

左図に示す While ループは、テキストベースのプログラミング言語の Do ループや Repeat-Until ループと同様、条件が一致するまでサブダイアグラムを実行します。

While ループは、入力端子である条件端子が特定のブール値を受け取るまでサブダイアグラムを実行します。条件端子のデフォルトの動作および外観は、**True の場合、停止**（左図）です。条件端子が **True の場合、停止**の場合、While ループは条件端子が TRUE 値を受け取るまでそのサブダイアグラムを実行します。条件端子の動作と外観を変更するには、端子また

は While ループの枠を右クリックしてショートカットメニューから **True の場合、継続**を選択します。条件端子が **True の場合、継続**の場合、While ループは条件端子が FALSE 値を受け取るまでそのサブダイアグラムを実行します。また、操作ツールを使用して条件端子をクリックし、条件を変更することもできます。

また、While ループの条件端子を使用すると、基本的なエラー処理を実行できます。エラークラスタを条件端子に配線すると、エラークラスタの**ステータス**パラメータの TRUE または FALSE の値だけが端子に渡されます。ショートカットメニュー項目の **True の場合、停止**および **True の場合、継続**は、**エラーで停止**および**エラーの場合、継続**に変更されます。エラークラスタおよびエラー処理の詳細については、第 6 章「VI の実行とデバッグ」の「エラーチェックとエラー処理」のセクションを参照してください。

左図に示す繰り返し端子（出力端子）には、実行された繰り返しの回数が表示されます。繰り返しのカウントは常にゼロ（0）から始まります。最初の繰り返しでは、繰り返し端子は **0** を返します。

現在の繰り返しから次の繰り返しにデータを渡すには、While ループにシフトレジスタを追加します。シフトレジスタをループに追加する方法の詳細については、この章の「ループ内のシフトレジスタおよび「フィードバックノード（Feedback Node）」」のセクションを参照してください。

無限 While ループを回避する

図 8-1 のように、ブール制御器の端子が While ループの外側に配置され、ループ開始時の条件端子が **True の場合、停止**の場合に、制御器を FALSE に設定されていると、無限ループが発生します。ループの外側の制御器を TRUE に設定し、条件端子を **True の場合、継続**に設定した場合にも、無限ループが発生します。

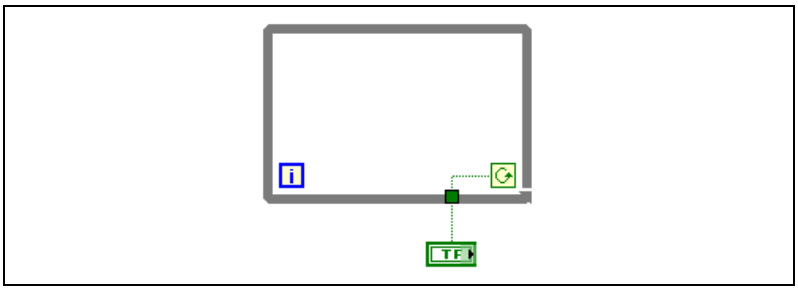


図 8-1　無限 While ループ

制御器の値はループの開始前に一度読み取られるだけなので、その値を変更しても無限ループは停止しません。無限ループを停止するには、ツールバー上の**実行を中断**ボタンをクリックして VI を中断する必要があります。

ループに自動指標付けを行う

For ループまたは While ループの入力トンネルに配列を配線する場合は、自動指標付けを有効にすると、その配列内のすべての要素を読み取って処理できます。配列の詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」を参照してください。

ループの枠上の入力トンネルに配列を配線し、その入力トンネルで自動指標付けを有効にすると、その配列の要素は最初の要素から 1 つずつループに入ります。自動指標付けが無効になっていると、配列全体がループに渡されます。配列の出力トンネルを自動指標付けすると、出力配列はループのすべての繰り返しから新しい要素を受け取ります。したがって、自動指標付けされた出力配列のサイズは常に繰り返し回数と等しくなります。

ループの枠上のトンネルを右クリックし、ショートカットメニューから**指標付け使用**または**指標付け不使用**を選択して、自動指標付けを有効または無効にします。While ループの自動指標付けはデフォルトでは無効になっています。

ループは 1 次元配列のスカラ要素、2 次元配列の 1 次元要素、というように指標を付けます。出力トンネルではその逆の操作が行われます。スカラ要素は 1 次元配列内に、1 次元配列は 2 次元配列内に、というように順番に配置されます。

自動指標付けで For ループの回数を設定する

For ループに入力された配列で自動指標付けを有効にすると、LabVIEW がカウント端子を配列のサイズに設定するため、カウント端子を配線する必要はありません。For ループを使用すると配列を一度に 1 要素ずつ処理できるので、For ループに配線するすべての配列に関して LabVIEW はデフォルトで自動指標付けを有効にします。配列を一度に 1 要素ずつ処理する必要がない場合は、自動指標付けを無効にします。

複数のトンネルに対して自動指標付けを有効にする場合や、カウント端子を配線する場合、回数は選択した小さい方の値になります。たとえば、それぞれ 10 個と 20 個の要素を持つ自動指標付けされた 2 つの配列がループに入り、値 15 をカウント端子に配線している場合、ループは 10 回実行し、2 番目の配列の最初の 10 個の要素だけを指標付けします。1 つのグラフ上で 2 つのデータソースからデータをプロットし、最初の 100 個の要素をプロットする場合は、カウント端子に 100 を配線します。ただし、要素が 50 個しか含まれていないデータソースがある場合は、

ループは 50 回実行し、最初の 50 個の要素にのみ指標付けします。配列のサイズを調べるには、「配列サイズ（Array Size）」関数を使用します。

配列の出力トンネルを自動指標付けすると、出力配列はループのすべての繰り返しから新しい要素を受け取ります。したがって、自動指標付けされた出力配列のサイズは常に繰り返し回数と等しくなります。たとえば、ループが 10 回実行される場合、出力される配列には 10 個の要素が入っています。出力トンネルで自動指標付けを無効にすると、ループの最後の繰り返しの要素だけがブロックダイアグラム内の次のノードに渡されます。自動指標付けが有効になっていることを示すために、括弧付きのグリフがループの枠に表示されます。出力トンネルと次のノード間のワイヤの太さも、ループが自動指標付けを使用しているかどうかを示します。自動指標付けを使用すると、ワイヤにはスカラではなく配列が入るためワイヤが太くなります。

While ループで自動指標付けを行う

While ループに入る配列に対して自動指標付けを有効にすると、While ループは For ループの場合と同様に配列に指標を付けます。ただし、While ループは特定の条件が一致するまで繰り返されるため、While ループが実行する繰り返し回数が配列のサイズによって制限されることはありません。While ループが入力配列の最後を過ぎても指標を付けるときは、配列要素タイプのデフォルト値がループに渡されます。「配列サイズ（Array Size）」関数を使用すると、デフォルト値が While ループに渡されないようにすることができます。配列サイズ関数は配列内にある要素の数を示します。While ループが配列サイズと同じ回数だけ繰り返した時点で実行を停止するように設定します。

注意　出力配列のサイズはあらかじめ決められないため、While ループで使用するよりも、For ループの出力に対して自動指標付けを有効にする方が効率的です。あまり多くの繰り返しを行うと、システムのメモリが不足する可能性があります。

ループを使用して配列を作成する

ループを使用して配列内の要素を処理する他に、For ループおよび While ループを使用して配列を作成することもできます。ループ内にある VI または関数の出力をループ枠に配線します。While ループを使用する場合は、作成されたトンネルを右クリックしてショートカットメニューから**指標付け使用**を選択します。For ループの場合、指標付けはデフォルトで有効になっています。トンネルの出力は、VI または関数が各ループ繰返し後に返す各値の配列です。

配列作成のサンプルについては、`examples¥general¥arrays.llb` を参照してください。

ループ内のシフトレジスタおよび「フィードバックノード（Feedback Node）」

ループのある繰り返しから次の繰り返しに値を転送するには、For ループおよび While ループでシフトレジスタまたは「フィードバックノード（Feedback Node）」を使用します。

シフトレジスタ

前の繰り返しからループを介して値を渡す場合には、シフトレジスタを使用します。左図に示すシフトレジスタは、ループ枠の左右に一対の端子があり、互いに向かい合っています。右の端子には上矢印が付いており、繰り返し処理完了時のデータが格納されます。LabVIEW は、レジスタの右側に接続されたデータを次の繰り返しに転送します。シフトレジスタを作成するには、ループの右または左の枠を右クリックし、ショートカットメニューから**シフトレジスタを追加**を選択します。

シフトレジスタはどのタイプのデータでも転送でき、シフトレジスタに配線された最初のオブジェクトのデータタイプに自動的に適応します。各シフトレジスタの端子に配線するデータは、同じタイプである必要があります。

シフトレジスタを初期化するには、制御器または定数をループの左側にあるシフトレジスタ端子に配線します。シフトレジスタを初期化すると、VI の実行時に初めてループが実行されるときにシフトレジスタが渡す値がリセットされます。レジスタを初期化しないと、ループは、前回実行されたときにレジスタに書き込まれた値を使用するか、まだ一度も実行されていない場合はそのデータタイプのデフォルト値を使用します。

VI を繰り返して実行するには、初期化されていないシフトレジスタを持つループを使用します。こうすると、VI を実行するたびに、シフトレジスタの初期出力は前回の実行の最後で得た値になります。以降の VI の実行間で状態情報を保持するには、初期化されていないシフトレジスタを使用してください。ループの実行後、シフトレジスタに格納されている最後の値は右の端子の出力です。

ループには、複数のシフトレジスタを追加できます。ループ内で複数の操作が前の繰り返し値を使用している場合、複数のシフトレジスタを使用し、ストラクチャにそれら複数処理のデータ値を格納します。

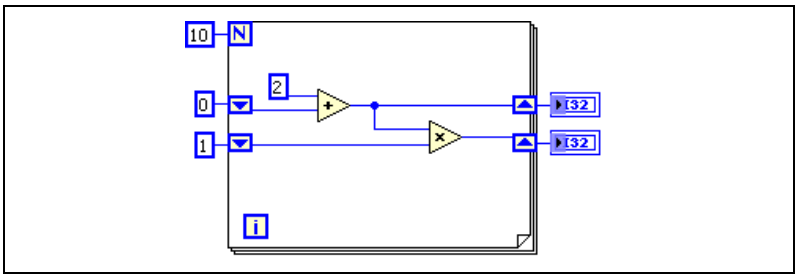


図 8-2　複数のシフトレジスタ

スタックシフトレジスタ

スタックシフトレジスタを使用すると、ループの以前の繰り返しのデータにアクセスできます。スタックシフトレジスタを作成するには、左の端子を右クリックしてショートカットメニューから**要素を追加**を選択します。スタックシフトレジスタは、前の複数の繰り返し値を記憶し、それらの値を次の繰り返りに渡します。

スタックシフトレジスタは、ループの左側のみでしか作成できません。右側の端子は、現在の繰り返しで生成されたデータを次の繰り返しに渡すだけです。

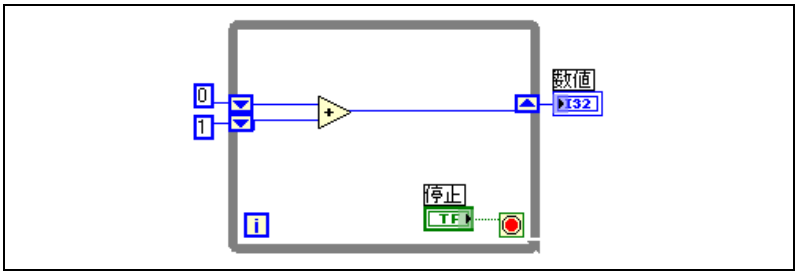


図 8-3　スタックシフトレジスタ

他の要素を左側の端子に追加した場合、最後の 2 回の繰り返しで生成された値は、次の繰り返りに渡されます。最新の繰り返し値は一番上のシフトレジスタに格納されます。下の端子は、前の繰り返しから渡されるデータを格納します。

シフトレジスタをトンネルと入れ替える

ループの繰り返しから次の繰り返しに値を転送する必要がなくなった場合には、シフトレジスタをトンネルに入れ替えます。シフトレジスタを入れ替えるには、シフトレジスタを右クリックしてショートカットメニューから**トンネルと置換**を選択します。

For ループの出力シフトレジスタ端子をトンネルに入れ替えると、For ループはデフォルトで指標付けが有効になっているため、ループ外のノードに配線されたワイヤは壊れます。トンネルを右クリックしてショートカットメニューから**ソースで指標付け不使用**を選択し、指標付けと壊れたワイヤの自動修正を無効にします。指標付けを有効にする場合は、壊れたワイヤと表示器端子を削除して、トンネルを右クリックして**表示器を作成**を選択します。

ループ内の指標付けの詳細については、この章の「ループに自動指標付けを行う」のセクションを参照してください。

トンネルをシフトレジスタに入れ替える

ループの繰り返しから次の繰り返しに値を転送する場合は、トンネルをシフトレジスタに入れ替えます。トンネルをシフトレジスタに入れ替えるには、トンネルを右クリックしてショートカットメニューから**シフトレジスタと置換**を選択します。右クリックしたトンネルの反対側のループの境界線にトンネルがない場合には、LabVIEW は自動的にシフトレジスタ端子のペアを作成します。右クリックしたトンネルの反対側のループの境界線にトンネルが存在する場合には、LabVIEW はクリックしたトンネルをシフトレジスタ端子に入れ替え、カーソルがシフトレジスタのアイコンに変わります。ループの反対側のトンネルをクリックしてトンネルをシフトレジスタに入れ替えるか、またはブロックダイアグラムをクリックしてシフトレジスタの反対側のループの境界線上にシフトレジスタを配置します。シフトレジスタがトンネルの後ろに表示された場合には、シフトレジスタは配線されていません。

指標付けが有効になった状態で While ループのトンネルをシフトレジスタに変換すると、シフトレジスタが自動指標付けを行うことができないため、ループ外のノードに配線されたすべての配線が壊れます。壊れたワイヤを削除し、出力ワイヤを右側のシフトレジスタからもう一方のトンネルに配線し、トンネルを右クリックしてショートカットメニューから**指標付け使用**を選択し、トンネルをノードに配線します。

ループ内の指標付けの詳細については、この章の「ループに自動指標付けを行う」のセクションを参照してください。

フィードバックノード（Feedback Node）

左図に示す「フィードバックノード」は、、サブ VI、関数、またはサブ VI や関数のグループの出力を同じ VI、関数、またはグループの入力に配線した場合に、For ループまたは While ループでのみ、自動的に表示されます。シフトレジスタと同様に、「フィードバックノード」は、ループが繰り返しを完了するとデータを保存し、その値をループの次の繰り返しに渡して、データタイプを転送します。「フィードバックノード」を使用すると、ループで不必要に長いワイヤを使用する必要がなくなります。「フィードバックノード」の矢印はワイヤのデータフローの方向を示します。

また、「フィードバックノード」を選択して、For ループまたは While ループの内側のみに配置することができます。データとトンネルを接続するワイヤを分岐する前に「フィードバックノード」をワイヤ上に配置すると、「フィードバックノード」は各値をトンネルに渡します。データとトンネルを接続するワイヤを分岐した後に「フィードバックノード」をワイヤ上に配置すると、「フィードバックノード」は VI または関数の入力に各値を渡し、最後の値をトンネルに渡します。

たとえば、図 8-4 の For ループは 10 回繰り返されます。「フィードバックノード」は、値を「和（Add)」関数に渡す前にループの前回の繰り返しの値をトンネルに渡します。トンネルの値は、常に前回の繰り返しからの値です。ループの最後の繰り返しで「フィードバックノード」は最後の値（この場合 45）を保持しますが、トンネルまたは数値表示器に値を渡しません。VI の実行終了時の数値表示器の値は 36 です。これは最後の繰り返し値ではなく、その前のループの繰り返し値です。

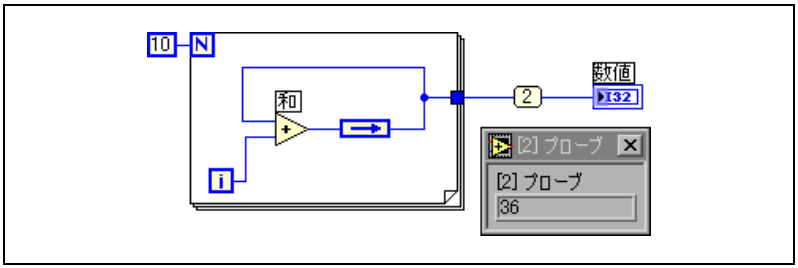


図 8-4　For ループで次の値を最後の値に渡す

図 8-5 の For ループは 10 回繰り返されます。ただし、「フィードバックノード」はループが繰り返されるたびに、以前の繰り返し値を「和」関数の入力にのみ渡します。最後のループの繰り返しで、「フィードバックノード」は以前の繰り返し値（36）を「和（Add)」関数の入力にのみ渡します。「和」関数は、繰り返し端子が生成する値（9）と、「フィードバックノード」が渡す値（36）を合計し、その結果をトンネルに送信します。VI 実行終了時の数値表示器の値は **45** です。

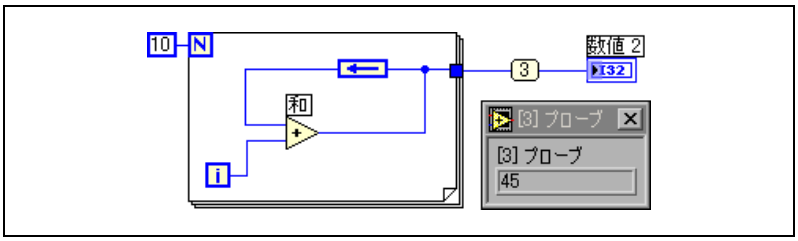


図 8-5　For ループで最後の値を渡す

フィードバックノードを初期化する

「フィードバックノード」を右クリックしてショートカットメニューから**初期化指定子端子**を選択し、初期化指定子端子をループの枠に追加して、ループを初期化します。「フィードバックノード」を選択した場合や、初期化したシフトレジスタを「フィードバックノード」に変換した場合は、ループに初期化指定子端子が自動的に表示されます。「フィードバックノード」を初期化すると、VI の実行時に初めてループが実行するときに「フィードバックノード」が渡す初期値がリセットされます。「フィードバックノード」を初期化しない場合、ループが一度も実行されないと、「フィードバックノード」はノードに最後に書き込まれた値またはデータタイプのデフォルト値を渡します。初期化指定子端子の入力が配線されていない場合、「フィードバックノード」の初期入力は前回の実行の最後で得た値になります。

シフトレジスタをフィードバックノードに入れ替える

シフトレジスタを「フィードバックノード」に入れ替えるには、シフトレジスタを右クリックして、ショートカットメニューから**フィードバックノードと置換**を選択します。「フィードバックノード」をシフトレジスタに入れ替えるには、フィードバックノードを右クリックして、ショートカットメニューから**シフトレジスタと置換**を選択します。

タイミングを制御する

データがチャートにプロットされる速度などのプロセスの実行速度を制御する必要がある場合があります。ループ内で「待機（Wait）」関数を使用すると、ループが再実行されるまでの待機時間（ミリ秒単位）だけ待つことができます。

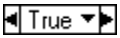
ループで「待機」関数を使用してメモリ使用状況を最適化する詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Memory and Speed Optimization」のセクションを参照してください。

ケースストラクチャとシーケンスストラクチャ

ケースストラクチャ、スタックシーケンスストラクチャ、フラットシーケンスストラクチャ、およびイベントストラクチャには複数のサブダイアグラムが含まれています。ケースストラクチャは、ストラクチャに渡される入力値に従って 1 つのサブダイアグラムを実行します。スタックシーケンスストラクチャおよびフラットシーケンスストラクチャは、そのすべてのサブダイアグラムを順番に実行します。ユーザがどのように VI を操作するかによって、イベントストラクチャはサブダイアグラムを実行します。

ケースストラクチャ

左図に示すケースストラクチャには、複数のダイアグラム、すなわちケースが含まれています。サブダイアグラムは一度に 2 つしか表示できず、またストラクチャは一度に 1 つしかケースを実行しません。入力値によって、どのサブダイアグラムが実行されるかが決まります。ケースストラクチャは、テキストベースのプログラミング言語におけるケースステートメントまたは `if...then...else` ステートメントに似ています。



ケースストラクチャの上部にあるケースセレクタラベルには、左図のように、中央にケースに対応するセレクタ値の名前があり、その両側に減分矢印と増分矢印があります。使用可能なケースをスクロールするには、減分矢印と増分矢印をクリックします。ケース名の隣にある下矢印をクリックして、プルダウンメニューからケースを選択することもできます。

左図に示すセレクタ端子に、入力値、すなわちセレクタを配線してどのケースを実行するかを決定します。整数、ブール値、文字列、または列挙型の値をセレクタ端子に配線する必要があります。セレクタ端子は、ケースストラクチャの左側の枠上の任意の場所に配置できます。セレクタ端子のタイプがブールの場合、ストラクチャには `TRUE` ケースと `FALSE` ケースがあります。セレクタ端子が整数、文字列、または列挙型の値の場合には、ストラクチャに任意の数のケースを使用することができます。

ケースストラクチャのデフォルトケースを指定して、範囲外の値を処理できるようにします。そうでない場合には、可能な入力値をすべてリストする必要があります。たとえば、セレクタが整数で `1`、`2`、および `3` のケースを指定する際、入力値が `4` または他の有効な整数値である場合は、実行するデフォルトケースを指定する必要があります。

ケースセレクタの値とデータタイプ

ケースセレクタラベルには、1 つの値またはリスト、および値の範囲を入力することができます。リストの場合は、カンマを使用して値を区切ります。数値範囲の場合、`10..20` で範囲を指定します。これは 10 から 20 のすべての数値を意味します。また、より大まかな範囲も使用できます。たとえば、`..100` は 100 以下のすべての数値を示し、`100..` は 100 以上のすべての数値を示します。また、`..5, 6, 7..10, 12, 13, 14` など、リストや範囲を結合することもできます。同じケースセレクタラベルに重なる範囲が含まれる値を入力すると、ケースストラクチャは、コンパクトな形式でラベルを再表示します。前述の例は、**`..10, 12..14`** として再表示されます。文字列の範囲の場合、`a..c` の範囲には、`a` および `b` のすべてが含まれますが、`c` は含まれません。`a..c,c` の範囲は、最後の値の `c` も含まれます。

ケースセレクタラベルに文字列や列挙された値を入力すると、`"red"`、`"green"`、`"blue"` などのように、値は引用符で囲まれて表示されます。ただし、文字列や列挙された値にカンマまたは範囲記号（`","` または `".."`) が含まれていない場合は、値を入力するときに引用符を入力する必要はありません。文字列の値には、英数字以外の文字に特別な円コードを使用します。たとえば、復帰文字には `\r`、改行文字には `\n`、タブには `\t` を使用します。これらの円コードのリストについては、『LabVIEW ヘルプ』を参照してください。

ケースストラクチャのセレクタ端子に接続されたワイヤのデータタイプを変更した場合、可能であれば、ケースストラクチャはケースセレクタの値を自動的に新しいデータタイプに変換します。たとえば、数値 19 を文字列に変換すると、文字列値は `"19"` になります。文字列を数値に変換する場合、LabVIEW は数値を表す文字列の値だけを変換します。他の値は文字列のまま残されます。数値をブール値に変換する場合、`0` は FALSE に変換され、`1` は TRUE に変換されます。それ以外のすべての数値は文字列になります。

セレクタ端子に配線されたオブジェクトとはタイプが異なるセレクタ値を入力すると、その値は赤で表示されて、ストラクチャを実行する前に値を削除または編集する必要があることを示し、VI は実行されません。また、浮動小数点演算特有の丸め誤差が生じる可能性があるため、浮動小数点値はケースセレクタ値として使用できません。浮動小数点値をケースに配線すると、LabVIEW は最も近い整数に値を丸めます。ケースセレクタラベルに浮動小数点値を入力すると、その値は赤で表示されて、ストラクチャを実行する前に値を削除または編集する必要があることを示します。

入力トンネルと出力トンネル

ケースストラクチャに複数の入力トンネルと出力トンネルを作成できます。入力はすべてのケースに利用できますが、ケースに各入力を使用する必要はありません。ただし、出力トンネルはケースごとに定義する必要があります。あるケースに出力トンネルを作成すると、他のすべてのケースの枠上の同じ位置にトンネルが現れます。1 つも出力トンネルが配線されていないと、そのストラクチャのすべての出力トンネルが白い正方形で表示されます。各ケース内の同じ出力トンネルに異なるデータソースを定義することができますが、各ケースのデータタイプに互換性がある必要があります。また、その出力トンネルを右クリックしてショートカットメニューから**未配線の場合はデフォルトを使用**を選択して、すべての未配線のトンネルのデータタイプにデフォルト値を使用することもできます。

エラー処理にケースストラクチャを使用する

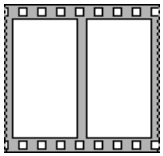
ケースストラクチャのセレクタ端子にエラークラスタを配線すると、ケースセレクタのラベルはエラーおよびエラーなしの 2 つのケースを表示します。また、ケースストラクチャの枠の色も、エラーの場合は赤、エラーなしの場合は緑に変わります。ケースストラクチャは、エラー状態に基づいて適切なケースサブダイアグラムを実行します。エラー処理の詳細については、第 6 章の「VI の実行とデバッグ」の「エラー処理」のセクションを参照してください。

シーケンスストラクチャ

シーケンスストラクチャには、順番に実行される 1 つまたは複数のサブダイアグラム、すなわちフレームが含まれています。シーケンスストラクチャは、LabVIEW で一般的に使用されます。シーケンスストラクチャの使用の詳細ついては、この章の「シーケンスストラクチャを使用する」および「シーケンスストラクチャの過度な使用を避ける」のセクションを参照してください。

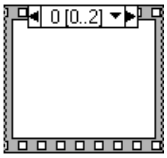
シーケンスストラクチャには、フラットシーケンスストラクチャとスタックシーケンスストラクチャの 2 種類があります。

フラットシーケンスストラクチャ

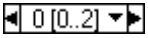


左図に示すフラットシーケンスストラクチャは、すべてのフレームを一度に表示し、最後のフレームまで左から右にフレームを実行します。シーケンスローカルの使用を避け、ブロックダイアグラムが一目で分かるようにするために、フラットシーケンスストラクチャを使用してください。フラットシーケンスストラクチャのフレームの追加や削除を行うと、ストラクチャは自動的にサイズ調整します。フラットシーケンスストラクチャでフレームを並べ替えるには、1 つのフレームから他のフレームに切り貼りします。

スタックシーケンスストラクチャ



左図に示すスタックシーケンスストラクチャは、各フレームを積み重ねていくため、一度に 1 つのフレームのみが表示されます。フレーム 0、次にフレーム 1 というように、最後のフレームまで実行します。スタックシーケンスストラクチャは、最後のフレームが実行された後のみにデータを返します。ブロックダイアグラム上のスペースを節約する場合は、スタックシーケンスストラクチャを使用します。



スタックシーケンスストラクチャの上部にある左図に示すシーケンスセレクタ識別子には、現在のフレーム数とフレームの範囲を含まれています。シーケンスセレクタ識別子を使用して、使用できるフレームを検索し、フレームを配置します。スタックシーケンスストラクチャのフレームラベルは、ケースストラクチャのケースセレクタラベルに似ています。フレームラベルには中央にフレーム番号があり、両側に増分矢印と減分矢印があります。使用可能なフレームをスクロールするには、減分矢印と増分矢印をクリックします。フレーム番号の隣にある下矢印をクリックして、プルダウンメニューからフレームを選択することもできます。スタックシーケンストラクチャを再配列するには、フレームの枠を右クリックして、ショートカットメニューから**このフレーム番号を変更**を選択してフレーム番号を選択します。

ケースセレクタラベルの場合とは異なり、フレームラベルに値を入力することはできません。スタックシーケンスストラクチャにフレームを追加したり、ストラクチャからフレームを削除したり、フレームの配置を変えたりすると、LabVIEW はフレームラベルの番号を自動的に調整します。

シーケンスストラクチャを使用する

シーケンスストラクチャを使用して、自然なデータ依存が存在しない場合の実行順序を制御します。他のノードからデータを受け取るノードは、データに関して他のノードに依存しているため、常に他のノードの実行が終了した後で実行されます。

ブロックダイアグラムの他の部分と同様に、シーケンスストラクチャの各フレーム内では、データ依存によってノードの実行順序が決まります。データ依存の詳細については、第 5 章「ブロックダイアグラムを作成する」の「データ依存と人工データ依存」のセクションを参照してください。

ケースストラクチャとは異なり、スタックシーケンスストラクチャのトンネルにはデータソースを 1 つしか使用できません。出力はどのフレームからでも可能ですが、データは、個々のフレームの実行が完了したときではなく、すべてのフレームの実行が完了したときにのみスタックシーケンスストラクチャから渡されます。ケースストラクチャの場合と同様に、入力トンネルのデータはすべてのフレームに使用することができます。

スタックシーケンスストラクチャのあるフレームから後続のフレームにデータを渡すには、左図に示すシーケンスローカル端子を使用します。データソースを含むフレームのシーケンスローカル端子に、外側を向いた矢印が表示されます。以降のフレーム内の端子には内向き矢印が含まれ、その端子がそのフレームのデータソースであることを示します。シーケンスローカル端子は、シーケンスローカルを配線した最初のフレームよりも前にあるフレーム内では使用できません。

シーケンスストラクチャの使用のサンプルについては、`examples¥general¥structs.llb` を参照してください。

シーケンスストラクチャの過度な使用を避ける

LabVIEW が持つ並列処理機能を活用するために、シーケンスストラクチャは過度に使用しないでください。シーケンスストラクチャによって実行順序は保証されますが、並列処理が妨げられます。たとえば、PXI、GPIB、シリアルポート、DAQ デバイスなどの I/O デバイスを使用する非同期タスクは、シーケンスストラクチャによって妨げられない限り、他の動作と並行して実行できます。シーケンスストラクチャにより、ブロックダイアグラムの一部が隠され、また左から右への自然なデータフローが妨げられます。

実行順序を制御する必要がある場合は、ノード間のデータ依存を確立することを検討してください。たとえば、エラー I/O を使用して、I/O 動作の実行順序を制御できます。エラー I/O の詳細については、第 6 章「VI の実行とデバッグ」の「エラー処理」のセクションを参照してください。

また、シーケンスストラクチャの異なるフレームから表示器を更新する場合には、シーケンスストラクチャを使用しないでください。たとえば、テストアプリケーションで使用される VI には、現在進行中のテストの名前を表示する**ステータス**表示器がある場合もあります。各テストが異なるフレームから呼び出されたサブ VI である場合、図 8-6 のスタックシーケンスストラクチャの不良ワイヤが示すように、各フレームから表示器を更新することはできません。

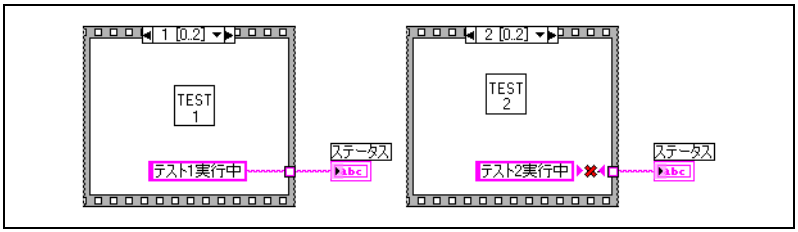


図 8-6　スタックシーケンスストラクチャの異なるフレームから表示器を更新する

スタックシーケンスストラクチャのすべてのフレームは、データがストラクチャから渡される前に実行されるため、**ステータス**表示器に値を割り当てられるのは 1 つのフレームのみです。

代わりに、図 8-7 に示すように、ケースストラクチャと While ループを使用してください。

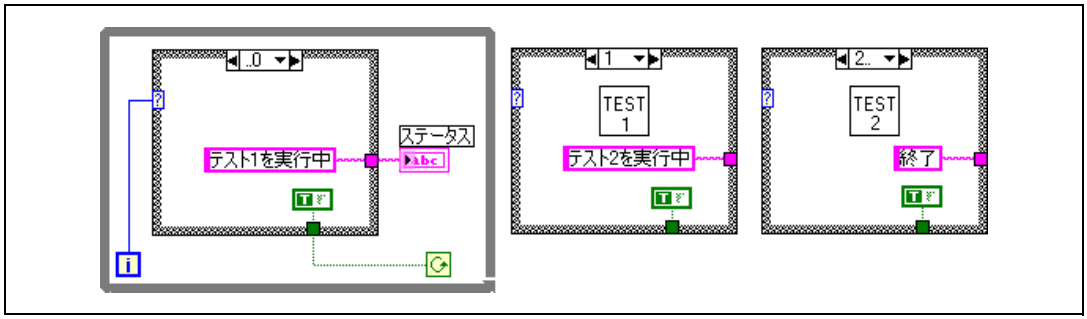


図 8-7　ケースストラクチャの異なるフレームから表示器を更新する

ケースストラクチャの各ケースは、シーケンスストラクチャのフレームと同じです。While ループの各繰り返しは、次のケースを実行します。**ステータス**表示器は、各ケースについて VI のステータスを表示します。ケースが実行するたびにストラクチャからデータが渡されるため、**ステータス**表示器は対応するサブ VI を呼び出すケースの前のケースで更新されます。

シーケンスストラクチャとは異なり、ケースストラクチャはどのケースにおいてもデータを渡して While ループを終了することができます。たとえば、最初のテストの実行中にエラーが発生した場合、ケースストラクチャは、条件端子に FALSE を渡してループを終了することができます。ただし、シーケンスストラクチャの場合には、エラーが発生した場合でもすべてのフレームを実行する必要があります。

シーケンスストラクチャを入れ替える

フラットシーケンスストラクチャをスタックシーケンスストラクチャに変換するには、フラットシーケンスストラクチャを右クリックして、ショートカットメニューから**スタックシーケンスと置換**を選択します。スタックシーケンスストラクチャをフラットシーケンスストラクチャに入れ替えるには、スタックシーケンスストラクチャを右クリックして、ショートカットメニューから**入替え→フラットシーケンスと置換**を選択します。

9

イベント駆動型プログラミング

LabVIEW は、データフローによってブロックダイアグラムの要素の実行順序が決まるデータフロー型プログラミング環境です。イベント駆動型プログラミングにより、LabVIEW のデータフロー環境を拡張し、ユーザによるフロントパネルの操作およびその他の非同期イベントによるダイアグラムの実行制御をより柔軟に行うことができます。

メモ イベント駆動型プログラミングは、LabVIEW 開発システムおよびプロフェッショナル開発システムでのみ使用できます。この機能を使って作成された VI を LabVIEW 基本パッケージで実行することはできますが、イベント処理コンポーネントを再構成することはできません。

詳細については

アプリケーションでイベントを使用する方法の詳細については、『LabVIEW ヘルプ』を参照してください。

イベントとは

イベントは、発生したアクションの非同期通知です。イベントは、ユーザインタフェース、外部 I/O、またはプログラムの他の部分から発生します。ユーザインタフェースのイベントには、マウスのクリック、キーボード操作などが含まれます。外部 I/O イベントには、データ集録が完了したときまたはエラー状態が発生したときに信号を送るハードウェアタイマまたはトリガが含まれます。他の種類のイベントは、プログラム的に生成され、プログラムの他の部分と通信するために使用されます。LabVIEW は、ユーザインタフェースをサポートし、プログラム的にイベントを生成しますが、外部 I/O イベントはサポートしません。

イベント駆動型プログラムでは、システムで発生したイベントは実行フローに直接影響します。反対に、手続きプログラムは、あらかじめ決められた順番で実行されます。通常、イベント駆動型プログラムはイベントの発生を待機するループを含み、イベントに反応してコードを実行し、反復して次のイベントを待機します。各イベントに対するプログラムの反応は、特定のイベントのために書かれたコードによって決まります。イベント駆動型プログラムの実行順序は、発生するイベントと、イベントの発生

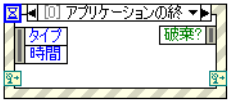
順序によって決まります。プログラムの中には、処理するイベントが頻発するために頻繁に実行する部分もあれば、イベントが発生しないために全く実行しない部分もあります。

イベントを使用する利点

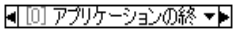
LabVIEW でユーザフェースイベントを使用すると、フロントパネルでのユーザによるアクションをブロックダイアグラムの実行と同期させることができます。イベントを使用すると、ユーザが特定のアクションを行うたびに、特定のイベント処理ケースを実行することができます。イベントを使用しない場合、ブロックダイアグラムは、フロントパネルのオブジェクトの状態をループ内でポーリングして、変更があったかを確認する必要があります。フロントパネルをポーリングするには、CPU 時間がかかり、急に発生した変更は検出できない場合があります。イベントを使用することによって、特定のユーザのアクションに反応することができ、フロントパネルをポーリングしてユーザが実行したアクションを検出する必要がなくなります。代わりに、LabVIEW は指定した操作が実行されるたびに、ブロックダイアグラムに通知します。イベントを使用すると、プログラムが必要とする CPU 処理が減り、ブロックダイアグラムのコードの簡略化、そしてユーザが実行したすべての操作にブロックダイアグラムが応答することが保証されます。

プログラム的に生成されたイベントを使用して、データ依存関係を持たないプログラムの異なる部分の間で通信を行うことができます。プログラム的に生成されたイベントは、ユーザインタフェースのイベントと同様に多くの利点を持ち、同じイベント処理コードを共有することができるため、イベントとキューを使用したステートマシンなど、高度のアキテクチャを容易に実装できるようにになります。

イベントストラクチャのコンポーネント



VI でイベントを処理するには、左図に示すイベントストラクチャを使用します。イベントストラクチャは、「ノーティフィケーションを待機（Wait On Notification）」関数を使用するケースストラクチャと似ています。イベントストラクチャは、複数のケースを持つことができ、各ケースは個別のイベント処理ルーチンです。1 つまたは複数のイベントを処理できるように各ケースを構成することができますが、一度に発生するイベントは 1 つののみです。イベントストラクチャは実行すると、構成されたイベントの 1 つが発生するまで待機し、次にそのイベントに対応するケースを実行します。イベントストラクチャは、イベントを 1 つ処理した後に実行を終了します。暗示的にループし、複数のイベントを処理することはありません。「ノーティフィケーションを待機」関数のように、イベントストラクチャは、イベント通知の待機中にタイムアウトする場合があります。タイムアウトが発生すると、特定のタイムアウトケースが実行されます。



左図に示すイベントストラクチャの上部にあるイベントセレクタラベルは、現在表示されているケースを実行するイベントを示します。他のイベントケースを表示するには、ケース名の隣にある下矢印をクリックして、ショートカットメニューから他のケースを選択します。

(Windows) セレクタラベル上にカーソルを移動し、マウスホイールを動かしている間に <Ctrl> キーを押す方法もあります。**(UNIX)** <Meta> キーを押します。

左図に示すイベントストラクチャの左上隅にあるタイムアウト端子は、イベントがタイムアウトになるまで待機する時間をミリ秒で指定します。デフォルト値は –1 で、これはイベント発生まで無限に待機することを示します。値をタイムアウト端子に配線した場合、タイムアウトケースを指定する必要があります。

左図に示す「イベントデータノード（Event Data Node）」は「名前でバンドル解除（Unbundle By Name）」関数に似た動作をします。このノードは各イベントケースの枠の内側に表示されます。ノードは、イベントが発生したときに LabVIEW が提供するデータを出力します。このノードを垂直方向にサイズ変更して、データ項目を追加し、ノード内の各データ項目を設定して任意のイベントデータ要素にアクセスできます。ノードは、どのイベントがケースを処理するよう構成されたかによって、イベントストラクチャの各ケースで異なるデータ要素を表示します。ある 1 つのケースを複数のイベントを処理するように構成した場合、「イベントデータノード」はそのケースに対して構成されたすべてのイベントに共通するイベントデータ要素のみを表示します。

左図に示す「イベントフィルタノード（Event Filter Node）」は「イベントデータノード」と似ています。このノードはフィルタイベントケースの右側の枠の内側に表示されます。ノードは、「イベントデータノード」から提供されるデータのうちイベントケースが変更できるデータを表示します。ノードは、どのイベントをケースが処理するように構成したかによって、異なるデータ要素を表示します。デフォルトでは、これらの項目は「イベントデータノード」の対応するデータ項目の場所に配置されています。値を「イベントフィルタノード」のデータ項目に配線していないと、そのデータ項目は変更されません。フィルタイベントの詳細については、この章の「通知およびフィルタイベント」のセクションを参照してください。

左図に示すダイナミックイベント端子は、イベントストラクチャを右クリックして、ショートカットメニューから**ダイナミックイベント端子を表示**を選択することによって使用できるようになります。これらの端子は、ダイナミックイベント登録のみで使用されます。これらの端子の使用方法の詳細については、この章の「イベントの動的登録」および「登録を動的に変更する」のセクションを参照してください。

メモ　ケースストラクチャと同様に、イベントストラクチャでもトンネルを使用することができます。ただし、デフォルトでは、各ケースに対してイベントストラクチャの出力トンネルを配線する必要はありません。未配線のすべてのトンネルは、そのトンネルデータタイプのデフォルト値を使用します。トンネルを右クリックして、ショートカットメニューで**未配線の場合はデフォルトを使用**を選択解除すると、すべてのケースでトンネルを配線する必要があるデフォルトのケースストラクチャの動作に戻ります。

データタイプのデフォルト値の情報については、『LabVIEW ヘルプ』を参照してください。

通知およびフィルタイベント

ユーザインタフェースイベントには、通知とフィルタの 2 種類があります。

通知イベントは、ユーザが制御器の値を変更した場合など、ユーザのアクションがすでに発生していることを示します。イベントが発生して LabVIEW が処理した後に、通知イベントを使用してイベントに応答します。特定のオブジェクトの同じ通知イベントに応答するように、イベントストラクチャをいくつでも構成することができます。イベントが発生すると、LabVIEW はイベントをパラレルで処理するよう構成されているイベントストラクチャにそのイベントのコピーを送信します。

フィルタイベントは、ユーザがあるアクションを実行したことを LabVIEW が処理する前に知らせるため、ユーザインタフェースとの対話にプログラムがどう応答するかをカスタマイズすることもできます。フィルタイベントを使用すると、イベントの処理に関わることができ、場合によってはイベントのデフォルト動作に優先することもできます。フィルタイベントのイベントストラクチャケースでは、LabVIEW が処理を終了する前にイベントデータの確認や変更を行ったり、変更によって VI に影響がおよぶのを防ぐため、イベント全体を破棄することができます。たとえば、VI のフロントパネルをユーザが対話式に閉じるのを防ぐため、「パネルを閉じる ?」イベントを破棄するようにイベントストラクチャを構成することができます。フィルタイベントは、通知イベントと区別しやすくするため、「パネルを閉じる ?」というように、名前の最後に疑問符が付いています。ほとんどのフィルタイベントは、同じ名前で疑問符が付かない通知イベントが関連付けられています。その通知イベントは、イベントが破棄されなかった場合にフィルタイベントの後に LabVIEW が生成するイベントです。

通知イベントと同様に、特定のオブジェクトで同じフィルタイベントに応答するように、イベントストラクチャをいくつでも構成することができます。ただし、LabVIEW はフィルタイベントを、イベントに構成された各イベントストラクチャに順番に送信します。LabVIEW が各イベントストラクチャにイベントを送信する順番は、イベントが登録された順番によっ

て決定されます。イベント登録の詳細については、この章の「LabVIEW でイベントを使用する」のセクションを参照してください。各イベントストラクチャは、LabVIEW が次のイベントストラクチャに通知する前に、イベントのイベントケースを完了するする必要があります。イベントストラクチャがイベントデータを変更した場合、LabVIEW はチェーン内の次のイベントストラクチャに変更されたデータを渡します。チェーン内のイベントストラクチャがイベントを破棄した場合、LabVIEW はチェーン内に残っているイベントストラクチャにイベントを渡しません。LabVIEW は、構成されたすべてのイベントストラクチャが一度も破棄せずにイベントを処理した場合にのみ、イベントをトリガしたユーザアクションの処理を完了します。

メモ　ナショナルインスツルメンツは、イベントを破棄またはイベントデータを変更する方法でユーザアクションを処理する場合は、フィルタイベントを使用することをお勧めします。ユーザが特定のアクションを行ったかどうかを検出するのみの場合は、通知イベントを使用します。

この章の「イベントストラクチャのコンポーネント」のセクションで説明されてる、フィルタイベントを処理するイベンストラクチャケースは、「イベントフィルタノード」を持っています。新しい値をこれらの端子に配線することによって、イベントデータを変更できます。データ項目を配線しない場合、項目は変更されません。TRUE 値を**破棄？**端子に配線することによって、イベントを完全に破棄できます。

メモ　イベントストラクチャ内にある 1 つのケースで、通知イベントとフィルタイベントの両方を処理することはできません。ケースは複数の通知イベントを処理できますが、すべてのイベントのイベントデータ項目が全く同じである場合のみ、複数のフィルタイベントを処理できます。

LabVIEW でイベントを使用する

LabVIEW は複数の異なるイベントを生成できます。必要のないイベントを生成することを避けるには、イベント登録を使用して、LabVIEW がどのイベントを通知するかを指定できます。LabVIEW は、静的および動的の 2 種類のイベント登録モデルをサポートしています。

静的登録では、VI のフロントパネル上のどのイベントをその VI のブロックダイアグラム上の各イベントストラクチャケースで処理するかを指定できます。LabVIEW は、VI の実行時にこれらのイベントを自動登録します。各イベントは、VI のフロントパネル上の制御器、VI のフロントパネルウィンドウ全体、または LabVIEW アプリケーションに関連付けられています。異なる VI のフロントパネルのイベントを処理するように静的にイベントストラクチャを構成することはできません。実行時にイベントス

トラクチャがそのイベントを処理するかを変更できないため、構成は静的です。静的登録の詳細については、この章の「イベントの静的登録」のセクションを参照してください。

VI サーバとイベント登録を統合し、アプリケーション、VI、および制御器リファレンスを使用することによって静的登録の制限を取り除き、実行時にどのオブジェクトのイベントを生成するかを指定することができます。動的登録は、LabVIEW がどのイベントを生成するか、そしていつ生成するかを制御する上での柔軟性があります。ただし動的登録は、イベントストラクチャで構成された情報を使用して登録を自動処理するのではなく、ブロックダイアグラム関数を持つ VI サーバリファレンスを使用して明示的に登録および登録解除する必要があるため、静的登録よりも複雑です。動的登録の詳細については、この章の「イベントの動的登録」のセクションを参照してください。

メモ　通常、LabVIEW はアクティブなフロントパネル上での直接的なユーザによる操作の結果としてのみ、イベントを生成します。LabVIEW は、VI サーバ、グローバル変数、ローカル変数、DataSocket などをユーザが使用した場合、「値変更」などのイベントを生成しません。プログラム的に値変更イベントを生成する方法については、『LabVIEW ヘルプ』の「値（信号）」プロパティを参照してください。通常、キューおよびノーティファイアの代わりにプログラム的にイベントを生成できます。

LabVIEW によって表示されるイベントデータには、タイムスタンプ、どのイベントが発生したかを示す列挙体、およびイベントをトリガしたオブジェクトへの VI サーバリファレンスが常に含まれます。タイムスタンプは、2 つのイベント間で通過した時間を計算したり、オカーレンスの順番を決定したりするために使用できるミリ秒カウンタです。イベントを生成したオブジェクトへのリファレンスは、そのオブジェクトの VI サーバクラスに厳密に分類化されています。アプリケーション、VI、および制御器など、どの種類のオブジェクトがイベントを生成しかによって、イベントはクラスに分類されます。ある 1 つのケースが異なる VI サーバクラスのオブジェクトの複数のイベントを処理する場合、リファレンスタイプはすべてのオブジェクトに共通の親クラスとなります。たとえば、イベントストラクチャの 1 つのケースがデジタル数値制御器およびカラーランプ制御器のイベントを処理するように構成した場合、デジタル数値とカラーランプ制御器は数値クラスであるため、イベントソース制御リファレンスは数値となります。VI サーバクラスの使用方法の詳細については、第 17 章「VI をプログラム的に制御する」を参照してください。

メモ　VI クラスと制御器クラスの両方で同じイベントを登録すると、LabVIEW はまず VI イベントを生成します。LabVIEW に含まれるオブジェクトのイベントが生成される前に、クラスタなどのコンテナオブジェクトの制御器イベントが生成されます。コンテナオブジェクトの VI イベントまたは制御器イベントのイベント

ストラクチャケースがイベントを破棄した場合、LabVIEW はそれ以上イベントを生成しません。

ブロックダイアグラムにある各イベントストラクチャおよび「イベント登録（Register For Events）」ノードは、イベントを格納するために使用されるキューを保有しています。イベントが発生すると、LabVIEW はイベント登録された各キューにイベントをコピーします。イベントストラクチャは、キュー内のすべてのイベントと、イベントストラクチャのダイナミックイベント端子に配線した「イベント登録ノード」のキュー内のイベントを処理します。LabVIEW はこれらのキューを使用して、登録された各イベントストラクチャに、イベントが発生する順番にイベントが確実に送信されることを確認します。

デフォルトでは、すべてのイベントストラクチャがイベントの処理を終了するまで、イベントを生成するオブジェクトを含むフロントパネルはロックされています。フロントパネルがロックされている間、LabVIEW はフロントパネルの動作を処理しませんが、それらの動作をバッファに配置し、フロントパネルがロック解除されたときに処理します。フロントパネルをロックすることによって、ウィンドウを移動する、スクロールバーを操作する、**実行停止**ボタンをクリックするなどの、特定の動作は影響を受けません。通知イベントのフロントパネルのロックを無効にするには、**イベントを編集**ダイアログボックスのオプションのチェックボックスをオフにします。待機しているイベントを完全に処理できないうちに LabVIEW の内部状態が変更されないようにするため、フロントパネルのロックはフィルタイベントに対して有効になっている必要があります。

LabVIEW は、イベントストラクチャがイベントを処理しようと待機していなくても、イベントを生成することができます。イベントストラクチャは一回の実行につき 1 つのイベントしか処理できないため、イベントストラクチャを While ループ内に配置して、発生するすべてのイベントをイベントストラクチャが処理できるようにします。

注意　イベントを処理するためのイベントストラクチャが実行しておらず、フロントパネルのロックが有効になっている場合、VI のユーザインタフェースは応答しなくなります。この場合、**実行停止**ボタンをクリックして VI を停止します。パネルのロックを無効にするには、イベントストラクチャを右クリックして、**イベントを編集**ダイアログボックスで**このイベントのイベントケースが完了するまでフロントパネルをロック**のチェックボックスをオフにします。フィルタイベントのフロントパネルのロックをオフにすることはできません。

イベントストラクチャ使用の際の注意事項および使用できるイベントについては、『LabVIEW ヘルプ』を参照してください。

イベントの静的登録

イベントの静的登録は、ユーザインタフェースイベントのみで利用できます。**イベントを編集**ダイアログボックスを使用して、静的に登録されたイベントを処理するようにイベントストラクチャを構成します。イベントソースを選択します。アプリケーション、VI、または個々の制御器を選択できます。「パネルサイズ変更」、「値変更」など、イベントソースが生成可能な特定のイベントを選択します。アプリケーションの要件に応じて、ケースを編集してイベントデータを処理します。**イベントを編集**ダイアログボックスおよびイベントの静的登録の詳細については、『LabVIEW ヘルプ』を参照してください。

イベントストラクチャを含む VI が実行されると、LabVIEW は自動的および透過的にイベントを静的登録します。LabVIEW は、その VI が実行されている間、または他の実行されている VI がその VI をサブ VI として呼び出した場合のみに、VI のイベントを生成します。

VI を実行すると、最上位 VI およびブロックダイアグラム上で VI が呼び出すサブ VI の階層は、予約という実行状態に設定されます。VI が予約状態にある場合は、VI を編集したり**実行**ボタンをクリックすることができません。これは、親 VI が実行されている間は、VI がサブ VI として呼び出される可能性があるからです。VI が予約状態に設定されている場合、その VI のブロックダイアグラムのイベントストラクチャで静的に構成したイベントは自動的に登録されます。最上位 VI の実行が終了すると、そのサブ VI 階層はアイドル状態に設定され、イベントは自動的に登録解除されます。

イベントの静的登録のサンプルのについては、`examples¥general¥uievents.llb` を参照してください。

イベントの動的登録

イベントの動的登録により、どのイベントを LabVIEW が生成するか、いつ生成するかを完全に制御できるようになります。イベント生成をアプリケーションの一部のみで発生させる場合、アプリケーションの実行中にイベントを生成する VI または制御器を変更する場合に、動的登録を使用します。動的登録では、イベントが生成される VI 内のみだけではなくサブ VI 内でもイベントを処理できます。

動的登録されたイベントの処理には、以下の主な 4 つのステップを実行します。

1. イベントを処理するオブジェクトへの VI サーバリファレンスを取得する。
2. VI サーバリファレンスを「イベント登録ノード」に配線することによって、オブジェクトのイベントを登録する。

3. イベントストラクチャを While ループで囲み、終了条件が発生するまでオブジェクトでイベントを処理する。
4. 「イベント登録解除（Unregister For Events）」関数を使用して、イベントの生成を停止します。

オブジェクトを動的にイベント登録するには、まずオブジェクトへの VI サーバリファレンスを取得します。「アプリケーションリファレンスを開く（Open Application Reference）」関数および「VI リファレンスを開く（Open VI Reference）」関数を使用して、アプリケーションと VI のリファレンスを取得します。制御器のリファレンスを取得するには、「プロパティノード」を使用してその制御器の VI をクエリするか、制御器を右クリックしてショートカットメニューから**作成→リファレンス**を選択して制御器リファレンス定数を作成します。VI サーバリファレンスの使用の詳細については、第 17 章「VI をプログラム的に制御する」の「アプリケーションリファレンスと VI リファレンス」のセクションを参照してください。

「イベント登録（Register For Events）」関数を使用して、イベントを動的に登録します。「イベント登録ノード」のサイズを変更して、1 つまたは複数のイベントソース入力を表示できます。アプリケーション、VI、または制御器リファレンスを各イベントソース入力に配線します。各入力を右クリックして、登録するイベントを**イベント**ショートカットメニューから選択します。選択できるイベントは、各イベントソース入力に配線した VI サーバリファレンスのタイプによって異なります。**イベント**ショートカットメニューにあるイベントは、イベントを静的登録したときに**イベントを編集**ダイアログボックスに表示されるイベントと同じイベントです。「イベント登録ノード」が実行されると、LabVIEW は各イベントソース入力リファレンスに関連付けられたオブジェクトを指定されたイベントに登録します。イベント登録すると、LabVIEW はイベントストラクチャが処理するまで発生したイベントをキューに入れます。ノードが実行される前にアクションが発生すると、他のオブジェクトが以前に登録されていない限り、イベントは発生しません。

メモ　「プロパティノード」とは異なり、「イベント登録ノード」では左上の入力に配線する必要はありません。既存の登録を変更する場合にのみこの入力を使用します。イベントの再登録の詳細については、この章の「登録を動的に変更する」のセクションを参照してください。

イベントストラクチャを右クリックして、ショートカットメニューから**ダイナミックイベント端子を表示**を選択することによって使用できるダイナミックイベント端子は、シフトレジスタと同じような働きをします。左側の端子は、イベント登録 refnum またはイベント登録 refnum のクラスタを受け取ります。内側の右にある端子は、配線しないと左の端子と同じデータを送信します。ただし、イベント登録 refnum またはイベント登

録 refnum のクラスタを、「イベント登録ノード」を介して内側の右の端子に配線し、イベント登録を動的に変更することができます。ダイナミックイベント端子の使用の詳細については、この章の「登録を動的に変更する」のセクションを参照してください。

「イベント登録ノード」の出力は、イベント登録 refnum であり、LabVIEW が登録されたイベントについての情報をイベントストラクチャに渡すために使用する厳密なデータタイプです。イベント登録 refnum 上にカーソルを移動すると、**ヘルプ**ウィンドウに登録されたイベントを表示できます。イベント登録ノードを構成したら、「イベント登録ノード」の**イベント登録 refnum** 出力をイベントストラクチャの左側のダイナミックイベント端子に配線して、登録したイベントを処理するようにイベントストラクチャを構成します。イベント登録 refnum のワイヤを分岐するのは避けてください。複数のイベントストラクチャが 1 つのキューからイベントを取り出すため競合状態が発生し、予期しない動作の原因になる場合があります。

イベントストラクチャを構成して動的登録されたイベントを処理するには、**イベントを編集**ダイアログボックスを使用します。ダイアログボックスの**イベントソース**セクションには、動的登録されたイベントのソースの一覧を表示する**ダイナミック**小見出しが含まれています。イベントソース名は、イベントストラクチャに接続された「イベント登録ノード」に配線したリファレンスと同じ名前で、同じ順序で表示されます。**ダイナミック**リストから、使用するイベントソースを選択します。「イベント登録ノード」を使用して登録したイベントと同じイベントが、**イベント**セクションにハイライトされて表示されていることを確認します。イベントを選択した後に、アプリケーションの要件に応じてケースを編集してイベントデータを処理します。

イベントの生成を停止するには、イベントストラクチャの右側にあるダイナミックイベント端子を、イベントストラクチャを含む While ループの外側にある「イベント登録解除（Unregister For Events）」関数の**イベント登録 refnum** 入力に配線します。「イベント登録解除」関数が実行されると、LabVIEW はイベント登録 refnum が指定するすべてのイベントを登録解除し、イベント登録 refnum に関連付けられたイベントキューを破棄して、キューに残っているすべてのイベントを破棄します。イベントを登録解除しないと、イベントストラクチャを含む While ループが終了した後に、イベントを生成するアクションをユーザが実行した場合、LabVIEW はイベントを無限にキューに入れ続けます。この結果、イベントを構成してフロントパネルをロックすると VI が反応しなくなります。この場合、VI がアイドルになるとイベントキューが破棄されます。

