LabVIEW™

Gプログラミング
リファレンスマニュアル

NATIONAL INSTRUMENTS™

1998年7月版
製品番号 321993A-01

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

Fax-on-Demand サポート（米国）
512 418 1111

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

海外オフィス
イスラエル 03 6120092、イタリア 02 413091、英国 01635 52354、オーストラリア 03 9879 5166
オーストリア 0662 45 79 90 0、オランダ 0348 433466、カナダ（オンタリオ） 905 785 0085
カナダ（ケベック） 514 694 8521、韓国 02 596 7456、シンガポール 2265886、スイス 056 200 51 51
スウェーデン 08 730 49 70、スペイン 91 640 0085、台湾 02 377 1200、デンマーク 45 76 26 00
ドイツ 089 741 31 30、ノルウェー 32 84 84 00、フィンランド 09 725 725 11、フランス 01 48 14 24 24
ベルギー 02 757 00 20、ブラジル 011 288 336、香港 2645 3186、メキシコ 5 520 2635

ナショナルインスツルメンツ米国本社
11500 North Mopac Expressway Austin, Texas 78759 USA　Tel: 512 794 0100

日本ナショナルインスツルメンツ（株）
〒105-0011 東京都港区芝公園 2-4-1 秀和芝パークビル B 館 5F　Tel：03-5472-2970

必ずお読みください

保証

お買い上げいただいたナショナルインスツルメンツのソフトウェア媒体は、材料およびソフトウェア製造上の欠陥によるプログラミング命令の実行不能に対して、ソフトウェアの受領書または他の文書（登録カード）によって立証された出荷日から 90 日間の保証が適用されます。上記保証期間中に、ソフトウェア媒体がプログラミング命令を実行しない欠陥がある旨の通知がナショナルインスツルメンツに対して行われた場合、ナショナルインスツルメンツは弊社の責任でそのソフトウェアを修理または交換します。ただし、ナショナルインスツルメンツはソフトウェア操作の中断や、エラー発生に関しては保証しません。

保証の対象としてお受けするには、いかなる機器についても工場から返品確認（RMA）番号を取得し、パッケージの外部にその番号を明記していただく必要があります。保証期間内の部品を所有者に返却する際の費用は、ナショナルインスツルメンツが負担します。

本書の内容については万全を期しており、技術面でのチェックも入念に行っております。技術上または印刷上の誤りがあった場合、ナショナルインスツルメンツは本書をお持ちのお客様に事前に通告することなく次回以降の版に変更を加える権利を有します。本書で誤りなどお気づきの点がありましたら弊社までご連絡ください。ナショナルインスツルメンツは、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。

ナショナルインスツルメンツは、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。ナショナルインスツルメンツの過失または不注意により発生した損害に対するユーザの賠償請求権は、ユーザが製品に支払われた金額を上限とします。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は、ナショナルインスツルメンツは、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。このナショナルインスツルメンツの限定責任は、契約が遵守された場合でも、契約に違反した場合でも不注意の場合でも、訴訟方式に関係なく適用されます。ナショナルインスツルメンツに対する訴訟は、訴訟の原因の発生より 1 年以内に申し立てる必要があります。ナショナルインスツルメンツは、妥当な管理限界を越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。ここに定めた保証では、インストール、操作、保守に関連するナショナルインスツルメンツの指示をユーザが守らなかったために生じた損害、欠陥、誤動作、動作故障は対象となりません。さらに、ユーザが製品を改造した場合や、ユーザによる乱用、誤操作、不注意の場合、および停電、電源サージ、火災、洪水、事故、第三者の行為、その他予期せぬ事象も、本保証の対象とはなりません。

著作権

商標

ナショナルインスツルメンツ社製品を医療および臨床用に使用する際の注意

ナショナルインスツルメンツの製品は、人体の治療や診断の用途に適した信頼性を保証することを目的とした部品および試験を使用して設計されておりません。ナショナルインスツルメンツの製品を医療用または臨床用に使用した場合、製品の故障またはユーザやアプリケーション設計者の過失により、身体に害を及ぼす可能性があります。ナショナルインスツルメンツの製品のアプリケーションを医療用または臨床用として使用する場合は、適切な訓練と資格を有する医療専門家が行うものとします。また当該製品の使用に際しては、重大な人身事故や死亡の危険を避けるため、従来医療用安全対策、機器、および手順を引き続き適用してください。ナショナルインスツルメンツの製品は、人体の健康や医療および臨床治療における安全性を保護、監視するための通常プロセス、手順、あるいは機器に代わるものではありません。

目次

本書について

第I部
Gの基本概念

第1章
Gプログラミングの概要

第2章
VIを編集する

第3章
サブ VI を使用する

第4章
VIとサブVIの実行およびデバッグ

第5章
VIの印刷および文書作成

第6章
VIおよびサブVIをセットアップする

第7章
環境をカスタマイズする

第II部
フロントパネルオブジェクト

第8章
フロントパネルオブジェクトの概要

第9章

数値制御器と数値表示器

第10章

ブール制御器とブール表示器

第11章
文字列制御器と文字列表示器

第12章
パスとRefnumの制御器および表示器

第13章
リストとリングの制御器および表示器

第14章
配列とクラスタの制御器および表示器

第15章
グラフとチャートの制御器および表示器

第18章
ブロックダイアグラムを配線する

第19章
ストラクチャ

第20章
フォーミュラノード

第21章
VIサーバ

第22章
属性ノード

第23章
グローバル変数とローカル変数

第IV部
上級トピック

第24章
カスタム制御器とType Def

第25章
他の言語で書かれたコードを呼び出す

第26章
Gの実行システムについて

第27章
アプリケーションを管理する

第28章
パフォーマンスについて

付録、用語集、索引

図、表

図一覧

表一覧

本書について

『G プログラミングリファレンスマニュアル（本書）』では、バーチャルインスツルメンツ（VI：仮想計測器）の作成について解説します。データ入出力用のインタフェース、LabVIEW の VI を使用して解析処理を行う方法、LabVIEW でのネットワーク処理やアプリケーション間通信についても説明します。本書をお使いになる前に『LabVIEW リリースノート』をお読みください。

『G プログラミングリファレンスマニュアル（本書）』は、G プログラミング言語を利用してバーチャルインスツルメンツ（VI：仮想計測器）の作成、編集、実行方法を解説したものです。本書では、フロントパネルおよびブロックダイアグラム、数値、ブール、文字列、配列、クラスタの各制御器および表示器、グラフおよびチャート、ブロックダイアグラムの配線について説明します。

本書の構成

本書は、4 つのパートで構成されています。第 1 章から第 7 章では、G の基本概念について説明します。第 8 章から第 16 章では、フロントパネルのオブジェクトの使用方法について説明します。第 17 章から第 23 章では、ブロックダイアグラムのプログラミングオブジェクトとその使用方法について説明します。第 24 章から第 27 章では、G の高度なテクニックについて説明します。

第 I 部　G の基本概念

第 I 部では、G プログラミングに関する基本的な情報、バーチャルインスツルメンツ（VI）の作成方法、編集、およびカスタマイズについて説明します。

「第 I 部　G の基本概念」の各章の内容は以下の通りです。

- 「第 1 章　G プログラミングの概要」では、G のユニークなプログラミング方法について説明します。また、G を使用してプログラム開発を始める方法についても説明します。
- 「第 2 章　VI を編集する」では、VI の作成や使用に関する基本的な機能、およびパレットやメニューについて説明します。また、オブジェクトの作成、ツールの交換、VI の呼び出し、実行、保存といった基本的な操作の方法についても説明します。

- 「第3章 サブVIを使用する」では、Gアプリケーションの階層設計の概念について説明するとともに、サブVIの２通りの作成方法を示します。また、2つのユーティリティ、すなわちVIの階層を表示する階層ウィンドウと、サブVIやその他のオブジェクトあるいはテキストの文字列を検索する検索ユーティリティについても説明します。
- 「第4章 VIとサブVIの実行およびデバッグ」では、VIの操作とデバッグ、および特殊な実行モードで使用するVIやサブVIのセットアップについて説明します。
- 「第5章 VIの印刷および文書作成」では、VIの印刷および文書作成に関する様々なことがらについて説明します。
- 「第6章 VIおよびサブVIをセットアップする」では、**VI設定**と**サブVIノード設定**のダイアログボックスを使用してVIの動作をカスタマイズする方法について説明します。
- 「第7章 環境をカスタマイズする」では、印刷、表示、**取り消し**などの機能に関するエディタの環境設定、および**制御器**パレットや**関数**パレットの内容の変更によって環境をカスタマイズする方法について説明します。

第II部 フロントパネルオブジェクト

第II部では、フロントパネルのオブジェクト、制御器および表示器、およびActiveXの制御器について説明します。

「第II部 フロントパネルオブジェクト」は、下記の章で構成されています。

「第8章 フロントパネルオブジェクトの概要」では、フロントパネルと、その2つのコンポーネントである制御器と表示器について説明します。また、他のプログラムからグラフィックを取り込んで制御器で使用する方法についても説明します。

「第9章 数値制御器と数値表示器」では、さまざまなスタイルの数値制御器や数値表示器の作成、操作、および構成方法について説明します。

「第10章 ブール制御器とブール表示器」では、ブール制御器やブール表示器の作成、操作、および構成方法について説明します。

「第11章 文字列制御器と文字列表示器」では、文字列制御器や文字列表示器、およびテーブルの使用方法について説明します。これらのオブジェクトには、**制御器→文字列と表**パレットを使用してアクセスできます。

「第12章 パスとRefnumの制御器および表示器」では、**制御器→パスとRefnum**パレットにより呼び出されるファイルパスの制御器およびrefnumsの使用方法について説明します。

「第13章 リストとリングの制御器および表示器」では、**制御器→リストとリング**パレットにより呼び出されるリストボックスとリングの制御器および表示器について説明します。

「第14章 配列とクラスタの制御器 および表示器」では、配列とクラスタの使用方法について説明します。配列やクラスタには、**制御器→配列とクラスタ**パレットによりアクセスします。

「第15章 グラフとチャートの制御器および表示器」では、**制御器→グラフ**パレットでのグラフやチャートの表示器の作成および使用方法について説明します。

「第16章 ActiveX制御器」では、Gベースのソフトウェアと他のアプリケーションとの対話能力を高めるActiveXコンテナの機能について説明します。

第III部 ブロックダイアグラムのプログラミング

第III部では、Gでのブロックダイアグラムの作成および操作に必要なコンポーネントについて説明します。

「第III部 ブロックダイアグラムのプログラミング」の各章の内容は下記の通りです。

- 「第17章 ブロックダイアグラムの概要」では、ブロックダイアグラムを作成する際に使用する3つのコンポーネントのうち、端子とノードについて説明します。もう1つのコンポーネントである配線については、「第18章 ブロックダイアグラムを配線する」で説明します。
- 「第18章 ブロックダイアグラムを配線する」では、ブロックダイアグラムの端子をワイヤを利用して接続する方法について説明します。
- 「第19章 ストラクチャ」では、Forループ、Whileループ、Caseストラクチャ、およびシーケンスストラクチャの使用方法について説明します。これらのストラクチャは、関数→ストラクチャパレットに入っています。
- 「第20章 フォーミュラノード」では、ブロックダイアグラムで数学式を実行するためのフォーミュラノードの使用方法について説明します。フォーミュラノードは、**関数→ストラクチャ**パレットから呼び出すことができます。
- 「第21章 VIサーバ」では、VIやアプリケーションのプログラム制御メカニズムについて説明します。また、VIやアプリケーションの遠隔制御の方法についても説明します。

- 「第22章 属性ノード」では、フロントパネル制御器の属性をプログラムを利用して設定したり、読み取ったりするための属性ノードの使用方法について説明します。便利な属性としては、表示色、制御器の表示、リング制御器のメニュー文字列、グラフやチャートのプロットの色、グラフカーソルなどがあります。
- 「第23章 グローバル変数とローカル変数」では、グローバル変数とローカル変数の定義方法および使用方法について説明します。グローバル変数を利用すると、複数のVIから特定の値のグループを簡単に呼び出すことができます。ローカル変数は、同様の操作を単一のVIから行う場合に使用します。

第IV部 上級トピック

第IV部は、仮装計測器の作成に使用されるGの上級機能およびテクニックに関する内容から構成されています。

「第IV部 上級トピック」の各章の内容は下記の通りです。

- 「第24章 カスタム制御器とType Def」では、カスタム制御器とType Defについて説明します。
- 「第25章 他の言語で書かれたコードを呼び出す」では、他の言語で書かれたコードのさまざまな呼び出し方法について説明します。
- 「第26章 Gの実行システムについて」では、VIのマルチタスク処理と実行について説明します。
- 「第27章 アプリケーションを管理する」では、Gアプリケーションのファイルの管理方法について説明します。
- 「第28章 パフォーマンスについて」は、3つのセクションに分かれています。第1のセクションでは、VIの実行時間に関するデータを表示し、シングルスレッド、マルチスレッド、およびマルチプロセッサのアプリケーションをモニタするパフォーマンスプロファイラについて説明します。第2のセクションでは、実行時の速度に影響を及ぼす要素について説明します。第3のセクションでは、メモリの使用に影響を及ぼす要素について説明します。
- 「第29章 移植性およびローカル化について」では、プラットフォーム間でのVIの移植およびVIのローカル化に関する問題について説明します。

付録、用語集、索引

- 「付録 A データ記憶形式」では、データの記憶形式について説明します。
- 「付録 B Gに関する一般的な質問」では、Gのユーザからよく寄せられるいくつかの質問に対する答を示します。
- 「付録 C カスタマーコミュニケーション」には、お客様からの技術的問題点を当社が解決するための情報をお客様が収集するのに役立つ用紙、および製品の資料に対するお客様のご意見を送っていただくための用紙が用意されています。
- 「用語集」は、本書で使用されている用語を数字、アルファベット、五十音順に並べ、それぞれの用語を解説を示します。略語、頭字語、メートル法の接頭辞、ニーモニック、記号も含まれています。
- 「索引」は、本書に含まれている重要な用語や内容を数字、アルファベットおよび五十音順に並べ、各項目の掲載ページとともに示します。

本書で使用する表記規則

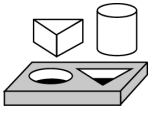
本書では、以下の表記規則を使用します。

<>　キーボード上のキーの名前は<Shift>のように山括弧で囲みます。DBIO<3..0>のように数字の間に省略部分があり全体が山括弧で囲まれている場合は、1つのビットまたは信号名に対応する値の範囲を示します。

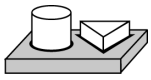
-　<Control-Alt-Delete>のように複数のキーの名前をハイフンでつなぎ、全体を山括弧で囲んだものは、示された名前のキーを同時に押す必要があることを示します。

→　→の記号は、ネストされたメニュー項目やダイアログボックスをたどって最終的な操作に至ることを示します。**ファイル→ページ設定→オプション**→代替フォントという順番で示されている場合は、まず**ファイル**メニューをプルダウンし、次に**ページ設定**、**オプション**という項目をそれぞれ選択し、最後にダイアログボックスから**代替フォント**というオプションを選択します。

『』および「」　参照すべきマニュアルもしくはマニュアルの箇所を表します。



太字のテキストの左にこのアイコンがある場合は、ここから LabVIEW についてさらに詳しく学習するため、示される指示の順番に従って作業を開始することを示します。



太字のテキストの左にこのアイコンがある場合は、示される指示の順番に従って LabVIEW についてさらに詳しく学習するための作業がここで終了することを示します。

☞	**太字の**テキストの左にこのアイコンがある場合は、重要な情報ですので注意してください。
⚠	太字のテキストの左にこのアイコンがある場合は、人体への損傷、データの損失、システムのクラッシュなどを防止するために必要な注意事項を示します。
monospace	この書体は、ディスクドライブ、パス、ディレクトリ、プログラム、サブプログラム、サブルーチン、デバイス、関数、操作、変数などに対応する名前、ファイル名や拡張子、およびプログラムから抽出されたステートメントや注釈などに使用されます。
[monospace]	[ ] 内のテキストは、説明の通りにコードの一部、プログラム例、シンタックス例など、キーボードから入力する必要があるテキストや文字を示します。
[**monospace**][**太字**]	[ ] の太字のテキストは、画面に自動的に表示されるメッセージや応答を示します。
ゴシック体	このフォントで示すテキストは、メニュー、メニュー項目、パラメータ、ダイアログボックス、ダイアログボックスのボタンやオプション、アイコン、ウィンドウ、Windows 95 のタブ、LED などの名前を示します。また、このフォントのテキストの左側余白にアイコンが表記されている場合には、アイコンに対応してそれぞれ作業の目的、備考、注意、警告を示します。複数のプラットフォームに関する情報が記載されている箇所では、プラットフォーム名がこの書体で示されます。
斜体	この書体は、変数、および Windows 3.*x* の場合のようにユーザがこの中から適切な語や値を入力することを示します。
パス	本書のパスの表示では、ドライブ名、ディレクトリ、フォルダ、ファイルなどの区切りに円コード（¥）を使用します。
太字明朝体	このフォントで示すテキストは、強調部分や重要な概念を示します。

関連資料

『G プログラミングリファレンスマニュアル』

『LabVIEW データ集録ベーシックマニュアル』

『LabVIEW Function and VI Reference Manual』

『LabVIEW クイックスタートガイド』

『LabVIEW オンラインリファレンス』**ヘルプ→オンラインリファレンス**を選択します。

『LabVIEW オンラインチュートリアル』（**Windows** のみ）LabVIEW ダイアログボックスから起動します。

『G プログラミングクイックリファレンスカード』

『LabVIEW 始めにお読みください (カード)』

『LabVIEW リリースノート』

『LabVIEW アップグレードノート』

カスタマーコミュニケーション

ナショナルインスツルメンツでは、当社の製品やマニュアルに関するお客様からのご意見をお待ちしております。当社の製品を使用して開発されたアプリケーションについての情報もお寄せください。お客様のアプリケーションに問題がある場合は、弊社がお手伝いさせていただきます。

お客様からのお問い合わせを円滑にするため、本書の「付録 C カスタマーコミュニケーション」には、お客様のご意見やシステム構成を記入していただくための用紙が用意されています。

第I部

Gの基本概念

第I部では、Gプログラミングに関する基本的な情報、および仮想計測器（VI）の作成や編集、カスタマイズ方法について説明します。

「第I部 Gの基本概念」の各章の内容は以下の通りです。

- 「第1章 Gプログラミングの概要」では、Gのユニークなプログラミング方法について説明します。また、Gを使用してプログラム開発を始める方法についても説明します。
- 「第2章 VIを編集する」では、VIの作成や使用に関する基本的な機能、およびパレットやメニューについて説明します。また、オブジェクトの作成、ツールの交換、VIの呼び出し、実行、保存といった基本的な操作の方法についても説明します。
- 「第3章 サブVIを使用する」では、Gアプリケーションの階層設計の概念について説明するとともに、サブVIの２通りの作成方法を示します。また、2つのユーティリティ、すなわちVIの階層を表示する階層ウィンドウと、サブVIやその他のオブジェクトあるいはテキストの文字列を検索する検索ユーティリティについても説明します。
- 「第4章 VIとサブVIの実行およびデバッグ」では、VIの操作とデバッグ、および特殊な実行モードで使用するVIやサブVIのセットアップについて説明します。
- 「第5章 VIの印刷および文書作成」では、VIの印刷および文書作成に関する様々なことがらについて説明します。
- 「第6章 VIおよびサブVIをセットアップする」では、**VI設定**と**サブVIノード設定**のダイアログボックスを使用してVIの動作をカスタマイズする方法について説明します。
- 「第7章 環境をカスタマイズする」では、印刷、表示、**取り消し**などの機能に関するエディタの環境設定、および**制御器**パレットや**関数**パレットの内容の変更によって環境をカスタマイズする方法について説明します。

1

Gプログラミングの概要

この章では、Gのユニークなプログラミング方法について説明します。また、Gを使用してプログラム開発を始める方法についても説明します。

Gとは？

Gは、LabVIEWの中核をなすプログラミング言語です。また、ナショナルインスツルメンツのアプリケーション開発環境であるBridgeVIEWにとっても不可欠な要素となっています。

CやBASICと同様、Gもプログラミング作業に必要な広範な関数ライブラリを備えた汎用プログラミング言語です。Gには、データ集録、GPIBおよびシリアル計測器制御、データ解析、データ表示、データ保存のためのライブラリが用意されています。また、従来のプログラムデバッグツールも備えており、ブレークポイントを設定したり、実行を動画化してプログラム内のデータの流れを確認したり、プログラムをシングルステップで実行することが可能なため、デバッグやプログラム開発を容易に行うことができます。

しかしGは､他のプログラミング言語とは著しく異なる点が一つあります。それは、他のプログラミング言語が**テキストベース**であるのに対し、Gは**グラフィカル**である点です。

VIのコンポーネント

Gは汎用プログラミングシステムですが、データ集録や計測器制御用に特別に設計された関数ライブラリや開発ツールも備えています。Gで作成したプログラムは外観や操作が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（VI：仮想計測器）**と呼ばれます。しかし、VIは従来のプログラミング言語の関数に類似しています。

VIは、対話式ユーザインタフェース、ソースコードとして機能するデータフローダイアグラム、およびVIを上位のVIから呼び出せるようにセットアップするアイコン接続で構成されます。より厳密には、VIの構造は次のようになっています。

- VI の対話式ユーザインタフェースは実際の計測器のパネルとよく似ているため**フロントパネル**と呼ばれます。フロントパネルは、ノブ、押しボタン、グラフ、およびその他の制御器や表示器で構成されます。マウスやキーボードを利用してデータを入力すると、結果がコンピュータの画面に表示されます。
- VI はG で作成した**ブロックダイアグラム**から命令を受け取ります。ブロックダイアグラムはプログラミングの問題に対して絵を使用してソリューションを求めます。ブロックダイアグラムは、VI のソースコードでもあります。
- VI は構造が階層的でモジュール形式になっています。VI は最上位のプログラムとして使用することも、他のプログラムやサブプログラムのサブプログラムとして使用することもできます。他の VI のなかで使用する VI は、**サブ VI** と呼ばれます。VI の**アイコン**と**コネクタ**は、他の VI がサブ VI にデータを渡すためのグラフィカルパラメータリストとしての役割を果たします。

このような特徴を備えたG には、**モジュール式プログラミング**の理念が最大限に反映されています。アプリケーションを一連のタスクに分割し、それらのタスクをさらに細かく分割していくと、複雑なアプリケーションでも一連の単純なサブタスクに分解することができます。個々のサブタスクを実行する VI を作成し、さらにこれらの VI を別のブロックダイアグラム上で組み合わせることで、より大きなタスクの実行が可能になります。最上位の VI は、アプリケーションの関数を表す一連のサブ VI の場合の集合で構成されます。

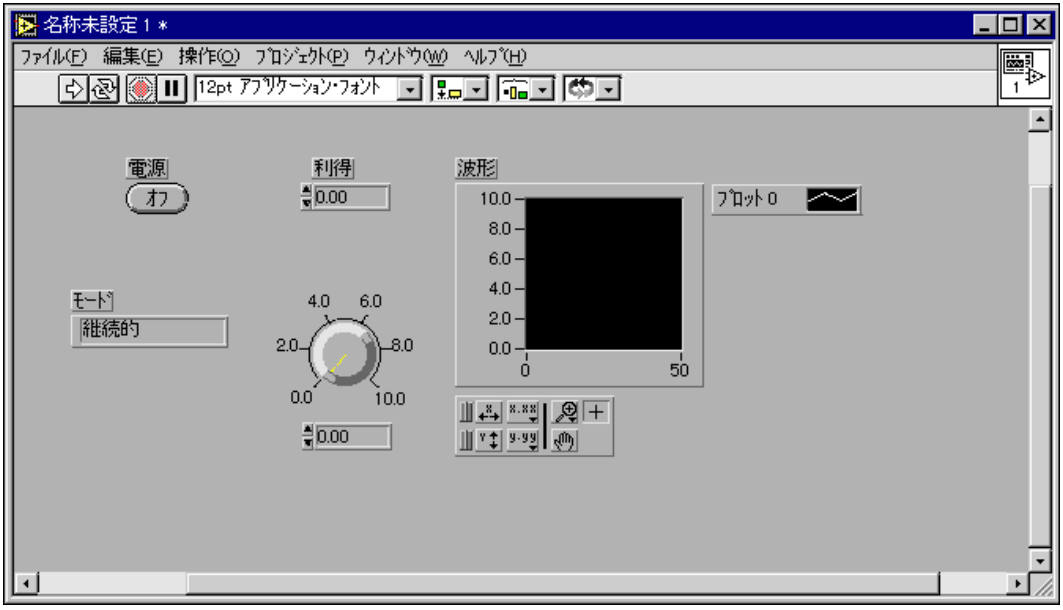
個々のサブ VI は、アプリケーションの他の部分と切り離して単独で実行できるため、デバッグはきわめて容易に行うことができます。さらに、下位のサブ VI の多くは複数のアプリケーションに共通するタスクを実行するケースが多いため、専用のサブ VI のセットをあらかじめ開発しておき、今後作成するアプリケーションに応用することもできます。

以下の項では、フロントパネル、ブロックダイアグラム、アイコン、コネクタ、およびその他の関連機能についてさらに詳しく説明します。

☞ **注**　LabVIEW あるいは BridgeVIEW のアプリケーションで G が使用するさまざまな機能のサンプルを参照するのには `examples` ディレクトリをご覧ください。

フロントパネル

VIのユーザインタフェースは、実際の計測器のユーザインタフェースすなわちフロントパネルによく似ています。VIのフロントパネルの外観は、次の図のような外観になります。



VIのフロントパネルは、主に**制御器**と**表示器**の入力装置をモデルにしたもので、VIのブロックダイアグラムにデータを提供します。表示器は計測器の出力装置をモデルにしたもので、VIのブロックダイアグラムで集録あるいは生成されたデータを表示します。

フロントパネルに追加したい制御器や表示器は、次の図で示した**制御器**パレットから選択します。

制御器や表示器は、サイズ、形、および位置を変更することができます。また、それぞれの制御器や表示器には、さまざまに属性を変更したり、別のメニューアイテムを選択するためのポップアップメニューが用意されています。このポップアップメニューは、下記の方法で呼び出すことができます。

- **（WindowsおよびUNIXの場合）**マウスの右ボタンでオブジェクトをクリックする。
- **（Macintoshの場合）**<command>キーを押しながらクリックする。

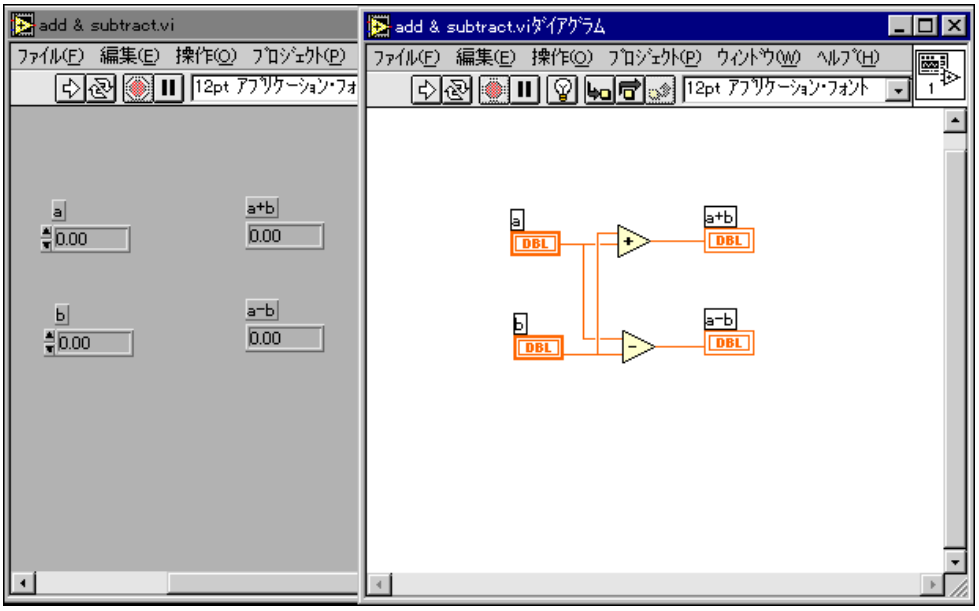
フロントパネルの作成方法については、本書の「第2章　VIを編集する」、および「第II部　フロントパネルオブジェクト」を参照してください。

ブロックダイアグラム

フロントパネルからブロックダイアグラムに切り替える場合は、メニューで**ウィンドウ→ダイアグラムを表示**を選択します。

ダイアグラムウィンドウには、VIのソースコードをグラフィックスで表したブロックダイアグラムが表示されます。ブロックダイアグラムは、データを送受信するオブジェクトや、特定の関数を実行するオブジェクト、実行の流れを制御するオブジェクトを相互に**ワイヤで接続する**ことによって作成します。

次に示す簡単なVIは、2つの数字の和および差を計算します。ダイアグラムには、ブロックダイアグラムの主要なプログラムオブジェクトである**ノード**、**端子**、および**ワイヤ**が表示されています。



制御器や表示器をフロントパネルに配置すると、ブロックダイアグラムにはそれに該当する端子が表示されます。端子は制御器や表示器に付属しており、削除することはできません。フロントパネル上で制御器や表示器を削除した場合にのみ削除されます。

Add および Subtract の関数のアイコンも端子を持っています。これらの端子は、ポートの出入口とみなすことができます。制御器に入力されるデータ（a と b）は、ブロックダイアグラム上の制御器の端子を経由してフロントパネルから**出力**され、Add 関数と Subtract 関数に渡されます。Add 関数と Subtract 関数は、内部の計算を終了すると、出力の端子に新たなデータ値を生成します。これらのデータは表示器の端子を経由して再度フロントパネルに**入力**され、表示されます。**データを生成する端子**はデータソース端子と呼ばれ、データを受け取る端子は**データシンク端子**と呼ばれます。

ノードは、プログラムの実行要素で、従来のプログラミング言語の文、演算子、関数、およびサブルーチンに相当します。Add 関数と Subtract 関数はともにノードの一種です。G には、算術、比較、変換、I/O その他の関数の広範なライブラリが用意されています。

ノードのもう一つのタイプとして、**ストラクチャ**があります。ストラクチャは、従来のプログラミング言語のループやケース文をグラフィックで表したものです。G プログラミング言語には、テキストベースの外部コードにリンクしたり、テキストベースの公式を評価するための特殊なノードもあります。

ワイヤは、ソース端子とシンク端子間のデータの通り道です。ソース端子を別のソースに接続したり、シンク端子を別のシンクに接続することはできません。ただし、1つのソース端子から複数のシンクに接続することは可能です。個々のワイヤは、ワイヤを通過する値のタイプによって形状や色が異なります。前の図では、ワイヤは細い実線で描かれ、値が数値スカラ値であることを示しています。

Gプログラムの実行を決定する原理は**データフロー**です。簡単に言うと、ノードはすべてのデータ入力を受け取った場合にのみ実行し、実行が終了した時点でそのすべての出力端子にデータを送ります。送られたデータは直ちにソースからシンク端子に渡されます。データフロー方式のプログラミングは、従来のプログラムの実行方式、すなわち命令を読み込んだ順に実行するコントロールフロー方式とは対照的です。コントロールフロー方式の実行は命令によって駆動されるのに対し、データフロー方式の実行は**データ駆動**、つまり**データ依存**により駆動されます。

ブロックダイアグラムを利用したプログラムの作成方法についての詳細は本書の「第III部　ブロックダイアグラムのプログラミング」を参照してください。

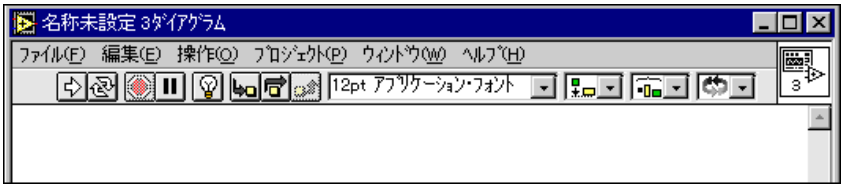
アイコンおよびコネクタ

VIのアイコンを別のVIのダイアグラムに配置すると、そのVIは**サブVI**になります。Gでは、このサブVIがいわゆるサブルーチンに相当します。サブVIの制御器や表示器は、呼び出し側のVIのダイアグラムとの間でデータの受け渡しを行います。

コネクタは、サブVIの制御器や表示器に対応する一組の端子で、**アイコン**は、VIの目的を絵で表したもの、またはVIやその端子を文字で説明したものです。

コネクタは、関数呼び出しのパラメータリストとよく似ており、コネクタの端子はパラメータと同じように動作します。個々の端子は、フロントパネル上の特定の制御器または表示器に対応しています。コネクタはその入力端子を介してデータを受け取り、サブVIの制御器を介してサブVIのコードにデータを渡したり、出力端子を介してサブVIの表示器から結果を受け取ります。

各VIにはデフォルトのアイコンがあります。これらのアイコンは、フロントパネルウィンドウやブロックダイアグラムウィンドウの右上隅のアイコンペーンに表示されます。次の図は、このVIアイコンの表示位置を示したものです。

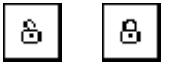


また、各VIにはコネクタがあります。このコネクタにアクセスするには、フロントパネルのアイコンペーンのポップアップメニューで**コネクタを表示**を選択します。コネクタを初めて画面に表示すると、コネクタのパターンが表示されます。必要に応じて別のパターンを選択することもできます。通常、コネクタには、フロントパネル上の制御器あるいは表示器1個に対して1つの端子が付いています。端子は、最大28個まで割り当てることができますので、将来VIを変更して新しい入力や出力が必要になることが予想される場合には、未接続の予備の端子をいくつか残しておくとよいでしょう。

詳しくは、「第3章 サブVIを使用する」を参照してください。

ヘルプの呼び出し

ヘルプウィンドウには、関数、定数、サブVI、制御器や表示器、およびダイアログボックスのメニュー項目に関するヘルプ情報が用意されています。ウィンドウを表示するためには、**ヘルプ**メニューで**ヘルプを表示**を選択するか、または<Ctrl-h>**(Windows)**、<command-h>**(Macintosh)**、<meta-h>**(Sun)**、<Alt-h>**(HP-UX)**を押します。キーボードに<Help>キーがある場合は、このキーを使用してもかまいません。カーソルを関数のアイコン、サブVIのノード、またはVIのアイコン（VIウィンドウの右上に表示されている現在開いているVIのアイコンも含む）上に移動すると、ヘルプ情報が表示されます。



ヘルプ→ヘルプをロックを選択するか、またはウィンドウの下のロックアイコンをクリックすると、現在のヘルプウィンドウの内容をロック（固定）することができます。ヘルプウィンドウの内容がロックされているときは、カーソルを別の関数やアイコンに移動してもヘルプウィンドウの表示内容は更新されません。ロックを解除する場合は、再度**ヘルプをロック**を選択するか、または再度ロックアイコンをクリックします。

フロントパネルのヘルプ

カーソルを制御器または表示器の上に移動すると、ヘルプウィンドウにその制御器または表示器に関する説明が表示されます。VI を作成する際には、各制御器および表示器の説明を入力しておくと便利です。詳しくは、「第2章　VI を編集する」の「オブジェクトに説明を付ける」の項を参照してください。

カーソルをフロントパネルの右上の VI アイコンに合わせたままにすると、ヘルプウィンドウにはその VI のヘルプ情報が表示されます。

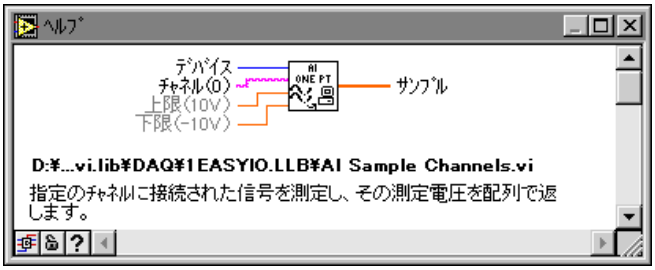
ブロックダイアグラムのヘルプ

関数やサブ VI のノードに関しては、ヘルプウィンドウにはそのノードの機能に関する説明のほかに、アイコン、入力、出力も表示されます。

ヘルプウィンドウでは、ウィンドウの下にある一番左のボタンを押すか、または**ヘルプ→シンプルヘルプ**のメニュー項目を選択するたびに、画面の簡易表示と詳細表示が切り替わります。

シンプル表示の画面では、重要な接続が強調され、重要でない接続は表示されません。この画面では、必要な接続のラベルは太字書体で表示され、推奨される接続のラベルは標準書体で表示されます。オプションの接続については、名前は表示されずに短いワイヤだけが表示され、必要があれば他にも選択肢があることが示されます。次の図は、データ集録 VI の単純表示の画面を示したものです。

詳細表示の画面では、ヘルプウィンドウにはすべての入力と出力が表示されます。入力のワイヤは左向きに、出力のワイヤは右向きに表示されます。また、オプションの接続のラベルはグレーの文字で表示されます。次の図は、上記と同じデータ集録VIの詳細表示の画面を示したものです。



サブVIのディスクの位置は、詳細モードではアイコンの下に表示されます。シンプルモードでは、サブVIの名前だけが表示されます。

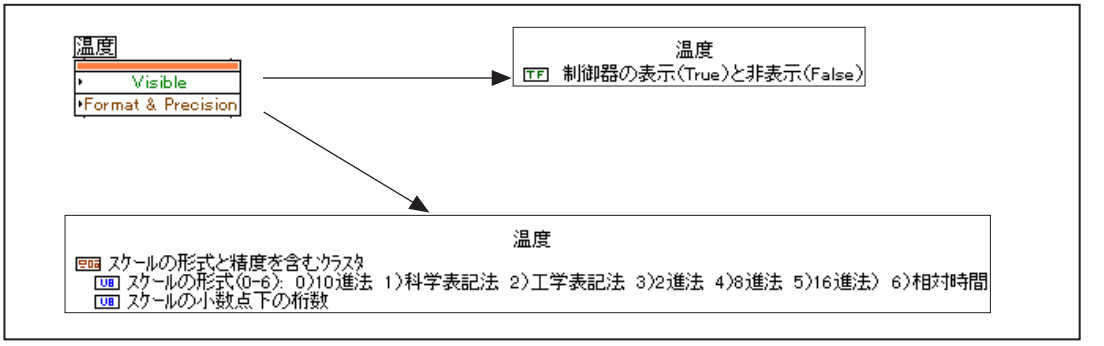
関数の入力を配線する必要のない場合は、ラベルの横にカッコ付きでデフォルトの値が表示されます。関数が複数のデータタイプを処理できる場合は、ヘルプウィンドウには最も一般的なタイプが表示されます。

サブVIノードの端子名は、フロントパネルの対応する制御器および表示器のラベルと一致します。サブVIの入力のデフォルト値は、ヘルプウィンドウでは自動的に表示されません。サブVIを作成する際には、制御器の名前に加えてカッコにデフォルト値も入力しておくと便利です。

配線ツールをワイヤの上に配置すると、ヘルプウィンドウにそのワイヤのデータタイプが表示されます。また、配線ツールをVIアイコンの別の場所に移動すると、ヘルプウィンドウではそれに対応したコネクタの端子がハイライト表示されます。

属性のヘルプ

ある属性の意味がわからないときは、ヘルプウィンドウを使用して属性の意味、そのデータタイプ、および使用可能な値を知ることができます。属性がクラスタのような複雑なタイプの場合は、ヘルプウィンドウにはそのデータストラクチャの階層的な説明が表示されます。次の図は属性ノードを表示したものですが、カーソルを別の属性の名前に合わせるとその属性に関するヘルプ情報が表示されます。



オンラインリファレンス

ヘルプウィンドウには、最も一般的に必要とされるヘルプ情報が要約された形で表示されます。さらに詳しいオンライン情報を見たい場合は、**ヘルプ→オンラインリファレンス**を選択します。ほとんどのブロックダイアグラムオブジェクトについては、そのオブジェクトのポップアップメニューで**オンラインリファレンス**を選択することができます。また、オンラインリファレンスは、ヘルプウィンドウの下に表示されている、左記のようなクエスチョンマークのボタンを押して呼び出すこともできます。

独自のオンラインヘルプファイルの作成については、詳しくは「第5章　VIの印刷および文書作成」を参照してください。

2

VIを編集する

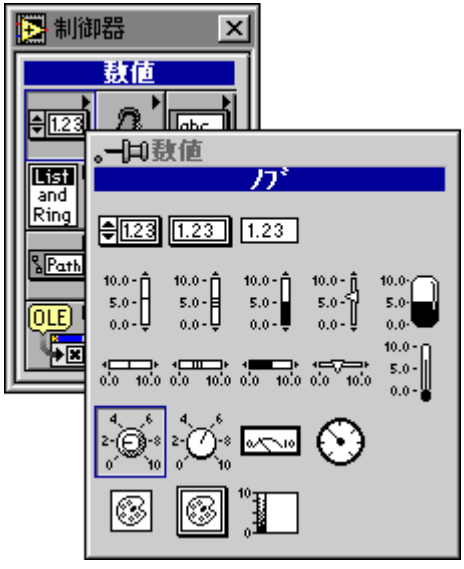
この章では、パレットやメニューをはじめ、VIの作成や使用に関する基本的な機能について説明します。また、オブジェクトの作成、ツールの交換、VIの呼び出し、実行、保存といった基本的な操作の方法についても説明します。

G環境

VIは、パレットやメニューを使用して作成します。フロントパネルやブロックダイアグラムに配置するオブジェクトは、パレットから選択します。これらのオブジェクトは、ツールやメニューを使用して移動したり変更したりできます。

☞ **注** **BridgeVIEWをご使用の場合は、パレットがBasic Gに変更されていることを確認してください。編集→パレットセットの選択を選択した後、パレットセットメニューからBasic Gを選択します。**

フロントパネルやブロックダイアグラム上にオブジェクトを作成するためには、それぞれ**制御器**パレットや**表示器**パレットからオブジェクトを選択します。たとえば、フロントパネル上にノブを作成したい場合は、次の図で示すように**制御器**パレットの**数値**パレットからノブを選択します。



パレットのオブジェクト上にポインタ（矢印）を移動すると、そのオブジェクトの名前がパレットの上部に表示されます。上の図は、ノブが選択された状態を示しています。この状態でマウスボタンをクリックし、フロントパネルまでドラッグした後マウスボタンを放すと、その位置にノブが配置されます。あるいは、パレット上でオブジェクトをクリックした後、ポインタをフロントパネル上に移動し、制御器の位置を示すアウトラインが表示されたら配置したい位置にアウトラインを合わせクリックします。オブジェクトを配置する際にオブジェクトをクリックアンドドラッグすることにより、オブジェクトの作成と同時にオブジェクトのサイズを変更することができます。

☞ **注**　同じパレット内で複数の関数を使用する必要がある場合は、パレットを開いたままにしておくことができます。パレットを開いたままにしておくためには、パレットの左上のピンを選択します。パレットが開いたままになっているときは、タイトルバーをドラッグして簡単に移動することができます。G 開発環境を終了するとその時点のパレット位置が自動的に記憶されるため、次にアプリケーションを起動する際にパレットは同じ位置に表示されます。

フロントパネルウィンドウまたはブロックダイアグラムウィンドウの空白領域を右のマウスボタンでクリックすると、**制御器**パレットと**関数**パレットのコピーが一時的に表示されます。小さいモニタを使用している場合は、**制御器**や**関数**パレットを閉じておき、かわりにこれらのポップアップパレットを利用してオブジェクトを作成することも可能です。

ウィンドウメニューの並べて表示メニュー項目を選択すると、フロントパネルやブロックダイアグラムを横に並べて表示したり、上下に重ねて表示することができます。

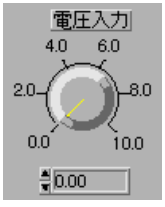
フロントパネルオブジェクトを作成すると、オブジェクトの名前を入力するための空白のラベルが表示されます。オブジェクトに名前を付けたいときは、名前を入力します。名前の入力が終わったら、数値キーパッドの<Enter>キーを押して入力を終了します。キーパッドをお持ちでない場合は、ツールバーの**入力ボタン**をクリックするか、ラベルの外側の任意の場所をクリックしてください。

☞ **注**　制御器を初めて作成する際に、制御器のラベルに名前を入力せずに任意の場所をクリックすると、ラベルが画面から消えます。ラベルを再度画面に表示したい場合は、制御器を右のマウスボタンでクリックし、**表示→ラベル**を選択します。

次の図は、制御器のポップアップメニューで**表示→ラベル**を選択した結果の例を示したものです。

フロントパネル上でオブジェクトを作成すると、それに対応する端子がブロックダイアグラム上に作成されます。制御器からデータを読み取ったり、表示器にデータを送出する場合は、この端子を使用します。

ウィンドウ→ダイアグラムを表示を選択すると、フロントパネルに対応するブロックダイアグラムを表示することができます。ブロックダイアグラムには、フロントパネルのすべての制御器や表示器の端子が含まれています。



フロントパネル

ブロックダイアグラム

ツールパレット

ツールとは、マウスカーソルの特殊な操作モードのことです。パネルやダイアグラム上でマウスをクリックした際、どのような操作が行われるのかは、使用するツールによって決まります。

ツールパレットに含まれるツールの多くは、次の図に示されています。このパレットは任意の場所に移動したり、クローズボックス（パレットの右上の×の付いたボックス）をクリックして一時的に閉じることができます。いったん閉じたパレットを再度画面に呼び出すためには、**ウィンドウ→ツールパレットを表示**を選択します。あるいは、<Ctrl-Shift> **(Windows)**、<command-Shift> **(Macintosh)**、<meta-Shift> **(Sun)**、または <Alt-Shift> **(HP-UX)** を押しながらマウスボタンをクリックすることにより、カーソルの位置に一時的に**ツール**パレットを表示させることもできます。

ツールパレットに表示される個々のツールおよびその機能／説明は下記の通りです。

操作ツール — 制御器の値の変更や制御器のテキストの選択に使用します。

位置決めツール — オブジェクトの位置決め、サイズ変更、選択に使用します。

ラベリングツール — テキストの編集やフリーラベルの作成に使用します。

配線ツール — ブロックダイアグラムのオブジェクト間の接続をします。

オブジェクトポップアップメニューツール — オブジェクトのポップアップメニューを開きます。

スクロールツール — スクロールバーを使用せずにウィンドウをスクロールします。

ブレークポイントツール — VI、関数、ループ、シーケンス、およびケースにブレークポイントを設定します。

プローブツール — ワイヤ上にプローブを作成します。

カラーコピーツール — カラーツールで貼り付けるための色をコピーします。

カラーツール — 前景や背景の色を設定します。

あるツールから別のツールに変更するには、編集モードで下記のいずれかの操作を実行します。

- **ツール**パレットで使用したいツールをクリックします。
- <Tab> キーを使用して最も一般的に使用されるツールを順に移動します。
- フロントパネルがアクティブなときは、スペースバーを押すと操作ツールと位置決めツールが切り替わります。ブロックダイアグラムがアクティブなときは、スペースバーを押すと配線ツールと位置決めツールが切り替わります。

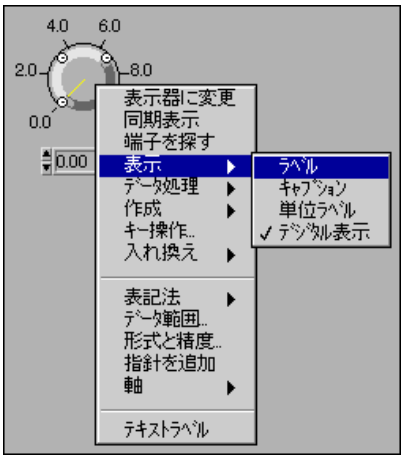
メニューの使用方法

VIウィンドウの最上部（またはMacintoshの画面の最上部）の**メニューバー**には、いくつかのプルダウンメニューが用意されています。**プルダウンメニュー**を開くためには、メニューバーの項目をクリックします。プルダウンメニューは、通常は包括的な性格を持ち、**開く**、**保存**、**コピー**、**貼り付け**といった他のアプリケーションにも共通の項目が含まれているほか、エディタ独自の項目が含まれています。なかには、キーボードからのショートカットを表示するメニューもあります。

最も頻繁に使用されるメニューは、オブジェクトの**ポップアップメニュー**です。フロントパネルやブロックダイアグラムの空白領域も含め、Gのほとんどすべてのオブジェクトにはメニュー項目やコマンドを含むコンテキスト対応のポップアップメニューが用意されています。コマンドやメニュー項目は、できるだけオブジェクトのポップアップメニューから選択するようにしてください。

ポップアップメニュー

先の項で述べたように、Gのオブジェクトすべてに対応するポップアップメニューが用意されています。オブジェクトのポップアップメニューを呼び出すためには、右のマウスボタンでオブジェクトをクリックします（Macintoshの場合は、<command>キーを押しながらオブジェクトをクリックします）。オブジェクトの外観や動作は、ポップアップメニューの項目を選択することによって変更できます。



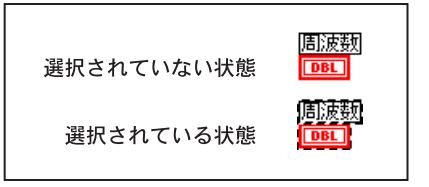
VIを編集する

編集に必要な操作方法がわかっている方は、フロントパネルの作成に関しては「第8章　フロントパネルオブジェクトの概要」を、ブロックダイアグラムの作成に関しては「第17章　ブロックダイアグラムの概要」をお読みください。

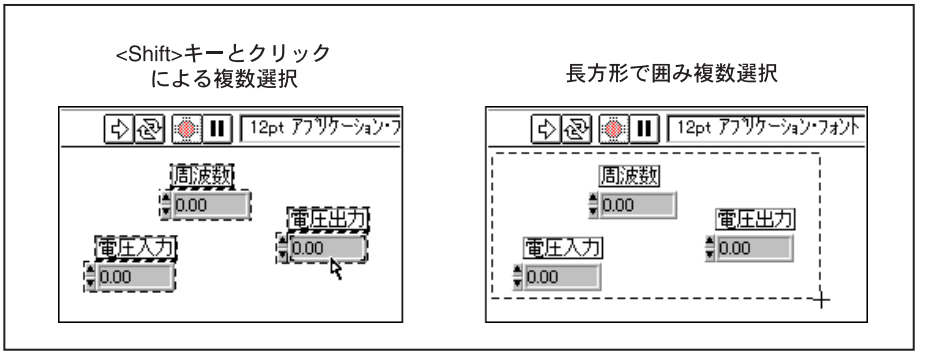
オブジェクトを選択する

位置決めツール

移動、コピー、削除といった編集操作の多くでは、対象となるオブジェクトを**選択する**必要があります。オブジェクトを選択するには、位置決めツールをオブジェクト上に置いた状態でマウスボタンをクリックします。オブジェクトを選択すると、そのオブジェクトは**マーキー**と呼ばれる可動の点線で囲まれます。



複数のオブジェクトを選択するには、<Shift>キーを押しながら追加するオブジェクトをそれぞれクリックします。オブジェクトの選択を解除する際も、<Shift>キーを押しながらクリックします。オブジェクトを選択するもう一つの方法は、長方形をドラッグして選択したいすべてのオブジェクトを囲む方法です。



配列やクラスタなどのオブジェクトを選択するには、これらのオブジェクトが完全に長方形の内側に入っていなければなりません。大部分のオブジェクトは、その一部が長方形内に入っていれば選択されます。新たなオブジェクトを追加選択したり、選択されているオブジェクトの選択を解除する場合は、<Shift>キーを押しながら長方形をドラッグします。

選択されていないオブジェクトをクリックするか、または空白領域をクリックすると、現在選択されているすべてのオブジェクトの選択が解除されます。フロントパネルオブジェクトとブロックダイアグラムオブジェクトを同時に選択することはできませんが、同じフロントパネルまたは同じブロックダイアグラム上で複数のオブジェクトを選択することができます。

VI、画像、およびテキストのドラッグアンドドロップ機能

VIをファイルシステムからブロックダイアグラムにドラッグすると、そのVIに対するサブVI呼び出しを作成することができます。この機能は、特殊制御器、Type Def、およびグローバル変数にも使用できます。また、他のアプリケーションからテキストや画像をドラッグしてフロントパネルやブロックダイアグラムにコピーすることもできます。

Windows 3.1で、サポートされている外部からのドラッグは、ファイルマネージャからのVIおよび制御器のドラッグのみです。Windows 95とWindows NTでは、32ビットOLE（object linking and embedding）をサポートしているため、ファイルマネージャやエクスプローラからだけでなくOLEをサポートしているアプリケーションからもテキストや画像をドラッグすることができます。

Macintoshで外部からドラッグアンドドロップを使用するためには、ドラッグマネージャが必要です。このドラッグマネージャはSystem 7.5以降のシステムに組み込まれています。System 7.0以降System 7.5までのシステムについては、ドラッグマネージャは機能拡張としてAppleより提供されています。

内部的に使用できるドラッグアンドドロップ機能は下記の通りです。

- フロントパネルの制御器および表示器をブロックダイアグラムにドラッグしてブロックダイアグラム定数を作成する、あるいはその逆にブロックダイアグラムの制御器および表示器をフロントパネルにドラッグしてフロントパネル定数を作成する。
- サブVIをパス制御器または定数にドラッグし、制御器または定数内にVIの完全パス名をドロップする。

使用可能な外部からの（他のアプリケーションからの）ドラッグアンドドロップ機能は下記の通りです。

- ファイルをパス制御器または定数にドラッグし、その完全パス名をドロップする。
- データログファイルをデータログファイルRefnumにドラッグし、データログファイル内にデータ構造体を含むクラスタを作成する。
- グラフィックスファイルを画像リング、フロントパネル、またはブロックダイアグラムにドラッグし、ファイルに含まれている画像をドロップする。
- VIファイルをVIのブロックダイアグラムにドラッグし、サブVIとしてドロップする。
- 特殊制御器ファイルをフロントパネルにドラッグし、そのファイルに保存されている特殊制御器をドロップする。
- **（Windows 95/NT）**OLEソースからテキストをドラッグし、そのテキストを文字列の制御器または定数、フロントパネル、ブロックダイアグラム、またはラベルにドロップする。
- **（Windows 95/NT）**OLEソースから画像をドラッグし、その画像を画像リング、フロントパネル、またはブロックダイアグラムにドロップする。
- **（Macintosh）**テキストクリッピング、または他のアプリケーションから選択されたテキストをドラッグし、そのテキストを文字列の制御器または定数、フロントパネル、ブロックダイアグラム、またはラベルにドロップする。
- **（Macintosh）**画像クリッピング、または他のアプリケーションから選択された画像をドラッグし、その画像を画像リング、フロントパネル、またはブロックダイアグラムにドロップする。

☞ **注**　**オブジェクトが正しくドロップされると、ドロップ先がハイライト表示されます。**

ドラッグアンドドロップ機能の詳細については、WindowsまたはMacintoshのシステムマニュアルを参照してください。

オブジェクトを配置する

位置決めツールでオブジェクトをクリックして配置し、そのままドラッグすることによりそのオブジェクトを任意の位置に移動することができます。

<Shift>キーを押しながらオブジェクトをドラッグすると、オブジェクトの移動方向が水平方向または垂直方向に制限されます。オブジェクトが水平方向に移動するか垂直方向に移動するかは、オブジェクトの最初の移動方向によって決まります。

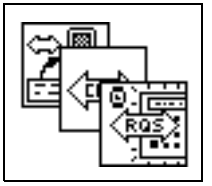
キーボードの矢印キーを使用して、矢印キーを一回押すごとに正確に1ピクセル分ずつ小きざみにオブジェクトを移動することができます。矢印キーを押し続けると、オブジェクトは1ピクセル刻みで連続して移動します。<Shift>キーを押しながら矢印キーを押し続けると、移動速度が速くなります。

ドラッグ中にオブジェクトの移動を中止したいときは、開いているすべてのウィンドウの外側にカーソルをドラッグし、輪郭の点線が消えたらマウスボタンを放します。移動はキャンセルされ、オブジェクトは元の位置に残ります。画面にほとんど空白領域がない場合は、メニューバー上でマウスボタンを放すと便利かつ安全に移動をキャンセルできます。

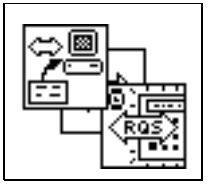
☞ **注**　**位置決めツールには、オブジェクトを拡大させる機能もあります。サイズを変更したくないときは、マウスをクリックする際に十分注意してください。オブジェクトを移動する時は中央を、拡大する時は角をドラッグしてください。**

オブジェクトの前後移動

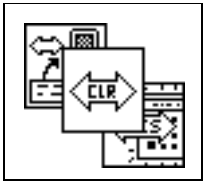
オブジェクトは､他のオブジェクトの上に重ねて配置することもできます。フロントパネルのツールバーの右端にある**再順序**には、重なっているオブジェクトの相対的な位置関係を変更するためのいくつかのコマンドがあります。ここでは、3重に重ねて配置されている3つのオブジェクトを例にとって考えてみましょう。層の一番下にあるのがオブジェクト、一番上にあるのがオブジェクト3とします。



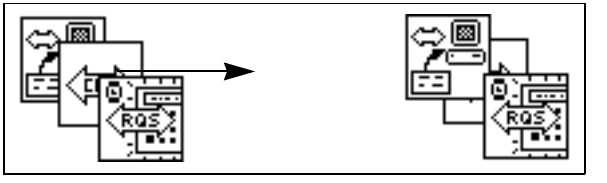
最前面へ移動コマンドは、選択したオブジェクトを一番上に移動します。選択したオブジェクトがオブジェクト1であれば、オブジェクト1が一番上に移動し、オブジェクト3が真ん中、オブジェクト2が一番下になります。



次に、オブジェクト2を選択して**最前面へ移動**を実行すると、オブジェクト2が一番上になり、オブジェクト1が真ん中、一番下がオブジェクト3というように順序が入れ替わります。



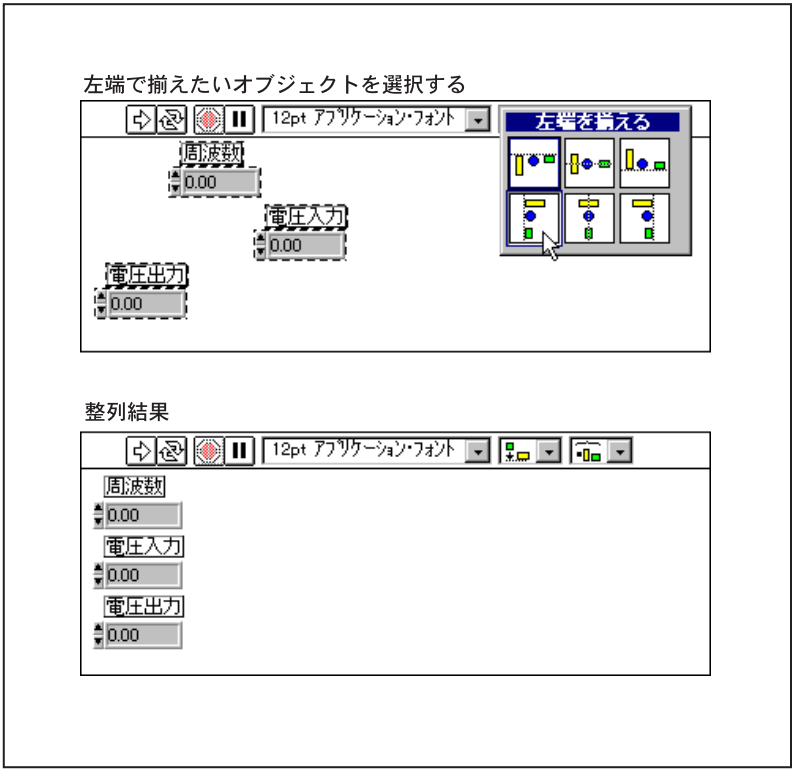
前面へ移動は、選択したオブジェクトを1つ上に移動します。上に示した最初の状態、すなわち下から順にオブジェクト1、オブジェクト2、オブジェクト3という順序で重なっているときに、オブジェクト1を選択してこの項目を選択すると、オブジェクト1が一つ上、すなわち真ん中に移動し、オブジェクト2が一番下に移動します。ただし、オブジェクト3の位置は変わりません。



最背面へ移動と**背面へ移動**は、**最前面へ移動**や**前面へ移動**とは逆に、それぞれ選択したオブジェクトを一番下および1つ下に移動します。

オブジェクトを整列する

整列させたいオブジェクトを選択したのち、それらのオブジェクトをどの軸に沿って整列させるかを左に示した**オブジェクトの整列**リングから選択します。次の図は、整列させたいオブジェクトの選択方法、およびその結果を示したものです。



オブジェクトを垂直軸に沿って整列させる場合は、オブジェクトの左端、中心、または右端のどこを揃えるかを選択します。一方、水平軸に沿って整列させる場合は、オブジェクトの上端、中心、または下端のどこを揃えるかを選択します。現在選択されている整列方法はパレットでは太い枠で囲んで示されます。Macintoshでは、<command-A>を押すとメニューを使用しなくてもこの整列方法が繰り返し選択されます。

オブジェクトを等間隔で配置する

オブジェクトを等間隔で配置する、つまり分散させるためには、分散させたいオブジェクトを選択したあと、**オブジェクトの間隔**リングから分散方法を選択します。選択したオブジェクトの端または中心が等距離になるように分散させることができるほか、リングの右側の4つのメニュー項目を利用して垂直あるいは水平方向のオブジェクト間の間隔を広げたり、狭くしたりすることもできます。Macintoshでは、<command-D>を押すとすべての選択に分散を繰り返し適用することができます。

オブジェクトを複製する

オブジェクトを複製する方法は、コピーと貼り付け、クローン化、ドラッグアンドドロップの3通りの方法があります。

VI内またはVI間でコピーと貼り付けを行うには、位置決めツールでオブジェクトを選択したあと、**編集→切り取り**または**編集→コピー**を選択します。次に、複製を配置したい場所をクリックして**編集→貼り付け**を選択します。オブジェクトを囲む選択マーキーをドラッグして複製すると、複数のオブジェクトを同時に複製することができます。また、他のアプリケーションからテキストや画像をコピーしてフロントパネルやブロックダイアグラムに貼り付けることもできます。

オブジェクトを複製するには、<Ctrl>**（Windows）**、<option>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**キーを押しながら位置決めツールでそのオブジェクトをクリックしたあと、コピーを新しい位置にドラッグします。UNIXでは、修正キーを使用しなくても中央のマウスボタンを利用してクリックアンドドラッグしてオブジェクトを複製することができます。

ラベルの付いているオブジェクトを複製またはコピーすると、そのオブジェクトの名前にcopyという語（コピーのコピーにはcopy 1、copy 2, などの語）を追加した名前がコピーのラベルとして自動的に作成されます。

フロントパネルの制御器をコピーした場合は、ダイアグラム上に新しい端子が作成されます。ただし、オリジナルの制御器のローカル変数や属性ノードはコピーされませんので注意してください。ローカル変数に対して**編集→コピー**または**編集→貼り付け**を使用した場合は、フロントパネルオブジェクトもコピーされます。また、ローカル変数の複製を作成すると、オリジナルの制御器に新しいリファレンスが作成されます。詳しくは、本書の「第22章　属性ノード」および「第23章　グローバル変数とローカル変数」を参照してください。

フロントパネルの制御器は、コピー、複製、ドラッグアンドドロップのどの方法でも、ダイアグラムにコピーし対応する定数を作成することができます。同様に、ユーザ定義定数をブロックダイアグラムからフロントパネルにドラッグして制御器を作成することもできます。

オブジェクトを削除する

オブジェクトを削除するには、そのオブジェクトを選択したあと、**編集→消去**を選択するか、あるいは<Backspace>または<Delete>キーを押します。ブロックダイアグラムの端子は、その端子に対応するフロントパネルの制御器または表示器を削除することによって削除できます。

大部分のオブジェクトは削除できますが、制御器や表示器の一部であるラベルやデジタル表示を作成することはできません。ただし、これらのコンポーネントは、オブジェクトのポップアップメニューで**表示→ラベル**または**表示→デジタルを表示**の選択を解除することによって、画面に表示されないようにすることができます。