また、LabVIEW は最上位 VI が実行を終了すると、すべてのイベントを自動的に登録解除します。ただし、ナショナルインスツルメンツでは、

特に長期実行されるアプリケーションにおいて、イベントを明示的に登録解除してメモリリソースを節約することをお勧めします。

ダイナミックイベントのサンプル

図 9-1 のブロックダイアグラムは、文字列制御器でマウス入力イベントを使用して、ダイナミックイベント登録を使用する方法を示しています。

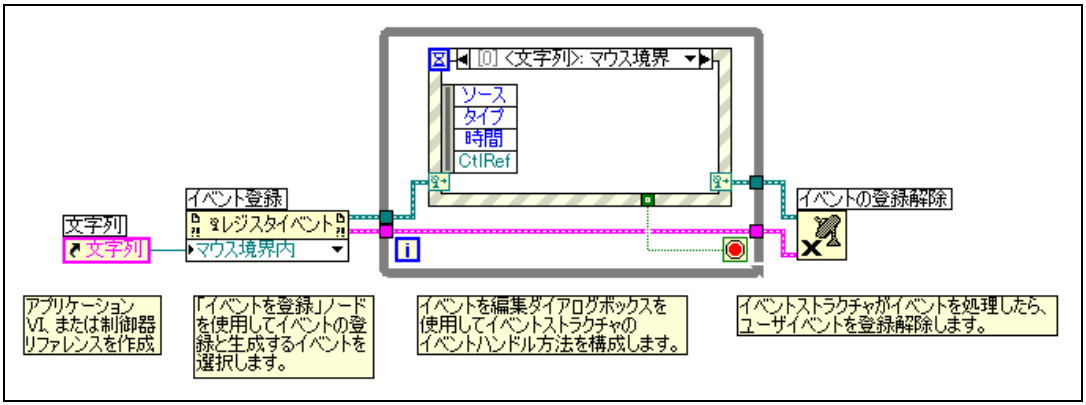


図 9-1　イベントを動的に登録する

イベントを動的に登録する方法の詳細については、『LabVIEW ヘルプ』を参照してください。

イベントの動的登録のサンプルについては、`examples¥general¥dynamicevents.llb` を参照してください。

登録を動的に変更する

イベントを動的に登録した場合、登録情報を実行時に変更し、LabVIEW がイベントを生成するオブジェクトを変更することができます。新規登録を作成するのではなく、refnum に関連付けられた既存の登録を変更したい場合は、「イベント登録ノード」の左上にある**イベント登録 refnum** 入力を配線します。**イベント登録 refnum** 入力を配線するとノードは自動的にサイズ変更して、イベント登録 refnum を当初に作成した「イベント登録ノード」で指定したタイプのリファレンス上に指定したイベントを表示します。**イベント登録 refnum** 入力が配線されている間は、関数を手動でサイズ変更したり再構成することはできません。

オブジェクトリファレンスを「イベント登録ノード」の**イベントソース**入力に配線し、**イベント登録 refnum** 入力も配線した場合、ノードは対応する元の「イベント登録ノード」の**イベントソース**入力を介して以前に登録されたリファレンスを置換します。非 Refnum 定数を**イベントソース**入力に配線して、個々のイベントを登録解除できます。**イベントソース**入

力を配線しないと、そのイベントの登録は変更されません。イベント登録 refnum に関連付けられたすべてのイベントを登録解除する場合は、「イベント登録解除」関数を使用します。

図 9-2 の例は、実行時に LabVIEW が生成するオブジェクトを動的に変更する方法を示します。以下のブロックダイアグラムが実行されると、LabVIEW は数値リファレンスを登録し、関連する**数値**制御器上でイベントが発生するのを待機します。**数値**制御器に対して「値変更」イベントが生成されると、「数値変更」イベントケースが「イベント登録ノード」を実行して、値変更イベントに登録された数値制御器を**数値**から**数値 2** に変更します。その後ユーザが**数値**制御器の値を変更しても、「値変更」イベントは生成されません。ただし、**数値 2** 制御器に対し変更が加えられると、「値変更」イベントは生成されます。LabVIEW が**数値 2** 制御器に対して「値変更」イベントを生成するたびに、「イベント登録ノード」は実行されますが、**数値 2** 制御器が「値変更」イベントにすでに登録されているため、影響はありません。

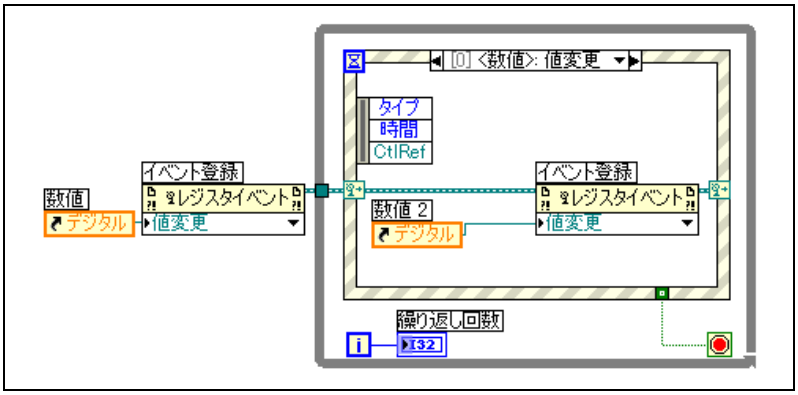


図 9-2　イベント登録を動的に変更する

メモ　静的に登録されたイベントを動的に変更することはできません。

ユーザイベント

独自のイベントをプログラム的に作成して、名前を付けることができます。このユーザ定義データを送信するイベントをユーザイベントと呼びます。キューやノーティファイアと同様、ユーザイベントを使用すると、アプリケーションの異なる部分が非同期で通信できるようになります。ユーザインタフェースとプログラム的に生成されたユーザイベントの両方を、同じイベントストラクチャ内で処理できます。

ユーザイベントを作成して登録する

ユーザイベントを定義するには、フロントパネル端子またはブロックダイアグラム定数などのブロックダイアグラムオブジェクトを「ユーザイベントを作成（Create User Event）」関数に配線します。オブジェクトのデータタイプにより、ユーザイベントのデータタイプが定義されます。オブジェクトのラベルがユーザイベント名になります。データタイプがクラスタの場合は、クラスタの各フィールドの名前とタイプによって、ユーザイベントのデータが定義されます。データタイプがクラスタ以外の場合、ユーザイベントはそのタイプの単一の値を送信し、オブジェクトのラベルはユーザイベントと単一データ要素の名前になります。

「ユーザイベントを作成」関数の**ユーザイベント出力**は、ユーザイベントの名前およびデータタイプを含む厳密に類別化された refnum です。「ユーザイベントを作成」関数の**ユーザイベント出力**は、イベント登録ノードの**イベントソース**入力に配線します。

ユーザイベントは、動的登録されたユーザインタフェースイベントと同じ方法で処理します。「イベント登録ノード」の**イベント登録 refnum** 出力を、イベントストラクチャの左側にあるダイナミックイベント端子に配線します。**イベントを編集**ダイアログボックスを使用して、イベントを処理するようにイベントストラクチャのケースを構成します。ユーザイベント名は、ダイアログボックスの**イベントソース**セクションにある**ダイナミック**小見出しに表示されます。

ユーザイベントデータ項目は、イベントストラクチャの左側の枠にある「イベントデータノード（Event Data Node）」に表示されます。ユーザイベントは通知イベントで、ユーザインタフェースイベントまたは他のユーザイベントと同じイベントストラクチャのイベントケースを共有できます。

ユーザイベントおよびユーザインタフェースイベントを組み合わせて、「イベント登録ノード」に配線することができます。

ユーザイベントを生成する

「ユーザイベントを生成（Generate User Event）」関数を使用して、ユーザイベントおよび関連データを、イベントを処理するように構成されたイベントストラクチャを介してアプリケーションの他の部分に渡します。「ユーザイベントを生成」関数は、ユーザイベント refnum およびイベントデータの値を受け入れます。データ値は、ユーザイベントのデータタイプと一致する必要があります。

ユーザイベントが登録されていない場合、「ユーザイベントを生成」関数は無効です。ユーザイベントが登録されても、待機しているイベントストラクチャがない場合、LabVIEW はイベントストストラクチャが実行されてイ

ベントを処理するまでユーザイベントとデータをキューに入れ続けます。別の「イベント登録ノード」を使用することによって、同じユーザイベントを何回も登録できます。この場合、「ユーザイベントを生成」関数が実行するたびに、イベント登録 refnum に関連付けられた各キューはユーザイベントおよび関連イベントデータの独自のコピーを受け取ります。

メモ　フロントパネル上でのユーザの操作をシミュレーションするには、既存のユーザインタフェースイベントと同じ名前およびデータタイプのイベントデータ項目を持つユーザイベントを作成します。たとえば、「値変更」ユーザインタフェースイベントがブール制御器に関連付けるのと同じイベントデータ項目である、`OldVal` と `NewVal` いう名前の 2 つのブールフィールドのクラスタを使用して、`MyValChg` というユーザイベントを作成することができます。シミュレーションされた `MyValChg` ユーザイベントと実際の「ブール値変更」イベントで、同じイベントストラクチャケースを共有できます。「ユーザイベントを作成ノード」がユーザイベントを生成した場合や、ユーザが制御器の値を変更した場合、イベントストラクチャはイベントケースを実行します。

ユーザイベントを登録解除する

ユーザイベントは、必要のなくなったら登録解除します。さらに、ユーザイベント refnum を「ユーザイベント破棄（Destroy User Event）」関数の**ユーザイベント**入力に配線すると、ユーザイベントを破棄することもできます。「イベント登録解除」関数の**エラー出力**を、「ユーザイベント破棄」関数の**エラー入力**に配線して、関数が正しい順序で実行されていることを確認します。

LabVIEW は最上位 VI が実行を終了すると、LabVIEW は自動的にすべてのイベントを登録解除し、既存のユーザイベントを破棄します。ただし、ナショナルインスツルメンツでは、特に長期実行されるアプリケーションにおいて、ユーザイベントを明示的に登録解除または破棄してメモリリソースを節約することをお勧めします。

ユーザイベントの例

図 9-3 のブロックダイアグラムはユーザイベントの使用方法を示します。ブロックダイアグラム定数クラスは、ユーザイベント名をマイデータ、イベントのデータタイプを文字列と呼ばれる文字列に指定します。「イベント登録ノード」は、ユーザイベント refnum をユーザイベントに登録します。While ループにあるイベントストラクチャは、イベントの発生を待機します。While ループと並列して、「ユーザイベントを生成」関数は、イベントストラクチャ内でユーザイベントケースを実行させるイベントを送信します。While ループが終了すると、VI はイベントを登録解除し、ユーザイベントを破棄します。

図 9-3 に示す VI を作成し、実行のハイライトを使用して、イベントデータが VI を介してノードからノードへどのように移動するかを表示します。

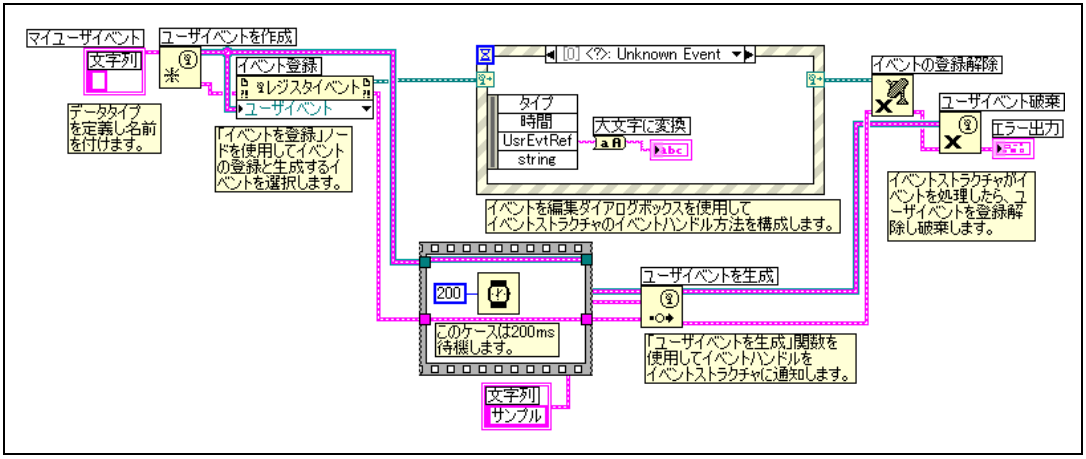


図 9-3　ユーザイベントを生成する

イベントの動的登録のサンプルについては、`examples¥general¥dynamicevents.llb` を参照してください。

10

文字列、配列、およびクラスタを使用してデータをグループ化する

データをグループ化するには、文字列、配列、およびクラスタを使用します。文字列は一連の ASCII 文字をグループ化します。配列は同じタイプのデータ要素をグループ化します。クラスタは混合タイプのデータ要素をグループ化します。

詳細については

文字列、配列、およびクラスタを使用してデータをグループ化する方法の詳細については、『LabVIEW ヘルプ』を参照してください。

文字列

文字列とは、表示可能または表示不可能な一連の ASCII 文字です。文字列は、プラットフォームに依存しない形式の情報やデータを提供します。
一般的な文字列の例には以下のようなものがあります。

- 単純なテキストメッセージを作成する。
- 数値データを文字列として計測器に渡し、その後文字列を数値に変換する。
- 数値データをディスクに保存する。数値を ASCII ファイルに保存するには、数値をディスクファイルに書き込む前に、まず数値を文字列に変換する必要がある。
- ダイアログボックスでユーザに指示を出す。

フロントパネルでは、文字列は表、テキスト入力ボックス、およびラベルとして表示されます。ブロックダイアグラムの文字列関数を使用して文字列を編集および操作します。ワードプロセッサや表計算といった他のアプリケーションで使用したり、他の VI や関数で使用するために、文字列をフォーマットします。

フロントパネルの文字列

文字列制御器を表示器を使用して、テキスト入力ボックスやラベルをシュミレーションします。文字列の制御器および表示器の詳細については、第 4 章「フロントパネルを作成する」の「文字列制御器および表示器」のセクションを参照してください。

文字列表示タイプ

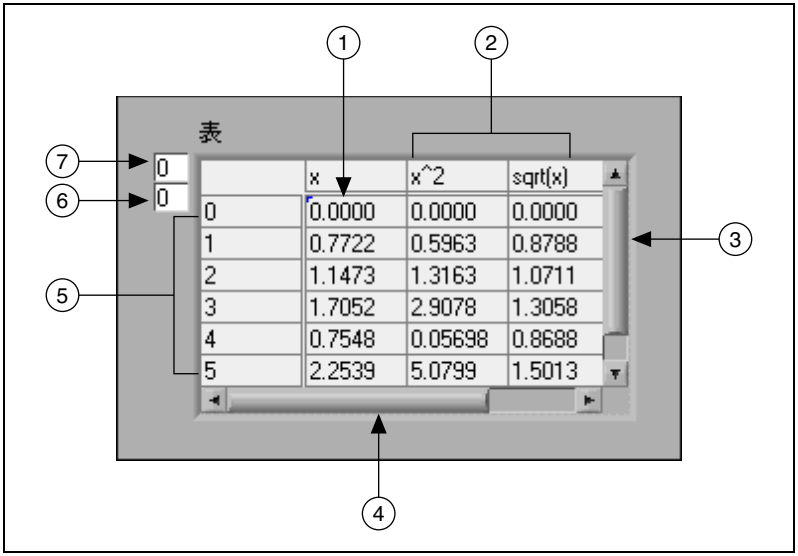
フロントパネルで文字列制御器または表示器を右クリックして、表 10-1 に示す表示タイプのいずれかを選択します。表には各表示タイプのサンプルメッセージも表示されています。

表 10-1　文字列表示タイプ

表示タイプ	説明	メッセージ
通常の表示	制御器のフォントを使用して印刷可能な文字を表示します。表示できない文字は四角で表示します。	4 つの表示タイプがあります。 「¥」は円記号です。
'¥'（円）コード表示	表示できないすべての文字を円コードで表示します。	4¥82¥C2¥82¥CC¥95¥¥¥8E¥A6¥83^¥83C¥83v¥82¥AA¥82¥A0¥82¥E8¥82¥DC¥82¥B7¥81B¥n¥81u¥81¥8F¥81v¥82¥CD¥89~¥8BL¥8D¥86¥82¥C5¥82¥B7¥81B
パスワード表示	スペースを含む各文字をアスタリスク（*）で表示します。	**********
16 進表示	各文字の ASCII 値を文字ではなく 16 進数で表示します。	3482 C282 CC95 5C8E A683 5E83 4383 7682 AA82 A082 E882 DC82 B781 420A 8175 818F 8176 82CD 897E 8B4C 8D86 82C5 82B7 8142

表

表制御器を使用して、フロントパネルに表を作成します。表の各セルは文字列であり、各セルは列と行により指定されます。したがって、表は 2 次元の文字列を表示します。図 10-1 は、表および表の各部を示します。配列の詳細については、この章の「配列」のセクションを参照してください。



1	指標値で示されるセル	5	行見出し
2	列見出し	6	水平指標
3	縦スクロールバー	7	垂直指標
4	横スクロールバー		

図 10-1　表の各部

文字列をプログラム的に編集する

文字列を編集するには、以下の手順に従って、文字列関数を使用します。

- 文字列内の文字またはサブ文字列を検索、取り出し、および置換する。
- 文字列内のすべてのテキストを大文字または小文字に変更する。
- 文字列内の一致パターンを検索して取り出す。
- 文字列から 1 行を取り出す。
- 文字列内のテキストを回転または逆にする。
- 複数の文字列を連結する。
- 文字列から文字を削除する。

文字列関数を使用して文字列を編集する方法のサンプルは、`examples¥general¥strings.llb` にあります。文字列をプログラム的に編集する際にメモリ使用を最小限にする詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Memory and Speed Optimization」のセクションを参照してください。

文字列をフォーマットする

他の VI、関数、またはアプリケーションでデータを使用する場合、データを文字列に変換してから、その文字列を VI、関数、またはアプリケーションが読める形式にフォーマットする必要がある場合がしばしばあります。たとえば、Microsoft Excel の場合、タブ、カンマ、またはスペースなどの区切り文字を含む文字列を使用する必要があります。これは、Excel では数字や語をセルに区切るために区切り文字を使用するからです。

たとえば、「ファイルに書込み（Write File）」関数を使用して数値の 1 次元配列をスプレッドシートに書き込む場合、配列を文字列にフォーマットし、各数値をタブなどの区切り文字で分割する必要があります。また、「スプレッドシートファイルに書込み（Write to Spreadsheet File）」VI を使用して数値の配列をスプレッドシートに書き込むには、「配列からスプレッドシート文字列に変換（Array to Spreadsheet String）」VI を使用して配列をフォーマットし、フォーマットと区切り文字を指定する必要があります。

以下のタスクを実行するには、文字列関数を使用します。

- 複数の文字列を連結する。
- 文字列から部分文字列を抽出する。
- データを文字列に変換する。
- ワープロや表計算アプリケーションで使用するために文字列をフォーマットする。

文字列をテキストおよびスプレッドシートファイルに保存するには、ファイル I/O の VI および関数を使用します。

指定子をフォーマットする

多くの場合、文字列をフォーマットするには、1 つまたは複数の指定子を文字列関数の**形式文字列**パラメータに入力する必要があります。フォーマット指示子コードは、文字列をデータに、またはデータを文字列に変換する方法を示します。LabVIEW は、変換コードを使用して、パラメータのテキスト形式を指定します。たとえば、形式指定子 `%x` は、16 進整数を文字列に、文字列を 16 進整数に変換します。

「文字列にフォーマット（Format Into String）」関数および「文字列からスキャン（Scan From String）」関数は、**形式文字列**パラメータに複数の形式指定子（拡張可能な関数に対する各入力または出力に 1 つずつ）を使用することができます。

「配列からスプレッドシート文字列に変換（Array To Spreadsheet String）」関数および「スプレッドシート文字列を配列に変換（Spreadsheet String To Array）」関数は、変換する入力が 1 つしかない

ため、**形式文字列**パラメータに 1 つの形式指定子しか使用しません。LabVIEW は、これらの関数にユーザが挿入した余分な指定子を特別な意味のない文字列として扱います。

数値と文字列

文字列データは ASCII 文字であり、数値データは ASCII 文字ではないという点で、数値データと文字列データは異なります。テキストファイルとスプレッドシートファイルには文字列のみが使用できます。テキストファイルまたはスプレッドシートファイルに数値データを書き込むには、まず数値データを文字列に変換する必要があります。

一連の数値を既存の文字列に追加するには、数値データを文字列に変換し、「文字列を連結（Concatenate Strings）」関数または他の文字列関数を使用して、新しい文字列を既存の文字列に追加します。文字列 / 数値変換を使用して、数値を文字列に変換します。

文字列にはグラフまたはチャートに表示する一連の数値を含めることもできます。たとえば、チャートにプロットする数値を含むテキストファイルを読み取ることができます。ただし、これらの数値は ASCII テキストであるため、まず数値を文字列として読み取ってから、その文字列を数値のセットに変換して、数値をチャートにプロットする必要があります。

図 10-2 は、一連の数値が含まれ、その文字列を数値に変換し、数値の配列を作成して、その数値をグラフにプロットする文字列を示します。

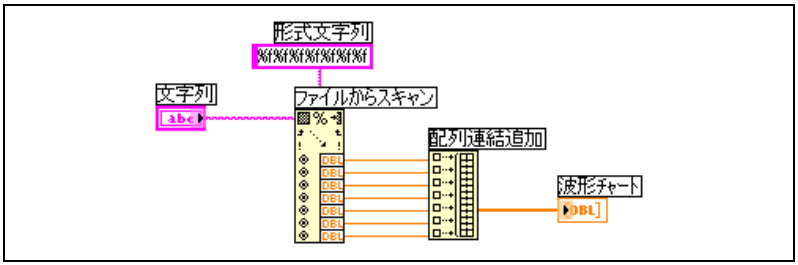


図 10-2　文字列を数値に変換する

データタイプを XML に変換する

拡張マークアップ言語（XML）は、タグを使用してデータを記述するフォーマット規格です。HTML タグとは異なり、XML タグは 1 つのデータをフォーマットする方法をブラウザに対して指定しません。その代わりに、XML タグは 1 つのデータを識別します。

たとえば、Web で書籍を販売している書店の場合、各書籍を以下の基準でライブラリに分類する必要があるとします。

- 本の種類（フィクションまたはノンフィクション）
- タイトル
- 著者
- 出版社
- 価格
- ジャンル
- あらすじ
- ページ数

各書籍に対して、XML ファイルを作成することができます。『Touring Germany's Great Cathedrals』というタイトルの本の XML ファイルは、以下のようなものになります。

```
<nonfiction>
<Title>Touring Germany's Great Cathedrals</Title>
<Author>Tony Walters</Author>
<Publisher>Douglas Drive Publishing</Publisher>
<PriceUS>$29.99</PriceUS>
<Genre>Travel</Genre>
<Genre>Architecture</Genre>
<Genre>History</Genre>
<Synopsis>This book fully illustrates twelve of
Germany's most inspiring cathedrals with full-color
photographs, scaled cross-sections, and time lines of
their construction.</Synopsis>
<Pages>224</Pages>
</nonfiction>
```

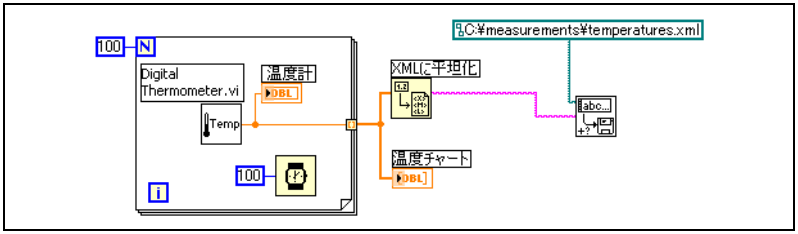
同様に、LabVIEW のデータを名前、値段、タイプによって分類することができます。ユーザ名の文字列制御器は XML で以下のように表記できます。

```
<String>
<Name>User Name</Name>
<Value>Reggie Harmon</Value>
</String>
```

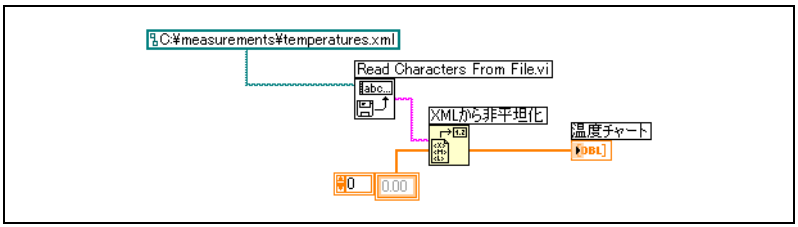
XML ベースのデータタイプ

LabVIEW のデータを XML に変換すると、データをファイルに保存したときに、データを説明するタグからデータの値、名前、およびタイプを簡単に識別できるように、データがフォーマットされます。たとえば、温度の値の配列を XML に変換して、データをテキストファイルに保存すると、各温度を識別する `<Value>` タグを見つけることによって温度を簡単に識別することができます。

「XML に平坦化 （Flatten to XML）」関数を使用して、LabVIEW データタイプを XML フォーマットに変換します。以下の例では、シミュレーションした温度を 100 個生成し、温度の配列をチャートにプロットした後、数字の配列を XML フォーマットに変換して XML データを `temperatures.xml` ファイルに書き込みます。



「XML から非平坦化（Unflatten from XML）」関数を使用して、XML フォーマットのデータタイプを LabVIEW データに変換します。以下の例では、`temperatures.xml` ファイルの 100 個の温度を読み取って、温度の配列をチャートにプロットします。



メモ　LabVIEW バリアントデータを XML に平坦化することはできますが、変数データを XML から非平坦化しようとすると、空の LabVIEW バリアントが生成されます。

データタイプを XML に変換する方法のサンプルについては、`examples¥file¥XMLex.llb` を参照してください。

LabVIEW XML スキーマ

LabVIEW は、確立された XML スキーマにデータを変換します。現在、カスタマイズスキーマを作成したり、各データに対して LabVIEW がタグを付ける方法を制御することはできません。また、VI または関数全体を XML に変換することもできません。

LabVIEW XML スキーマについては、`LabVIEW¥help` ディレクトリを参照してください。

データを配列やクラスタとグループ化する

データをグループ化するには、配列およびクラスタの制御器と関数を使用します。配列は同じタイプのデータ要素をグループ化します。クラスタは混合タイプのデータ要素をグループ化します。

配列

配列は要素および次元で構成されています。要素は配列を構成するデータです。次元は、配列の長さ、高さ、または深さです。配列は、1 以上の次元を持ち、メモリが許す限り、1 次元あたり $2^{31} - 1$ の要素を持つことができます。

数値、ブール値、パス、文字列、波形、およびクラスタデータタイプの配列を作成できます。同じタイプのデータの集合を処理する場合や同じ演算を繰り返し行う場合は、配列を使用することを検討してください。配列は、波形から収集したデータや、ループで生成されたデータ（各ループで配列の要素が 1 つずつ生成されます）を格納するのに適しています。

配列の配列は作成できません。しかし、複数次元の配列を使用したり、各クラスタが 1 つ以上の配列を含んでいるクラスタの配列を作成できます。配列に使用できる要素のタイプの詳細については、この章の「配列に関する制約」のセクションを参照してください。クラスタの詳細については、この章の「クラスタ」のセクションを参照してください。

指標

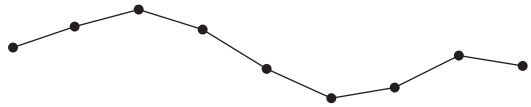
配列内の特定の要素を見つけるには、次元ごとに 1 つの指標が必要です。LabVIEW では、指標を使用すると、配列内を移動してブロックダイアグラム上の配列から要素、行、列およびページを取り出します。

配列の例

簡単な配列の例としては、太陽系の 9 つの惑星をリストしたテキストの配列があります。LabVIEW は、これを 9 つの要素を持つ文字列の 1 次元配列として表現します。

配列の要素には順番が付けられています。配列の指標を使用して、すべての特定の要素にすばやくアクセスできます。指標は 0 から始まります。つまり、指標は 0 ～ *n* − 1 の範囲になります。ここで、*n* は配列内の要素の数です。9 つの惑星の例では *n* = 9 で、指標は 0 ～ 8 の範囲になります。地球は 3 番目の惑星なので、指標は 2 になります。

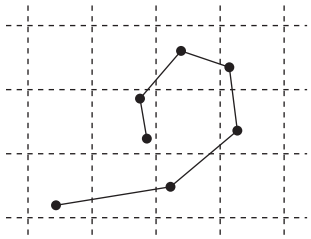
配列のもう 1 つの例として、図 10-3 のように、各連続要素が連続した時間間隔の電圧値を表す数値配列として表現された波形があります。



指標	0	1	2	3	4	5	6	7	8
電圧(V)	0.4	0.9	1.4	0.8	–0.1	–0.7	–0.3	0.3	0.2

図 10-3　数値配列の波形

より複雑な配列の例として、図 10-4 のように、点の配列として表現されたグラフがありますが、ここで各点は X および Y 座標を表す 2 つの数値を含むクラスタになっています。



指標	0	1	2	3	4	5	6
X 座標	0.4	2.2	3.3	3.2	2.4	1.8	1.9
Y 座標	0.2	0.5	1.3	2.3	2.6	1.9	1.2

図 10-4　点の配列のグラフ

前の例は 1 次元配列を使用しています。2 次元配列は要素をグリッドに格納します。要素を見つけるには（0 から始まる）列指標と行指標が必要です。図 10-5 は、6 列× 4 行の 2 次元配列を示しており、6 × 4 = 24 の要素を含んでいます。

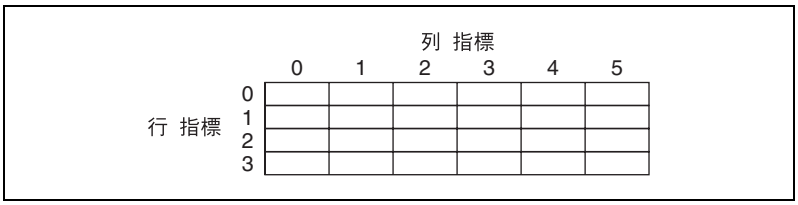
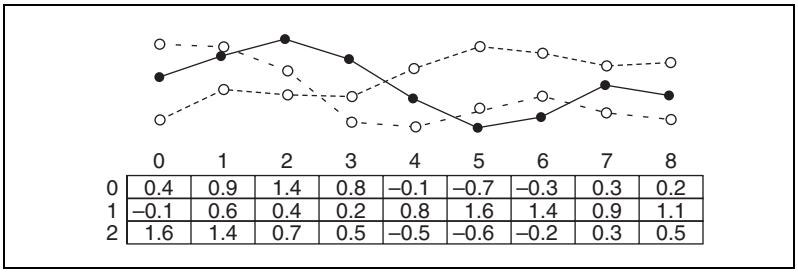


図 10-5　6 列× 4 行の 2 次元配列

たとえば、チェス盤には 8 列× 8 行の総計 64 の位置があります。各位置は空かチェスの駒が 1 つあるかのどちらかです。チェス盤は文字列の 2 次元配列として表現できます。各文字列は、盤上の対応する位置に置かれた駒の名前か、またはその場所に駒がない場合は空の文字列です。

配列に行を追加することによって、図 10-3 および図 10-4 の 1 次元配列の例を 2 次元に一般化できます。図 10-6 は、数値の 2 次元配列として表現された波形の集合を示しています。行指標で波形を選択し、列指標で波形上の点を選択します。



	0	1	2	3	4	5	6	7	8
0	0.4	0.9	1.4	0.8	–0.1	–0.7	–0.3	0.3	0.2
1	–0.1	0.6	0.4	0.2	0.8	1.6	1.4	0.9	1.1
2	1.6	1.4	0.7	0.5	–0.5	–0.6	–0.2	0.3	0.5

図 10-6　数値の 2 次元配列内の複数波形

その他の配列のサンプルについては、`examples¥general¥arrays.llb` を参照してください。配列の構築の詳細については、第 8 章「ループとストラクチャ」の「ループを使用して配列を作成する」のセクションを参照してください。

配列に関する制約

ほとんどすべてのデータタイプの配列を作成できますが、以下の例外があります。

- 配列の配列は作成できません。ただし、複数次元の配列を使用したり、「クラスタ配列作成（Build Cluster Array）」関数を使用して、各クラスタが 1 つ以上の配列を含むクラスタの配列を作成することはできます。
- サブパネル制御器の配列は作成することはできません。
- タブ制御器の配列は作成することができません。

- ActiveX 制御器の配列は作成することができません。
- チャートの配列は作成することができません。
- XY グラフの配列は作成することができません。

配列制御器、表示器、および定数を作成する

図 10-7 のようにフロントパネルに配列シェルを配置し、データオブジェクトまたは要素（数値、ブール値、文字列、パス、refnum、クラスタ制御器、またはクラスタ表示器など）を配列シェル内にドラッグして、フロントパネルに配列制御器または表示器を作成します。

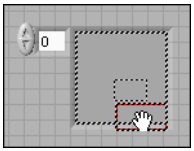


図 10-7　配列シェル

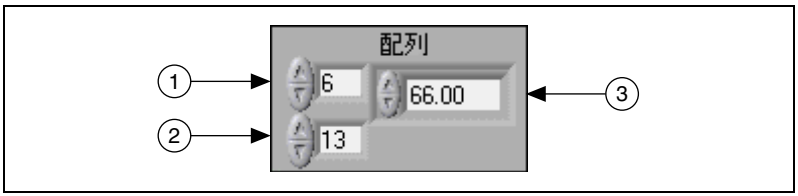
配列シェルは、小さなブール制御器でも大きな 3 次元グラフでも、新しいオブジェクトに合わせて自動的にサイズを変更します。フロントパネル上に多次元配列を作成するには、指標表示を右クリックして、ショートカットメニューから**次元を追加**を選択します。希望する次元の数値になるまで、指標表示のサイズを変更することができます。次元を 1 つずつ削除するには、指標表示を右クリックし、ショートカットメニューから**次元を削除**を選択します。また、指標表示のサイズを変更して、次元を削除することもできます。

フロントパネル上に特定の要素を表示するには、指標表示に指標番号を入力するか、指標表示上の矢印を使用して、該当する番号に移動します。

ブロックダイアグラム上に配列定数を作成するには、**関数**パレットで配列定数を選択して配列シェルをフロントパネルに置き、次に文字列定数、数値定数、またはクラスタ定数を配列シェル内に置きます。配列定数は、他の配列と比較する基準として使用できます。

配列指標表示

2 次元配列には行と列があります。図 10-8 のように、左側の 2 つのボックスの上の表示が行指標で、下の表示が列指標です。行と列の表示の右側にある表示は、指定された位置の値を示しています。図 10-8 は、6 行目および 13 列目の値が **66** であることを示しています。



1　行指標	2　列指標	3　行、列での値

図 10-8　配列制御器

行と列は 0 から始まり、1 番目の列は 0、2 番目の列は 1、以下同様に続きます。以下の配列の指標表示を行 1 および列 2 に変更すると、値 6 が表示されます。

0	1	2	3
4	5	6	7
8	9	10	11

配列次元の範囲を外れた列や行を表示しようとすると、配列制御器は淡色表示され、定義された値がないことを示し、データタイプのデフォルト値が表示されます。データタイプのデフォルト値は配列のデータタイプによって異なります。

一度に複数の行または列を示すには位置決めツールを使用します。

配列関数

以下のタスクように複数の配列を作成するには、配列関数を使用します。

- 配列から個々のデータ要素を抽出する。
- 配列内のデータ要素を挿入、削除、または置き換える。
- 配列を分割する。

配列関数を自動的にサイズ変更する

「インデックス配列（Index Array)」、「部分配列置換（Replace Array Subset)」、「配列要素挿入（Insert Into Array)」、「配列から削除（Delete From Array)」、および「部分配列（Array Subset)」関数は、配線された入力配列の次元に合わせて自動的にサイズ変更されます。たとえば、1 次元配列をこれらの関数の 1 つに配線すると、その関数は 1 つの指標入力を示します。2 次元配列を同じ関数に配線すると、その関数は、行と列に対してそれぞれ 1 つずつ、合計 2 つの指標入力を示します。

位置決めツールを使用して、手動で関数をサイズ変更することにより、これらの関数で複数の要素、つまりサブ配列（行、列、またはページ）にアクセスすることができます。これらの関数の 1 つを拡張する際は、その関数に配線された配列の次元によって決められた増分だけ拡張します。1 次元配列をこれらの関数の 1 つに配線すると、この関数は 1 つの指標入力分拡張します。2 次元配列を同じ関数に配線すると、関数は、行と列にそれぞれ 1 つずつ、合計 2 つの指標入力を拡張します。

配線する指標入力によって、アクセスまたは変更するサブ配列の形状が決まります。たとえば、「インデックス配列（Index Array）」関数への入力が 2 次元配列であり、**行**入力にのみ配線した場合、配列の 1 次元行すべてが抽出されます。**列**入力にのみ配線すると、配列の 1 次元列すべてを取り出します。**行**入力と**列**入力を配線した場合は、配列の 1 つの要素が抽出されます。入力グループはそれぞれ独立しており、配列のすべての次元のどの部分にもアクセスできます。

図 10-9 のようなブロックダイアグラムでは、「インデックス配列（Index Array）」関数を使用して 2 次元配列から行と要素を取り出します。

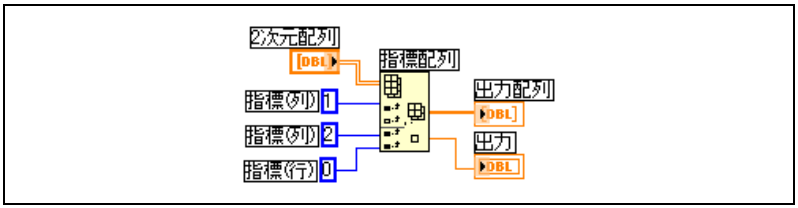


図 10-9　2 次元配列に指標を付ける

配列内の複数の連続した値にアクセスするには、「インデックス配列」関数を拡張します。ただし、増分ごとに値を指標入力に配線しないでください。たとえば、2 次元配列から 1 番目、2 番目、および 3 番目の行を取り出すには、「インデックス配列」関数を 3 回の増加で拡張し、1 次元配列表示器を各**サブ配列**出力に配線します。

ループで配列関数を使用する際にメモリ使用を最小限にする詳細については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Memory and Speed Optimization」のセクションを参照してください。

クラスタ

クラスタは、電話ケーブルのワイヤの束のように、タイプが混在しているデータ要素をグループ化します。電話ケーブルの場合はケーブル内の各ワイヤがクラスタの各要素に相当します。クラスタは、テキストベースのプログラミング言語におけるレコードまたは構造体に似ています。

いくつかのデータ要素をクラスタにまとめることによって、ブロックダイアグラム上のワイヤの混雑を取り除き、サブ VI に必要なコネクタペーン端子の数を減らします。コネクタペーンには、最大 28 個の端子があります。プログラムで使用する制御器および表示器がフロントパネル上に 29 以上ある場合は、制御器および表示器のいくつかを 1 つのクラスタにグループ化して、このクラスタをコネクタペーン上の端子に割り当てます。

クラスタおよび配列要素はどちらも順番が付いていますが、1 つずつ要素に指標を付けるのではなく、すべてのクラスタ要素を一度にバンドル解除する必要があります。また、「名前でバンドル解除（Unbundle By Name）」関数を使用して特定のクラスタ要素にアクセスすることもできます。クラスタは、固定サイズであるという点でも配列と異なります。配列の場合と同様に、クラスタは制御器または表示器のいずれかです。クラスタでは、制御器と表示器を併用することはできません。

ブロックダイアグラム上のほとんどのクラスタは、ピンク色の配線パターンとデータタイプ端子を持っています。数値のクラスタは（ポイントと呼ぶこともあります）、茶色の配線パターンとデータタイプ端子を持っています。茶色の数値クラスタを「和（Add）」や「平方根（Square Root）」のような数値関数に配線し、クラスタのすべての要素に対して同時に同じ演算を実行することができます。

クラスタ要素には、その要素のシェル内での位置とは無関係の論理的順序があります。クラスタに配置する最初のオブジェクトは要素 0、2 番目のオブジェクトが要素 1 などのようになります。要素を削除すると、この順序は自動的に調整されます。クラスタ順序は、その要素がブロックダイアグラムの「バンドル（Bundle）」関数および「バンドル解除（Unbundle）」関数の端子として表示される順序を指定します。クラスタの枠を右クリックして、ショートカットメニューから**クラスタ内制御器の並べ替え**を選択して、クラスタの順序を表示および変更することができます。

クラスタを配線する場合、両方のクラスタの要素数が同じになる必要があります。対応する要素はクラスタ順で決まり、データタイプも適合している必要があります。たとえば、1 つのクラスタの倍精度浮動小数点数値がクラスタ順で他のクラスタの文字列に対応する場合、ブロックダイアグラム上の配線は破線で表示され、VI は実行されません。異なる表現の数値の場合、LabVIEW はこれらの数値を同じ表現に強制的に変換します。数値の変換の詳細については、付録 B「多形性関数」の「数値変換」のセクションを参照してください。

クラスタを作成して操作するタスクでは、クラスタ関数を使用します。たとえば、以下のようなタスクを実行できます。

- クラスタから個々のデータ要素を抽出する。
- クラスタに個々のデータ要素を追加する。
- クラスタを個々のデータ要素に分割する。

11

ローカル変数とグローバル変数

LabVIEW では、ブロックダイアグラム端子を使用して、フロントパネルオブジェクトとの間でデータを読み書きします。しかし、フロントパネルオブジェクトにはブロックダイアグラム端子が 1 つしかないため、アプリケーションは、複数の位置からこの端子内のデータにアクセスする必要があります。

ローカル変数とグローバル変数は、アプリケーション内のワイヤで配線できない場所の間で情報をやり取りします。1 つの VI の複数の位置からフロントパネルオブジェクトにアクセスするにはローカル変数を使用します。複数の VI 間でアクセスし、データをやり取りするにはグローバル変数を使用します。

詳細については

ローカル変数とグローバル変数の使用方法の詳細については、『LabVIEW ヘルプ』を参照してください。

ローカル変数

1 つの VI の複数の位置からフロントパネルオブジェクトにアクセスし、ワイヤで配線できないブロックダイアグラムストラクチャ間でデータをやり取りするには、ローカル変数を使用します。

ローカル変数を使用すると、フロントパネルの制御器や表示器との間でデータの読み書きができます。ローカル変数への書き込みは、他の端子へデータを渡すのに似ています。しかし、ローカル変数を使用すると、データを制御器に書き込んだり、表示器からデータを読み取ることができます。ローカル変数を使用すると、事実上、入力と出力の両方としてフロントパネルオブジェクトにアクセスできます。

たとえば、ログインするのにパスワードなどを入力する必要がある場合、新しいユーザがログインするたびに**ログイン**および**パスワード**プロンプトを入力しなおしてもよいですが、ローカル変数を使用すると、ログインするときは**ログイン**および**パスワード**文字列制御器から読み取り、ログアウトするときはこれらの制御器に空の文字列を書き込みます。

ローカル変数を作成する

既存のフロントパネルオブジェクトまたはブロックダイアグラム端子を右クリックし、ショートカットメニューから**作成→ローカル変数**を選択して、ローカル変数を作成します。そのオブジェクトのローカル変数がブロックダイアグラムに表示されます。

また、**関数**パレットからローカル変数を選択して、ローカル変数をブロックダイアグラムに配置することもできます。左図に示すローカル変数ノードは、制御器または表示器に関連付けられていません。ローカル変数ノードを右クリックして、**選択項目**ショートカットメニューからフロントパネルオブジェクトを選択します。ショートカットメニューには、所有ラベルを持つすべてのフロントパネルオブジェクトのリストが表示されます。

LabVIEW は、所有ラベルを使用してローカル変数をフロントパネルオブジェクトと関連付けるので、フロントパネル制御器および表示器には、わかりやすい所有ラベルを付けてください。所有ラベルとフリーラベルの詳細については、第 4 章「フロントパネルを作成する」の「ラベルを付ける」のセクションを参照してください。

グローバル変数

同時に実行される複数の VI 間でデータにアクセスしてやり取りするにはグローバル変数を使用します。グローバル変数は LabVIEW の標準のオブジェクトです。グローバル変数を作成すると、フロントパネルだけがあってブロックダイアグラムのない特殊なグローバル VI が自動的に作成されます。グローバル VI のフロントパネルに制御器および表示器を追加して、VI に含まれるグローバル変数のデータタイプを定義します。事実上、このフロントパネルはいくつかの VI がデータにアクセスできるコンテナです。

たとえば、同時に 2 つの VI が実行中であると想定します。各 VI は While ループを含んでおり、データ点を波形チャートに書き込みます。 最初の VI には、両方の VI を終了させるためのブール制御器が含まれています。1 つのブール制御器で両方のループを終了させるには、グローバル変数を使用する必要があります。両方のループが同じ VI 内の 1 つのブロックダイアグラム上にあると想定した場合は、ローカル変数を使用してループを終了できます。

グローバル変数を作成する

関数パレットから左図に示すグローバル変数を選択して、ブロックダイアグラムにそのグローバル変数を配置します。グローバル変数ノードをダブルクリックして、グローバル VI のフロントパネルを表示します。標準フロントパネルの場合と同様に、このフロントパネル上に制御器と表示器を配置します。

LabVIEW は、所有ラベルを使用してグローバル変数を識別するので、フロントパネル制御器と表示器には、わかりやすい所有ラベルを付けてください。所有ラベルとフリーラベルの詳細については、第 4 章「フロントパネルを作成する」の「ラベルを付ける」のセクションを参照してください。

それぞれが 1 つのフロントパネルオブジェクトを持つ複数の単一グローバル VI を作成したり、複数のフロントパネルオブジェクトを持つ 1 つのグローバル VI を作成することができます。複数のオブジェクトを持つグローバル VI は、関連する変数をグループ化できるため、より効率的です。VI のブロックダイアグラムは、グローバル VI のフロントパネル上の制御器や表示器と関連付けられたいくつかのグローバル変数ノードを含むことができます。これらのグローバル変数ノードは、グローバル VI のブロックダイアグラム上に置いた最初のグローバル変数ノードのコピーか、または現在の VI 上に置いたグローバル VI のグローバル変数ノードのいずれかです。サブ VI を他の VI 上に置く場合と同じやり方で、グローバル VI を他の VI 上に置きます。ブロックダイアグラム上に新しいグローバル変数ノードを置くたびに、そのグローバル変数ノードおよびそのコピーにのみ関連付けられた新しい VI が作成されます。サブ VI の詳細については、第 7 章「VI およびサブ VI を作成する」の「サブ VI」のセクションを参照してください。

グローバル VI のフロントパネル上にオブジェクトを置いたら、保存して元の VI のブロックダイアグラムに戻ります。次に、アクセスするグローバル VI のオブジェクトを選択する必要があります。グローバル変数ノードを右クリックして、**選択項目**ショートカットメニューからフロントパネルオブジェクトを選択します。ショートカットメニューには、所有ラベルを持つすべてのフロントパネルオブジェクトのリストが表示されます。

読み取り変数と書き込み変数

ローカル変数またはグローバル変数を作成すると、変数とのデータの読み書きが可能になります。デフォルトで、新しい変数はデータを受け取ります。この種類の変数は表示器として機能するため、書き込みローカルまたは書き込みグローバルとなります。新しいデータをローカルまたはグローバル変数に書き込むと、関連付けられているフロントパネルの制御器や表示器は新しいデータに更新されます。

また、変数は、データソースあるいは読み取りローカルまたはグローバルとして動作するようにも構成できます。変数を右クリックして、ショートカットメニューから**読み取りに変更**を選択し、変数を制御器として機能するように構成します。このノードを実行すると、VI は関連付けられているフロントパネルの制御器または表示器のデータを読み取ります。

データを与えるのではなくブロックダイアグラムからデータを受け取るように変数を変更するには、変数を右クリックして、ショートカットメニューから**書き込みに変更**を選択します。

ブロックダイアグラムでは、制御器と表示器の場合と同様に、読み取りローカルまたは読み取りグローバルと書き込みローカルまたは書き込みグローバルを区別することができます。読み取りローカルまたは読み取りグローバルの境界線は、制御器と同じ太い線です。書き込みローカルまたはグローバルの境界線は、表示器と同じ細い線です。

ローカル変数とグローバル変数の使用方法のサンプルについては、`examples¥general¥locals.llb` および `examples¥general¥globals.llb` を参照してください。

ローカル変数とグローバル変数を慎重に使用する

ローカル変数とグローバル変数は高度な LabVIEW の概念です。これらの変数は本来、LabVIEW データフロー実行モデルの一部ではありません。これらの変数を使用すると、ブロックダイアグラムの読み取りが難しくなる可能性があるため、変数は慎重に使用してください。ローカル変数やグローバル変数をコネクタペーンの代わりに使用したり、シーケンスストラクチャの各フレームの値にアクセスするために使用するなど、変数の使用法を誤ると、VI が予期しない動作をする可能性があります。ブロックダイアグラム上に長いワイヤを配線するのを避けるためや、データフローの代わりに使用するなど、ローカル変数やグローバル変数を多用しすぎると、パフォーマンスが低下します。LabVIEW データフロー実行モデルの詳細については、第 5 章の「ブロックダイアグラムを作成する」の「ブロックダイアグラムのデータフロー」のセクションを参照してください。

ローカルおよびグローバル変数を初期化する

VI を実行する前に、ローカル変数とグローバル変数に既知のデータ値が含まれていることを確認します。そうしないと、VI が正しく動作しない原因となるデータが変数に含まれている可能性があります。

変数を初期化しないまま、VI が最初に変数を読み取ると、その変数にはフロントパネルオブジェクトに関連付けられているデフォルト値が含まれています。

競合状態

競合状態は、並列に実行される複数のコードが、ローカル変数またはグローバル変数などの同じ共有リソースの値を変更する場合に発生します。図 11-1 は競合状態の例を示しています。

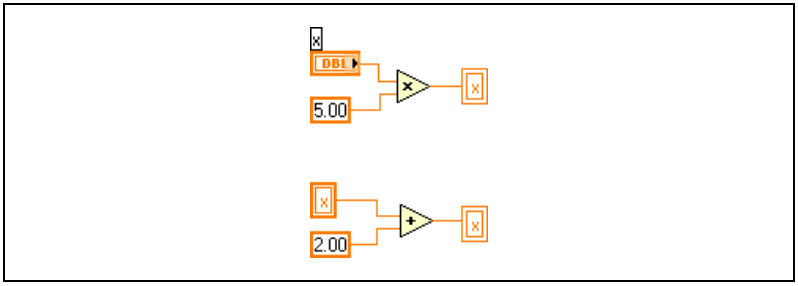


図 11-1　競合状態

この VI の出力は、演算が実行される順序によって異なります。2 つの演算の間にはデータの依存性がないため、どちらを先に実行するかを決める方法がありません。競合状態を避けるには、読み取った変数と同じ変数に書き込まないことです。データ依存の詳細については、第 5 章「ブロックダイアグラムを作成する」の「データ依存と人工データ依存」のセクションを参照してください。

ローカル変数を使用する場合のメモリに関する注意事項

ローカル変数はデータバッファのコピーを作成します。ローカル変数から読み取る場合は、その変数が関連付けられている制御器からのデータ用に新しいバッファを作成します。

ローカル変数を使用してブロックダイアグラム上のある場所から別の場所に大量のデータを転送する場合、通常より多くのメモリを使用するため、ワイヤを使用してデータを転送する場合よりも実行速度が低下します。実行中にデータを格納する必要がある場合には、シフトレジスタを使用するようにしてください。

グローバル変数を使用する場合のメモリに関する注意事項

グローバル変数から読み取る場合、LabVIEW はそのグローバル変数に格納されているデータのコピーを作成します。

大量の配列や文字列を操作する場合は、グローバル変数の操作に必要な時間とメモリはかなりの量になることがあります。1 つの配列要素を変更する場合でも LabVIEW は配列全体を格納および変更するため、配列の場合のグローバル変数の操作は特に効率が悪くなります。アプリケーション内のいくつかの場所からグローバル変数から読み取る場合、複数のメモリバッファを作成しますが、これは効率が悪く、パフォーマンスを低下させます。

12

グラフおよびチャート

グラフ形式でデータを表示するにはグラフやチャートを使用します。

グラフとチャートではデータの表示方法と更新方法が異なります。グラフを持つ VI は通常、データを配列で収集してそのデータをグラフにプロットします。これはデータを格納した後にそのプロットを生成するスプレッドシートに似ています。これに対し、チャートはすでに表示されているデータ点に新しいデータ点を追加して、履歴を作成します。既に集録したデータと関連して現在の読み取り値や測定値をチャートで見ることができます。

詳細については

グラフとチャートの使用方法の詳細については、『LabVIEW ヘルプ』を参照してください。

グラフとチャートのタイプ

グラフとチャートには以下のタイプがあります。

- **波形チャートとグラフ**：一定のレートで集録されるデータを表示します。
- **XY グラフ**：トリガが発生したときに集録されるデータなど、不定レートで集録されるデータを表示します。
- **強度チャートと強度グラフ**：第 3 の次元の値を色で表示して 2 次元プロット上に 3 次元データを表示します。
- **デジタル波形グラフ**：パルスまたはデジタルラインのグループとしてデータを表示します。コンピュータは、デジタルデータを他のコンピュータにパルスで転送します。
- **(Windows) 3 次元グラフ**：フロントパネルにある ActiveX オブジェクトの 3 次元プロット上に 3 次元データを表示します。

グラフとチャートのサンプルについては、`examples¥general¥graphs` を参照してください。

グラフおよびチャートのオプション

グラフとチャートは異なる方法でデータをプロットしますが、それらには共通するいくつかのオプションがあり、ショートカットメニューからアクセスできます。グラフのみまたはチャートのみで使用可能なオプションの詳細については、この章の「グラフをカスタマイズする」のセクションと「チャートをカスタマイズする」のセクションを参照してください。

波形および XY グラフとチャートには、強度、デジタル、および 3 次元のグラフとチャートとは異なるオプションがあります。強度、デジタル、および 3 次元のグラフとチャートのオプションの詳細については、この章の「強度グラフおよびチャート」、「3 次元グラフ」、および「デジタル波形グラフ」のセクションを参照してください。

グラフおよびチャート上の複数の x スケールと y スケール

すべてのグラフは複数の x および y スケールをサポートし、すべてのチャートは複数の y スケールをサポートします。共通の x スケールまたは y スケールを持たない複数のプロットを表示するには、グラフまたはチャート上で複数のスケールを使用します。グラフまたはチャートのスケールを右クリックし、ショートカットメニューから**スケール複製**を選択して、グラフまたはチャートに複数のスケールを追加します。

グラフおよびチャートのエイリアス除去ラインプロット

エイリアス除去ラインを使用すると、グラフおよびチャート内のラインプロットの外観をよりよくすることができます。エイリアス除去ラインの描画を有効にすると、ラインプロットはより滑らかに表示されます。エイリアス除去ラインの描画によって、ライン幅、ラインスタイル、ポイントスタイルなどが変更されることはありません。

メモ　エイリアス除去ラインの描画は、デジタル波形グラフでは使用できません。

エイリアス除去ラインプロットを有効にするには、プロット凡例を右クリックし、ショートカットメニューから**エイリアス除去**を選択します。プロット凡例が表示されていない場合は、グラフまたはチャートを右クリックし、ショートカットメニューから**表示項目→プロット凡例**を選択します。

メモ　エイリアス除去ラインの描画は多くの演算を必要とするため、エイリアス除去ラインプロットを使用すると性能が低下します。

グラフとチャートの外観をカスタマイズする

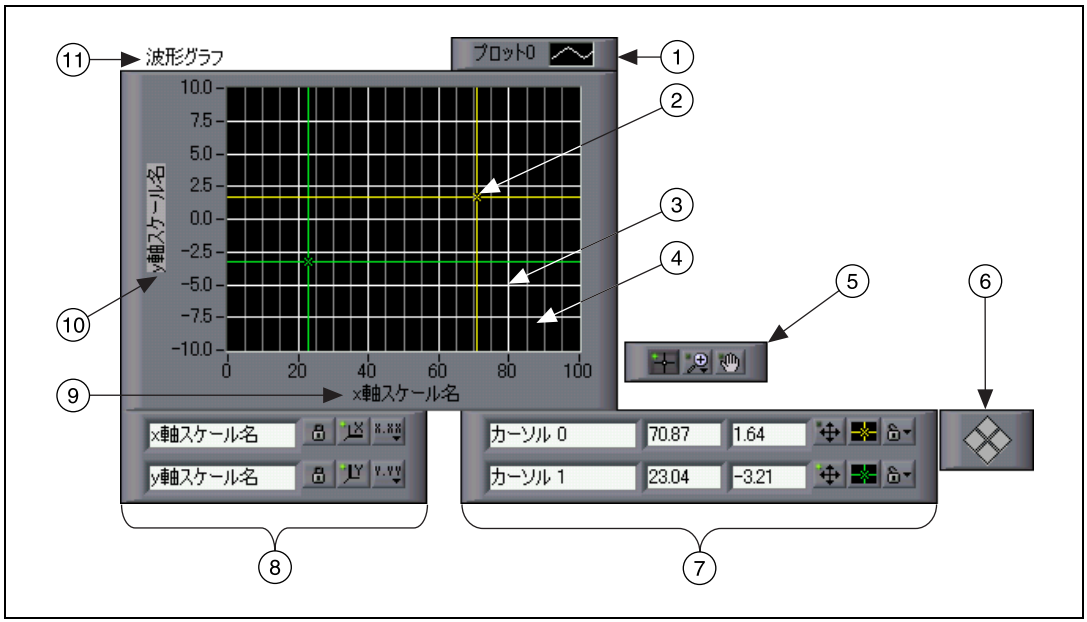
オプションを表示または非表示にすることによって、グラフとチャートの外観をカスタマイズします。グラフまたはチャートを右クリックし、ショートカットメニューから**表示項目**を選択して、以下のオプションを表示または非表示にします。

- **プロット凡例**：プロットの色とスタイルを定義します。凡例をサイズ変更し、複数のプロットを表示します。
- **スケール凡例**：スケールのラベルを定義し、スケールのプロパティを構成します。
- **グラフパレット**：VI の実行中にスケーリングとフォーマット処理を変更します。
- **X スケールと Y スケール**：x スケールと y スケールをフォーマットします。スケールの詳細については、この章の「軸のフォーマット」のセクションを参照してください。
- **カーソルの凡例（グラフのみ）**：定義された点座標にマーカを表示します。グラフ上に複数のカーソルを表示できます。
- **X スクロールバー**：グラフまたはチャート内のデータをスクロールします。スクロールバーを使用して、グラフまたはチャートが現在表示していないデータを表示します。

グラフをカスタマイズする

グラフカーソルの動作、スケーリングオプション、およびグラフ軸を変更できます。図 12-1 はグラフの要素を示しています。

波形グラフおよび強度グラフをカスタマイズして、データ表示条件に合わせたり、詳細情報を表示することができます。チャートで使用可能なオプションには、スクロールバー、凡例、パレット、デジタル表示、および時間に関するスケール表記が含まれます。



1	プロット凡例	4	最小グリッドマーク	7	カーソルの凡例	10	Y- スケール
2	カーソル（グラフのみ）	5	グラフパレット	8	スケール凡例	11	ラベル
3	グリッドマーク	6	カーソルムーバ	9	X- スケール		

図 12-1　グラフの要素

前述の凡例に示したほとんどの項目は、グラフを右クリックし、ショートカットメニューから**表示項目**を選択して、該当する要素を選択することによって追加できます。

グラフカーソル

グラフ上のカーソルによって、プロット上の点の正確な値を読み取ることができます。カーソル値はカーソルの凡例内に表示されます。カーソルを追加するには、グラフを右クリックし、ショートカットメニューから**表示項目→カーソルの凡例**を選択して、カーソルの凡例の行で任意の場所をクリックし、カーソルをアクティブにします。複数のカーソルを追加するには、位置決めツールを使用して、カーソルの凡例を拡張します。

すべてのグラフ上にカーソルとカーソルディスプレイを配置したり、プロット上のカーソルにラベルを付けることができます。カーソルをプロット上にロックしたり、複数のカーソルを同時に移動することもできます。グラフにはカーソルをいくつでも使用できます。

自動スケール

グラフは、水平と垂直のスケールを自動的に調節して、配線したデータを反映させることができます。この動作を自動スケーリングと呼びます。グラフを右クリックし、ショートカットメニューから **X スケール→自動スケール X** または **Y スケール→自動スケール Y** を選択して、自動スケーリングを有効または無効にします。デフォルトでは、グラフに対する自動スケーリングは有効です。ただし、自動スケーリングによって性能が低下することがあります。

水平または垂直のスケールを直接変更するには、操作ツールまたはラベリングツールを使用します。

波形グラフのスケール凡例

スケールにラベルを付け、スケールのプロパティを構成するには、スケール凡例を使用します。

操作ツールを使用して、左図に示す**スケール形式**ボタンをクリックし、形式、精度、およびジャンピングモードを構成します。

各スケールの自動スケーリング、スケールの表示、スケールラベル、およびプロットを切り替えたり、スケールラベル、グリッド、グリッド線、およびグリッドカラーをフォーマットするには、左図に示す**スケールロック**ボタンを使用します。

軸のフォーマット

グラフプロパティおよび**チャートプロパティ**ダイアログボックスの**形式と精度**タブを使用して、グラフまたはチャート上に x 軸と y 軸の表示方法を指定します。

デフォルトでは、x 軸は浮動小数点表記を使用して時間というラベルが付くよう構成され、y 軸は自動形式を使用して振幅というラベルが付くよう構成されています。グラフまたはチャートを右クリックして**プロパティ**を選択し、**グラフプロパティ**ダイアログボックスまたは**チャートプロパティ**ダイアログボックスを表示して、グラフまたはチャートの軸を構成します。

グラフプロパティまたは**チャートプロパティ**ダイアログボックスの**形式と精度**タブを使用して、グラフまたはチャートの軸に数値形式を指定します。軸の名前を変更したり、軸スケールの表示形式を設定するには、**スケール**タブを選択します。デフォルトで、グラフまたはチャートの軸には、自動的に指数表記に切り替わる前に、最大 6 桁が表示されます。

直接入力モード優先チェックボックスをオンにすると、形式文字列を直接入力するためのテキストオプションが表示されます。スケールの表示および軸の数値精度をカスタマイズするには、形式文字列を入力します。形式文字列の詳細については、『LabVIEW ヘルプ』の「形式文字列を使用する」を参照してください。

グラフを動的にフォーマットする

ダイナミックデータタイプ出力を波形グラフに配線して、グラフのプロット凡例と x スケールタイムスタンプを自動的にフォーマットします。たとえば、正弦波を生成して絶対時間を使用するように「信号シミュレーション（Simulate Signal）」Express VI を構成し、波形ブラフに「信号シミュレーション」Express VI の出力を配線すると、グラフのプロット凡例は自動的にプロットラベルを正弦波に更新して、VI を実行する際には x 軸が時間と日付を表示します。

ダイナミックデータタイプを含むプロット凡例ラベルを無視するには、グラフを右クリックしてショートカットメニューから**属性を無視**を選択します。ダイナミックデータタイプを含むタイムスタンプの構成を無視するには、グラフを右クリックしてショートカットメニューから**タイムスタンプを無視**を選択します。

メモ　波形グラフに表示するデータが絶対時間のタイムスタンプを含む場合は、**タイムスタンプを無視**を選択して、このタイムスタンプを無視します。グラフでダイナミックデータタイプを使用する方法および Express VI の詳細については、『LabVIEW 入門』マニュアルを参照してください。また、ダイナミックデータタイプの使用方法については、第 5 章「ブロックダイアグラムを作成する」の「ダイナミックデータタイプ」のセクションも参照してください。

スムーズアップデート

スムーズアップデートをオフにしてオブジェクトを更新すると、オブジェクトの内容が消去されて新しい値が描画され、ちらつきが目立ちます。スムーズアップデートを使用すると、消去と描画によって発生するちらつきを防ぐことができます。オフスクリーンバッファを使用してちらつきの回数を最小にするには、グラフを右クリックし、ショートカットメニューから**上級→スムーズアップデート**を選択します。**スムーズアップデート**を使用すると、ご使用のコンピュータおよびビデオシステムによっては性能が低下することがあります。

チャートをカスタマイズする

既に格納されているデータを上書きして波形全体を表示するグラフとは異なり、チャートは既に格納されているデータの記録を定期的に更新して維持します。

波形チャートおよび強度チャートをカスタマイズして、データ表示条件に合わせたり、詳細な情報を表示することができます。チャートで使用可能なオプションには、スクロールバー、凡例、パレット、デジタル表示、および時間に関するスケール表記などがあります。チャート記録の長さ、更新モード、およびプロット表示の動作を変更できます。

チャート記録の長さ

既にチャートに追加されているデータ点をバッファ、つまりチャート記録に格納します。チャート記録バッファのデフォルトのサイズは 1,024 データ点です。チャートを右クリックし、ショートカットメニューから**チャート記録の長さ**を選択すると、記録バッファを構成できます。チャートのスクロールバーを使用して、それまでに収集したデータを表示できます。チャートを右クリックし、ショートカットメニューから**表示項目→ X スクロールバー**を選択して、スクロールバーを表示します。

チャートの更新モード

チャートは以下の 3 種類のモードを使用して、データをスクロールします。チャートを右クリックして、ショートカットメニューから**上級→更新モード**を選択します。**ストリップチャート**、**スコープチャート**、または**スイープチャート**のいずれかを選択します。デフォルトのモードは、**ストリップチャート**です。

- **ストリップチャート**：左側に古いデータ、右側に新しいデータがあるチャートを左から右へスクロールして、実行中のデータを連続的に表示します。ストリップチャートは紙テープに記録するチャートレコーダに似ています。
- **スコープチャート**：チャートを左から右へ部分的にスクロールして、パルスや波形などのデータ項目を 1 つ表示します。新しい値について、チャートでは最後の値の右側にその新しい値がプロットされます。プロットがプロット領域の右境界線に達すると、LabVIEW ではそのプロットを消去して左境界線から再びプロットを開始します。スコープチャートの再トレースした表示はオシロスコープに似ています。
- **スイープチャート**：スコープチャートと同様に動作しますが、垂直の線で分けられた右側に古いデータを表示し、左側に新しいデータを表示する点が異なります。プロットがプロット領域の右境界線に達しても、スイープチャートではプロットは消去されません。スイープチャートは EKG 表示に似ています。

オーバーレイ対スタックプロット

1 つの垂直スケールを使用して複数のプロットをチャートに表示することをオーバレイプロットと呼び、複数の垂直スケールを使用してチャート上に表示することをスタックプロットと呼びます。図 12-2 は、オーバーレイプロットとスタックプロットの例を示しています。

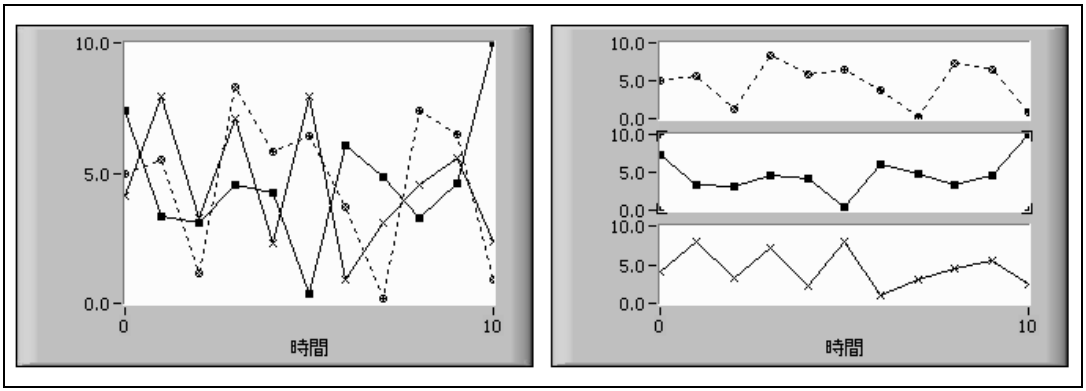


図 12-2　オーバーレイおよびスタックプロットを持つチャート

複数の縦軸スケールとしてチャートプロットを表示するには、チャートを右クリックしてショートカットメニューから**スタックプロット**を選択します。単一の縦軸スケールとしてチャートプロットを表示するには、チャートを右クリックして**プロットを重ねる**を選択します。

さまざまな種類のチャートのサンプルと、その受け入れ可能なデータタイプについては、`examples¥general¥graphs¥charts.llb` の Charts VI を参照してください。

波形グラフと XY グラフ

波形グラフは、均一にサンプリングされた測定値を表示します。XY グラフは、均一または不均一にサンプリングされた一組の点を表示します。図 12-3 は波形グラフと XY グラフの例を示しています。

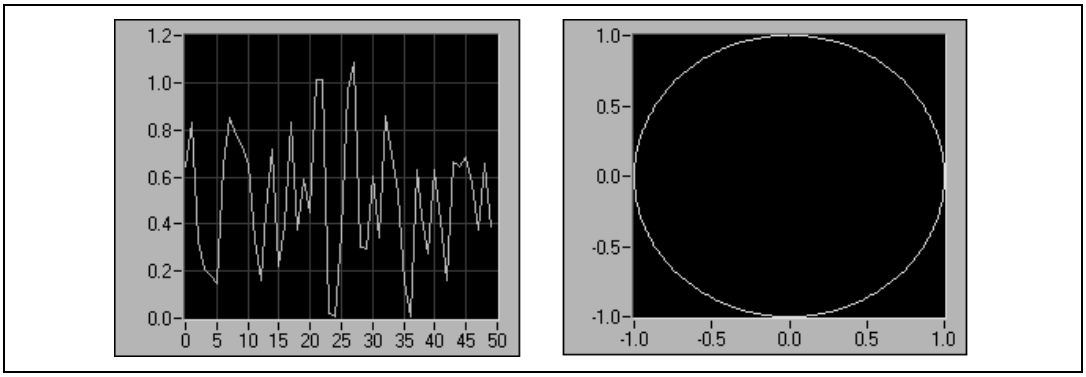


図 12-3　波形グラフと XY グラフ

波形グラフは、時間変化集録波形のように x 軸に沿って均一に分布した点で、$y = f(x)$ のような一価関数のみをプロットします。XY グラフは、円形やタイムベースが変化する波形などの多価関数をプロットする汎用デカルトグラフ作成オブジェクトです。どちらのグラフも、任意の数の点を含むプロットを表示できます。

これらのタイプのグラフはどちらも、複数のデータタイプを受け入れます。これによって、表示の前に行う必要のあるデータの処理を最小限に抑えられます。

シングルプロット波形グラフのデータタイプ

波形グラフは、シングルプロット波形グラフに対して 2 つのデータタイプを受け入れます。

グラフは値の配列を 1 つ受け入れ、そのデータをグラフ上の点として解釈し、さらに $x = 0$ で始まる x 指標を 1 つ増分します。また、グラフは x の初期値、Δx、および y データの配列のクラスタを受け入れます。

シングルプロット波形グラフが受け入れるデータタイプの例については、`examples¥general¥graphs¥gengraph.llb` の Waveform Graph VI を参照してください。

マルチプロット波形グラフ

マルチプロット波形グラフは 2 次元配列の値を受け入れます。ここで、配列の各行はシングルプロットです。グラフはデータをグラフ上の点として解釈し、$x = 0$ で始まる x 指標を 1 回増分します。その配列の各列をプロットとして扱うには、2 次元配列のデータタイプをグラフに配線し、グラフを右クリックして、ショートカットメニューから**配列を転置**を選択します。DAQ デバイスは各チャネルを個別の列として格納した 2 次元配列としてデータを返すため、DAQ デバイスから複数のチャネルをサンプリングするときに特に便利です。このデータタイプを受け入れるグラフのサンプルについては、`examples¥general¥graphs¥gengraph.llb` の Waveform Graph VI の (Y) マルチプロット 1 グラフを参照してください。

マルチプロットの波形グラフは、x 値のクラスタ、 Δx 値、および y データの 2 次元配列も受け入れます。グラフは y データをグラフ上の点として解釈し、$x = 0$ で始まる ξ 指標を Δx ずつ増分します。このデータタイプは、すべて均一の規則的なレートでサンプリングした複数の信号を表示するのに便利です。このデータタイプを受け入れるグラフのサンプルについては、`examples¥general¥graphs¥gengraph.llb` の Waveform Graph VI の (Xo, dX, Y) マルチプロット 3 グラフを参照してください。

マルチプロット波形グラフは、クラスタを含む配列のプロット配列を受け入れます。各クラスタには、y データを含む点配列が含まれています。内部配列はプロット内の点を記述し、外部配列は各プロットに対して 1 つのクラスタがあります。図 12-4 は、**y** クラスタのこの配列を示しています。

図 12-4　クラスタ y の配列

各プロット内の要素数が異なる場合は、2 次元配列ではなくマルチプロット波形グラフを使用します。たとえば、チャネル別に異なる時間を使用して、複数のチャネルからデータをサンプリングする場合は、2 次元配列ではなくこのデータストラクチャを使用します。これは、2 次元配列では各行の要素数が同じでなくてはならないためです。クラスタの配列の内部配列の要素の数は同じでなくてもかまいません。このデータタイプを受け入れるグラフの例については、`examples¥general¥graphs¥gengraph.llb` の Waveform Graph VI の (Y) マルチプロット 2 グラフを参照してください。

マルチプロットの波形グラフは、*x* の初期値のクラスタ、Δ*x* 値、およびクラスタを含む配列を受け入れます。各クラスタには、*y* データを含む点配列が含まれています。配列をクラスタとしてバンドルするには「バンドル（Bundle）」関数を使用し、そのクラスタを配列にするには「配列作成（Build Array）」関数を使用します。指定した入力を含むクラスタの配列を作成する「クラスタ配列作成（Build Cluster Array）」も使用できます。このデータタイプを受け入れるグラフのサンプルについては、`examples¥general¥graphs¥gengraph.llb` の Waveform Graph VI の (Xo, dX, Y) マルチプロット 2 グラフを参照してください。

マルチプロットの波形グラフは、*x* 値のクラスタの配列、Δ*x* 値、および *y* データの配列を受け入れます。これは、各プロットの *x* 軸の固有の開始点および増分を指定できるため、マルチプロット波形グラフのデータタイプの中で最も一般的です。このデータタイプを受け入れるグラフのサンプルについては、`examples¥general¥graphs¥gengraph.llb` の Waveform Graph VI の (Xo, dX, Y) マルチプロット 1 グラフを参照してください。

シングルプロット XY グラフのデータタイプ

シングルプロット XY グラフは、*x* 配列と *y* 配列を含むクラスタを受け入れます。XY グラフは点の配列も受け入れます。ここで、点は *x* 値と *y* 値を含むクラスタです。

シングルプロット XY グラフのデータタイプのサンプルについては、`examples¥general¥graphs¥gengraph.llb` の XY Graph VI を参照してください。

マルチプロット XY グラフのデータタイプ

マルチプロット XY グラフは、プロットの配列も受け入れます。ここで、プロットは x 配列と y 配列を含むクラスタです。マルチプロット XY グラフは、プロットのクラスタの配列も受け入れます。ここで、このプロットは点の配列です。点は *x* 値と *y* 値を含むクラスタです。

マルチプロット XY グラフのデータタイプのサンプルについては、`examples¥general¥graphs¥gengraph.llb` の XY Graph VI を参照してください。

波形チャート

波形チャートは、1 つまたは複数のプロットを表示する特殊なタイプの数値表示器です。波形チャートのサンプルについては、`examples¥general¥graphs¥charts.llb` を参照してください。

一度に 1 つまたは複数の値をチャートに渡す場合、LabVIEW はデータをチャート上の点として解釈し、*x* = 0 で始まる *x* 指標を 1 回増分します。チャートはこの入力をシングルプロットの新規データとして扱います。チャートを波形データタイプに渡すと、*x* 指標は特定の時間形式に適合します。

チャートにデータを送る頻度によって、チャートの再描画回数が決まります。

複数のプロットのデータを波形チャートに渡すには、データをスカラ数値のクラスタ内にバンドルできます。ここで、各数値は各プロットの 1 つの点を表します。

一度の更新でプロットの複数の点を渡す場合は、数値のクラスタの配列をチャートに配線します。各数値は、各プロットの 1 つの *y* 値の点を表します。

表示するプロットの数を実行時まで指定できない場合、または一度の更新で複数のプロットの複数の点を渡す場合は、チャートに数値または波形の 2 次元配列を配線します。波形グラフと同様に、波形チャートはデフォルトで行を各プロットの新規データとして扱うようになっています。配列内の列を各プロットの新規データとして扱うには、2 次元配列データタイプをチャートに配線し、チャートを右クリックして、ショートカットメニューから**配列を転置**を選択します。

強度グラフおよびチャート

デカルト平面上に色のブロックを配置することによって、2 次元プロット上に 3 次元データを表示するには、強度グラフおよびチャートを使用します。たとえば、強度グラフとチャートを使用して、温度パターンや大きさが標高を表す地形などのパターンのあるデータを表示することができます。強度グラフおよびチャートは、数値の 2 次元配列を受け入れます。配列内の各数値は特定の色を表します。2 次元配列内の要素の指標が色のプロット位置を設定します。図 12-5 は強度チャートの操作の概念を示しています。

入力配列

列 = y

行 = x	0	1	2
0	50	50	13
1	45	61	10
2	6	13	5

カラーマップの定義

配列要素 = z	色
5	青
6	紫
10	薄い赤
13	濃い赤
45	橙
50	黄色
61	緑

結果プロット

	0	1	2
3	濃い赤	薄い赤	青
2	黄色	緑	濃い赤
1	黄色	橙	紫

図 12-5　強度チャートのカラーマップ

データの行は、グラフまたはチャート上の新しい列として表示されます。行を画面上の行として表示する場合は、2 次元配列データタイプをグラフまたはチャートに配線し、グラフまたはチャートを右クリックして、ショートカットメニューから**配列を転置**を選択します。

配列の指標は、カラーブロックの左下頂点に相当します。カラーブロックには、配列の指標で定義された 2 点間の領域である単位領域があります。強度グラフまたはチャートは、最大 256 の個別の色を表示できます。

強度チャート上にデータのブロックをプロットすると、デカルト平面の原点は最後のデータブロックの右側に移動します。チャートが新規データを処理すると、その新規データは古いデータの右側に表示されます。チャートの表示が満杯になると、古いデータはチャートの左側にスクロールされて表示されなくなります。この動作はストリップチャートの動作に似ています。これらのチャートの詳細については、この章の「チャートの更新モード」のセクションを参照してください。

強度チャートおよびグラフのサンプルについては、`examples¥general¥graphs¥charts.llb` を参照してください。

カラーマッピング

カラーランプ数値制御器にカラーを定義するのと同じ方法で、強度グラフおよびチャートに、対話式でカラーマッピングを設定できます。カラーランプの詳細については、第 4 章「フロントパネルを作成する」の「カラーランプ」のセクションを参照してください。

強度グラフとチャートにプログラムでカラーマッピングを設定するには、「プロパティノード（Property Node）」を 2 つの方法で使用します。通常は、「プロパティノード」に数値対色マッピングを指定します。この方法では、「Z スケール：マーカ値」プロパティを指定します。このプロパティは配列のクラスタで構成され、各クラスタには数値のリミット値とその値を表示するための対応するカラーが含まれています。この方法でカラーマッピングを指定する場合、上限を超えた色を指定するには、「Z スケール：ハイカラー」プロパティを使用し、下限を超えた色を指定するには、「Z スケール：ローカラー」プロパティ を使用します。強度グラフおよびチャートは全部で 254 色に制限され、下限と上限を超えた色を合わせて合計 256 色になります。254 色を超える数の色を指定すると、強度グラフまたはチャートは、指定された色を補間して 254 色のカラーテーブルを作成します。

強度グラフ上にビットマップを表示する場合は、「カラーテーブル」プロパティを使用してカラーテーブルを指定します。この方法では、最大 256 色の配列を指定できます。チャートに渡されるデータは、強度チャートのカラースケールに基づいて、このカラーテーブルの指標に割り当てられます。カラースケールの範囲が 0 ～ 100 の場合は、データ内の値 0 が指標 1 に値 100 が指標 254 に割り当てられて、その間の値は 1 と 254 の間に補間されます。0 より小さい任意の値は、すべて下限を超えた色 (指標 0) に割り当てられ、100 を超える任意の値はすべて上限を超えた色 (指標 255) に割り当てられます。

メモ　強度グラフまたはチャートで表示する色は、ご使用のビデオカードが表示できる数の厳密な色に制限されます。また、ご使用のディスプレイに割り当てられた色の数による制限もあります。

強度チャートオプション

強度チャートの任意の部分の多くは波形チャートと共通しています。チャートを右クリックしてショートカットメニューから**表示項目**を選択すると、それらの部分を表示または非表示にできます。さらに、強度チャートは色を第 3 次元として扱うため、カラーランプ制御器に似たスケールが色に対する値の範囲とマッピングを定義します。

波形チャートと同様に、強度チャートは以前の更新の記録データ、つまりバッファを保持しています。チャートを右クリックし、**ショートカットメニュー**から**チャート記録の長さ**を選択すると、このバッファを構成できます。強度チャートにおけるデフォルトのサイズは 128 データ点です。強度チャートの表示には、多くのメモリを使用します。たとえば、512 点の記録と 128 の *y* 値で単精度チャートを表示するには、512 ＊ 128 ＊ 4 バイト (単精度数のサイズ)、つまり 256 KB のメモリが必要です。

強度グラフオプション

強度グラフは強度チャートと同様に動作しますが、以前のデータを保持せず更新モードがないという点のみ異なります。新規データが強度グラフに渡されるたびに、古いデータは新規データに置き換えられます。

強度グラフには他のグラフと同様のカーソルがあります。各カーソルは、グラフ上の指定された点を示す *x* 値、*y* 値、および *z* 値を表示します。

デジタル波形グラフ

タイミングダイアグラムまたはロジックアナライザを操作する場合は特に、デジタル波形グラフを使用してデジタルデータを表示します。デジタルデータ集録の詳細については、『LabVIEW Measurements Manual』を参照してください。

デジタル波形グラフは、入力として、デジタル波形データタイプ、デジタルデータタイプ、およびこれらのデータタイプの配列を受け入れます。デフォルトでは、デジタル波形グラフがデジタルバスを縮小すると、グラフはシングルプロットでデジタルデータをプロットします。デジタルデータの配列を接続する場合は、デジタル波形グラフが、その配列の順番に従って異なるプロットとして配列の各要素をプロットします。

図 12-6 のデジタル波形グラフはシングルプロットでデジタルデータをプロットします。VI では、**数値**配列にある数値をデジタルデータに変換して、**バイナリ表記**のデジタルデータ表示器にある数値のバイナリ表記を表示します。デジタルグラフでは、数値 0 は最上部ラインなしで表示され、すべてのビット値が 0 であることを示します。数値 255 は最下部ラインなしで表示され、すべてのビット値が 1 であることを示します。

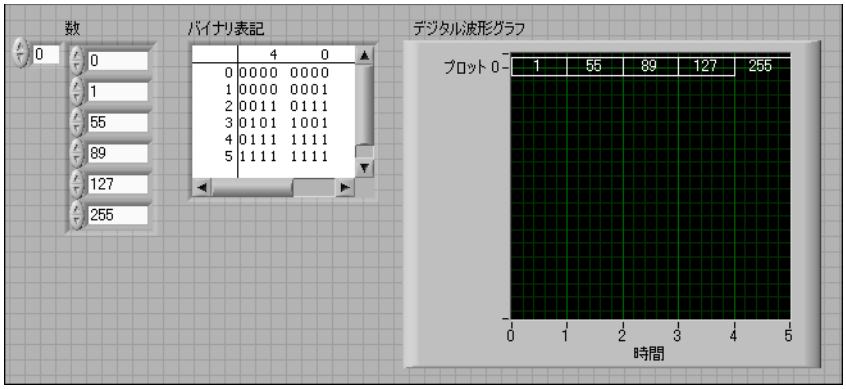


図 12-6　シングルプロットでデジタルデータをプロットする

デジタル波形グラフのプロット構成の詳細については、『LabVIEW ヘルプ』を参照してください。

デジタルデータの各サンプルをプロットするには、y 軸を右クリックしてショートカットメニューから**デジタルバスを展開**を選択します。各プロットはそれぞれ異なるサンプルを表示します。

図 12-7 のデジタル波形グラフは**数値**のデジタル配列制御器にある 6 つの数値を示します。

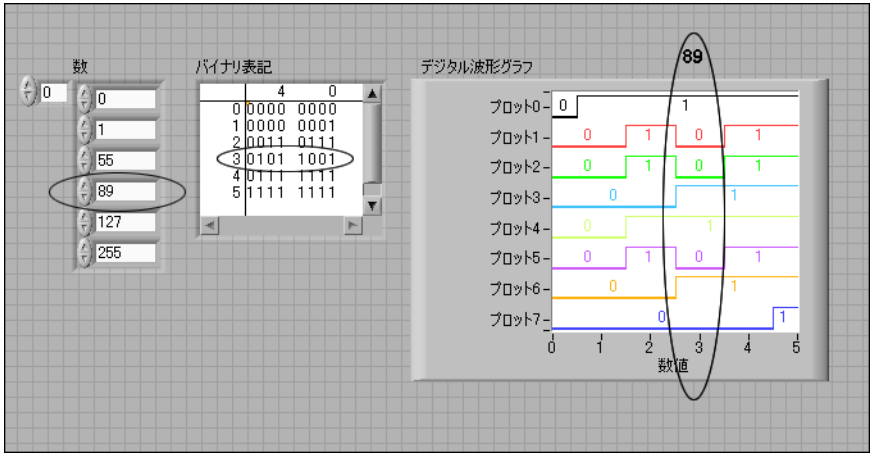


図 12-7　デジタル形式での整数のグラフ化

バイナリ表記のデジタル制御器は数値のバイナリ表記を表します。表の各列はビットを表します。たとえば、数値 89 には 7 ビットのメモリが必要です（列 7 の 0 は未使用のビットを示します）。デジタル波形グラフのポイント 3 は必要な 7 ビットをプロットして、数値 89 と 0 の値を表示し、プロット 7 に未使用の 8 番目のビットを表示します。

図 12-8 の VI では、数値配列をデジタルデータに変換し、「波形作成（Build Waveform）」関数を使用して開始時間（Δt）およびデジタルデータ制御器に入力した数値を組み合わせて、デジタルデータを表示します。

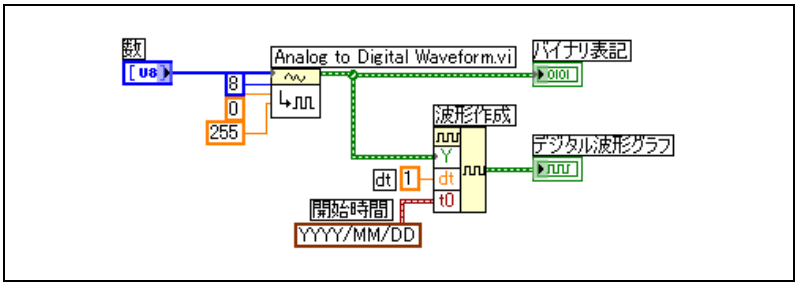


図 12-8　数値配列をデジタルデータへ変換する

デジタルデータ制御器、デジタル波形データタイプ、およびデータをデジタルデータに変換する方法については、第 4 章「フロントパネルを作成する」の「デジタル波形制御器」および「デジタルデータ制御器」のセクションを参照してください。

データをマスクする

図 12-7 の VI は、各プロットがデータ内の 1 つのビットを表現するグラフを生成します。グラフ上にデータを表示する前に、データ内のビットを選択したり、並べ替えたり、結合することもできます。ビットの選択、並べ替え、および結合を、データのマスクと呼びます。

マスクを使用すると、2 つ以上の異なるビットのプロットを結合し、1 つのプロットに表示することができます。8 ビット整数の配列の場合は、1 つのプロットに 8 ビットまで表示できます。16 ビット整数の配列の場合は、1 つのプロットに 16 ビットまで表示でき、以下同様に続きます。1 つのプロットに同じビットを 2 回以上プロットすることもできます。

3 次元グラフ

メモ　3D グラフ制御器は、Windows 対応の LabVIEW 開発システムおよびプロフェッショナル開発システムでのみ使用できます。

表面上の温度分布、時間 / 周波数共同領域解析、航空機の動きといった実世界の多くのデータセットでは、3 次元データを視覚化する必要があります。3 次元グラフを使用すると、3 次元データを視覚化でき、3 次元グラフのプロパティを変更することによってデータの表示方法を変更できます。

使用可能な 3 次元グラフは以下のとおりです。

- **3D 曲面**：3 次元空間に曲面を描画します。この制御器をフロントパネル上にドロップすると、曲面を表すデータを受信するサブ VI に配線されます。
- **3D パラメトリック曲面**：3 次元空間に複雑な曲面を描画します。この制御器をフロントパネル上にドロップすると、曲面を表すデータを受信するサブ VI に配線されます。
- **3D カーブ**：3 次元空間に線を描画します。この制御器をフロントパネル上にドロップすると、線を表すデータを受信するサブ VI に配線されます。

曲線や曲面をプロットするには、3 次元グラフ VI とともに 3 次元グラフを使用します。曲線はグラフ上の個々の点から構成され、各点は、*x*、*y*、および *z* 座標で示されます。この VI は、これらの点を線で結びます。曲線は、飛行機の飛行経路など、動いている物体の経路を視覚化するのに最適です。

3 次元グラフでは、ActiveX テクノロジと 3D 表現を処理する VI を使用します。3 次元グラフプロパティ VI のプロパティ（基本、軸、グリッド、およびプロジェクションプロパティなど）を設定すると、実行時の動作を変更できます。

曲面プロットは *x*、*y*、および *z* のデータを使用してグラフ上に点をプロットします。その後、これらの点を結んでデータの 3 次元曲面画像を生成します。たとえば、曲面プロットを使用して地形図を作成できます。

3 次元グラフを選択すると、LabVIEW は 3D グラフ制御器が含まれたフロントパネル上に ActiveX コンテナを配置します。LabVIEW はまた、3D グラフ制御器のリファレンスをブロックダイアグラム上に配置します。LabVIEW は、3 つの 3 次元グラフ VI のいずれかにこのリファレンスを配線します。

波形データタイプ

波形データタイプには、波形のデータ、開始時刻、および　Δ*t* があります。「波形作成（Build Waveform）」関数を使用して波形を作成します。波形の集録または解析に使用する多くの VI および関数は、デフォルトで波形データタイプを受け取り、波形データタイプを返します。波形グラフまたは波形チャートに波形データタイプを配線すると、波形のデータ、開始時刻、および　′x に基づいて、グラフまたはチャート上に自動的に波形がプロットされます。波形グラフまたは波形チャートに波形データタイプの配列を配線すると、グラフまたはチャート上にすべての波形が自動的にプロットされます。波形データタイプの使用方法については、『LabVIEW Measurements Manual』の Chapter 5「Creating a Typical Measurement Application」を参照してください。

デジタル波形データタイプ

デジタル波形データタイプには、デジタル波形の開始時刻、Δx、データ、および属性があります。「波形作成（Build Waveform）」関数を使用してデジタル波形を作成します。デジタル波形データタイプをデジタル波形グラフに接続すると、タイミング情報とデジタル波形のデータに基づいてグラフは自動的に波形をプロットします。デジタル波形のサンプルや信号を表示するには、デジタル波形データタイプにデジタルデータ表示器を配線します。デジタル波形データタイプの使用方法については、『LabVIEW Measurements Manual』の Chapter 5「Typical Measurement Applications」を参照してください。

13

グラフィック & サウンド VI

VI のグラフィックやサウンドを表示したり変更したりするには、グラフィック＆サウンド VI を使用します。

詳細については

VI でのグラフィックとサウンドの使用方法の詳細については、『LabVIEW ヘルプ』を参照してください。

ピクチャ表示器を使用する

ピクチャ表示器は、線、円、テキスト、その他の画像形状を含むピクチャを表示する汎用表示器です。ピクチャ表示器はピクセルレベルで制御できるので、ほとんどすべての画像オブジェクトを作成できます。グラフィックアプリケーションの代わりにピクチャ表示器やグラフィック VI を使用して、LabVIEW でグラフィックの作成、変更、および表示を行います。

ピクチャ表示器は、ピクセルベースの座標系を持ち、原点 (0, 0) は制御器の左上隅に位置するピクセルです。座標の水平（*x*）成分は右へ向かって増加し、座標の垂直（*y*）成分は下へ向かって増加します。

ピクチャが大きすぎてピクチャ表示器がそのピクチャを表示できない場合、表示器の表示領域内に収まるピクセルだけを表示します。ピクチャ全体を表示するには、位置決めツールを使用して表示器をサイズ変更し、VI を再実行します。また、ピクチャ表示器の垂直および水平スクロールバーを表示できるので、表示器の表示領域外にあるピクセルを表示することができます。表示器を右クリックしてショートカットメニューから**項目を表示→スクロールバー**を選択して、表示器のスクロールバーを表示します。

ピクチャクラスにある VI サーバプロパティを使用して、ピクチャ表示器のサイズ変更や表示器内のピクチャ領域のサイズ変更など、プログラム的にピクチャ表示器を変更できます。VI サーバのプロパティの使用方法については、第 17 章「VI をプログラム的に制御する」を参照してください。

フロントパネル上にピクチャ表示器を配置すると、表示器は空白の四角形領域として表示され、左図のような対応する端子がブロックダイアグラム上に表示されます。

ピクチャ表示器に画像を表示するには、プログラムで表示器へ画像を書き込む必要があります。画像をクリップボードへコピーしたり、ピクチャ表示器へ貼り付けすることはできません。ピクチャ関数 VI を使用して、一連の描画手順を指定できます。各 VI は、描画手順を説明した入力値を受け取ります。VI は、これらの入力値に基づいて、表示用にピクチャ表示器に渡すこれらの手順を簡潔に説明したものを作成します。ピクチャ関数 VI を使用してピクチャ表示器で画像を表示する方法については、この章の「ピクチャ関数 VI」のセクションを参照してください。

ピクチャプロット VI

ピクチャ表示器を使用して一般的なタイプのグラフを作成するには、ピクチャプロット VI を使用します。このグラフには、極座標、波形グラフ表示、XY グラフ、スミスプロット、レーダープロット、およびグラフスケールなどがあります。

ピクチャプロット VI は低レベル描画関数を使用して、データのグラフィカルな表示を作成し、描画コードをカスタマイズして機能を追加します。このグラフィカルな表示器は標準の LabVIEW 制御器のような対話式ではありませんが、それらを使用すると、現在の標準の制御器ではできない方法で情報を表示できます。たとえば、標準の波形グラフとは異なる機能を持つプロットを作成するには、「波形プロット（Plot Waveform）」VI を使用します。

極プロット VI をサブ VI として使用する

極グラフの隣接する特定の四分円またはグラフ全体を一度に描くには、「極プロット（Polar Plot）」VI を使用します。LabVIEW の標準波形グラフと同様に、そのコンポーネントの色を指定し、グリッドを表示して、スケールの範囲と形式を指定することができます。

「極プロット」VI は、1 つの VI にさまざまな機能を与えます。したがって、この VI には入力端子のための複雑なクラスタが含まれています。デフォルト値とカスタム制御器を使用して、VI を簡単に作成することができます。デフォルトのクラスタ入力を作成する代わりに、`examples¥picture¥demos.llb` 内の Polar Plot Demo VI からカスタム制御器をコピーし、それをフロントパネル上に配置します。

プロット波形 VI および XY プロット VI をサブ VI として使用する

点、接続された点、バーなど、さまざまなスタイルで波形を描くには、標準波形グラフの動作をエミュレートする「波形プロット（Plot Waveform）」VI を使用します。標準の波形グラフと同様に、そのコンポーネントの色を指定し、グリッドを表示して、スケールの範囲と形式を指定できます。

「波形プロット（Plot Waveform）」VI により、1 つの VI にさまざまな機能を追加することができます。したがって、その VI には入力端子のための複雑なクラスタが含まれています。デフォルト値とカスタム制御器を使用して、VI を簡単に作成することができます。デフォルトのクラスタ入力を作成する代わりに、`examples¥picture¥demos.llb` 内の「波形（Waveform）」および XY プロット（XY Plots）」VI からカスタム制御器をコピーし、それをフロントパネル上に配置します。

「XY プロット」VI と「マルチ -XY プロット（Plot Multi-XY）」VI は、「波形プロット」VI と似ています。XY をプロットする VI は、3 つの追加プロットスタイル、すなわち 2 つの分散プロットスタイルと、個別の *x* 位置で線を描画してその *x* 値に対する *y* の最小値と最大値にマークを付ける 1 つのプロットスタイルを持つため、プロットの装飾的な外観を指定するには別の制御器を使用します。

スミスプロット VI をサブ VI として使用する

電気通信業界などの伝送ラインの特性を調べるには、「スミスプロット（Smith Plot）」VI を使用します。伝送ラインはエネルギーや信号を送る媒体です。伝送ラインはワイヤを指す場合や、信号を送る環境を指す場合があります。伝送ラインは送信中の信号に影響を与えます。この影響は伝送ラインのインピーダンスと呼ばれ、交流信号を減衰させたり、位相を変化させます。

伝送ラインのインピーダンスは、ラインの抵抗とリアクタンスの測定値です。通常、インピーダンス z は $z = r + jx$ の形式の複素数として表されます。ここで、抵抗（r）およびリアクタンス（x）はその要素です。

伝送ラインのインピーダンスを表示するには、「スミスプロット」VI を使用します。プロットは一定の抵抗とリアクタンスを表す円で構成されます。

与えられたインピーダンス $r + jx$ は、r の円と x の円の交点の位置を定めることによってプロットできます。インピーダンスのプロット後、視覚効果として「スミスプロット」VI を使用し、インピーダンスを整合させて伝送ラインの反射係数を計算します。

「スミスプロット」VI により、1 つの VI にさまざまな機能を追加することができます。したがって、これらの VI には入力端子のための複雑なクラスタが含まれています。デフォルト値とカスタム制御器を使用して、VI を簡素化することができます。デフォルトのクラスタ入力を作成する代わりに、`examples¥picture¥demos.llb` 内のサンプルの Smith Plot VI からカスタム制御器をコピーしてフロントパネル上に配置します。

負荷インピーダンスをグラフ表示している場合、インピーダンスは $r + jx$ の形式の複素数として表現できます。

スミスプロットの詳細データの損失を避けるには、「正規化スミスプロット (Normalize Smith Plot)」VI を使用してデータを正規化します。「正規化スミスプロット」VI で正規化したデータは「スミスプロット」VI に直接渡すことができます。通常、スミスプロットデータは、システムの特性インピーダンス (Z_0) に従ってサイズを調整します。

ピクチャ関数 VI

ピクチャ表示器内に形状を描画してテキストを入力するには、ピクチャ関数 VI を使用します。点、線、形状、およびピックスマップを描画できます。平坦化されていないデータのピックスマップは 2 次元配列の色で、それぞれの値は、カラー深度により色または RGB カラー値の配列の指標に相当します。

ピクチャ関数パレットの 1 段目には、点および線の描画に使用する VI があります。点は、ピクセルの xy 座標を表す 2 つの 16 ビット符号付き整数のクラスタです。

ピクチャ関数 VI を使用すると、ピクチャでは画像ペンの位置が記憶されます。ほとんどのピクチャ関数 VI では、原点 (0, 0) を基準にした絶対座標で指定する必要があります。「ライン描画（Draw Line）」VI と「ペン移動（Move Pen）」VI を使用すると、絶対座標または相対座標を指定できます。相対座標は現在のペンの位置が基準となります。「ペン移動」VI を使用すると、描画せずにペンの位置を変更できます。ペンの位置を変更できるのは、「ポイント描画（Draw Point）」VI、「ペン移動（Move Pen）」VI、「ライン描画（Draw Line）」VI、および「複数ライン描画（Draw Multiple Lines）」VI のみです。

ピクチャ関数パレットの 2 段目には、形状の描画に使用する VI があります。これらの VI はそれぞれ、ピクチャの四角形の領域内に形状を描画します。四角形は、左右上下のピクセルを表す 4 つの値のクラスタで指定します。

ピクチャ関数パレットの 3 段目には、ピクチャ内でのテキスト描画に使用する VI があります。「テキスト境界取得（Get Text Rect）」VI はテキストを描画しません。代わりに、文字列の境界枠の四角形のサイズの計算に使用します。

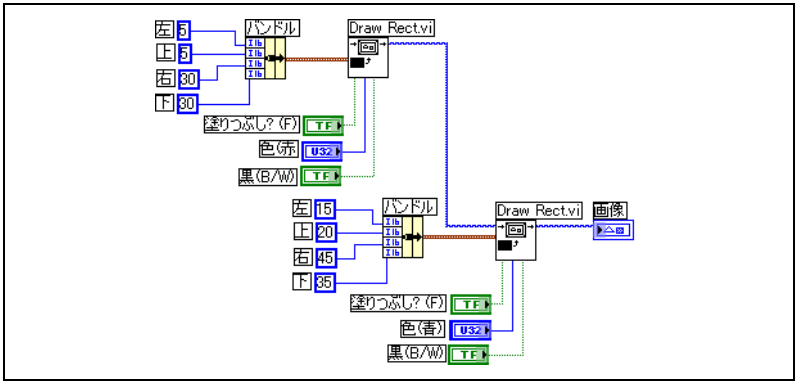
ピクチャ関数パレットの 4 段目には、ピクチャ内での平坦化および非平坦化ピックスマップの描画、マスクの画像への適用、ソース画像のサブセットの取得、およびピクチャデータタイプの平坦化画像データクラスタへの変換に使用する VI があります。

ピクチャ関数パレットの最後の段には、空ピクチャ定数があり、空のピクチャを使用開始するときや変更するときに使用します。また、パレットの最後の段には、赤、緑、青の値を対応する RGB カラーへ変換したり、各カラーをそれぞれ赤、緑、青のコンポーネントへ変換する場合に使用する VI もあります。

ピクチャ関数 VI で作成したピクチャは、1 つのピクチャ表示器またはピクチャ関数 VI の**ピクチャ**入力にのみ配線できます。LabVIEW は、開いているフロントパネル上でピクチャ表示器を更新するときにピクチャを描画します。

各ピクチャ関数 VI はその描画手順を**ピクチャ**入力へ接続する描画手順に連結して、**新規ピクチャ**出力で連結した描画手順を返します。

以下のブロックダイアグラムは、「四角形描画（Draw Rect）」VI を使用して 2 つの重なり合う四角形を描画します。



ピクチャ関数 VI を使用した色の作成と変更

多くのピクチャ関数 VI には、形状およびテキストの色を変更する**色**入力端子があります。色を指定する最も簡単な方法は、カラーボックス定数を使用し、その定数をクリックして色を選択することです。

カラーボックス定数ではなく、計算結果によって色を作成するには、カラーボックスがどのように数値を使用して色を指定しているかを理解する必要があります。

32 ビット符号付き整数は色を表し、下位の 3 バイトが赤、緑、および青の要素を表します。青の範囲については、32 ビット整数の配列を作成します。ここで、各要素の青の値は変更されて、赤および緑の値よりも大きくなります。灰色の範囲を作成するには、赤、緑、および青の各要素の値が等しい 32 ビット整数の配列を作成します。

グラフィック形式 VI

ビットマップ（.bmp）、ポータブルネットワークグラフィック（.png）、Joint Photographic Experts Group（.jpg）ファイルなどの複数の標準グラフィックファイルフォーマットにデータを読み書きするには、グラフィック形式 VI を使用します。この VI を使用すると、以下のタスクを実行できます。

- グラフィックファイルからデータを読み取りピクチャ表示器に表示する。
- データを読み取り画像を処理する。
- 画像を書き込んで他のアプリケーションで表示する。

ビットマップデータは 2 次元配列で、カラー深度によって各点が異なります。たとえば、モノクロつまり 1 ビット画像の場合、各点はブール値です。4 ビットおよび 8 ビット画像の場合、各点はカラーテーブル内の指標です。24 ビット True-Color 画像の場合、各点は赤、緑、および青（RGB）の値を混合したものです。

グラフィックファイルの読み書きを行う VI は、単純で平坦化された形式のデータを利用します。これは、1 次元配列に格納されたデータを使い、画像ファイルをディスクに書き込む方法に似ています。これらの画像ファイルはピックスマップで、概念はビットマップと同じです。「平坦化ピックスマップ描画（Draw Flattened Pixmap）」VI を使用して、この平坦化されたデータを直接表示します。

2 次元配列としてデータを処理するには、「非平坦化ピックスマップ（Unflatten Pixmap）」VI および「平坦化ピックスマップ（Flatten Pixmap）」VI を使用して、そのデータを適切な形式に変換します。

サウンド VI

サウンドファイルおよび関数をユーザの VI に総合するには、サウンド VI を使用します。この VI を使用すると、以下のタスクを実行できます。

- ユーザが特定の操作を実行したときに、録音された警告などのサウンドファイルを実行する VI を作成する。
- VI が実行を開始または完了したとき、または VI 内のある点に達したときにサウンドファイルを実行する VI を作成する。
- サウンドデータを集録するためのサウンド入力デバイスを構成する。サウンド入力 VI を使用して、サウンドデータを集録します。デバイスから入力される任意のサウンド情報を読み取ることもできます。
- 他のサウンド VI からサウンドデータを受け入れるためのサウンド出力デバイスを構成する。デバイスを通るサウンド量を制御したり、サウンドを再生または一時中止したり、システムからサウンドを消すことができます。

14

ファイル I/O

ファイル I/O 操作により、ファイルとのデータ受け渡しができます。以下の操作を含むさまざまなファイル I/O 処理には、ファイル I/O および関数を使用します。

- データファイルの開閉。
- ファイルに対するデータの読み書き。
- スプレッドシート形式のファイルに対する読み書き。
- ファイルおよびディレクトリの移動および名前変更。
- ファイル特性の変更。
- 構成ファイルの作成、変更、および読み取り。

通常の I/O 操作を実行するには、上位 VI を使用します。各ファイル I/O 操作を個別に制御するには、下位 VI および関数を使用します。

詳細については

ファイル I/O 操作の実行については、「LabVIEW ヘルプ」を参照してください。

ファイル I/O の基本

通常のファイル I/O 操作には、以下のプロセスが含まれます。

1. ファイルを作成するか、またはファイルを開きます。パスを指定するか、ダイアログボックス内でファイルの場所を指定することによって、既存ファイルの保存場所または新規ファイルの作成場所を指定します。ファイルを開くと、refnum がそのファイルを表します。refnum の詳細については、第 4 章「フロントパネルを作成する」の「オブジェクトまたはアプリケーションへのリファレンス」のセクションを参照してください。
2. ファイルから読み取るか、ファイルに書き込みます。
3. ファイルを閉じます。

ほとんどのファイル I/O VI および関数は、1 回のファイル I/O 操作で 1 つのステップのみを実行します。ただし、通常のファイル I/O 操作用に設計された一部の高レベルファイル I/O VI は、3 つのステップをすべて実行します。これらの VI は低レベル関数ほど効率的でない場合もありますが、使いやすさの面では優れています。

ファイル I/O 形式を選択する

使用するファイル I/O VI は、ファイルの形式によって異なります。ファイルに対するデータの読み書きは、テキスト、バイナリ、およびデータログの 3 つの形式で実行できます。使用する形式は、集録または作成の対象となるデータと、そのデータにアクセスするアプリケーションによって異なります。

使用する形式を決定するには、以下の基本的なガイドラインに従ってください。

- Microsoft Excel など、他のアプリケーションでもデータを使用できるようにする場合は、テキストファイルを使用します。これは、テキストファイルが最も一般的であり、最も移植性が高いためです。
- ランダムアクセスによるファイルの読み書きを実行する必要がある場合や、速度の向上とディスク容量の節約が非常に重要な場合は、バイナリファイルを使用します。これは、バイナリファイルの方がテキストファイルよりもディスク容量および速度の面で効率が良いためです。
- 複雑なデータレコードやさまざまなデータタイプを LabVIEW で処理する場合は、データログファイルを使用します。これは、データへのアクセス元を LabVIEW に限定する場合や、複雑なデータ構造を保存する必要がある場合、データログファイルがデータの保存に最適であるためです。

テキストファイルを使用する場合

ディスク容量とファイル I/O 速度が重要でない場合や、ランダムアクセスによる読み書きを行う必要がない場合、数値の精度が重要でない場合に、他のユーザまたはアプリケーションでもデータを使用できるようにするには、データに対してテキスト形式のファイルを使用します。

テキストファイルは、最も使いやすく最も共有しやすい形式です。テキストファイルの読み書きは、ほとんどすべてのコンピュータで実行できます。テキストベースのファイルの読み取りは、さまざまなテキストベースのプログラムで行うことができます。テキスト文字列は、ほとんどの計測制御アプリケーションで使用されます。

ワープロまたはスプレッドシートアプリケーションなど、他のアプリケーションからもデータにアクセスする場合は、テキストファイルにデータを保存します。テキスト形式でデータを保存するには、文字列関数を使用してすべてのデータをテキスト文字列に変換します。テキストファイルには、さまざまなデータタイプの情報を格納できます。

通常、データの ASCII 表現はデータ自体よりもサイズが大きいため、元のデータがテキスト形式でなくグラフやチャートなどのデータである場

合、テキストファイルはバイナリファイルやデータログファイルよりも多くのメモリを使用します。たとえば、−123.4567 という数値は、4 バイトの単精度浮動小数点数として保存できます。ただし、この数値の ASCII 表現は、1 文字あたり 1 バイトずつの合計 9 バイトを使用します。

また、テキストファイル内の数値データにランダムにアクセスするのは困難です。これは、文字列内の各文字が正確に 1 バイトの容量を使用していても、数値をテキストとして表現するために必要な容量が一定ではないためです。テキストファイル内の 9 番目の数値を検出するには、LabVIEW はまず、その前にある 8 つの数値を読み取り変換する必要があります。

テキストファイルに数値データを保存すると、精度が低下する場合があります。コンピュータは数値データをバイナリデータとして保存しますが、ユーザは一般的に数値データを 10 進表記法でテキストファイルに書き込みます。精度の低下は、テキストファイルへデータを書き込むときに発生する場合があります。バイナリファイルでは、精度の低下が問題となることはありません。

テキストファイルでのファイル I/O の使用例については、`examples¥file¥smplfile.llb` と `examples¥file¥sprdsht.llb` を参照してください。

バイナリファイルを使用する場合

整数などのバイナリデータの保存には、ディスク上の一定のバイト数を使用します。たとえば、0 ～ 40 億の数値（1、1,000、1,000,000 など）をバイナリ形式で保存すると、1 つの数値あたり 4 バイトが使用されます。

数値データを保存したり、ファイルの特定の数値にアクセスしたり、ファイルの数値にランダムアクセスするには、バイナリファイルを使用します。人間が読めるテキストファイルとは異なり、バイナリファイルを読み取ることができるのはコンピュータのみです。バイナリファイルは、最も容量が小さく高速なデータ保存形式です。バイナリファイルには複数のデータタイプを使用できますが、一般的ではありません。

バイナリファイルは効率的なデータ保存形式です。これは、使用するディスク容量が少なくてすむだけでなく、データの保存や取得時にデータをテキストに変換したり、テキストから変換したりする必要がないためです。バイナリファイルは、1 バイトのディスク容量で 256 個の値を表現できます。拡張精度数値や複素数の場合を除き、バイナリファイルには、データのバイト単位の画像がメモリに保存されていたとおりに格納されています。データがこのように格納されていると変換が不要になるため、ファイルを速く読み取ることができます。LabVIEW がデータを格納する方法の詳細については、『LabVIEW Data Storage』アプリケーションノートを参照してください。

メモ　テキストファイルとバイナリファイルは、バイトストリームファイルとして知られています。これは、データを連続した文字またはバイトとして保存することを意味します。

バイナリファイルに対する倍精度浮動小数点数の配列の読み書きのサンプルについては、`examples¥file¥smplfile.llb` 内の Read Binary File VI と Write Binary File VI を参照してください。

データログファイルを使用する場合

データのアクセスおよび操作を LabVIEW でのみ行う場合や、複雑なデータストラクチャをすばやく簡単に保存するには、データログファイルを使用します。

データログファイルでは、スプレッドシートと同様に、同じ構造を持つ一連のレコードとしてデータが保存されます。各行がレコードを表します。データログファイル内の各レコードには、同じデータタイプを関連付ける必要があります。LabVIEW は、各レコードを保存するデータを含むクラスタとしてファイルに書き込みます。ただし、データログレコードの構成要素は、ファイルの作成時に指定した任意のデータタイプにすることができます。

たとえば、レコードのデータタイプが文字列および数値のクラスタであるデータログを作成できます。この場合、データログの各レコードは文字列および数値のクラスタになります。ただし、最初のレコードを（`"abc",1`）にして、2 番目のレコードを（`"xyz",7`）にすることができます。

データログファイルを使用すると、読み書きを高速化する操作がほとんど必要ありません。また、元のデータブロックをレコードとして読み取ることができ、ファイル内のそのデータブロックの前にあるレコードをすべて読み取る必要がないため、データの取得が簡単になります。データログファイルでは、レコード番号だけでレコードにアクセスできるため、ランダムアクセスが高速で簡単になります。LabVIEW は、データログファイルの作成時にレコード番号を各レコードに順番に割り当てます。

データログファイルには、フロントパネルとブロックダイアグラムからアクセスできます。フロントパネルからデータログファイルへのアクセスの詳細については、この章の「フロントパネルのデータを記録する」のセクションを参照してください。

LabVIEW は、関連する VI を実行するたびにデータログファイルにレコードを書き込みます。LabVIEW がデータログファイルに書き込んだ後で、そのレコードを上書きすることはできません。データログファイルを読み取るときは、一度に 1 つまたは複数のレコードを読み取ることができます。

データログファイルの読み書きの例については、`examples¥file¥datalog.llb` を参照してください。

上位ファイル I/O VI を使用する

以下のデータタイプの読み書きといった通常の I/O 操作を実行するには、上位ファイル I/O VI を使用します。

- テキストファイル内の文字。
- テキストファイル内の行。
- スプレッドシートテキストファイル内の単精度数値の 1 次元配列または 2 次元配列 。
- バイナリファイル内の単精度数値または 16 ビット符号付き整数の 1 次元配列または 2 次元配列 。

上位 VI を使用してファイルに読み書きすることによって、プログラミングの労力と時間を節約できます。上位 VI は、ファイルの開閉だけでなく、読み書きを実行します。VI が実行のたびに閉じる / 開くの操作を行うため、上位 VI をループ内に配置しないでください。複数の操作を実行している間にファイルを開いたままにしておく方法については、この章の「ディスクストリーミング」のセクションを参照してください。

上位 VI にはファイルパスを入力する必要があります。ファイルパスを配線しないと、読み書きの対象となるファイルの指定を要求するダイアログボックスが表示されます。エラーが発生すると、上位 VI はエラーについて説明するダイアログボックスを表示します。ユーザは、実行を停止するか継続するかを選択できます。

図 14-1 は、高レベルの「スプレッドシートファイルに書き込み（Write To Spreadsheet File）」VI を使用して Microsoft Excel のスプレッドシートファイルに数値を送信する方法を示しています。この VI を実行すると、LabVIEW は、既存ファイルへのデータの書き込みまたは新規ファイルの作成をユーザに要求します。

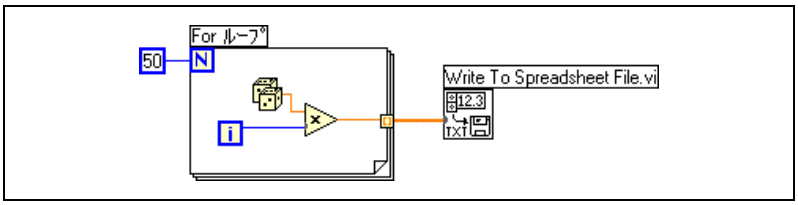


図 14-1　上位 VI によるスプレッドシートへの書き込み

バイナリファイル VI を使用して、バイナリ形式のファイルに対して読み書きを実行します。データは、整数または単精度浮動小数点数を使用することができます。

下位および上級のファイル I/O および関数を使用する

各ファイル I/O 操作を個別に制御するには、下位ファイル I/O および関数と上級ファイル I/O 関数を使用します。

ファイルを作成するかまたはファイルを開いて、ファイルに対するデータの読み書きを行い、ファイルを閉じるには、主要な下位関数を使用します。以下のタスクを実行するには、他の下位関数を使用します。

- ディレクトリの作成。
- ファイルの移動、コピー、または削除。
- ディレクトリの内容のリスト。
- ファイル特性の変更。
- パスの操作。

左図に示すパスは、ディスク上のファイルの場所を識別する LabVIEW データタイプです。パスは、ファイルが格納されているボリューム、ファイルシステムの最上位とそのファイル間のディレクトリ、およびそのファイルの名前を記述します。パス制御器またはパス表示器では、特定のプラットフォームの標準構文を使用してパスを入力または表示します。パス制御器および表示器の詳細については、第 4 章「フロントパネルを作成する」の「パス制御器および表示器」のセクションを参照してください。

図 14-2 は、下位 VI および関数を使用して、Microsoft Excel のスプレッドシートファイルに数値を送る方法を示しています。この VI を実行すると、「開く／作成／置換ファイル（Open/Create/Replace）」VI が `numbers.xls` ファイルを開きます。「ファイルに書き込み（Write File）」関数は、数字の文字列をファイルに書き込みます。「ファイルを閉じる（Close File）」関数はファイルを閉じます。ファイルを閉じないと、ファイルがメモリ内に残っているため、他のアプリケーションまたは他のユーザはそのファイルにアクセスできません。

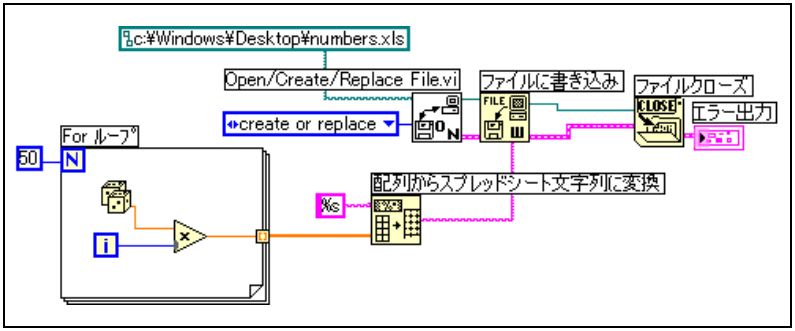


図 14-2　下位 VI によるスプレッドシートへの書き込み

図 14-2 の VI と図 14-1 の VI を比較してください。これらの VI は同じタスクを実行します。図 14-2 では、「配列からスプレッドシート文字列に変換（Array To Spreadsheet String）」関数を使用して、数値の配列を文字列としてフォーマットする必要があります。図 14-1 の「スプレッドシートファイルに書き込み（Write To Spreadsheet File）」VI は、ファイルを開き、数値の配列を文字列に変換して、ファイルを閉じます。