Whileループなどのストラクチャを削除すると、その内容も削除されます。ストラクチャだけを削除しその内容を残しておきたい場合は、ストラクチャの端でポップアップメニューを呼び出し、**While ループを削除**（または他のストラクチャの名前）を選択します。この操作を行うと、ストラクチャだけが削除され内容はそのまま残されます。また、ストラクチャの境界をまたいでいるワイヤはすべて自動的に配線し直されます。

オブジェクトにラベルを付ける

ラベルは、フロントパネルやブロックダイアグラムのコンポーネントに注釈を付けるテキストブロックです。ラベルには、**所有ラベル**と**フリーラベル**という2種類のラベルがあります。所有ラベルは特定のオブジェクトに注釈を付けるためのもので、そのオブジェクトに付属し、そのオブジェクトに追従して移動します。これらのラベルは、画面に表示されないようにすることはできますが、ラベルを所有者するオブジェクトと切り離して単独でコピーしたり削除したりすることはできません。フリーラベルはどのオブジェクトにも付属しないラベルで、単独で作成、移動、あるいは廃棄することができます。これらのラベルは、フロントパネルやブロックダイアグラムに注釈を付けるために使用します。フリーラベルを作成したり、ラベルのタイプを編集する場合は、Iビームと長方形のカーソルを持ったラベリングツールを使用します。

フロントパネルのキャプションラベル

フロントパネルのオブジェクトはどれも名前が付いています。この名前を使用すると、オブジェクトを他のオブジェクトと区別することができます。この名前は、オブジェクトの端子、ローカル変数、および属性ノードにも使用されます。

オブジェクトが制御器の場合、ラベルのテキストはブロックダイアグラム上でその制御器に接続されるワイヤの名前を示します。オブジェクトが表示器で、その表示器がコネクタペーンに接続されているときは、ラベルはVIの呼び出し元のブロックダイアグラム上のコネクタペーン端子に接続されたワイヤの名前を示します。このことは、名前ラベルのテキストを変更したときはかならずVIをコンパイルし直す必要があること、およびVIの呼び出し元もコンパイルし直すことが必要になる場合があることを意味します。

フロントパネルオブジェクトには、キャプションラベルを付けることもできます。キャプションは、オブジェクト名に影響を及ぼさないため、オブジェクトを補足説明する名前として使用することができます。キャプションの画面への表示／非表示の切り替えや、キャプションの変更は、属性ノードを使用してプログラム的に行うことができます。

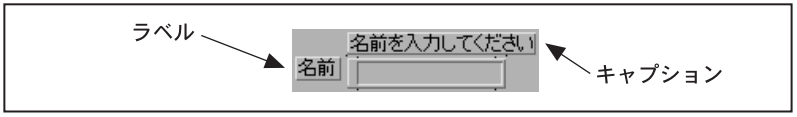


図 2-1 フロントパネルラベルのダイアログボックス

ラベルは、フロントパネルオブジェクトをドロップする際に表示されます。オブジェクトの編集中に画面にキャプションを表示したい場合は、次の図で示すようにオブジェクトのポップアップメニューを呼び出し、**表示→キャプション**を選択します。

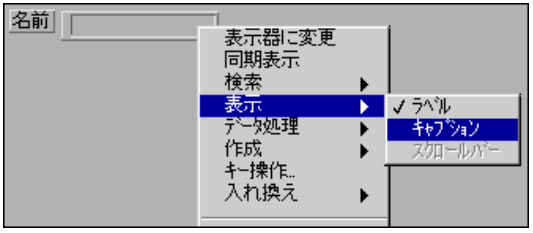


図 2-2 ラベルのポップアップメニュー

キャプションの表示の切り替えやキャプションの変更をプログラム上で行いたい場合は、次の図で示した属性を持つ属性ノードを使用します。

図 2-3 キャプションの属性を表示する属性ノード

オブジェクトのヘルプウィンドウを呼び出す（**ヘルプ→ヘルプを表示する**）と、次の図に示すように、フロントパネルオブジェクトのキャプションはボックスの中に表示され、オブジェクトのラベルはカッコ内に表示されます。

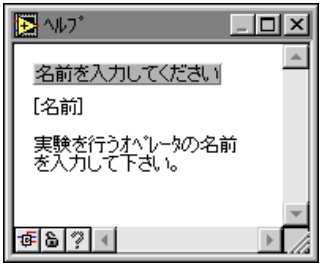
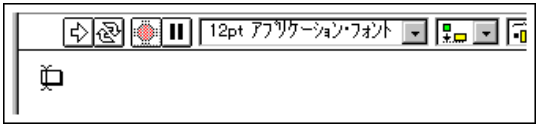


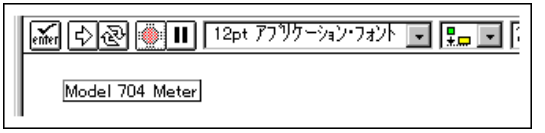
図 2-4 コンテキスト対応のヘルプのダイアログボックス

フリーラベル

フリーラベルを作成するには、**ツール**パレットからラベリングツールを選択して、任意の空白領域をクリックします。



左のマージンの位置にテキストカーソルとともに小さなボックスが表示され、テキストを入力できる状態になります。ラベルに表示したいテキストを入力し、ツールバーの入力ボタンまたはテンキーパッドの **Enter キー**を押して入力を終了します。



テキストの入力は、ラベルの外での任意の場所をクリックして終了させることもできます。ラベルにテキストを入力しなかった場合は、ラベルの外でクリックした時点でラベルは画面から消えます。

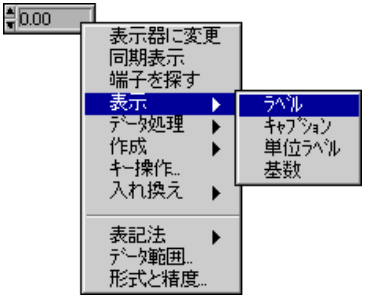
☞ **注**　**制御器または表示器の上にラベルあるいはその他のオブジェクトを重ねて（あるいは部分的に重ねて）配置すると、画面の更新速度が遅くなり、制御器や表示器がちらつくことがあります。このような問題を避けるため、フロントパネルオブジェクトはラベルやその他のオブジェクトと重ねないように注意してください。**

ラベリングツールでコピーしたいラベルのテキストを選択すると、そのテキストをコピーできます。ラベリングツールでテキストをダブルクリックして単語単位で選択します。テキストを3回クリック（トリプルクリック）すると、テキスト全体がハイライト表示に変わります。テキストを選択した状態で**編集→コピー**を選択し、テキストをクリップボードにコピーします。次に、2つ目のラベルのテキストを選択します。さらに**編集→貼り付け**を選択すると、2つ目のラベルのハイライト表示のテキストがクリップボードにコピーしたテキストと差し替えられます。クリップボードのテキストを利用して新しいラベルを作成する場合は、画面上で新しいラベルを配置したい場所をラベリングツールでクリックしたあと、**編集→貼り付け**を選択します。

フロントパネル上に制御器または表示器を作成すると、空白の所有ラベルが一緒に表示され、新しい制御器または表示器の名前を入力することができます。ラベルに何も入力せずにラベルの外側の任意の場所をマウスでクリックすると、ラベルは画面から消えます。再度ラベルを表示したい場合は、制御器または表示器のポップアップメニューで**表示→ラベル**を選択します。

ストラクチャや関数にはデフォルトのラベルが付いていますが、これらのラベルは表示の選択をしない限り画面には表示されません。

隠れている（画面に表示されていない）ラベルを表示させるためには、次の図で示すようにオブジェクトのポップアップメニューで**表示→ラベル**を選択します。



ラベルが表示され、テキストを入力できる状態になります。ラベルに何も入力せずにラベルの外側をクリックすると、ラベルは画面から消えます。

☞ **注**　**ブロックダイアグラム上のサブVIにはユーザが編集できるラベルはありません。サブVIのラベルには、常にそのサブVIの名前が表示されます。これに対して、関数ラベルは編集することができ、ブロックダイアグラムでの関数の使用が反映されます。たとえば、Add関数のラベルを利用すると、どんな値を加えるか、あるいはそれらの値がなぜブロックダイアグラムのそのポイントで加えられるかといったことを示すことができます。関連事項については、この章の「VIに説明を付ける」を参照してください。**

テキストの特性

テキストの属性は、次の図で示すようにツールバーの**フォント**リングのメニューから選択すると変更できます。オブジェクトまたはテキストを選択してからこのリングの項目を選択すると、変更内容は選択したすべてのものに適用されます。オブジェクトやテキストを選択していない場合は、変更はデフォルトのフォントに対してのみ適用されます。つまり、新しく作成するラベルには新規デフォルトが適用されるということになります。デフォルトフォントを変更しても、既存のラベルのフォントが変更されることはなく、新規に作成するラベルのフォントだけが変更されます。

フロントパネルがアクティブな状態でフォントリングから**フォントダイアログ...**を選択すると、次の図のようなダイアログボックスが表示されます。一方、ブロックダイアグラムがアクティブな状態のときには、このダイアログボックスの左下にあるチェックボックスの選択が、**パネルのデフォルト**ではなく**ダイアグラムのデフォルト**が選択された状態になります。

パネルのデフォルトまたは**ダイアグラムのデフォルト**のいずれかのチェックボックスが選択されていると、このダイアログボックスのその他の選択事項は、フロントパネルまたはブロックダイアグラムの新しいラベルに対して適用されます。

上の図では、フォントリングに**アプリケーション**という語が表示されています。このリングには、ほかにも**システムフォント**、**ダイアログフォント**、**現在使用中のフォント**というメニュー項目が用意されています。このリングの最後の項目である**現在使用中のフォント**は、選択された最後のフォントスタイルを指します。

アプリケーション、システム、およびダイアログの各フォントは、インタフェースの特定の箇所に使用されます。これらのフォントは、プラットフォーム間で最適なマッピングが得られるようにあらかじめ定義されています。これらのいずれかのフォントを含むVIを別のプラットフォームに移動すると、これらのフォントは可能な限り密接にマップします。

あらかじめ定義されているフォントは下記の通りです。

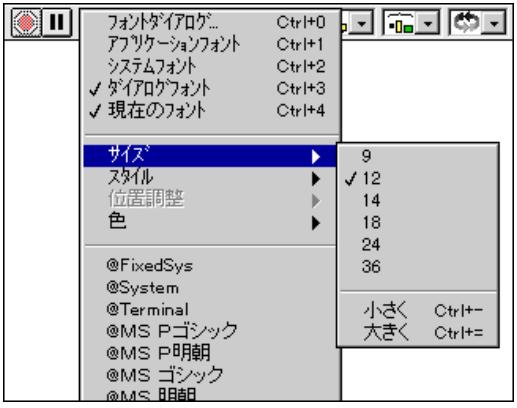
- アプリケーションフォントは、最もよく使用される**デフォルト**フォントです。このフォントは、**制御器**パレット、**関数**パレット、および新しい制御器のテキストに使用されます。
- システムフォントは、メニューに使用されるフォントです。

- ダイアログフォントは、ダイアログボックス内のテキストに使用されるフォントです。

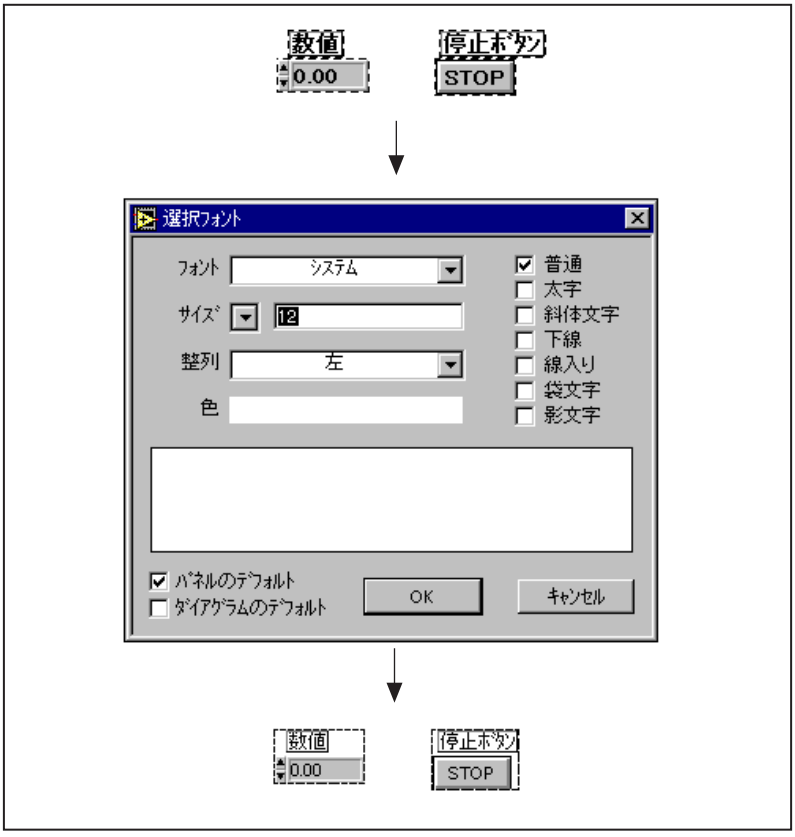
パネルのデフォルトと**ダイアグラムのデフォルト**のチェックボックスをクリックすると、選択されているフォントがフロントパネル、ブロックダイアグラム、またはその両方の現在のフォントになります。新しく作成するラベルには、現在のフォントが使用されます。これらのチェックボックスを使用すると、ブロックダイアグラムでは小さいフォント、フロントパネルに対しては大きいフォントというように、フロントパネルとブロックダイアグラムで異なるフォントを設定することができます。

これら3種類のフォントの移植性に関する問題については、詳しくは「第29章　移植性およびローカル化について」の「解像度とフォント」の項を参照してください。3つのカテゴリーのフォントのデフォルト設定の変更については、詳しくは「第7章 環境をカスタマイズする」の「フォントの環境設定」の項を参照してください。

フォントリングには、次の図で示すように**サイズ**、**スタイル**、**位置調整**、および**色**というメニュー項目もあります。



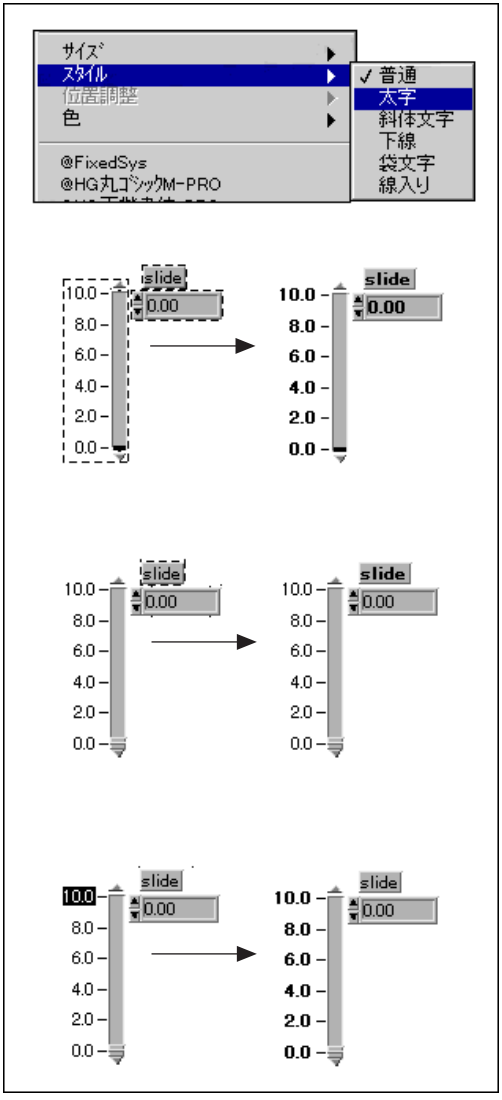
これらのサブメニューで選択したフォントは、ユーザが選択したオブジェクトに適用されます。たとえば、ノブとグラフを選択して新しいフォントを選択すると、ラベル、目盛り（スケール）、およびデジタル表示はすべて新しいフォントに変わります。次の図は、数値およびブール制御器の現在のフォントをシステムフォントに変更する場合の例を示したものです。



G では、フォントを変更してもフォント属性は可能な限り保持されます。たとえば、いくつかのオブジェクトのフォントを Courier フォントに変更した場合、それらのオブジェクトのサイズやスタイルは可能な限りそのまま受け継がれます。同様に、複数のテキストを選択してサイズを変更した場合も、それらのテキストがすべて同じフォントに変更されることはありません。ただし、あらかじめ定義されたフォントや現在のフォントを選択した場合、あるいは選択したオブジェクトをフォントのダイアログボックスを使用して選択したフォントや属性の組み合わせに変更する場合には、これらのルールは適用されません。

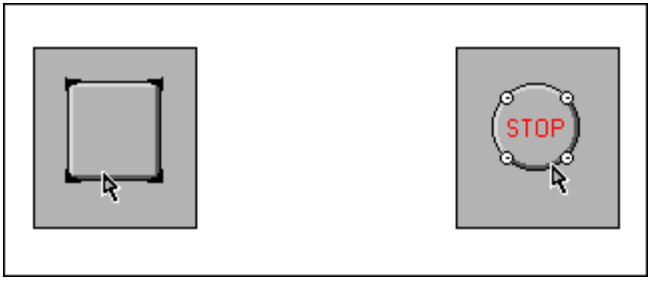
スライドのように複数のテキストを持つオブジェクトを使用する場合は、テキストの選択がオブジェクトや現在選択されているテキストに影響を与えることに注意してください。たとえば、スライド全体を選択して Bold を選択すると、スケールやデジタル表示、ラベルはすべて太字フォントに変わります。ラベルだけを選択して Bold を選択するした場合は、ラベルだけが太字に、また、スケールの目盛りの数字（マーカー）だけを選択して

Boldを選択した場合は、目盛りの数字だけが太字に変わります。次の図は、これら3通りの変更方法を示したものです。



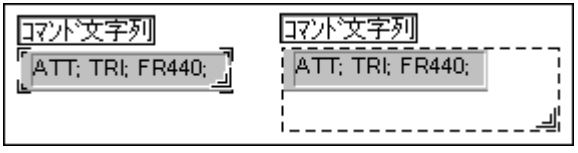
オブジェクトのサイズ変更

ほとんどのオブジェクトは、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が変えられるオブジェクトの上に移動すると、下に示すように四角いオブジェクトではその隅にサイズ変更ハンドルが表示され、丸いオブジェクトではサイズ変更のための円が表示されます。



丸いオブジェクトのサイズは、カーソルで円をドラッグして変更します。同様に、四角いオブジェクトは、隅のハンドルをドラッグしてサイズを変更します。

位置決めツールがサイズ変更ハンドルの上を通過すると、ツールが**サイズ変更カーソル**に変わります。次の図に示すように、このカーソルをクリックし、点線の枠が希望する大きさになるようにドラッグします。



サイズ変更の操作をキャンセルするには、枠の隅をウィンドウの外側までドラッグし、点線の枠が消えたらマウスボタンを放します。オブジェクトは、元のサイズのままになります。

オブジェクトのなかには、縦横いずれか一方のサイズだけしかを変更できないものや、ノブのようにサイズを変更しても縦横の比率は変化しないものもあります。これらのオブジェクトでは、サイズ変更カーソルの形は同じでも、大きさを示す点線の枠は一方向にしかドラッグできません。サイズの変更を縦横いずれかの方向だけに制限したい場合、あるいは現在の縦横の比率を変えたくない場合は、<Shift>キーを押しながらクリックアンドドラッグの操作を行います。

ラベルのサイズ変更

ラベルのサイズも、他のオブジェクトと同様にサイズ変更カーソルを使用して変更できます。ラベルのサイズは通常は**自動サイズ調整**が行われ、入力したテキストがボックス内に収まるように大きさが変更されます。ラベルのテキストは、改行コードを入力したりラベルのボックスのサイズを変更しない限り、1行のままになります。自動サイズ調節機能のオン／オフを切り替えるには、ラベルのポップアップメニューで**テキストにサイズを合わす**を選択します。

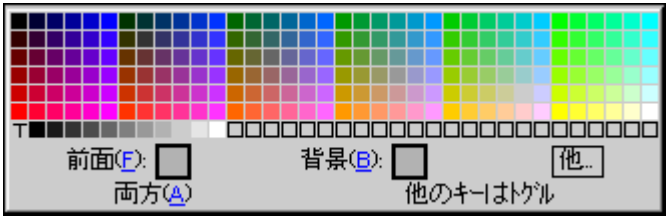
作業スペースを追加する

フロントパネルやブロックダイアグラムに新たな作業スペースを追加するには、<Ctrl-Click>**（Windows）**、<option-click>**（Macintosh）**、<meta-click>**（Sun）**、<Alt-click>**（HP-UX）**したあと、位置決めツールで空いている領域でドラッグします。UNIXのプラットフォームでは、修正キーを使用しなくても、中央のマウスボタンを押しながら空いている領域でドラッグすることも可能です。クリックすると、新たな作業スペースを定義する破線の枠が表示されます。

オブジェクトを色付けする

Gでは、画面はモニタの機能に応じて白黒、グレーの濃淡、またはカラーで表示されます。Gのオブジェクトの多くは、色を変更することができますが、色を変えられないオブジェクトもあります。たとえば、フロントパネルオブジェクトと接続するブロックダイアグラム上の端子やワイヤは、そこを通過するデータの種類や表示方法によって使用する色が決まっているため、変更することはできません。表示モードが白黒の場合も、色を変更することはできません。

オブジェクトの色やウィンドウの背景色を変更するためには、ツールパレットの色付けツール（左の図参照）でオブジェクトまたは背景をポップアップし、次のようなパレットを呼び出します。



マウスボタンを押しながらパレット上を移動すると、オブジェクトの色が現在のカーソル位置の色に変わり、オブジェクトが新しい色でどのように表示されるかを確認することができます。パレット上のある色の位置でマウスボタンを放すとその色が選択され、オブジェクトがその色で表示されます。色の選択の操作をキャンセルするには、カーソルをパレットの外に移動してマウスボタンを放します。

パレットでTと表示されたボックスを選択すると、オブジェクトは透明になります。この機能を使用すると、複数のオブジェクトを重ねて配置することができます。たとえば、表示器の上に透明な制御器を配置したり、標準的な3次元コンテナを使用せずに数値制御器を作成したりすることができます。この機能は単にオブジェクトを透明表示に変えるだけで、マウスやキーボードの操作に対するオブジェクトの反応に影響はありません。

オブジェクトのなかには、背景と前景を個別に色付けできるものもあります。たとえば、ノブでは前景色ダイヤルの部分、背景色がつまみ部分の基本の色などとなります。色の選択ボックスの下にある表示領域には、現在色付けしようとしているのが前景であるか、背景であるか、またはその両方であるかが示されます。カラーインジケータを囲む黒い枠により、前景または背景どちらが選択されているかを表示します。デフォルト設定では、前景と背景の両方が選択された状態になります。

前景と背景の選択は、<f>（**前景**）または<b>（**背景**）のキーを押すことによって切り替えることができます。前景と背景の両方を選択するには、<a>（**両方**）を押します。他のいずれかのキーを押しても、前景と背景の選択を切り替えることができます。

カラーパレットの**他...**を選択すると、色をカスタマイズするためのダイアログボックスが表示されます。次の図は、**色の選択**のダイアログボックスを示したものです。

色の構成3要素である赤、緑、青の値はそれぞれ8ビットで定義され、各成分を合成した色は24ビットで定義されます。したがって、各成分の値の範囲は0から255になります。色の成分の値を変更する場合は、その成分の値の入力フィールドをダブルクリックして新たに値を入力します。基本色の1つを変更するには、色の四角形をクリックして選択項目の1つを選択します。選択した色の各成分の値は、それぞれの値のフィールドに表示されます。

パレットで最後に選択した色が現在の色になります。色付けツールでオブジェクトをクリックすると、そのオブジェクトの色が現在の色に設定されます。

また、**カラー**パレットを使用しなくても、あるオブジェクトの色をコピーし、その色で別のオブジェクトを色付けすることもできます。その場合は、まず最初にコピーしたい色のオブジェクトを色付けツールで<Ctrl-click>**（Windows）**、<command-click>**（Macintosh）**、<meta-click>**（Sun）**、または<Alt-click>**（HP-UX）**します。色付けツールが、選択したオブジェクトの色をしたスポイトの形に変わります。次に、色付けツールで別のオブジェクトをクリックすると、そのオブジェクトが最初に選択したオブジェクトと同じ色で色付けされます。

取り消し

取り消しコマンドの<Ctrl-Z>（Macintoshでは<Cmd-Z>）や**やり直し**コマンドの<Ctrl-Shit-Z>（Macintoshでは<Cmd-Shift-Z>）のほかに、取り消し再実行が可能な動作の回数の設定、さらにVIがインタフェースを変更した場合に自動的にVIの呼び出し元を更新することができます。詳しくは、「第7章　環境をカスタマイズする」の「取り消し」の項を参照してください。

オブジェクトに説明を付ける

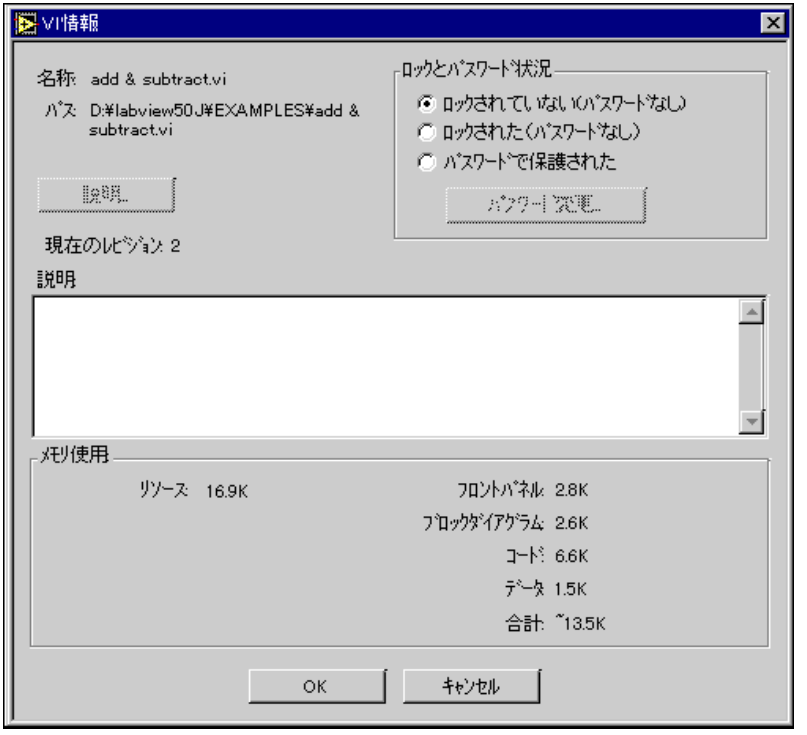
Gの制御器や表示器などのオブジェクトの説明を入力したい場合は、オブジェクトのポップアップメニューから**データ処理→説明 ...** を選択します。説明文を編集する際に、編集モードになっていなければなりません。次の図に示すように、ダイアログボックスに説明を入力し、OKをクリックして説明を保存します。今後そのオブジェクトのポップアップメニューで**説明 ...** を選択すると、この説明が表示されます。

オブジェクトの説明は、ヘルプウィンドウでオブジェクトにカーソルを合わせた場合にも表示されます。作成したVIに対してヘルプ情報を割り当てる最も良い方法は、すべての制御器や表示器に対して説明を入力する方法です。

☞ **注**　サブVIの説明を呼び出し元のVIのダイアグラムから編集することはできません。VIの説明は、そのVIのフロントパネルが開いているときに**ウィンドウ→VI情報を表示**を選択することによって編集できます。

VIに説明を付ける

ウィンドウ→VI情報を表示...を選択すると、次に示すように情報ダイアログボックスに現在のVIの情報が表示されます。このダイアログボックスでは、次のような操作を実行することができます。



- VIの説明の入力。説明ウィンドウには、長い説明文を編集したり表示したりするためのスクロールバーがあります。
- VIのロックおよびロックの解除。ロックされたVIは、実行することはできますが編集することはできません。
- VIのパスワードを設定。パスワードについての詳細は、詳しくは「第27章 アプリケーションを管理する」を参照してください。
- 現在のレビジョン番号の確認。
- VIパスの確認。
- VI　が使用するメモリ量の確認。情報ボックスの**メモリ使用**の欄には、VIが使用しているディスクおよびシステムのメモリ量が表示されます(この数字はVIが使用しているメモリ量を表し、VIのサブVIが使用するメモリ量は含まれません)。

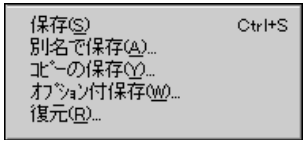
また、メモリの使用量はそれぞれ、フロントパネル、ブロックダイアグラム、VIコード、およびデータスペースが必要とするメモリ量に分けて表示されます。メモリの使用量は、VIを編集したり実行したりする場合にはとくに大きく変化します。通常、最もメモリを多く消費するのはブロックダイアグラムです。ダイアグラムを編集する必要がない場合はVIを保存し、ブロックダイアグラムを閉じておくことでメモリの消費量を減らすことができます。また、サブVIのフロントパネルを保存し、閉じることによってもメモリを節約できます。

VIを保存する

VIは、個別ファイルとして保存したり、複数のVIを1つのグループとしてVIライブラリに保存したりすることができます。

VIを個別ファイルとして保存する

ファイルメニューには、VIを個別ファイルとして保存する際に使用する項目が5つあります。



既存のVIの変更を現在そのVIが保存されているディスク領域に保存する場合は、**保存**を選択します。そのVIが新規のVIである場合は、VIの名前と保存場所を指定するためのファイルダイアログボックスが表示されます。

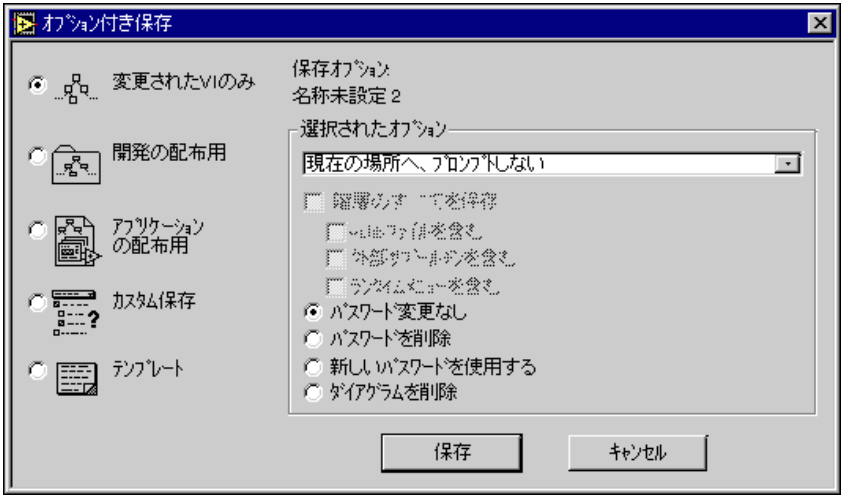
VIを新しい名前で保存したい場合は、**ファイル**メニューから**別名で保存...**、**コピー保存...**、または**オプション付き保存...**を選択します。

別名で保存...を選択すると、メモリ内のVIのコピーは選択した名前を付けてディスクに保存されます。保存が完了すると、メモリ内のVIは新しいバージョンを示します。さらに、元のVIを呼び出していたメモリ内のVIはすべて新しいVIを参照します。そのVIを別の名前で保存すると、ディスク内の元のVIは上書きも削除もされません。

コピー保存...を選択すると、メモリ内のVIのコピーは入力した名前を付けてディスクに保存されます。メモリ内のVIの名前は変更されません。

オプション付き保存...を選択すると、VIの階層全体の保存を選択できる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では、VIにパスワードを付けて保存する方法や、ブロックダイアグラムを付けないで保

存する方法をオプションで選択できます。この方法は、VI を配布したり、バックアップコピーを作成する際に使用すると便利です。



このメニュー項目の使用方法についての詳細は、「第27章　アプリケーションを管理する」を参照してください。

注意　**ブロックダイアグラムを付けずに保存した場合、後でそのVIを編集することはできません。必ず元のVIのコピーを作成してください。**

ファイル→復元 ... オプションを使用すると、現在の VI を最後に保存したときの状態に戻すことができます。その際、これまでVIに対して行った変更を無効にしてもかまわないかどうかを確認するダイアログボックスが表示されます。