下位ファイル I/O VI および関数の使用例については、`examples¥file¥datalog.llb` 内の Write Datalog File Example VI を参照してください。

ディスクストリーミング

下位ファイル I/O VI および関数を使用してディスクストリーミングを行うと、メモリリソースを節約できます。ディスクストリーミングは、ループ内などで複数の書き込みを実行している間、ファイルを開いたままにしておくためのテクニックです。高レベルの書き込み操作は使いやすい反面、実行するたびにファイルの開閉を行うため、オーバーヘッドが増えます。同じファイルを頻繁に開閉しないようにすると、VI の効率を向上させることができます。

ディスクストリーミングにより、ファイルの開閉時に関数がオペレーティングシステムと対話する回数が減ります。一般的なディスクストリーミング操作を作成するには、ループの前に「開く／作成／置換ファイル（Open/Create/Replace File）」VI を配置し、ループの後に「ファイルを閉じる（Close File）」関数を配置します。これにより、ファイルの開閉によるオーバーヘッドを伴わずにループ内でファイルへの連続書き込みを実行できます。

ディスクストリーミングは、実行速度が重要な、大量のデータ集録に最適です。集録の実行中は、ファイルにデータを連続的に書き込むことができます。最適な結果を得るには、集録が完了するまで、解析 VI や解析関数などの他の VI や関数を実行しないでください。

テキストファイルとスプレッドシートファイルを作成する

テキストファイルにデータを書き込むには、データを文字列に変換する必要があります。スプレッドシートファイルにデータを書き込むには、文字列をスプレッドシート文字列としてフォーマットする必要があります。スプレッドシート文字列は、タブなどの区切り文字を含んでいる文字列です。文字列のフォーマットの詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「文字列をフォーマットする」のセクションを参照してください。

テキストを読み取ることができるほとんどのワープロアプリケーションではフォーマット済みテキストが不要なため、テキストファイルにテキストを書き込む場合はフォーマットする必要がありません。テキストファイルにテキスト文字列を書き込むには、「文字をファイルに書き込み（Write Characters To File）」VI を使用します。この VI はファイルの開閉を自動的に行います。

グラフ、チャート、または集録データ内の一連の数値をスプレッドシート文字列に変換するには、「スプレッドシートファイルに書き込み（Write To Spreadsheet File）」VI または「配列からスプレッドシート文字列に変換（Array To Spreadsheet String）」関数を使用します。これらの VI および関数の使用方法の詳細については、この章の「上位ファイル I/O VI を使用する」および「下位および上級のファイル I/O および関数を使用する」のセクションを参照してください。

ワープロアプリケーションでは、上位ファイル I/O VI が処理できないさまざまなフォント、色、スタイル、およびサイズを使用してテキストが構成されるため、ワープロアプリケーションからテキストを読み取るとエラーが発生する場合があります。

スプレッドシートアプリケーションまたはワープロアプリケーションに数値やテキストを書き込む場合は、文字列関数と配列関数を使用してデータをフォーマットし、文字列を結合します。その後、ファイルにデータを書き込みます。これらの関数を使用してデータのフォーマットおよび結合を行う方法については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」を参照してください。

データのフォーマットとファイルへの書き込み

文字列、数値、パス、およびブール値データをテキストとしてフォーマットし、フォーマットされたテキストをファイルに書き込むには、「ファイルにフォーマット（Format Into File）」関数を使用します。多くの場合、「文字列にフォーマット（Format Into String）」関数を使用して文字列を個別にフォーマットし、「文字をファイルに書き込み（Write Characters to File）」VI または「ファイルに書き込み（Write File）」関数を使用して結果の文字列を書き込む代わりに、この関数を使用できます。

文字列のフォーマットの詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「文字列をフォーマットする」のセクションを参照してください。

ファイルからデータをスキャンする

ファイル内のテキストをスキャンして文字列、数値、パス、およびブール値を探し、そのテキストをデータタイプに変換するには、「ファイルからスキャン（Scan From File）」関数を使用します。多くの場合、「ファイル読み取り（Read File）」関数または「ファイルから文字を読み取り（Read Characters From File）」VI を使用してファイルからデータを読み取り、「文字列からスキャン（Scan From String）」関数を使用して結果の文字列をスキャンする代わりに、この関数を使用できます。

バイナリファイルを作成する

バイナリファイルを作成するには、一連の数値を集録し、その数値をファイルに書き込みます。16 ビット整数または単精度浮動小数点数の 1 次元配列および 2 次元配列をファイルに保存するには、「I16 ファイルに書き込み （Write To I16 File）」VI と「SGL ファイルに書き込み（Write To SGL File ）」VI を使用します。作成したファイルを読み取るには、「I16 ファイル読み取り（Read From I16 File）」VI と「SGL ファイル読み取り（Read From SGL File）」VI を使用します。

倍精度浮動小数点数や 32 ビット符号なし整数などの異なるデータタイプの数値を書き込むには、下位関数または上級ファイル関数を使用します。ファイルを読み取るときは、「ファイル読み取り（Read File）」関数を使用して数値のデータタイプを指定します。

バイナリファイルに対する浮動小数点数の読み書きのサンプルについては、`examples¥file¥smplfile.llb` 内の Read Binary File VI と Write Binary File VI を参照してください。

データログファイルを作成する

フロントパネルのデータロギングを有効にするか、下位関数または上級ファイル関数を使用してデータ集録およびファイルへのデータの書き込みを行うことによって、データログファイルの作成および読み取りを実行できます。フロントパネルからデータログファイルへの作成およびアクセスの詳細については、この章の「フロントパネルのデータを記録する」のセクションを参照してください。

データログファイル内のデータをフォーマットする必要はありません。ただし、データログファイルの読み書きを行うときは、データタイプを指定する必要があります。たとえば、温度の測定値およびその温度の測定時刻と日付を集録した場合は、データログファイルにデータを書き込み、1 つの数値と 2 つの文字列のクラスタとしてそのデータを指定します。データログファイルへのデータの書き込みのサンプルについては、`examples¥file¥datalog.llb` 内の Simple Temp Datalogger VI を参照してください。

温度の測定値およびその温度の測定時刻と日付が集録されているファイルを読み取る場合は、1 つの数値と 2 つの文字列のクラスタの読み取りを指定します。データログファイルへのデータの読み取りのサンプルについては、`examples¥file¥datalog.llb` 内の Simple Temp Datalog Reader VI を参照してください。

ファイルに波形を書き込む

ファイルに波形を送るには、「波形をファイルに書き込み（Write Waveforms to File）」VI および 「波形をスプレッドシートファイルにエクスポート（Export Waveforms to Spreadsheet File）」VI を使用します。スプレッドシートファイル、テキストファイル、またはデータログファイルに波形を書き込むことができます。

作成した波形を VI 内でのみ使用する場合は、波形をデータログファイル（`.log`）として保存します。データロギングの詳細については、この章の「データログファイルを使用する場合」のセクションを参照してください。

図 14-3 の VI は複数の波形を集録し、グラフ上に表示して、スプレッドシートファイルに書き込みます。

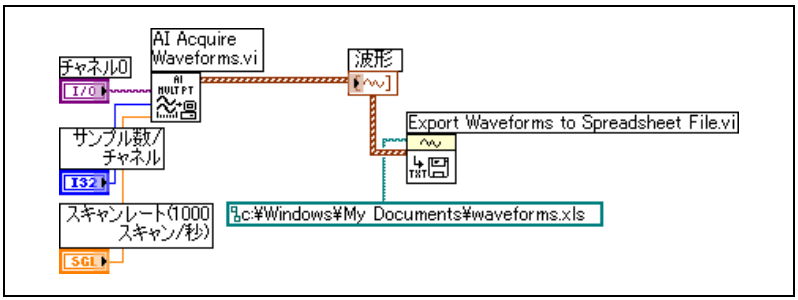


図 14-3　スプレッドシートファイルへの複数の波形の書き込み

ファイルから波形を読み取る

ファイルから波形を読み取るには、「ファイルから波形読み取り（Read Waveform from File）」VI を使用します。1 つの波形を読み取った後、「波形作成 (Build Waveform)」関数を使用して波形データタイプの要素を追加または編集したり、「波形属性取得（Get Waveform Attribute）」関数を使用して波形要素を抽出できます。

図 14-4 の VI は、ファイルから波形を読み取り、その波形の **dt** 要素を編集して、編集後の波形をグラフにプロットします。

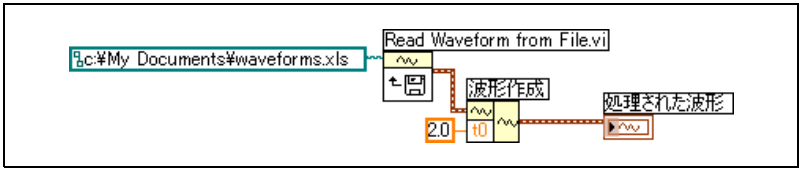


図 14-4　ファイルからの波形の読み取り

「ファイルから波形読み取り（Read Waveform from File）」VI は、ファイルから複数の波形を読み取ることもできます。この VI は波形データタイプの配列を返します。この配列はマルチプロットグラフに表示できます。ファイル内の 1 つの波形にアクセスする場合は、図 14-5 のように、波形データタイプの配列に指標を付ける必要があります。配列への指標付けの詳細については、第 10 章「文字列、配列、およびクラスタを使用してデータをグループ化する」の「配列」のセクションを参照してください。この VI は、複数の波形が格納されているファイルにアクセスします。「指標配列（Index Array）」関数はファイル内の最初および 3 番目の波形を読み取り、その波形を 2 つの別個の波形グラフ上にプロットします。グラフの詳細については、第 8 章「ループとストラクチャ」を参照してください。

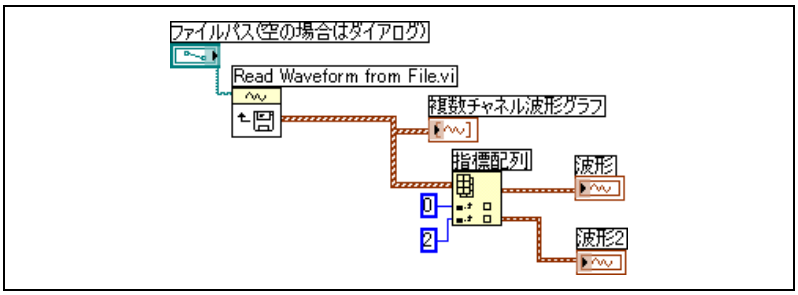


図 14-5　ファイルからの複数の波形の読み取り

フロースルーパラメータ

多くのファイル I/O VI および関数には、対応する入力パラメータと同じ値を返すフロースルーパラメータ（通常は Refnum またはパス）があります。関数の実行順序を制御するには、このパラメータを使用します。最初に実行するノードのフロースルー出力を、次に実行するノードの対応する入力に接続することによって、人工データ依存を確立します。このフロースルーパラメータを使用しない場合は、シーケンスストラクチャを使用して、希望の順序でファイル I/O 操作が実行されるようにする必要があります。人口データ依存の詳細については、第 5 章「ブロックダイアグラムを作成する」の「データ依存と人工データ依存」のセクションを参照してください。

構成ファイルを作成する

標準の Windows 構成 (`.ini`) ファイルを読み取って作成したり、これらの VI によって生成されるファイルを複数のプラットフォームで使用できるように、プラットフォームに依存しない形式でパスなどのプラットフォーム固有のデータを書き込むには、構成ファイル VI を使用します。構成ファイル VI は、ファイルの構成に標準ファイル形式を使用しません。任意のプラットフォームで構成ファイル VI を使用し、VI によって作成されたファイルを読み書きできますが、構成ファイル VI を使用して Mac OS や UNIX 形式で構成ファイルを作成したり修正することはできません。

構成ファイル VI の使用例については、`examples¥file¥config.llb` を参照してください。

メモ　Windows 構成ファイルの標準の拡張子は `.ini` ですが、ファイルの内容が正しい形式であれば、構成ファイル VI はどのような拡張子を持つファイルでも処理します。内容の構成については、この章の「Windows 構成ファイルの形式」のセクションを参照してください。

構成ファイルを使用する

標準的な Windows 構成ファイルは、テキストファイルにデータを保存するための特殊な形式です。.ini ファイルは特殊な形式に従っているため、ファイル内のデータにプログラムで簡単にアクセスできます。

たとえば、次のような内容を持つ構成ファイルについて考えてみます。

```
[ データ ]
Value=7.2
```

図 14-6 のように、構成ファイル VI を使用するとこのデータを読み取ることができます。この VI は「キー読み取り（Read Key)」VI を使用して、データという**セクション**から値という**キー**を読み取ります。この VI は、ファイルが Windows 構成ファイルの形式であれば、ファイルがどのように変更されても動作します。

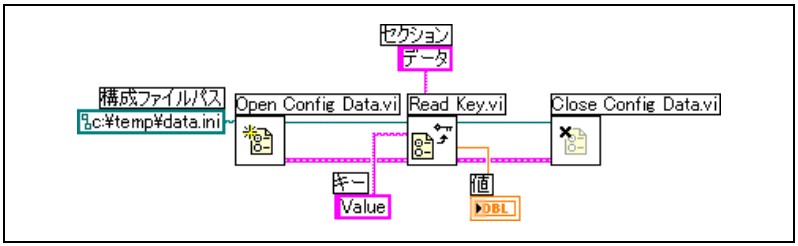


図 14-6　.ini ファイルからのデータの読み取り

Windows 構成ファイルの形式

Windows 構成ファイルは、名前付きのセクションに分けられたテキストファイルです。各セクション名は括弧で囲まれています。1 つのファイル内でセクション名が重複しないようにしてください。セクションには、等号（=）で区切られたキーと値のペアが含まれています。各セクション内でキー名が重複しないようにしてください。キー名は構成の環境設定を表し、値名はその環境設定の設定値を表します。次の例は、ファイル内の項目の配置を示しています。

```
[Section 1]
key1=value
key2=value

[Section 2]
key1=value
key2=value
```

キーパラメータの値の部分には、構成ファイル VI を使用して以下のデータタイプを指定します。

- 文字列
- パス
- ブール
- 64 ビット倍精度浮動小数点数
- 32 ビット符号付き整数
- 32 ビット符号なし整数

構成ファイル VI は、未処理またはエスケープ文字列データの読み書きを実行できます。この VI は、未処理データを ASCII に変換せずに、そのデータをバイト単位で読み書きします。変換された文字列、つまりエスケープ文字列の場合、LabVIEW は、構成ファイル内の非表示テキスト文字を等価の 16 進エスケープコード（復帰文字では `¥0D`）として保存します。また、LabVIEW は、構成ファイル内の円コード文字を 2 つの円コード（`¥` を `¥¥`）として保存します。構成ファイル VI の**未処理文字列の読み取り？**または **未処理文字列書き込み？**入力を、未処理データの場合は TRUE に設定し、エスケープデータの場合は FALSE に設定します。

VI が構成ファイルに書き込むと、スペース文字を含む文字列データまたはパスデータの前後には引用符が付けられます。文字列に引用符が含まれている場合、LabVIEW は引用符を `¥"` として保存します。テキストエディタを使用して構成ファイルの読み書きを行うと、LabVIEW によって引用符が `¥"` に置換されていることがわかります。

LabVIEW は、プラットフォームに依存しない標準的な UNIX パス形式でパスデータを `.ini` ファイルに保存します。この VI は、構成ファイル内に保存されている絶対パス `/c/temp/data.dat` を次のように解釈します。

- **(Windows)** `c:¥temp¥data.dat`
- **(Mac OS)** `c:temp:data.dat`
- **(UNIX)** `/c/temp/data.dat`

この VI は、相対パス `temp/data.dat` を次のように解釈します。

- **(Windows)** `temp¥data.dat`
- **(Mac OS)** `:temp:data.dat`
- **(UNIX)** `temp/data.dat`

フロントパネルのデータを記録する

他の VI やレポートで使用するためにデータを記録するには、フロントパネルのデータロギングを使用します。たとえば、グラフからデータを記録し、そのデータを別の VI の別のグラフ内で使用できます。

VI を実行するたびに、フロントパネルのデータロギングによって別のデータログファイルに保存されます。このファイルは、テキストが区切られた形式になっています。データは以下の方法で取り出すことができます。

- 対話形式でデータを取り出すには、データの記録元と同じ VI を使用します。
- プログラムでデータを取り出すには、VI をサブ VI として使用します。
- ファイル I/O VI および関数を使用します。

各 VI は、データログファイルの場所を記録するログファイル連結を保持します。LabVIEW は、その場所に記録済みのフロントパネルデータを保持します。ログファイルの連結とは、VI データの記録先となるデータログファイルと VI の関係です。

データログファイルには、VI の各実行時のタイムスタンプおよびデータを含むレコードが格納されています。データログファイルにアクセスするときは、回収モードで VI を実行し、フロントパネル制御器を使用してデータを表示することによって、必要なレコードを選択します。回収モードで VI を実行すると、フロントパネルの一番上に数値制御器が表示されます。この制御器を使用するとレコード間を移動できます。この数値制御器の例については、図 14-7 を参照してください。

自動および対話形式でのフロントパネルのデータロギング

自動ロギングを有効にするには、**操作→ VI 終了後にログ**を選択します。VI のフロントパネルデータを初めて記録する際、LabVIEW はデータログファイルに名前を入力するようプロンプトします。VI を実行するたびにデータが記録され、VI を再び実行するたびに新規レコードがデータログファイルに追加されます。LabVIEW がデータログファイルに書き込んだ後で、そのレコードを上書きすることはできません。

データを対話形式で記録するには、**操作→データロギング→ログ**を選択します。LabVIEW は、データログファイルにデータを即座に追加します。対話形式のデータロギングでは、データを記録する時刻を選択できます。自動的なデータロギングでは、VI を実行するたびにデータが記録されます。

メモ　波形チャートでは、フロントパネルのデータロギングによって一度に 1 つのデータ点のみが記録されます。チャート表示器に配列を接続した場合、データログファイルには、チャートに表示される配列のサブセットが格納されます。

記録済みのフロントパネルデータを対話形式で表示する

データを記録した後、**操作→データロギング→回収**を選択することによって、データを対話形式で表示できます。図 14-7 のようなデータ回収ツールバーが表示されます。

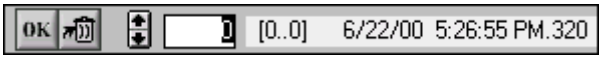


図 14-7　データ回収ツールバー

ハイライトされている数字は、現在表示中のデータレコードを示します。角括弧内の数字は、現在の VI についてログしたレコードの範囲を示します。VI を実行するたびにレコードが記録されます。日付と時刻は、選択したレコードがログされた日付と時刻を示します。次のレコードまたは前のレコードを表示するには、増分矢印または減分矢印をクリックします。キーボードの上矢印キーと下矢印キーを使用することもできます。

データ回収ツールバーだけでなくフロントパネルの外観も、ツールバーで選択したレコードに応じて変化します。たとえば、増分矢印をクリックして別のレコードに進むと、データを記録した時点におけるその特定のレコードのデータが制御器と表示器に表示されます。回収モードを終了し、表示していたデータログファイルを持つ VI に戻るには、**OK** ボタンをクリックします。

レコードを削除する

回収モード時に特定のレコードを削除できます。回収モードで個々のレコードを削除対象としてマークするには、そのレコードを表示し、**ごみ箱**ボタンをクリックします。**ごみ箱**ボタンをもう一度クリックすると、そのレコードは削除対象から外されます。

削除対象としてマークしたレコードをすべて削除するには、回収モード時に**操作→データロギング→データを排出**を選択します。

OK ボタンをクリックする前にマークしたレコードを削除しないと、LabVIEW はマークしたレコードを削除するようプロンプトします。

ログファイル連結を消去する

フロントパネルデータの記録または回収時に使用するデータログファイルと VI を関連付けるには、ログファイル連結を使用します。1 つの VI に複数のデータログファイルを関連付けることができます。これは、VI データをテストしたり比較するのに役立ちます。たとえば、VI を初めて実行したときのログデータと、次にその VI を実行したときのログデータを比較することができます。複数のデータログファイルと VI を関連付けるには、**操作→データロギング→ログファイル連結の消去**を選択して、ログファイ

ル連結を消去する必要があります。自動ロギングを有効にするか、データを記録することを対話式に選択したときのいずれかで、次回その VI を実行すると、LabVIEW はデータロギングを指定するようプロンプトします。

ログファイル連結を変更する

別のログファイルに対してフロントパネルデータの記録または回収を行うようにログファイルの連結を変更するには、**操作→データロギ ング→ログファイル連結の変更**を選択します。LabVIEW は、別のログファイルを選択するか、新規ログファイルを作成するようプロンプトします。別のデータを VI 内に取り込む場合や、VI から別のデータログファイルにデータを追加する場合は、ログファイル連結を変更できます。

プログラムでフロントパネルデータを取り出す

サブ VI を使用するか、ファイル I/O VI および関数を使用して、記録済みデータを取り出すこともできます。

サブ VI を使用してフロントパネルデータを取り出す

サブ VI を右クリックし、ショートカットメニューから**データベースアクセスを可能にする**を選択すると、図 14-8 のように、サブ VI の周りに黄色いボックスが表示されます。

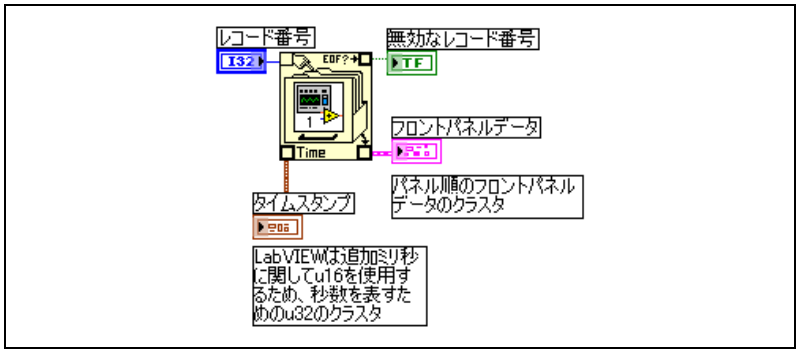


図 14-8　記録済みデータの取り出し

ファイルキャビネットのような形をした黄色いボックスには、データログファイルからデータにアクセスするための端子が含まれています。サブ VI に対するデータベースアクセスを可能にすると、サブ VI の入力および出力が実際には出力として働き、そのログデータを返します。**レコード #** は取り出すレコードを示し、**無効レコード**はレコード番号が存在するかどうかを示します。**タイムスタンプ**はレコードの作成時刻であり、**フロントパネルデータ**はフロントパネルオブジェクトのクラスタです。フロントパ

ネルオブジェクトのデータにアクセスするには、**フロントパネルデータ**クラスタを「バンドル解除（Unbundle）」関数に接続します。

図 14-9 のように、サブ VI 上の対応する端子に直接配線することによって、特定の入力および出力の値を取り出すこともできます。

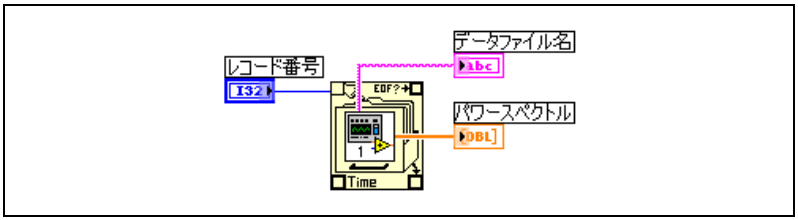


図 14-9　サブ VI の端子を経由したログデータの取り出し

VI を実行すると、サブ VI は実行されません。その代わりに、サブ VI のフロントパネルから VI のフロントパネルにログデータがクラスタとして返されます。

メモ　サブ VI または Express VI を拡張可能ノードとして表示する場合は、そのノードに対するデータベースアクセスを有効にすることはできません。

レコードを指定する

サブ VI には *n* 個のログデータがあり、*–n* ～ *n* – 1 の任意の数値をサブ VI の**レコード #** 端子に配線できます。負でないレコード番号を使用して、最初の記録済みレコードを基準としてレコードにアクセスできます。0 は最初のレコードを表し、1 は 2 番目のレコードを表し、最後のレコード を表す *n* – 1 まで以下同様に続きます。

負のレコード番号を使用して、最後に記録済みレコードを基準としてレコードにアクセスできます。–1 は最後のレコードを表し、–2 は最後から 2 番目のレコードを表し、最初のレコードを表す *–n* まで以下同様に続きます。*–n* ～ *n* – 1 の範囲外の数値を**レコード #** 端子に接続すると、**無効レコード #** 出力は TRUE になり、サブ VI はデータを取り出しません。

ファイル I/O 関数を使用してフロントパネルデータを取り出す

「ファイル読み取り（Read File）」関数などのファイル I/O VI および関数を使用して、フロントパネルから記録済みデータを取り出すこともできます。フロントパネルのデータログファイルの各レコードのデータタイプによって、2 つのクラスタが作成されます。一方のクラスタにはタイムスタンプが含まれており、もう一方のクラスタにはフロントパネルデータが含

まれています。タイムスタンプクラスタには、秒を表す 32 ビット符号なし整数と、LabVIEW 標準時間（1904 年 1 月 1 日午前 0:00）から経過したミリ秒を表す 16 ビット符号なし整数が含まれています。

フロントパネルデータログファイルのレコードには、プログラムで作成したデータログファイルへのアクセスに使用するものと同じファイル I/O 関数を使用してアクセスします。図 14-10 のように、「ファイルを開く（File Open）」関数へのタイプ入力として**データログタイプ**を入力します。

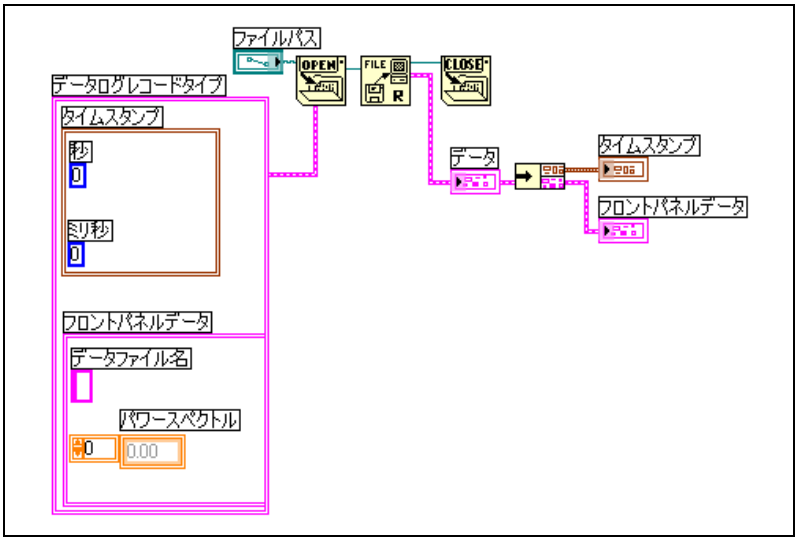


図 14-10　「ファイルを開く（File Open）」関数を使用した記録済みデータの取り出し

LabVIEW データディレクトリ

デフォルト `LabVIEW` データディレクトリを使用して、`.lvm` や `.txt` ファイルなど、LabVIEW が生成するデータファイルを保存します。LabVIEW は、デフォルトファイルディレクトリに `LabVIEW` データディレクトリをインストールします。これは、ご使用のオペレーティングシステムで LabVIEW が生成するデータファイルの整理および検索をするときに役立ちます。デフォルトでは、「LabVIEW 計測ファイル書き込み（Write LabVIEW Measurement File）」Express VI は、このディレクトリで生成した `.lvm` ファイルを保存し、「LabVIEW 計測ファイル読み取り（Read LabVIEW Measurement File）」Express VI は、このディレクトリから読み取ります。また、左図に示すデフォルトデータディレクトリの定数およびデフォルトデータディレクトリのプロパティは、デフォルトで `LabVIEW` データディレクトリを返します。

ツール→オプションを選択し、プルダウンメニューから**パス**を選択して、異なるデフォルトディレクトリを指定します。デフォルトデータディレクトリは、デフォルトディレクトリとは異なります。デフォルトディレクトリは、新規 VI、カスタム制御器、VI テンプレート、または作成したその他の LabVIEW ドキュメント用に指定したディレクトリです。

LabVIEW 計測データファイル

LabVIEW 測定データファイル（`.lvm`）には、「LabVIEW 計測ファイル書き込み（Write LabVIEW Measurement File）」Express VI が生成するデータが含まれています。LabVIEW データファイルは、スプレッドシートアプリケーションまたはテキスト編集アプリケーションで開くことができるタブ区切りテキストファイルです。Express VI が生成するデータの他に、`.lvm` ファイルには、データが生成された日付や時間などのデータに関する情報が含まれています。

Express VI を使用してデータを保存および回収する方法については、『LabVIEW 入門』マニュアルを参照してください。

LabVIEW 測定データファイルは NI DIAdem とともに使用することもできます。

15

VI の文書化と印刷

LabVIEW を使用して、VI の文書化および印刷を行うことができます。

この章の「VI を文書化する」のセクションでは、開発段階でブロックダイアグラムやフロントパネルについての情報を VI の印刷文書に記録する方法を説明しています。

この章の「VI を印刷する」のセクションでは、VI の印刷オプションについて説明しています。VI についての情報の印刷に適したオプションと、VI によって生成されたデータおよび結果の報告に適したオプションがあります。使用する印刷方法（手作業とプログラムのどちらで印刷するかなど）、レポート形式に必要なオプションの数、作成するスタンドアロンアプリケーションにおける機能の必要性、および VI を実行するプラットフォームなどには、いくつかの要素が影響します。

詳細については

VI の文書化と印刷およびレポート作成の詳細については、『LabVIEW ヘルプ』を参照してください。

VI を文書化する

開発作業の追跡、完成した VI の文書化、および VI のユーザに対する指示の作成を実行できます。ドキュメントは、LabVIEW 内で表示したり、印刷したり、HTML、RTF、またはテキストファイルに保存できます。

VI のドキュメントを効果的に作成するには、VI のレビジョン履歴にコメントを追加して VI とオブジェクトの説明を作成します。

VI のレビジョン履歴を設定する

レビジョン番号など VI の開発履歴を表示するには、各 VI の**履歴**ウィンドウを使用します。VI に対する変更にともなって、**履歴**ウィンドウでは、その変更を記録および追跡します。**履歴**ウィンドウを表示するには、**ツール→ VI レビジョン履歴**を選択します。レビジョン履歴を印刷したり、HTML、RTF、またはテキストファイルに保存したりすることもできます。リビジョン履歴の印刷またはファイルへの保存については、この章の「ドキュメントを印刷する」のセクションを参照してください。

レビジョン番号

レビジョン番号によって、VI に対する変更を簡単に追跡できます。レビジョン番号はゼロで始まり、VI を保存するたびに 1 ずつ増加します。VI のタイトルバーや**履歴**ウィンドウのタイトルバーに現在のレビジョン番号を表示するには、**ツール→オプション**を選択し、一番上のプルダウンメニューから**レビジョン履歴**を選択して、**タイトルバーにレビジョン番号を表示する**チェックボックスにチェックマークを付けます。

LabVIEW の**履歴**ウィンドウに表示する番号は、現在のレビジョン番号に 1 を加えた次のレビジョン番号です。履歴にコメントを追加すると、コメントのヘッダに次のレビジョン番号が含まれます。履歴のみを変更した場合、VI を保存してもレビジョン番号は増加しません。

レビジョン番号は、**履歴**ウィンドウ内のコメントとは無関係です。コメント間のレビジョン番号の食い違いは、コメントを付けずに VI を保存したことを示します。

厳密に言えば履歴は開発時にのみ使用するため、実行ファイル作成の際、VI のブロックダイアグラムを削除すると、その履歴は LabVIEW によって自動的に削除されます。ブロックダイアグラムの削除については、第 7 章「VI およびサブ VI を作成する」の「VI を配布する」のセクションを参照してください。**履歴**ウィンドウは、ランタイムバージョンの VI では使用できません。VI にブロックダイアグラムがない場合でも、**VI プロパティ**ダイアログボックスの**一般**ページにはレビジョン番号が表示されます。レビジョン履歴を削除し、レビジョン番号を 0 にリセットするには、**履歴**ウィンドウの**リセット**ボタンをクリックします。

VI およびオブジェクトの説明を作成する

VI またはオブジェクトの目的を説明したり、その使用方法をユーザに指示するには、VI とそのオブジェクト (制御器や表示器など) の説明を作成します。LabVIEW で説明を表示したり、印刷したり、HTML、RTF、またはテキストファイルに保存したりすることができます。

VI の説明を作成、編集、または表示するには、**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**ドキュメント**を選択します。オブジェクトの説明の作成、編集、および表示を行うには、オブジェクトを右クリックし、ショートカットメニューから**説明とヒント**を選択します。ヒントラベルは、VI の実行中にカーソルをオブジェクト上に移動したときに表示される簡単な説明です。**説明とヒント**ダイアログボックスにヒントを入力しないと、ヒントラベルは表示されません。VI のアイコンまたはオブジェクト上に個別にカーソルを移動すると、**ヘルプ**ウィンドウに VI またはオブジェクトの説明が表示されます。

ドキュメントを印刷する

VI ドキュメントを印刷したり、HTML、RTF、またはテキストファイルに保存するには、**ファイル→印刷**を選択します。ドキュメントの作成は、ビルトイン形式または、カスタムの形式を選択できます。作成するドキュメントには、以下の項目を含めることができます。

- アイコンとコネクタペーン
- フロントパネルとブロックダイアグラム
- 制御器、表示器、およびデータタイプ端子
- VI およびオブジェクトの説明
- VI 階層
- サブ VI のリスト
- レビジョン履歴

メモ　特定のタイプの VI 用に作成するドキュメントには、以前のすべての項目を含めることはできません。たとえば、多形性 VI にフロントパネルまたはブロックダイアグラムがない場合は、多形性 VI 用に作成するドキュメントにこれらの項目を含めることができません。

HTML、RTF、またはテキストファイルにドキュメントを保存する

VI ドキュメントを HTML、RTF、、またはテキストファイルとして保存できます。HTML ファイルと RTF ファイルはほとんどのワープロアプリケーションにインポートでき、これらのファイルを使用するとコンパイルされたヘルプファイルを作成できます。また、LabVIEW によって生成された HTML ファイルを使用して VI ドキュメントをウェブに表示することもできます。HTML ファイルと RTF ファイルを使用してヘルプファイルを作成する方法については、この章の「独自のヘルプファイルを作成する」のセクションを参照してください。ドキュメントの印刷方法およびドキュメントの HTML、RTF、およびテキストファイルへのプログラムでの保存方法については、この章の「VI をプログラムで印刷する」のセクションを参照してください。

RTF ファイルにドキュメントを保存する場合は、オンラインヘルプファイルを作成するか、あるいはワープロ用ファイルを作成するかを指定します。ヘルプファイル形式では、グラフィックを外部のビットマップファイルに保存します。ワープロファイル形式では、グラフィックをドキュメント内に埋め込みます。HTML ファイルでは、すべてのグラフィックを JPEG 形式、PNG 形式、または GIF 形式で外部に保存します。

HTML ファイル用のグラフィック形式を選択する

HTML ファイルにドキュメントを保存する場合は、グラフィックファイルの形式とカラー深度を選択できます。

JPEG 形式はグラフィックの圧縮率が高い反面、グラフィックの細部が損なわれることがあります。この形式は写真に最も適しています。線画や、フロントパネル、ブロックダイアグラムなどの場合、JPEG 形式で圧縮するとグラフィックがぼやけたり、色むらが発生することがあります。JPEG グラフィックは常に 24 ビットグラフィックです。モノクロなどカラー深度を低く選択した場合、グラフィックは、指定したカラー深度で保存されますが 24 ビットグラフィックのままです。

PNG 形式でも、JPEG 形式ほどではありませんが、高圧縮率でグラフィックが圧縮されます。ただし、PNG 圧縮では細部が損なわれません。また、PNG 形式では、1 ビット、4 ビット、8 ビット、および 24 ビットのグラフィックがサポートされています。ビットを下げると、得られるグラフィックは JPEG 形式よりも圧縮率が大幅に高くなります。PNG 形式は GIF（Graphics Interchange Format）形式に取って代わるものです。PNG 形式は JPEG や GIF よりも優れていますが、Web ブラウザでは JPEG や GIF ほどサポートされていません。

GIF 形式もグラフィックの圧縮性能が優れており、しかも大半のウェブブラウザでサポートされています。ライセンス上の関係で、LabVIEW ではグラフィックを圧縮 GIF ファイルとして保存できませんが、将来的にサポートする可能性もあります。LabVIEW が保存する未圧縮 GIF ファイルを圧縮 GIF ファイルに変換するには、グラフィック形式コンバータを使用してください。高品質の圧縮 GIF ファイルには、ドキュメントを保存する際に PNG 形式を選択し、グラフィック形式コンバータを使用して、保存する PNG ファイルを GIF ファイルに変換します。PNG 形式は元のグラフィックを損なわずに再現するため、PNG 形式を変換元にすると高品質のグラフィックを作成できます。LabVIEW で作成された HTML ファイルを、`.gif` 拡張子の付 いた GIF ファイルを参照するように変更します。

グラフィックファイルの命名規約

外部グラフィックとともに HTML または RTF ドキュメントを作成する場合、LabVIEW は制御器および表示器のデータタイプ端子を同じ名前のグラフィックファイルに保存します。VI に同じタイプの端子が複数ある場合、LabVIEW はそのタイプのグラフィックファイルを 1 つだけ作成します。たとえば、VI に 32 ビット符号付き整数の入力が 3 つある場合、LabVIEW は 1 つの `ci32.`*x* ファイルを作成します。ここで、*x* はグラフィック形式に対応した拡張子です。

独自のヘルプファイルを作成する

LabVIEW で生成した HTML ファイルまたは RTF ファイルを使用して、独自のコンパイルされたヘルプファイルを作成することができます。**(Windows)** LabVIEW で生成した個別の HTML ファイルを HTM ヘルプファイルにコンパイルすることができます。

LabVIEW が生成した RTF ファイルを **(Windows)** WinHelp、**(Mac OS)** QuickHelp、または **(UNIX)** HyperHelp ファイルにコンパイルすることができます。

ファイル→ VI プロパティを選択し、**カテゴリ**プルダウンメニューから**ドキュメント**を選択して、VI から HTML ファイルまたはコンパイル済みの ヘルプファイルへのリンクを作成することができます。

VI を印刷する

主に以下の方法を使用して VI を印刷できます。

- アクティブウィンドウの内容を印刷するには、**ファイル→ウィンドウの印刷**を選択します。
- フロントパネル、ブロックダイアグラム、サブ VI、制御器、VI 履歴に関する情報など、VI についてのより総合的な情報を印刷するには、**ファイル→印刷**を選択します。この方法による VI の印刷については、この章の「ドキュメントを印刷する」のセクションを参照してください。
- 任意の VI ウィンドウまたは VI ドキュメントをプログラムで随時印刷するには、VI サーバを使用します。この方法による VI の印刷については、第 17 章「VI をプログラム的に制御する」を参照してください。

アクティブウィンドウを印刷する

アクティブなフロントパネルまたはブロックダイアグラムのウィンドウの内容を最小限のプロンプトで印刷するには、**ファイル→ウィンドウの印刷**を選択します。LabVIEW はアクティブウィンドウの作業スペースを印刷しますが、アクティブウィンドウのサイズによる制限はありません。タイトルバー、メニューバー、ツールバー、スクロールバーは印刷されません。

ファイル→ウィンドウの印刷を選択した場合や、プログラム的に印刷する場合、LabVIEW による VI の印刷方法を設定するには、**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**印刷オプション**を選択します。プログラムによる印刷の詳細については、この章の「VI をプログラムで印刷する」のセクションを参照してください。

VI をプログラムで印刷する

ファイル→ウィンドウの印刷および**ファイル→印刷**を選択して表示されるダイアログボックスを使用せずに、VI をプログラム的に印刷するには、以下の方法を使用します。

- VI の実行が終了するたびにそのフロントパネルを自動的に印刷するように VI を設定します。
- VI を印刷するサブ VI を作成します。
- レポートを印刷したり、VI のドキュメントや VI が返すデータを含む HTML 形式レポートを保存するには、レポート生成 VI を使用してください。
- プログラム的に VI ウィンドウを印刷したり、VI のドキュメントを印刷したり、または HTML、RTF、またはテキストファイル形式でドキュメントを保存するには、VI サーバを使用します。この方法による VI の印刷については、第 17 章「VI をプログラム的に制御する」を参照してください。

メモ　作成したアプリケーションから VI のドキュメントを印刷する場合は、フロントパネルのみを印刷できます。

VI 実行後に VI のフロントパネルを印刷する

VI の実行が終了して、VI のフロントパネルを印刷するには、**操作→ VI 終了後に印刷**を選択します。また、**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**印刷オプション**を選択して、**VI が実行を完了する度に、自動的にパネルを印刷**チェックボックスにチェックマークを付ける方法もあります。

このオプションを選択する方法は、フロントパネルがアクティブウィンドウである場合の**ファイル→ウィンドウの印刷**の選択に似ています。

VI をサブ VI として使用した場合、そのサブ VI の実行が終了すると、そのサブ VI が呼び出し側 VI に戻る前に印刷が行われます。

サブ VI を使用して上位 VI のデータを印刷する

VI の実行が終了するたびに VI を印刷するのではなく、ユーザがボタンをクリックしたときや、ある状況 (テストの失敗など) が発生したときにのみ印刷が必要となる場合があります。また、場合によっては、プリントアウトの形式をより正確に制御したり、部分制御器のみを印刷する必要があります。このような場合は、終了時に印刷するように設定したサブ VI を使用できます。

サブ VI を作成し、希望の印刷方法に合わせてフロントパネルをフォーマットします。上位 VI 内で**操作→ VI 終了後に印刷**を選択する代わりに、

サブ VI 内でこれを選択します。印刷するときは、サブ VI を呼び出し、印刷するデータをサブ VI に接続します。

レポートを生成および印刷する

レポートを印刷するか、または VI のドキュメントや VI が返すデータを含む HTML 形式レポートを保存するには、レポート生成 VI を使用してください。「簡易 VI パネル印刷または文書化（Easy Print VI Panel or Documentation）」VI を使用して、VI のドキュメントを含む基本的なレポートを生成します。「簡易テキストレポート（Easy Text Report）」VI を使用して、VI が返すデータを含む基本的なレポートを生成します。他のレポート生成 VI を使用すると、より複雑なレポートを生成することができます。

メモ　レポート生成は、LabVIEW 開発システムおよびプロフェッショ ナル開発システムでのみ使用できます。

レポート生成 VI を使用すると、以下のタスクを実行します。

- レポートへのテキスト、グラフィック、表、または VI ドキュメントの追加。
- テキストのフォント、サイズ、スタイル、および色の設定。
- レポートの向き (縦または横) の設定。
- レポートのヘッダおよびフッタの設定。
- 余白およびタブの設定。

その他の印刷方法

LabVIEW の標準的な印刷方法がニーズに合わない場合は、以下に挙げるその他の方法を使用してください。

- データを 1 行ずつ印刷します。シリアルポートにラインプリンタを接続している場合は、シリアル互換性の関数を使用してプリンタにテキストを送信します。通常、これを行うには、プリンタのコマンド言語についての知識が多少必要です。
- Microsoft Excel など、他のアプリケーションにデータをエクスポートし、そのデータをファイルに保存して、そのアプリケーションから印刷します。
- **（Windows、Mac OS X、および UNIX）** System Exec VI を使用します。
- **（Mac OS）** AESend Print Document VI を使用します。
- **（Windows）** 別のアプリケーションでデータを印刷するには、ActiveX を使用します。ActiveX の詳細については、第 19 章「Windows の接続性」を参照してください。

16

VI をカスタマイズする

アプリケーションのニーズに応じて動作するように VI とサブ VI を構成できます。たとえば、ユーザ入力が必要なサブ VI として VI を使用する場合は、その VI を呼び出すたびにフロントパネルが表示されるように VI を構成します。

VI はさまざまな方法で構成できます。その構成方法は、その VI 内で構成するか、VI サーバを使用してプログラム的に構成するかといずれかです。VI サーバを使用して VI の動作方法を構成する詳細については、第 17 章「VI をプログラム的に制御する」を参照してください。

詳細については

VI のカスタマイズの詳細については、『LabVIEW ヘルプ』を参照してください。

VI の外観と動作を設定する

VI の外観と動作を設定するには、**ファイル→ VI プロパティ**を選択します。ダイアログボックスの一番上にある**カテゴリ**プルダウンメニューを使用して、さまざまなオプションのカテゴリから選択します。オプションには次のカテゴリがあります。

- **一般**：VI の現在の保存先のパス、VI のレビジョン番号、レビジョン履歴、および VI の最終保存後の変更内容を表示します。このページを使用してアイコン、または VI のアライメントグリッドのサイズを編集することもできます。
- **ドキュメント**：このページを使用して、VI の説明を追加したり、ヘルプファイルのトピックにリンクします。文書オプションの詳細については、第 15 章「VI の文書化と印刷」の「VI を文書化する」のセクションを参照してください。
- **セキュリティ**：このページを使用して、VI をロックしたり、パスワードで保護します。
- **ウィンドウの外観**：このページを使用して、さまざまなウィンドウの外観を設定します。
- **ウィンドウサイズ**：このページを使用して、ウィンドウのサイズを設定します。

- **実行**：このページを使用して、VI の実行方法を構成します。たとえば、VI を開くとすぐに実行するように構成したり、サブ VI として呼び出されると一時停止するように構成することができます。また、さまざまな優先順位で実行するように VI を構成できます。たとえば、他の操作が完了するのを待たずに VI を実行する必要がある場合、時間重視（最高）の優先順位で実行するように VI を構成します。マルチスレッド VI の作成の詳細については、『Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability』アプリケーションノートを参照してください。
- **編集オプション**：このページを使用して、現在の VI のアライメントグリッドサイズを設定したり、端子を右クリックしてショートカットメニューから**作成→制御器**または**作成→表示器**を選択して、LabVIEW が作成する制御器または表示器のスタイルを変更します。アライメントグリッドの詳細については、第 4 章「フロントパネルを作成する」の「オブジェクトを整列および均等に配置する」のセクションを参照してください。

メニューをカスタマイズする

作成するすべての VI に対してカスタムメニューを作成したり、VI のメニューバーの表示 / 非表示を構成できます。メニューバーの表示 / 非表示を切り替えるには、**ファイル→ VI プロパティ**を選択し、**カテゴリ**プルダウンメニューから**ウィンドウの外観**ページを選択し、**カスタマイズ**するボタンをクリックし、**メニューバーを表示する**チェックボックスをオンまたはオフにします。

メニューの構成には、メニューの作成と、ユーザがメニュー項目を選択したときにその選択に対応するブロックダイアグラムコードを書くという作業が含まれます。

メモ　カスタムメニューは VI 実行時にのみ表示されます。

メニューを作成する

VI の編集時は静的に、VI の実行時はプログラム的に、カスタムメニューの作成またはデフォルトの LabVIEW メニューの変更を行うことができます。**編集→ランタイムメニュー**を選択し、**メニュー編集**ダイアログボックス内でメニューを作成すると、ランタイムメニューファイル（`.rtm`）が作成されます。このファイルを使用すると、デ フォルトメニューバーではなくカスタムメニューバーが使用できるようになります。`.rtm` ファイルを作成して保存する際に、VI と `.rtm` ファイルの相対パスを同一にしておく必要があります。

カスタムの .rtm ファイルと VI を関連付けるには、**メニュー編集**ダイアログボックスを使用します。VI を実行すると、VI はその .rtm ファイルからメニューを読み取ります。**メニュー編集**ダイアログボックスを使用して、LabVIEW がデフォルトメニューで提供するメニュー項目であるアプリケーション項目、またはユーザが追加するメニュー項目であるユーザ項目を持つカスタムメニューを作成することができます。LabVIEW はアプリケーション項目の動作を定義しますが、ユーザ項目の動作はユーザがブロックダイアグラムで制御します。ユーザメニュー項目の選択の詳細については、本章の「メニュー選択処理」のセクションを参照してください。

メニュー編集ダイアログボックスを使用して、VI の編集時にメニューをカスタマイズします。メニュー関数を使用して、実行時にメニューをプログラム的にカスタマイズします。これらの関数を使用して、ユーザ項目の属性を挿入、削除、変更することができます。アプリケーション項目を追加または削除するだけで、LabVIEW がアプリケーション項目の動作および状態を定義します。

メニュー選択処理

カスタムメニューを作成するときは、大文字と小文字の区別がない固有の文字列識別子（であるタグ）を各メニュー項目に割り当てます。ユーザがメニュー項目を選択したときは、「メニュー項目情報取得（Get Menu Selection）」関数を使用してそのタグをプログラム的に取り出します。
各メニュー項目のタグの値に基づいて、LabVIEW はブロックダイアグラム上で各メニュー項目に対しハンドラを提供します。ハンドラとは While ループとケースストラクチャの組み合わせで、これを使用すると、どのメニューが選択されているかを調べて適切なコードを実行することができます。

カスタムメニューを作成した後、そのメニュー内の各項目を実行または処理するケースストラクチャをブロックダイアグラムで作成します。このプロセスはメニュー選択処理と呼ばれます。LabVIEW は、すべてのアプリケーション項目を自動的に処理します。

図 16-1 では、ユーザが選択したメニュー項目を「メニュー項目情報取得」関数が読み取ってケースストラクチャに渡し、ケースストラクチャでメニュー項目が実行されています。

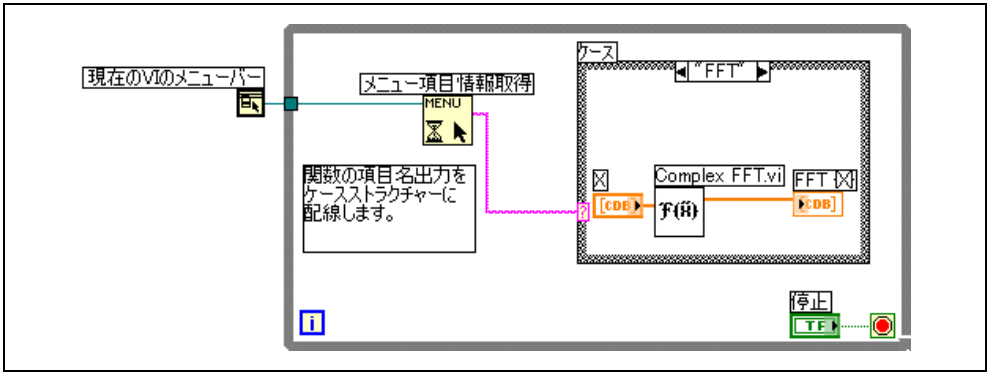


図 16-1　メニュー処理を使用したブロックダイアグラム

あるメニュー項目の処理に長時間かかることがわかっている場合は、「メニュー項目情報取得」関数の**ブロックメニュー**入力にブール制御器を配線し、そのブール制御器を TRUE に設定してメニューバーを無効にします。これにより、LabVIEW がメニュー項目を処理している間はユーザがメニュー上で他の項目を選択できなくなります。LabVIEW によるメニュー項目の処理が完了した後、「メニュートラッキング有効（Enable Menu Tracking）」関数に TRUE 値を配線してメニューバーを有効にしてください。

また、イベントストラクチャを使用してメニューイベントを処理することもできます。イベントストラクチャの詳細については、第 9 章「イベント駆動型プログラミング」を参照してください。

17

VI をプログラム的に制御する

ブロックダイアグラム、ActiveX テクノロジ、および TCP プロトコルを介して VI サーバにアクセスし、VI や他の LabVIEW のインスタンスと通信して VI および LabVIEW をプログラム的に制御することができます。VI サーバの操作はローカルコンピュータ上やネットワークを介してリモートで実行できます。

詳細については

プログラムで VI を制御する方法の詳細については、『LabVIEW ヘルプ』を参照してください。

VI サーバの機能

VI サーバを使用すると、以下のプログラム的操作を実行できます。

- リモートで VI を呼び出します。
- ウェブ上で LabVIEW の他のインスタンスから呼び出す VI をエクスポートするサーバとして、LabVIEW のインスタンスを構成します。たとえば、遠隔地でデータを集録して記録するデータ集録アプリケーションがあれば、いつでもローカルコンピュータからそのデータをサンプリングできます。LabVIEW 環境設定を変更してウェブ上で VI にアクセスできるようにすることによって、最新のデータの転送がサブ VI 呼び出しと同じように簡単になります。ネットワークの細部は VI サーバによって処理されます。また、VI サーバはプラットフォームにまたがって実行可能なので、異なるプラットフォーム上でクライアントとサーバを実行できます。
- VI および LabVIEW のプロパティを編集します。たとえば、ダイナミックに VI ウィンドウの位置を設定したり、フロントパネルの特定の場所が表示されるようにスクロールできます。また、プログラム的にディスクにすべての変更内容を保存できます。
- 各 VI に対し**ファイル→ VI プロパティ**ダイアログボックスを使用して手作業で更新するのではなく、複数の VI のプロパティを更新します。
- バージョン番号や改版などの LabVIEW のインスタンスの情報を取り出します。また、LabVIEW を実行しているプラットフォームなどの環境情報も取り出すことができます。

- VI を開くときにすべてのサブ VI をロードするのではなく、他の VI で呼び出す必要がある場合に、ダイナミックにそれらの VI をメモリにロードします。
- アプリケーションのプラグインアーキテクチャを作成することにより、カスタマに配布した後でもアプリケーションに機能を追加できるようになります。たとえば、データをフィルタ処理する VI のセットがあり、その VI すべてが同じパラメータを使用するとします。これらの VI が 1 つのプラグインディレクトリからダイナミックにロードされるようにアプリケーションを設計することにより、これらの VI のセットの一部だけを組み込んだ状態で出荷し、プラグインディレクトリに新しいフィルタ処理 VI をロードするだけでユーザのフィルタ処理オプションをさらに拡張することができます。

VI サーバアプリケーションを作成する

VI サーバアプリケーションのプログラミングモデルは Refnum をベースにしています。Refnum はファイル I/O、ネットワーク接続、および LabVIEW の他のオブジェクトでも使用されます。Refnum の詳細については、第 4 章「フロントパネルを作成する」の「オブジェクトまたはアプリケーションへのリファレンス」のセクションを参照してください。

通常、まず LabVIEW のインスタンスまたは VI の Refnum を開きます。次に、他の VI に対するパラメータとして Refnum を使用します。VI はプロパティを取得（読み取り）または設定（書き込み）し、メソッドを実行して、参照されている VI をダイナミックにロードします。最後に、Refnum を閉じ、参照されている VI をメモリから解放します。

VI サーバアプリケーションを作成するには、以下のアプリケーション制御の関数およびノードを使用します。

- **アプリケーションリファレンスを開く (Open Application Reference)**：サーバを介してアクセスしているローカルまたはリモートアプリケーションのリファレンスを開いたり、リファレンスを開いて LabVIEW のリモートインスタンスにアクセスします。
- **VI リファレンスを開く（Open VI Reference）**：ローカルまたはリモートコンピュータの VI のリファレンスを開くか、ダイナミックにディスクから VI をロードします。
- **プロパティノード（Property Node）**：VI、オブジェクト、またはアプリケーションのプロパティを取得または設定します。プロパティの詳細については、この章の「プロパティノード」のセクションを参照してください。

- **インボークノード（Invoke Node）**：VI、オブジェクト、またはアプリケーション上のメソッドを呼び出します。メソッドの詳細については、この章の「インボークノード」のセクションを参照してください。
- **リファレンス呼び出しノード（Call By Reference Node）**：ダイナミックにロードした ＶＩ を呼び出します。
- **リファレンスを閉じる（Close Reference）**：VI サーバを使用してアクセスした VI、オブジェクト、またはアプリケーションの開いているリファレンスを閉じます。

アプリケーションリファレンスと VI リファレンス

VI サーバ機能へのアクセスは、アプリケーションオブジェクトと VI オブジェクトの 2 つの主要クラスのオブジェクトのリファレンスを通じて行います。これらのオブジェクトのいずれかに対するリファレンスを作成すると、そのオブジェクト上で操作を実行する VI や関数にそのリファレンスを渡すことができます。

アプリケーション Refnum は LabVIEW のローカルまたはリモートのインスタンスを参照します。アプリケーションプロパティおよびメソッドを使用して、LabVIEW の環境設定を変更したり、システム情報を返すことができます。VI の Refnum は LabVIEW のインスタンスの VI を参照します。

LabVIEW のインスタンスの Refnum を使用すると、LabVIEW を実行しているプラットフォーム、バージョン番号、現在メモリにロードされているすべての VI のリストなどの LabVIEW の環境についての情報を取り出すことができます。現在のユーザ名や LabVIEW の他のインスタンスのエクスポート VI のリストなどの情報を設定することもできます。

VI の Refnum を作成すると、メモリに VI がロードされます。VI は、その Refnum を閉じるまでメモリにあります。開いている 1 つの VI について複数の Refnum が同時に存在する場合は、その VI のすべての Refnum を閉じるまで、その VI はメモリ内に入ったままです。VI の Refnum を使用すると、フロントパネルウィンドウの位置などのダイナミックなプロパティだけでなく、**ファイル→ VI プロパティ**ダイアログボックスで操作できる VI のすべてのプロパティを更新できます。また、プログラムで VI の文書を印刷したり、別の場所に VI を保存したり、その文字列をエクスポートおよびインポートして他の言語に変換したりすることもできます。

アプリケーションと VI の設定値を操作する

VI サーバを使用したアプリケーションや VI の設定値の取得および設定は、プロパティノードやインボークノードを使用して行います。プロパティノードやインボークノードを使用しないと、多くのアプリケーションや VI の設定値を取得したり設定することができません。

アプリケーションクラスと VI クラスのプロパティおよびメソッドのサンプルについては、`examples¥viserver` を参照してください。

プロパティノード

アプリケーションや VI のさまざまなプロパティを取得または設定するには、プロパティノードを使用します。プロパティノードからプロパティを選択するには、操作ツールでプロパティ端子をクリックするか、そのノードの白い領域を右クリックし、ショートカットメニューから**プロパティ**を選択します。

1 つのノードを使用して複数のプロパティの読み書きができます。ただし、一部のプロパティには書き込みができません。プロパティノードをサイズ変更して新しい端子を追加するには、位置決めツールを使用します。小さな矢印がプロパティの右側にあるものはプロパティの読み取り、左側にあるものはプロパティの書き込みになります。プロパティの動作を変更するには、そのプロパティを右クリックし、ショートカットメニューから**読み取りに変更**または**書き込みに変更**を選択します。

ノードは上から下の順に実行されます。プロパティノードは実行前にエラーが発生すると実行されません。エラーが発生したかどうかを常に確認してください。プロパティでエラーが発生すると、残りのプロパティを無視し、エラーを返します。**エラー出力**クラスタには、どのプロパティによってエラーが発生したのかを示す情報が含まれています。

プロパティノードが開いて、アプリケーションまたは VI のリファレンスを返した場合は、「リファレンスを閉じる（Close Reference）」関数を使用してアプリケーションまたは VI リファレンスを閉じます。リファレンスが必要なくなると、LabVIEW は制御器リファレンスを閉じます。制御器リファレンスを明示的に閉じる必要はありません。

プロパティノードの自動リンク

フロントパネルオブジェクトからプロパティノードを作成する（オブジェクトを右クリックしてショートカットメニューから**作成→プロパティノード**を選択）と、フロントパネルオブジェクトに自動的にリンクされたプロパティノードがブロックダイアグラム上に作成されます。これらのプロパティノードは自動的に作成元のオブジェクトにリンクされるので、**Refnum** 入力はありません。また、プロパティノードをフロントパネルオブジェクトの端子や制御器 refnum に配線する必要はありません。制御器リファレンスの詳細については、この章の「フロントパネルオブジェクトを制御する」のセクションを参照してください。

インボークノード

アプリケーションや VI に対して動作（メソッド）を実行するにはインボークノードを使用します。プロパティノードとは異なり、1 つのインボークノードではアプリケーションや VI に対して 1 つのメソッドしか実行できません。メソッドを選択するには、操作ツールでメソッド端子をクリックするか、そのノードの白い領域を右クリックしてショートカットメニューから**メソッド**を選択します。

メソッド名は常にインボークノードのパラメータリストの最初の端子名になります。メソッドによって値が返される場合は、メソッド端子には返される値が表示されます。それ以外の場合、メソッド端子に値は表示されません。

インボークノードによって上から下にパラメータがリストされ、メソッド名が上に、オプションのパラメータが下に淡色表示されます。

アプリケーションクラスのプロパティとメソッドを操作する

LabVIEW のローカルまたはリモートのインスタンス上でプロパティを取得または設定したり、メソッドを実行したり、またはその両方を行うことができます。図 17-1 は、ローカルコンピュータ上のメモリに入っているすべての VI をフロントパネル上の文字列配列に表示する方法を示しています。

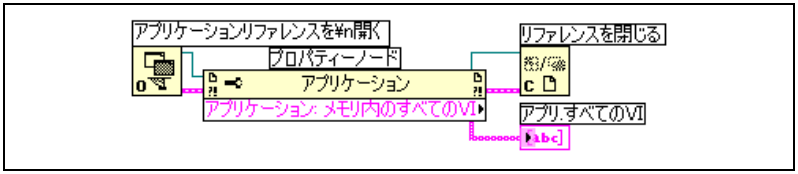


図 17-1　ローカルコンピュータ上のメモリ内の VI をすべて表示する

Refnum を**リファレンス**入力に配線しないと、プロパティノードやインボークノードは LabVIEW の現在のインスタンスのリファレンスを使用します。LabVIEW の他のインスタンスのプロパティまたはメソッドを操作する場合は、アプリケーション Refnum を**リファレンス**入力に配線する必要があります。

リモートコンピュータ上のメモリにある VI を検索するには、図 17-2 のように「アプリケーションリファレンスを開く（Open Application Reference）」関数の**マシン名**入力に文字列制御器を配線し、IP アドレスやドメイン名を入力します。また、図 17-1 で使用されている**メモリ内のすべての VI** プロパティは LabVIEW のローカルインスタンスのみに適用されるので、**メモリ内のエクスポート VI** プロパティを選択する必要があります。

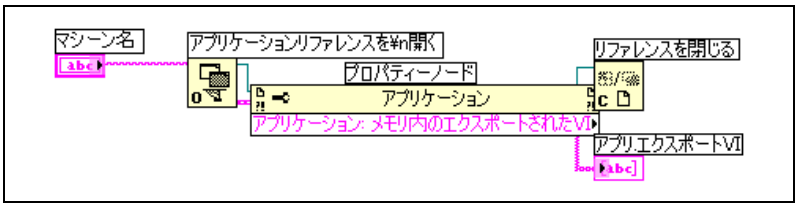


図 17-2　リモートコンピュータ上のメモリ内の VI をすべて表示する

VI クラスのプロパティとメソッドを操作する

VI のプロパティの取得や設定、VI 上でのメソッドの実行、またはその両方を実行できます。図 17-3 のように、LabVIEW はインボークノードを使用して VI のフロントパネルオブジェクトをそれぞれのデフォルト値にもう一度初期化します。フロントパネルが開き、プロパティノードを使用してデフォルト値が表示されます。

Refnum を**リファレンス**入力に配線しない場合、プロパティノードまたはインボークノードは、プロパティノードまたはインボークノードを含む VI のリファレンスを使用します。他の VI のプロパティまたはメソッドを操作する場合、VI Refnum を**リファレンス**入力に配線する必要があります。

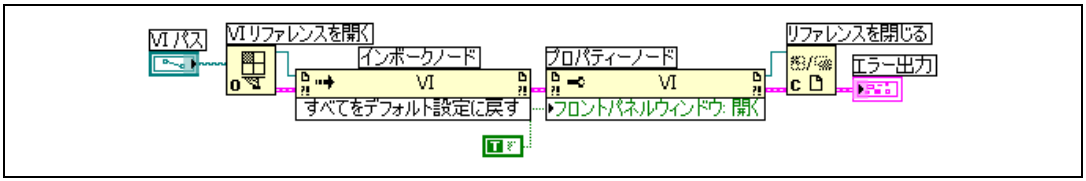


図 17-3　VI クラスのプロパティとインボークノードを使用する

プロパティノードはインボークノードと同様に動作します。プロパティノードに VI Refnum を配線すると、すべての VI クラスプロパティにアクセスできます。

アプリケーションクラスと VI クラスのプロパティおよびメソッドを操作する

VI によっては、アプリケーションクラスと VI クラスの両方のプロパティまたはメソッドにアクセスする必要があります。図 17-4 のように、アプリケーションクラスと VI クラスの Refnum を別々に開いたり、閉じたりする必要があります。

図 17-4 は、ローカルコンピュータ上のメモリに入っている VI のパスをフロントパネルにある文字列の表示器の配列に表示する方法を示しています。メモリ内のすべての VI を検索するには、アプリケーションクラスプロパティにアクセスする必要があります。これらの各 VI へのパスを調べ

るには、VI クラスのプロパティにアクセスする必要があります。For ループの実行回数はメモリ内の VI 数によって決まります。メモリ内の各 VI に対して VI Refnum が必要なので、「VI リファレンスを開く（Open VI Reference）」関数および「リファレンスを閉じる（Close Reference）」関数を For ループ内部に入れる必要があります。アプリケーション Refnum は、For ループがすべての VI のパス検索を終了した後で閉じてください。

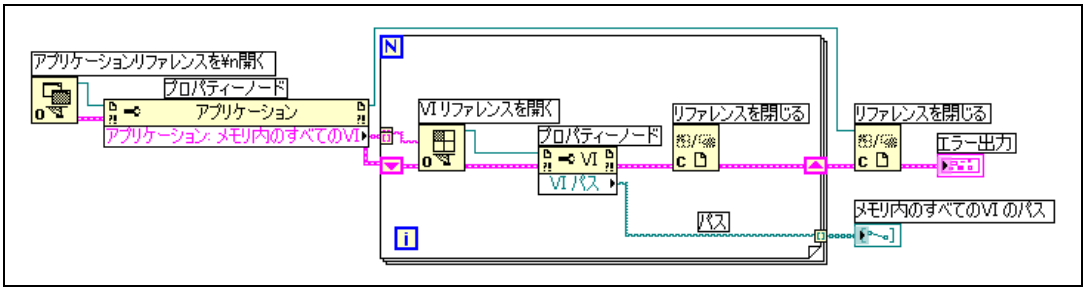


図 17-4　アプリケーションクラスと VI クラスのプロパティおよびメソッドを使用する

VI のダイナミックなロードと呼び出し

スタティックにリンクされたサブ VI を使用せずに、ダイナミックに VI を呼び出すことができます。スタティックにリンクされたサブ VI は呼び出し側 VI のブロックダイアグラム上に直接配置するものです。このサブ VI は呼び出し側 VI のロードと同時にロードされます。

スタティックにリンクされたサブ VI とは異なり、ダイナミックにロードされたサブ VI は呼び出し側 VI がそのサブ VI の呼び出しを実行するまでロードされません。サブ VI は呼び出し側 VI で必要になるまでロードされないので、呼び出し側 VI が大きい場合、サブ VI をダイナミックにロードすることによってロード時の時間とメモリを節約でき、さらに操作が終了するとサブ VI をメモリから解放できます。

リファレンス呼び出しノード（Call By Reference Node）と厳密に類別化された VI Refnum

VI をダイナミックに呼び出すには、「リファレンス呼び出しノード（Call By Reference Node）」を使用します。

「リファレンス呼び出しノード 」には厳密に類別化された VI Refnum が必要です。厳密に類別化された VI Refnum によって、呼び出す VI のコネクタペーンが識別されます。厳密に類別化された VI Refnum は、VI への永久的な関連を確立したり、VI の名前や位置などの他の VI 情報を含むわ

けではありません。「リファレンス呼び出しノード （Call By Reference Node）」の入力と出力は、他の VI に配線する場合と同じように配線できます。

図 17-5 は「リファレンス呼び出しノード 」を使用して、「周波数応答（Frequency Response）」VI をダイナミックに呼び出す方法を示しています。「リファレンス呼び出しノード 」には「VI リファレンスを開く（Open VI Reference）」関数と「リファレンスを閉じる（Close Reference）」関数を使用する必要があります。これは、プロパティノードやインボークノードを使用する場合と似ています。

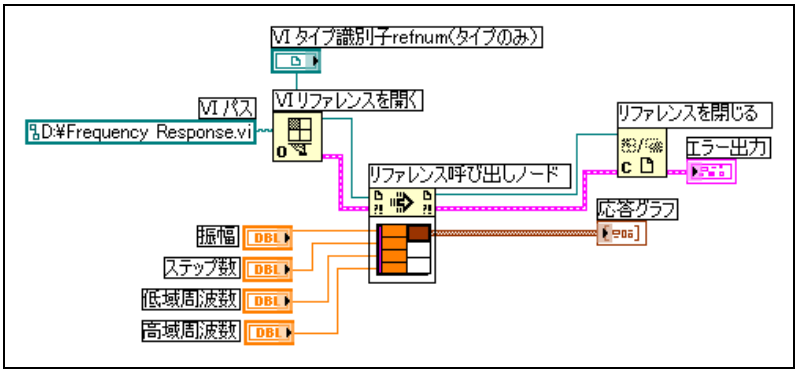


図 17-5　「リファレンス呼び出しノード 」を使用する

厳密に類別化された Refnum として指定された VI は、コネクタペーン情報のみを提供します。このように Refnum と VI 間には固定された関連性がありませんので、これだけではその VI すべての情報は得られません。「リファレンス呼び出しノード 」関数は、選択する VI の Refnum を取得することで VI コネクタペーンの情報を取得し、その他の情報を得るために 「VI リファレンスを開く」関数の **VI パス**入力で指定された VI パスから情報を引き出します。この方法で「リファレンス呼び出しノード 」に必要とする情報を提供することができます。

リモートコンピュータ上での VI の編集と実行

アプリケーションと VI Refnum の両方で重要な点は、それらのネットワークにおける透過性です。つまり、オブジェクトの Refnum は、リモートコンピュータでも、ユーザのコンピュータで開く場合と同じ方法で開くことができます。

いくつかの制約はありますが、リモートオブジェクトの Refnum を開くと、ローカルオブジェクトとほとんど同じ方法でリモートオブジェクトを処理できます。リモートオブジェクトでの操作の場合、VI サーバはネット

ワークを介した操作に関する情報を送信し、結果を返します。アプリケーションは操作がリモートであるかローカルであるかにかかわらず、見た目には同じように機能します。

フロントパネルオブジェクトを制御する

制御器リファレンスは、テキストベースのプログラミング言語でのユーザインタフェースのオブジェクトリファレンスに相当します。制御器リファレンスは、テキストベースのプログラミング言語のポインタには相当しません。

Refunum および**旧バージョンの Refnum** パレットにある制御器 refnum を使用して、フロントパネルオブジェクトのリファレンスを他の VI に渡します。また、フロントパネルオブジェクトを右クリックし、ショートカットメニューから**作成→リファレンス**を選択して、制御器リファレンスを作成することもできます。サブ VI に制御器リファレンスを渡すと、プロパティノードおよびインボークノードを使用して、プロパティの読み書きや、参照されるフロントパネルオブジェクトのメソッドの呼び出しを実行できます。

イベントを使用して、フロントパネルオブジェクトからプログラム的にブロックダイアグラムの動作を制御する方法については、第 8 章「ループとストラクチャ」の「ケースストラクチャとシーケンスストラクチャ」のセクションを参照してください。

厳密に類別化された制御器 refnum と非厳密に類別化された制御器 refnum

厳密に類別化された制御器 refnum は、同じ種類のデータの制御器 refnum のみを受け入れます。たとえば、厳密に類別化された制御器 refnum のタイプが 32 ビット整数のスライドである場合は、制御器 refnum 端子に 32 ビット整数のスライドのみしか配線できません。8 ビット整数のスライド、倍精度スカラのスライド、または 32 ビット整数クラスタのスライドは、制御器 refnum 端子に配線できません。

制御器から作成する制御器リファレンスはデフォルトで、厳密に類別化された制御器 refnum となります。フロントパネル上の制御器リファレンスの左下隅の赤い星は、その制御器リファレンスが、厳密に類別化された制御器リファレンスであることを示します。ブロックダイアグラムでは、制御器リファレンス端子に配線されたプロパティノードやインボークノード上に（厳密）が表示されます。

メモ　ラッチ式機械的動作は厳密に類別化された制御器リファレンスと互換性がないので、ラッチ式機械的動作を行うブール制御器では非厳密に類別化された制御器リファレンスが生成されます。

非厳密に類別化された制御器リファレンスは、より広範囲のタイプのデータを受け入れます。たとえば、非厳密に類別化された制御器リファレンスがスライドである場合、制御器リファレンス端子には 32 ビット整数スライド、単精度スライド、または 32 ビット整数スライドのクラスタのいずれかを配線できます。非厳密に類別化された制御器リファレンスのタイプが制御器である場合、すべてのタイプの制御器の制御器リファレンスを制御器リファレンス端子に配線できます。

メモ　非厳密に類別化された制御器リファレンス端子にプロパティノードを配線する場合、プロパティ値によってバリアントデータが生成されます。バリアントデータは使用前に変換する必要がある場合があります。チャートの履歴データプロパティは、チャートのリファレンスが厳密に類別化されている場合にのみ使用できます。バリアントデータの詳細については、第 5 章「ブロックダイアグラムを作成する」の「バリアントデータを処理する」のセクションを参照してください。

18

LabVIEW のネットワーク動作

VI は、他のアプリケーションやリモートコンピュータで実行中の他のプロセスと通信（ネットワーク動作）を行うことができます。LabVIEW のネットワーク動作機能を使用して、以下のタスクを実行することができます。

- ナショナルインスツルメンツの DataSocket テクノロジを使用して、ネットワーク上で実行中の他の VI とライブデータを共有。
- ウェブ上にフロントパネル画像および VI ドキュメントをパブリッシュ。
- VI からデータを電子メールで送信。
- TCP、UDP、Apple Events、PPC Toolbox などの低レベルのプロトコルを介して、他のアプリケーションや VI と通信する VI を作成。

詳細については

LabVIEW のネットワーク動作の詳細については、『LabVIEW ヘルプ』を参照してください。

ファイル I/O、VI サーバ、ActiveX、およびネットワーク動作から選択する

アプリケーションによっては、ネットワーク化が最良のソリューションにならない場合もあります。他の VI やアプリケーションが読み取れるデータを含むファイルを作成する場合は、ファイル I/O の VI および関数を使用します。ファイル I/O VI および関数の使用方法の詳細について は、第 14 章「ファイル I/O」を参照してください。

他の VI を制御する場合は、VI サーバを使用します。ローカルおよびリモートコンピュータ上にある VI および他の LabVIEW アプリケーションの制御方法の詳細については、第 17 章「VI をプログラム的に制御する」を参照してください。

(Windows) Excel スプレッドシートへの波形グラフの埋め込みなど、マイクロソフト社のアプリケーションの機能にアクセスする場合は、ActiveX VI および関数を使用します。ActiveX が有効なアプリケーション

へのアクセス、および他の ActiveX アプリケーションの LabVIEW へのアクセスを許可する方法の詳細については、第 19 章「Windows の接続性」を参照してください。

ネットワークのクライアントおよびサーバとしての LabVIEW

LabVIEW をクライアントとして使用すると、別のアプリケーションの機能を使用したり、データをサブスクライブ（読み取り）でき、またサーバとして使用すると LabVIEW の機能を他のアプリケーションで使用可能にすることができます。VI サーバを使用してローカルおよびリモートコンピュータ上の VI を制御する方法の詳細については、第 17 章「VI をプログラム的に制御する」を参照してください。プロパティノードを使用してプロパティにアクセスしたり、インボークノードを使用してメソッドを呼び出すことによって、VI を制御します。

プロパティにアクセスしたり他のアプリケーションのメソッドを呼び出すには、プロパティやメソッドへのアクセスに使用するネットワークプロトコルをあらかじめ確立しておく必要があります。使用できるプロトコルには、HTTP や TCP/IP などがあります。選択するプロトコルはアプリケーションによって異なります。たとえば、HTTP プロトコルはウェブへのパブリッシュに適していますが、他の VI が作成したデータを受信する VI を作成する場合は使用できません。これを行うには、TCP プロトコルを使用します。

LabVIEW がサポートしている通信プロトコルの詳細については、この章の「低レベル通信アプリケーション」のセクションを参照してください。

(Windows) LabVIEW を ActiveX サーバまたはクライアントとして使用する ActiveX テクノロジの使用方法の詳細については、第 19 章「Windows の接続性」を参照してください。

DataSocket テクノロジを使用する

ウェブ上またはローカルコンピュータ上で他の VI とライブデータを共有するには、ナショナルインスツルメンツの DataSocket テクノロジを使用します。ウェブブラウザが異なるインターネットテクノロジーをまとめているのと同様に、DataSocket は確立されている計測およびオートメーション用の通信プロトコルをまとめています。

DataSocket テクノロジにより、**DataSocket に接続**ダイアログボックスを通じてフロントパネルから複数の入出力メカニズムにアクセスできます。**DataSocket に接続**ダイアログボックスを表示するには、フロントパ

ネルオブジェクトを右クリックし、ショートカットメニューから**データ操作→ DataSocket に接続**を選択します。ウェブブラウザで URL を指定するのと同様に、URL を指定してデータのパブリッシュ（書き込み）やサブスクライブ（読み取り）を行います。

たとえば、フロントパネル上の温度表示器のデータをウェブ上の他のコンピュータと共有する場合、**DataSocket に接続**ダイアログボックスで URL を指定することによって温度計のデータをパブリッシュします。他のコンピュータ上のユーザは自分のフロントパネル上に温度計を配置し、**DataSocket に接続**ダイアログボックスで URL を選択することによって、そのデータをサブスクライブします。フロントパネル上での DataSocket テクノロジの使用方法の詳細については、この章の「フロントパネル上で DataSocket を使用する」のセクションを参照してください。

DataSocket テクノロジの詳細については、『Integrating the Internet into Your Measurement System』ホワイトペーパーを参照してください。このホワイトペーパーは、インストール CD の `manuals` ディレクトリまたはナショナルインスツルメンツのウェブサイト `ni.com` から PDF 形式で提供されています。

URL を指定する

URL では、`dstp`、`ftp`、`file` などの通信プロトコルを使用してデータが転送されます。URL で使用するプロトコルは、パブリッシュするデータのタイプとネットワークがどのように構成されているかによって異なります。

DataSocket を使用してデータをパブリッシュしたりデータをサブスクライブする場合は、以下のプロトコルを使用できます。

- **DataSocket 転送プロトコル（`dstp`）**：DataSocket 接続のためのネイティブプロトコルです。このプロトコルを使用する場合は、VI は DataSocket Server と通信します。データには名前の付いたタグを割り当てる必要があり、これは URL に追加されます。DataSocket 接続では名前付きのタグを使用して、DataSocket サーバ上の特定のデータ項目をアドレス指定します。このプロトコルを使用するには、DataSocket Server を実行する必要があります。
- **（Windows）OLE for Process Control （`opc`）**：自動製造工程によって生成されるデータにようなリアルタイムの生産データを共有するように特別に設計されています。このプロトコルを使用するには、OPC サーバを実行する必要があります。

- **(Windows)** logos：ネットワークとローカルコンピュータ間のデータの転送に使用されるナショナルインスツルメンツ社独自のテクノロジです。
- **ファイル転送プロトコル** (ftp)：このプロトコルを使用すると、データの読み取り元のファイルを指定できます。

メモ　DataSocket を使用して FTP サイトからテキストファイル読み取るには、[text] を DataSocket の URL の末尾に追加します。

- file：このプロトコルを使用すると、データが含まれているローカルファイルまたはネットワークファイルをリンクできます。

表 18-1 は、各プロトコル URL の例を示しています。

表 18-1　DataSocket URL の例

URL	例
dstp	dstp://servername.com/numeric、numeric はデータの名前付きのタグです。
opc	opc:¥National Instruments.OPCTest¥item1 opc:¥¥machine¥National Instruments.OPCModbus¥Modbus Demo Box.4:0 opc:¥¥machine¥National Instruments.OPCModbus¥Modbus Demo Box.4:0?updaterate=100&deadband=0.7
logos	logos://computer_name/process/data_item_name
ftp	ftp://ftp.ni.com/datasocket/ping.wav
file	file:ping.wav file:c:¥mydata¥ping.wav file:¥¥machine¥mydata¥ping.wav

ライブデータを共有するには、dstp、opc、および logos の URL を使用します。これは、これらのプロトコルがリモートおよびローカルの制御器や表示器を更新できるためです。ファイルからデータを読み取るには、ftp および file の URL を使用します。これは、これらのプロトコルがリモートおよびローカルの制御器や表示器を更新できないからです。

DataSocket 接続の使用例については、examples¥comm¥datasktx.llb を参照してください。

DataSocket でサポートされているデータ形式

DataSocket を使用すると、以下のデータをパブリッシュしたりサブスクライブしたりすることができます。

- **未処理テキスト**：文字列を文字列表示器に渡す場合は未処理テキストを使用します。
- **タブ付きテキスト**：スプレッドシート内でデータを配列として使用します。LabVIEW によってタブ付きデータがデータ配列に変換されます。
- `.wav` **データ**：VI または関数にサウンドをパブリッシュする場合は、`.wav` データを使用します。
- **バリアントデータ**：ナショナルインスツルメンツの Measurement Studio ActiveX コントロールなど他のアプリケーションからデータをサブスクライブするにはバリアントデータを使用します。

フロントパネル上で DataSocket を使用する

フロントパネルオブジェクト内のデータをパブリッシュしたりサブスクライブするには、フロントパネルの DataSocket 接続を使用します。フロントパネルオブジェクトのデータを他のユーザと共有することを、データをパブリッシュするといいます。パブリッシュされているデータを取り出してフロントパネル上に表示することを、データをサブスクライブするといいます。

DataSocket 接続は、ブロックダイアグラムでプログラミングをしないで直接フロントパネルから DataSocket 接続を使用できる点で、ウェブサーバや ActiveX 接続とは異なります。各フロントパネル制御器や表示器では、それぞれの DataSocket 接続を介してデータのパブリッシュやサブスクライブが可能です。フロントパネルの DataSocket 接続ではデータのみパブリッシュし、フロントパネル制御器のグラフィックはパブリッシュしません。したがって、DataSocket 接続を介してサブスクライブする VI は独自のデータ操作を実行できます。

フロントパネル上で制御器の値を直接設定してからデータをパブリッシュしたり、ブロックダイアグラムを作成して VI や関数の出力を表示器に配線し、その表示器からデータをパブリッシュすることができます。制御器および表示器とともに DataSocket 接続を使用する通常のシナリオは以下のとおりです。

- 他のユーザが制御器や表示器を介してサブスクライブできるように、制御器を操作してデータをパブリッシュするには、フロントパネル制御器から値をパブリッシュします。たとえば、フロントパネルに温度を調整するノブを配置すると、他のコンピュータのユーザはそのデータをサブスクライブしてサブ VI や関数に配線された制御器で使用したり、データを表示器に表示できます。

- 他のユーザがデータをサブスクライブして自分のフロントパネルにある制御器または表示器にパブリッシュしたり、サブ VI や関数の入力に配線されている制御器で結果をデータとして使用できるようにするには、フロントパネル表示器に表示されている値をパブリッシュします。たとえば、平均温度を計算してフロントパネルにある温度計に平均温度を表示する VI は、温度データをパブリッシュできます。
- 自分の VI のフロントパネル表示器にデータを表示するには、他の VI のフロントパネルにある制御器や表示器に表示されている値をサブスクライブします。制御器でデータをサブスクライブする場合、サブ VI や関数の入力に制御器を配線すると VI 内でそのデータを使用できます。
- 他のユーザの VI のフロントパネルから自分の VI のフロントパネルにある制御器を操作できるようにするには、自分の VI のフロントパネル制御器をパブリッシュおよびサブスクライブできるようにします。VI を実行すると、VI のフロントパネル制御器によって、DataSocket 接続を介して他の VI またはアプリケーションからパブリッシュされる現在の値が自分の VI のフロントパネル制御器に取り出されます。他のユーザがフロントパネルにある制御器の値を変更すると、DataSocket 接続によって自分の VI のフロントパネル制御器に新しい値がパブリッシュされます。自分のフロントパネル制御器の値を操作すると、自分の VI が他のユーザのフロントパネルに値をパブリッシュします。

データをサブスクライブしているフロントパネルオブジェクトは、データをパブリッシュするオブジェクトと同じものである必要はありません。ただし、フロントパネルオブジェクトのデータタイプは同じである必要があります。数値データの場合は、データタイプが異なると強制的に変換されます。たとえば、VI でタイプが整数のデジタル表示器を用意して、別の VI の温度計が生成した浮動小数点数のデータを読み取ることができますが、この際、データタイプは整数に変換されます。

フロントパネルの DataSocket 接続の主な目的はライブデータの共有です。ローカルファイル、FTP サーバ、または Web サーバでデータを読み取るには、「DataSocket 読み取り（DataSocket Read）」関数、ファイル I/O VI および関数、またはアプリケーション制御 VI および関数を使用します。

ブロックダイアグラムでライブデータの読み書きを行う

ブロックダイアグラムから、DataSocket 関数を使用してプログラムでデータの読み取りまたは書き込みができます。

DataSocket 接続を介してプログラムでライブデータを書き込むには、「DataSocket 書き込み（DataSocket Write ）」関数を使用します。図 18-1 は、数値を書き込む方法を示しています。

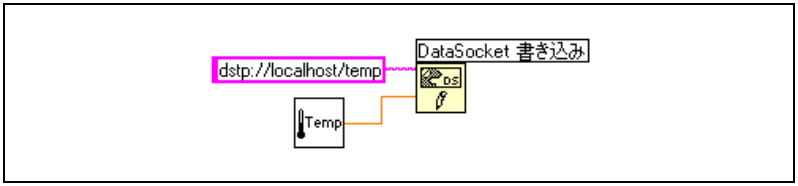


図 18-1　「DataSocket 書き込み」を使用してデータをパブリッシュする

「DataSocket 書き込み」関数は多形性なので、ほとんどのデータタイプを**データ**入力に配線できます。多形性 VI および関数の詳細については、第 5 章「ブロックダイアグラムを作成する」の「多形性 VI および関数」のセクションを参照してください。

DataSocket 接続からプログラムでライブデータを読み取るには、「DataSocket 読み取り（DataSocket Read)」関数を使用します。図 18-2 は、データを読み取って倍精度浮動小数点数に変換する方法を示しています。

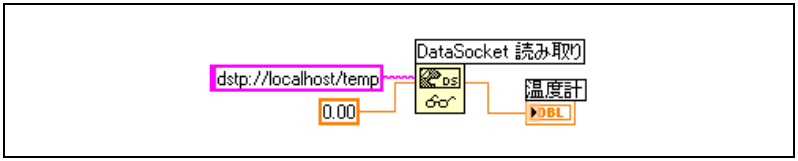


図 18-2　「DataSocket 読み取り (DataSocket Read)」を使用して 1 つの値を読み取る

「DataSocket 読み取り」の**タイプ**入力に制御器または定数を配線して、ライブデータを特定のデータタイプに変換します。タイプを指定しないと、「DataSocket 読み取り」の**データ**出力はバリアントデータを返します。このバリアントデータは、「バリアントからデータに変換（Variant to Data)」関数で操作する必要があります。

DataSocket 接続をプログラムで開閉する

「DataSocket を開く（DataSocket Open)」関数と「DataSocket を閉じる（DataSocket Close)」関数を使用して、DataSocket 接続が開閉する際の制御をします。「DataSocket を開く」関数を使用して DataSocket 接続を開く場合、以下の条件のうちの 1 つを満たすまで接続は開いたままです。「DataSocket を閉じる」関数を使用して DataSocket 接続をはっきりと閉じる場合は、VI を閉じるか、または VI で実行を停止します。「DataSocket を開く」関数の **URL** 入力は、DataSocket の URL しか受け入れません。「DataSocket を開く」関数は、「DataSocket 読み取り」関数および「DataSocket 書き込み（DataSocket Write)」関数の **URL** 入力として使用できる DataSocket 接続 Refnum を返します。

DataSocket データをバッファし処理する

DataSocket 転送プロトコル（dstp）を使用する場合、デフォルトでは、DataSocket Server は最新の値のみをサブスクライバにパブリッシュします。あるクライアントが他のクライアントが値を読み取る前に値をパブリッシュした場合、新しい値が、古い未処理の値をクライアントが読み取る前に上書きします。この未処理データの損失は、サーバまたはクライアントで発生します。データの損失は、DataSocket データをサブスクライブしている状態でサーバにパブリッシュされた最も新しい値のみを受け取る場合には、問題になりません。ただし、サーバにパブリッシュされたすべての値を受け取りたい場合は、クライアント上でデータをバッファ処理する必要があります。

メモ　また、クライアント側のバッファ処理は、opc、logos、および file などの他のプロトコルにも適用されます。dstp バッファリングを使用する場合は、DataSocket サーバマネージャを使用してサーバ側のバッファ処理を構成する必要があります。サーバ側のバッファ処理の詳細については、『DataSocket ヘルプ』を参照してください。

dstp バッファ処理では、データの配信は保証されません。サーバまたはクライアントでバッファ中のデータがバッファサイズを超える場合、バッファは新しい値の代わりに古い値を破棄します。データストリーム内で破棄された値を検出するには、「バリアント属性設定（Set Variant Attribute）」関数にパブリッシュされたデータを配線して、パブリッシュしている側のそれぞれの値を固有に識別し、サブスクライバ内で破棄されたシーケンス ID をチェックします。

「DataSocket を開く（DataSocket Open）」関数の**モード**入力を、**BufferedRead** または **BufferedReadWrite** に設定し、「プロパティノード（Property Node）」を使用して DataSocket プロパティを FIFO バッファのサイズに設定します。そうすることで、値が変わるたびに上書きするのではなく、バッファ内でクライアントが受け取る値を格納することを確認できます。

メモ　DataSocket プロパティを使用して FIFO バッファのサイズに設定する場合、「DataSocket を開く（DataSocket Open）」関数の**モード**入力を **BufferedRead** または **BufferedReadWrite** に設定する必要があります。それ以外の場合は、サーバにある項目は接続に対しバッファ処理されません。

図 18-3 は、DataSocket バッファ処理を使用しています。

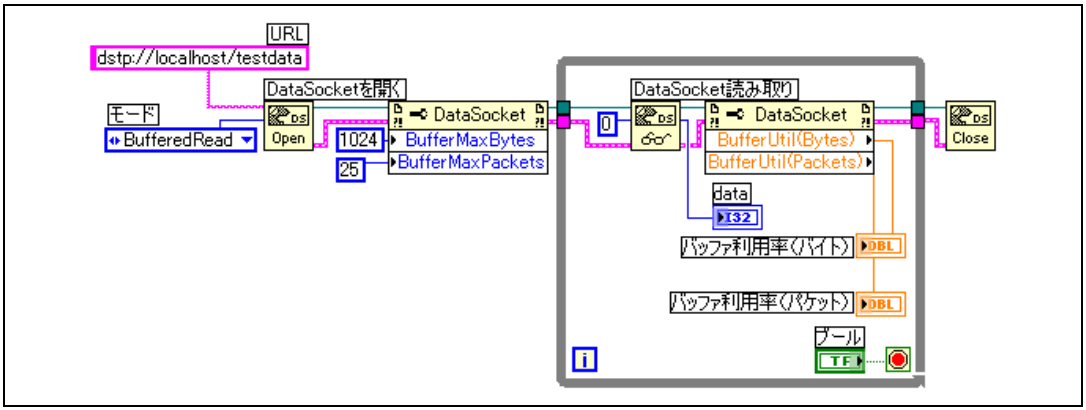


図 18-3　DataSocket バッファ処理

メモ　バッファリングは、「DataSocket 読み取り（DataSocket Read）」関数を使用してサーバがパブリッシュしたデータをサブスクライブする場合のみに適用されます。バッファリングは、フロントパネルの DataSocket 接続を使用してデータをサブスクライブした場合は有効ではありません。

診断を報告する

「バッファ利用率（バイト）」プロパティまたは「バッファ利用率（パケット）」プロパティを使用して、指定したバッファの診断情報をリクエストします。これらのプロパティを使用して、クライアント上の使用中バッファの割合（%）をチェックし、現在のバッファのサイズが十分であるかどうかを確認します。これらのプロパティのいずれかの値がバッファの最大値に近い値になっている場合、バッファサイズを増やして、サーバがパブリッシュするすべての値を受け取るようにします。

DataSocket クライアントのバッファサイズの指定の詳細については、『LabVIEW ヘルプ』を参照してください。

DataSocket とバリアントデータ

他のアプリケーションからデータをサブスクライブする場合など、プログラムでデータを読み取る VI や他のアプリケーションがデータを元のデータタイプに戻すことができないことがあります。また、場合によっては、データタイプで許容されないタイムスタンプや警告などの属性をデータに追加する必要もあります。

このような場合は、「バリアントへ変換（To Variant）」関数を使用して、DataSocket 接続に書き込むデータをプログラムでバリアントデータに変換します。図 18-4 は、温度の読み取り値を連続して集録し、そのデータをバリアントデータに変換し、タイムスタンプを属性としてデータに追加するブロックダイアグラムを示しています。

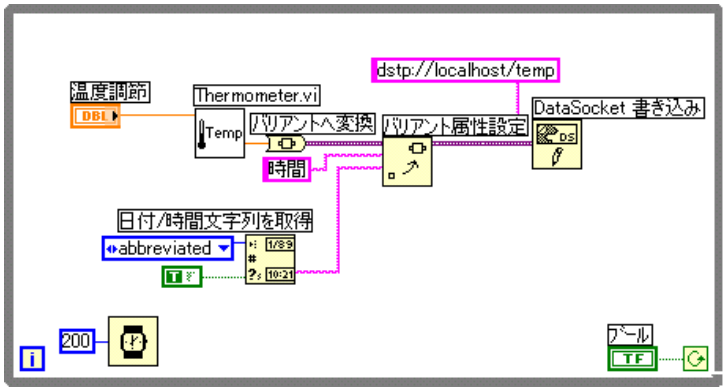


図 18-4　ライブ温度データをバリアントデータに変換する

他の VI がライブデータを読み取る場合、その VI はバリアントデータを操作可能なデータタイプに変換する必要があります。図 18-5 は、DataSocket 接続から温度データを連続して読み取り、バリアントデータを温度読み取り値に変換し、各読み取り値に対応するタイムスタンプ属性を取得して、フロントパネルに温度とタイムスタンプを表示するブロックダイアグラムを示しています。

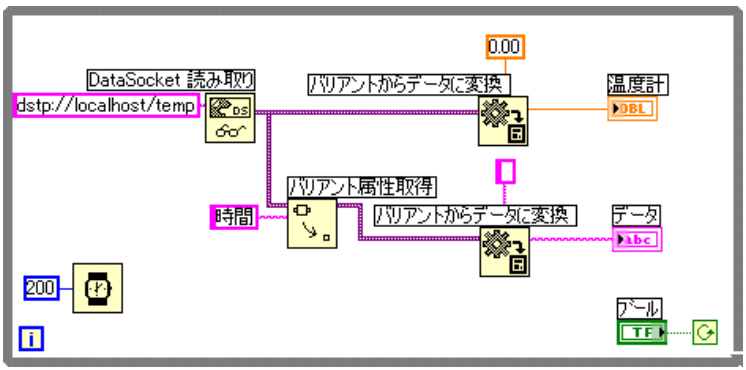


図 18-5　ライブバリアントデータを温度データに変換する

ウェブ上に VI をパブリッシュする

HTML ドキュメントを作成したり、フロントパネルの画像をウェブ上にパブリッシュしたり、VI をウェブページに埋め込むには、LabVIEW ウェブサーバを使用します。パブリッシュされているフロントパネルへのブラウザアクセスを制御したり、ウェブ上に表示する VI を構成することができます。

メモ　ウェブ上の VI を制御したり、ウェブ上にパブリッシュする VI にセキュリティ機能を追加するには、LabVIEW エンタープライズコネクティビティツールセットを使用します。このツールセットの詳細については、ナショナルインスツルメンツのウェブサイト `ni.com/jp` を参照してください。

ウェブサーバのオプション

ツール→オプションを選択し、一番上のプルダウンメニューから**ウェブサーバ**項目のいずれかを選択して、以下のオプションを設定します。

- ルートディレクトリおよびログファイルを設定する。
- ウェブサーバを使用可能にする。
- ブラウザのによる VI フロントパネルへのアクセスを制御する。
- どの VI のフロントパネルをウェブ上に表示するかを構成する。

ウェブ上で VI をパブリッシュする前に、**オプション**ダイアログボックスの**ウェブサーバ：構成**ページで、ウェブサーバを使用可能にする必要があります。この章の「HTML ドキュメントを作成する」のセクションで説明するウェブパブリッシュツールを選択してウェブサーバを使用可能にすることもできます。VI をパブリッシュする前に VI をメモリにロードしておく必要があります。

HTML ドキュメントを作成する

ツール→ウェブパブリッシュツールを選択してウェブパブリッシュツールを使用することにより、以下のタスクを実行できます。

- HTML ドキュメントを作成する。
- HTML ドキュメントにフロントパネルの静止画像または動画を組み込む。現在、動画をサポートしているのは Netscape ブラウザのみです。
- クライアントがリモートで表示および制御できる VI を組み込む。
- 組み込まれた VI フロントパネル画像の上下にテキストを追加する。
- 画像または組み込まれた VI の周りに枠を付ける。
- ドキュメントをプレビューする。

- ディスクにドキュメントを保存する。
- ウェブサーバを使用可能にして、ウェブ上に HTML ドキュメントとフロントパネル画像をパブリッシュする。

フロントパネル画像をパブリッシュする

現在メモリ内にある VI のフロントパネルの静止画像を返すには、ウェブブラウザまたは HTML ドキュメントで `.snap` URL を使用します。URL のクエリパラメータによって VI 名と画像の属性を指定します。たとえば、`http://`*`web.server.address`*`/.snap?`*`VIName.vi` というように指定します。ここで *`VIName.vi`* は表示する VI の名前です。

現在メモリ内にある VI のフロントパネルの動画を返すには、`.monitor` URL を使用します。URL のクエリパラメータに VI 名、動画の属性、および画像の属性を指定します。たとえば、`http://`*`web.server.address`*`/.monitor?`*`VIName.vi` というように指定します。ここで *`VIName.vi`* は表示する VI の名前です。

フロントパネルの画像形式

ウェブサーバでは、フロントパネルの画像を JPEG および PNG の画像形式で生成できます。

JPEG 形式はグラフィックの圧縮率が高い反面、グラフィックの細部が損なわれることがあります。この形式は写真に最も適しています。線画や、フロントパネル、ブロックダイアグラムなどの場合、JPEG 形式で圧縮するとグラフィックがぼやけたり、色むらが発生することがあります。PNG 形式でも、JPEG 形式ほどではありませんが、高圧縮率でグラフィックが圧縮されます。ただし、PNG 圧縮では細部が損なわれません。

フロントパネルをリモートで参照および制御する

LabVIEW の標準ウェブサーバに接続することによって、LabVIEW 内部から、またはウェブブラウザから VI フロントパネルをリモートで参照したり制御したりすることができます。フロントパネルをクライアントからリモートで開く場合、ウェブサーバはフロントパネルをクライアントに送信しますが、ブロックダイアグラムとすべてのサブ VI はサーバコンピュータに残ります。ブロックダイアグラムがサーバ上で実行していることを除いて、VI をクライアント上で実行している場合と同様にフロントパネルと対話することができます。この機能を使用して、フロントパネル全体をパブリッシュしたり、リモートアプリケーションを安全で簡単に、しかも迅速に制御することができます。

メモ　VI 全体を制御するには、LabVIEW ウェブサーバを使用してください。DataSocket サーバを使用して、VI の 1 つのフロントパネル制御器に対してデータを読み書きします。DataSocket サーバの使用方法の詳細については、この章の「DataSocket テクノロジを使用する」のセクションを参照してください。

クライアント用のサーバを構成する

クライアントが LabVIEW またはウェブブラウザを使用してリモートでフロントパネルを参照および制御するためには、サーバコンピュータのユーザがまずサーバを構成する必要があります。**ツール→オプション**を選択して、上部のプルダウンメニューから**ウェブサーバ**ページを選択し、ウェブサーバを構成します。これらのページを使用して、サーバに対するブラウザのアクセスを制御し、リモートで参照可能なフロントパネルを指定します。また、これらのページを使用して、複数のクライアントが VI の制御を待機している場合に、特定のリモートクライアントが VI を制御できるタイムリミットを設定することもできます。

ウェブサーバを使用すると、複数のクライアントが 1 つのフロントパネルに同時に接続できますが、フロントパネルを制御できるのは一度に 1 クライアントのみです。サーバコンピュータのユーザは、いつでも VI を制御することができます。コントローラがフロントパネル上の値を変更すると、すべてのクライアントのフロントパネルに変更が適用されます。ただし、クライアントのフロントパネルにはすべての変更は反映されません。通常、クライアントのフロントパネルには、フロントパネルオブジェクトのディスプレイに加えられた変更は反映されず、フロントパネルオブジェクトの実際の値に加えられた変更が反映されます。たとえば、コントローラがチャートのマッピングモードまたはスケールのマーカ間隔を変更したり、コントローラがチャートのスクロールバーを表示 / 非表示にすると、これらの変更はコントローラのフロントパネルにのみ反映されます。

リモートパネルライセンス

サーバに接続する可能性のある複数のクライアントをサポートするには、ライセンスを構成する必要があります。

メモ　**(Windows 9x)** LabVIEW 開発システムとアプリケーションビルダには、1 つのクライアントがリモートでフロントパネルを表示および制御することができるリモートパネルライセンスが含まれています。LabVIEW プロフェッショナル開発システムには、5 つのクライアントがフロントパネルを表示および制御することができるリモートパネルライセンスが含まれています。Windows 9x では、リモートパネルライセンスをアップグレードして、これ以上の数のクライアントをサポートするようにすることはできません。
(Windows 2000/NT/XP、Mac OS、および UNIX) LabVIEW 開発システムとアプリケーションビルダには、1 つのクライアントがフロントパネルをリモートで表示および制御することができるリモートパネルライセンスが含まれていま

す。LabVIEW プロフェッショナル開発システムには、5 つのクライアントがフロントパネルを表示および制御することができるリモートパネルライセンスが含まれています。リモートパネルライセンスをアップグレードして、より多くのクライアントをサポートするようにすることもできます。

LabVIEW またはウェブブラウザからフロントパネルを参照および制御する

クライアントは、LabVIEW またはウェブブラウザからフロントパネルをリモートで参照および制御することができます。

メモ　フロントパネルをリモートで参照および制御するには、まず参照および制御する VI のあるサーバコンピュータでウェブサーバを有効にする必要があります。

フロントパネルを LabVIEW で参照および制御する

LabVIEW をクライアントとして使用してリモートフロントパネルを参照するには、新規 VI を開いて**操作→リモートパネルに接続**を選択し、**リモートパネルに接続**ダイアログボックスを表示します。このダイアログボックスを使用して、サーバのインターネットアドレスと参照する VI を指定します。デフォルトでは、リモート VI のフロントパネルはオブザーバモードになっています。VI をリクエストする際に、**リモートパネルに接続**ダイアログボックスの**制御をリクエスト**チェックボックスにチェックを入れて、制御器をリクエストすることができます。VI がコンピュータに表示されたら、フロントパネルの任意の部分を右クリックして、ショートカットメニューから**制御をリクエスト**を選択することもできます。また、フロントパネルウィンドウの下部にあるステータスバーをクリックしてもこのメニューにアクセスすることができます。他のクライアントが制御していなければ、フロントパネルが制御可能であることを示すメッセージが表示されます。他のクライアントが VI を制御している場合には、他のクライアントが制御を停止するか、制御がタイムアウトになるまで、サーバはその要求を待ち行列に入れます。サーバコンピュータのユーザのみが**ツール→リモートパネル接続マネージャ**を選択してクライアントの待ち行列リストをモニタすることができます。

リモートコンピュータ上で実行している VI によって生成されるデータを保存する場合には、リモートフロントパネルではなく DataSocket または TCP を使用します。DataSocket の使用方法の詳細については、この章の「DataSocket テクノロジを使用する」のセクションを参照してください。TCP の使用方法の詳細については、この章の「TCP と UDP」のセクションを参照してください。

クライアントから参照および制御するすべての VI がメモリ内にある必要があります。要求された VI がメモリ内にある場合、サーバは VI のフロントパネルを要求側のクライアントに送信します。VI がメモリにない場合には、**リモートパネルに接続**ダイアログボックスの**接続状況**のセクションにエラーメッセージが表示されます。

ウェブブラウザからフロントパネルを参照および制御する

ウェブブラウザを使用して、LabVIEW がインストールされていないクライアントからリモートでフロントパネルを参照および制御するには、LabVIEW ランタイムエンジンをインストールする必要があります。LabVIEW ランタイムエンジンには、ブラウザのプラグインディレクトリにインストールされる LabVIEW ブラウザプラグインパッケージが含まれます。LabVIEW の CD には、LabVIEW ランタイムエンジンのインストーラが含まれています。

クライアントは LabVIEW ランタイムエンジンをインストールし、サーバコンピュータのユーザはクライアントに参照および制御させる VI を参照する `<OBJECT>` および `<EMBED>` タブを含む HTML ファイルを作成します。このタグには、VI に対する URL リファレンスと VI を LabVIEW ブラウザプラグインに渡すようにウェブブラウザに指示する情報が含まれています。クライアントは、ウェブブラウザウィンドウの上部のアドレスまたは URL フィールドにウェブサーバのウェブアドレスを入力して、ウェブサーバに移動します。プラグインは、ウェブブラウザウィンドウにフロントパネルを表示し、クライアントがリモートフロントパネルと通信できるようにウェブサーバと通信します。クライアントは、ウェブブラウザのリモートフロントパネルウィンドウの下部の**制御をリクエスト**を選択するか、またはフロントパネル内で右クリックしてショートカットメニューから**制御をリクエスト**を選択して制御を要求します。

現在、他のクライアントおよび同じネットワーク接続上の他のブラウザウィンドウが制御していなければ、フロントパネルが制御可能であることを示すメッセージが表示されます。他のクライアントが VI を制御している場合には、他のクライアントが制御を停止するか、制御がタイムアウトになるまで、サーバはその要求を待ち行列に入れます。サーバコンピュータのユーザのみが**ツール→リモートパネル接続マネージャ**を選択してクライアントの待ち行列リストをモニタすることができます。

メモ　ナショナルインスツルメンツでは、ウェブブラウザでフロントパネルを参照および制御する場合には、Netscape 4.7 以降、または Internet Explorer 5.5 以降をご使用になることをお勧めします。

リモートフロントパネルの参照および制御でサポートされていない機能

ウェブブラウザの制約により、フロントパネルの次元や位置の操作を試みるユーザインタフェースアプリケーションは、フロントパネルがウェブページの一部として表示される場合には正しく動作しません。ウェブサーバと LabVIEW ブラウザのプラグインは、複雑なユーザインタフェースアプリケーションとの整合性を保とうとしますが、特にダイアログボックスやサブ VI ウィンドウを表示するアプリケーションについては、ウェブブラウザで使用した場合に正常に動作しない可能性があります。ナショナルインスツルメンツでは、このようなアプリケーションをウェブブラウザでの使用のためにエクスポートしないことをお勧めします。

While ループを使用しているのに「待機（Wait）」関数がない VI はエクスポートしないでください。このような VI は、バックグラウンドタスクの実行時間が制限されるため、リモートで参照または制御する場合にフロントパネルが応答しない可能性があります。

また、VI によっては、リモートコンピュータから実行したときとローカルで実行したときで動作が異なる場合もあります。フロントパネルに組み込まれた ActiveX コントロールは、LabVIEW とほぼ無関係に描画および操作するため、リモートクライアントには表示されません。VI が標準のファイルダイアログボックスを表示した場合、リモートでファイルシステムを検索することができないために、コントローラはエラーを受け取ります。また、パス制御器の参照ボタンはリモートパネルでは無効状態になります。

フロントパネルをリモートで参照しているクライアントは、接続しているフロントパネルが作成したアプリケーションのものかどうかによって異なる動作をする可能性があります。特に、作成したアプリケーションのフロントパネルの場合、クライアントがフロントパネルに接続する前にフロントパネルに加えられたプログラム的な変更はクライアントコンピュータには反映されません。たとえば、クライアントがそのフロントパネルに接続する前にプロパティノードによって制御器のキャプションが変更された場合、クライアントは変更されたキャプションではなく変更前のキャプションを参照することになります。

VI サーバや、呼び出し時にフロントパネルを表示するように構成されているサブ VI を使用して、ダイナミックに開いたり実行したりしている VI のフロントパネルをリモートで表示することができるのは、コントローラのみです。VI を制御していないクライアントはフロントパネルを表示することはできません。

フロントパネル制御器のプロパティをポーリングすることによって特定のユーザインタフェース効果を実現するブロックダイアグラムの場合、リモートコンピュータから VI を制御するとパフォーマンスが低下する可能性があります。「フロントパネルアクティビティを待機（Wait for Front Panel Activity）」関数を使用すると、このような VI のパフォーマンスを向上することができます。

VI からデータを電子メールで送信する

SMTP E-mail VI を使用して、データやファイルの添付を含む電子メールを送信ができます。この電子メール送信には、簡易メール転送プロトコル（SMTP）が使用されています。LabVIEW は SMTP の認証をサポートしていません。SMTP E-mail VI を使用して情報を受け取ることはできません。SMTP E-mail VI は、多目的インターネットメール拡張仕様（MIME）形式を使用してメッセージを暗号化します。この形式の場合、バイナリデータファイルを含む複数のドキュメントを電子メールで送信できます。ドキュメントに使用する文字セットなど、各添付ファイルのプロパティを記述することもできます。VI からデータを電子メールで送信する詳細については、『LabVIEW ヘルプ』を参照してください。

メモ　SMTP E-mail VI は、開発システム、プロフェッショナル開発システムのみでご利用いただけます。

受信者の電子メールアドレスの他に、SMTP サーバのウェブアドレスが必要です。使用する各 SMTP E-mail VI に**メールサーバ**を配線することによって、メールサーバを指定します。**メールサーバ**は、SMTP E-mail VI を実行しているコンピュータからサービスをリクエストできる外部サーバコンピュータの IP アドレスまたはホスト名である必要があります。有効なメールサーバを指定すると、SMTP E-mail VI はサーバへの接続を開き、電子メールの受信者および内容を記述したサーバコマンドを送信します。サーバは、各受信者にメッセージを送信するか、他の SMTP サーバにメッセージを転送します。SMTP E-mail VI のサンプルについては、`examples¥comm¥smtpex.llb` を参照してください。

メモ　SMTP E-mail VI は、日本語のようなマルチバイト文字をサポートしていません。

文字セットを選択する

SMTP E-mail VI の**文字セット**入力パラメータは、電子メールのメッセージまたは添付ファイルに使用される文字セットを指定します。文字セットは、文字と文字コード間のマッピングを記述したものです。

文字は、書き言葉（文字、数字、句読点、言語によっては単語全体）の基本単位です。大文字やアクセント記号などを使用して文字を変更すると、その文字は別の文字になります。たとえば、O、o、Ö、および Ô は異なる文字です。

文字コードは文字を表す数字です。コンピュータは数字のみしか解釈できないため、文字を解釈するには、文字と数字が関連付けられている必要があります。

文字セットは、文字とコンピュータ上で文字を表す数字の関係です。たとえば、ASCII 文字セットでは、A、B、C の文字コードは、それぞれ 65、66、67 となります。

US-ASCII 文字セット

電子メールで通常使用される文字セットは、US-ASCII または ASCII です。多くの電子メールアプリケーションは、この文字セットをデフォルトで使用し、他の文字セットとは使用しません。ASCII 文字セットは、英語で使用される文字とほとんどの句読点を表すもので、全部で 128 文字あります。他のほとんどの文字セットは ASCII を拡張したものです。

多くの言語が ASCII に存在しない文字を必要とするため、ASCII 文字セットを使用するだけでは十分ではない場合があります。たとえば、ASCII を使用してドイツ語の Müller を書くことはできません。ü が ASCII の文字セットで定義されていないためです。

ISO Latin-1 文字セット

多くの言語で ASCII にない文字が必要なため、それらの言語を使用する国で新しい文字セットが作成されました。通常、これらの文字セットは、最初の 128 文字コードが ASCII で、次の 128 文字コードがその言語で必要な文字を定義します。これらの文字セットの中には、異なる文字モードを使用して同じ文字を表すものもあります。これは、ある文字セットが他の文字セットで書かれたテキストを表示する場合に問題を引き起こす可能性があります。この問題を解決するには、標準文字セットを使用します。広く使用されている標準文字セットは、ISO Latin-1 で、ISO-8859-1 としても知られています。この文字セットは、ほとんどの西ヨーロッパ言語、そしてそれらの言語を処理するほとんどの電子メールアプリケーションで使用されています。

Mac OS 文字セット

アップルコンピュータ社（Apple Computer, Inc.）は、Latin-1 が定義される以前に独自の拡張文字セットを開発しました。Mac OS の文字セットは、ASCII に基づいていますが、上段の 128 文字コードのセットは ISO Latin-1 でなく異なる文字セットを使用します。そのため、Mac OS 文字セットで書かれたアクセント記号が含まれる電子メールは、ISO Latin-1 テキストを処理する電子メールアプリケーションでは正しく表示されません。この問題を解決するために、Mac OS 電子メールアプリケーションは、メールを送信する前にテキストを ISO Latin-1 に変換します。他の Mac OS 電子メールアプリケーションが ISO Latin-1 文字セットの使用を指定されているテキストを受信すると、テキストを Mac OS 文字セットに変換します。

字訳

字訳とは、文字を他の文字セットにマッピングすることです。電子メールを送信する際、電子メールを他の文字セットで対応する文字に変換する必要がある場合に、字訳を行います。SMTP E-mail VI を使用して、テキストを送信する前に他の文字セットにマッピングする文字セットを指定します。たとえば、標準 ASCII 文字を使用してメッセージを作成し、その文字セットを MacRoman と指定します。SMTP E-mail VI はテキストを字訳して、電子メールを `iso-8859-1`（ISO Latin-1）の文字セットで送信します。SMTP E-mail VI の**字訳**入力パラメータを使用して、VI が使用する字訳を指定します。字訳は、仮想文字セット、目標文字セット、そして字訳ファイルまたはマッピングファイルで定義されています。字訳ファイルは、ある文字が他の文字にマッピングするように指定します。

字訳ファイルは、256 のエントリを持つ 256 バイトのバイナリファイルです。ファイルの各エントリは、仮想文字セットの文字に対応し、目標文字セットの新しい文字コードを含んでいます。たとえば、`61` のコードを持つ文字 `a` を、`41` のコードを持つ文字 `A` にマッピングした場合、字訳ファイルの `61` の指標を持つエントリは `41` の値を含む必要があります。エントリに指標と同じ値が含まれている場合、字訳ファイルは仮想文字セットの文字を変更しません。たとえば、字訳ファイルの `61` の指標を持つエントリに `61` の値が含まれている場合、字訳ファイルは文字を修正しません。

字訳入力で字訳ファイルを目標文字セットとして指定する場合、マッピングは指定された順序で適用されます。たとえば、字訳エントリが (MacRoman iso-8859-1 `macroman.trl`, MacRomanUp MacRoman `asciiup.trl`) の場合、`macroman.trl` によって MacRoman 文字セットは iso-8859-1 に変更され、`macroman.trl` によって MacRomanUp は MacRoman に変更されます。`.trl` ファイルのサンプルについては、`vi.lib¥Utility¥SMTP` を参照してください。

低レベル通信アプリケーション

LabVIEW では、コンピュータ間の通信に使用できる低レベルプロトコルがいくつかサポートされています。

プロトコルはそれぞれ異なりますが、特にリモートアプリケーションのネットワーク位置を参照する方法が異なっています。各プロトコルは一般的に他のプロトコルと互換性がありません。たとえば、Mac OS と Windows の間で通信する場合、両方のプラットフォームで動作する TCP などのプロトコルを使用する必要があります。

TCP と UDP

転送制御プロトコル（Transmission Control Protocol：TCP）とユーザデータグラムプロトコル（User Datagram Protocol：UDP）は、LabVIEW によってサポートされているすべてのプラットフォームで使用可能です。TCP は信頼性のある接続ベースのプロトコルです。TCP にはエラー検出機能があり、データは重複することなく正しい順序で確実に届きます。これらの理由から、通常 TCP はネットワークアプリケーションに最適とされています。

UDP は TCP よりも高パフォーマンスで、接続の必要もありませんが、配信は保証されません。通常は、配信の保証が重要でない場合にアプリケーションで UDP を使用します。たとえば、あるアプリケーションで送信先に頻繁にデータを転送しているため、データの一部のセグメントが失われても問題にならない場合などです。LabVIEW アプリケーションでの TCP および UDP の使用方法の詳細については、『Using LabVIEW with TCP/IP and UDP 』アプリケーションノートを参照してください。