前回保存したVIに新たに変更を加えると、その VI のタイトルバーと**ウィンドウ**メニューに表示される開いている VI のリストにアスタリスクが表示されます。このアスタリスクは、VIを保存すると消え、その後新たにVIを変更するまで表示されません。

バックアップコピーの保存については、「第27章　アプリケーションを管理する」を参照してください。

注意　**VIは、`LabVIEW`や`BridgeVIEW`ディレクトリ内の`vi.lib`ディレクトリに保存しないでください。このディレクトリは、LabVIEWやBridgeBIEWの新しいバージョンをリリースする際に、必要に応じて更新されます。そのため、`vi.lib`にVIが格納されていると、将来新たにインストールを行う際に重複が発生する危険性があります。VIを関数パレットに表示したい場合は、詳しくは「第7章　環境をカスタマイズする」の「制御器パレットおよび関数パレットをカスタマイズする」の項を参照してください。**

VIをVIライブラリ（.LLB）に保存する

複数のVIを1つのグループにまとめてVIライブラリに保存することができます。ただし、ライブラリとして保存する理由が特にない場合は、個々のVIを個別ファイルとしてディレクトリに保存するほうが賢明です。下記のことがらを考慮したうえで、VIをライブラリに保存するかどうかを決定してください。

VIをライブラリとして保存する理由

- Microsoft Windows 3.1を使用している場合、あるいはファイルをWindows 3.1に転送する予定がある場合は、VIをライブラリ（`.llb`ファイル）として保存すると最大255文字でファイル名を指定することができます。他のオペレーティングシステムも長いファイル名をサポートしています（Macintoshは31文字、Windows 95、Windows NT、およびUNIXシステムでは255文字）。
- VIを他のプラットフォームに転送する場合は、複数のVIを個別ファイルとして転送するよりもライブラリとして転送するほうが簡単です。
- ディスク容量が重要な問題となる場合は、ファイルをVIライブラリ内に保存することでファイルを圧縮し、ディスクスペースを多少節約することができます。

VIを個別ファイルとして保存する理由

- VIを個別ファイルとして保存すると、ファイルシステムを使用してVIを管理することができます（ソースコードのコピー、移動、名前の変更、バックアップ、管理）。
- VIライブラリ内には階層を作成することはできません（VIライブラリにはサブディレクトリやフォルダを入れることはできません）。
- ファイルのロードや保存は、VIライブラリよりもファイルシステムからの方が短時間で実行できます。また、一時的なファイルはロードや保存を実行するのにさほど大きなディスク容量を必要としません。
- VIや制御器を個別ファイルに保存する方がプロジェクト全体を同じファイルに保存するよりもプログラミングの柔軟性を高めることができます。
- VIライブラリには、Professional Developer Toolkitのソースコード制御器との互換性がありません。

☞ **注**　**G言語ソフトウェア（LabVIEWおよびBridgeVIEW）とともに提供されるVIの多くは、VIライブラリとして保存されています。それにより、すべてのプラットフォームで一貫した場所に格納されるようになっています。**

VIライブラリを作成する

VIライブラリの作成に際しては、下記の手順に従ってください。

1. **ファイル**メニューから**別名で保存...**または**コピー保存...**を選択します。
2. **（Macintosh）編集→環境設定→その他**でネイティブファイルダイアログを使用するように構成する場合は、表示されるダイアログボックスで**LLBを使用**をクリックします。
3. 上記手順または2の結果表示されるダイアログボックスで、**新規 ...**または**新規VIライブラリ**のボタンをクリックします。
4. 表示されるダイアログボックスで新しいライブラリの名前を入力し、**VIライブラリ**ボタンをクリックします。ライブラリ名を入力する際に`.llb`という拡張子を入力しなかった場合は、ライブラリ名に自動的にこの拡張子が追加されます。
5. VI名を指定（希望する場合）して新しいライブラリに保存するためのダイアログボックスが表示されます。

既存のVIライブラリへ保存する

新しいVIは、ディレクトリに保存する場合と同じようにライブラリに保存することができます。**ファイル**メニューでいずれかの保存オプションを選択すると、ファイルダイアログボックスに.llbという拡張子の付いたライブラリファイル名が表示されます（Macintoshでネイティブファイルダイアログを使用している場合は、**LLBを使用**ボタンを押してからVIライブラリを選択する必要があります）。ライブラリファイルの名前をダブルクリックするか、または**開く**を押すと、ダイアログボックスに項目が表示され、ファイルをライブラリに保存できる状態になります。

ライブラリの内容を編集する

ファイル→VIライブラリを編集 ...オプションを使用すると、VIライブラリからファイルを削除することができます。VIライブラリを編集のダイアログボックスが表示されたら、ファイルに**最上位**のマークを付けることができます。このマークには次の2通りの使用方法があります。その一つは、アプリケーション作成ライブラリを使用してアプリケーションを作成する際に特定のVIを**最上位**として設定しておくと、アプリケーションを実行する際にそれらのVIが自動的に開かれるようにする方法です。もう一つは、WindowsまたはMacintoshのファイルシステムで特定のライブラリをダブルクリックしたとき、あるいはWindowsまたはUNIXのコマンドラインでライブラリ名を指定してG開発環境を立ち上げたときに、最上位のすべてのVIが自動的に開かれるようにする方法です。

また、ファイルダイアログボックスに`.llb`ファイルの内容を表示する際には、最上位のVIは区切り線によって他のVIと区別されます。

3

サブVIを使用する

この章では、Gアプリケーションの階層設計の概念について説明するとともに、サブVIの2通りの作成方法を示します。また、2つのユーティリティ、すなわちVIの階層を表示する階層ウィンドウと、サブVIやその他のオブジェクトあるいはテキストの文字列を検索する検索ユーティリティについても説明します。

階層化設計

Gアプリケーションを作成するには、VIの階層的特性を理解し、その特性を利用することが大切です。作成したVIは、それよりも上位のVIのブロックダイアグラムでサブVIとして使用することができます。つまり、サブVIはC言語におけるサブルーチンに相当します。Cプログラムで使用できるサブルーチンの数に制限がないのと同様、Gプログラムで使用できるサブVIの数にも制限はありません。また、サブVIを別のサブVIの中で呼び出すこともできます。

アプリケーションを作成する場合、最上位のVIから始めてそのアプリケーションに対する入力と出力を定義します。次に、ほかのVIをサブVIとして使用して、ブロックダイアグラムを流れるデータに対して必要な操作を実行します。ブロックダイアグラムに多数のアイコンがある場合は、それらを下位のVIとしてグループ化することでブロックダイアグラムの簡略化を図ります。このような手順でモジュール化していくと、明解で、デバッグや保守が容易なアプリケーションを作成することができます。

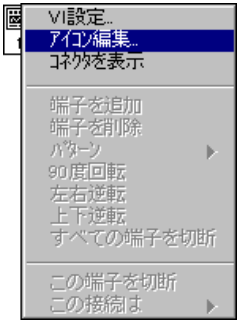
サブVIは、VIから作成することも、VIの選択部分（一部）から作成することもできます。この章では、その両方の方法について説明します。

VIからサブVIを作成する

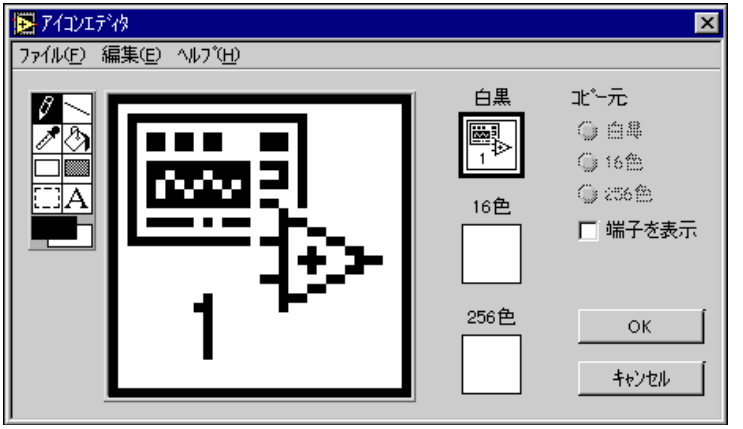
VIの**アイコン**は、VIを図で表した記号です。VIの**コネクタ**は、制御器や表示器を入力端子と出力端子に割り当てます。VIを別のVIのブロックダイアグラムから呼び出したい場合は、まず最初にそのVIのアイコンとコネクタを作成します。この項では、VIのアイコンやコネクタの作成方法および編集方法について説明します。

アイコンを作成する

アイコンを作成するには、フロントパネルの右上隅のアイコンをダブルクリックするか、または次の図で示すようにアイコンのポップアップメニューで**アイコン編集...**を選択します。



アイコン編集...を選択すると、アイコンエディタが表示されます。



ウィンドウの左側にあるツール（上の図参照）を使用して、大きなピクセルの編集領域でアイコンをデザインします。編集領域の右側の該当するボックスに、標準サイズのアイコンが表示されます。

使用するモニタの領域に応じて、白黒、16色、および256色のそれぞれの表示モードのアイコンを個別にデザインすることができます。それぞれのバージョンのアイコンは個別にデザインし、保存します。エディタのデフォルトモードは**白黒**ですが、別のカラーメニュー項目をクリックすることにより、モードを切り替えることができます。アイコンエディタの右側の**コピー元**メニューを使用すると、カラーアイコンを白黒アイコンにコピーしたり、白黒アイコンをカラーアイコンにコピーすることができます。

注　**アイコンを作成する際に、少なくとも白黒アイコンを１つ作成しておくことをおすすめします。カラーアイコンしか作成していない場合、そのアイコンを関数パレットに追加してもカラーアイコンは表示されません。また、カラーアイコンは印刷不可能となり、白黒モニタにもアイコンは表示されません。このような場合には、VIに白黒アイコンがないと、空白のアイコンが使用されるからです。詳しくは、「第７章　環境をカスタマイズする」の「制御器パレットおよび関数パレットをカスタマイズする」の項を参照してください。**

編集領域の左側にあるツールアイコンの機能は下記の通りです。

鉛筆 ― ピクセルを1つずつ描画または消去します。描画を水平線および垂直線に限定するには<Shift>キーを使用します。

線 ― 直線を描画します。描画を水平線、垂直線、および対角線に限定するには<Shift>キーを使用します。

スポイト ― アイコンの構成要素から前景色として使用する色を選択します。スポイトで背景色を選択する場合は、<Shift>キーを使用します。

塗りつぶし ― 輪郭で囲まれた領域を前景色で塗りぶつします。

四角形 ― 長方形の境界を前景色で描きます。アイコンに前景色のフレームを付ける場合は、このツールをダブルクリックします。正方形にしたい場合は<Shift>キーを使用します。

塗りつぶした四角形 ― 境界線の色を前景色で、内部を背景色で塗りつぶした四角形を描画します。アイコンに前景色のフレームを付けて背景色で塗りつぶす場合は、ダブルクリックします。正方形にしたい場合は<Shift>キーを使用します。

選択 ― 移動、コピー、または削除するアイコンの領域を選択します。アイコン全体を選択する場合はダブルクリックします。<Delete>キーを押すと、選択した部分が消去されます。長方形を正方形に限定したい場合は<Shift>キーを使用します。

テキスト ― アイコンにテキストを入力します。別のフォントを選択する場合はこのツールをダブルクリックします。アイコンの近くでテキストを移動するためにはカーソルキーを使用します。

前景色と背景色 ― 前景と背景の現在の色を表示します。新しい色を選択するパレットを表示するためには、前景または背景のいずれか一方をクリックします。

<Ctrl>（**Windows**）、<option>（**Macintosh**）、<meta>（**Sun**）、または<Alt>（**HP-UX**）キーを押し続けると、すべてのツールが一時的にスポイトに変わります。

編集画面右側のボタンの機能は下記の通りです。

OK ― 描画をVIアイコンとして保存し、フロントパネルに戻ります。

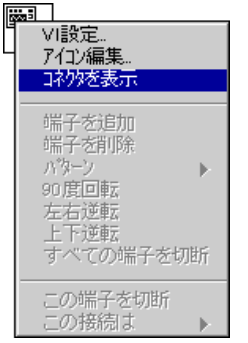
キャンセル ― 変更を保存せずにフロントパネルに戻ります。

☞ **注**　**アイコンの画像の切り取り、コピー、貼り付けは、編集メニューを使用して行います。アイコンの一部を選択した状態で画像を貼り付けると、選択した部分に適合するように画像のサイズが調節されます。**

コネクタ端子パターンを定義する

サブVIとのデータの受け渡しは、サブVIのコネクタペーンの端子を介して行います。接続は、VIに必要な数の端子を選択し、それぞれの端子にフロントパネルの制御器または表示器を割り当てることによって定義します。コネクタペーン上で端子を必要とするのは、サブVIへの配線を通じてプログラムで使用する制御器および表示器だけです。

VIのコネクタがフロントパネルの右上隅に表示されていない場合は、次の図で示すようにアイコンペーンのポップアップメニューから**コネクタを表示**を選択します。ブロックダイアグラムにはコネクタペーンはありません。

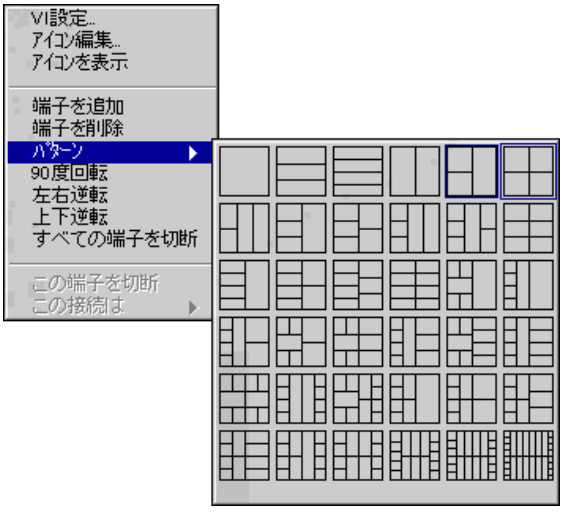


コネクタは、フロントパネルの右上隅にあるアイコンにかわりコネクタが表示されます。最初に選択されるパターンでは、フロントパネル上の制御器と同じ数の端子がコネクタペーンの左側に、フロントパネル上の表示器と同じ数の端子がコネクタペーンの右側に表示されます。それが不可能な場合は、最も近い数の端子が自動的に選択されます。

コネクタ上の四角形はそれぞれ個々の端子領域を表します。これらの端子領域は、VIへの入力端子またはVIからの出力端子として使用できます。異なる端子パターンを使用したい場合は、別のパターンを選択できます。

端子パターンの選択と変更

VIに対して別の端子パターンを選択するには、コネクタのポップアップメニューで**パターン**を選択します。



現在アイコンに割り当てられている端子パターンは、太枠で囲まれています。このパターンを変更するには、別のパターンを選択します。

現在のパターンに端子を1つ追加したい場合は、端子を追加したい場所にカーソルを移動し、ポップアップメニューから**端子を追加**を選択します。

パターンの端子を削除したい場合は、端子のポップアップメニューで**端子を削除**を選択します。

コネクタの端子パターンの配置を変更したい場合は、コネクタペーンのポップアップメニューから**左右逆転**、**上下逆転**、または**90° 回転**選択します。

端子を制御器および表示器へ割り当てる

コネクタの端子パターンが決定したら、下記の手順に従ってそれぞれの端子にフロントパネルの制御器と表示器を割り当てます。

1. コネクタの端子をクリックします。ツールが自動的に配線ツールに変わり、端子の色が黒い表示に変わります。

2. 選択した端子に割り当てるフロントパネルの制御器または表示器をクリックします。選択した制御器の外枠がマーキーで囲まれます。

3. フロントパネルの空白領域にカーソルを移動しクリックします。マーキーが消え、選択した端子が接続先の制御器のデータの色に変わり、その端子が割り当てられたことを示します。

☞ **注**　**コネクタの端子の色が白で表示された場合は、その端子が接続されていないことを表します。その場合は、端子が正しい色で表示されるまでステップ1からステップ3の操作を繰り返してください。**

配線ツールを使用してコネクタの端子をフロントパネルの制御器や表示器に割り当てますが、コネクタと制御器あるいはコネクタと表示器がワイヤで配線されるわけではありません。

4. ステップ1とステップ2の操作を行い、それぞれの制御器および表示器を端子に接続します。

あるいは、制御器または表示器を選択してから端子を選択してもかまいません。また、必要な端子数よりも多く端子のあるパターンを選択することもできます。将来VIを変更して新しい入力や出力が必要になることが予想される場合には、予備の端子を接続しないまま残しておくことができます。使用可能な接続が余分にあるということは、入力や出力を新しく追加しても、すでにこのVIをサブVIとして使用しているほかのVIには影響を与えないことになります。未接続の端子がVIの動作に影響を与えることはありません。また、端子の数よりも多くのフロントパネル制御器を使用することもできます。

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29個以上の制御器または表示器をフロントパネルで使用したい場合は、その一部をクラスタとしてグループ化し、そのクラスタをコネクタに端子を割り当てます。詳しくは、「第14章 配列とクラスタの制御器 および表示器」の「クラスタ」の項を参照してください。

サブVIに必要な接続、推奨される接続、オプションの接続

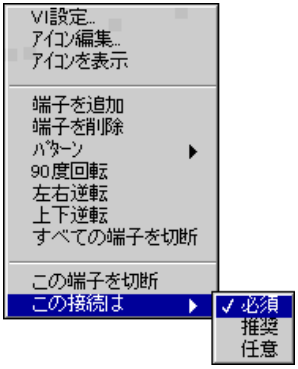
Gには、サブVIの接続の配線漏れを防ぐための機能があります。必須、推奨、およびオプションの接続がコネクタペーンに表示され、また、ヘルプウィンドウにも同じ内容が表示されます。

`vi.lib`に収められているVIの入力および出力は、あらかじめ**必須**、**推奨**、または**オプション**のいずれかに設定されています。Gでは、ユーザが作成したVIの入力と出力はデフォルトで**推奨**に設定されます。

入力が**必須**に設定されているVIは、その入力を正しく配線しない限りサブVIとして実行することはできません。入力または出力が推奨に設定されているVIは、実行することは可能ですが、警告が有効になっている場合はエラーリストウィンドウに警告メッセージが表示されます。

接続が**必須**、**推奨**、**オプション**のどれに設定されているかを知りたい場合、あるいは接続をこれらのいずれかに設定したい場合は、コネクタペーンの端子をクリックし、**この接続は**のサブメニューから該当するオプションを

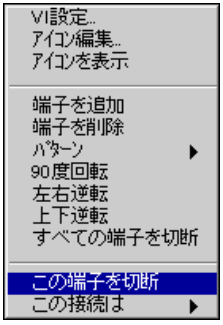
選択します。次の図で示すように、現在選択されているオプションにはチェックマークが表示されます。



ヘルプウィンドウでは、**必須**の接続は太字で、**推奨**の接続は普通の書体で、**オプション**の接続はグレーの文字で表示されます。

端子の接続を削除する

端子と制御器や表示器との間の接続は、個別に削除することも、すべての接続を一括して削除することもできます。特定の接続だけを削除するには、次の図で示すように、コネクタ上の接続を削除したい端子をポップアップし、ポップアップメニューから**この端子を切断**を選択します。



端子の色が白に変わり、選択した接続が存在しなくなったことを示します。コネクタのすべての接続を削除する場合は、コネクタのポップアップメニューから**すべての端子を切断**を選択します。ただし、**端子を削除**とは異なり、**この端子を切断**では端子パターンから端子が削除されるわけではない点に注意してください。

端子の接続を確認する

端子にどの制御器または表示器が割り当てられているかを確認したい場合は、コネクタペーンが表示されている状態で配線ツールで制御器、表示器、または端子をクリックすると、端子に割り当てられているオブジェクトが選択されます。

選択範囲からサブVIを作成する

VIの一部を、別のVIから呼び出すことのできるサブVIに変換することができます。VIの一部を選択したのち、**編集→選択範囲をサブVIに変換**を選択すると、選択した部分がサブVIになります。新しいサブVIの制御器および表示器は自動的に作成され、このサブVIは既存のワイヤに自動的に接続されます。また、元のVIのブロックダイアグラムの選択した部分は、このサブVIのアイコンに差し替えられます。

選択範囲からサブVIを作成することは、下記の点を除いては、選択したオブジェクトを削除してサブVIと差し替えることと同じです。

- 選択範囲に含まれるフロントパネル端子は、呼び出し側のVIから削除されません。これらのフロントパネル端子は呼び出し側のVIに残され、サブVIに配線されます。
- 選択範囲に含まれる属性はすべて、呼び出し側のVIに残され、サブVIに配線されます。選択された属性はサブVIではフロントパネル端子に差し替えられ、属性の値をサブVIに入力あるいはサブVIから出力するためのチャンネルの役割を果たします。
- 選択範囲に含まれるローカル変数は、サブVIではフロントパネル端子に差し替えられます。このローカル変数は、呼び出し側のVIに残され、サブVIに配線されます。選択範囲に同じローカル変数の複数のインスタンスが存在する場合は、最初のインスタンスがフロントパネル端子になり、残りのインスタンスがサブVIのローカル変数として残されます。呼び出し側のVIには1つのインスタンスだけが残されます。それ以外の選択されたインスタンスは呼び出し側のVIから削除され、サブVIに移動されます。
- ローカル変数はフロントパネル端子からの読み取りまたは書き込みが可能であるため、読み取りと書き込みの両方のローカル変数のインスタンスを選択すると、サブVIでは2つのフロントパネルオブジェクトが作成されます。一方はローカル変数の値をサブVIに渡し、もう一方はサブVIからの値を渡します。読み取りのローカル変数を表すために作成されたフロントパネル端子の名前には［読み取り］という接頭辞が付けられ、書き込みのローカル変数を表すために作成されたフロントパネル端子の名前には［書き込み］という接頭辞が付けられます。

ルールおよび推奨事項

選択範囲からサブVIを作成する機能は非常に便利ですが、そのメカニズムは見かけほど単純ではありません。VIの論理階層を作成するためには、やはり綿密なプランニングが必要になります。また、選択範囲に含めるオブジェクトについても慎重に検討しVIの機能に影響がないようにする必要があります。明らかに問題が生じる場合はサブVIは作成されず、操作が実行されなかった原因について説明するダイアログボックスが表示されます。また、問題が生じる可能性がある場合は、その問題に関する説明がダイアログボックスに表示されます。その場合は、作業を続行するか中断するかを選択する必要があります。

☞ **注**　**選択範囲にローカル変数や属性ノードが含まれていない場合は、シーケンスストラクチャを使用してサブVIに転送したいコードをカプセル化することにより、結果を確認することができます。**

この機能を使用する際に知っておくと便利なルールおよび推奨事項を次に示します。

接続の数

作成されるサブVIの入出力数がコネクタペーンの最大接続数である28個を超えないように、選択する範囲の大きさに注意してください。

コネクタペーンでは、選択範囲に含まれる各フロントパネル端子、各属性、および特定のローカル変数のためのスロットがそれぞれ1つずつ必要になります。選択範囲の中にこれらの項目が多すぎると、コネクタペーンのスロットが足りなくなる場合があります。

端子の数が上限を超えないようにするためには、ダイアグラムの選択範囲を小さくするか、またはデータを配列およびクラスタにグループ化したうえでダイアグラムから変換する領域を選択してください。

サイクル

ダイアグラムでサイクルを発生させるおそれのある範囲を選択しないようにしてください。サイクルは、データフローがサブVIの出力を起点とし、サブVIへの入力として終了する場合に発生します。

範囲を選択する際にサイクルを識別することは困難ですが、Gではサイクルは自動的に検出されます。サイクルが検出されると、選択した範囲から新たなVIを作成するか、VIの作成をキャンセルするかを尋ねるダイアログボックスが表示されます。新たなVIを作成することを選択した場合は、選択範囲から名称未設定の新たなVIが作成されます。元のダイアグラムの選択した項目は、そのまま残されます。

ループ内の属性ノード

選択範囲内のループに属性ノードが含まれないようにしてください。

属性ノードは呼び出し側の VI に残されてサブ VI に配線されるため、サブ VI を実行してもループの反復のたびに属性ノードが変更されることはありません。このような場合は、サブ VI は作成できません。

非論理的な選択

意味のない選択範囲をサブ VI に変換しないようにしてください。たとえば、シーケンスストラクチャを含めずに、シーケンスストラクチャ内のあるオブジェクトとシーケンスストラクチャ外部の別のオブジェクトで構成される選択範囲を変換しても意味がありません。

このような場合は、問題について説明する画面が表示されます。

ループ内のローカル変数とフロントパネル端子

選択範囲の中にループ内のローカル変数やフロントパネル端子を含めないようにしてください。選択したフロントパネル端子やローカル変数は呼び出し側の VI に残されてサブ VI に配線されるため、サブ VI を実行してもループの反復のたびにこれらの項目の値が変更されることはありません。選択範囲にこれらのオブジェクトが含まれていると、呼び出し側の VI の機能に変更が生じる可能性があります。

このような場合は、作業を続行するかキャンセルするかを尋ねる警告画面が表示されます。

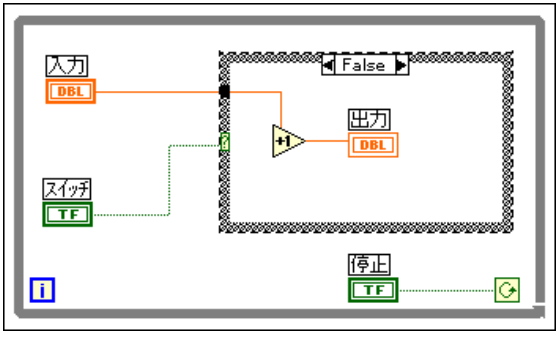
属性ノード、ローカル変数、またはフロントパネル端子を含む Case ストラクチャ

値が書き込まれる属性ノード、フロントパネル端子、またはローカル変数を含む Case ストラクチャを選択範囲の中に含めないようにしてください。

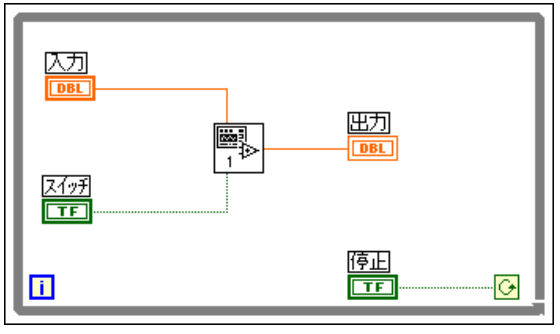
このような項目を含む範囲を選択した場合、オブジェクトはサブ VI に配線されるため、サブ VI を実行すると常にそのオブジェクトに決まった値が書き込まれます。オブジェクトは元のブロックダイアグラム中に存在しますが、値を書き込むことができるのは Case ストラクチャの TRUE 部分または FALSE 部分が実行された場合に限られます。そのため、呼び出し側の VI の機能に変更が生じる可能性があります。

そのような場合は、作業を続行するかキャンセルするかを尋ねる警告画面が表示されます。作業を続行する場合は、すべての場合に値が提供されるようにサブ VI を編集する必要があります。

次の図のような場合について考えてみましょう。



ここでは、サブVIに変換するためにCaseストラクチャが選択されています。フロントパネル端子の出力は元のダイアグラムに残され、次の図で示すようにサブVIを介してコネクタに接続されます。



元のブロックダイアグラムでは、ケースのFALSE部分が実行された場合にのみ出力に値が書き込まれます。一方、選択範囲をサブVIに変換した場合は、サブVI内でケースのTRUEまたはFALSEのどちらの部分が実行された場合にも出力に値が書き込まれます。別のケースにデータを渡すためには、サブVIを編集する必要があります。

階層ウィンドウを使用する

階層ウィンドウは、Type Def やグローバル変数も含め、メモリ内のすべての VI の呼び出し階層を図で表示します。

階層ウィンドウでは、さまざまな外観の表示を指定することができます。たとえば、レイアウトを左右または上下に表示したり、`vi.lib` の VI、グローバル変数、あるいは Type Def を含めるか除外するかを指定することも可能です。

このほか階層ウィンドウの便利な機能としては、**VI 設定 ...**、**アイコン編集 ...**、**VI 情報を表示 ...**、**文書を印刷 ...** などのメニュー項目、階層ノードを別の VI のブロックダイアグラムにサブ VI としてドラッグまたはコピーして貼り付ける機能、階層ノードを名前で検索する機能などがあります。

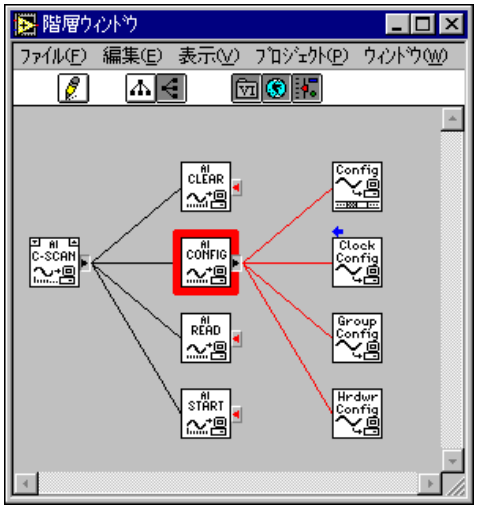
階層ウィンドウを開く

階層ウィンドウは、下記のいずれかの方法で開くことができます。

- **プロジェクト→ VI 階層を表示**を選択します。階層ウィンドウに、現在アクティブなウィンドウの VI がフォーカスノードとして表示されます。
- サブ VI、グローバル変数、または Type Def のポップアップメニューで、**VI 階層を表示**を選択します。階層ウィンドウに、選択したサブ VI、グローバル変数、または Type Def がフォーカスノードとして表示されます。
- 階層ウィンドウがすでに開いている場合は、**ウィンドウ**メニューの開いているウィンドウのリストで階層ウィンドウを選択することにより、画面のいちばん前に表示することができます。

階層ウィンドウの左右表示と上下表示の切り替えは、**表示**メニューでいずれかの項目を選択するか、またはウィンドウの上にある**横レイアウト**または**縦レイアウト**のボタンを使用して行います。

左右表示では、サブ VI は呼び出し側の VI の右側に表示されます。上下表示では、サブ VI は呼び出し側 VI の下に表示されます。サブ VI は、呼び出し側の VI と必ず線で結ばれています。次の図は、左右表示の階層ウィンドウを示したものです。



ノードのそばの矢印ボタンと矢印は、表示と非表示の状態を示します（下記参照）。

- ノードの方を向いた赤の矢印ボタンは、サブVIの一部または全部が非表示状態であることを示します。このボタンをクリックすると、ノードの一番下のサブVIが表示されます。

- ノードのサブVIの方を向いた黒の矢印ボタンは、直属のサブVIが全部表示されていることを示します。

- ノードの発呼者側を向いた青い矢印は、そのノードに画面に表示されていない別の発呼者が存在することを示します。

ノードにサブVIがない場合は、赤と黒の矢印ボタンは表示されません。

ノードに対して動作を実行すると、そのノードはフォーカスノードになり、赤い太枠で囲まれます。別のノードに対して動作を実行すると、そのノードのフォーカスは解除されます。

階層ウィンドウでカーソルをオブジェクト上に移動してアイドル状態にすると、ノードアイコンの下にノードの名前が表示されます。また、**表示**メニュー項目を使用すると、名前の代わりにパス全部を表示させることもできます。

階層ウィンドウのいずれかのアイコンをダブルクリックするとそのVIが開きます。

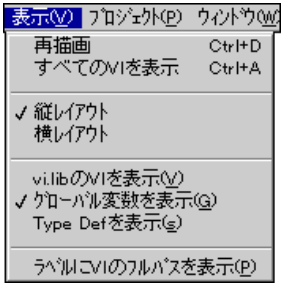
ヘルプウィンドウを開いて、カーソルをアイコンの上に移動すると、そのVIに関するヘルプ情報が表示されます。

階層ウィンドウのオプション

階層ウィンドウには、階層ウィンドウの表示を制御するためのオプションや、ウィンドウに表示されているノードに対して動作を実行するためのオプションがあります。オプションは、**表示**メニュー、階層ウィンドウ上部のボタン、ノードのポップアップメニュー、あるいはマウスをクリックし使用することができます。また**表示**メニューは階層ウィンドウの空白領域をポップアップして呼び出すこともできます。