「UDP マルチキャストオープン（UDP Multicast Open）」VI を「UDP 接続を開く（UDP Open）」関数の代わりに使用して、読み取り、書き込み、もしくはマルチキャスト IP アドレスへの UDP データの読み取りと書き込みが可能な接続を開きます。マルチキャスト IP アドレスは、マルチキャストグループを定義します。マルチキャスト IP アドレスは、`224.0.0.0` ～ `239.255.255.255` の範囲になります。クライアントがマルチキャストグループに参加する場合、クライアントはグループのマルチキャスト IP アドレスにサブスクライブします。クライアントがマルチキャストグループをサブスクライブすると、クライアントはマルチキャスト IP アドレスに送信するデータを受け取ります。UDP マルチキャスティングの使用方法の詳細については、『Using LabVIEW with TCP/IP and UDP 』アプリケーションノートを参照してください。

Apple Events および PPC Toolbox (Mac OS)

Apple Events は、最も一般的な Mac OS 専用の通信形式です。Apple Events はアクションを要求するメッセージを送信したり、他の Mac OS アプリケーションからの情報を返したりする場合に使用します。

プログラム間通信ツールボックス（Program-to-Program Communication Toolbox：PPC Toolbox）は、Mac OS アプリケーション間でデータを送受信する低レベルの形式です。PPC Toolbox は、情報の転送にかかるオーバヘッドが少ないため、Apple Events よりも高パフォーマンスです。ただし、PPC Toolbox は転送できる情報のタイプを定義していないため、多くのアプリケーションでサポートされていません。PPC Toolbox は PPC Toolbox をサポートしているアプリケーション間で大量の情報を送信する場合の最適な方法です。LabVIEW アプリケーションでの AppleEvents と PPC Toolbox の使用方法については、『Using Apple Events and the PPC Toolbox to Communicate with LabVIEW Applications on the Macintosh』アプリケーションノートを参照してください。

Pipe VI（UNIX）

UNIX の名前付きパイプに対して開閉および読み書きを行うには、パイプ VI を使用します。LabVIEW と関連性のないプロセス間の通信を行うには、名前付きパイプを使用します。

システムレベルのコマンドを実行する（Windows および UNIX）

他の Windows アプリケーションや、UNIX コマンドラインアプリケーションを VI 内部から実行したり起動するには、「システム実行（System Exec）」VI を使用します。「システム実行」VI を使用すると、起動するアプリケーションでサポートされているパラメータを含むシステムレベルのコマンドラインを実行できます。

19

Windows の接続性

.NET または ActiveX テクノロジを使用して、LabVIEW から他の Windows アプリケーションにアクセスできます。LabVIEW を .NET クライアントとして使用して、.NET サーバに関連付けられているオブジェクト、プロパティ、およびメソッドにアクセスできます。LabVIEW は .NET サーバではありません。.NET を介して、他のアプリケーションが直接 LabVIEW と通信することはできません。.NET が有効な VI を使用すると、Windows のサービスおよび API に接続できます。.NET Framework には、COM+ コンポーネント サービス、ASP Web 開発フレームワーク、および SOAP、WSDL、UDDI などの Web サービス プロトコルのサポートが用意されています。.NET Framework は .NET 環境のプログラミングの基礎です。これは、ウェブベースのアプリケーション、スマート クライアントアプリケーション、および XML Web サービスの作成、導入、および実行に使用されます。

LabVIEW などの Windows アプリケーションは、ActiveX オートメーションを使用することにより、他の Windows アプリケーションがアクセスできる公開オブジェクトのセット、コマンド、および関数を提供します。LabVIEW を ActiveX のクライアントとして使用すると、ActiveX が有効になっている他のアプリケーションに関連するオブジェクト、プロパティ、メソッド、およびイベントにアクセスできます。また、LabVIEW は ActiveX サーバとしても機能するため、他のアプリケーションから LabVIEW のオブジェクト、プロパティ、およびメソッドにアクセスできます。

.NET については、Microsoft の .NET テクノロジを参照してください。.NET Framework をインストールする必要があります。.NET およびその Framework のインストールの詳細については、MSDN（Microsoft Developer Network）ウェブサイトを参照してください。*ActiveX* とは、Microsoft の ActiveX テクノロジおよび OLE テクノロジのことです。ActiveX の詳細情報については、Microsoft Developer's Network Documentation の Kraig Brockschmidt 著、第 2 版『Inside OLE』、および Don Box 著、『Essential COM』を参照してください。

詳細については

.NET および ActiveX テクノロジの使用方法の詳細については、『LabVIEW ヘルプ』およびナショナルインスツルメンツのウェブサイト `ni.com` または `ni.com/jp` を参照してください。

.NET 環境

.NET 環境を構成するさまざまな要素の背景について、以下に説明します。この情報は .NET の理解に役立つものですが、LabVIEW の .NET コンポーネントを使用するにあたり、必ずしもこの情報の習得が必須というわけではありません。

- **共通言語ランタイム（CLR）**：言語統合、セキュリティの実施、メモリ、バグ処理、プロセス管理、およびスレッド管理などの、ランタイムサービスを管理する一連のライブラリです。CLR は .NET の基礎を形成し、すべてのプログラミング言語が生成する IL（Intermediate Language）を使用して、.NET と他の言語の間での通信を促進します。

  .NET とさまざまなプログラムとの通信を支援するために、CLR はプログラミング言語とオペレーティングシステムの境界を補うデータタイプシステムを提供します。開発に CLR を使用すると、メモリやスレッドの集合としてではなく、データタイプの集合としてシステムを表示します。CLR には、CLR IL メタデータ形式で情報を生成するコンパイラやリッカーが必要です。Win32 システムでは、すべてのプログラミング言語のコンパイラが、アセンブリコードではなく、CLR IL コードを生成しています。

- **クラスライブラリ**：入出力、文字列操作、セキュリティ管理、ネットワーク通信、スレッド管理、テキスト管理、ユーザインタフェースの設計上の特徴など、標準機能を提供する一連のクラスです。このクラスには Win32/COM システムが使用するのと同じ機能があります。.NET Framework では、ある .NET 言語で作成したクラスを別の .NET 言語で使用することができます。

- **アセンブリ**：DLL、OCX、または COM にあるコンポーネントの実行ファイルに似た導入の単位です。アセンブリは DLL および .NET CLR を使用して作成した実行ファイルです。アセンブリは、1 つのファイルからも、複数のファイルからも構成することができます。アセンブリには、アセンブリ名、バージョン情報、ローカル情報、パブリッシャのセキュリティ情報、アセンブリを作成したファイルのリスト、従属アセンブリのリスト、リソース、およびエクスポートしたデータタイプについては情報を含む明示リストがあります。1 つのファイルから成るアセンブリには、明示リストおよび必要なリソースを含む 1 つのファイルにすべてのデータが入っています。複数のファイルから成るアセンブリには、この他にビットマップ、アイコン、サウンドファイルなどの外部リソースが含まれていたりします。あるいは、このアセンブリには、1 つのファイルにコアコードが入っていて、別のファイルにはヘルプのライブラリが入っていることがあります。

  アセンブリはパブリックにもプライベートにもなります。.NET は、アプリケーションディレクトリとしてプライベートアセンブリが同じディレクトリにあること、また、パブリックアセンブリが GAC

（Global Assembly Cache）と呼ばれるシステムワイドグローバルキャッシュにあることが必要です。通常アプリケーションの開発では、そのアプリケーション言語で使用するプライベートアセンブリを作成します。また、開発時にはバージョンの制御も指定します。アセンブリ名は、明示リストを含むファイルから任意のファイル拡張子を差し引いたファイル名になります。

- **GAC（Global Assembly Cache）**：システムで使用可能なパブリックアセンブリのリストです。GAC は、COM で使用するレジストリに似ています。

.NET 関数およびノード

LabVIEW を .NET クライアントとして使用して、.NET サーバに関連付けられているオブジェクト、プロパティ、およびメソッドにアクセスします。

- **.NET** パレットにある「コンストラクタノード（Constructor Node）」を使用し、アセンブリから .NET のコンストラクタを選択して、実行するクラスのインスタンスを作成します。ブロックダイアグラムにこのノードを配置すると、LabVIEW では **.NET Constructor を選択**ダイアログボックスが表示されます。
- .NET クラスに関連付けられているプロパティを取得（読み取り）したり設定（書き込み）したりするには、**.NET** パレットにある「プロパティノード（Property Node）」を使用します。
- .NET クラスに関連付けられているメソッドを呼び出すには、**.NET** パレットにある「インボークノード（Invoke Node）」を使用します。
- 接続する必要がなくなり、.NET オブジェクトへのすべてのリファレンスを閉じるには、**.NET** パレットにある「リファレンスを閉じる（Close Reference）」関数を使用します。
- ベースクラスへの .NET リファレンスをアップキャストするには、**.NET** パレットにある「より一般的なクラスに変換（To More Generic Class）」関数を使用します。
- 導出クラスへの .NET リファレンスをダウンキャストするには、**.NET** パレットにある「より指定されたクラスに変換（To More Specific Class）」関数を使用します。

.NET クライアントとしての LabVIEW

.NET アセンブリに関連付けられたオブジェクトにアクセスすると、LabVIEW が .NET クライアントとして動作します。LabVIEW を .NET クライアントとして使用するには、以下のような主な 3 つの手順があります。

1. コンストラクタを使用して .NET オブジェクトを作成し、オブジェクトへのリファレンスを構築します。
2. 「プロパティノード」または「インボークノード」に .NET オブジェクトリファレンスを書き込み、プロパティまたはメソッドを選択します。
3. .NET オブジェクトリファレンスを閉じて、オブジェクトへの接続を終了します。

.NET オブジェクトにアクセスするには、ブロックダイアグラムの「コンストラクタノード」を使用して、必要な .NET オブジェクトを作成します。アセンブリからオブジェクトのクラスを選択するには、「コンストラクタノード」を使用します。ブロックダイアグラムに「コンストラクタノード」を配置すると、LabVIEW には GAC（Global Assembly Cache）内のすべてのパブリックアセンブリをリストした **.NET Constructor を選択**ダイアログボックスが表示されます。プライベートアセンブリを選択する場合は、ダイアログボックスの**ブラウズ**ボタンをクリックしてプライベートアセンブリのファイルシステムを参照します。.NET アセンブリは `.dll` および `.exe` のファイルタイプです。プライベートアセンブリを選択すると、次に **.NET Constructor を選択**ダイアログボックスを立ち上げた際に、そのアセンブリがそのダイアログボックスの**アセンブリ**プルダウンメニューに表示されます。

アセンブリとクラスを選択すると、そのクラスのコンストラクタが **.NET Constructor を選択**ダイアログボックスの**コンストラクタ**セクションに表示されます。ダイアログボックスを閉じるには、コンストラクタを選択して **OK** ボタンをクリックします。LabVIEW には「コンストラクタノード」で選択したクラス名が表示されます。

「コンストラクタノード」が初期化パラメータを指定してオブジェクトを作成できること以外は、このコンストラクタノードは ActiveX の「オートメーションオープン（Automation Open）」関数に似ています。「コンストラクタノード」から「プロパティノード」または「インボークノード」へ .NET サーバリファレンスを接続して、ショートカットメニューからプロパティまたはメソッドを選択できます。「リファレンスを閉じる（Close Reference）」関数を使用して、.NET オブジェクトへのリファレンスを閉じます。

.NET オブジェクトの作成の詳細については、『LabVIEW ヘルプ』を参照してください。

データタイプマッピング

LabVIEW は .NET プロパティ、メソッド、およびコンストラクタパラメータのデータタイプを LabVIEW のデータタイプに変換するので、LabVIEW でデータを読み取ったり解釈したりできます。.NET Refnum として変換できないデータタイプが LabVIEW に表示されます。以下の表は .NET データタイプおよび変換した LabVIEW データタイプを示します。

表 19-1　.NET および LabVIEW データタイプ

.NET タイプ	LabVIEW タイプ
System.Int32、System.UInt32、System.Int16、System.UInt16	I32, U32, I16, U16
System.String	abc
System.Boolean	TF
System.Byte、System.Char、System.UByte	I8, U8, U8
System.Single、System.Double、System.Decimal	SGL, DBL, DBL
System.Array	対応するタイプの配列として表示される
列挙	（スパース）
DateTime	
その他の .NET オブジェクト	

.NET アプリケーションを導入する

.NET コンポーネントを含む VI を作成して、LLB または DLL の実行ファイルにその VI を組み込むことができます。

実行ファイルを導入する

.NET オブジェクトを使用するアプリケーションを作成する場合、スタンドアロンアプリケーションのディレクトリに VI が使用するプライベートアセンブリをコピーして、ターゲットコンピュータに .NET Framework がインストールされていることを確認する必要があります。

VI を導入する

VI を導入する場合、VI が使用するプライベートアセンブリを最上位レベル VI のディレクトリにコピーする必要があります。VI で使用するすべてのアセンブリが最上位レベル VI の同じディレクトリ構造にあるか、また、ターゲットコンピュータに .NET Framework がインストールされているかを確認します。

DLL を導入する

LabVIEW で作成した DLL を導入する場合、その DLL を使用するアプリケーションのディレクトリに VI が使用するプライベートアセンブリをコピーして、ターゲットコンピュータに .NET Framework がインストールされているかを確認する必要があります。

.NET クライアントアプリケーションを構成する（上級）

.NET は構成ファイルを使用してアプリケーションへ管理機能を提供します。構成ファイルには XML コンテンツがあり、通常、`.config` 拡張子が付いています。LabVIEW で .NET クライアントアプリケーションを構成するには、最上位レベル VI またはスタンドアロンアプリケーション用の構成ファイルを提供します。構成ファイルに、`My App.vi.config` または `MyApp.exe.config` のように、`.config` 拡張子の付いたアプリケーションの名前をつけます。

ActiveX のオブジェクト、プロパティ、メソッド、およびイベント

ActiveX が有効になっているアプリケーションには、他のアプリケーションからアクセス可能な、公開されているプロパティやメソッドを持つオブジェクトが含まれています。オブジェクトには、ボタン、ウィンドウ、ピクチャ、ドキュメント、ダイアログボックスなどユーザに見えるものと、アプリケーションオブジェクトなどユーザに見えないものがあります。アプリケーションに関連付けられているオブジェクトにアクセスし、プロパティを設定するか、そのオブジェクトのメソッドを呼び出すことによって、アプリケーションにアクセスします。

オブジェクト、プロパティ、およびメソッドの詳細については、第 17 章「VI をプログラム的に制御する」の「アプリケーションと VI の設定値を操作する」のセクションを参照してください。

イベントとは、マウスをクリックする、キーを押す、メモリ容量不足になるなど、オブジェクト上で行う動作またはそれに伴った状況を意味します。オブジェクトにこれらのアクションが発生すると、オブジェクトはイベント固有のデータとともにイベントを送信し、ActiveX コンテナに警告します。

「レジスタイベントコールバックノード（Register Event Callback Node）」を使用して ActiveX イベントを処理する詳細については、この章の「ActiveX イベント」のセクションを参照してください。

ActiveX VI、関数、制御器、および表示器

ActiveX が有効になっている他のアプリケーションに関連付けられているオブジェクト、プロパティ、メソッド、およびイベントにアクセスするには、以下の VI、関数、制御器、および表示器を使用します。

- ActiveX オブジェクトへのリファレンスを作成するには、オートメーション Refnum 制御器を使用し ます。フロントパネル上にあるこの制御器を右クリックして、アクセスするタイプライブラリからオブジェクトを選択します。
- 「オートメーションオープン（Automation Open ）」関数を使用して、ActiveX オブジェクトを開きます。
- ActiveX コンテナを使用して、フロントパネル上の ActiveX オブジェクトにアクセスして表示します。この制御器を右クリックし、ショートカットメニューから **ActiveX オブジェクトを挿入**を選択して、アクセスするオブジェクトを選択します。
- 「プロパティノード」を使用して、ActiveX オブジェクトに関連付けられたプロパティを取得（読み取り）および設定（書き込み）します。
- 「インボークノード」を使用して、ActiveX オブジェクトに関連付けられているメソッドを呼び出します。
- 「レジスタイベントコールバックノード（Register Event Callback Node）」を使用して、ActiveX オブジェクトに発生するイベントを処理します。
- バリアント制御器および表示器を使用して、ActiveX コントロールにデータを渡したり、ActiveX コントロールからデータを受け取ります。バリアントデータの詳細については、第 5 章「ブロックダイアグラムを作成する」の「バリアントデータを処理する」のセクションを参照してください。

ActiveX クライアントとしての LabVIEW

ActiveX 有効の他のアプリケーションに関連付けられているオブジェクトに LabVIEW がアクセスすると、LabVIEW は ActiveX のクライアントとして動作します。LabVIEW を ActiveX のクライアントとして以下のように使用することができます。

- Microsoft Excel などのアプリケーションを開き、ドキュメントを作成して、そのドキュメントにデータを追加します。
- コンテナのフロントパネルに Microsoft Word ドキュメントや Excel スプレッドシートなどのドキュメントを埋め込みます。
- 他のアプリケーションで使用されているヘルプファイルを呼び出す **ヘルプ**ボタンなど他のオブジェクトをフロントパネルに配置します。
- 別のアプリケーションで作成した ActiveX コントロールにリンクします。

LabVIEW はオートメーション Refnum 制御器または ActiveX コンテナを使用して ActiveX オブジェクトにアクセスします。これは、どちらもフロントパネルオブジェクトです。ActiveX オブジェクトを選択するには、オートメーション Refnum 制御器を使用します。ボタンやドキュメントなど、表示可能な ActiveX オブジェクトを選択してフロントパネル上に配置するには、ActiveX コンテナを使用します。両方のオブジェクトともブロックダイアグラム上にオートメーション Refnum 制御器として表示されます。

ActiveX 有効のアプリケーションにアクセスする

ActiveX が有効になっているアプリケーションにアクセスするには、ブロックダイアグラム上でオートメーション Refnum 制御器を使用してアプリケーションへのリファレンスを作成します。呼び出しているアプリケーションを開く「オートメーションオープン（Automation Open）」関数に制御器を配線します。そのオブジェクトに関連付けられているプロパティを選択してアクセスするには、「プロパティノード」を使用します。そのオブジェクトに関連付けられているメソッドを選択してアクセスするには、「インボークノード」を使用します。「リファレンスを閉じる（Close Reference）」関数を使用してオブジェクトへのリファレンスを閉じます。リファレンスを閉じるとメモリからオブジェクトが削除されます。

たとえば、Microsoft Excel を開く VI を作成することによって、画面に Excel を表示したり、ワークブックやスプレッドシートを作成したり、LabVIEW で表を作成したり、Excel スプレッドシートにその表を書き込んだりすることができます。

Excel のクライアントとして LabVIEW を使用する方法の例については、`examples¥comm¥ExcelExamples.llb` の Write Table To XL VI を参照してください。

メモ　ActiveX カスタムインタフェースを含むアプリケーションは アイコンで表示さ れます。カスタムインタフェースのオブジェクトを選択するには、アイコンをクリックします。カスタムインタフェースの詳細については、この章の「カスタム ActiveX オートメーションインタフェースのサポート」のセクションを参照してください。

フロントパネルに ActiveX オブジェクトを挿入する

フロントパネルに ActiveX コントロールを挿入するには、ActiveX コンテナを右クリックして、ショートカットメニューから **ActiveX オブジェクトを挿入**を選択し、挿入する ActiveX コントロールまたはドキュメントを選択し ます。ActiveX プロパティブラウザまたはプロパティページを使用して、ActiveX オブジェクトのプロパティを設定したり、「プロパティノード」を使用してプロパティをプログラム的に設定することができます。ActiveX プロパティの設定方法の詳細については、この章の「ActiveX プロパティを設定する」のセクションを参照してください。

そのオブジェクトに関連付けられているメソッドを呼び出すには、「インボークノード」を使用します。

たとえば、ActiveX コンテナを使用して Microsoft Web Browser Control にアクセスしたり、メソッドの Navigate クラスを選択したり、URL メソッドを選択したり、URL を指定することによって、フロントパネル上にウェブページを表示できます。

ActiveX コンテナを使用すると、ブロックダイアグラム上のオートメーション refnum 制御器を「オートメーションオープン（Automation Open）」関数に配線したり、「リファレンスを閉じる（Close Reference）」関数を使用してオブジェクトへのリファレンスを閉じる必要がありません。ActiveX コンテナによって呼び出しアプリケーションが LabVIEW に埋め込まれているので、「インボークノード」や「プロパティノード」に直接配線できます。ただし、ActiveX コンテナに他のオートメーション Refnum 制御器を返すプロパティやメソッドが含まれている場合は、それらの追加リファレンスを閉じる必要があります。

ActiveX オブジェクトの設計モード

ActiveX コンテナを右クリックし、ショートカットメニューから**上級→設計モード**を選択して、VI の編集中にコンテナを設計モードで表示します。設計モードでは、イベントは生成されずイベント手順は実行されません。デフォルトモードは実行モードで、ユーザと同様にオブジェクトと対話します。

ActiveX プロパティを設定する

ActiveX サーバ開始後、または、ActiveX コントロールやドキュメントの挿入後、ActiveX プロパティブラウザ、プロパティページ、およびプロパティノードを使用して、そのコントロールまたはドキュメントに関連付けられたプロパティを設定することができます。

ActiveX プロパティブラウザ

ActiveX プロパティブラウザを使用して、ActiveX コンテナ内の ActiveX コントロールまたはドキュメントに関連付けられたすべてのプロパティを表示したり設定したりできます。ActiveX プロパティブラウザにアクセスするには、フロントパネルのコンテナにあるコントロールまたはドキュメントを右クリックして、ショートカットメニューから**プロパティブラウザ**を選択します。また、**ツール→上級→ ActiveX プロパティブラウザ**を選択することもできます。ActiveX プロパティブラウザを使用すると、ActiveX オブジェクトのプロパティを対話式で簡単に設定することができますが、ブラウザを使用してプロパティをプログラムで設定することはできません。また、ActiveX プロパティブラウザは、コンテナに入っている ActiveX オブジェクトにのみ使用できます。実行モードまたは VI の実行中は ActiveX プロパティブラウザを使用できません。

ActiveX プロパティページ

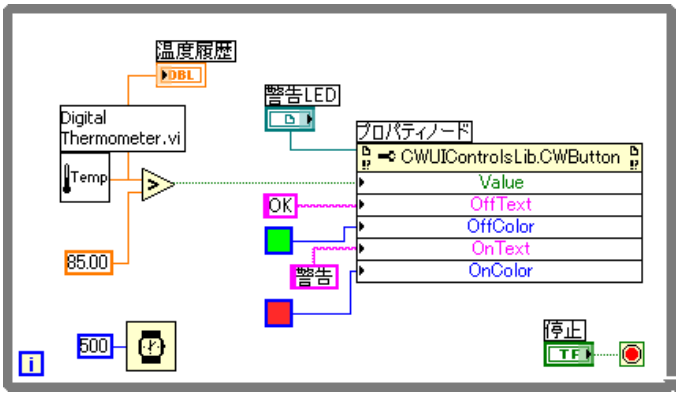
多くの ActiveX オブジェクトには、別のタブのオブジェクトに関連付けられたプロパティを構成するプロパティページが含まれます。ActiveX プロパティページにアクセスするには、フロントパネルのコンテナにあるオブジェクトを右クリックして、ショートカットメニューからオブジェクト名を選択します。

ActiveX プロパティブラウザと同様に、ActiveX プロパティページを使用して、ActiveX オブジェクトのプロパティを対話式に設定することができますが、プロパティをプログラム的に設定することはできません。また、プロパティページは、コンテナに入っている ActiveX オブジェクトにのみ使用できます。プロパティページは、すべての ActiveX オブジェクトに使用できるわけではありません。また、実行モードまたは VI を実行している場合にはプロパティページは使用できません。

プロパティノード

「プロパティノード」を使用して ActiveX プロパティをプログラム的に設定します。たとえば、ActiveX オブジェクトを使用して、温度が制限値を超えた場合にユーザに対して警告するには、「プロパティノード」を使用してオブジェクトの値プロパティに制限値を指定します。

以下の例では、温度が華氏 85 度以上になったときに、National Instruments Measurement Studio User Interface ActiveX Library の一部である CWButton ActiveX コントロールの値プロパティを変更します。



この場合、CWButton コントロールは LED としての役割を果たし、色が変わって、温度が制限値を超えた場合（CWButton コントロールが ON の状態）に警告を表示します。

メモ　この例では、CWButton コントロールの OffText、OffColor、OnText、および OnColor プロパティはプログラム的に設定する必要はないので、ActiveX プロパティブラウザまたはプロパティページを使用して、これらのプロパティを設定することができます。ActiveX プロパティブラウザの詳細についてはこの章の「ActiveX プロパティブラウザ」、プロパティページの詳細についてはこの章の「ActiveX プロパティページ」をそれぞれ参照してください。

ActiveX サーバとしての LabVIEW

他のアプリケーションからの ActiveX 呼び出しによって、LabVIEW のアプリケーション、VI、および制御器のプロパティとメソッドを使用できます。Microsoft Excel などの ActiveX 有効の他のアプリケーションが LabVIEW からプロパティ、メソッド、および個々の VI を要求する場合、LabVIEW は ActiveX サーバとして機能します。

たとえば、Excel のスプレッドシートに VI グラフを埋め込むと、スプレッドシートから VI 入力にデータを入力して VI を実行できます。VI を実行すると、グラフにデータがプロットされます。

Excel スプレッドシートで LabVIEW のプロパティおよびメソッドを使用する方法のサンプルについては、`examples¥comm¥freqresp.xls` を参照して ください。

カスタム ActiveX オートメーションインタフェースのサポート

LabVIEW を使って ActiveX サーバからプロパティやメソッドにアクセスする ActiveX クライアントを作成している場合は、サーバにより公開されるカスタムインタフェースにアクセスできます。カスタムインタフェースにアクセスするのに IDispatch を使用する必要はありません。ただし、ActiveX サーバの開発では、これらのカスタムインタフェースのプロパティおよびメソッドのパラメータのデータタイプがオートメーション (IDispatch) であることを確認してください。サーバの開発には、複数のオブジェクトではなく、1 つのオブジェクトから複数のインタフェースを表示する必要があります。ただし、従来どおり LabVIEW 環境でインタフェースを使用することができます。カスタムインタフェースへのアクセスの詳細については、サーバ開発プログラミング環境に関するドキュメントを参照してください。

定数を使用して ActiveX VI のパラメータを設定する

ActiveX ノードのパラメータには、有効な値の離散リストを取り込むものがあります。これらのパラメータ値を設定する場合は、リング定数から該当する名前を選択します。ActiveX VI の作成時にリング定数にアクセスするには、データ値を受け付けるノードのパラメータを右クリックして、ショートカットメニューから**作成→定数**を選択します。リング定数で選択できる定数は、ノードに渡される Refnum によって異なります。図 19-1 および 19-2 は、リング定数および数値定数を使用してパラメータ値を設定する方法のサンプルを示しています。

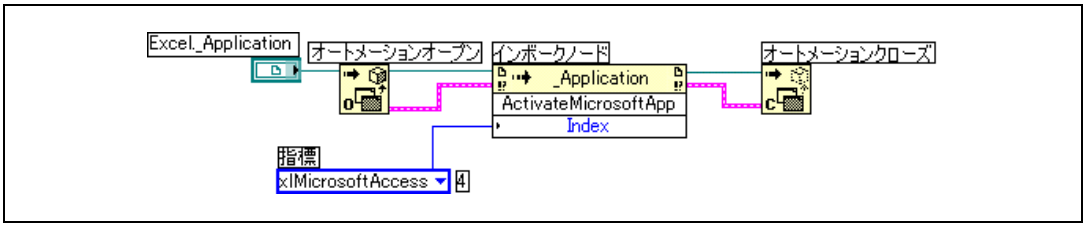


図 19-1　リング定数でデータ値を設定する

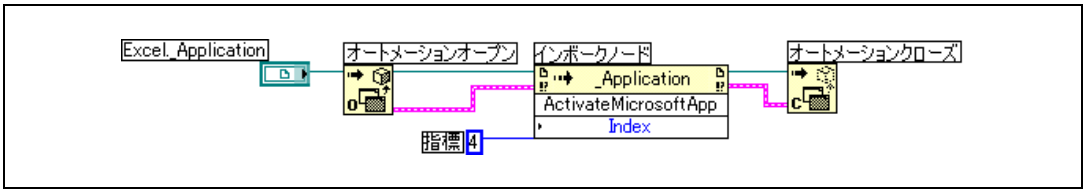


図 19-2　数値定数でデータ値を設定する

データ値を受け取るパラメータには、パラメータ名の左側に小さな矢印が表示されます。対応する数値データを表示するには、リング定数を右クリックしてショートカットメニューから**表示項目→デジタル表示**を選択します。

図 19-1 と図 19-2 の VI はどちらも、Microsoft Excel アプリケーションにアクセスしてメソッドを実行します。**ActivateMicrosoftApp** メソッドの**指標**パラメータには、**MicrosoftWord**、**MicrosoftPowerPoint**、**MicrosoftMail**、**MicrosoftAccess**、**MicrosoftFoxPro**、**MicrosoftProject**、および **MicrosoftSchedulePlus** というオプションがあります。

図 19-1 の **MicrosoftAccess** オプションに対応する**指標**パラメータの数値を識別するには、リング定数のプルダウンメニューから **MicrosoftAccess** オプションを選択します。現在選択されているオプションの数値がリング定数の隣りのボックスに表示されます。リング定数を使用する代わりに、図 19-2 のように数値定数にオプションの数値を入力することもできます。

ActiveX イベント

アプリケーションで ActiveX イベントを使用するには、イベントを登録して、イベントが発生したらそれを処理します。ActiveX イベント登録は、第 9 章「イベント駆動型プログラミング」の「イベントの動的登録」のセクションで説明しているダイナミックイベント登録に似ています。ただし、ActiveX イベント VI のアーキテクチャは、第 9 章「イベント駆動型プログラミング」で説明しているイベント処理のアーキテクチャとは異なります。以下のコンポーネントが通常の ActiveX イベント VI を形成します。

- イベントを生成する ActiveX オブジェクト。
- 生成するイベントのタイプを指定および登録する「レジスタイベントコールバックノード（Register Event Callback Node）」。
- 指定したイベントを処理するために書いたコードを含む「コールバック（Callback）」VI。

コンテナにある ActiveX オブジェクト上またはオートメーション Refnum を使用して指定した ActiveX オブジェクトに対してイベントを生成したり処理したりできます。たとえば、ActiveX コンテナから Windows ツリー制御器を呼び出して、このツリー制御器に表示されている項目に関するダブルクリックイベントを生成するように指定できます。

「レジスタイベントコールバックノード（Register Event Callback Node）」は複数のイベントを処理する拡大可能なノードで、「イベント登録ノード（Register For Events Node）」に似ています。

ActiveX オブジェクトへのリファレンスを「レジスタイベントコールバックノード」に接続して、そのオブジェクトに対して生成するイベントを指定すると、ActiveX オブジェクトがイベントに登録されます。イベントに登録した後、イベントを処理するために書かれたコードを含む「コールバック（Callback）」VI を作成します。

ActiveX イベントを処理する

コントロールが登録されたイベントを生成する場合は、「コールバック（Callback）」VI を作成して ActiveX コントロールからイベントを処理します。イベントが発生すると、「コールバック」VI が実行されます。コールバック VI を作成するには、「レジスタイベントコールバックノード（Register Event Callback Node）」の **VI Ref** 入力を右クリックして、ショートカットメニューから**コールバック VI を作成**を選択します。LabVIEW では以下の要素を含む再入可能 VI が作成されます。

- **イベント共通データ**には以下の要素が含まれています。
  - **ソース**は、LabVIEW または ActiveX など、イベントのソースを指定する数値制御器です。1 の値は ActiveX イベントを示します。
  - **タイプ**はどのイベントが発生するかを指定します。これはユーザインタフェースイベントの列挙型で、ActiveX およびその他のイベントソースの 32 ビット符号なし整数です。ActiveX イベントでは、イベントタイプは登録されたイベントのメソッドコードまたは ID を表します。
  - **時間**は、いつイベントが生成されたかを指定するタイムスタンプ（ミリ秒）です。
- **CtlRef** は、イベントが発生した ActiveX またはオートメーション Refnum へのリファレンスです。
- **イベントデータ**は、コールバック VI が処理するイベントに対する特定のパラメータのクラスタです。「レジスタイベントコールバックノード（Register Event Callback Node）」からイベントを選択すると、LabVIEW は適切な**イベントデータ**を指定します。文字列の制御器および表示器の詳細については、第 9 章「イベント駆動型プログラミング」の「通知およびフィルタイベント」のセクションを参照してください。

- **イベントデータ出力**は、コールバック VI が処理するイベントに対する特定の変更可能なパラメータのクラスタです。この要素はフィルタイベントにのみ有効です。
- （オプション）**ユーザパラメータ**は、ActiveX オブジェクトがイベントを生成したときに、LabVIEW がコールバック VI を介してユーザへ渡すデータです。

メモ　使用する VI のコネクタペーンがイベントデータのコネクタペーンと一致していれば、既存の VI をコールバック VI として使用することができます。ActiveX イベントが複数回発生した場合に LabVIEW がコールバック VI を同時に呼び出すことができるように、コールバック VI は再入可能 VI にすることをお勧めします。

20

テキストベースのプログラミング言語からのコード呼び出し

「ライブラリ関数呼び出しノード（Call Library Function Node）」を使用すると、LabVIEW で最も標準的な共有ライブラリを呼び出すことができます。「コードインタフェースノード（Code Interface Node : CIN）」を使用して、LabVIEW で C コードを呼び出すこともできます。

外部コードを呼び出す際のプラットフォーム固有の留意点については、『LabVIEW Development Guidelines』マニュアルの Chapter 6「LabVIEW Style Guide」の「Call Library Function Nodes and Code Interface Nodes」のセクションを参照してください。テキストベースのプログラミング言語からコードを呼び出す方法の詳細については、『Using External Code in LabVIEW』のマニュアルを参照してください。

詳細については

テキストベースのプログラミング言語からコードを呼び出す方法の詳細については、『LabVIEW ヘルプ』を参照してください。

ライブラリ関数呼び出しノード

最も標準的な共有ライブラリまたは DLL を呼び出すには、「ライブラリ関数呼び出しノード（Call Library Function Node）」を使用します。この関数を使用すると、LabVIEW でインタフェースを作成し、既存のライブラリや LabVIEW 用に特別に作成された新規ライブラリを呼び出すことができます。ナショナルインスツルメンツでは、「ライブラリ関数呼び出しノード」を使用して外部コードに対するインタフェースを作成することをお勧めします。

コードインタフェースノード

C で記述されたソースコードを呼び出す別の方法として CIN を使用する方法があります。通常、CIN よりもライブラリ関数呼び出しノードの方が簡単に使用できます。

21

フォーミュラと方程式

LabVIEW で複雑な方程式を使用する場合、ブロックダイアグラム上でさまざまな演算関数を組み合わせて配線する必要はありません。慣れている数学的環境で方程式を構築し、その方程式をアプリケーションに統合することができます。

LabVIEW 環境で数学演算を実行するには「フォーミュラノード（Formula Node ）」および「数式ノード （Expression Node）」を使用します。さらに追加機能が必要な場合は、数学アプリケーションである MATLAB にリンクして方程式を作成できます。

詳細については

方程式および構文を使用して使用可能な関数や演算子を活用する方法と発生する可能性のあるエラーの説明については、『LabVIEW ヘルプ』を参照してください。

LabVIEW で方程式を使用する方法

「フォーミュラノード」、「数式ノード 」、および「MATLAB スクリプトノード（MATLAB Script Node）」を使用すると、ブロックダイアグラムで数学演算を実行できます。

メモ スクリプトノードは、MATLAB スクリプトサーバを呼び出して MATLAB スクリプトを実行します。したがって、「MATLAB スクリプトノード」を使用するには、コンピュータ上に MATLAB をインストールする必要があります。LabVIEW は、Windows でのみ使用可能な ActiveX テクノロジを使用してスクリプトノードを実行します。スクリプトノードは「フォーミュラノード」と似ていますが、スクリプトノードを使用すると、既存の MATLAB スクリプトを ASCII 形式で LabVIEW にインポートしたり、そのスクリプトを LabVIEW で実行したりできます。「フォーミュラノード」とともに使用することによって、ノードとのデータの受け渡しが可能になります。

フォーミュラノード（Formula Nodes）

「フォーミュラノード」はテキストベースの便利なノードで、ブロックダイアグラム上で数学演算を実行できます。外部コードやアプリケーションにアクセスする必要がなく、方程式を作成するために低レベルの演算関数を配線する必要もありません。「フォーミュラノード」では、テキストベースの方程式表現だけでなく、テキストベースの if ステートメント、while ループ、for ループ、および do ループも使用できます。これらは C 言語のプログラマによく知られているものです。これらのプログラミング要素は C 言語でのプログラムに使用されるものと似ていますが、同一ではありません。

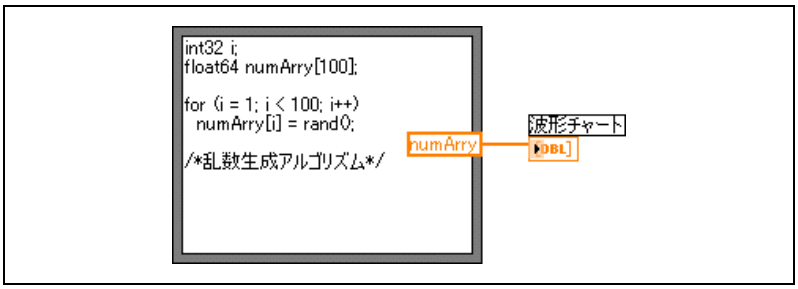
方程式に変数が多くある場合や方程式が複雑な場合、または既存のテキストベースのコードを使用する場合に、「フォーミュラノード」が便利です。既存のテキストベースのコードは、グラフィカルに再作成するのではなく、コピーして「フォーミュラノード」に貼り付けます。

「フォーミュラノード」は、類別化チェックを使用して、配列指標が数値データであることと、ビット演算に対するオペランドが整数データであることを確認します。また、「フォーミュラノード」は、配列指標が範囲内であることも確認します。配列については、範囲外の値はデフォルトでゼロ、範囲外の割り当てはデフォルトで nop に設定され、演算が行われないことを示します。

「フォーミュラノード」は、自動タイプ変換も行います。

「フォーミュラノード」を使用する

「フォーミュラノード」は、For ループ、While ループ、ケースストラクチャ、スタックシーケンスストラクチャ、およびフラットシーケンスストラクチャのようなサイズ変更可能なボックスです。ただし、次の例のように、「フォーミュラノード」はサブダイアグラムを含まず、セミコロンで区切られた、C 言語に似たステートメントを含んでいます。C 言語の場合と同じように、コメントをスラッシュとアスタリスクのペアで囲んで（/* コメント */）追加できます。



「フォーミュラノード」の使用例については、`examples¥general¥structs.llb` の Equations VI を参照してください。

「フォーミュラノード」の変数

変数を使用して作業する場合は、以下の点に注意してください。

- 1 つの「フォーミュラノード」内の変数または方程式の数に制限はありません。
- 複数の入力値または出力値どうしが同じ名前を使用することはできませんが、出力と入力を同じ名前にすることはできます。
- 「フォーミュラノード」の枠を右クリックして、ショートカットメニューから**入力端子を追加**を選択することによって入力変数を宣言します。「フォーミュラノード」内では入力変数を宣言できません。
- 「フォーミュラノード」の枠を右クリックして、ショートカットメニューから**出力端子を追加**を選択することによって出力変数を宣言します。出力変数名は、入力変数名または「フォーミュラノード」内で宣言した変数名と一致する必要があります。
- 変数を右クリックし、ショートカットメニューから**入力に変更**または**出力に変更**を選択すると、変数を入力または出力に変更できます。
- 「フォーミュラノード」内部で変数を宣言する場合、入力配線または出力配線に関連付けなくても使用できます。
- すべての入力端子は配線する必要があります。
- 変数を浮動小数点値のスカラにすることができます。浮動小数点値のスカラの精度はコンピュータの構成によって異なります。また、変数として整数や数値配列も使用できます。
- 変数に単位を付けることはできません。

数式ノード（Expression Nodes）

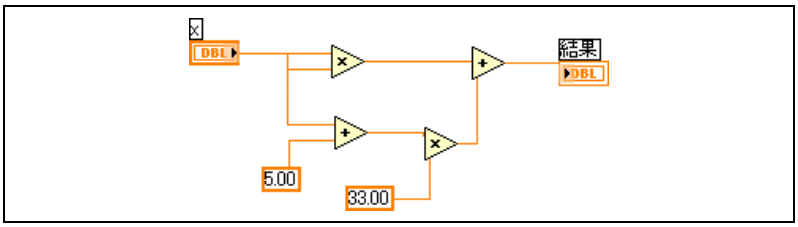
変数を 1 つ含む数式や方程式を計算するには、「数式ノード」を使用します。「数式ノード」は、方程式に変数が 1 つのみの場合には有効ですが、それ以外の場合は複雑になります。

「数式ノード」は、変数の値として入力端子に渡される値を使用します。出力端子は計算値を返します。

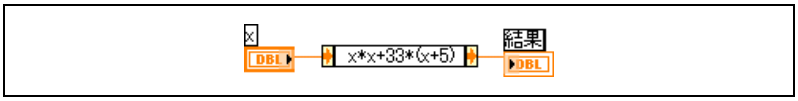
たとえば、以下に示す簡単な方程式を想定してみます。

$$x \times x + 33 \times (x + 5)$$

次の図のブロックダイアグラムでは、数値関数を使用してこの方程式を表現します。



次の図のように「数式ノード」を使用すると、ブロックダイアグラムがより簡単になります。



「数式ノード」の多形性

「数式ノード」の入力端子は、その端子に配線されている制御器や定数と同じデータタイプになります。出力端子は入力端子と同じデータタイプになります。入力のデータタイプは、複素数でないスカラ数値、複素数でないスカラ数値の配列、または複素数ではないスカラ数値のクラスタにすることができます。配列やクラスタを使用することによって、「数式ノード」は入力配列やクラスタの各要素に方程式を適用します。

MATLAB スクリプトノード（MATLAB Script Nodes）

ブロックダイアグラム上で MATLAB スクリプトを作成、ロード、および編集するには、「MATLAB スクリプトノード（ MATLAB Script Node）」を使用します。

スクリプトノードを使用するには、MATLAB をインストールする必要があります。MATLAB で記述されたスクリプトが既にある場合は、それをスクリプトノードにインポートできます。

スクリプトノード端子を入力または出力としてスクリプト内の変数に割り当てると、MATLAB と LabVIEW の間で値をやり取りできます。方程式の記述方法によって、端子の関数を決めることができます。たとえば、スクリプトに割り当てステートメント $X = i + 3$ が含まれている場合、i を入力端子として割り当てることによってスクリプトノード が X を計算する方

法を制御できます。さらに、*X* を出力端子に割り当てるとスクリプト計算の最終的な結果を取り出すことができます。

既存のスクリプトがない場合は、ブロックダイアグラム上にスクリプトノードを配置し、MATLAB の構文を使用してスクリプトを作成できます。LabVIEW は、スクリプトを実行するプログラムであるスクリプトサーバエンジンと通信します。LabVIEW は業界で確立されているプロトコルを使用して、通信したりスクリプトサーバエンジンを制御します。スクリプトサーバエンジンは MATLAB とともにインストールされます。

MATLAB スクリプト言語の性質により、作成した端子のデータタイプをスクリプトノードによって決めることはできません。各スクリプトノード端子には、LabVIEW データタイプを割り当てる必要があります。表 21-1 は、LabVIEW データタイプとそれに対応する MATLAB のデータタイプを示しています。

表 21-1　LabVIEW および MATLAB のデータタイプ

LabVIEW データタイプ	MATLAB データタイプ
DBL	実数
abc	文字列
	文字列
[DBL]	実数ベクトル
[DBL]	実数行列
CDB	複素数
[CDB]	複素ベクトル
[CDB]	複素行列

LabVIEW データタイプを MATLAB がサポートするデータタイプに変換するには、変換関数または文字列／配列／パス変換関数を使用してください。

MATLAB スクリプトについてのプログラム上の提案

以下のプログラミング手法を使用すると、スクリプトのデバッグ作業が簡単になります。

- スクリプトを作成し、LabVIEW にスクリプトをインポートする前に、テストとデバッグを兼ねて MATLAB 内で作成したスクリプトを実行します。
- データタイプを確認します。新しい入力端子または出力端子を作成する場合は、端子のデータタイプが正しいことを確認します。「エラー入力（Error In）」関数と「エラー出力（Error Out）」関数を使用してこれを監視します。
- 入力および出力の制御器や表示器を作成して、スクリプトノードが LabVIEW と MATLAB の間でやり取りする値を監視できるようにします。これによって、必要に応じてスクリプトノードのどこで間違って計算しているのかを突き止めることができます。
- エラーチェックのパラメータを情報のデバッグに利用します。スクリプトノードに**エラー出力**端子用の表示器を作成し、エラー情報を実行時に表示できるようにします。「フォーミュラノード」でもコンパイル時のエラーが表示されます。

A

LabVIEW の構成

この付録では、LabVIEW のファイルシステムの構造と、ファイルの保存に適した場所について説明します。

LabVIEW のディレクトリストラクチャの構成

このセクションでは、Windows、Mac OS、および UNIX での LabVIEW ファイルシステムの構造について説明します。LabVIEW では、プラットフォームの可用性に応じて、GPIB、DAQ、VISA、IVI、モーションコントロール、および IMAQ ハードウェア用のドライバソフトウェアがインストールされます。ハードウェアの構成の詳細については、『LabVIEW Measurements Manual 』の Chapter 3「Configuring Measurement Hardware」を参照してください。

インストールが完了すると、LabVIEW ディレクトリには以下のグループが含まれます。

ライブラリ

- `user.lib`：ユーザが作成する制御器および VI を保存します。LabVIEW では、制御器が**ユーザ制御器**パレット上に表示され、VI は**ユーザライブラリ**パレット上に表示されます。LabVIEW をアップグレードまたはアンインストールしても、このディレクトリは変更されません。`user.lib` ディレクトリにあるファイルの保存方法については、第 3 章「LabVIEW 環境」の「VI と制御器をユーザおよび計測器ドライバサブパレットに追加する」のセクションを参照してください。
- `vi.lib`：標準 VI のライブラリが含まれています。LabVIEW では、これらの VI は**関数**パレットにある関連グループに表示されます。`vi.lib` ディレクトリにファイルを保存しないでください。LabVIEW では、ファイルをアップグレードまたは再インストールすると上書きされます。
- `instr.lib`：PXI、VXI、GPIB、シリアル計測器、およびコンピュータベースの計測器の制御に使用される計測器ドライバが含まれています。ナショナルインスツルメンツの計測器ドライバをインストールする場合は、計測器ドライバをこのディレクトリに配置します。このドライバは LabVIEW によって**計測器ドライバ**パレットに追加されます。

構造とサポート

- `menus`：LabVIEW が**制御器**および**関数**パレットの構造の構成に使用するファイルが含まれています。
- `resource`：LabVIEW アプリケーションの追加サポートファイルが含まれています。ユーザのファイルをこのディレクトリに保存しないでください。アップグレードまたは再インストール時にそれらのファイルが LabVIEW によって上書きされる場合があります。
- `project`：LabVIEW の**ツール**メニューの項目になるファイルが含まれています。
- `templates`：通常の VI のテンプレートが含まれています。
- `www`：ウェブサーバを介してアクセスできる HTML ファイルを保存します。

実習と手順

- `examples`：VI のサンプルが含まれています。サンプルを検索するには、**ヘルプ→サンプルの検索**を選択します。

マニュアル

- `manuals`：PDF 形式のドキュメントが含まれています。このフォルダにはヘルプファイルは含まれていません。**ヘルプ→ LabVIEW ドキュメントライブラリを検索**を選択して、PDF にアクセスします。
- `help`：ヘルプファイルが含まれています。**ヘルプ→オンラインヘルプレファレンス**を選択して、『LabVIEW ヘルプ』にアクセスできます。

Mac OS

Mac OS 版では、前述のディレクトリの他に、LabVIEW アプリケーションのサポートファイルを含む共有ライブラリフォルダを使用します。

ファイル保存の推奨場所

vi.lib および resource ディレクトリは、LabVIEW システム専用として LabVIEW にインストールされています。これらのディレクトリにユーザのファイルを保存しないでください。

ユーザのファイルは以下のディレクトリに保存できます。

- user.lib：**ユーザ制御器**または**ユーザライブラリ**パレットに表示される任意の制御器または VI です。user.lib ディレクトリにファイルを保存する方法については、第 3 章「LabVIEW 環境」の「VI と制御器をユーザおよび計測器ドライバサブパレットに追加する」のセクションを参照してください。

メモ　サブ VI を変更することなくプロジェクト間で移植可能な場合のみ、user.lib ディレクトリにサブ VI を保存します。user.lib 内の VI パスは、labview ディレクトリを基準とした相対パスです。その他の場所に保存するサブ VI パスは呼び出し側 VI を基準とした相対パスです。そのため、特別な場合に VI を変更するために user.lib から VI をコピーしても、user.lib にあるサブ VI へのパスは変更されません。

- instr.lib：**計測器ドライバ**パレットに表示する任意の計測器ドライバ VI です。
- project：LabVIEW の機能を拡張するために使用する VI を保存します。このディレクトリに保存する VI は**ツール**メニューに表示されます。
- www：ウェブサーバを介してアクセスできる HTML ファイルを保存します。
- help：**ヘルプ**メニューで使用できるようににする VI、PDF、.hlp ファイルを保存します。
- LabVIEW データ：.lvm または .txt ファイルなど、LabVIEW で生成される任意のデータファイルです。

また、ユーザが作成する LabVIEW ファイルを保存するディレクトリは、ハードドライブ上のどこに作成してもかまいません。必要なディレクトリ作成の詳細については、第 7 章「VI およびサブ VI を作成する」の「VI を保存する」のセクションを参照してください。

B

多形性関数

関数の多形性には、どの入力も多形性でない関数、入力の一部が多形性である関数、入力のすべてが多形性である関数など、さまざまな度合いがあります。関数の入力には、数値またはブール値を受け入れるものがあります。数値や文字列を受け入れるものもあります。また、スカラ数値だけでなく、数値配列、数値クラスタ、数値クラスタの配列などを受け入れるものもあります。また、1 次元配列しか受け入れないものもありますが、配列要素はどのタイプでもかまいません。複素数値を含むすべてのタイプのデータを受け入れる関数もあります。多形性単位の作成と使用の詳細については、『Polymorphic Units in LabVIEW 』アプリケーションノートを参照してください。

詳細については

多形性関数の詳細については、『LabVIEW ヘルプ』を参照してください。

数値変換

数値表記法を他の数値表記法に変換できます。複数の異なる表記法の数値入力を関数に配線すると、関数は通常、精度の大きい方の形式で出力を返します。この関数は、実行前に小さい表記法を最も精度の大きい形式に強制し、LabVIEW は変換される端子に強制ドットを付けます。

「商（Divide)」、「Sine」、「Cosine」などの関数には、常に浮動小数点を出力するものもあります。入力側に整数を配線すると、これらの関数は整数を倍精度浮動小数点数に変換してから計算を実行します。

浮動小数点のスカラ値には通常、倍精度浮動小数点数の使用が最適です。単精度浮動小数点数を使用してもほとんど実行時間は短縮できず、オーバーフローがかなり発生しやすくなります。たとえば解析ライブラリでは、倍精度浮動小数点数を使用します。必要な場合に限り、拡張精度浮動小数点数を使用します。拡張精度の演算のパフォーマンスと精度はプラットフォームによって変わります。浮動小数点のオーバーフローの詳細については、第 6 章「VI の実行とデバッグ」の「不定データまたは予想外のデータ」のセクションを参照してください。

整数の場合は通常、32 ビット符号付き整数の使用が最適です。

数値表記の異なる接続先に出力を配線する場合、LabVIEW は以下の規則に従ってデータを変換します。

- **符号付きまたは符号なし整数から浮動小数点数**：32 ビット整数から単精度浮動小数点数への変換を除き、変換は完全に一致します。この場合、精度を 32 ビットから 24 ビットに下げます。
- **浮動小数点数から符号付きまたは符号なし整数**：範囲外の値を整数の最小値または最大値にします。For ループの繰り返し端子などほとんどの整数オブジェクトは浮動小数点数を丸めます。0.5 の指数部分を丸めて最も近い偶数にします。たとえば、6.5 は 7 ではなく 6 になります。
- **整数から整数**：範囲外の値を整数の最小値や最大値にしません。ソースが接続先よりも小さい場合、符号付きのソースの符号を拡張し、符号なしソースの余分なビットに 0 を入れます。ソースが接続先より大きい場合、その値の最下位ビットのみをコピーします。

数値関数の多形性

演算関数は数値入力データを使用します。関数の説明に記述されている例外を除いて、出力には入力と同じ数値表記法が使用され、複数の入力が異なる表記法である場合、出力は入力の数値表記法の中で最大幅のものになります。

演算関数は、数値、数値の配列、数値のクラスタ、数値のクラスタの配列、複素数などで実行できます。使用可能な入力タイプの正式な再帰定義は次のとおりです。

数値タイプ = 数値スカラ OR 配列 (数値タイプ) OR
クラスタ (数値タイプ)

数値スカラは、浮動小数点数、整数、または複素数浮動小数点数の場合があります。配列の配列は使用できません。

配列にはサイズおよび次元数の制限がありません。クラスタの要素数にも制限はありません。関数の出力タイプは入力タイプと同じ数値表記法です。入力が 1 つである関数については、その関数は配列またはクラスタの各要素の演算を行います。

2 つの入力を持つ関数については、以下の入力の組み合わせを使用できます。

- **同一**：入力が両方とも同じ構造を持ち、出力が入力と同じ構造を持ちます。
- **1 つのスカラ**：一方の入力が数値スカラで、他方が配列またはクラスタの場合、出力は配列またはクラスタになります。
- **配列**：1 つの入力が数値配列で、他方がその数値配列の数値タイプの場合、出力は配列になります。

同一の入力の場合、LabVIEW は構造の各要素について関数を実行します。たとえば、2 つの配列要素の和は要素ごとに実行します。この場合、両方の配列が同じ次元である必要があります。要素数の異なる配列の加算も可能ですが、この場合の出力は最も少ない入力と同じ要素数の配列になります。クラスタは同じ要素数を持ち、各要素が同じタイプである必要があります。

「積（Multiply）」関数を使用して行列を乗算することはできません。2 つの行列で「積」関数を使用すると、LabVIEW は 1 行目の最初の数値同士のように対応する要素ごとに乗算を行い、以下同様に処理を続けます。

スカラおよび配列（またはクラスタ）を含む演算の場合、LabVIEW はスカラおよび構造の各要素について関数を実行します。たとえば、配列の次元にかかわらず、配列のすべての要素から同じ数値を減算できます。

ある数値タイプおよびそのタイプの配列を含む演算の場合、LabVIEW は各配列の要素について関数を実行します。たとえば、グラフはポイントの配列で、ポイントは *x* および *y* の 2 つの数値タイプのクラスタです。グラフを *x* 方向に 5 単位、*y* 方向に 8 単位移動するには、そのグラフに点（5, 8）を加えます。

図 B-1 は「和（Add）」関数での可能な多形の組み合わせを示します。

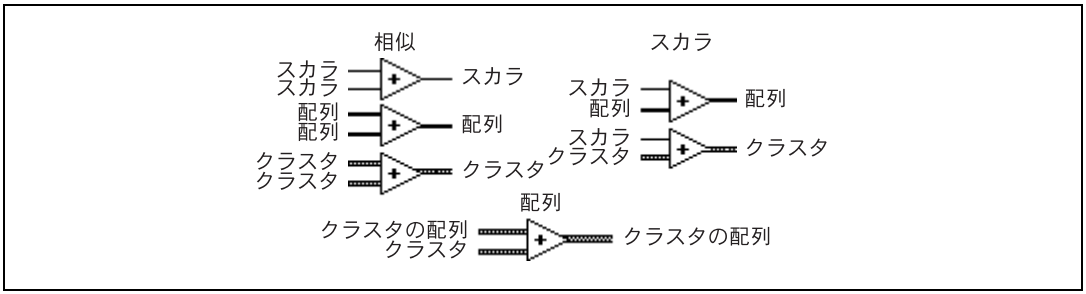


図 B-1　「和」関数での多形の組み合わせ

ブール関数の多形性

論理関数はブールまたは数値入力データのいずれかを使用します。入力が数値の場合、LabVIEW はビット単位の演算を実行します。入力が整数の場合、出力は同じ表記法になります。入力が浮動小数点数の場合は、倍長整数に丸められるため出力も倍長整数になります。

論理関数は、数値の配列またはブール値の配列、数値またはブール値のクラスタ、数値またはブール値のクラスタの配列などで実行できます。

使用可能な入力タイプの正式な再帰定義は次のとおりです。

論理タイプ = ブールスカラ OR 数値スカラ OR
配列 (論理タイプ)OR クラスタ (論理タイプ)

複素数および配列の配列は使用できないので除外します。

2 つの入力がある論理関数では、演算関数と同じ入力の組み合わせを持つことができます。ただし、論理関数には基本操作が 2 つのブール値の間または 2 つの数値の間に限られるという制約があります。たとえば、ブール値と数値の間で AND は使用できません。図 B-2 に「和」関数に対するブール値の組み合わせをいくつか示します。

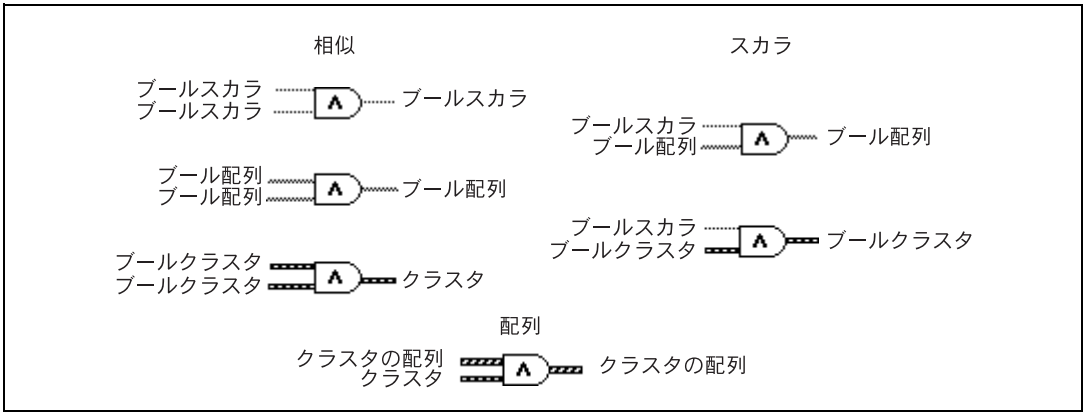


図 B-2　「和」関数のブールの組み合わせ

配列関数の多形性

ほとんどの配列関数は、すべてのタイプの n 次元を受け入れます。ただし、関数の説明にある配線ダイアグラムでは、数値配列をデフォルトデータタイプとして示します。

文字列関数の多形性

「文字列長（String Length）」、「大文字に変換（To Upper Case）」、「小文字に変換（To Lower Case）」、「文字列反転（Reverse String）」、および「回転境界（Rotate String）」は、文字列、クラスタ、文字列の配列、およびクラスタの配列を受け入れます。「大文字に変換」および「小文字に変換」は数値、数値のクラスタ、および数値の配列も受け入れ、それらは文字の ASCII コードとみなされます。幅および精度の入力はスカラである必要があります。