表示メニューのオプション

表示メニューには、階層ウィンドウの表示に関する下記の項目があります。これらの項目の多くは、ウィンドウ上部のボタンによっても選択することができます。



- **再描画** — 線の交差が最小限に、対称性が最大限になるようにウィンドウのレイアウトをし直します。階層ノードに対して一連の動作を実行した後で使用すると便利なオプションです。フォーカスノードが存在する場合は、そのノードが表示させるようにウィンドウが自動的にスクロールされます。フォーカスノードが存在しない場合は、サブVIの最初のルートが表示されるようにスクロールされます。
- **すべてのVIを表示** — 表示されていないすべてのVIを表示します。このメニュー項目が選択されると、フォーカスノードは存在しません。このメニュー項目は、表示メニューの他の設定には影響を与えません。つまり、階層に含まれていない`vi.lib`のVI、グローバル変数、Type Defは、このオプションを選択しても表示されません。
- **縦レイアウト** — 呼び出し側のVIをサブVIの上に表示し、ノードを上から下に並べます。
- **横レイアウト** — 呼び出し側のVIはサブVIの左に表示し、ノードを左から右に並べます。
- `vi.lib`の**VIを表示** — 階層ウィンドウに`vi.lib`内のVIを表示する／しないを切り替えます。

- **グローバル変数を表示** — 階層ウィンドウにグローバル変数を表示する／しないを切り替えます。
- **Type Defsを表示** — 階層ウィンドウにType Defを表示する／しないを切り替えます。
- **ラベルにVIのフルパスを表示** — 各階層ノードのヒントラベルにVIの完全パスを表示するか、VIの名前だけを表示するかを選択します。

ツールバーのボタン

階層ウィンドウのツールバーには、ウィンドウの表示を制御するためのボタンがあります。

これらのボタンの説明は下記の通りです。ボタンのなかには**表示**メニュー項目と同じ機能を持つものもあります。

- **レイアウトのやり直し** — 線の交差が最小限に、対称性が最大限になるようにウィンドウのレイアウトをし直します。階層ノードに対して一連の動作を実行した後で使用すると便利なオプションです。フォーカスノードが存在する場合は、そのノードが表示されるようにウィンドウが自動的にスクロールされます。フォーカスノードが存在しない場合は、サブVIの最初のルートが表示されるようにスクロールされます。

- **縦レイアウト** — 呼び出し側のVIをサブVIの上に表示し、ノードを上から下に並べます。

- **横レイアウト** — 呼び出し側のVIはサブVIの左に表示し、ノードを左から右に並べます。

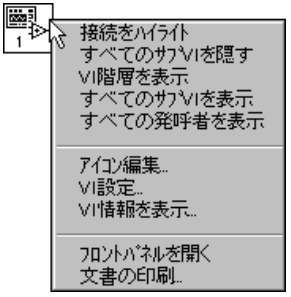
- **vi.libのVIを表示** — 階層ウィンドウに`vi.lib`内のVIを表示する／しないを切り替えます。

- **グローバル変数を表示** — 階層ウィンドウにグローバル変数を表示する／しないを切り替えます。

- **Type Defsを表示** — 階層ウィンドウにType Defを表示する／しないを切り替えます。

階層ノードのポップアップメニュー

階層ウィンドウに表示されているノード（あるいはサブVI、グローバル変数、Type Def）をポップアップすると、表示制御項目や選択したノードに関係のあるコマンド実行項目を含むメニューが表示されます。



各メニュー項目の機能は下記の通りです。

- **機能をハイライト** ― 選択されたノードをフォーカスノードにして、一段下の発呼者およびサブVIを接続しているエッジを赤くハイライト表示します。
- **一段下のサブVIを表示** ― ノードがサブVIの一部あるいは全部を隠している場合に、ノードを広げて一段下のサブVIをすべて表示します。ノードとサブVIを接続するエッジは赤でハイライト表示されます。この項目を選択すると、メニューの表示は**すべてのサブVIを隠す**に切り替わります。
- **すべてのサブVIを隠す** ― ノードの一段下のすべてのサブVIが表示されている場合に、ノードを畳んでサブVIチェーン全体を隠します。この項目を選択すると、メニューの表示は**一段下のサブVIを表示**に切り替わります。
- **VI階層を表示** ― 選択されたノードをフォーカスノードにして、呼び出しチェーンとサブVIチェーンに属するノードを表示します。無関係のルートも表示されますが、これらのルートのサブVIはすべて隠されています。ノードの呼び出しチェーンとサブVIのチェーンを接続するエッジは赤でハイライト表示されます。
- **すべてのサブVIを表示** ― 選択されたノードをフォーカスノードにして、サブVIチェーン全体を広げます。ノードのサブVIチェーンを接続するエッジは赤でハイライト表示されます。
- **すべての発呼者を表示** ― 選択されたノードをフォーカスノードにして、呼び出しチェーン全体を広げます。呼び出しチェーンを接続するエッジは赤でハイライト表示されます。
- **アイコン編集...** ― ノードのアイコンを編集するためのアイコンエディタを表示します。

- **VI設定...** ― ノードのVI設定ダイアログボックスを表示します。
- **VI情報を表示...** ― ノードのVI情報を表示ダイアログボックスを表示します。
- **フロントパネルを開く** ― VI、グローバル変数、またはType Defのフロントパネルを開きます。
- **文書を印刷** ... ― VIの印刷したい部分を選択するための文書を印刷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についての詳細は、「第5章　VIの印刷および文書作成」の「文書を印刷する」の項を参照してください。

マウスクリックによるノードの操作

ノードの選択、あるいはノードのコピーやドラッグは、マウスクリックによって行うことができます。また、ポップアップメニューのオプションを選択するためのショートカットとして使用することもできます。マウスクリックによる操作を実行する場合は、位置決めツールがあるところでは位置決めツールを使用します。

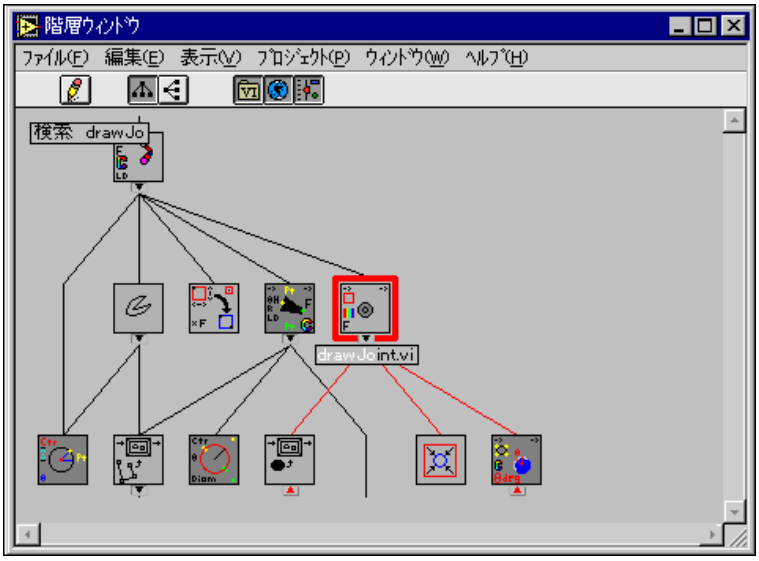
ショートカットとして使用できるマウスクリックは下記の通りです。ポップアップメニュー項目に関する詳細な説明については、この章の「階層ノードのポップアップメニュー」の項を参照してください。

- 赤の矢印ボタンをクリックすると、一段下のサブVIを表示動作が実行されます。
- <Shift>キーを押しながら赤または黒の矢印ボタンをクリックすると、すべてのサブVIを表示動作が実行されます。
- <Ctrl>キー**（Windows）**、<option>キー**（Macintosh）**、<meta>キー**（Sun）**、または<Alt>キー**（HP-UX）**を押しながらノードをクリックすると、VI階層を表示動作が実行されます。
- ノードをダブルクリックすると、フロントパネルを開く動作が実行されます。
- ノードをシングルクリックすると、選択したノードをブロックダイアグラムにドラッグしたり、クリップボードにコピーしてサブVIとして使用することができます。
- 複数のノードを選択して別のブロックダイアグラムやフロントパネルにコピーしたい場合は、<Shift>キーを押しながらそれぞれのノードをマウスでクリックします。また、長方形をドラッグして複数のオブジェクトを選択することもできます。
- <Tab>キーを押すと、**ツール**パレットの位置決めツールとスクロールツールが切り替わります。

階層ウィンドウで VI を検索する

検索は、ノードの名前を入力するだけで開始することができます。小さな検索ウィンドウが現れ、入力したテキストが表示されます。検索は直ちに実行され、該当するノードが見つかるとそのノードがハイライト表示され、ヒントラベルに名前が表示されます。

次の図は、階層ノードの検索機能で drawJoint.vi という名前のノードを検出したときの状態を示したものです。



検索は、キーボードからの入力によって実行されます。検索ウィンドウに表示されている名前と一致するノードが見つからなかった場合は、警告音が鳴り、それ以上文字を入力することはできません。<Backspace> または <Delete> キーを使用すると、文字を削除して名前を入力し直すことができます。

一定時間内にキーを押さなかった場合は、検索ウィンドウは自動的に閉じます。<Esc> キーを押すと、直ちに検索ウィンドウを閉じることができます。

一致する名前のノードが見つかったら、右または下向きの矢印キー、または <Enter> キー（**Windows** および **Sun**）、または <Return> キー（**Macintosh** および **HP-UX**）を押して次に一致するノードの検索を開始することができます。1 つ前の一致するノードに戻りたいときは、左または上向きの矢印キー、または <Shift-Enter>（**Windows** および **Sun**）、または <Shift-Return>（**Macintosh** および **HP-UX**）を押します。

VI、オブジェクト、およびテキストを検索する

複数のサブVIあるいは1つの大きなVIからなるアプリケーションを作成する際には、特定のオブジェクトや文字列の検索が必要になる場合があります。下記のオブジェクトについては、**プロジェクト→検索**コマンドを使用して指定した名前と一致するすべてのインスタンスを検索することができます。

- VI
- 組み込み関数
- Type Def
- グローバル変数
- ローカル変数
- 属性ノード
- ブレークポイント
- フロントパネル端子
- テキスト

Type Defの詳細については「第24章　カスタム制御器とType Def」の「Type Def」の項を参照してください。ローカル変数およびグローバル変数の詳細については、「第23章　グローバル変数とローカル変数」を、また属性ノードについては「第22章　属性ノード」をそれぞれ参照してください。ブレークポイントについては「第4章　VIとサブVIの実行およびデバッグ」の「ブレークポイントツールを設定する」の項を参照してください。

検索コマンドのほかにも、多くのオブジェクトにはポップアップメニューが用意されており、これを使用すると関連オブジェクトを素早く検索することができます。たとえば、制御器のポップアップメニューには、その制御器に関係のあるローカル変数や属性ノード、対応する端子が検索できるメニュー項目が用意されています。

検索ダイアログボックス

検索ダイアログボックスを画面に呼び出すためには、**プロジェクト→検索...**を選択するか、あるいは<Ctrl-f>**（Windows）**、<command-f>**（Macintosh）**、<meta-f>**（Sun）**、または<Alt-f>**（HP-UX）**を押します。

また、ショートカットとして、テキストやオブジェクトを選択してからダイアログボックスを呼び出すこともできます。このダイアログボックスには、先に選択した検索したいテキストまたはオブジェクトが自動的に表示されます。

VIおよびその他のオブジェクトを検索する

VIまたはその他のオブジェクトを検索する場合は、検索ダイアログボックスの**オブジェクト**ボタンをクリックします。

検索したいオブジェクトを選択するには、**オブジェクトを選択**の横のボタンをクリックします。**オブジェクト選択**メニューが表示されます。

このメニューで選択できる項目は下記の通りです。

- **関数** — **関数**パレットを表示します。組み込み関数、VI、またはカスタマイズして**関数**パレットに表示したオブジェクトを選択できます。
- VI — メモリに入っているすべてのVI(ただし`vi.lib`に入っているVIは除く）のパレットを表示します。

- **Type Def** ― メモリに入っているすべてのType Def（ただし`vi.lib`に入っているType Defは除く）のパレットを表示します。
- **グローバル** ― メモリに入っているすべてのグローバル変数（ただし`vi.lib`に入っているグローバル変数は除く）のパレットを表示します。
- **vi.lib内のオブジェクト** ― `vi.lib`に入っているすべてのVI、Type Def、およびグローバル変数を表示します。
- **他** ― 属性ノード、ブレークポイント、およびフロントパネル端子を選択するためのメニューを表示します。
- **名前でVIを選択...** ― 名前でVIを選択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は、メモリ内のすべてのVIの名前をアルファベット順に表示します。キーボードからオブジェクト名を入力すると、リスト内の希望する項目に即座にジャンプすることができます。表示の3つのチェックボックス（**VI**、**グローバル**、および**Type Def**）を使用すると、リストボックスに表示したいオブジェクトの種類を種類別に表示させることができます。

テキストを検索する

特定のテキストを検索したい場合は、検索ダイアログボックスの**テキスト**ボタンをクリックします。

文字列検索オプション

検索したいテキストはキーボードから入力することができます。またオプションとして、検索の対象を特定の領域のVI（フロントパネル、ブロックダイアグラム、VI情報を表示テキスト、または履歴情報）やラベルや説明などのオブジェクトの要素にしぼり込むこともできます。

検索対象の文字列は、下記のメニュー項目を使用して指定することができます。

- **大文字と小文字を区別する** — 大文字と小文字を判別し、入力した文字列と正確に一致する文字列だけを検索します。
- **完全に一致する単語だけを検索する** — 前または後ろにスペースやプラス記号（+）など英数字以外の文字が付いているか、またはラインで始まるか終わるかしている文字列のみを検索します（このオプションが選択されていなければ、コマンドはより長い文字列の一部分であろうとすべての文字列と照合します）。
- **正規表現** — 文字列を正規表現として扱います。正規表現の仕様は、Match Pattern関数と同じです（これらの仕様については、**ヘルプ→オンラインリファレンス→関数およびVIのリファレンス**を参照してください）。

テキスト検索オプション

オプション...ボタンをクリックすると、テキスト検索オプション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には、個々のVIの文字列検索範囲を指定するためのメニュー項目があります。

これらのメニュー項目は、検索の対象となる場所を指定します。

- **VI内の検索項目** — VIのフロントパネル、ブロックダイアグラム、VI情報テキスト、履歴テキストを検索対象にするかどうかを指定します。少なくとも項目のいずれか1つが選択されていなくてはなりません。
- **オブジェクトのラベルを検索** — 表示されているオブジェクト名ラベルや隠されているオブジェクト名ラベルを検索対象にするかどうかを指定します。
- **オブジェクトのデータと部品を検索** — 表示されているデータや部品、あるいは隠されているデータや部品を検索対象にするかどうかを指定します。オブジェクトのデータや部品には、オブジェクト名ラベル以外のオブジェクトに属するものがすべて含まれます。また、オプションで様々なオブジェクトについての説明を検索対象にすることもでき

ます。**オブジェクトのラベルを検索**と**オブジェクトのデータと部品の検索**では、少なくとも項目のいずれか1つが選択されていなければなりません。

☞ **注**　**テキスト検索オプションは、検索ごとに保存されます。したがって、あるはずのテキストの文字列が見つからない場合は、テキスト検索オプションが正しく設定されているか確認してください。**

検索範囲を絞り込む

検索範囲リングメニューの3つのオプションは、検索の対象となるVIを指定します。

- **メモリ上のすべてのVI** — 現在メモリにロードされているすべてのVIを検索対象として指定します。2つのチェックボックスにより vi.lib および階層ウィンドウのVIを検索対象に含めるか除外するかを指定することができます。
- **選択されたVI** — 指定した組み合わせのVIを選択して検索することができます。階層ウィンドウは、検索範囲に含めることも除外することもできます。**選択...**ボタンを押すと、検索ダイアログボックスに**VIを選択**メニューが表示されます。
- **特定のVI** — 特定のVIだけを検索します。このリングメニューに検索ダイアログボックスを呼び出したときにアクティブであったVIの名前が表示されます。

検索ボタンをクリックすると、指定したテキストの検索が開始されます。

検索結果ウィンドウ

検索終了後複数の該当項目が見つかった場合は、次に示すような検索結果ウィンドウに検索結果がすべて表示されます。該当する項目が1件しか見つからなかった場合は、検索結果ウィンドウを介さずに直ちに結果が表示されます。

検索結果ウィンドウは、**プロジェクト→検索結果...** を選択して呼び出すこともできます。

このウィンドウでは、次のメニュー項目が使用できます。

- **消去** — 検索結果を保存するために使用していたメモリ領域を消去し、解放します。
- **...ヘジャンプ** — 現在選択されている検索結果を反転表示します。項目をダブルクリックして反転表示に変えることもできます。すでに反転表示になっている項目にはチェックマークが付けられます。
- **検索...** — 検索ダイアログボックスを表示します。
- **停止** — 検索の実行中は、**検索...** ボタンのかわりにこのボタンが表示されます。検索対象の VI の数が多くて時間がかかる場合は、該当する項目が見つかるたびに検索結果ウィンドウの内容が更新され、検索状況が表示されます。検索は、**停止** ボタンをクリックして中断することができます。

検索結果に含まれているオブジェクトあるいは VI を削除すると、その検索結果の項目は無効になります。ただし、テキストを変更しても検索結果の内容は更新されないため、検索条件に適合しなくなった検索結果が反転表示される場合がありますので注意してください。そのような場合は、新たに検索を実行すると検索結果が更新されます。

前後の検索項目を表示する

プロジェクト→次を検索 を選択すると、検索リストの前後の項目が反転表示されます。隠されているテキストについては、そのテキストを表示するために必要なすべてのオブジェクトが一時的にグレーで表示され、テキストが選択されます。マウスボタンをクリックするか、いずれかのキーを押すとオブジェクトは再度隠された状態になります。

グローバル変数、ローカル変数、および属性ノードの検索ポップアップメニュー

フロントパネルの制御器、表示器、グローバル変数、ローカル変数、および属性ノードについては、検索のポップアップメニュー項目が使用できます。ローカル変数または属性ノードをポップアップして、対応する制御器やフロントパネル端子、あるいはその他のローカル変数および属性ノードを検索することができます。反対に、フロントパネルの制御器や表示器には、対応するすべてのローカル変数および属性ノードを検索するためのポップアップメニューがあります。グローバルのノードからは、それに対応するグローバルの定義、あるいはその他のすべてのグローバルノードを検索することができます。反対に、グローバルの制御器には、メモリ中の対応するすべてのグローバルリファレンスを検索するためのポップアップメニューがあります。複数の項目が検出された場合は、検索結果ウィンドウが表示されます。

4

VIとサブVIの実行およびデバッグ

この章では、VIの操作およびデバッグについて説明するとともに、VIやサブVIを特殊なモードで実行するためのセットアップについて説明します。

VIを実行する

VIを実行する


実行ボタン

VIは、**操作→実行**を選択するか、または実行ボタンをクリックすることにより実行できます。Gは、必要に応じてVIをコンパイルします。


VIを最上位で実行中

VIを最上位で実行している間（つまりそのVIは発呼者がないのでサブVIでないとき）は、**実行**ボタンは左記のボタンに変わります。


VIの発呼者が実行中

VIをサブVIとして実行しているときは、実行ボタンは左記のボタンに変わります。


連続実行ボタン

連続実行ボタンを押すと、実行を停止するか一時中断するまでVIが繰り返し実行されます。


停止ボタン

停止ボタンを押すと、**最上位VI**の実行が停止されます。VIが最上位で実行中の複数のVIによって使用されているときは、このボタンはグレーで表示されます。

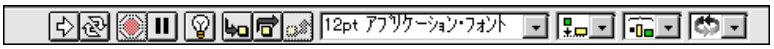

一時停止ボタン

このボタンを押すと、実行が一時中断されます。再度このボタンを押すと、いつでも実行を再開することができます。

複数のVIを同時に実行することができます。その場合は、最初のVIの実行を開始した後、次に実行したいVIのフロントパネルまたはブロックダイアグラムに切り替え、前述の方法で実行を開始します。ただし、サブVIを最上位のVIとして実行した場合は、呼び出し側のVIすべてはサブVIの実行が終了するまで壊れた状態になりますので注意してください。サブVIを最上位のVIおよびサブVIとして同時に実行することはできません。

VIが実行可能な状態で正しく実行されないときは、本章の「壊れたVIを修正する」および「実行可能なVIをデバッグする」の項を参照してください。

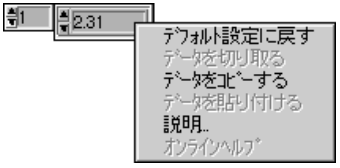
VIの編集時には、フォントリング、整列リング、間隔リング、再順序リングなど、VI編集用のツールがツールバーに表示されます。



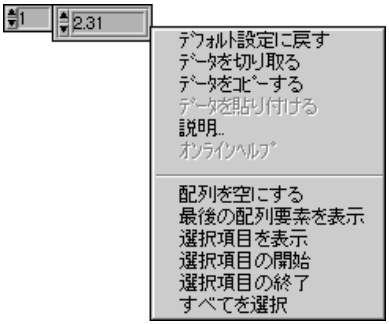
VIを実行すると、これらのツールは次に示すようなデバッグツールに変わります。


実行ボタン

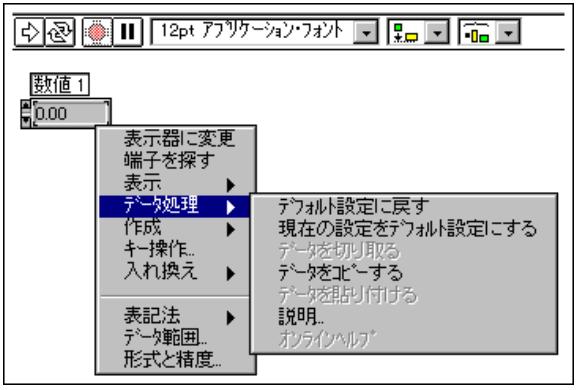
VIの実行中に制御器をポップアップすると、すべてのオブジェクトのメニューオプションが簡略化されていることがわかります。これらは、VIを編集する際にポップアップメニューに表示される**データ処理**サブメニューのオプションと同じです。



このメニューには、制御器の内容の切り取り、コピー、貼り付けなどを行うためのオプションや、制御器をデフォルト値に設定したり、制御器に関する説明を呼び出すためのオプションがあります。複雑な制御器では、これらのオプションのほかにさらにいくつかのオプションが追加されます。たとえば、配列のメニューには、値の範囲をコピーするためのオプションや、配列の最後の要素に移動するためのオプションがあります。



VIの編集時のオブジェクトのポップアップメニューには、次の図で示すように編集メニューと**データ処理**サブメニューが含まれています。このサブメニューには、VIの実行時に表示されるメニュー項目と、**現在の設定をデフォルト設定にする**という追加オプションが含まれます。



メニューバーの**操作**メニューには、現在のVIを実行するためのコマンドが含まれています。以下の項では、実行に関連したいくつかの基本的な操作について説明します。

VIを停止する


実行停止ボタン

通常の状況では、VIを最後まで実行させますが、実行を直ちに中断する必要がある場合は、**実行停止**ボタンをクリックするか、または**操作→実行停止**を選択します。**実行停止**コマンドは、即時に最上位のVIを中断させます。中断されたVIは、その作業が完了していない可能性が高いため、このVIが生成したデータを信頼することはできません。Gはそのとき開いているファイルを閉じ、実行中のデータ集録を中断しますが、**実行停止**ボタンやメニューの**実行停止**オプションに依存するVIを作成することは避けてください。たとえば、オペレータが停止するまでWhileループ中で無限に繰り返し実行するVIを作成する場合は、ループの条件付き端子をフロントパネルのブールスイッチに接続するようにします。

オペレータが間違って**実行停止**ボタンをクリックしてもVIの実行が停止しないようにするには、VIのフロントパネルのアイコンペーンのポップアップメニューで**VI設定→ウィンドウオプション→停止ボタンを表示する**のチェックボックスの選択を解除し、実行停止ボタンが画面に表示されないようにしておきます。詳しくは、「第6章　VIおよびサブVIをセットアップする」の「ウィンドウオプション」の項を参照してください。

VIを繰り返し実行する


連続実行ボタン

VIを繰り返し実行するためには、**連続実行**ボタンをクリックします。VIは直ちに実行を開始し、VIの実行中は**連続実行**ボタンの矢印が白から黒に変わります。VIを停止したいときは、**連続実行**ボタンを再度クリックします。VIは、正常に終了すると停止します。


連続実行中のVI

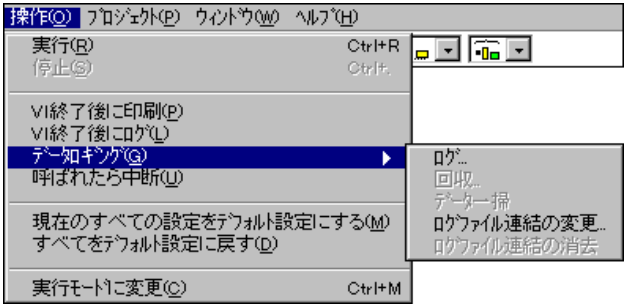
連続実行中のVIの動作、およびツールバーの状態は、**実行**ボタンまたは**操作→実行**コマンドを使用して実行を一回行ったときと同じになります。

フロントパネルのデータロギング

フロントパネルのデータロギングは、VIのすべてのフロントパネル制御器のタイムスタンプとデータをデータログファイルに保存します。それぞれが異なるテストからのデータを記録した、複数のデータログファイルを作成することができます。記録したデータは、VIを使用した対話方式の回収、プログラムによるデータ回収、または標準のファイルI/O関数を使用して回収することができます。

各VIは、フロントパネルのデータが記録されているデータログファイルの場所を示すログファイル連結を保持します。データロギングが有効になっているときにこの連結がない場合は場所が指定されていないことを示し、ファイルの場所を指定するためのプロンプトが表示されます。

フロントパネルのデータロギングの構成および制御は、次の図で示すように**操作**メニューとその**データロギング**サブメニューを使用して行います。



VI に対する自動データロギングが有効になっていると、VI の実行が完了するたびに VI のフロントパネルのデータが記録されます。VI に対して自動データロギングが有効になっているかどうかは、**操作→VI 終了後にログ**オプションで確認できます。**操作→ VI 終了後にログ**を選択すると、自動データロギングが無効のときには有効になり、有効のときには無効になります。

VI のフロントパネルデータを対話形式で記録するためには、**操作→データロギング→ログ ...** を選択します。

☞ **注**　**フロントパネルのデータロギングでは、波形チャートは一度に 1 つのデータポイントだけを返します。配列をチャート表示器に配線した場合は、そのレコードのチャートに表示される配列のサブセットがログデータに含まれます。**

VI のログファイルの連結は、**操作→データロギング→ログファイル連結の変更 ...** メニュー項目を使用して変更することができます。

入力ボタン

レコード番号を入力したあと、**入力**ボタンをクリックするか、または数値キーパッド上の<Enter>キーを押すと、特定のレコードを選択することができます。


削除ボタン

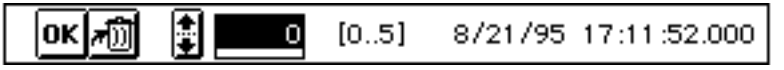

削除マークのついたレコードボタン

削除ボタンをクリックすると、特定の消去するレコードにマークを付けることができます。選択したレコードが消去するようにマークされているときは、**削除**ボタンはいっぱいになったごみ箱として表示されます。いっぱいになったごみ箱をクリックすると、選択したレコードの削除マークを解除することができます。選択したレコードは、**操作→データロギング→データー掃 ...** オプションを選択するか、または **OK** ボタンをクリックするか**再度回収 ...** を選択してデータ回収モードから抜け出さない限り消去されません。データ回収モードから抜け出す際に、削除マークの付いたレコードがある場合は、マークの付いたレコードを削除するかどうかを尋ねるプロンプトが表示されます。

データログを表示していたVIに戻るには、OKボタンをクリックします。

操作→ログファイル連結の消去を選択すると、VIとログファイルの関係が解除されます。関係が解除されていると、次にVIのフロントパネルのデータをログする際にログファイルを指定するよう指示するプロンプトが表示されます。

記録したデータを対話形式で見るためには、メニューから**操作→回収...**を選択します。ツールバーが、次に示すようなデータ回収のツールバーに変わります。



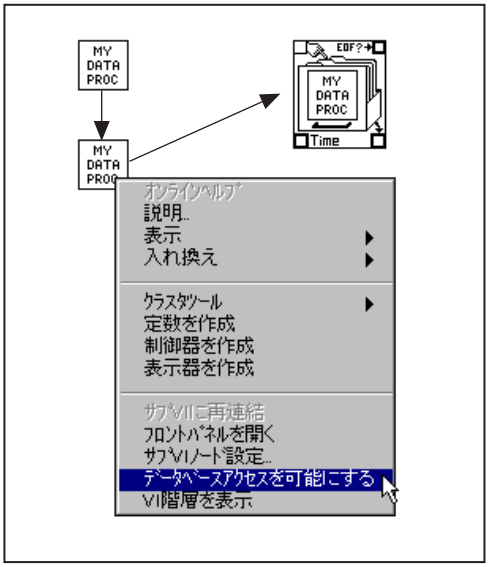
ハイライト表示された数字は、現在選択および表示されているているレコードの番号を示します。カギカッコ内の数字は、記録したレコードの範囲を示します。日付と時刻は、選択したレコードが記録された日付と時刻を示します。上向き矢印をクリックすると次のレコードを見ることができ、下向き矢印をクリックすると前のレコードを見ることができます。また、キーボード上の上向きまたは下向き矢印キーを使用することもできます。

プログラムでデータを回収する

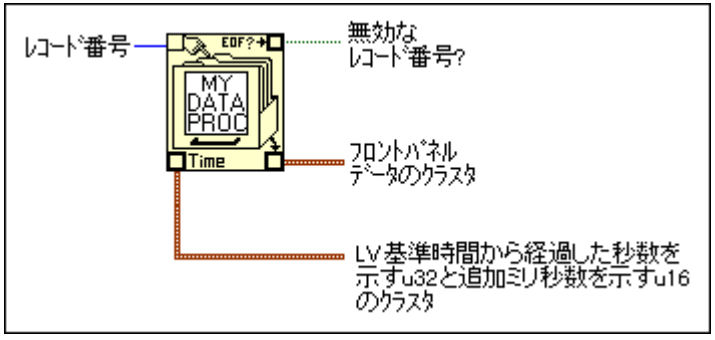
VIから記録したデータは、**データベースアクセスを可能にする**オプション、またはファイルをデータログファイルとして読み取る標準のファイルI/O関数を使用して回収することができます。

データベースにアクセスする

データは、**データベースアクセスを可能にする**オプションを使用して回収することができます。このオプションは、次の図に示すように、フロントパネルのデータをVIのデータログファイルに記録したサブVIのポップアップメニューにあります。データログファイルの周りには、ファイルキャビネットの形をしたハローが表示されます。この**ハロー**には、データログファイルのデータにアクセスするための端子があります。呼び出しダイアグラムを実行すると、サブVIは実行されません。代わりに、データログファイルの指定したレコードからデータを回収します。また、データが記録された時刻、および指定したレコード番号が有効かどうかを示すブール値も返されます。



次の図は、記録したデータにアクセスするためのハロー端子を示したものです。



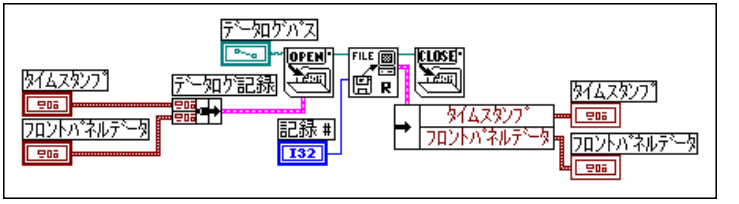
サブ VI の記録したレコードの数を n とすると、*–n* ～ *n–*1 の任意の数をレコード番号端子に配線することができます。**最初に**記録されたレコードを基準にしてレコードにアクセスする場合は、負でない数のレコード番号を使用します。0 は最初のレコード、1 は 2 番目のレコードを指し、*n–*1 は最後のレコードを指します。

一方、最後に記録されたレコードを基準にしてレコードにアクセスする場合は、負のレコード番号を使用します。この場合、-1 は最後のレコード、-2 は最後から 2 番目のレコードを指し、*–n* は最初のレコードを指します。レコード番号の端子に *–n* ～ *n–*1 の範囲外の番号を配線すると、無効なレコード番号?の出力が TRUE に設定され、データは回収されません。

ファイル I/O 関数を使用してデータを回収する

VI のフロントパネルから記録したデータは、G のファイル I/O 関数を使用して回収することもできます。フロントパネルデータロギングで作成されたデータログファイルの個々のレコードは、タイムスタンプおよびフロントパネルデータを含むクラスタです。タイムスタンプのクラスタは、G の基準時間からの経過時間を秒で示す符号なしの 32 ビット整数とミリ秒で示す符号なしの 16 ビット整数で構成されます。このクラスタの後ろには、パネル順のフロントパネルデータのクラスタが続きます。

フロントパネルデータロギングで作成されたデータログファイルのレコードには、プログラムによって作成されたデータログファイルにアクセスする際に使用するのと同じ G のファイル I/O 関数を使用してアクセスすることができます。次の例で示すように、上記で説明した正しいタイプを File Open 関数に対するタイプ入力として入力します。



VI をデバッグする

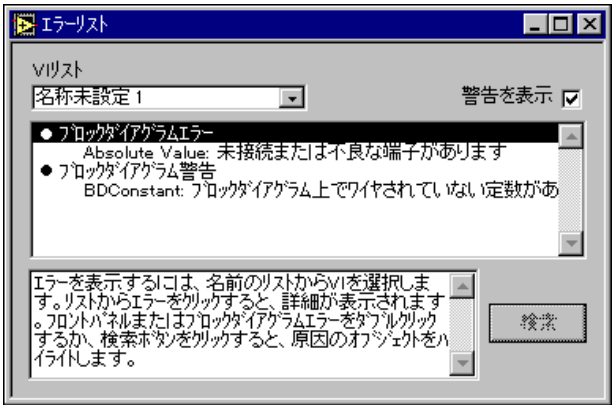
この項では、仮想計測器のエラーや問題の解決方法について説明します。

壊れた VI を修正する

壊れた VI は、コンパイルしたり実行したりすることはできません。通常、VI の作成中または編集中は、ダイアグラムのすべてのアイコンが配線されるまで**VI 実行**ボタンは壊れたまま表示されます。アイコンを配線したあともこの壊れた実行ボタンが表示される場合は、**編集→不良ワイヤの削除**を選択するか、または<Ctrl-b>**（Windows）**、<command-b>**（Macintosh）**、<meta-b>**（Sun）**、<Alt-b>**（HP-UX）**を押すことにより、壊れた VI を修正できる場合があります。


壊れた実行ボタン

VI が壊れた原因を知りたいときは、**壊れた実行**ボタンをクリックします。次の図に示すように、すべてのエラーを列記したエラーリストウィンドウが表示されます。



警告ボタン

このウィンドウは、VI の**警告**ボタンをクリックするか、または**ウィンドウ→エラーリストを表示**を選択して呼び出すこともできます。VI の**警告**ボタンは、VI に警告すべき事態（オブジェクトのオーバーラップなど）が存在し、かつエラーリストウィンドウの**警告を表示**がチェックされている場合にのみ表示されます。**環境設定**ダイアログボックスを使用すると、デフォルト設定で警告が表示されるようにプログラムを構成することができます。

エラーの発生箇所を確認したいときは、そのエラーについて説明したテキストをダブルクリックします。関連ウィンドウが前面に表示され、エラーの原因となっているオブジェクトがハイライトで表示されてエラーが示されます。**VI リスト**のポップアップメニューで VI の名前を選択し、エラーおよび警告を確認します。

以下に、編集時に VI が破れる最も一般的な理由を示します。

- 入力を必要とする関数端子が配線されていない。たとえば、入力はすべて数学関数に配線する必要があります。他の設計を試す場合でも、ダイアグラム上に配線されていない関数を残したまま VI を実行することはできません。
- ブロックダイアグラムに壊れたワイヤ（データタイプが一致しないワイヤやどこにも接続されていないワイヤなど）が存在する。表示されていないワイヤスタブも含め、壊れたワイヤを**編集→不良ワイヤの削除**で取り除きます。
- サブ VI が壊れているか、あるいはサブ VI のアイコンをダイアグラムに配置したあとでそのサブ VI のコネクタを編集した。
- 表示されないオブジェクト、無効にしたオブジェクト、あるいは属性ノードによって変更したオブジェクトに問題がある。可能な場合は、属性ノードを使用して問題を取り除き、オブジェクトを復元します。

エラーメッセージの意味を理解する

次の表は、エラーリストウィンドウに表示されるエラーメッセージ、およびその意味する内容を示したものです。

表 4-1 エラーメッセージ

エラーメッセージ	意　　味
関数名：未配線または不良の端子があります。	必須入力が配線されていないか、またはタイプが適切ではありません。必須入力を正しいデータタイプに配線してください。
コードインタフェースノード：オブジェクトコードがロードされていません。	CIN のオブジェクトコードが正しくリンクされていません。ノードをポップアップしてコードをロードし直してください。
制御器：制御器がタイプ定義と一致しません。	制御器をタイプ定義と一致するように編集するか、ポップアップしてType def VIを更新するか、またはType Def VIを制御器から切り離します。
列挙に重複入力があります。	列挙制御器内のすべての項目は重複がなく、個別のものでなければなりません。
これ以外にもエラーがあります...	エラーウィンドウのリストには一度に 100 件のエラーしか表示されません。リストに表示されていないエラーは、リストに表示されているエラーが修復されたあとに表示されます。
Forループ：Nが未配線で、入力の指標付けトンネルがありません。	ループの反復実行回数を決定するためには、N が配線されているか、または配列が入力で指標付けされている必要があります。
グローバルまたはローカル変数：指定されたコンポーネントが存在しません。	VIをロードした後で変数の名前が変更されたため、グローバル変数またはローカル変数の名前が一致しません。変数をポップアップして別の名前を選択してください。
グローバル変数：サブVIが抜けています。	呼び出し側の VI をロードする際に、グローバルサブ VI が見つかりませんでした。原因としては名前の変更などが考えられます。サブ VI を正しいグローバルサブ VI と差し替えてください。
ノード：優先度がサブルーチンのVIは非同期ノードを持てません。	サブルーチンが同じ優先レベルのVIにはWaitなどの非同期関数を使用できません。
右シフトレジスタ：タイプが未定義です。	使用していないシフトレジスタを削除する必要があります。
右シフトレジスタ：左側のシフトレジスタの一部が配線されていません。	シフトレジスタには左側のすべての入力が入るか、またはどの入力も入らないかのいずれかでなければなりません。

表 4-1 エラーメッセージ（続き）

エラーメッセージ	意　　味
シーケンス:1つまたは複数のシーケンスローカルが割り当てられていません。	シーケンスストラクチャのローカル変数への配線がなされていません。使用しないシーケンスローカルは削除してください。
サブVI名：サブVIへの連結が不良です。	サブVIを最後にロードしたあとでサブVIのコネクタパターンが変更されています。または、サブVIの制御器または表示器がコネクタ端子に割り当てられていません。コネクタパターンを変更したことが原因である場合は、サブVIのポップアップメニューの**サブVIに再連結**コマンドを実行してください。
サブVI名：回帰的参照（廃棄してください）。	呼び出し側のVIの名前をそのいずれかのサブVIと同じ名前に変更したため、このVIが自分自身を再帰的に呼び出すかたちで登録されています(VIとサブVIが別々のディレクトリに入っている可能性があります)。VIをそのVIのブロックダイアグラムに配置しようとすると、Gは再帰を防ごうとします。
サブVI名：LabVIEWサブルーチンリンクエラー。	CINルーチンへのリンクに問題があります。
サブVI名：優先度がサブルーチンのVIは、優先度がサブルーチン以外のサブVIを呼び出せません。	サブVIをサブルーチンの優先順位で実行されるようにするか、または呼び出し側のVIをサブルーチン以外の優先順位に変更する必要があります。
サブVI名：サブVIはすでに実行しています。	サブVIの1つがすでに別のVIのサブVIとして実行されている場合、VIを実行することはできません。
サブVI名：サブVIは、パネル順位かクラスタ順位モードに入っています。	いずれかのサブVIのパネルまたはクラスタの順位を変更しているときには、VIを実行することはできません。
サブVI名：サブVIは対話式回収モードに入っています。	いずれかのサブVIが対話式回収モードのときには、VIを実行することはできません。
サブVI名：サブVIがありません。	呼び出し側のVIをロードする際にサブVIが見つかりませんでした。原因としては、名前の変更などが考えられます。サブVIを正しいサブVIと差し替えるか、または欠けているサブVIを探し出してそのサブVIを開いてください。
サブVI名：サブVIが実行不可能です。	サブVIが壊れています。サブVIを開き、エラーを削除してください。

表 4-1 エラーメッセージ（続き）

エラーメッセージ	意　　味
端子：フロントパネル上の関連付けられた配列またはクラスタに要素がありません。タイプが未定義です。	配列またはクラスタに制御器または表示器を配置する必要があります。
Type Def：有効なType Defを検出できません。	呼び出し側のVIをロードする際に、タイプ定義VIが見つかりませんでした。原因としては、名前の変更などが考えられます。目的のタイプ定義VIが見つかった場合はそのVIを開き、問題のある制御器をポップアップしてタイプ定義から切り離すか、または正しい制御器と差し替えてください。
(Un) Bundle By Name：空のクラスタ、または一部の要素に名前が付いていません。	Bundle By NameまたはUnbundle By Name関数に配線した入力クラスタに何も入っていないか、またはラベルの付いてないコンポーネントが入っています。
単位：誤った単位構文。	ノードのテキストが正しい単位表記になっていません。認識できない文字の直前に？が挿入されます。

本章に記載されていないメッセージが表示され、その意味がわからない場合は、ナショナルインスツルメンツにお問い合わせください。

実行可能なVIをデバッグする

プログラムを実行しても予想通りの結果が得られないといった場合でも、下記の手順を実行することによって問題を解決できることがあります。

- すべてのVI警告を取り除きます。エラーリストウィンドウで**警告を表示**を選択すると、VIに関する警告がリストボックスに表示されます。問題の原因を特定し、VIからその原因を取り除きます。
- ワイヤのパスをチェックし、ワイヤが正しい端子に接続されているかチェックします。操作ツールでワイヤを3回クリック（トリプルクリック）すると、そのワイヤのパス全体がハイライトで表示されます。ある端子から出ているように見えるワイヤが、実は別の端子から出ている場合もあります。ワイヤの末端がノードに接続されている部分をよく注意して確認してください。
- ヘルプウィンドウ（**ヘルプ→ヘルプを表示**）を使用して関数を正しく配線します。
- 関数やサブVIの未配線の入力のデフォルト値が正しい値になっているかチェックします。
- ブレークポイント、実行のハイライト表示、各ステップごとの実行といった手段を使用して、VIが計画通りに実行されるかチェックします。

- 本章の「プローブツールを使用する」の項で説明するプローブ機能を使用して、データの中間値を観察します。また、関数やサブVI（とくにI/Oを実行する関数やサブVI）のエラー出力もチェックします。
- さまざまな入力値を使用してVIやサブVIの動作を観察します。浮動小数点数値制御器については、通常の値の他にNaNおよび±Infという値も入力してチェックします。
- VIの実行速度が予想以上に遅い場合は、サブVIの実行のハイライト表示がオフになっていることを確認します。また、使用していないサブVIウィンドウは閉じておきます。
- 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、整数をさらに小さい整数に変換したりすることによってオーバーフローが発生していないか確認します。また、本章の「不定データを検出する」の項も参照してください。
- 制御器や表示器のデータの範囲、および範囲エラーに対する動作をチェックします。制御器や表示器が、エラーに対して予想通りの動作で応答しない場合があります。
- Forループの反復回数が誤って0になっていないか、空の配列を生成していないかチェックします。本章の「実行をハイライト表示する」および「不定データを検出する」の項も合わせてお読みください。
- シフトレジスタをループのある回の実行から別の回の実行までのデータを保存するように特に設定していない場合は、シフトレジスタが正しく初期化されているかどうかを確認します。
- 接続元および接続先のポイントのクラスタ要素の順番をチェックします。データタイプやクラスタサイズの不一致は編集時に検出されますが、同じタイプの要素の不一致は検出されません。クラスタの順番は、クラスタシェルのポップアップメニューの**クラスタ順位** ... オプションを使用してチェックします。
- ノードの実行順序をチェックします。ワイヤで接続されていないノードは、実行順序が決まっていません。ノードの空間的な配置によって実行順序が決定されることはありません。すなわち、テキスト言語の文とは異なり、接続されていないノードはダイアグラムの左から右、上から下の順には**実行されません**。
- 余分なVIがないかどうかをチェックします。関数とは異なり、配線されていないサブVIは（入力を必須または推奨入力として構成していない限り）必ずしもエラーを生成しません。誤ってブロックダイアグラム上に配線されていないサブVIを配置すると、ダイアグラムの実行時にそのサブVIも実行するため、プログラムは余分な動作を実行してしまう可能性があります。
- 隠れたVIがないかどうかチェックします。次の3つの場合には、意図に反してサブVIが隠れてしまう可能性があります。あるノードを別のノードの上に直接ドロップした場合、ストラクチャのサイズを縮小し

たためにサブVIが表示範囲からはずれてしまった場合、およびメインのダイアグラムの領域外にサブVIを配置した場合です。最後の場合では、ブロックダイアグラムを限界までスクロールします。さらに、VIで使用されているサブVIのインベントリを、**プロジェクト**メニューの**このVIのサブVI**オプションおよび**開かれていないサブVI**オプションと比較して、余分なVIが存在しないかチェックします。また、階層ウィンドウでVIのサブVIをチェックするか、またはエラーリストウィンドウで隠れたオブジェクトのリストをチェックします（**警告を表示**のチェックボックスが選択されている場合）。VIへの入力を必須として指定すると、隠れたVIによる間違った結果を避けることができます。詳しくは、「第3章 サブVIを使用する」の「サブVIに必要な接続、推奨される接続、オプションの接続」の項を参照してください。

警告メッセージについて

あるオブジェクトが別のオブジェクトによって完全に隠されているときは、すべてのオブジェクトが表示されていないことを示す警告メッセージが生成されます。たとえば、端子がストラクチャのエッジの下に隠れていたり、トンネルどうしが重なり合っていたりすると、エラーリストウィンドウに警告メッセージが表示されます。警告メッセージは、プログラム中の潜在的な問題をデバッグするための補助手段であり、VIの実行を妨げるものではありません。

不定データを検出する

計算エラー、あるいは結果が意味をなさないことを示すために浮動小数点デジタル表示部に表示される記号値は、2種類あります。**NaN**（＝数字でない）は、負の数の平方根を求める演算などで生成される可能性のある浮動小数点値を表す記号です。もう1つは**Inf**で、これは1を0で割る場合などに生成される特殊な浮動小数点値です。

不定データは、その後の動作をすべて破壊してしまうおそれがあります。浮動小数点演算は、明示的または暗黙に整数やブールに変換すると意味のない値になるNaNおよび±Infを伝播します。たとえば、1を0で割るとInfが生成されますが、この値をワード整数に変換すると、32,767という一見正常値のような値が生成されます。VIでこの種のエラーが発生しないという確信がない限り、整数タイプに変換する前に中間値の浮動小数点値の有効性をチェックする必要があります。

Forループを0回実行すると、予想外の値が生成される可能性があります。たとえば、初期化されたシフトレジスタの出力は初期値になります。しかし、シフトレジスタを初期化しなかった場合、出力は各データタイプのデフォルト値（0、FALSE、または空の文字列）か、またはダイアグラムを最後に実行したときにシフトレジスタにロードされた最後の値のいずれかになります。

配列の境界を超えて指標を作成すると、配列要素のデータタイプのデフォルト値が生成されます。このようなミスを犯しやすい例としては、While ループを使用して配列を最後の要素を超えて指標付けする、Index Array 関数の指標入力に大きすぎる値を渡す、あるいは Index Array 関数に空白の配列を渡す場合など、多くの場合があります。

特定の入力値に対して不定出力値を生成する可能性のある VI を設計する場合は、NaN や空白配列などの特殊値によって問題を特定しようとしないでください。代わりに、VI が問題を特定するメッセージを生成しているか、または定義されたデフォルトデータだけを生成しているかを確認する必要があります。

たとえば、受け取った配列を使用して For ループに自動的に指標を付ける VI を作成する場合は、入力配列が空のときに VI の動作をチェックする必要があります。出力エラーコードを生成するか、ループによって作成された値をあらかじめ定義された値に置き換えます。

☞ **注**　**ー無限大から＋無限大の範囲を持つ浮動小数点の表示器または制御器でも、NaN を受け取ると範囲エラーを生成する場合があります。**

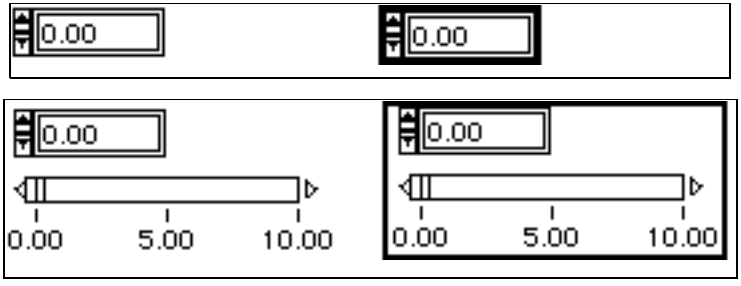
VI の範囲エラーを修正する

範囲エラーインジケータ

次の状況下では、**実行**ボタンの代わりに範囲エラーインジケータが表示されます。

- 範囲外の値を受け取ると実行を中止するように（制御器の**データ範囲...** メニュー項目を使用して）構成されたサブ VI の制御器が、範囲外の値を受け取ったとき。
- 呼び出し側の VI に範囲外の値を返そうとした場合に実行を中止するように構成されたサブ VI の表示器が、範囲外の値を返そうとしたとき。
- VI が実行していない状態でエラー時に実行を中止するよう設定した制御器に、オペレータが範囲外の値を入力したとき。

次の図の右側に示すように、範囲外の制御器または表示器は赤い太枠で囲って表示されます。



詳しくは、「第9章　数値制御器と数値表示器」の「数値制御器と数値表示器の範囲オプション」の項を参照してください。

☞ **注**　**VIおよびサブVIは、そのいずれかの表示器に実行を中断せずに範囲外の値を渡すことができます。エラー条件は、サブVIがこのような値を返そうとした場合にのみ発生します。また、オペレータはVIの実行中でもその制御器に範囲外の値を入力することができます。範囲外エラー条件は、VIが実行を開始するのを妨げるにすぎず、すでに開始しているVIを中止させるわけではありません。**

デバッグ機能

この項では、下記のデバッグ機能について説明します。

- VIをシングルステップで実行して個々の実行ステップを観察する
- 実行をハイライト表示し、発生するデータの流れを観察する
- プローブを使用してデータを表示する
- 実行を一時停止するためのブレークポイントを設定し、シングルステップの実行、プローブを使用したワイヤのデータのチェック、あるいは表示器の編集を行えるようにする
- 実行を保留して表示器を編集したり、実行回数を制御したり、あるいはVIの頭に戻る

VIをシングルステップで実行する


一時停止ボタン

デバッグを行うためには、ブロックダイアグラムの各ノードを1つずつ実行することが必要になる場合があります。このような実行方法はシングルステップの実行と呼ばれます。VIを実行する前に、**一時停止**ボタンを押してシングルステップの実行を開始させます。また、VIの実行は、**一時停止**ボタンを押していつでも停止することができます。**一時停止**ボタンを解除すれば、いつでも通常の実行に戻ることができます。VIを停止した状態で**実行**ボタンをクリックすると、実行は再開されますが、VIの**一時停止**はセットされたままになります。VIが再度呼び出されると、実行は一時停止します。


飛び越えるボタン

VIをシングルステップモードで実行しているときにアクティブな3つのステップボタンのいずれかを押すと、次のステップに進みます。次のステップをどこで実行するかは、どのステップボタンを押すかによって決まります。


中に入るボタン

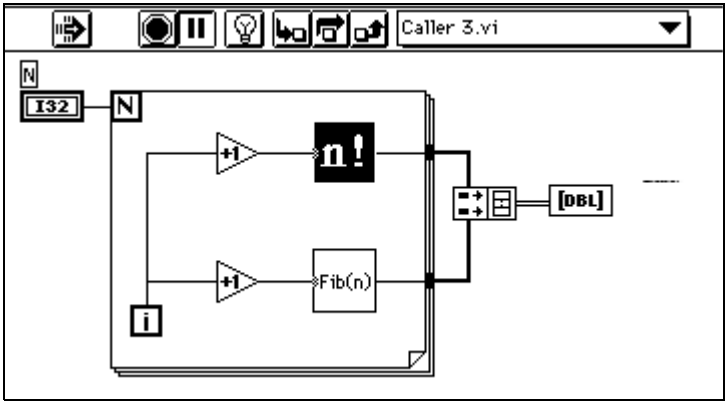
カーソルをいずれかのステップボタン上に合わせると、そのボタンを押したときの次のステップについて説明したヒントラベルが表示されます。

外に出るボタン

VIをシングルステップで実行する際には、実行のハイライト表示を使用してノード間のデータの流れを追跡することができます。詳しくは、本章の「実行をハイライト表示する」の項を参照してください。

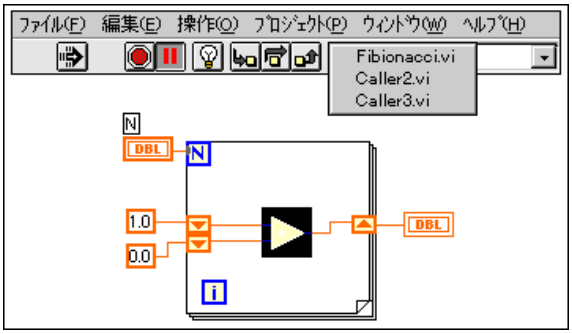
シングルステップでのVIの実行例

次の図は、シングルステップで実行しているVIの例を示したものです（Caller 1.vi）。このVIは、現在VI Caller 3.viのために実行しています。現在は、（**実行**ボタン上の矢印が示すように）サブVIのFibonacci.viが実行中です。次に実行されるノードは、Factorial.viというVIです。**中に入るボタン**をクリックすると、そのブロックダイアグラムが開き、シングルステップの実行がスタートします。**飛び越えるボタン**をクリックすると、サブVIを実行したのち、停止します。**外に出るボタン**をクリックすると、ループの現在のフレームの実行を終了したのち停止します。



サブVIのFibonacci.viに移行し、ループの中に入った後に、**外に出る**ボタンをクリックしてそのまま1秒間マウスボタンを押したままにしておくと、メニューが表示されます。このメニューで、VIをどこまで実行するかを選択します。ブロックダイアグラムを選択すると、**ブロックダイアグラム**のすべてのノードが実行されるまでVIを実行した後、停止します。

呼び出しスタック中のいずれかのVIを選択すると、そのVIを最後まで実行します。選択したVIの発呼者は、その時点で停止します。



ステップボタンを使用する

ステップボタンの機能は下記の通りです。


飛び越えるボタン

飛び越えるボタンを押すと、ストラクチャ（シーケンスやループなど）やサブVIを実行して、次のノードで停止します。キーボードショートカットを使用する場合は、<Ctrl>**（Windows）**、<command>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**を押しながら右向き矢印キーを押します。


中に入るボタン

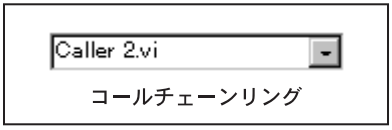
中に入るボタンを押すと、サブVIまたはストラクチャ（シーケンスやループなど）の最初のステップを実行したのち、そのサブVIまたはストラクチャ内の次のステップで停止します。キーボードショートカットを使用する場合は、<Ctrl>**（Windows）**、<command>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**を押しながら下向き矢印キーを押します。これらのステップボタンは、シングルステップモードのVIまたはサブVI内での実行に関してのみ有効です。シングルステップモードのVIにシングルステップモードのサブVIと通常モードのサブVIが1つずつ存在する場合は、最初のサブVIは呼び出し時にステップモードで実行しますが、2つめのサブVIは呼び出し時に通常のモードで実行します。現在のブロックダイアグラム、ストラクチャ、あるいはVIの実行を終了し、停止するためには、**外に出る**ボタンを押します。キーボードショートカットを使用する場合は、<Ctrl>**（Windows）**、<command>**（Macintosh）**、<meta>**（Sun）**、<Alt>**（HP-UX）**を押しながら上向き矢印キーを押します。


外に出るボタン

VIの実行が終了すると、ステップボタンはグレーの表示に変わります。

コールチェーンを読み込む

サブ VI が一時停止すると、**コールチェーン**リングが点滅します。このメニューには、最上層のVIからこのサブVIまでの発呼者のリストが表示されます。これは、現在実行中であるか否かに関係なくすべての呼び出し中のVIのリストを表示する**プロジェクト→このVIの発呼者**のメニュー項目とは異なる点に注意してください。コールチェーンリングから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 を呼び出している VI が選択されます。これは、ブロックダイアグラムに複数のインスタンスが存在する場合に、サブVIの現在のインスタンスを見分けるのに役立ちます。



コールチェーンリング

本章の「ステップボタンを使用する」の項で示したキーボードショートカットに加え、下記のコマンドも使用できます。

- サブVIやストラクチャをシングルステップで実行しているときは、**外に出る**ボタンをクリックしてマウスボタンを押したままにしておくと、メニューが表示されます。VI をどこまで実行して停止するかを選択します。
- ブロックダイアグラムの制御器、表示器、ローカル変数、グローバル変数をダブルクリックすると、それに対応するフロントパネルオブジェクトがハイライトで表示されます。
- <Ctrl>**（Windows）**、<option>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**を押しながらサブ VI をダブルクリックすると、そのサブVIのブロックダイアグラムが最前面に表示されます。

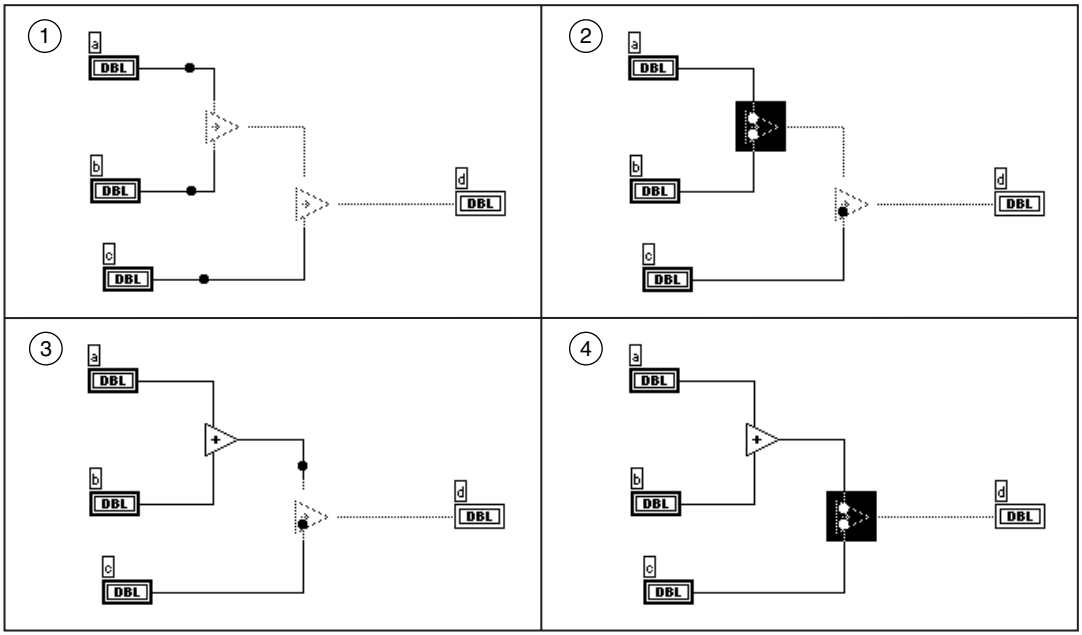
実行をハイライト表示する

デバッグを行うためには、VIのブロックダイアグラムの実行を動画で表示すると便利です。

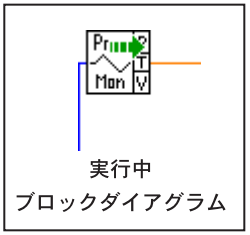

オフ　オン
実行のハイライトボタン

この機能を使用するためには、**実行のハイライト**ボタンをクリックします。ボタンの外観が変化します。このボタンをクリックすると、いつでも通常の表示モードに戻ることができます。実行のハイライト表示は、通常はノード間をデータがどのように流れるかを知るためにシングルステップモードと組み合わせて使用します。ハイライト表示は、VIのパフォーマンスを大幅に低下させます。

実行のハイライト表示を使用すると、ノードからノードへのデータの動きがワイヤ上をバブルが移動するように示されます。また、シングルステップの実行では、次の図のシーケンスで示すように次のノードが素早く点滅します。



実行のハイライトボタンをオンにしてサブVIをシングルステップで実行するときは、ブロックダイアグラムのサブVIアイコン上の実行グリフが、現在どのVIが実行中でどのVIが実行待ちになっているかを示します。次の図の矢印は、サブVIが実行中であることを示しています。



階層ウィンドウは、VIが一時停止または中断あるいはその両方の状態にあるかどうかを表示します（階層ウィンドウについての詳細は「第3章　サブVIを使用する」の「階層ウィンドウを使用する」の項を参照してください）。一時停止のグリフは、サブVIが一時停止または中断あるいはその両方の状態にあることを示します。緑（モノクロ表示では中が空洞の矢印）

の一時停止グリフは、このサブVIが呼び出されたときに一時停止することを示します。赤（モノクロ表示では中を塗りつぶしたグリフ）の一時停止グリフは、このサブVIが現在一時停止していることを示します。

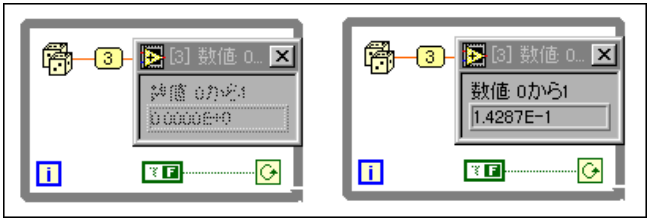
プローブツールを使用する

選択されているプローブツール

プローブツールを使用すると、実行して問題のある結果や予想外の結果を生成するVIの中間値をチェックすることができます。たとえば、一連の演算で構成された複雑なブロックダイアグラムでは、そのうちのいずれかの演算が不正なデータを返す可能性があります。

問題の原因を特定する1つの方法として、表示器をいずれかの演算の出力ワイヤに配線して中間値を表示させる方法があります。しかし、表示器をフロントパネル上に配置してその端子をブロックダイアグラムに配線する方法は、デバッグの方法としては適切ではありません。この方法は時間がかかる上に、フロントパネルやブロックダイアグラムに不要な項目を作成するため、あとで削除しなければなりません。

プローブツールを選択してそのカーソルをワイヤ上に移動するか、またはワイヤのポップアップメニューで**プローブ**を選択します。次の図で示すように、データを表示していない浮動プローブウィンドウが表示されます。VIを実行すると、ワイヤを通して渡されるデータが直ちにプローブウィンドウに表示されます。



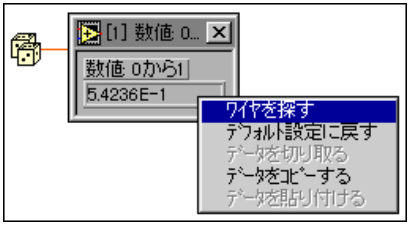
個々のプローブには自動的に固有の番号が付けられます。また、プローブを接続したワイヤにも同じ番号が付けられるため、どのプローブがどのワイヤに接続されているかが容易にみつけられます。ただし、フロントパネル制御器の名前の長さがプローブウィンドウと同じかそれより長い場合は、番号は表示されません。