文字列変換関数の多形性

「パスを文字列に変換（Path To String）」および「文字列をパスに変換（String To Path）」関数は多形性です。つまり、スカラ値、スカラの配列、スカラのクラスタ、スカラのクラスタの配列などを入力できます。出力は入力と構成は同じですが、新しいタイプになります。

その他の文字列→数値変換関数の多形性

「数値を 10 進数文字列に変換（Number To Decimal String）」、「数値を 16 進数文字列に変換（Number To Hexadecimal String）」、「数値を 8 進数文字列に変換（Number To Octal String）」、「数値を工学形式文字列に変換（Number To Engineering String）」、「数値を小数文字列に変換（Number To Fractional String）」、および「数値を指数文字列に変換（Number To Exponential String）」はクラスタおよび数値の配列を受け入れ、文字列のクラスタおよび文字列の配列を生成します。「10 進数文字列を数値に変換（Decimal String To Number）」、「16 進数文字列を数値に変換（Hexadecimal String To Number）」、「8 進数文字列を数値に変換（Octal String To Number）」、および「小数／指数文字列を数値に変換（Fract/Exp String To Number）」は文字列のクラスタや配列を受け入れ、数値のクラスタや配列を生成します。幅および精度の入力はスカラである必要があります。

クラスタ関数の多形性

「バンドル（Bundle）」および「バンドル解除（Unbundle）」関数は、オブジェクトを入力または出力端子に配線するまで、それらの各端子のデータタイプを表示しません。配線すると、これらの端子は対応するフロントパネル制御器や表示器端子のデータタイプと同じように表示されます。

比較関数の多形性

「等しい？（Equal?）」、「等しくない？（Not Equal?）」、および「セレクト（Select）」関数は、入力が同じタイプであればどのようなタイプでも使用できます。

「以上？（Greater or Equal?）」、「以下？（Less or Equal?）」、「より小さい？（Less?）」、「よりも大きい？（Greater?）」、「最大最小（Max & Min）」および「範囲内と強制（In Range and Coerce?）」関数は、入力が同じタイプであれば、複素数、パス、または Refnum を除くすべてのタイプを使用できます。数値、文字列、ブール、文字列の配列、数値のクラスタ、文字列のクラスタなどを比較できます。ただし、数値と文字列の比較、文字列とブール値の比較などはできません。

値を 0 と比較する関数は、数値スカラ、クラスタ、および数値の配列を受け入れます。これらの関数は、入力と同じデータ構造のブール値を出力します。

「数値／パス /Refnum ではない？（Not A Number/Path/Refnum?）」関数は、値を 0 と比較する関数と同じ入力タイプを受け入れます。この関数は、パスや Refnum も受け入れます。「数値／パス /Refnum ではない?」関数は、入力と同じデータ構造のブール値を出力します。

「10 進数？（Decimal Digit?）」、「16 進数？（Hex Digit?）」、「8 進数？（Octal Digit?）」、「印刷可能？（Printable?）」、および「空白？（White Space?）」関数は、スカラ文字列やスカラ数値入力、文字列または非複素数のクラスタ、文字列または非複素数の配列などを受け入れます。出力は、入力と同じデータストラクチャのブール値から成ります。

「空の文字列／パス？（Empty String/Path?）」関数は、パス、スカラ文字列、文字列のクラスタ、文字列の配列などを受け入れます。出力は、入力と同じデータ構造のブール値から成ります。

「等しい ?」、「等しくない ?」、「数値／パス /Refnum ではない ?」、「空の文字列／パス ?」、および「セレクト」関数ではパスや Refnum を使用できますが、他の比較関数はパスや Refnum を入力として受け入れません。

配列やクラスタを受け取る比較関数は通常、ブール配列や同じ構造のクラスタを返します。関数から 1 つのブール値を返す場合、関数を右クリックして、ショートカットメニューから**比較モード→基礎群の比較**を選択します。関数による基礎群の比較方法の詳細については、付録 C「比較関数」の「配列とクラスタを比較する」のセクションを参照してください。

対数関数の多形性

対数関数は数値入力データを使用します。入力が整数の場合、出力は倍精度浮動小数点数になります。それ以外の場合、出力は入力と同じ数値表記法になります。

これらの関数は、数値、数値の配列、数値のクラスタ、数値のクラスタの配列、複素数などで実行できます。使用可能な入力タイプの正式な再帰定義は次のとおりです。

数値タイプ = 数値スカラ OR 配列 (数値タイプ)OR
クラスタ (数値タイプ)

配列の配列は使用できないので除外します。

配列にはサイズおよび次元数の制限はありません。クラスタの要素数にも制限はありません。出力タイプは入力と同じ数値表記法であり、関数はクラスタまたは配列の各要素に対して実行します。2 入力対数関数の詳細については、この付録の「数値関数の多形性」のセクションを参照してください。以下は 2 入力対数関数の場合に使用可能な入力タイプの組み合わせを示します。

- **同一**：入力が両方とも同じ構造を持ち、出力が入力と同じ構造を持ちます。
- **1 つのスカラ**：一方の入力が数値スカラで、他方が数値配列またはクラスタの場合、出力は配列またはクラスタになります。

C

比較関数

比較関数を使用して、ブール値、文字列、数値、配列、およびクラスタを比較します。ほとんどの比較関数は、1 つの入力をテストするか 2 つの入力を比較し、ブール値を返します。

詳細については

比較関数の詳細については、『LabVIEW ヘルプ』を参照してください。

ブール値を比較する

比較関数では、ブール値 TRUE の方がブール値 FALSE よりも大きいとみなされます。

文字列を比較する

LabVIEW では、ASCII 文字の数値表現に基づいて文字列が比較されます。たとえば、a（10 進数の 97）は A（65）よりも大きくなります。ここで、A は数字の 0（48）よりもよりも大きく、0 はスペース文字（32）より大きいとみなされます。不一致が発生するまで文字列の先頭から 1 文字ずつ比較し、その時点で比較は終了します。たとえば、文字列 abcd と abef の場合、LabVIEW は e の値よりも小さい c が見つかるまで評価します。また、文字が存在する場合の方が、存在しない場合よりも大きいとみなされます。したがって、文字列 abcd は abc よりも長いので、前者の方が大きいとみなされます。

「10 進数？（Decimal Digit?）」関数や「印刷可能？（Printable?）」関数など、文字列のカテゴリをテストする関数は、文字列の最初の文字だけを評価します。

数値を比較する

比較関数は、数値を同じ形式に変換してから比較します。NaN（Not a Number）値を持つ 1 つまたは 2 つの入力の比較では、不一致を示す値が返されます。NaN 値の詳細については、第 6 章「VI の実行とデバッグ」の「不定データまたは予想外のデータ」のセクションを参照してください。

配列とクラスタを比較する

一部の比較関数には、データの配列またはクラスタを比較するための 2 つのモードがあります。基礎群の比較モードでは、2 つの配列またはクラスタを比較する場合、1 つのブール値が返されます。要素の比較モードでは、要素が個別に比較され、ブール値の配列またはクラスタが返されます。

基礎群の比較モードでは、文字列比較操作と配列比較操作はまったく同じプロセスに従います。つまり、文字列は ASCII 文字の配列とみなされます。

比較関数を右クリックし、ショートカットメニューから**比較モード→要素を比較**または**比較モード→基礎群の比較**を選択して、関数のモードを変更します。一部の比較関数は基礎群の比較モードでのみ動作するため、ショートカットメニューの項目が表示されません。

配列

複数次元の配列を比較する場合は、関数に配線される各配列の次元数が同じである必要があります。基礎群の比較モードのみで動作する比較関数は、文字列の比較と同じ方法で配列を比較します。つまり、不一致が発生するまで最初の要素から要素を 1 つずつ比較します。

要素の比較モード

要素の比較モードでは、比較関数は入力配列と同じ次元のブール値配列を返します。出力配列の各次元は、その次元の 2 つの入力配列のうち小さい方と同じサイズになります。各次元（行、列、またはページなど）に沿って、関数は各入力配列内の対応する要素の値を比較し、対応するブール値を出力配列として生成します。

基礎群の比較モード

基礎群の比較モードでは、比較関数は、配列内で要素を比較した後に 1 つのブール結果を出力します。比較関数は、配列内の後ろにある要素を、配列内の前にある要素の補助的なものとみなします。そのため、関数は以下のステップを実行して比較の結果を確定します。

- 各入力配列に対応する要素の比較を配列の先頭から開始します。
- 対応する要素が等しくない場合、関数は停止し、この比較の結果を返します。
- 対応する要素が等しい場合、比較関数は次の値のペアを処理し、不一致を検出するか、どちらかの入力配列の最後に達するまで比較を続けます。
- 入力配列内のすべての値が等しいのに、片方の配列の最後に余分な要素がある場合、長い方の配列は短い方の配列よりも大きいとみなされます。たとえば、配列 `[1,2,3,2]` は配列 `[1,2,3]` よりも大きいとみなされます。

クラスタ

比較するクラスタには同じ数の要素が含まれていて、クラスタ内の各要素は互換性のあるタイプである必要があります。また、要素は同じクラスタ順位である必要があります。たとえば、倍精度数値と文字列を含むクラスタは、倍長整数数値と文字列を含むクラスタと比較できます。

要素の比較モード

要素の比較モードでは、比較関数が返すクラスタには、入力クラスタ内の要素一つ一つに対応するブール要素が含まれています。

基礎群の比較モード

基礎群の比較モードでは、比較関数は 1 つのブール値を返します。関数は不一致が検出されるまで対応する要素を比較し、不一致が検出された時点で結果を確定します。2 つのクラスタが等しいとみなされるのは、すべての要素が等しい場合のみです。

ソート済みのデータを含む 2 つのレコードを比較する場合は、クラスタに対して基礎群の比較モードを使用します。ここで、クラスタの後方にある要素は、クラスタの前方にある要素の補助的なものとみなされます。たとえば、2 つの文字列（ラストネームの次にファーストネーム）を含んでいるクラスタでは、関数はラストネームのフィールドが一致した場合にのみファーストネームのフィールドを比較します。

D

技術サポートおよびサービス

技術サポートおよびその他の専門サービスについては、ナショナルインスツルメンツのウェブサイト（ni.com/jp）の下記のセクションを参照してください。

- **サポート**：オンライン技術サポートには以下のリソースがあります。
  - **セルフヘルプリソース**：解答やソリューションがすぐに必要な場合は、技術サポートリソースの広範なライブラリ（ni.com/support/ja）をご利用いただけます（英語、スペイン語でも表示可）。これらのリソースは、登録ユーザの方ならほとんどの製品で無償でご利用いただくことができ、ソフトウェアドライバおよびアップデート、技術サポートデータベース、製品マニュアル、トラブルシューティングウィザード、ハードウェアの適合性に関するドキュメント、サンプルプログラム、チュートリアルおよびアプリケーションノート、計測器ドライバ、ディスカッションフォーラム、計測用語集などが含まれています。
  - **技術者によるサポートオプション**：弊社のエンジニアや計測 / オートメーション専門技術者までお問い合わせいただく場合は、ni.com/support/ja にアクセスしてください。オンラインシステムをご利用になりますと、システムがご質問内容を判別し、担当の弊社技術者がお電話、ディスカッションフォーラム、または E メールで回答いたします。
- **トレーニング**：自習形式のチュートリアル、ビデオ、および対話式 CD については、ni.com/jp/training にアクセスしてください。また、インストラクタによる実践コースも世界各地で開催しております。
- **システムインテグレーション**：時間の制約がある場合や社内で技術リソースが不足している場合、または、その他のプロジェクトで簡単には解消しない問題がある場合などは、ナショナルインスツルメンツのアライアンスパートナーによるサービスをご利用いただけます。詳しくは、最寄りの NI 営業所にお電話いただくか、ni.com/jp/alliance をご覧ください。

NI のウェブサイト（ni.com）で問題が解消しない場合は、最寄りの営業所またはナショナルインスツルメンツ本社までお問い合わせください。世界各地の弊社営業所の連絡先は、本書の巻頭に掲載されています。また、弊社ウェブサイトの Worldwide Offices セクション（ni.com/

niglobal）から各支社のウェブサイトにアクセスすることもできます。各支社のサイトでは、お問い合わせ先、サポート電話番号、E メールアドレス、イベント等に関する最新情報を提供しています。

用語

記号	接頭語	値
m	ミリ	10^{-3}
k	キロ	10^{3}
M	メガ	10^{6}

数字または記号

Δ　デルタ。偏差。Δ*x* は、ある指標から次の指標まで *x* の変化値を示します。

π　パイ（Pi）。

∞　無限。

1D　一次元。

1 次元　1 行の要素のみの配列の場合は 1 次元になります。

2D　2 次元。

2 次元　複数の行と列を持つ配列の場合のように、2 つの次元を持つこと。

3D カーブ　特殊なパラメトリックプロット (*x*(*t*), *y*(*t*), *z*(*t*))。ここでパラメータ *t* は指定された間隔で実行されます。

3D　3 次元。

A

A　アンペア。

AC　交流。

ASCII　情報交換用米国標準コード（American Standard Code for Information Interchange）。

C

CIN　「コードインタフェースノード（CIN）」の項を参照。

D

D	デルタ、偏差。D*x* は、ある指標から次の指標まで *x* の変化値を示します。
DAQ	「データ集録（DAQ）」の項を参照。
DAQ チャネルウィザード	DAQ アナログチャネルおよびデジタルチャネルの命名や構成を支援するユーティリティ。Measurement & Automation Explorer のデータ設定 **（Windows）** または DAQ チャネルウィザード **（Mac OS）** で使用できます。
DLL	ダイナミックリンクライブラリ（Dynamic Link Library）。

F

FIFO	先入れ先出し（First-in-first-out）のメモリバッファ。最初に格納されたデータは最初に受け入れ側に送られます。
For ループ	サブダイアグラムを一定回数実行する反復ループストラクチャ。以下のようなテキストベースのコードに相当します。`For i = 0 to n - 1, do...`

G

GPIB	「General Purpose Interface Bus」の項を参照。
General Purpose Interface Bus	GPIB は HP-IB と同義語です。電子計測器制御にコンピュータで使用する標準バスです。また、ANSI/IEEE 規格 488-1978、488.1-1987、および 488.2-1992 に規定されているので、IEEE 488 バスとも呼ばれます。

H

hex	16 進数。基数を 16 とする数。

I

I/O	入出力。通信チャネル、オペレータ入力装置、またはデータ集録およびデータ制御インタフェースを使用して、コンピュータシステムとの間で行うデータ転送。
IEEE	米国電子電気技術者協会（Institute for Electrical and Electronic Engineers）。

`Inf` 無限を表す浮動小数点のデジタル表示値。

IP インターネットプロトコル。

L

LabVIEW Laboratory Virtual Instrument Engineering Workbench（ラボラトリ仮想計測器エンジニアリングワークベンチ）。LabVIEW は、テキスト行ではなくアイコンを使用してプログラムを作成するグラフィカルなプログラミング言語です。

LabVIEW のシステム時間 LabVIEW が絶対時間のリファレンスとして使用する日付と時刻。LabVIEW のシステム時間は、万国標準時の 1904 年 1 月 1 日午前 0:00 として定義されています。

LED 発光ダイオード（Light-emitting diode）。

LLB VI ライブラリ。

M

Measurement & Automation Explorer ナショナルインスツルメンツの Windows 用のハードウェア構成および診断環境。

N

NI-DAQ NI 測定デバイスに含まれているドライバソフトウェア。NI-DAQ は、データをデバイスから構成、集録、および生成し、デイバスへデータを送信するために、LabVIEW などのアプリケーション開発環境（ADE）から呼び出すことができる VI および関数の広範なライブラリです。

NI-DAQmx 計測デバイスを制御するための、新しい VI、関数、開発ツールが搭載された新しい NI-DAQ ドライバ。NI-DAQmx は NI-DAQ の以前のバージョンと比較して、測定タスク、チャネル、およびスケールを構成する DAQ アシスタント、向上されたパフォーマンス、拡張された機能、そして簡易化されたアプリケーションプログラミングインタフェース（API）を持つ点で優れています。

`NaN` 数字ではないことをを表す浮動小数点のデジタル表示値。通常は、log(−1) などの不定の演算結果です。

O

OLE　オブジェクトのリンクと埋め込み（Object Linking and Embedding）。

P

PPC　プログラム間の通信（Program-to-program communication）。

PXI　計測器用の拡張（PCI eXtensions for Instrumentation）。モジュール形式、コンピュータベースの計測プラットフォーム。

R

refnum　リファレンス番号。開いたファイルに対し LabVIEW によって関連付けられた識別子。開いているファイルに対して関数または VI に操作を実行させる場合に refnum を使用します。

S

SCXI　信号調節拡張機構（Signal Conditioning eXtensions for Instrumentation）。ナショナルインスツルメンツのセンサ付近の外部シャーシ内の条件付き低レベル信号用製品ラインです。ノイズの多い環境の高レベル信号のみが DAQ デバイスに送信されます。

T

TCP/IP　転送制御プロトコル / インターネットプロトコル（Transmission Control Protocol/Internet Protocol）。あるコンピュータから他のコンピュータへデータパケットを転送する際の標準形式です。TCP/IP は、データパケットを処理する TCP、そしてコンピュータからコンピュータへデータパケットを送信する IP の 2 つの部分から構成されています。

U

UDP　ユーザデータグラムプロトコル（User Datagram Protocol）。

URL　Uniform Resource Locator。通常ウェブ上の、サーバのリソースを認識する論理アドレス。たとえば、ナショナルインスツルメンツのウェブサイトの URL は、`http://www.ni.com/` です。

V

VI	「バーチャルインスツルメンツ (VI：仮想計測器)」の項を参照。
VISA	「バーチャルインスツルメンツソフトウェアアーキテクチャ（仮想計測器ソフトウェアアーキテクチャ）」の項を参照。
VI クラス	VI のプロパティおよびメソッドにアクセスできる仮想計測器に対するリファレンス。
VI サーバ	プログラムによってローカルにもリモートにも VI や LabVIEW を制御するメカニズム。
VI ライブラリ	特定の用途に関連する VI 群を含む特殊ファイル。
VXI	VME eXtensions for Instrumentation（バス）。

W

While ループ	一定の条件が満たされるまでコードの一部を繰り返し実行するループストラクチャ。

あ

アイコン	ブロックダイアグラム上にあるノードの図式的な表示。
アクティブウィンドウ	現在ユーザの入力が可能なウィンドウで、通常は一番上に表示されているウィンドウです。アクティブウィンドウのタイトルバーはハイライトされます。ウィンドウをアクティブにするには、そのウィンドウをクリックするか、ウィンドウメニューからそのウィンドウを選択します。
アプリケーションソフトウェア	LabVIEW 開発システムを使用して作成され、LabVIEW ランタイムシステム環境で実行されるアプリケーション。

い

位置決めツール	オブジェクトの移動およびサイズ変更を行うツール。
イベント	アナログまたはデジタル信号の状態。

イベントデータノード　イベントストラクチャの左右に添付される、そのケースが処理するために構成した利用可能なデータを示すノード。ある 1 つのケースが複数のイベントを処理するように構成した場合、処理されたすべてのイベントタイプに共通するデータのみを使用することができます。

色付けツール　前景色および背景色を設定するためのツール。

インプレース　エラー I/O 端子またはシフトレジスタなど、複数の端子が同じメモリ領域を使用している状態。

え

エラー出力　VI から出力されるエラーストラクチャ。

エラーストラクチャ　ブールステータス表示器、数値コード表示器、および文字列ソース表示器から構成されています。

エラー入力　VI に入るエラーストラクチャ。

エラーメッセージ　ソフトウェアまたはハードウェアの動作不良または受け入れられないデータ入力が行われたことを示す。

お

オブジェクト　制御器、表示器、ノード、ワイヤ、取り込んだピクチャなど、フロントパネルまたはブロックダイアグラム上の項目の総称。

オブジェクトショートカットメニューツール　オブジェクトのショートカットメニューにアクセスするためのツール。

か

階層ウィンドウ　VI およびサブ VI の階層を図で表示するウィンドウ。

外部トリガ　A/D 変換などのイベントをトリガする外部トリガソースからの電圧パルス。

カウント端子　For ループの端子。カウント端子の値によって、For ループがサブダイアグラムを実行する回数が決まります。

カラーコピーツール　色決めツールで貼り付ける色をコピーします。

空の配列　要素が 0 で、データタイプが定義されている配列。たとえば、データ表示ウィンドウに数値制御器があるにも関わらずどの要素にも値が定義されていない配列は、空の数値配列です。

関数　内蔵されている実行要素。テキストベースのプログラミング言語の演算子、関数またはステートメントに相当します。

関数パレット　VI、関数、ブロックダイアグラムのストラクチャ、および定数を含むパレット。

き

競合状態　並列に実行される 2 つ以上のコードが、グローバル変数またはローカル変数で代表される同じ共有リソースの値を変更する場合に発生。

強制　データ要素の数値表記法を変更するため、LabVIEW が実行する自動変換。

強制ドット　2 つの異なる数値データタイプを持つデータを配線するとブロックダイアグラムノード上に警告を表示します。任意のバリアントデータタイプを配線した場合にも表示されます。

強度マップ / プロット　色を使用して 2 次元プロット上に 3 次元データを表示する方法。

共有ライブラリ　実行可能プログラムモジュールを含むファイル。いくつもの異なるプログラムがこのプログラムモジュールを使用して一部の機能を実行することができます。その他の開発者と、VI で作成した機能を共有したい場合に共有ライブラリを使用すると便利です。

行列　一次方程式の係数を表わす数字または数学的要素の長方形配列。

く

クラス　プロパティ、メソッド、およびイベントを含むカテゴリ。クラスは階層順に並べられていて、各クラスは下位にあるクラスに関連したプロパティおよびメソッドを受け継いでいます。

クラスタ　数値、ブール値、文字列、配列、クラスタなど、順序付けられてはいるが、指標付けされていない任意のデータタイプの要素の集合。要素は、すべて制御器またはすべて表示器のいずれかである必要があります。

クローン化　制御器や他のオブジェクトをコピーすること。<Ctrl> キーを押したまま、コピーする制御器やオブジェクトをクリックし、新しい位置までコピーをドラッグします。**（Mac OS）**の場合は、<Option> キーを押します。**（Sun）**では <Meta> キーを押します。**（Linux）**では <Alt> キーを押します。**（UNIX）**中央マウスボタンでオブジェクトをクリックし、そのコピーを新しい位置にドラッグすることもできます。

グラフ　1 つ以上のプロットの 2 次元表示。グラフはブロックとしてデータを受け取り、プロットします。

グラフ制御器　デカルト平面上にデータを表示するフロントパネルオブジェクト。

繰り返し端子　現時点までに完了した繰り返し回数が入っている For ループまたは While ループの端子。

グリフ　小さなピクチャまたはアイコン。

グループ　ユーザが定義する入出力チャネルまたはポートの集合。グループにはアナログ入力、アナログ出力、デジタル入力、デジタル出力、カウンタ／タイマチャネルなどが含まれます。1 つのグループには 1 種類のチャネルのみが含まれています。作成後は、グループの参照にタスク ID 番号を使用します。一度に最大で 16 個のグループを定義できます。グループを消去するには、空のチャネル配列とグループ番号をグループ構成 VI に渡します。グループのメンバを変更するためにグループを消去する必要はありません。タスクがアクティブになっているグループを再構成すると、LabVIEW はタスクを消去し、警告を返します。グループを再構成すると、LabVIEW がそのタスクを再開することはありません。

グローバル変数　ブロックダイアグラム上の複数の VI 間でデータにアクセスし、データのやり取りを行います。

け

ケース　ケースストラクチャの 1 つのサブダイアグラム。

ケースストラクチャ　ケースストラクチャへの入力に基づいてそのサブダイアグラムのいずれかを実行する条件分岐制御ストラクチャです。制御フロー言語では、IF、THEN、ELSE、および CASE ステートメントの組み合わせになります。

計測器ドライバ　プログラム可能な計測器を制御する VI。

現在の VI　フロントパネル、ブロックダイアグラム、またはアイコンエディタがアクティブウィンドウになっている VI。

こ

コードインタフェースノード（CIN）	CIN。ブロックダイアグラムの特殊ノードで、テキストベースのコードを VI にリンクする場合に使用します。
構成ユーティリティ	Windows の場合は 「Measurement & Automation Explorer」を参照。Mac OS の場合は 「NI-DAQ」を参照。
構文	特定のプログラミング言語においてステートメントが遵守する必要がある一連のルール。
コネクタ	入力端子と出力端子を含む VI または関数ノードの一部。ノードとのデータのやり取りはコネクタを経由して行われます。
コネクタペーン	フロントパネルまたはブロックダイアグラムウィンドウの右上隅にあり、VI 端子のパターンを表示する領域。これによって VI に配線できる入力と出力が定義されます。
壊れた VI	エラーのために実行できない VI。壊れた実行ボタンに壊れた矢印が表示されます。
壊れた**実行**ボタン	エラーのために VI が実行できない場合に**実行**ボタンから置き換わるボタン。
コンパイル	高レベルコードをコンピュータが**実行**できるコードに変換するプロセス。VI を作成または修正後初めて実行する前に、VI が自動的にコンパイルされます。

さ

最上位 VI	VI 階層の最上位にある VI。サブ VI から VI を区別する名称。
サイズ変更円またはハンドル	オブジェクトの枠上に表示され、オブジェクトの変更可能領域を示す円またはハンドル。
サブ VI	他の VI のブロックダイアグラムで使用される VI。サブルーチンに相当します。
サブダイアグラム	ストラクチャの枠内にあるブロックダイアグラム。
サンプル	アナログまたはデジタルの 1 つの入力または出力データ点。

し

シーケンスストラクチャ　「フラットシーケンスストラクチャ」または「スタックシーケンスストラクチャ」の項を参照。

シーケンスローカル　スタックシーケンスストラクチャのフレーム間でデータのやり取りを行う端子。

四角形　4 つの 16 ビット整数を含むクラスタ。最初の 2 つの値は左上隅の垂直および水平座標を示します。最後の 2 つの値は右下隅の垂直および水平座標を示します。

実行のハイライト　VI のデータフローを示すために VI の実行を図式的に表示するデバッグテクニック。

自動指標付け　ループストラクチャがその境界で配列を分解したり組み立てたりする機能。自動指標付けをオンにした状態で配列がループに入ると、ループは自動的にそれを 1 次元配列から抽出したスカラや 2 次元配列から抽出した 1 次元配列などに分解します。データがループを出るとき、ループはその逆の手順でデータを配列に組み立てます。

自動スケール　スケールがプロットした値の範囲に合わせて調整される機能。グラフのスケールでは、この機能によりスケールの最大値と最小値が決まります。

シフトレジスタ　ループストラクチャのオプションのメカニズムで、ループの 1 回の繰り返しから次の繰り返しへ変数の値を渡します。シフトレジスタは、テキストベースのプログラミング言語の静的変数に似ています。

ショートカットメニュー　オブジェクトを右クリックすることによってアクセスできるメニュー。メニュー項目はそのオブジェクトに固有のものです。

条件端子　VI が次の繰り返しを実行するかどうかを決めるブール値を含む While ループの端子。

人工データ依存　データフロープログラミング言語において、データの値ではなくデータの到着によってノードの実行がトリガされる状態。

す

スウィープチャート　オシロスコープの動作を模倣した数値制御器。スコープチャートに似ています。異なるのは、ディスプレイが線で区切られ、古いデータと新しいデータが分離される点です。

数値制御器と表示器　数値データの操作および表示に使用するフロントパネルオブジェクト。

スカラ　スケール上の一点で表すことができる数字。配列とは異なり、スカラは 1 つの値です。スカラのブール値とクラスタはそれぞれのデータタイプの明示的な単一のインスタンスです。

スクロールツール　ウィンドウ内の移動に使用するツール。

スケール　チャート、グラフ、および数値制御器や表示器の一部で、測定単位を示すために一定の間隔で一連のマークまたは点を付けた部分。

スコープチャート　オシロスコープの動作を模倣した数値制御器。

スタックシーケンスストラクチャ　数値順にサブダイアグラムを実行するプログラム制御ストラクチャ。データに依存しないノードを指定した順に強制実行するときに使用します。スタックシーケンスストラクチャは、各フレームを一度に 1 つずつ表示し、最後のフレームが実行されるまでを順番にフレームを実行します。

ストラクチャ　フラットスーケンスストラクチャ、スタックシーケンスストラクチャ、ケースストラクチャ、For ループ、While ループなどのプログラム制御要素。

ストリップチャート　紙テープに記録するチャートレコーダを模倣した数値プロット表示器。データをプロットするたびにスクロールします。

スライダ　スライド制御器および表示器の可動部分。

せ

制御器　対話式で VI にデータを入力したり、グログラムにサブ VI にデータを入力するためのノブ、押しボタン、ダイアルなどのフロントパネルオブジェクト。

制御器パレット　フロントパネル制御器、表示器、および装飾オブジェクトを含むパレット。

制御フロー　命令の順序によって実行順序が決まるプログラミング体系。テキストベースのプログラミング言語の多くは制御フロー言語です。

整数　自然数、負の自然数、または 0 のいずれか。

絶対座標　ピクチャ表示器の原点（0,0）を基準とした画像座標。

絶対パス　ファイルシステムの最上位を基準として位置を示すファイルまたはディレクトリのパス。

そ

操作ツール　データを操作する制御器にデータを入力するツール。

相対座標　現在のペンの位置を基準にしたピクチャ座標。

た

タイプ定義　複数の VI が使用できるカスタムオブジェクトのマスタコピー。

多形性　異なる表現、タイプ、または構造のデータに応じてノードが自動的に調整できる機能。

端子　データのやり取り行われるノード上のオブジェクトまたは領域。

ち

チェックボックス　選択や選択解除ができるダイアログボックス内の小さな正方形。通常、チェックボックスはユーザによる設定が可能な複数のオプションに関連付けられています。複数のチェックボックスを選択することができます。

チャート　1 つまたは複数のプロットの 2 次元表示です。表示には、ユーザの設定した最大数まで過去のデータが保持されます。チャートはデータを受け取り、表示を点または配列ごとに更新し、表示目的でバッファに過去の点を一定数保持します。「スコープチャート」、「ストリップチャート」、「スウィープチャート」の項も参照。

チャネル　1. 物理：　アナログまたはデジタル信号を測定または生成することができる端子またはピン。単一の物理チャネルは、異なるアナログ入力チャネルまたは 8 つのラインのデジタルポートの場合のように、複数の端子を含むことができます。また、カウンタも物理チャネルになることができますが、カウンタ名は、カウンタがデジタル信号を測定または生成する端子の名前ではありません。
2. バーチャル：　名前、物理チャネル、入力端子接続、測定または生成のタイプ、およびスケール情報を含むプロパティ設定の集合。タスクの外側（グローバル）またはタスクの内側（ローカル）で NI-DAQmx バーチャルチャネルを定義できます。バーチャルチャネルを従来型 NI-DAQ または以前バージョンで構成するのはオプションですが、NI-DAQmx で行うすべての測定に不可欠です。従来型 NI-DAQ では、MAX でバーチャルチャネルを構成します。NI-DAQmx では、バーチャルチャネルを MAX またはご使用のプログラムで構成でき、チャネルをタスクの一部として、または個別に構成できます。
3. スイッチ：　スイッチチャネルは、スイッチ上の接続ポイントを示します。スイッチトポロジによって、一本または複数の信号ワイヤ（通常、1、2、または 4）で構成されています。バーチャルチャネルは、スイッチチャネルで作成できません。スイッチチャネルは、NI-DAQmx スイッチ関数または VI でのみ使用します。

チャネル名　DAQ チャネルウィザードでチャネル構成に指定する固有の名前。

つ

ツール　特定の操作を実行する特殊カーソル。

ツールバー　VI の実行やデバッグに使用するコマンドボタンが並んでいるバー。

ツールパレット　フロントパネルやブロックダイアグラムオブジェクトの編集やデバッグに使用するツールで構成されているパレット。

通知イベント　LabVIEW に対して、ユーザが制御器の値を変更した場合など、ユーザのアクションがすでに発生していることを知らせるイベント。

て

データ依存　データフロープログラミング言語で、別のノードからデータを受け取るまでノードが実行できない状態。「人工データ依存」の項も参照。

データ集録（DAQ）　1. センサ、トランスデューサ、およびテストプローブまたは装置からアナログまたはデジタル電気信号を集録および測定する。
2. アナログまたはデジタル電気信号を生成する。

データタイプ　情報の形式。LabVIEW では、数値、配列、文字列、ブール値、パス、refnum、列挙体、波形、クラスタといったデータタイプが、ほとんどの VI および関数に使用できます。

データフロー　実行可能なノードで構成されるプログラミング体系。ノードは必要な入力データをすべて受けとった場合のみ実行され、実行されると自動的に出力を作成します。LabVIEW はデータフローシステムです。

データロギング　一般には、データを集録すると同時にそれをディスクファイルに格納すること。LabVIEW のファイル I/O VI および関数はデータロギングできます。

データログファイル　ファイル作成時に指定した 1 つの任意データタイプの一連のレコードとしてデータを格納するファイル。データログファイルのレコードはすべて 1 つのタイプで構成される必要がありますが、そのタイプが複合的であっても支障ありません。たとえば、各レコードを文字列、数値、配列を含むクラスタとして指定できます。

ディザリング　アナログ入力信号にガウスノイズを加えること。ディザリングを適用した後、入力データを平均化することによって効果的に、さらに 0.5 ビットだけ分解能を増やすことができます。

定数　「ユニバーサル定数」および「ユーザ定義定数」の項を参照。

ディレクトリ　ファイルを扱いやすいグループに分類する構造。ディレクトリとは、ファイルの位置を示すアドレスのようなもので、中にはファイルやファイルのサブディレクトリが入っています。

デバイス　実環境の入出力ポイントを制御および監視する 1 つの構成要素としてアクセスすることができる計測器またはコントローラ。多くの場合、デバイスはいずれかのタイプの通信ネットワークを介してホストコンピュータに接続されます。

デフォルト　あらかじめ設定されている値。値を指定しないと、多くの VI 入力でデフォルト値が使用されます。

と

ドライバ　デバイスやデバイスの種類に固有のソフトウェア。デバイスが受け入れるコマンドのセットを含みます。

ドライブ　a–z までの文字にコロン（:）が付いたもの。論理ディスクドライブを示します。

ドラッグ	オブジェクトの選択、移動、コピー、または削除を行うために、画面上のカーソルを使用すること。
トンネル	ストラクチャ上のデータ入力端子または出力端子。

の

ノード	プログラム実行要素。ノードはテキストベースのプログラミング言語のステートメント、演算子、関数、およびサブルーチンに似ています。ブロックダイアグラムでは、ノードに関数、ストラクチャ、およびサブ VI が含まれています。

は

配線ツール	端子間のデータパスを定義するツール。
バイトストリームファイル	一連の ASCII 文字またはバイトでデータを格納するファイル。
配列	一定の順序で並べられ、指標付けされている同一タイプのデータ要素。
波形	特定のサンプリング速度で読み取った複数の電圧の読み取り値。
波形チャート	一定の速度でデータ点をプロットする表示器。
バーチャルインスツルメンツ (VI：仮想計測器)	実際の計測器の外観と機能を模倣した LabVIEW のプログラム。
バーチャルインスツルメンツソフトウェアアーキテクチャ（仮想計測器ソフトウェアアーキテクチャ）	VISA。GPIB、VXI、RS-232、その他の計測器を制御するための単一のインタフェースライブラリ。
バッファ	集録されたデータや生成されたデータの一時格納場所。
パネルウィンドウ	フロントパネル、ツールバー、およびアイコンとコネクタペーンが含まれている VI ウィンドウ。
パレット	使用可能なオプションを表すアイコン表示。
範囲	測定、受信、または送信される数量の上限と下限の間の領域。範囲の上限値と下限値を示すことによって表現します。

ハンドル　メモリの 1 ブロックへのポインタを指すポインタ。基準配列や文字列を管理します。文字列の配列は、文字列へのハンドルが入っているメモリブロックのハンドルとなります。

凡例　グラフまたはチャートが所有するオブジェクト。このグラフまたはチャートにプロット名とプロット方式を表示します。

ひ

表記法　数値データタイプのサブタイプ。符号付きと符号なしバイト整数、ワード整数、倍長整数のほか、単精度、倍精度、拡張精度の浮動小数点数があります。

表示器　グラフや LED などの出力を表示するフロントパネルオブジェクト。

ピクセル　デジタル化されたピクチャの最小単位。

ピクチャ　ピクチャ表示器によって画像の作成に使用される一連の図式的な手順。

ピクチャ表示器　線、円、テキスト、その他の画像形状を含むことがあるピクチャを表示する汎用表示器。

ピックスマップ　画像を格納する標準フォーマット。各ピクセルを色の値で表します。ビットマップはピックスマップをモノクロで表したものです。

ヒントラベル　端子名を識別する小さな黄色のテキストバナー。配線する端子の識別を容易にします。

ふ

ファイル refnum　「refnum」の項を参照。

フィルタイベント　インタフェースの動作を制御します。

フィルタ処理　測定する信号から不要な信号を除去する信号処理の一種。

フォーミュラノード　テキストとして入力された式を実行するノード。式が長く、ブロックダイアグラムの形式で作成するには面倒な場合に特に効果的です。

フラットシーケンスストラクチャ　数値順にサブダイアグラムを実行するプログラム制御ストラクチャ。データに依存しないノードを指定した順に強制実行するときに使用します。フラットシーケンスストラクチャは、すべてのフレームを一度に表示し、最後のフレームまで左から右にフレームを実行します。

フリーラベル	フロントパネルまたはブロックダイアグラム上にあり、他のどのオブジェクトにも属さないラベル。
フレーム	フラットまたはスタックシーケンスストラクチャのサブダイアグラム。
フロントパネル	VI の対話形式ユーザインタフェース。フロントパネルの外観は、オシロスコープやマルチメータなどの実際の計測器に似ています。
ブール制御器および表示器	ブール（TRUE または FALSE）データの操作や表示に使用するフロントパネルオブジェクト。
ブレークポイント	デバッグに使用される、実行の一時停止。
ブレークポイントツール	VI 、ノード、またはワイヤにブレークポイントを設定するためのツール。
ブロックダイアグラム	プログラムまたはアルゴリズムを図式的に説明または表記したもの。ブロックダイアグラムは、ノードという実行可能なアイコンと、ノード間でデータをやり取りするワイヤで構成されています。ブロックダイアグラムが VI のソースコードです。ブロックダイアグラムは VI のブロックダイアグラム**ウィンドウ**にあります。
プルダウンメニュー	メニューバーからアクセスするメニュー。通常、プルダウンメニューの項目は一般的なものです。
プローブ	VI で中間値をチェックするデバッグ機能。
プローブツール	ワイヤ上にプローブを作成するためのツール。
プロット	グラフまたはチャートのいずれかで示したデータ配列を図式的に表すこと。
プロパティノード	VI またはアプリケーションのプロパティの設定または検出。

へ

平坦化されたデータ	文字列に変換された何らかのタイプのデータで、通常はファイルに書き込むためのものです。
ベクトル	1 次元配列。
ヘルプウィンドウ	LabVIEW オブジェクトの上にカーソルを移動すると、各オブジェクトに関する基本情報を表示。ヘルプ情報のあるオブジェクトには、VI、関数、定数、ストラクチャ、パレット、プロパティ、メソッド、イベント、およびダイアログボックスなどがあります。
変換	データ構要素のタイプを変更します。

ほ

ポイント　水平および垂直座標を表す 2 つの 16 ビット整数を含んでいるクラスタ。

ま

マルチスレッドアプリケーション　複数の異なる実行スレッドを別々に実行するアプリケーション。マルチプロセッサコンピュータでは、異なるスレッドを別個のプロセッサ上で同時に実行できます。

め

メソッド　オブジェクトがメッセージを受け取ったときに実行される手順。メソッドは常にクラスに関連付けられています。

メニューバー　アプリケーションのメインメニューの名前を一覧表示する水平バー。メニューバーは、ウィンドウのタイトルバーの下に表示されます。メニューおよびコマンドの中には多くのアプリケーションに共通するものもありますが、各アプリケーションのメニューバーはそれぞれ異なります。

メモリバッファ　「バッファ」の項を参照。

も

文字列　テキストとしての値の表現。

文字列制御器および表示器　テキストの処理や表示に使用するフロントパネルオブジェクト。

ゆ

ユーザ定義定数　設定した値を送り出すブロックダイアグラムオブジェクト。

ユニバーサル定数　特定の ASCII 文字や標準の数値定数を送出する、編集できないブロックダイアグラムオブジェクト。たとえば p。

ら

ライブラリ　「VI ライブラリ」の項を参照。

ラベリングツール　ラベルを作成し、テキストウィンドウにテキストを入力するツール。

ラベル　フロントパネルまたはブロックダイアグラム上のオブジェクトや領域に名前を付けたり、説明を示すために使用するテキストオブジェクト。

り

離散　独立した変数が不連続の値を持つこと。通常、時間がこれに該当します。

リストボックス　コマンドに対する選択肢をすべて一覧表示するダイアログボックス内のボックス。たとえば、ディスク上のファイル名の一覧表示などがあります。

リング制御器　32 ビット整数を一連のテキストラベルやグラフィックに関連付ける特殊な数値制御器。0 から始まり順次増分します。

ろ

ローカル変数　VI のフロントパネル上の制御器または表示器に読み書きできるようにする変数。

わ

ワイヤ　ノード間のデータパス。

ワイヤ屈折点　2 本のワイヤセグメントの接合点。

ワイヤスタブ　配線ツールを VI または関数ノード上に移動したときに未配線の端子部分に表示される短いワイヤ。

ワイヤセグメント　水平または垂直の 1 本のワイヤ。

ワイヤの接点　3 本以上のワイヤセグメントの結点。

ワイヤブランチ　接点から接点まで、端子から接点まで、端子間に接点がない場合は端子から端子までの、すべてのワイヤセグメントを含むワイヤの部分。

索引

数値

1 ステップずつ VI を実行する、6-6
2 次元制御器および表示器、4-9
3 次元グラフ、12-18
3 次元制御器および表示器、4-9

A

ActiveX、19-1
　ActiveX 有効のアプリケーションにアクセスする、19-8
　VI、19-7
　VI サーバ、17-1
　イベント、19-7、19-13
　イベント処理、19-14
　オブジェクト、19-6
　カスタムインターフェース、19-12
　カスタムインターフェースを選択する、19-9
　関数、19-7
　クライアント、19-8
　コールバック VI、19-14
　コンテナ、19-7
　コントロール、19-7
　サーバ、19-11
　サブパレットを作成する、3-8
　スクリプトノードを実行するために、21-1
　設計モード、19-9
　定数を使用してパラメータを設定する、19-12
　ネットワーク動作、18-1
　パラメータを設定するための定数、19-12
　表示器、19-7
　プロパティノード、19-10
　プロパティブラウザ、19-10
　プロパティページ、19-10
　プロパティを設定する、19-10
　プロパティを表示する、19-10
　プロパティをプログラム的に設定する、19-10
　フロントパネルにオブジェクトを挿入する、19-9
　メソッドを呼び出す。『LabVIEW ヘルプ』を参照。
　リモートフロントパネル、18-16
ActiveX コンテナ
　設計モード、19-9
ActiveX を使用してオブジェクトを埋め込む、19-9
AppleEvents、18-21
ASCII
　文字セットを使用する、18-18

B

BMP ファイル、13-6

C

CIN、20-1
C コード
　LabVIEW から呼び出す、20-1

D

DAQ
　VI および関数、7-4
　構成ユーティリティ、1-4
　ソリューションウィザード、1-5
　チャネルウィザード、1-4
　チャネルビューワ、1-5
　チャネル名を渡す、4-20
DataSocket、18-2
　URL、18-3
　データ形式、18-5
　バッファに格納されたデータ、18-8
　バリアントデータ、18-9
　プログラムで接続を閉じる、18-7
　プログラムで接続を開く、18-7
　ブロックダイアグラム、18-6
　プロトコル、18-3

フロントパネル、18-5
フロントパネルオブジェクトを制御する。『LabVIEW ヘルプ』を参照。
DataSocket 用の URL、18-3
DLL
LabVIEW から呼び出す、20-1
作成する
VI を配布する、7-14
Do ループ。「While ループ」の項を参照。
dstp DataSocket プロトコル、18-3

E

Express VI、5-18
VI として構成を保存する。『LabVIEW ヘルプ』を参照。
アイコン、7-9
拡張可能なノード、7-9
Express VI からサブ VI を作成する、5-19
『LabVIEW ヘルプ』を参照。
Express VI をサブ VI に変換する、5-19
『LabVIEW ヘルプ』を参照。

F

file DataSocket プロトコル、18-4
For ループ
カウント端子、8-2
繰り返し端子、8-2
自動指標付けで回数を設定する、8-4
シフトレジスタ、8-6
使用する。『LabVIEW ヘルプ』を参照。
タイミングを制御する、8-10
デフォルトデータ、6-11
ftp DataSocket プロトコル、18-4

G

GIF ファイル、15-4
GPIB
構成する、1-4

H

HTML
「ウェブ」の項を参照。
ウェブ上に VI をパブリッシュする、18-11
グラフィック形式、15-4
ドキュメントを作成する、18-11
ヘルプファイル、15-5
レポートを生成する、15-7
～にドキュメントを保存する、15-3

I

I/O
エラー、6-13
制御器と表示器、4-20
データタイプ（表）、5-4
ファイル。「ファイル I/O」の項を参照。
～名制御器と表示器、4-20
Inf(無限大) 浮動小数点値
不定データ、6-10
ini ファイル
読み書き、14-12
ISO Latin-1
文字セットを使用する、18-18
IVI
計測器ドライバ。『LabVIEW ヘルプ』を参照。
論理名を渡す、4-20

J

Joint Photographic Experts Group ファイル、13-6
JPEG ファイル、13-6、15-4
ウェブサーバ、18-12

L

LabVIEW、1-1
オプション、3-8
カスタマイズする、3-8
labview.ini、3-8

LabVIEW のディレクトリ構造、A-1
Mac OS、A-2
構成、A-1
logos DataSocket プロトコル、18-4
.lvm ファイル、14-20

M

Mac OS
文字セットを使用する、18-19
MATLAB
スクリプトノード、21-4
スクリプトをデバッグする、21-6
データタイプ、21-5
Measurement & Automation Explore
VXI
構成する、1-4
MRU メニュー項目、3-4

N

NaN（数値以外）浮動小数点値
不定データ、6-10
.NET
GAC（Global Assembly Cache）、19-3
アセンブリ、19-2
アプリケーションを導入する、19-5
オブジェクトを作成する。『LabVIEW ヘルプ』を参照。
環境、19-2
関数、19-3
共通言語ランタイム（CLR）、19-2
クライアントとしての LabVIEW、19-4
クラスライブラリ、19-2
コンストラクタノード、19-3
データタイプをマッピングする、19-5
.NET Framework、19-2
NI-DAQ 構成ユーティリティ、1-4

O

OLE for Process Control DataSocket プロトコル、18-3
opc DataSocket プロトコル、18-3

P

PDF ライブラリ、1-1
PNG ファイル、13-6、15-4
ウェブサーバ、18-12
PPC Toolbox、18-21

R

refnum
オートメーション、19-7
厳密に類別化された、17-7
制御器、17-9
制御器と表示器、4-25
使用する。『LabVIEW ヘルプ』を参照。
データタイプ（表）、5-3
ファイル I/O、14-1
リファレンス呼び出しノード、17-7
Repeat-Until ループ。「While ループ」の項を参照。
RTF
～にドキュメントを保存する、15-3
rtm ファイル、16-2

S

SMTP
VI を使用する、18-17
字訳、18-19
文字セット、18-18
System Exec VI、18-21

T

TCP、18-20
VI サーバ、17-1

U

UDP、18-20

V

VI
アンロックする。『LabVIEW ヘルプ』を参照。
移植する、7-15
印刷する、15-5
ウェブ上で制御する、18-11
ウェブ上にパブリッシュする、18-11
エラー処理、6-12
外観と動作を設定する、16-1
階層、7-11
開発する、7-1
共有する、7-14
均等に配置する、7-14
検索する。『LabVIEW ヘルプ』を参照。
厳密に類別化された Refnum、17-7
更新する、7-14
コピーする、A-3
コマンドラインから起動する。『LabVIEW ヘルプ』を参照。
壊れた、6-2
最後に保存されたバージョンに戻す。『LabVIEW ヘルプ』を参照。
作成する、7-1
サブ VI として呼び出された VI を制御する、7-4
サンプル、1-4
実行可能
デバッグする。『LabVIEW ヘルプ』を参照。
実行する、6-1
実行モード時に開く。『LabVIEW ヘルプ』を参照。
修正する、6-2
説明を作成する、15-2
ダイナミックに呼び出す、17-7
ダイナミックにロードする、17-7
多形性、5-15
デバッグ方法、6-3
ドラッグアンドドロップする。『LabVIEW ヘルプ』を参照。
名前を付ける、7-13
バージョンを比較する、7-2
バーチャルインスツルメンツ。「VI」の項を参照。
プログラムで制御する、17-1
文書化する、15-1
保存する、7-12
ライブラリ、7-12
ライブラリから削除する。『LabVIEW ヘルプ』を参照。
ライブラリ内の最上位としてマークを付ける。『LabVIEW ヘルプ』を参照。
ライブラリに追加する、3-6
リファレンス。『LabVIEW ヘルプ』を参照。
リモートで呼び出す、17-1
ローカライズする、7-15
ロックする。『LabVIEW ヘルプ』を参照。
VISA
リソース名を渡す、4-20
VI オブジェクト
VI サーバ、17-3
設定を操作する、17-3
VI 検索パス
編集する。『LabVIEW ヘルプ』を参照。
VI サーバ
VI オブジェクト、17-3
VI の設定を操作する、17-3
アプリケーションオブジェクト、17-3
アプリケーションを作成する、17-2
インポークノード、17-5
ウェブ上で LabVIEW の他のインスタンスを呼び出す、17-1
機能、17-1
厳密に類別化された VI Refnum、17-7
ネットワーク動作、18-1
プロパティノード、17-4
フロントパネルオブジェクトを制御する、17-9
リファレンス呼び出しノード、17-7
リモートアプリケーション、17-8
リモートで VI を呼び出す、17-1
VI 全体でステップを実行する
VI をデバッグする、6-6

VI として Express VI 構成を保存する、5-19
『LabVIEW ヘルプ』を参照。
VI を HTML ファイルまたはコンパイル済みのヘルプファイルにリンクする。『LabVIEW ヘルプ』を参照。
VI を移植する、7-15
VI を開発する、7-1
ガイドライン、1-3
開発作業を追跡する。「VI を文書化する」の項を参照。
VI を更新する、7-14
VI を実行する、6-1
VI をダイナミックに呼び出す、17-7
VI を文書化する
印刷する、15-3
オブジェクトおよび VI の説明を作成する、15-2
作成したヘルプファイルにリンクする
ヒントラベルを作成する、15-2
プログラムで、15-3
ヘルプファイル、15-5
レビジョン履歴、15-1
VI を保存する
旧バージョンの形式、7-14
個別ファイル、7-12
最後に保存されたバージョンに戻す。『LabVIEW ヘルプ』を参照。
ライブラリ、7-12
VI を連続実行する、6-1
VI をローカライズする、7-15
VXI
VI、1-3

W

While ループ
エラー処理、6-14
繰り返し端子、8-3
自動指標付け、8-5
シフトレジスタ、8-6
条件端子、8-2
使用する。『LabVIEW ヘルプ』を参照。
タイミングを制御する、8-10
無限、8-3

X

XML
サンプル、10-6
スキーマ、10-8
データタイプを変換する、10-7
～から変換する、10-7
XML 用スキーマ、10-8
XY グラフ、12-9
データタイプ、12-11
x スケール
複数、12-2

Y

y スケール
複数、12-2

あ

アイコン、2-4
Express VI、7-9
印刷する、15-3
作成する、7-8
サブ VI、7-9
編集する、7-8
アドオンツールセット、1-1
パレット内に、3-8
アプリケーション
VI サーバを作成する、17-2
スタンドアロンを作成する、7-15
VI を配布する、7-14
アプリケーションオブジェクト
VI サーバ、17-3
設定を操作する、17-3
アプリケーション制御関数、5-9
アプリケーションノート、1-3
アプリケーションビルダ。「スタンドアロンアプリケーション」の項を参照。
アプリケーションフォント、4-28

い

移動する
オブジェクト。『LabVIEW ヘルプ』を参照。
クラスタ。『LabVIEW ヘルプ』を参照。
サブパレット。『LabVIEW ヘルプ』を参照。
配列。『LabVIEW ヘルプ』を参照。
ワイヤ。『LabVIEW ヘルプ』を参照。
イベント
「イベントストラクチャ」の項を参照。
ActiveX、19-7、19-13
ActiveX を処理する、19-14
LabVIEW で使用できる～。『LabVIEW ヘルプ』を参照。
サポートされた、9-1
処理、9-5
静的登録、9-8
設定する。『LabVIEW ヘルプ』を参照。
ダイナミック～を登録解除する、9-10
通知、9-4
定義された、9-1
動的
サンプル、9-11
登録、9-8
動的に登録する。『LabVIEW ヘルプ』を参照。
フィルタ、9-4
フロントパネルをロックする、9-7
ユーザ、9-12
ユーザ～を作成する、9-13
ユーザ～を生成する、9-13
ユーザ～を登録解除する、9-14
ユーザ～を登録する、9-13
イベント駆動型プログラミング。「イベント」、「イベントストラクチャ」の項を参照。
イベント処理、9-5
イベントストラクチャ
「イベント」の項を参照。
使用する、9-2
入れ替える
配列内の要素。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上のオブジェクト。『LabVIEW ヘルプ』を参照。
フロントパネル上のオブジェクト、4-2
文字列内のテキスト。『LabVIEW ヘルプ』を参照。
色
オプション、3-8
画像内で作成する、13-5
画像内で変更する、13-5
ハイカラー制御器および表示器、4-9
パレット、4-12
ボックス、4-12
マッピング、12-14
ランプ
回転式制御器および表示器、4-12
ローカラー制御器および表示器、4-9
色を決める
システムカラー、4-30
『LabVIEW ヘルプ』を参照。
前景オブジェクト。『LabVIEW ヘルプ』を参照。
透明なオブジェクト。『LabVIEW ヘルプ』を参照。
背景オブジェクト。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクト、4-5
色をコピーする。『LabVIEW ヘルプ』を参照。
ユーザカラーを定義する。『LabVIEW ヘルプ』を参照。
印刷する
VI 実行後のフロントパネル、15-6
VI のドキュメント、15-3
アクティブウィンドウ、15-5
オプション、3-8
サブ VI を使用する、15-6
上位 VI からのデータ、15-6
ドキュメントを保存する
HTML に、15-3
RTF に、15-3
テキストファイルに、15-3
プログラムで、15-6
方法、15-7
レポート、15-7

インストーラ
作成する、7-15
インターネット。「ウェブ」の項を参照。
インボークノード、17-5
ActiveX、19-7

う

ウィンドウサイズ
定義する。『LabVIEW ヘルプ』を参照。
ウェブ
HTML ドキュメントを作成する、18-11
LabVIEW の他のインスタンスを呼び出す、17-1
VI を制御する、18-12
VI をパブリッシュする、18-11
技術サポート、D-1
プロフェッショナルサービス、D-1
フロントパネルを参照する、18-12
ウェブサーバ
VI を制御する、18-12
オプション、18-11
フロントパネルを参照する、18-12
有効にする、18-11
リモートフロントパネルのクライアント
LabVIEW、18-14
ウェブブラウザ、18-15
複数、18-13
リモートフロントパネルのライセンス、18-13
ウェブ上に VI をパブリッシュする、18-11
ウェブパブリッシュツール、18-11

え

エイリアス除去ラインプロット、12-2
エラー
I/O、6-13
ウィンドウ、6-2
カスタムを定義する。『LabVIEW ヘルプ』を参照。
既存の～をキャンセルする。『LabVIEW ヘルプ』を参照。
クラスタ、6-13
コネクタペーン、7-7
コンポーネント、6-14
レポート。『LabVIEW ヘルプ』を参照。
検索する、6-2
コード、6-13
壊れた VI、6-2
自動処理、6-12
自動処理する、6-12
処理、6-12
While ループを使用する、6-14
計測器制御。『LabVIEW ヘルプ』を参照。
ケースストラクチャを使用する、6-14
メソッド、6-13
単位不適合、5-23
通知。『LabVIEW ヘルプ』を参照。
デバッグ方法、6-3
表示する、6-2
リスト、6-2
例外の制御。『LabVIEW ヘルプ』を参照。
～として通常の状態。『LabVIEW ヘルプ』を参照。
～をチェックする、6-13
エラーとして通常の状態。『LabVIEW ヘルプ』を参照。
エラーの通知。『LabVIEW ヘルプ』を参照。
演算。「方程式」の項を参照。
演算子。「ノード」の項を参照。

お

オートメーション
カスタムインターフェース、19-12
制御器 Refnum、19-7
オーバレイプロット、12-8
オブジェクト
ActiveX、19-6
ActiveX を使用してフロントパネルに挿入する、19-9
移動する。『LabVIEW ヘルプ』を参照。

オプション項目を表示する、4-2
均等に配置する、4-6
『LabVIEW ヘルプ』を参照。
検索する。『LabVIEW ヘルプ』を参照。
制御器を表示器に、表示器を制御器に変更する、4-2
整列させる、4-6
『LabVIEW ヘルプ』を参照。
説明を印刷する、15-3
説明を作成する、15-2
選択する。『LabVIEW ヘルプ』を参照。
等間隔に配置する、4-6
等間隔に配置する『LabVIEW ヘルプ』を参照。
透明。『LabVIEW ヘルプ』を参照。
並び替える。『LabVIEW ヘルプ』を参照。
パレットに挿入する。『LabVIEW ヘルプ』を参照。
ヒントラベルを作成する、15-2
プログラムで制御する、17-9
ブロックダイアグラム、5-1
ブロックダイアグラム上で手動配線する、5-10
ブロックダイアグラム上で配線する、5-12
ブロックダイアグラム上に入れ替える。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上に挿入する。『LabVIEW ヘルプ』を参照。
フロントパネルおよびブロックダイアグラム端子、5-1
フロントパネル上で入れ替える、4-2
フロントパネル上で重ねる、4-19
フロントパネル上でグループ化およびロックする、4-6
フロントパネル上でサイズ変更する、4-7
ウィンドウサイズに応じて～、4-7
フロントパネル上でスケールする、4-7
フロントパネル上でタブ順序を設定する、4-4
フロントパネル上のキャプション、4-27
作成する。『LabVIEW ヘルプ』を参照。
フロントパネルに非表示
『LabVIEW ヘルプ』を参照。
オプション項目、4-2
フロントパネルの色を決める、4-5
色をコピーする。『LabVIEW ヘルプ』を参照。
ラベルを付ける、4-26
サイズ変更する。『LabVIEW ヘルプ』を参照。
作成する。『LabVIEW ヘルプ』を参照。
編集する。『LabVIEW ヘルプ』を参照。
オブジェクトを均等に配置する、4-6
『LabVIEW ヘルプ』を参照。
オブジェクトを等間隔に配置する、4-6
『LabVIEW ヘルプ』を参照。
オブジェクトを並び替える。『LabVIEW ヘルプ』を参照。
オブジェクトを整列させる、4-6
『LabVIEW ヘルプ』を参照。
オプション
格納する、3-8
設定する、3-8
『LabVIEW ヘルプ』を参照。
温度計
「数値」の項を参照。
スライド制御器および表示器、4-10
オンライン技術サポート、D-1