プローブを実行のハイライト表示、シングルステップの実行、あるいはブレークポイントと組み合わせて使用すると、より簡単に値を確認することができます。シングルステップでの実行時、あるいは各ブレークポイントでの一時停止時には、データは使用可能であれば直ちに更新されます。ノードで実行が停止したときに入力を確認してみてください。

VIを実行する前にプローブを挿入してデータをチェックすることができます。シングルステップでの実行中に、実行したワイヤに対してプローブを作成すると、プローブが更新され、そのワイヤのデータが表示されます。ブレークポイントをセットしたノードの入力を調べたいときには、この方法を使用すると便利です。

プローブを使用してデータを変更することはできません。また、プローブがVIの実行に影響を与えることはありません。

次の図で示すように、プローブウィンドウの表示器上でポップアップして**ワイヤを探す**を選択すると、関連付けられたワイヤを見つけることができます。



ワイヤを探すオプションを選択すると、該当するワイヤを含むブロックダイアグラムがすべてのVIの一番手前に表示され、ワイヤがハイライト表示されます。

プローブウィンドウに表示された値をデフォルト値にリセットするには、**デフォルト設定に戻す**を選択します。

データを同じVIまたは別のVIの数値制御器にコピーするには、**データをコピーする**を選択します。

プローブを作成する

プローブを作成する際には、ワイヤのデータタイプと一致するデフォルトスタイルのプローブが作成されます。たとえば、データタイプが数値のときには、デジタル表示器がワイヤのデータを読み取ります。

プローブの制御器は、内蔵制御器から選択したり、カスタム制御器またはType Defとして保存された制御器から選択することもできます。それには、ワイヤのポップアップメニューで**カスタムプローブ**を選択したのち、右側

に表示される**カスタムプローブ**パレットから制御器を選択します。**制御器を選択...**を選択した場合は、ファイルダイアログボックスを使用してファイルシステムに保存されたカスタム制御器またはType Defを選択します。Type Defは、プローブのデータに対して使用した場合は標準のカスタム制御器として扱われます。

カスタムプローブパレットでは、ワイヤのデータタイプと一致する部品のみが使用可能になります。ワイヤと同じデータタイプを使用できない制御器を選択すると、データタイプの不一致を示す警告音が鳴り、プローブは作成されません。たとえば、配列やクラスタは完全なデータタイプではないため、配列やクラスタに対して**配列とクラスタ**パレットの配列シェルやクラスタシェルを使用することはできません。

カスタムプローブパレットには、**制御器**パレットにパレットを追加する場合と同じように独自のパレットを追加することができます。追加方法についての詳細は、「第7章 環境をカスタマイズする」を参照してください。

ブレークポイントツールを設定する

VIやノード、あるいはワイヤには、実行を一時中断するためのブレークポイントを設定することができます。ノードには、サブVI、関数、ストラクチャ、コードインタフェースノード（CIN）、フォーミュラノード、および属性ノードが含まれます。ブロックダイアグラムにブレークポイントを設定すると、ダイアグラムのすべてのノードを実行した後で実行が一時中断されます。ワイヤにブレークポイントを設定すると、データがそのノードを通過した後で実行が一時中断されます。

実行中にブレークポイントに達したら、ステップボタンを使用してシングルステップで先に進む、プローブでワイヤのデータをチェックする、フロントパネル制御器の値を変更する、次のブレークポイントまで進む、あるいは実行を最後まで終了させる、といった操作を行います。**実行**ボタンをクリックすると、簡単に次のブレークポイント、あるいはサブVIが呼び出されるポイントまで進むことができます。

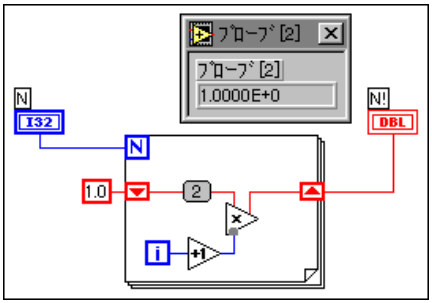
ブレークポイントを設定するためには、**ツール**パレットでブレークポイントツールを選択したのち、ブロックダイアグラム、ノード、またはワイヤをクリックします。同じオブジェクト上で再度このツールをクリックすると、いつでもブレークポイントを解除することができます。左に示すように、ブレークポイントがセットされているか解除されているかは、ツールの外観によって示されます。

次の表は、ブレークポイントの表示方法および実行の停止位置をブレークポイントの設定位置別に示したものです。

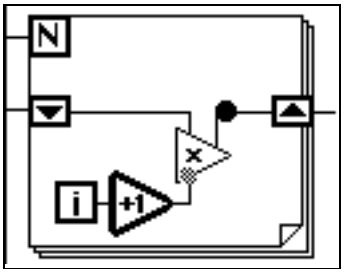
表 4-2 ブレークポイントの設定

ブレークポイントの位置	ブレークポイントのハイライト表示の方法	停止位置
ブロックダイアグラム	ブロックダイアグラムの周りに赤い枠を表示。ダイアグラムがストラクチャ内にある場合は赤い枠も同じくストラクチャ内に表示。	ブロックダイアグラムのすべてのノードの実行が終了したあと。ループの場合はループの各回の実行後。
ノード	ノードを囲む赤い枠で表示。	ノードを実行する前。この位置で停止した場合はプローブツールを使用してノードへのすべての入力信号をチェックすることが可能。
ワイヤ	ワイヤ中央の赤の丸い点。ワイヤにプローブを接続した場合はプローブの周囲に赤い枠を表示。	ワイヤをデータが通過したあと。

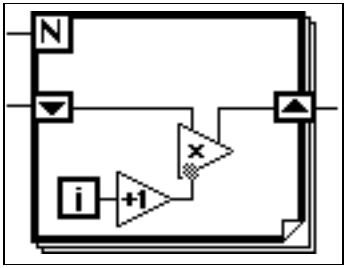
次の例では、プローブを接続したワイヤにブレークポイントが設定されています。VIは、プローブがデータを表示した後に一時停止します。



次の例では、回数端子ノードにブレークポイントが設定されています。VIは回数端子を実行する前に停止します。また、ワイヤにもブレークポイントが設定されています。VIは、Multiplyを実行した後に停止します。



次の例では、For ループにブレークポイントが設定されています。VIは Multiply を実行した後に停止します。



ブレークポイントによってVIが一時停止すると、そのダイアグラムが前面に表示され（フロントパネルが閉じられているときはフロントパネルも開きます）、停止位置のノードまたはワイヤがマーキーでハイライト表示されます。

ブレークポイントはVIとともに保存されますが、実行時にのみアクティブになります。

実行を中断する

サブVIの実行を中断することにより、制御器や表示器を編集したり、発呼者に戻る前に必要な回数だけサブVIを実行したり、サブVIの先頭に戻ったりすることができます。

サブVIを中断モードに設定するには、サブVIを開いて**操作→呼び出されたら中断**オプションを選択します。あるいは、編集モードでフロントパネルのコネクタペーンをポップアップして、**VI設定→実行オプション→呼び出されたら中断**...を選択することもできます。サブVIは、呼び出された時点で自動的に中断します。シングルステップの実行時に呼び出されたら中断...オプションを選択した場合は、VIはすぐには中断モードに切り替わりません。

サブVIに対する特定の呼び出しの際に実行を中断したいときは、**呼び出されたら中断**...ではなくサブVIノードのポップアップメニューの**サブVIノード設定**...を使用します。**サブVIノード設定**...は、サブVIの特定の呼び出し時（特定のインスタンス）にのみ実行を中断させます。

自動中断を認識する

実行中に制御器または表示器が範囲を超えるとVIは自動的に中断し、範囲エラーの原因となった端子またはローカル変数が選択されます。

範囲エラーで停止するように設定された制御器または表示器を含むサブVIには、条件付きブレークポイントが存在します。範囲エラーが発生すると、サブVIはブレークポイントに達した場合と同様に停止します。範囲エラーが発生しなかった場合は、サブVIは正常に実行されます。

サブVIに渡された値によって範囲エラーが発生した場合は、そのサブVIのフロントパネルが開かれるかまたは前面に表示され、サブVIは中断モードのままになります。この時点では、サブVIの制御器の値が発呼者であるVIから渡される入力値になりますが、必要な場合はこれらの値を変更することができます。実際には、範囲エラー表示器が点灯したときは、サブVIを実行するためにはこれらの値を変更する必要があります。サブVIの表示器は、デフォルト値、またはサブVIを最後に実行したとき（フロントパネルが開かれていたとき）の値のどちらかを表示します。

サブVIの中断時にツールバーボタンを使用する


実行ボタン

現在のサブVIを実行してから発呼者に戻りたいときは、中断モードで**実行**ボタンを押します（または、**操作→実行**コマンドを選択します）。必要な回数だけ繰り返し実行します。

サブVIが終了すると、表示器にサブVIのその回の実行結果が表示されます。ただし、呼び出し側のVIに異なる値を返したいときは、表示器の値を変更することができます。実際、表示器の値を設定できるのは中断モードのときだけです。サブVIの表示器の値を呼び出し側のVIに返す準備ができたら、**発呼者へ戻る**ボタンをクリックします。

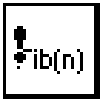

最初へ戻るボタン

VIの先頭に戻りたいときは、**最初へ戻る**ボタンをクリックします。


発呼者へ戻るボタン

中断したサブVIを実行していない間は、**発呼者へ戻る**ボタンが表示されます。呼び出し側のVIに戻る場合は、このボタンをクリックします。ただし、現在のVIを実行せずに発呼者に戻ることも可能です。現在のVIを実行したいときは、かならず**実行**ボタンをクリックしてから発呼者に戻るようにしてください。

中断時に階層ウィンドウを呼び出す



階層ウィンドウには、中断したサブVIを示す感嘆符グリフが表示されます。左の図は、**呼び出されたら中断**がオンになっているときの階層ウィンドウ内のサブVIを示したものです。

デバッグ機能を無効にする

さまざまなデバッグ機能を無効にした状態でVIをコンパイルすると、メモリの消費量を減らしてパフォーマンスを向上させることができます。デバッグ機能を無効にするためには、フロントパネルのコネクタペーンのポップアップメニューで**VI設定→ウィンドウオプション**を選択し、**デバッグを許可する**のチェックボックスの選択を解除します。

ダイアグラムの一部をコメントアウトする

ときには、ブロックダイアグラムの一部を無効にした状態でVIを実行することが必要になる場合があります。それには、定数ブールをセレクタに配線して、無効にしたい部分をCaseストラクチャの中に入れるのが最も簡単な方法です。無効にしたいダイアグラムを、実行されないCaseストラクチャのフレームの中に格納します。

Caseストラクチャが何らかのデータ値を生成する場合は、出力トンネルの定数を作成する必要があります。

5

VIの印刷および文書作成

この章では、VIの印刷および文書作成について説明します。

印刷

GでVIを印刷するには主に4つの方法があります。

- 現在のウィンドウの内容を素早く印刷したいときは、**ウィンドウの印刷...**オプションを使用します。
- フロントパネル、ブロックダイアグラム、サブVI、制御器、VIの履歴などに関する情報も含めてVIのデータを包括的に印刷したいときは、**文書を印刷...**オプションを選択します。
- アプリケーションの制御によってVIのフロントパネルを自動的に印刷したいときは、プログラム印刷を使用します。
- VIウィンドウあるいはVI文書をプログラム印刷で印刷したいときは、VIサーバを使用します。

また、プリンタがシリアルポートまたはパラレルポートに接続されているときは、Serial Port VIsを使用してテキストをプリンタに転送することができます。通常、この方法を使用するためにはプリンタのコマンド言語に関してある程度の知識が必要になります。

WindowsおよびUNIXでは、**関数→通信**のSystem Exec VIオプションを使用してテキストを印刷することもできます。Macintoshでは、**関数→通信→AppleEvent**のAESend Print Document VIオプションを使用します。Windowsでは、他のアプリケーションにデータを印刷するよう命令するためにはActiveX automationを使用します。

印刷の設定

どの印刷方法を使用する場合でも、すべての印刷物の形態は環境設定とSetupという2つのダイアログボックスの設定によって決定されます。

印刷オプションは、環境設定ダイアログボックスの印刷のページを使用して設定します。

プリンタ固有の情報やフォーマットは、**ファイル**メニューの**プリンタ設定 ...**（Macintoshでは**ページ設定**）オプションを使用して設定します。ほとんどのプリンタでは、用紙の向き（横か縦）などの設定はこのダイアログボックスで行います。また、多くのプリンタは、代替書体や用紙サイズといったプリンタ固有の設定を行うための機能を備えています。**プリンタ設定 ...**（Macintoshでは**ページ設定**）の設定は、VIとともに保存されます。

PostScript印刷

PostScriptプリンタを使用できる場合は、他の出力オプションを使用する代わりに**編集→環境設定→印刷→PostScript印刷**オプションを選択することができます。

PostScriptプリンタには次のようなメリットがあります。

- PostScript印刷では画面の画像をより正確に再生できます。
- PostScript印刷では解像度の高いグラフを印刷できます。
- PostScript印刷ではパターン、ラインのスタイル、および書体をより正確に再生できます。

PostScript印刷を使用するかどうかは、環境設定ダイアログボックスで指定します。環境設定ダイアログボックスについての詳細は、「第7章　環境をカスタマイズする」を参照してください。

アクティブウィンドウを印刷する

現在アクティブなウィンドウの内容（フロントパネルまたはブロックダイアグラム）を印刷する場合は、メニューの**ウィンドウの印刷 ...**項目を使用します。このオプションを使用すると、最小限のプロンプトで素早く印刷することができます。VIのより包括的なデータを印刷したいときは、**文書を印刷 ...**を使用します。

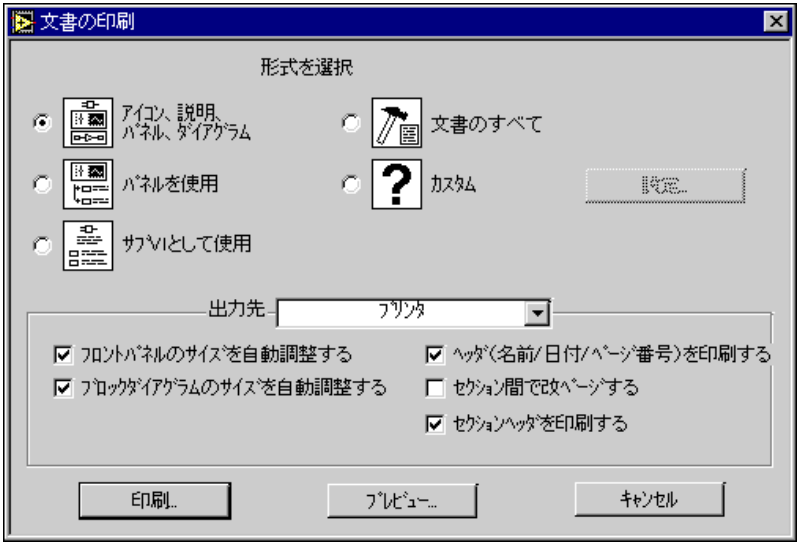
ウィンドウの印刷 ...は、VIをプログラム印刷する場合と同じフォーマットで印刷します。VI設定ダイアログボックスの**実行オプション**の項には、プログラム印刷または**ウィンドウの印刷...**オプションを使用して印刷するVIのフォーマットをより細かく制御するためのいくつかのオプションがあります。これらのオプションについての詳細は、本章の「ページレイアウトを設定する」の項を参照してください。

VIを文書化する

この項では、用紙やRTFあるいはHTMLファイルへのVI文書の印刷、および画像ファイルのフォーマットに関することがらについて説明します。

文書を印刷する

VIの詳細な内容を印刷したいときは、次の図で示した**文書を印刷...**ダイアログボックスを使用します。このダイアログボックスで、VIの数種類のフォーマットのなかからフォーマットを選択します。また、独自のフォーマットを作成することもできます。



印刷フォーマットを設定する

文書の印刷ダイアログボックスには、5種類の印刷フォーマットのほかに、独自のフォーマットを作成するためのカスタムオプションがあります。これらのフォーマットの説明の左側には、それぞれのフォーマットのアイコンが表示されます。

アイコン、説明、パネル、および**ダイアグラム**フォーマット（デフォルト設定）は、アイコン、VIの説明、フロントパネル、およびブロックダイアグラムを印刷します。

パネルを使うフォーマットは、フロントパネル、VIの説明、制御器名およびそれらの説明を印刷します。

サブVIとして使うフォーマットは、アイコン、コネクタ、VIの説明、端子（データタイプ）、名前、および接続されている制御器の説明を印刷します。このフォーマットは、Gの関数リファレンスフォーマットによく似ています。

文書のすべてフォーマットは、アイコン、コネクタ、説明、フロントパネル、すべてのフロントパネル制御器に関する情報、ブロックダイアグラム、およびサブVIの名前のリストを含むすべてのデータを印刷します。

カスタムフォーマットは、現在のカスタム設定を使用して印刷します。カスタム設定を変更するには、カスタムオプションを選択したあとで**設定...**を選択します。このダイアログボックスについての詳細は、本章の「カスタム印刷設定を作成する」の項を参照してください。

用途の選択ボックスでは、プリンタ、HTMLファイル、RTFファイル、または標準のテキストファイルのどれに出力するかを選択します。**用途**機能については、本章の「制御器およびVIの説明をRTFまたはHTMLファイルに印刷／エクスポートする」の項を参照してください。

その他の印刷オプションを設定する

文書の印刷ダイアログボックスには、プリントアウトのスケール、改ページ、ヘッダを制御するためのいくつかのページレイアウトオプションがあります。以下では、これらのオプションについて説明します。

フロントパネルのサイズを自動調整すると**ブロックダイアグラムのサイズを自動調整する** — これらのオプションを使用すると、フロントパネルやブロックダイアグラムをページ内にうまくおさまるように元のサイズの4分の1に縮小することができます。

ヘッダを印刷する — 各ページの上の欄にヘッダを印刷します。このヘッダには、ページ番号、VIの名前、および最後にVIを修正した日付が記入されます。

セクション間で改ページする — 下記のセクション間に改ページを挿入します。

- コネクタアイコンと説明
- フロントパネル
- フロントパネル制御器の詳細リスト
- ブロックダイアグラム
- ブロックダイアグラムの詳細
- VIの階層
- サブVIのリスト

セクションヘッダを印刷する — 各セクションごとにヘッダ（たとえば、サブVI情報の前にはサブVIのリストというヘッダ）を印刷します。**文書の印刷**を使用して印刷する場合は、各フォーマットに対してこの機能が自動的に設定されます。

カスタム印刷設定を作成する

カスタム印刷設定ダイアログボックスを呼び出すためには、文書の印刷ダイアログボックスで**カスタム**ボタンを選択します。**カスタム**を選択すると**設定**ボタンがアクティブになり、設定ボタンをクリックすると次のような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には、印刷できる項目のカテゴリーが表示されます。これらのカテゴリーは、印刷される順番通りに表示されます。また、このダイアログボックスのページレイアウトオプションは、プリントアウトのスケール、改ページ、およびヘッダを制御します。

カスタム印刷設定の各オプションのアイコンは、説明の左側に表示されます。

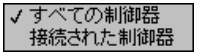
アイコンと説明

VIコネクタとアイコン — VIのアイコンの絵とその入力および出力を印刷します。

VI説明 — VIの説明を印刷します。

フロントパネル — フロントパネルを印刷します。

制御器 — 制御器および表示器の名前のリストを印刷します。配列、クラスタ、またはrefnumに遭遇すると、サブ制御器が印刷されます。ポップアップメニューから下記のオプションを選択します。



説明 — 制御器の名前の横に説明を印刷します。

データタイプ情報を含める — プリンタに出力する場合は、制御器の名前の左に各制御器の端子が印刷されます。テキストファイルに保存する場合は、データタイプは説明の後に印刷されます。

ブロックダイアグラム — ブロックダイアグラムを印刷します。

非表示フレーム — Caseストラクチャおよびシーケンスストラクチャの見えないフレームを印刷します。

上位のフレームを繰り返す — 見えないフレームを印刷する際に、見えるフレームを再度順番に印刷します。

VI階層 — メモリに入っている現在のVIの階層、およびVIとサブVIの間の接続を示す線を印刷します。現在のVIはボックスでハイライト表示されます。

サブVIのリスト — 使用されるすべてのサブVIのアイコン、名前、およびパスを印刷します。

VI履歴 — 現在のVIに履歴情報がある場合にその履歴情報を印刷します。

プログラム印刷

ウィンドウの印刷や文書の印刷のダイアログボックスを使用せずにVIで制御されているデータを印刷する方法としては、2通りの方法があります。

- プログラム印刷を使用すると、VIの実行が終了するたびに自動的にそのVIのパネルを印刷することができます。
- VIサーバを使用すると、任意のVIのパネルまたは文書を印刷することができます。

プログラム印刷を行うには、**操作→VI終了後に印刷**を選択します。

このオプションが選択されていると、VIの実行が終了するたびにそのVIのフロントパネルの内容が印刷されます。VIがサブVIの場合は、サブVIの実行が終了した時点でパネルを印刷し、そのあとで発呼者に戻ります。

印刷スケジュールを管理する

ときには、毎回VIに印刷を実行させる必要がない場合があります。たとえば、ユーザがボタンを押したときにのみ、あるいは何らかの条件（テストエラーなど）が発生したときにのみ印刷を実行させたい場合です。また、印刷のフォーマットをより細かく調整したい場合もあります。さらに、制御器の1つのサブセット、あるいは1つの特定の制御器だけを印刷したい場合もあります。

用途に合わせてフォーマットしたパネルを持つサブVIを作成することができます。VIで**操作→VI終了後に印刷**を選択する代わりに、サブVIから同じオプションを選択します。印刷を実行したいときは、そのサブVIを呼び出し、印刷したいデータを渡すことができます。

この方法を使用すると、プリントアウトの外観を完全に制御でき、Gの制御器やツールを使用してごく簡単なプリントアウトや非常に複雑なプリントアウトを作成することができます。

印刷するタイミングを制御できるもう1つの方法として、VIサーバによる印刷を使用する方法があります。これらのオプションを使用すると、いつでもVIを印刷することができます。詳しくは、「第21章　VIサーバ」を参照してください。

印刷の補助機能

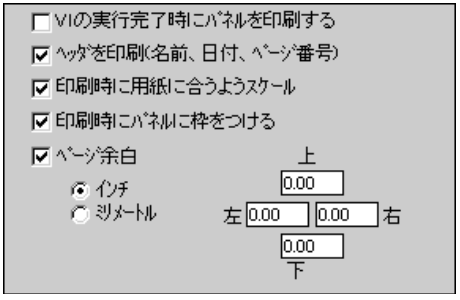
透明やカラーツールを使用すると、制御器で印刷したくない部分を印刷しないようにすることができます。たとえば、透明を使用して制御器のエッジを単純な形にすることができます。

プリントアウトの一部を強調したいときは、**装飾体**パレットの図形オブジェクトを使用します。たとえば、制御器の特定の部分をボックスで囲むことにより、その部分をパネルの他の部分と区別することができます。

また、ビットマップ画像を使用してプリントアウトをカスタマイズし、会社のロゴなどの要素をレポートに追加することもできます。

ページレイアウトを設定する

編集→環境設定および**プリンタ設定**（MacintoshやUNIXでは**ページ設定**）の設定により、プリントアウトの外観が違ってきます。プリントアウトの外観をかえるレイアウトオプションのオン／オフの切り替えには、**VI設定→実行**オプションを使用します。次に示すこれらのページレイアウトオプションは、**ウィンドウの印刷...**の動作にも影響します。



メニューの**VIの実行完了時にパネルを印刷する**の項目は、プログラム印刷をオンにするためのもう1つの手段です。

ヘッダを印刷を選択すると、VIの名前、最後に修正した日付、およびページ番号を含むヘッダが各ページの上部に印刷されます。

印刷時にページに合うようスケールを選択すると、パネルが1つのページよりも大きい場合にはパネルが最小限の数のページに収まるように、プリントアウトが元のサイズの4分の1の大きさに縮小されます。

印刷時にパネルに枠をつけるを選択すると、パネルの周りに枠が印刷されます。

ページ余白を選択すると、プリントアウトにインチまたはミリメートル単位で絶対余白を設定することができます。上下、左右の余白は、それぞれ個別に指定することができます。余白の幅は、プリンタの物理パラメータによって制限されます。プリンタの許容値よりも小さい余白を設定した場合は、プリンタの最小許容余白が実際の余白になります。

その他の印刷方法を使用する

ときには、上記の印刷方法が作成したアプリケーションに適さない場合があります。こうした場合にも、さまざまな印刷条件に対応できる印刷方法があります。

たとえば、上記の印刷方法はページの全データの印刷を目的としたものですが、アプリケーションによっては行単位でデータを印刷するほうが望ましい場合もあります。行単位で印刷するプリンタがシリアルポートまたはパラレルポートに接続されている場合は、シリアルポートVIを使用してプリンタにテキストを送ります。通常、この方法を使用するためにはプリンタのコマンド言語に関してある程度の知識が必要になりますが、この方法はGユーザが作成した多くのアプリケーションで問題なく使用できます。

思い通りの印刷結果が得られない場合は、データをファイルに保存したあと、ほかのアプリケーションから印刷することを検討した方がよいかもしれません。

他の印刷方法についての詳細は、「付録B　Gに関する一般的な質問」の「文字列を印刷するには？」の項を参照してください。

制御器およびVIの説明をRTFまたはHTMLファイルに印刷／エクスポートする

VIの説明や制御器は、リッチテキストフォーマット（Rich Text Format: RTF）やハイパーテキストマークアップ言語（Hyper Text Markup Language: HTML）フォーマットのファイルに印刷またはエクスポートすることができます。これは、ほとんどの文書作成ソフトウェアへのRTFファイルのインポートを可能にする便利な手段です。RTFファイルは、オンラインヘルプファイルのソースファイルでもあります。HTMLフォーマットは、とくにWWW（World Wide Web）への出力を目的としたオンラインマニュアルに使用します。

文書をRTFファイルに印刷する際には、オンラインヘルプファイル（オンラインリファレンス）に適したファイルを作成するのか、またはワープロ作業に適したファイルを作成するのかを選択することができます。ワープロフォーマットでは、画像は文書中に書き込まれます。文書をオンラインヘルプファイル用にRTFフォーマットで印刷すると、画像は外部のビットマップファイル（.BMP）に保存されます。HTMLファイルでは、すべての画像はJPEGファイル（.JPG）またはポータブルネットワーク画像ファイル（.PNG）のフォーマットで保存されます。

文書をRTFまたはHTMLフォーマットで印刷するには、**ファイル→文書の印刷**を選択します。次のようなダイアログボックスが表示されます。

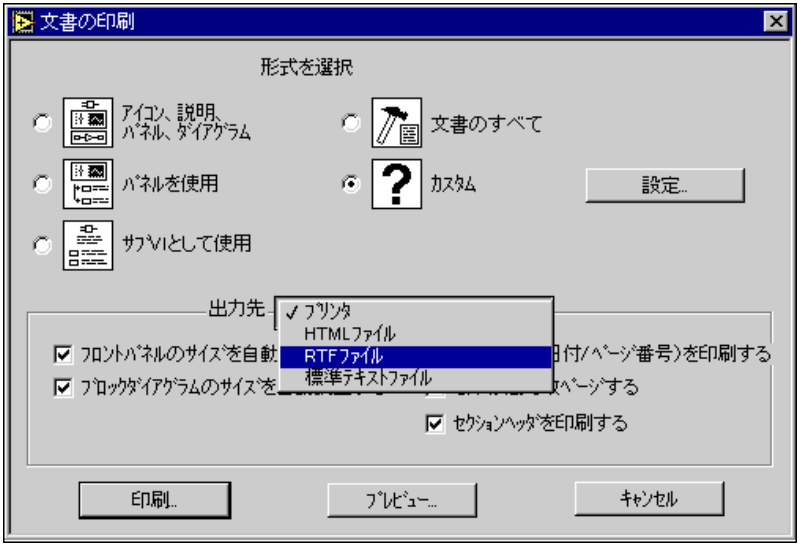


図 5-1 文書の印刷ダイアログボックス

用途ではRTFファイルまたはHTMLファイルを選択します。

RTFファイルを選択した場合は、次に示すようにダイアログボックスに新たなオプションが表示されます。画像を外部ファイルに保存する**ヘルプコンパイラソース（外部で作成された画像）**を選択して、ファイルをオンラインヘルプのフォーマットで保存します。**用途**と色の**奥行き**を選択します。

図 5-2 文書の印刷ダイアログボックス、RTF ファイル

制御器および VI の文書に対して HTML ファイルを選択した場合は、次のようなダイアログボックスが表示されます。

図 5-3 文書の印刷ダイアログボックス、HTML ファイル

ここでは、画像ファイルのフォーマットと色の深度を選択することができます。JPEG フォーマットを使用するとデータをかなり圧縮できますが、画像の細部が一部失われるおそれがあります。

このフォーマットは写真に最も適しています。線画、パネル、およびダイアグラムに使用すると、JPEGの圧縮によって画像にぶれと色むらが生じます。JPEGでは、画像は常に24ビットです。JPEGでこれより低い色の深度（たとえば白黒）を選択した場合、画像は指定した深度で補足されますが、24ビット画像であることに変わりはありません。

PNGフォーマットにも圧縮機能がありますが、圧縮率はかならずしもJPEGと同じレベルになるとは限りません。ただし、PNGでは圧縮によって画像の細部が失われることはありません。また、PNGは1ビット、4ビット、8ビット、および24ビット画像をサポートしています。低いビット深度では、得られる画像の圧縮率はJPEGよりも高くなります。PNGフォーマットは、GIFフォーマットに代わるものとして開発されたものです。JPEGやGIFよりも優れた点が多数ありますが、JPEGやGIFほど一般的にサポートされてはいません。

HTMLファイルの画像をGIFフォーマットで保存したい場合は、シェアウェア画像フォーマットコンバータを使用して画像を変換することができます。その場合、正しい拡張子を付けてHTMLをGIF画像を参照するように変更します。この方法でGIF画像を生成する場合、オリジナル画像がそのまま再生されるPNG画像から変換をスタートし、そのうえでGIFに変換するようにしてください。GIFは、現在は使用権の問題があるため直接サポートされてはいませんが、将来は直接サポートされるようになる可能性があります。

VIの文書を保存すると、文書の名前を入力するためのダイアログボックスが表示されます。RTFまたはHTMLで保存する文書のデフォルトのファイル名は、かならずVIの名前と`.rtf`または`.html`という拡張子で構成されます（例：`AI Clear.rtf`）。（Windows 3.1プラットフォームで保存したHTMLの名前は、VI名と`.htm`という拡張子で構成されます）

次の表は、VIの各パラメータとそれに対応するJPEGファイルの名前の例を示したものです。Testという名前のVIを保存する際にJPEGの画像フォーマットを選択すると、下記のJPEGファイルが作成されます。

表 5-1 作成されるJPEGファイル

VIのパラメータ	JPEGファイルの名前
VIのアイコン	`testi.jpg`
VIのコネクタ	`testc.jpg`
VIのフロントパネル	`testp.jpg`
VIのブロックダイアグラム	`testd.jpg`
VIの階層ウィンドウ	`testh.jpg`
サブVIのアイコン	*`name of subVI`* `+ i.jpg`

制御器および表示器の端子の画像は、端子に対応する名前を持つ画像ファイルに保存されます。したがって、VIに同じタイプの入力が複数存在する場合は、そのタイプの端子ごとに1つの画像ファイルが作成されます。たとえば、VIにint32という入力が3つある場合は、`ci32.jpg`という1つのファイルが作成されます。

それぞれの端子の名前は次のようになります。最初の文字は、端子が制御器のためのものであるか表示器のためのものであるかにより、`c`または`i`となります。端子が配列を表す場合は、最初の文字のあとに端子の次元を表すコード（例：`2d`）が続きます。さらに、データのタイプを表す省略コードが続きます。次に、画像ファイルの名前のサンプルを示します。

表5-2 画像ファイルの命名例

名　前	端　子
`ci32.jpg`	制御器、int32
`i2di32.jpg`	表示器、2d array of int 32s
`c3dstr.jpg`	制御器、3d array of strings