か

カーソル
グラフ、12-4
グラフから削除する。『LabVIEW ヘルプ』を参照。
グラフに追加する。『LabVIEW ヘルプ』を参照。
階層ウィンドウ、7-11
印刷する、15-3
検索する。『LabVIEW ヘルプ』を参照。
回転式制御器および表示器、4-10
開発作業を追跡する。「VI を文書化する」の項を参照。
開発のガイドライン、1-3

カウント端子、8-2
設定する自動指標付け、8-4
書き込みグローバル、11-3
書き込みローカル、11-3
拡張可能なノード、7-9
拡張プローブ。「プローブ」の項を参照。
格納する
作業環境オプション、3-8
カスタマー
技術サポート、D-1
トレーニング、D-1
プロフェッショナルサービス、D-1
カスタマイズする
VI の外観と動作、16-1
エラーコード。『LabVIEW ヘルプ』を参照。
作業環境、3-6
パレット、3-6
プローブ。『LabVIEW ヘルプ』を参照。
メニュー、16-2
カスタムオートメーションインターフェース、19-12
カスタムプローブウィザード、6-8
画像。「グラフィック」の項を参照。
画面解像度、4-30
カラーをマッピングする、12-14
空ディスク容量をチェックする。『LabVIEW ヘルプ』を参照。
空のパス、4-14
関数、5-7
アプリケーション制御、5-9
クラスタ、5-8
検索する。『LabVIEW ヘルプ』を参照。
サイズ変更する。『LabVIEW ヘルプ』を参照。
時間、5-8
上級、5-9
数値、5-7
ダイアログ、5-8
多形性、B-1
端子を削除する、5-10
端子を追加する、5-10
配列、5-8
波形、5-9
パレット
ウィンドウタイトル。『LabVIEW ヘルプ』を参照。
カスタマイズする、3-6
操作と検索、3-2
ファイル I/O、5-9
ブール、5-7
ブロックダイアグラム、5-7
文字列、5-7
リファレンス。『LabVIEW ヘルプ』を参照。
関数パレットのウィンドウタイトル。『LabVIEW ヘルプ』を参照。
簡単なメニュー、3-4

き

キーボードショートカット、4-3
タブ順序を設定する、4-4
ボタンを制御する、4-4
技術サポート、D-1
技術サポートデータベース、D-1
既存のエラーをキャンセルする。『LabVIEW ヘルプ』を参照。
起動時のログインプロンプト
表示する。『LabVIEW ヘルプ』を参照。
キャプション、4-27
作成する。『LabVIEW ヘルプ』を参照。
サブ VI ヒントラベル。『LabVIEW ヘルプ』を参照。
キュー
バリアントデータ、5-22
旧バージョン
VI を保存する、7-14
旧バージョンの制御器および表示器、4-9
競合状態、11-4
強制ドット、5-14
強度グラフ、12-12
オプション、12-15
カラーマッピング、12-14
強度チャート、12-12
オプション、12-14
カラーマッピング、12-14
共有する
VI、7-14

パレットセット。『LabVIEW ヘルプ』を参照。
ファイル、7-2
プログラムによるライブデータ、18-6
他の VI やアプリケーションとの間でライブデータを、18-2
共有ライブラリ
LabVIEW から呼び出す、20-1
作成する、7-15
VI を配布する、7-14
協力レベル
設定する。『LabVIEW ヘルプ』を参照。
極グラフ、13-2
記録
「リビジョン履歴」の項を参照。
オプション、3-8
チャート、12-7
均等に配置する
VI、7-14
フロントパネル上のオブジェクトを～、4-6
『LabVIEW ヘルプ』を参照。

く

クライアント
ActiveX、19-8
.NET、19-4
リモートフロントパネルのウェブブラウザ、18-15
リモートフロントパネルの複数の、18-13
リモートフロントパネル用に LabVIEW を、18-14
クラスタ
移動する。『LabVIEW ヘルプ』を参照。
エラー、6-13
コンポーネント、6-14
レポート。『LabVIEW ヘルプ』を参照。
関数、5-8
サイズ変更する。『LabVIEW ヘルプ』を参照。
制御器と表示器、4-15
データタイプ（表）、5-3
多形性、B-5
配線パターン、10-14
配列とクラスタの間で変換する。『LabVIEW ヘルプ』を参照。
比較する、C-2
要素の順序
変更する、10-14
『LabVIEW ヘルプ』を参照。
クラスタ要素の順序、10-14
変更する、10-14
『LabVIEW ヘルプ』を参照。
グラフ、12-1
XY、12-9
データタイプ、12-11
3 次元、12-18
エイリアス除去ラインプロット、12-2
オプション、12-2
カーソル、12-4
削除する。『LabVIEW ヘルプ』を参照。
追加する。『LabVIEW ヘルプ』を参照。
外観をカスタマイズする、12-3
強度、12-12
オプション、12-15
極、13-2
グラフィック、13-2
作成する。『LabVIEW ヘルプ』を参照。
消去する。『LabVIEW ヘルプ』を参照。
ズームする。『LabVIEW ヘルプ』を参照。
スクロールする、12-3
スケールする、12-5
スケールのフォーマット、12-5
スミスプロット、13-3
スムーズアップデート、12-6
タイプ、12-1
追加する。『LabVIEW ヘルプ』を参照。
デジタル
データをマスクする、12-17
伝送ライン、13-3
動作をカスタマイズする、12-3
波形、12-9
データタイプ、12-9、12-10
複数のスケール、12-2

グラフィック
VI アイコンに追加する。『LabVIEW ヘルプ』を参照。
色を作成する、13-5
色を変更する、13-5
インポートする、4-5
ウェブ上にフロントパネルをパブリッシュする、18-12
グラフ、13-2
形式、13-6
HTML ファイル用、15-4
形状を描画する、13-4
テキストを入力する、13-4
ドラッグアンドドロップする。『LabVIEW ヘルプ』を参照。
ピクチャ制御器および表示器
使用する、13-1
データタイプ（表）、5-4
ピックスマップ、13-4
グラフィックをインポートする、4-5
グラフとチャートをズームする。『LabVIEW ヘルプ』を参照。
繰り返し端子
For ループ、8-2
While ループ、8-3
繰り返す
コードのブロック
For ループ、8-2
While ループ、8-2
グリッド、4-6
オプション、3-8
グリッドにスナップ、4-6
グループ化する
データ
クラスタ、10-13
配列、10-8
文字列、10-1
フロントパネルオブジェクト、4-6
ライブラリ内の VI、7-12
グローバル変数
競合状態、11-4
作成する、11-2
初期化する、11-4
慎重に使用する、11-4
メモリ、11-5
読み取りと書き込み、11-3

け

警告
デフォルトで表示する。『LabVIEW ヘルプ』を参照。
表示する、6-3
ボタン、6-3
形式
フロントパネル上のテキスト、4-27
文字列、10-4
指定子、10-4
形式文字列パラメータ、10-4
形式文字列を使用する。『LabVIEW ヘルプ』を参照。
計測器
構成する、1-4
制御する、7-3
計測器ドライバ、D-1
LabVIEW。『LabVIEW ヘルプ』を参照。
ゲージ
「数値」の項を参照。
カラーランプを追加する、4-12
フロントパネル、4-10
ケースストラクチャ
エラー処理、6-14
使用する。『LabVIEW ヘルプ』を参照。
セレクタ端子
値、8-12
データタイプ、8-12
デフォルトケースを指定する、8-11
検索する
LabVIEW マニュアルの PDF バージョン、1-1
VI 階層。『LabVIEW ヘルプ』を参照。
エラー、6-2
オブジェクト、テキスト、VI。『LabVIEW ヘルプ』を参照。
パレット上の制御器、VI、および関数、3-2
厳密タイプチェック、5-23

厳密に類別化された Refnum
VI、17-7
制御器、17-9

こ

構成する
VI の外観と動作、16-1
ダイナミックイベント。『LabVIEW ヘルプ』を参照。
フロントパネル、4-3
フロントパネル制御器、4-1
フロントパネル表示器、4-1
メニュー、16-2
ユーザイベント。『LabVIEW ヘルプ』を参照。
リモートフロントパネル用にサーバを、18-13
LabVIEW、18-15
ウェブブラウザ、18-15
構成ファイル VI
.ini ファイルの読み書き、14-12
形式、14-13
目的、14-13
構造とサポートのディレクトリ
WWW、A-2
テンプレート、A-2
プロジェクト、A-2
メニュー、A-2
リソース、A-2
コードインターフェースノード、20-1
コールバック VI
ActiveX、19-14
コネクタペーン、2-5
印刷する、15-3
設定する、7-6
必須および任意の入出力、7-8
コピーする
VI、A-3
グラフィック、4-5
フロントパネルまたはブロックダイアグラム上のオブジェクトを～。『LabVIEW ヘルプ』を参照。
コマンドライン
VI を起動する。『LabVIEW ヘルプ』を参照。
コマンドラインから VI を起動する。『LabVIEW ヘルプ』を参照。
壊れた VI
一般的な原因、6-3
エラーを表示する、6-2
修正する、6-2
コンテナ、4-19
ActiveX、19-7
サブパネル制御器、4-19
タブ制御器、4-19
コントロール
ActiveX、19-7
コンピュータベース計測器
構成する、1-4
コンボボックス、4-13

さ

サーバ
ActiveX、19-11
リモートフロントパネル用に設定する、18-13
LabVIEW、18-15
ウェブブラウザ、18-15
最近使用したメニュー項目、3-4
最後に保存されたバージョンに戻す。『LabVIEW ヘルプ』を参照。
サイズ変更する
Express VI、7-9
関数
クラスタ。『LabVIEW ヘルプ』を参照。
サブ VI、7-9
ノード。『LabVIEW ヘルプ』を参照。
配列。『LabVIEW ヘルプ』を参照。
表。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクト、4-7
ウィンドウサイズに応じて～、4-7
ユーザ定義定数、5-5
ラベル。『LabVIEW ヘルプ』を参照。
サイズを決める。「サイズを変更する」の項を変更する。
サウンド、13-6

作業環境オプション
格納する、3-8
設定する、3-8
設定する『LabVIEW ヘルプ』を参照。
削除する
関数から端子を～、5-10
ストラクチャ。『LabVIEW ヘルプ』を参照。
多形性 VI からのインスタンス。『LabVIEW ヘルプ』を参照。
データログレコード、14-16
配列要素。『LabVIEW ヘルプ』を参照。
パレットセット。『LabVIEW ヘルプ』を参照。
不良ワイヤ、5-14
フロントパネルまたはブロックダイアグラム上のオブジェクトを～。『LabVIEW ヘルプ』を参照。
ライブラリの VI。『LabVIEW ヘルプ』を参照。
作成する
.NET オブジェクト
VI、7-1
VI サーバアプリケーション、17-2
VI の説明、15-2
アイコン、7-8
オブジェクトの説明、15-2
共有ライブラリ
VI を配布する、7-14
グラフ。『LabVIEW ヘルプ』を参照
計測器ドライバアプリケーション。『LabVIEW ヘルプ』を参照。
サブ VI、7-5、7-10
回避する状況。『LabVIEW ヘルプ』を参照。
サブパレット。『LabVIEW ヘルプ』を参照。
スタンドアロンアプリケーション
VI を配布する、7-14
スプレッドシートファイル、14-8
制御器リファレンス。『LabVIEW ヘルプ』を参照。
多形性 VI、5-16
チャート
データログファイル、14-10
テキストファイル、14-8
バイナリファイル、14-9
配列、10-11
パレットセット、3-7
ヒントラベル、15-2
ブロックダイアグラム、5-1
フロントパネル、4-1
メニュー、16-2
ユーザイベント、9-13
ユーザ定義定数、5-5
レビジョン履歴、15-1
サブ VI
Express VI から作成する、5-19
『LabVIEW ヘルプ』を参照。
アイコン、7-9
階層、7-11
拡張可能なノード、7-9
現在のインスタンスを調べる、6-10
コピーする、A-3
作成する、7-5、7-10
回避する状。『LabVIEW ヘルプ』を参照。
実行を中断する、6-9
上位 VI からのデータを印刷する、15-6
設計する、7-11
多形性 VI、5-15
動作を制御する、7-4
配置時に名前を表示する。『LabVIEW ヘルプ』を参照。
ヒントラベルの制御器キャプション。『LabVIEW ヘルプ』を参照。
フロントパネル、7-11
フロントパネルデータを取り出す、14-17
呼び出し側 VI のチェーンを表示する、6-10
リモートフロントパネル、18-16
サブ VI のインスタンス
実行を中断する、6-9
調べる、6-10
サブパネル制御器、4-19

サブパレット
ActiveX を作成する、3-8
移動する。『LabVIEW ヘルプ』を参照。
構成する、3-7
作成する。『LabVIEW ヘルプ』を参照。
サブルーチン。「サブ VI」の項を参照。
サポート
技術、D-1
参照する。「表示する」の項も参照。
サンプル、1-4
配列、10-8
1 次元配列、10-8
2 次元配列、10-9
サンプルコード、D-1

し

シーケンスストラクチャ、8-13
「フラットシーケンスストラクチャ」、「スタックシーケンスストラクチャ」の項を参照
過度な使用、8-15
実行順序を制御する、5-26
使用する。『LabVIEW ヘルプ』を参照。
スタックシーケンスと置換、8-16
フラットとスタックを比較する、8-13
ローカル変数とローバル変数を使用してアクセスする、11-4
シーケンスローカル端子、8-15
時間関数、5-8
次元
配列、10-8
指針
追加する、4-11
システムインテグレーションサービス、D-1
システムカラー、4-30
『LabVIEW ヘルプ』を参照。
システムフォント、4-28
システムレベルのコマンド、18-21
システムレベルのコマンドを実行する、18-21
実行
中断する
VI をデバッグする、6-9
ハイライト
VI をデバッグする、6-5
自動プローブ。『LabVIEW ヘルプ』を参照。
データバブルを表示する。『LabVIEW ヘルプ』を参照。
フロー、5-25
シーケンスストラクチャを使用して制御する、8-14
実行可能 VI
デバッグする。『LabVIEW ヘルプ』を参照。
実行速度
制御する、8-10
実行の順序、5-25
シーケンスストラクチャを使用して制御する、8-14
実行のハイライト
VI をデバッグする、6-5
実行フロー、5-25
実行モード
VI を～で開く。『LabVIEW ヘルプ』を参照。
実行モードで VI を開く。『LabVIEW ヘルプ』を参照。
実行を中断する
VI をデバッグする、6-9
実習と手順のディレクトリ
サンプル、A-2
指定されたプローブ、6-7
自動指標付け
For ループ、8-4
While ループ、8-5
デフォルトデータ、6-11
自動定数ラベル
表示する。『LabVIEW ヘルプ』を参照。
自動的にログインする。『LabVIEW ヘルプ』を参照。
自動配線機能、5-12
自動ワイヤルーティング、5-13
シフトレジスタ、8-6
トンネルに入れ替える、8-8
字訳
電子メール送信をする、18-19

修正する
VI、6-2
デバッグ方法、6-3
不良ワイヤ、5-14
集録する
デジタルデータサブセット、4-23
上級関数、5-9
消去する
グラフとチャート。『LabVIEW ヘルプ』を参照。
表示器。『LabVIEW ヘルプ』を参照。
条件端子、8-2
小数点
ローカライズする。『LabVIEW ヘルプ』を参照。
ショートカットメニュー
実行モード時、3-4
ショートメニュー、3-4
所有ラベル、4-26
編集する。『LabVIEW ヘルプ』を参照。
シンク端子。「表示器」の項を参照。
シングルステップ
VI をデバッグする、6-6
人工データ依存、5-26
診断リソース、D-1
シンボリックカラー。「システムカラー」の項を参照。

す

スイープチャート、12-7
数学。「方程式」の項を参照。
数式ノード（Expression Node）、21-3
数値
オーバーフローとアンダーフロー、6-11
関数、5-7
形式、4-11
制御器と表示器、4-10
使用する。『LabVIEW ヘルプ』を参照。
測定の単位、5-22
多形性、B-2
データタイプ（表）、5-2
データをスプレッドシートまたはテキストファイルに書き込む、10-5
範囲外、4-18
比較する、C-1
表記法を変更する。『LabVIEW ヘルプ』を参照。
フォーミュラ、21-1
変換する、B-1
方程式、21-1
文字列と～、10-5
ユニバーサル定数、5-5
数値以外（NaN）の浮動小数点値
不定データ、6-10
数値のアンダーフロー、6-11
数値のオーバーフロー、6-11
スクリプトノード
MATLAB、21-4
スクロールする
グラフ、12-3
チャート、12-3
スクロールバー
非表示にする、4-26
リストボックス、4-15
スケールする
グラフ、12-5
フロントパネルオブジェクト、4-7
スコープチャート、12-7
スタック
バリアントデータ、5-22
スタックシーケンスストラクチャ、8-14
「シーケンスストラクチャ」の項を参照。
フラットシーケンスと置換、8-16
スタックプロット、12-8
スタンドアロンアプリケーション
.NET、19-5
作成する、7-15
VI を配布する、7-14
ステートメント、5-5
ストラクチャ、8-1
For ループ、8-2
While ループ、8-2
イベント、9-2
グローバル変数、11-2
ケース、8-11
削除する。『LabVIEW ヘルプ』を参照。
使用する。『LabVIEW ヘルプ』を参照。
スタックシーケンス、8-14

デバッグする。『LabVIEW ヘルプ』を参照。
配線する。『LabVIEW ヘルプ』を参照。
フラットシーケンス、8-13
ブロックダイアグラム上の、2-4
ローカル変数、11-1
ストリップチャート、12-7
スプレッドシートファイル
作成する、14-8
数値データを～に書き込む、10-5
スペース
フロントパネルまたはブロックダイアグラムに追加する、4-9
スマートプローブ。「プローブ」の項を参照。
スミスプロット、13-3
スムーズアップデート
グラフの、12-6
描画時。『LabVIEW ヘルプ』を参照。
スライダ
追加する、4-10
スライド制御器および表示器、4-10
「数値」の項を参照。
スレッド
マルチを実行する。『LabVIEW ヘルプ』を参照。

せ

制御器、4-1
2 次元、4-9
3 次元、4-9
I/O 名、4-20
refnum、4-25
使用する。『LabVIEW ヘルプ』を参照。
アイコン、5-1
入れ替える、4-2
色を決める、4-5
印刷する、15-3
オートメーション Refnum、19-7
オプション項目を表示する、4-2
回転式、4-10
カラーボックス、4-12
カラーランプ、4-12
キーボードショートカット、4-3
旧バージョン、4-9
クラスタ、4-15
グループ化およびロックする、4-6
サイズ変更する、4-7
ウィンドウサイズに応じて～、4-7
サブ VI のヒントラベルのキャプション。『LabVIEW ヘルプ』を参照。
数値、4-10
使用する。『LabVIEW ヘルプ』を参照。
スタイル、16-2
スライド、4-10
ダイアログ、4-26
使用する。『LabVIEW ヘルプ』を参照。
タイプ定義、4-1
タイムスタンプ、4-11
タブ、4-19
端子
アイコン、5-1
データタイプ、5-1
端子（表）、5-2
データタイプ（表）、5-2
データタイプ端子、5-1
デジタル、4-11
名前を付ける、7-11
任意、7-8
ハイカラー、4-9
配列、4-15
パス、4-14
使用する。『LabVIEW ヘルプ』を参照。
パレット、3-1
カスタマイズする、3-6
操作と検索、3-2
必須、7-8
非表示。『LabVIEW ヘルプ』を参照。
非表示にする
『LabVIEW ヘルプ』を参照。
オプション項目、4-2
表示器に変更する、4-2
ブール、4-12
使用する。『LabVIEW ヘルプ』を参照。

ブロックダイアグラム上に作成する。
『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上の、5-1
フロントパネルでの使用のガイドライン、4-29
文字列、4-13
表、10-2
表示タイプ、10-2
ユーザインタフェースの設計、4-29
ライブラリに追加する、3-6
リストボックス、4-15
使用する。『LabVIEW ヘルプ』を参照。
リング、4-17
使用する。『LabVIEW ヘルプ』を参照。
列挙体、4-18
上級、4-18
使用する。『LabVIEW ヘルプ』を参照。
ローカラー、4-9
制御器と表示器のスタイル、16-2
制御器ライブラリ
VI と制御器を追加する、3-6
制御器リファレンス、17-9
厳密に類別化された、17-9
作成する。『LabVIEW ヘルプ』を参照。
非厳密に類別化された、17-10
制御する
VI をリモートで、18-12
計測器、7-3
サブ VI として呼び出された VI、7-4
ソースコード、7-2
プログラムで VI を、17-1
プログラムでフロントパネルオブジェクトを、17-9
フロントパネルオブジェクトをリモートで。『LabVIEW ヘルプ』を参照。
制御フロープログラミングモデル、5-25
整数
オーバーフローとアンダーフロー、6-11
変換する、B-1
生成する
ユーザイベント、9-13
静的イベント
登録する、9-8
世界各地での技術サポート、D-1
設計する
サブ VI、7-11
ダイアログボックス、4-30
プロジェクト、7-1
ブロックダイアグラム、5-28
フロントパネル、4-29
設計モード
ActiveX コンテナ、19-9
設定。「オプション」の項を参照。
設定する
協力レベル。『LabVIEW ヘルプ』を参照。
作業環境オプション、3-8
『LabVIEW ヘルプ』を参照。
セット、3-7
共有する。『LabVIEW ヘルプ』を参照。
削除する。『LabVIEW ヘルプ』を参照。
作成する、3-7
変更する。『LabVIEW ヘルプ』を参照。
編集する、3-7
セレクタ端子
値、8-12
選択する
オブジェクト。『LabVIEW ヘルプ』を参照。
多形性 VI のデフォルトインスタンス。『LabVIEW ヘルプ』を参照。
ツールを手動で～。『LabVIEW ヘルプ』を参照。
ワイヤ、5-14

そ

操作順序。「タブ順序」の項を参照。
挿入する
配列内の要素。『LabVIEW ヘルプ』を参照。
パレットにオブジェクトを～。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上のオブジェクト。『LabVIEW ヘルプ』を参照。

ソースコード。「ブロックダイアグラム」の項を参照。
ソースコードの制御、7-2
ソース端子。「制御器」の項を参照。
属性
バリアントデータ、5-22
測定単位、5-22
ソフトウェアドライバ、D-1

た

ダイアル
「数値」の項を参照。
カラーランプを追加する、4-12
フロントパネル、4-10
ダイアログ関数、5-8
ダイアログボックス
制御器、4-26
使用する。『LabVIEW ヘルプ』を参照。
設計する、4-30
ネイティブファイル。『LabVIEW ヘルプ』を参照。
表示器、4-26
フォント、4-28
ラベル、4-26
リモートフロントパネル、18-16
リング制御器、4-17
対数関数
多形性、B-7
ダイナミックイベント
サンプル、9-11
登録する、9-8
『LabVIEW ヘルプ』を参照。
ダイナミックデータタイプ、5-19
～から変換する、5-20
『LabVIEW ヘルプ』を参照。
～に変換する、5-21
『LabVIEW ヘルプ』を参照。
タイプ定義、4-1
タイミング
制御する、8-10
タイムスタンプ
「数値」の項を参照。
制御器と表示器、4-11
データタイプ（表）、5-3
多形性
VI
手動でインスタンスを選択する。『LabVIEW ヘルプ』を参照。
ショートカットメニューを編集する。『LabVIEW ヘルプ』を参照。
～からインスタンスを削除する。『LabVIEW ヘルプ』を参照。
～にインスタンスを追加する。『LabVIEW ヘルプ』を参照。
作成する、5-16
関数、B-1
数式ノード（Expression Node）、21-4
単位、B-1
多形性 VI のインスタンス
削除する。『LabVIEW ヘルプ』を参照。
手動で選択する、5-15
追加する。『LabVIEW ヘルプ』を参照。
多形性 VI のインスタンス。「多形性 VI」の項を参照。
タブ順序
設定する、4-4
タブ制御器、4-19
使用する。『LabVIEW ヘルプ』を参照。
単位ラベル、5-22
タンク
「数値」の項を参照。
スライド制御器および表示器、4-10
端子、2-3
印刷する、15-3
カウント、8-2
設定する自動指標付け、8-4
関数から削除する、5-10
関数に追加する、5-10
強制ドット、5-14
繰り返し
For ループ、8-2
While ループ、8-3
検索する。『LabVIEW ヘルプ』を参照。
シーケンスローカル、8-15
条件、8-2
推奨、7-8
制御器および表示器（表）、5-2
セレクタ、8-11
定数、5-4
任意、7-8

配線する、5-10
パターン、7-7
必須、7-8
表示する、5-2
ヒントラベルを表示する。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム、5-1
フロントパネルオブジェクトと～、5-1
ヘルプウィンドウの外観、7-8
短縮メニュー、3-4
端子を接続する、5-10

ち

地域指定小数点。『LabVIEW ヘルプ』を参照。
チャート、12-1
エイリアス除去ラインプロット、12-2
オーバレイプロット、12-8
オプション、12-2
外観をカスタマイズする、12-3
強度、12-12
オプション、12-14
記録の長さ、12-7
作成する。『LabVIEW ヘルプ』を参照。
消去する。『LabVIEW ヘルプ』を参照。
ズームする。『LabVIEW ヘルプ』を参照。
スクロールする、12-3
スケールのフォーマット、12-5
スタックプロット、12-8
タイプ、12-1
追加する。『LabVIEW ヘルプ』を参照。
動作をカスタマイズする、12-7
波形、12-12
複数のスケール、12-2
チャートをスクロールする、12-3
注釈、4-26
編集する。『LabVIEW ヘルプ』を参照。

つ

追加する
VI をライブラリに～、3-6
関数に端子を～、5-10
グラフィックを VI アイコンに。『LabVIEW ヘルプ』を参照。
制御器をライブラリに～、3-6
多形性 VI からのインスタンス。『LabVIEW ヘルプ』を参照。
ディレクトリを VI 検索パスに～。『LabVIEW ヘルプ』を参照。
フロントパネルにスペースを～、4-9
通信、18-1
ActiveX、19-1
AppleEvents、18-21
DataSocket、18-2
Mac OS、18-21
PPC、18-21
System Exec VI、18-21
TCP、18-20
UDP、18-20
UNIX、18-21
VI、7-4
VI サーバ、17-1
関数、7-4
システムレベルのコマンドの実行する、18-21
低レベル、18-20
パイプ、18-21
ファイル I/O、14-1
プロトコル、18-20
通知イベント、9-4
ツール
手動で選択する。『LabVIEW ヘルプ』を参照。
パレット、3-3
ツールセット、1-1
パレット内に、3-8
ツールバー、3-4
ツリー制御器、4-16
使用する。『LabVIEW ヘルプ』を参照。

て

定義する
　エラーコード。『LabVIEW ヘルプ』を参照。
　ユーザカラー。『LabVIEW ヘルプ』を参照。
定数、5-4
　ActiveX でパラメータを設定する、19-12
　作成する。『LabVIEW ヘルプ』を参照。
　配列、10-11
　編集する。『LabVIEW ヘルプ』を参照。
　ユーザ定義、5-5
　ユニバーサル、5-5
ディスクストリーミング、14-7
ディスク容量
　オプション、3-8
　チェックする。『LabVIEW ヘルプ』を参照。
ディレクトリ
　ライブラリに変換する。『LabVIEW ヘルプ』を参照。
　ライブラリを～に変換する。『LabVIEW ヘルプ』を参照。
ディレクトリパス
　「パス」の項を参照。
　「プローブ」の項を参照。
低レベル通信、18-20
データ
　VI から電子メールを送信する、18-17
データ依存、5-26
　競合状態、11-4
　シーケンスストラクチャを使用して制御する、8-14
　人工、5-26
　存在しない、5-26
　フロースルーパラメータ、14-12
データ集録。「DAQ」の項を参照。
データタイプ
　MATLAB （表）、21-5
　.NET、19-5
　XML から変換する、10-7
　XML に変換する、10-7
　印刷する、15-3
　ケースストラクチャ、8-12
　制御器および表示器（表）、5-2
　デフォルト値、5-2
　波形、12-19
データバブル
　実行のハイライト時に表示する。『LabVIEW ヘルプ』を参照。
データフロー
　監視する、6-5
データフロープログラミングモデル、5-25
　メモリを管理する、5-27
データロギング
　自動、14-15
　対話形式、14-15
　プログラムでデータを取り出す、14-17
　レコードを削除する、14-16
　ログファイル連結を消去する、14-16
　ログファイル連結を変更する、14-17
データログファイル I/O、14-4
　ファイルを作成する、14-10
データログレコードを削除する、14-16
データを記録する。「データロギング」の項を参照。
データを取り出す
　サブ VI を使用する、14-17
　ファイル I/O 関数を使用する、14-18
　プログラムで、14-17
テキスト
　形式、4-27
　検索する。『LabVIEW ヘルプ』を参照。
　ドラッグアンドドロップする。『LabVIEW ヘルプ』を参照。
　入力ボックス、4-13
　リング制御器、4-17
テキストファイル
　作成する、14-8
　数値データを～に書き込む、10-5
　ファイル I/O、14-2
　～にドキュメントを保存する、15-3
テキストベースのプログラミング言語からのコード呼び出し、20-1
デジタルグラフ
　エイリアス除去ラインプロット、12-2
　データをマスクする、12-17
デジタル制御器および表示器、4-11
デジタルデータ

圧縮する、4-24
サブセットを集録する、4-23
デジタル波形データタイプ、12-19
パターンを検索する、4-25
連結する、4-24
デジタルデータをマスクする、12-17
サブセットを集録する、4-23
デジタル波形グラフ
デジタルデータ入力を表示する、12-15
プロットを構成する。『LabVIEW ヘルプ』を参照。
デジタル波形データタイプ、12-19
デバッグする
MATLAB スクリプト、21-6
オプション、3-8
壊れた VI、6-2
実行可能 VI。『LabVIEW ヘルプ』を参照。
自動エラー処理、6-12
ストラクチャ。『LabVIEW ヘルプ』を参照。
ツール
無効にする、6-10
デバッグツールを無効にする、6-10
デフォルトデータ、6-11
非表示のワイヤ、5-28
不定データ、6-10
プローブ、6-6
作成する。『LabVIEW ヘルプ』を参照。
プローブを作成する。『LabVIEW ヘルプ』を参照。
方法、6-3
エラー処理、6-12
実行のハイライト、6-5
実行を中断する、6-9
シングルステップ、6-6
ブレークポイントツール、6-8
プローブツール、6-6
ブロックダイアグラムのセクションをコメントアウトする、6-10
ループ回数、6-11
デフォルトケース、8-11
デフォルト値
データタイプ、5-2
デフォルトデータ
For ループ、6-11
ディレクトリ、14-19
配列、6-11
デフォルトファンクションキー設定を無視する。『LabVIEW ヘルプ』を参照。
デフォルトプローブ、6-7
電子メール
VI から送信する、18-17
字訳、18-19
文字セット、18-18
転送制御プロトコル、18-20
伝送ライン、13-3
伝送ラインのインピーダンス、13-3
テンプレート
作成する。『LabVIEW ヘルプ』を参照。
使用する。『LabVIEW ヘルプ』を参照。
点滅速度。『LabVIEW ヘルプ』を参照。
電話による技術サポート、D-1

と

透明
オブジェクト。『LabVIEW ヘルプ』を参照。
ラベル。『LabVIEW ヘルプ』を参照。
登録解除する
ダイナミックイベント、9-10
ユーザイベント、9-14
登録する
イベントを静的に～、9-8
イベントを動的に～、9-8
『LabVIEW ヘルプ』を参照。
ユーザイベント、9-13
『LabVIEW ヘルプ』を参照。
ドキュメント
PDF ライブラリ、1-1
ガイド、1-1
ディレクトリ構造、A-2
他のリソースとともに使用する、1-1
本書で使用する表示規則、xx
オンラインライブラリ、D-1
本書の概要、xix
本書の構成、xx
本書の使用方法、xix

ドット
強制、5-14
ドライバ
計測器、D-1
LabVIEW。『LabVIEW ヘルプ』を参照。
ソフトウェア、D-1
ドラッグアンドドロップする。『LabVIEW ヘルプ』を参照。
トラブルシューティング。「デバッグする」の項を参照。
トラブルシューティングリソース、D-1
取り消しオプション、3-8
トレーニング
カスタマー、D-1
ドロップスルークリック。『LabVIEW ヘルプ』を参照。
トンネル、8-1
シフトレジスタに入れ替える、8-8
入力と出力、8-13

な

ナショナルインスツルメンツ
カスタマートレーニング、D-1
技術サポート、D-1
システムインテグレーションサービス、D-1
世界各地の営業所、D-1
プロフェッショナルサービス、D-1
ナショナルインスツルメンツへのお問い合わせ、D-1
名前を付ける
VI、7-13
制御器、7-11

に

入門、1-1
入力完成機能、4-15
リストボックス、4-15

ね

ネイティブファイルダイアログボックス。『LabVIEW ヘルプ』を参照。
ネットワーク動作。「通信」の項を参照。

の

ノード、2-4
MATLAB スクリプトノード、21-4
インボーク、17-5
サイズ変更する。『LabVIEW ヘルプ』を参照。
実行フロー、5-26
ブロックダイアグラム、5-5
プロパティ、17-4
リファレンス呼び出し、17-7
ノブ
「数値」の項を参照。
カラーランプを追加する、4-12
フロントパネル、4-10

は

バージョン
旧バージョンの形式で VI を保存する、7-14
最後に保存された～に戻す。『LabVIEW ヘルプ』を参照。
比較する、7-2
配線する
ガイド。『LabVIEW ヘルプ』を参照。
自動的に、5-12
手動、5-10、5-12
ストラクチャ。『LabVIEW ヘルプ』を参照。
単位、5-23
ツール、5-12
方法、5-28
バイトストリームファイル、14-4
バイナリ
ファイル I/O、14-3
ファイルを作成する、14-9
浮動小数点演算、6-11
パイプ通信、18-21
配列
移動する。『LabVIEW ヘルプ』を参照。
関数、5-8
クラスタと配列の間で変換する。『LabVIEW ヘルプ』を参照。
グローバル変数、11-5

サイズ変更する。『LabVIEW ヘルプ』を参照。
作成する、10-11
サンプル
1 次元配列、10-8
2 次元配列、10-9
次元、10-8
指標、10-8
表示、10-11
制御器と表示器、4-15
データタイプ（表）、5-3
制約、10-10
多形性、B-4
定数、10-11
デフォルトデータ、6-11
比較する、C-2
要素を入れ替える。『LabVIEW ヘルプ』を参照。
要素を削除する。『LabVIEW ヘルプ』を参照。
要素を挿入する。『LabVIEW ヘルプ』を参照。
ループに自動指標付けを行う、8-4
ループを使用して～を作成する、8-5
～のサイズ、6-11
配列の指標、10-8
表示、10-11
波形
関数、5-9
グラフ、12-9
データタイプ、12-9、12-10
グラフィック、13-2
制御器と表示器
データタイプ（表）、5-3
チャート、12-12
データタイプ、12-19
ファイルから読み取る、14-11
ファイルに書き込む、14-10
パス
オプション、3-8
空、4-14
制御器と表示器、4-14
使用する。『LabVIEW ヘルプ』を参照。
データタイプ（表）、5-3
ディレクトリを VI 検索パスに追加する。『LabVIEW ヘルプ』を参照。
ファイル I/O、14-6
無効、4-14
ユニバーサル定数、5-5
リモートフロントパネル、18-16
パスワード保護、7-14
パターン
端子、7-7
バックパネル。「ブロックダイアグラム」の項を参照。
発呼者
表示する、6-10
～のチェーン、6-10
バッファに格納されたデータ
DataSocket、18-8
ローカル変数、11-5
パフォーマンス
オプション、3-8
デバッグツールを無効にする、6-10
ローカル変数とグローバル変数、11-4
パラメータ
データタイプ（表）、5-2
パラメータリスト。「コネクタペーン」の項を参照。
バリアントデータ
ActiveX、19-7
DataSocket、18-9
処理、5-21
制御器と表示器
データタイプ（表）、5-4
属性、5-22
属性を編集する。『LabVIEW ヘルプ』を参照。
～に変換する、5-22
平坦化されたデータ、5-22
貼り付け
グラフィック、4-5
パレット
オブジェクトを挿入する。『LabVIEW ヘルプ』を参照。
オプション、3-8
カスタマイズする、3-6
カラーパレット、4-12

関数、3-2
カスタマイズする、3-6
共有する。『LabVIEW ヘルプ』を参照。
更新する。『LabVIEW ヘルプ』を参照。
構成する、3-7
制御器、3-1
カスタマイズする、3-6
セット、3-7
操作と検索、3-2
ツール、3-3
ツールセット、3-8
変更する。『LabVIEW ヘルプ』を参照。
モジュール、3-8
リファレンス。『LabVIEW ヘルプ』を参照。
パレットセットを変更する。『LabVIEW ヘルプ』を参照。
パレットを更新する。『LabVIEW ヘルプ』を参照。
範囲外の数値、4-18

ひ

比較関数、C-1
多形性、B-6
比較する
VI のバージョン、7-2
クラスタ、C-2
数値、C-1
配列、C-2
ブール値、C-1
文字列、C-1
引き延ばす。「サイズを変更する」の項を参照。
ピクチャ。「グラフィック」の項を参照。
ピクチャ制御器および表示器
使用する、13-1
データタイプ（表）、5-4
ピクチャリング制御器、4-17
非厳密に類別化された制御器リファレンス、17-10
ピックスマップ、13-4
ビットマップファイル、13-6
非表示にする
スクロールバー、4-26
フロントパネルオブジェクト。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクト内のオプション項目、4-2
メニューバー、4-26
非表示のフロントパネルオブジェクト。『LabVIEW ヘルプ』を参照。
表、4-16、10-2
使用する。『LabVIEW ヘルプ』を参照。
描画する
「グラフィック」の項も参照。
スムーズアップデート。『LabVIEW ヘルプ』を参照。
表示器、4-1
refnum
使用する。『LabVIEW ヘルプ』を参照。
2 次元、4-9
3 次元、4-9
ActiveX、19-7
I/O 名、4-20
refnum、4-25
アイコン、5-1
入れ替える、4-2
色を決める、4-5
印刷する、15-3
オプション項目を表示する、4-2
回転式、4-10
カラーボックス、4-12
カラーランプ、4-12
旧バージョン、4-9
クラスタ、4-15
グループ化およびロックする、4-6
サイズ変更する、4-7
ウィンドウサイズに応じて～、4-7
消去する。『LabVIEW ヘルプ』を参照。
数値、4-10
使用する。『LabVIEW ヘルプ』を参照。
スタイル、16-2
スライド、4-10
制御器に変更する、4-2
ダイアログ、4-26
タイプ定義、4-1
タイムスタンプ、4-11

タブ、4-19
端子
アイコン、5-1
データタイプ、5-1
端子（表）、5-2
データタイプ（表）、5-2
データタイプ端子、5-1
デジタル、4-11
任意、7-8
ハイカラー、4-9
配列、4-15
パス、4-14
使用する。『LabVIEW ヘルプ』を参照。
必須、7-8
非表示。『LabVIEW ヘルプ』を参照。
非表示にする
『LabVIEW ヘルプ』を参照。
オプション項目、4-2
ブール、4-12
使用する。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上に作成する。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上の、5-1
フロントパネルでの使用のガイドライン、4-29
文字列、4-13
表示タイプ、10-2
ユーザインタフェースの設計、4-29
列挙体
上級、4-18
ローカラー、4-9
表示する
エラー、6-2
警告、6-3
自動定数ラベル。『LabVIEW ヘルプ』を参照。
端子、5-2
ヒントラベル。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクト内のオプション項目、4-2
フロントパネルをリモートで、18-12
呼び出し側 VI のチェーン、6-10
非表示のフロントパネルオブジェクト。『LabVIEW ヘルプ』を参照。
ヒントラベル
作成する、15-2
制御器キャプション。『LabVIEW ヘルプ』を参照。
端子上で表示する。『LabVIEW ヘルプ』を参照。
表示する。『LabVIEW ヘルプ』を参照。

ふ

ファイル I/O
.lvm ファイル、14-20
LabVIEW データ形式、14-20
refnum、14-1
下位 VI および関数、14-6
関数、5-9
基本操作、14-1
形式、14-2
構成ファイル VI
.ini ファイルの読み書き、14-12
形式、14-13
目的、14-13
上位 VI、14-5
上級ファイル関数、14-6
スプレッドシートファイル
作成する、14-8
ディスクストリーミング、14-7
データログファイル、14-4
作成する、14-10
テキストファイル、14-2
作成する、14-8
デフォルトディレクトリを選択する。『LabVIEW ヘルプ』を参照。
デフォルトデータディレクトリ、14-19
ネットワーク動作、18-1
バイトストリームファイル、14-4
バイナリファイル、14-3
作成する、14-9
波形を書き込む、14-10
波形を読み取る、14-11
パス、14-6
フロースルーパラメータ、14-12

フロントパネルのデータを記録する、14-15
ファイル I/O 形式
データログファイル、14-4
テキストファイル、14-2
バイナリファイル、14-3
ファイルから読み取る、14-1
ファイルに書き込む、14-1
ファイルの共有、7-2
ファイルの保存場所、A-3
ファイルを保存する
推奨場所、A-3
ファンクションキー設定
デフォルトを無視する。『LabVIEW ヘルプ』を参照。
フィードバックノード、8-9
シフトレジスタに入れ替える、8-10
初期化する、8-10
フィルタイベント、9-4
ブール関数、5-7
多形性、B-4
ブール制御器および表示器、4-12
値を比較する、C-1
使用する。『LabVIEW ヘルプ』を参照。
データタイプ（表）、5-3
フォーミュラ。「方程式」の項を参照。
フォーミュラノード、21-2
C 言語に似たステートメントを入力する、21-2
図、21-2
変数、21-3
方程式を入力する、21-2
フォント
アプリケーション、4-28
オプション、3-8
システム、4-28
設定、4-27
ダイアログ、4-28
複数列リストボックス。「リストボックス制御器」の項を参照。
不定データ
Inf（無限大）、6-10
NaN（数値以外）、6-10
配列、6-11
防ぐ、6-12
～をチェックする、6-11
浮動小数点数
オーバーフローとアンダーフロー、6-11
変換する、B-1
フロントパネル上のプルダウンメニュー、4-17
フリーラベル、4-26
作成する。『LabVIEW ヘルプ』を参照。
不良ワイヤ、5-14
フルメニュー、3-4
ブレークポイントツール
VI をデバッグする、6-8
ブレークポイントをハイライトする。『LabVIEW ヘルプ』を参照。
フロースルーパラメータ、14-12
プローブ
VI をデバッグする、6-6
一般、6-6
カスタム、6-8
作成する。『LabVIEW ヘルプ』を参照。
作成する。『LabVIEW ヘルプ』を参照。
指定された、6-7
デフォルト、6-7
表示器、6-7
～のタイプ、6-6
プローブツール
「プローブ」の項を参照。
VI をデバッグする、6-6
プログラミング例、D-1
プログラム間の通信、18-21
プロジェクト計画、7-1
プロジェクト設計、7-1
プロジェクトを計画する、7-1
ブロックダイアグラム、2-2
DataSocket、18-6
VI サーバ、17-1
印刷する、15-5
オブジェクト、5-1
オブジェクトを入れ替える。『LabVIEW ヘルプ』を参照。
オブジェクトを均等に配置する、4-6
『LabVIEW ヘルプ』を参照。

オブジェクトをコピーする。『LabVIEW ヘルプ』を参照。
オブジェクトを削除する。『LabVIEW ヘルプ』を参照。
オブジェクトを挿入する。『LabVIEW ヘルプ』を参照。
オブジェクトを等間隔に配置する、4-6
『LabVIEW ヘルプ』を参照。
オブジェクトを並び替える。『LabVIEW ヘルプ』を参照。
オブジェクトを整列させる、4-6
『LabVIEW ヘルプ』を参照。
オプション、3-8
関数、5-7
強制ドット、5-14
計画する、7-1
サイズ変更せずにスペースを追加する、5-28
自動配線する、5-12
手動配線する、5-10、5-12
ストラクチャ、8-1
使用する。『LabVIEW ヘルプ』を参照。
制御器と表示器を作成する。『LabVIEW ヘルプ』を参照。
セクションをコメントアウトする、6-10
設計する、5-28
ソースコードを制御する、7-2
端子
関数から削除する、5-10
関数に追加する、5-10
制御器および表示器（表）、5-2
表示する、5-2
フロントパネルオブジェクトと～、5-1
端子を検索する。『LabVIEW ヘルプ』を参照。
定数、5-4
データタイプ（表）、5-2
データフロー、5-25
ノード、5-5
パスワード保護する、7-14
バリアントデータ、5-21
フォント、4-27
ラベル、4-26
サイズ変更する。『LabVIEW ヘルプ』を参照。
作成する。『LabVIEW ヘルプ』を参照。
編集する。『LabVIEW ヘルプ』を参照。
ブロックダイアグラム上で手動配線する、5-12
ブロックダイアグラムのセクションをコメントアウトする
VI をデバッグする、6-10
プロット
エイリアス除去、12-2
オーバレイ、12-8
グラフに追加する。『LabVIEW ヘルプ』を参照。
スタック、12-8
プロトコル
DataSocket、18-3
低レベル通信、18-20
プロパティ
ActiveX
参照する、19-10
設定する
プログラムで、19-10
プロパティノード、17-4
ActiveX、19-10
オブジェクトまたは端子を検索する。『LabVIEW ヘルプ』を参照。
リストボックス項目を変更する、4-15
プロパティを、19-6
プロフェッショナルサービス、D-1
フロントパネル、2-1
ActiveX を使用してオブジェクトを挿入する、19-9
DataSocket、18-5
イベントを使用してロックする、9-7
印刷する、15-5
VI 実行後、15-6
ウィンドウサイズを定義する『LabVIEW ヘルプ』を参照。
ウェブ上に画像をパブリッシュする、18-12

オブジェクト
ブロックダイアグラム端子と～、5-1
オブジェクトを検索する。『LabVIEW ヘルプ』を参照。
オブジェクトの色を決める、4-5
色をコピーする。『LabVIEW ヘルプ』を参照。
背景と前景。『LabVIEW ヘルプ』を参照。
オブジェクトの順序、4-4
オブジェクトを入れ替える、4-2
オブジェクトを重ねる、4-19
オブジェクトを均等に配置する、4-6
『LabVIEW ヘルプ』を参照。
オブジェクトをグループ化およびロックする、4-6
オブジェクトをコピーする。『LabVIEW ヘルプ』を参照。
オブジェクトをサイズ変更する、4-7
ウィンドウサイズに応じて～、4-7
オブジェクトを削除する。『LabVIEW ヘルプ』を参照。
オブジェクトをスケールする、4-7
オブジェクトを等間隔に配置する、4-6
『LabVIEW ヘルプ』を参照。
オブジェクトを並び替える。『LabVIEW ヘルプ』を参照。
オブジェクトをリモートで制御する。『LabVIEW ヘルプ』を参照。
オブジェクトを整列させる、4-6
『LabVIEW ヘルプ』を参照。
オプション、3-8
オプションのオブジェクト項目を表示する、4-2
キーボードショートカット、4-3
キャプション、4-27
作成する。『LabVIEW ヘルプ』を参照。
グラフィックをインポートする、4-5
計画する、7-1
異なる画面解像度で表示する、4-30
サイズ変更せずにスペースを追加する、4-9
サブ VI、7-11
サブパネル制御器でロードする、4-19
制御器、4-9
制御器と表示器のスタイル、16-2
制御器を表示器に、表示器を制御器に変更する、4-2
設計する、4-29
タイプ定義、4-1
タブ順序を設定する、4-4
データロギング、14-15
データを記録する、14-15
データを取り出す
サブ VI を使用する、14-17
ファイル I/O 関数を使用する、14-18
テキストの特性、4-27
透明なオブジェクト。『LabVIEW ヘルプ』を参照。
非表示にする
オブジェクト。『LabVIEW ヘルプ』を参照。
オプションオブジェクト項目、4-2
非表示のオブジェクト。『LabVIEW ヘルプ』を参照。
表示器、4-9
表示器を消去する。『LabVIEW ヘルプ』を参照。
フォント、4-27
プログラムでオブジェクトを制御する、17-9
ラベル、4-26
サイズ変更する。『LabVIEW ヘルプ』を参照。
作成する。『LabVIEW ヘルプ』を参照。
編集する。『LabVIEW ヘルプ』を参照。
リモートで参照する、18-12
リモートで制御する、18-12
フロントパネルオブジェクトの前景色。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクトの背景色。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクトを重ねる、4-19
フロントパネルオブジェクトをグループ解除する、4-6
フロントパネルオブジェクトをタブで移動する、4-4

フロントパネル上のスイッチ、4-12
フロントパネル上のライト、4-12
フロントパネルの、4-1
フロントパネルの静止画像、18-12
フロントパネルの動画、18-12
フロントパネルまたはブロックダイアグラム上のオブジェクトのクローンを作成する。『LabVIEW ヘルプ』を参照。
分割する
　文字列。『LabVIEW ヘルプ』を参照。

へ

平坦化されたデータ
　バリアントデータ、5-22
ヘルプ
　「ヘルプウィンドウ」の項を参照。
　技術サポート、D-1
　プロフェッショナルサービス、D-1
ヘルプウィンドウ、3-5
　VI の説明を作成する、15-2
　オブジェクトの説明を作成する、15-2
　端子の外観、7-8
ヘルプファイル、1-2
　HTML、15-5
　RTF、15-5
　実行可能 VI。『LabVIEW ヘルプ』を参照。
　独自に作成する、15-5
変換する
　LabVIEW データタイプを HTML に、10-7
　XML から LabVIEW データに、10-7
　クラスタと配列の間で変換する。『LabVIEW ヘルプ』を参照。
　数値を文字列に、10-5
　ディレクトリをライブラリに～。『LabVIEW ヘルプ』を参照。
　ライブラリをディレクトリに～。『LabVIEW ヘルプ』を参照。
編集する
　多形性 VI のショートカットメニュー。『LabVIEW ヘルプ』を参照。
　パレットセット、3-7
　メニュー、16-2
　ラベル。『LabVIEW ヘルプ』を参照。
変数
　グローバル
　　競合状態、11-4
　　作成する、11-2
　　初期化する、11-4
　　慎重に使用する、11-4
　　メモリ、11-5
　　読み取りと書き込み、11-3
　ローカル、11-1
　　競合状態、11-4
　　作成する、11-2
　　初期化する、11-4
　　慎重に使用する、11-4
　　メモリ、11-5
　　読み取り変数と書き込み変数、11-3

ほ

方程式
　MATLAB
　　スクリプトノード、21-4
　　スクリプトをデバッグする、21-6
　LabVIEW に統合する、21-1
　使用する方法、21-1
　数式ノード（Expression Node）、21-3
　フォーミュラノード、21-2
方程式を計算する、21-1
ポータブルネットワーク画像ファイル、13-6
ボタン
　キーボードショートカットで制御する、4-4
　フロントパネル、4-12
ホットメニュー。『LabVIEW ヘルプ』を参照。
ポップアップメニュー。「ショートカットメニュー」の項を参照。
本書で使用する表記規則、xx

ま

マッピング
　文字、18-19
マニュアル。「ドキュメント」の項を参照。
マルチスレッド
　実行する。『LabVIEW ヘルプ』を参照。

む

無限 While ループ、8-3
無効なパス、4-14
無効にする
　一時的な自動ワイヤルーティング、5-13
　デバッグツール、6-10
　ブロックダイアグラムのセクション
　　VI をデバッグする、6-10
無効パス、4-14

め

メーター
　「数値」の項を参照。
　カラーランプを追加する、4-12
　フロントパネル、4-10
メソッド
　ActiveX、19-6
メニュー、3-4
　コンボボックス、4-13
　ショートカット、3-4
　　多形性 VI の～を編集する。
　　『LabVIEW ヘルプ』を参照。
　選択を処理する、16-3
　短縮、3-4
　編集する、16-2
　ホット。『LabVIEW ヘルプ』を参照。
　リファレンス。『LabVIEW ヘルプ』
　　を参照。
　リング制御器、4-17
メニューバー
　非表示にする、4-26
メニュー編集、16-2
メモリ
　圧縮する。『LabVIEW ヘルプ』を参照。
　解放する。『LabVIEW ヘルプ』を参照。
　強制ドット、5-14
　グローバル変数、11-5
　データフロープログラミングモデルを使
　　用して管理する、5-27
　デバッグツールを無効にする、6-10
　バリアントデータを使用した読み書
　　き、5-22
　ローカル変数、11-5
メモリを圧縮する。『LabVIEW ヘルプ』
を参照。
メモリを解放する。

も

文字セット
　ASCII、18-18
　ISO Latin-1、18-18
　Mac OS、18-19
　電子メールで使用する、18-18
文字のフォーマット、4-27
モジュール
　パレット内に、3-8
文字列、10-1
　関数、5-7
　グローバル変数、11-5
　形式、10-4
　　指定子、10-4
　形式文字列を使用する。『LabVIEW ヘル
　　プ』を参照。
　コンボボックス、4-13
　数値～、10-5
　制御器と表示器、4-13
　　表示タイプ、10-2
　多形性、B-5
　テキストを入れ替える。『LabVIEW ヘル
　　プ』を参照。
　比較する、C-1
　表、10-2
　プログラム的に編集する、10-3
　分割する。『LabVIEW ヘルプ』を参照。
　ユニバーサル定数、5-5
文字列をフォーマットする。『LabVIEW ヘル
プ』を参照。

ゆ

ユーザイベント
　『LabVIEW ヘルプ』を参照。
　作成する、9-13
　サンプル、9-14
　生成する、9-13
　登録解除する、9-14
　登録する
　　『LabVIEW ヘルプ』を参照。

ユーザインターフェース。「フロントパネル」の項を参照。
ユーザ定義エラーコード。『LabVIEW ヘルプ』を参照。
ユーザ定義定数、5-5
ユーザ定義の色。『LabVIEW ヘルプ』を参照。
ユーザデータグラムプロトコル、18-20
ユーザプローブ。「プローブ」の項を参照。
ユーザライブラリ
　VI と制御器を追加する、3-6
ユニバーサル定数、5-5

よ

予想外のデータ。「デフォルトデータ」および「不定データ」の項を参照。
呼び出し側 VI のチェーン
　表示する、6-10
読み取りグローバル、11-3
読み取りローカル、11-3

ら

ライブラリ
　VI、A-1
　VI と制御器を追加する、3-6
　管理する、7-13
　共有、7-15
　　VI を配布する、7-14
　計測器、A-1
　構成、A-1
　最上位 VI としてマークを付ける。『LabVIEW ヘルプ』を参照。
　ディレクトリ構造、A-1
　ディレクトリに変換する。『LabVIEW ヘルプ』を参照。
　ディレクトリを～に変換する。『LabVIEW ヘルプ』を参照。
　ユーザ、A-1
　～から VI を削除する。
　～として VI を保存する、7-12
　　推奨場所、A-3
ライブラリ内の最上位 VI としてマークを付ける。『LabVIEW ヘルプ』を参照。
ライブラリ関数呼び出しノード、20-1
ラインプロット
　エイリアス除去、12-2
ラベル
　自動定数を表示する。『LabVIEW ヘルプ』を参照。
　ダイアログボックス、4-26
　透明。『LabVIEW ヘルプ』を参照。
ラベルを付ける
　キャプション、4-27
　グローバル変数、11-3
　サイズ変更する。
　測定の単位、5-22
　定数、5-4
　フォント、4-27
　フリーラベルを作成する。『LabVIEW ヘルプ』を参照。
　編集する。『LabVIEW ヘルプ』を参照。
　ローカル変数、11-2
ランタイムメニューファイル、16-2

り

リストする。「表示する」の項も参照。
リストボックス、4-15
リストボックス制御器、4-15
　使用する。『LabVIEW ヘルプ』を参照。
リファレンス呼び出しノード、17-7
リモートで VI を呼び出す、17-1
リモートフロントパネルのライセンス、18-13
リング制御器、4-17
　使用する。『LabVIEW ヘルプ』を参照。

る

ループ回数
　For、8-2
　While、8-2
　自動指標付け、8-4
　シフトレジスタ、8-6

使用する。『LabVIEW ヘルプ』を参照。
タイミングを制御する、8-10
デフォルトデータ、6-11
配列を作成する、8-5
無限、8-3
ループに指標を付ける、8-4
For ループ、8-4
While ループ、8-5

れ

例外の制御。『LabVIEW ヘルプ』を参照。
レコード、14-15
削除する、14-16
サブ VI を使用してフロントパネルデータを取り出している間に指定する、14-18
列挙体制御器、4-18
上級、4-18
使用する。『LabVIEW ヘルプ』を参照。
データタイプ（表）、5-3
列挙体、4-18
上級、4-18
レビジョン番号
タイトルバーに表示する。『LabVIEW ヘルプ』を参照。
レビジョン履歴
印刷する、15-3
作成する、15-1
数値、15-2
レポート
印刷する、15-7
生成する、15-7
エラークラスタ。『LabVIEW ヘルプ』を参照。
レポート生成 VI、15-7
レポートを生成する、15-7
エラークラスタ。『LabVIEW ヘルプ』を参照。

ろ

ローカル変数、11-1
オブジェクトまたは端子を検索する。『LabVIEW ヘルプ』を参照。
競合状態、11-4
作成する、11-2
初期化する、11-4
慎重に使用する、11-4
メモリ、11-5
読み取りと書き込み、11-3
ログファイル連結、14-15
消去する、14-16
変更する、14-17
ロック解除する
VI。『LabVIEW ヘルプ』を参照。
フロントパネルオブジェクト、4-6
ヘルプウィンドウ、3-5
ロックする
VI。『LabVIEW ヘルプ』を参照。
イベントを使用してフロントパネルを～、9-7
フロントパネルオブジェクト、4-6
ヘルプウィンドウ、3-5

わ

ワイヤ、2-4
移動する。『LabVIEW ヘルプ』を参照。
選択する、5-14
不良、5-14
ルーティング、5-13
ワイヤのルーティング、5-13
割り当てる
ブロックダイアグラムにパスワードを、7-14