プログラムを使用したRTFファイルおよびHTMLファイルの作成

プログラムを使用したRTFファイルおよびHTMLファイルの作成についての詳細は、「第21章　VIサーバ」を参照してください。ここでは、複数のVIの文字列のエクスポートおよびインポートについても記載されています。

ローカル化について

VIのフロントパネル上の文字列をローカル化することにより、VIのプリントアウトもローカル化することができます。VIの文字列のローカル化についての詳細は、「第29章　移植性およびローカル化について」の「VIのローカル化」の項を参照してください。

独自のヘルプファイルを作成する

正しい開発ツールを使用すると、独自のオンラインヘルプあるいはオンラインリファレンスの文書を作成することができます。ヘルプ文書にはフォーマット化したテキスト文書を使用し、これらの文書にトピックを入力することでVIへリンクさせることができます。

ソースファイルは、プラットフォームに関係なくすべてWindowsのヘルプフォーマットでなければなりません。ヘルプ文書は複数のプラットフォーム用に対して作成することができます。

ソース文書を作成する際には、ヘルプコンパイラを使用してヘルプ文書を作成します。複数のプラットフォームのヘルプファイルを作成する際には、かならずそのヘルプファイルを使用するプラットフォーム専用のヘルプコンパイラを使用する必要があります。下記のコンパイラはどれでも使用することができます。Windows用のコンパイラには、ヘルプ文書を作成するためのツールも含まれています。

- **（Windows）**エクセルソフト株式会社のXLsoft。電話：03–5440–7876
- **（Windows）**WexTec Systems, Inc.社のDoc-To-Help。電話：（米国）800 939 8324
- **（Macintosh）**Altura Software社のQuickView。電話：（米国）408 655 8005
- **（UNIX）**Bristol Technologies社のHyperHelp。電話：（米国）203 438 6969

ヘルプファイルを作成しコンパイルしたら、それらのファイルを直接VIにリンクすることができます。ヘルプファイルにリンクしたいVIのVIコネクタペーンのポップアップメニューで、**VI設定→文書作成**を選択します。**ヘルプタグ**ボックスを選択し、ヘルプファイルにトピックリンクを入力します。**参照...**ボタンをクリックしてヘルプファイルを選択すると、**ヘルプのパス**ボックスにファイルのパスが表示されます。

6

VIおよびサブVIをセットアップする

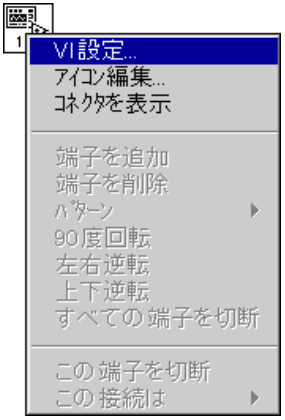
この章では、**VI設定**および**サブVIノード設定**のダイアログボックスを使用してVIの動作をカスタマイズする方法について説明します。

ポップアップパネルを作成する

ときには、1つのフロントパネルではさまざまなオプションや表示を表示しきれない場合があります。そのような問題を解決するためには、最上層のVIで高レベルのオプションが表示され、サブVIで関連オプションが表示されるようにVIを構成します。

Gは、サブVIを呼び出す際、そのフロントパネルを開かずにサブVIを実行します。サブVIが呼び出されたときにそのフロントパネルを開き、サブVIの実行が終了したらフロントパネルを閉じるようにするには、**VI設定**または**サブVIノード設定**のダイアログボックスを使用します。

VI設定ダイアログボックスを呼び出すには、次に示すようにフロントパネルの右上のVIアイコンをポップアップし、**VI設定...**を選択します。この操作は編集モードで行う必要があります。



このダイアログボックスには、VIの外観や動作をカスタマイズするためのいくつかのオプションが含まれています。ダイアログボックスの一番上にあるリングを使用して、**実行オプション**、**ウィンドウオプション**、**文書作成**という異なるオプションのカテゴリーのなかからいずれかを選択します。

VI設定のオプション

この項では、VIの実行、ビジュアルコンテキスト、文書作成、およびメニュー項目のカスタマイズの各オプションについて説明します。

実行オプション

VI設定ダイアログボックスには、最初は現在のVIの実行オプションが表示されます。**実行オプション**は、次の図に示す通りです。

最上位のVIがロードされた直後にサブVIのパネルを開くようにしたいときは、**ロードされたらフロントパネルを表示する**オプションをオンにします。

このページのオプションのなかで最も便利な2つのオプションは、**呼び出されたらフロントパネルを表示する**と**元に閉じてあったら閉じる**です。サブVIに対してこの2つのオプションをオンにしておくと、そのサブVIが呼び出されるとそのサブVIのフロントパネルが自動的に開き、元々閉じていた場合には自動的に閉じます。

開かれたら実行するオプションは、VIを開いたときに自動的に実行するように設定します。

呼び出されたら中断するオプションは、**操作→呼び出されたら中断する**を選択するのと同じです。このデバッグオプションについては、「第4章　VIとサブVIの実行およびデバッグ」の「デバッグ機能」の項で説明しています。

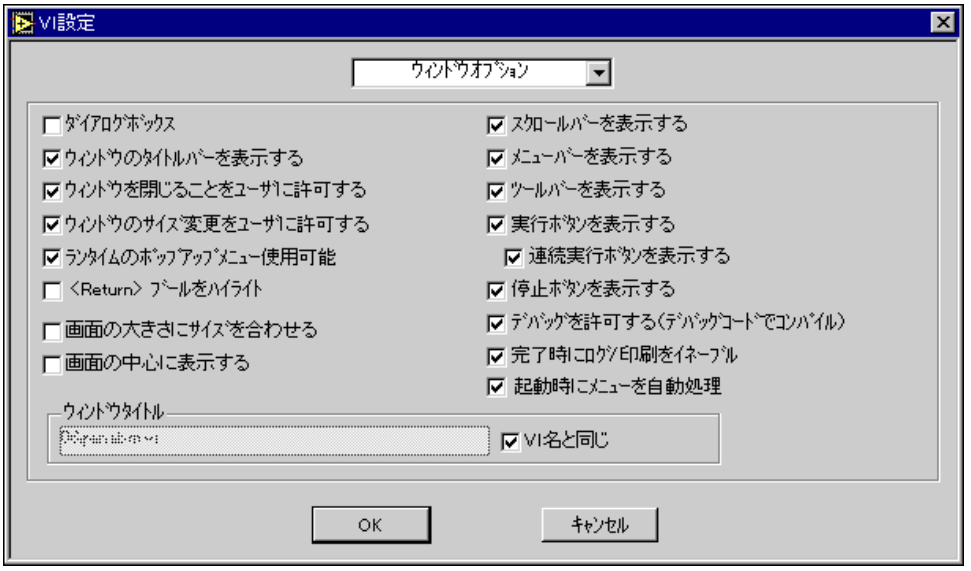
再入実行、**優先順位**、および**優先実行システム**は、VIの実行方法に関する上級機能です。これらの機能が必要になるのは、特殊なアプリケーションに限られます。これらのオプションについては、それぞれ「第26章 Gの実行システムについて」の「再入実行」と「VI設定ダイアログボックスでの優先順位の割り当て」の項で説明しています。もう1つの上級機能として、**マルチスレッド処理**があります。マルチスレッド処理では、複数のVIを同時に実行している最中でもそれらのVIがマウスやキーボードからの入力に応答でき、また、CPUは実行スレッド間で均等に時間を配分します。この機能についても、「第26章 Gの実行システムについて」で説明しています。

その他のオプションは、VIの実行時にVIを印刷したり、印刷のフォーマットをカスタマイズするために使用します。プログラム印刷については、「第5章 VIの印刷および文書作成」の「プログラム印刷」の項で詳しく説明しています。

ウィンドウオプション

ウィンドウオプションは、VIの実行時に適用されますが、編集モードでは適用されません。これらのオプションは、Gの機能へのアクセスを制限したり、ウィンドウの外観や機能を変更することにより、ユーザがプログラムと対話する機能を制限するために使用します。このウィンドウが開いている間はユーザは他のウィンドウと対話することができないため、VIにダイアログボックスと同じような外観や機能を持たせることができます。また、スクロールバーやツールバーを削除したり、ウィンドウを中央に表示したり、画面に画面の大きさに合わせて自動的にサイズを変更したりすることもできます。

VIウィンドウのタイトルは、VIのファイル名よりもさらに、内容がわかりやすい名前に変更することができます。VIウィンドウの名前がローカル言語に変換される可能性がある場合、この機能はVIのローカル化にとって重要な意味を持ちます。VIの編集中にVIウィンドウの名前を変更するには、**VI設定**を選択したのち、次の図で示すように一番上のリングで**ウィンドウオプション**を選択します。**VI名と同じ**の選択を解除し、任意のVIウィンドウ名を入力します。



ダイアログボックスオプションは、システムダイアログボックスと同様、このダイアログボックスが開いている間はユーザが他のウィンドウと対話できないようにするためのものです。

ただし、UNIX では、このメニュー項目を使用しても他のアプリケーションのウィンドウが手前に表示されないようにすることはできません。ウィンドウが他のすべてのウィンドウよりも常に手前に表示されるようにする方法はありません。

ランタイムのポップアップメニュー使用可能は、このフロントパネル上のオブジェクトに、実行モードでデータを操作するためのポップアップメニューを表示させるかどうかを決定します。

メニューで**<Return>プールをハイライト**項目を選択すると、現在<Enter>キー（**Windows** および **HP-UX**）または <Return>（**Macintosh** および **Sun**）キーと関連付けられているプールがハイライトで表示されます。キーを制御器に関連付けるには、キー操作ダイアログボックスを使用します。これについては、「第8章　フロントパネルオブジェクトの概要」の「制御器のキー操作オプション」の項で説明しています。

メニューの**画面の大きさにサイズを合わせる**項目は、VI を実行する際 VI のパネル画面内に収まるようにサイズを自動的に調節します。VI には元のサイズや位置は記録されないため、編集モードに切り替えた後も VI は同じ位置に表示されます。

画面の中心に表示するは、フロントパネルを自動的にコンピュータの画面の中央に表示します。

その他のメニュー項目は、ウィンドウのさまざまな機能の表示／非表示を切り替えるためのものです。個々のボタンを非表示にするには各ボタンの選択を解除します。また、ツールバー全体を非表示にするには**ツールバーを表示**オプションの選択を解除します。

VI実行終了後にログ／印刷をイネーブルは、自動データロギング（「第4章 VIとサブVIの実行およびデバッグ」の「フロントパネルのデータロギング」の項参照）、およびプログラム印刷（「第5章　VIの印刷および文書作成」の「プログラム印刷」の項参照）の有効／無効を切り替えます。

プログラムを使用したVIウィンドウのタイトル変更については、「第21章 VIサーバ」を参照してください。

起動時にメニューを自動処理の選択を解除すると、Get Menu Selection関数を使用してメニューを選択できるようになるまで実行時のメニューバーが使用不能の状態になります。詳しくは、本章の「メニュー選択の操作」の項を参照してください。

注意　**メニューバーやツールバーを非表示にすると、実行モードから編集モードに戻る方法を表示できなくなります。VIを編集モードに戻すには、操作→編集モードに変更オプションに対応する、<Ctrl-m>（Windows）、<command-m>（Macintosh）、<meta-m>（Sun）、または<Alt-m>（HP-UX）ホットキーを使用してください。そうすると、メニューバーとパレットを表示することができます。**

文書作成オプション

次に示した**VI設定→文書作成**のダイアログボックスには、VIの履歴を表示する履歴ウィンドウへの入力に関するメニュー項目が表示されます。履歴ウィンドウについての詳細は、「第27章　アプリケーションを管理する」の「VIの履歴ウィンドウ」の項を参照してください。

文書作成メニューの履歴項目を使用するためには、**履歴デフォルトを使用する（環境設定ダイアログ）**の選択を解除しておく必要があります。このオプションが選択されていると、履歴の環境設定が使用されます。履歴の環境設定は、新たなVIを作成する際に使用したデフォルト設定をセットアップする点を除いては同一であるのに対し、VI設定の文書作成メニュー項目は現在のVIに対してのみ適用されます。

このVIが保存される度に記録を追加するを選択すると、VIを保存するたびにVIの履歴が追加されます。履歴ウィンドウのコメントボックスに**コメント**を入力しなかった場合は、ヘッダだけが履歴に追加されます。**環境設定→履歴**のダイアログボックスでこのメニュー項目が選択されていると、ヘッダにレビジョン番号、日付と時間、およびVIの名前が含まれます。

VIが閉じられる時にコメントを尋ねるオプションを選択すると、履歴ウィンドウが表示されます。ロードしたあとで変更したVIを閉じる際には、すでに変更を保存した後であってもコメントを入力することができます。

VIが保存される時にコメントを尋ねるオプションを選択すると、VIを保存する際に履歴ウィンドウが表示されます。編集中ではなく編集作業が終わった後で変更に関するコメントを入力したいときは、このオプションを

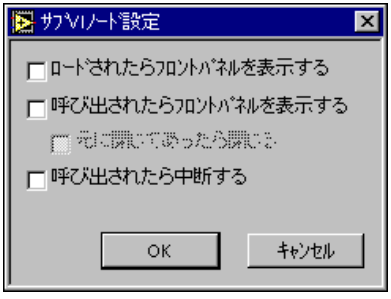
使用すると便利です。このオプションを選択しなかった場合は、**保存**を選択してから保存が完了するまでVIの履歴を変更することはできません。

☞ **注**　**VIの履歴だけを変更した場合は、VIを保存する際あるいは閉じる際にコメント入力のプロンプトは表示されません。**

VI設定→文書作成ダイアログボックスの右側にある項目を使用すると、作成したオンラインヘルプにアクセスすることができます。カーソルを**ヘルプタグ**ボックスに移動し、このVIに関して表示したいトピックを入力します。次に、**ヘルプのパス**ボックスにヘルプファイルのパスを入力するか、または**参照 ...** ボタンをクリックしてファイルを探します。ファイル名とファイルのパスが**ヘルプのパス**ボックスに表示されます。これらのメニュー項目の設定が終わったら、ヘルプダイアログボックスの下にあるオンラインヘルプのアイコンをクリックして選択したヘルプファイルにアクセスします。ヘルプファイルの作成方法については、「第5章 VIの印刷および文書作成」を参照してください。

サブVIノード設定ダイアログボックス

ブロックダイアグラム上でサブVIのポップアップメニューから**サブVIノード設定 ...** を選択すると、次のようなダイアログボックスが表示されます。



これらのメニュー項目は、VI設定ダイアログボックスにある項目のサブセットです。両者の違いは、サブVI設定ダイアログボックスがサブVIに対する特定の呼び出しに関する項目だけを指定するために使用されるのに対し、VI設定ダイアログボックスのサブVI実行項目はそのVIに対するすべての呼び出しの動作を指定するために使用される点です。

メニューバーをカスタマイズする

メニューバーは作成するすべてのVIについてカスタマイズすることができますが、カスタムのメニューバーはVIの実行中にしか表示されません。

メニューのカスタマイズには、メニューの作成とメニューの処理という2つの作業があります。

カスタムメニューの作成には2通りの方法、すなわち編集時に行う静的な方法と実行時に行う動的な方法があります。メニューエディタを使用すると、カスタムメニューのテンプレートを静的に作成し、それをランタイムメニュー（RTM）ファイルと呼ばれるファイルに保存することができます(詳しくは、本章の「メニューエディタ」の項を参照してください)。また、メニューエディタではRTMファイルを対応するVIに関連付けることもできます。VIの実行時には、VIは関連付けられたRTMファイルからメニューをロードします。実行時には、ダイアグラムからGの関数を使用してプログラム上でメニュー項目を挿入、削除、あるいは変更することができます(詳しくは、本章の「動的なメニュー関数」の項を参照してください)。

ダイアグラムは、カスタムメニュー項目を処理します。個々のメニュー項目は、タグと呼ばれる大文字／小文字の区別のない文字列からなる識別子を持っています。メニュー項目を選択するときは、選択する項目のタグをGet Menu SelectionというG関数を使用してプログラム上で検索することができます（詳しくは、本章の「メニュー選択の操作」の項を参照してください)。タグの値に基づき、ダイアグラムで個々のオプションごとにハンドラーを提供することができます。

☞ **注**　　**カスタムメニューバーは、ランタイムメニューとも呼ばれます。**

メニューエディタ

メニューエディタは、RTMファイルを作成、編集し、対応するVIと関連付けるためのインタフェースを提供します。VIに対してメニューエディタを開くには、**編集→メニューを編集 ...** を選択します。現時点では、図6-1に示すように**デフォルト**、**最小**、**カスタム**という3つのタイプのメニューをVIと関連付けることができます。デフォルトオプションは、標準メニューを提供します。最小オプションは、標準メニューから使用頻度の低い項目を削除します。カスタムオプションは、RTMファイルをVIと関連付けるために使用します。デフォルトタイプと最小タイプのメニューは、編集することができません。

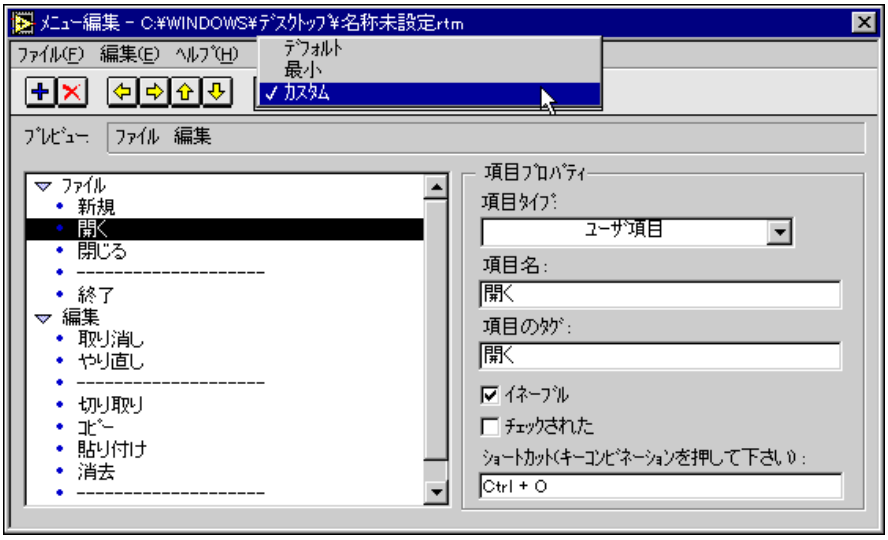


図 6-1 メニューエディタ

メニューエディタでは、左側にメニューの階層リストが、右側にメニューの階層リストから選択された項目のプロパティが表示されます。**プレビュー**領域では実行時に表示されるメニューを見ることができます。左に表示されるツールバーのボタンをクリックするか、または**編集**メニューで挿入オプションを選択すると、新しいメニュー項目を追加することができます。

メニュー項目は、すなわち**ユーザ項目**、**アプリケーション項目**、**区切り線**の3つのタイプのいずれかです。メニュー項目のタイプは、**項目タイプ**リングを使用して変更することができます。

ユーザ項目は、ダイアグラム内でプログラムによって処理されます。ユーザ項目には、実際のメニューに表示される名前と、文字列からなる固有の識別子（大文字／小文字の区別が不要）であるタグを持っています。タグは、ダイアグラムのなかでユーザ項目を識別します。編集時にわかりやすいように、入力した名前はかならずタグにコピーされます。ただし、いつでもタグを編集して名前と異なる文字列に変えることができます。メニュー項目が有効であるためには、そのタグに値がなくてはなりません。無効なオプションは [**???**] と表示されます。メニューエディタは、与えられたメニュー階層でタグが重複しないように、必要に応じて番号を追加します。

ユーザ項目は、それぞれの属性を設定することにより、有効／無効の切り替えを可能にしたり、チェックマークを付けたり消したりできるようにすることができます。また、適切なキーの組み合わせを入力することにより、

ユーザ項目にショートカット（アクセルレータ）を設定することもできます。ショートカットのフォーマットは、<Menu Key[-Shift]-key>です。

☞ **注**　PCのユーザは、項目名に下線（_）を使用することにより、<ALT>キーを使用してメニューを選択できるようにすることができます。たとえば、File/Exit という名前の項目を<ALT+F+X>で選択できるようにするには、_File and E_xit という項目名を使用します。

☞ **注**　Macintosh OS 7以下のバージョンでは、ショートカットのシフトキーやファンクションキーは表示されません。メニューを Macintosh OS 7 に移植できるようにするためには、ショートカットにシフトキーやファンクションキーを使用しないでください。

☞ **注**　Macintosh プラットフォームでは、メニューバーの項目はいずれもプルダウンメニューでなければなりません。一方、Windows 95、Windows NT、および UNIX のユーザは、メニューバーで選択可能項目を作成することができます。Macintosh では選択可能項目は使用しないでください。

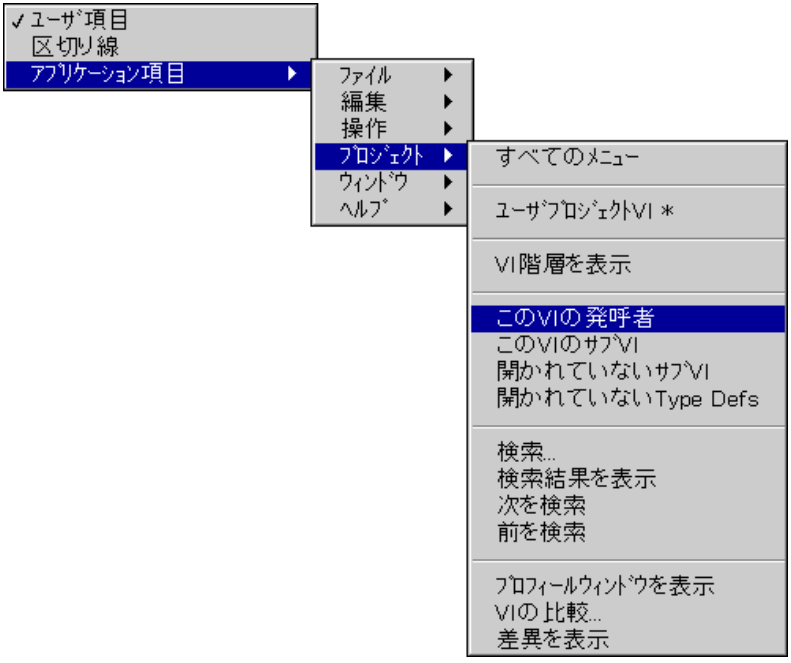


図 6-2 アプリケーション項目の選択

アプリケーション項目は、G 言語のソフトウェアが提供するオプションです。これらは、デフォルトメニューに含まれるオプションです。ある特定の項目を選択するためには、図で示すように、**項目タイプ→アプリケーション項目**を選択したのち、階層メニューを使用して選択を行います。このプロセスを使用すると、個々の項目あるいはサブメニュー全体を追加するこ

とができます。アプリケーション項目は、ソフトウェアによって内部的に処理されます。これらの項目のタグは、ダイアグラムには表示されません。アプリケーション項目の名前、タグ、およびその他の属性は変更できません。ソフトウェアは、APP_ で始まるタグはアプリケーション項目専用のタグとして使用します。アプリケーション項目のタグは、本章の最後に表6-1で一覧を示します。

区切り線は、メニューに区切り線を挿入します。区切り線に属性を設定することはできません。

メニュー階層は、ツールバーの矢印ボタンをクリックするか、あるいは**編集**メニューの階層操作オプション、またはドラッグアンドドロップを使用して、メニュー階層の中で移動することができます。階層は、サブメニューグリフをクリックするか、または**編集**メニューで拡大／縮小オプションを選択することにより、見やすいように縮小または拡大することができます。

RTMファイルは、**ファイル**メニューのオプションを使用してロードしたり保存したりすることができます。**編集**メニューのオプションを使用すると、メニュー項目の切り取り、コピー、貼り付け、挿入、あるいは削除を行うことができます（詳しくは、本章の「メニューエディタのオプション」の項を参照してください）。メニュー編集の制御器について詳しく知りたいときは、**ヘルプ→ヘルプを表示**を選択して、マウスをその制御器の上に移動します。

メニューエディタのオプション

ファイルメニューでは、下記のオプションを使用できます。

開く — 既存のランタイム時メニューのファイルを開きます。RTMファイルを開くと、メニューエディタはカスタムタイプに切り替わります。

新規 — 新たなランタイム時メニューのファイルを作成します。それまでに編集したRTMファイルを保存するよう指示するメッセージが表示されます。

保存 — 現在のランタイム時メニューを保存します。

別名で保存 — 現在のランタイム時メニューを別のファイルに保存します。

閉じる — メニュー編集を終了します。

編集メニューでは、下記のメニュー項目を使用できます。

切り取り — 現在選択されているメニュー項目またはテキストを削除し、クリップボードにコピーします。

コピー — 現在選択されているメニュー項目またはテキストをクリップボードにコピーします。

すべてのメニューをコピー — メニューの階層全体をクリップボードにコピーします。

貼り付け — クリップボードの内容を貼り付けます。

格納／展開 — 現在選択されているサブメニュー項目を縮小または拡大します。

すべて格納 — すべてのサブメニュー項目を縮小します。

すべて展開 — すべてのサブメニュー項目を拡大します。

ユーザ項目を挿入 — メニューから選択したオプションの後にユーザ項目を挿入します。

区切り線を挿入 — メニューから選択したオプションの後に区切り線を挿入します。

アプリケーション項目を挿入 — サブメニューから使用可能なアプリケーション項目を選択します。

項目を削除 — メニューから選択した項目を削除します。

スーパー項目に変更 — 選択した項目の下にある項目を選択した項目のサブ項目にします。

サブ項目に変更 — 選択した項目の上にある項目を選択した項目のサブ項目にします。

上に移動 — 選択した項目を上に移動します。選択した項目にサブ項目があるときは、サブ項目も選択した項目とともに移動されます

下に移動 — 選択した項目を下に移動します。選択した項目にサブ項目があるときは、サブ項目も選択した項目とともに移動されます。

メニューエディタのツールバー項目

ツールバーでは、下記の項目を使用できます。

メニュー選択した項目の後にユーザ項目を挿入します。

メニュー選択した項目を削除します。

選択した項目の下にある項目を選択した項目のサブ項目にします。

選択した項目を上の項目のサブ項目にします。

選択した項目を上に移動します。選択した項目にサブ項目があるときは、サブ項目も選択した項目とともに移動されます。

選択した項目を下に移動します。選択した項目にサブ項目があるときは、サブ項目も選択した項目とともに移動されます。

メニュー選択の操作

この項では、Get Menu Selection関数とEnable Menu Tracking関数について詳しく説明します。これらの関数の説明は、本章の「メニュー選択処理関数」の項に記載されています。これらの関数は、refnumで特定されたメニューに対して実行されます。VIのメニューrefnumは、左の図、および次の図6-3と図6-4に示したCurrent VI's Menuから受け取ります。

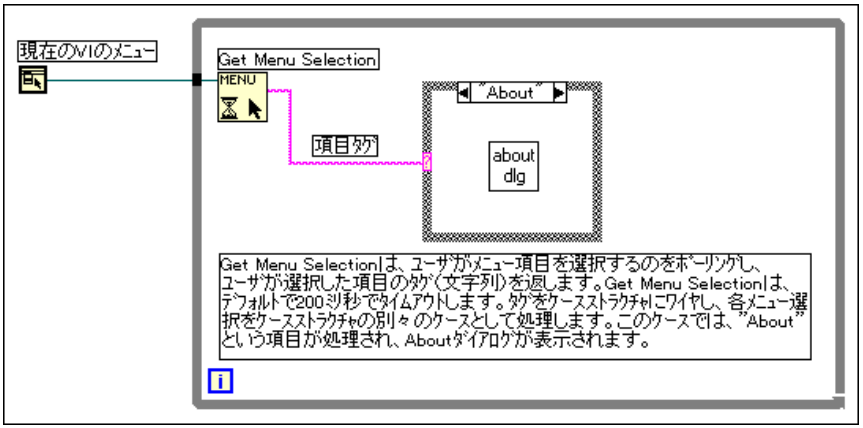


図 6-3 Get Menu Selection関数

1つの項目を選択すると、Get Menu Selection関数が最初の項目を読み取るまでは次の項目の選択はできなくなります。ただし、項目の読み取りが済んだあとは、前の項目の処理が終わっていなくても次の項目を選択することができます。選択の処理に時間がかかる場合、通常は上記の方法はとりません。そのような場合は、ブロックメニューモードでGet Menu Selectionを呼び出します。このモードでは、項目を読み取ったあとのメニューの追跡は禁止されます。次の図で示すように、Enable Menu Tracking関数を使用して選択項目を処理した後、メニューが使用可能になります。

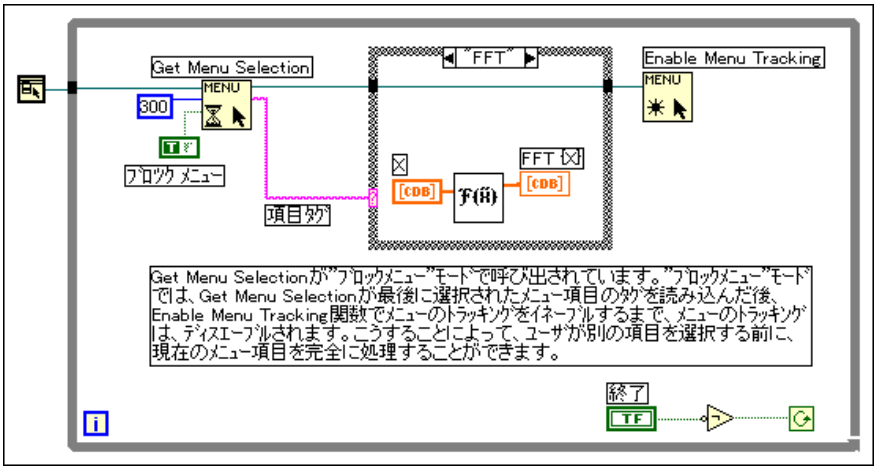


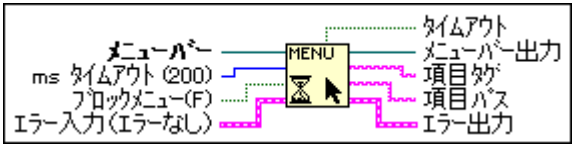
図 6-4 Enable Menu Tracking 関数

メニュー選択処理関数

下記のメニュー選択処理関数が使用できます。

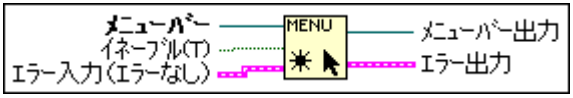
Get Menu Selection

最後に選択したメニュー項目の**項目タグ**を返します。オプションとしてタイムアウトの設定が行えます。**項目パス**は、メニュー階層中の項目の位置を記述した文字列で、パスのフォーマットはコロン（**:**）で区切ったメニュータグのリスト形式になります。**ブロックメニュー**の設定 True のときは、項目タグを読み取ったあとメニューの選択は禁止されます。



Enable Menu Tracking

メニュー選択の追跡を有効または無効にします。



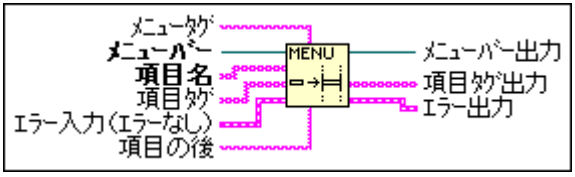
関数は、refnumによって特定されたメニューに対して実行されます。VIのメニューrefnumは、左に示したCurrent VI's Menuから受け取ります。項目は項目タグ（文字列）によって特定されますが、ときには項目パス（文字列）によって特定される場合もあります。項目パスは、メニューのルートから項目までの項目タグをコロンでつないでリストにしたものです。

動的なメニュー関数

VIの実行中に、Insert Menu Items、Delete Menu Items、Get Menu Item Info、Set Menu Item Info、およびGet Menu Shortcut Infoといったメニュー管理関数を使用してVIのメニューバーの項目を変更することができます。これらの関数は**アプリケーション制御**パレットにあり、このパレットは**関数→アプリケーション制御**を選択することによりアクセスできます。これらの関数は、refnumによって特定されたメニューに対して実行されます。メニューのrefnumは、左記のCurrent VI's Menuから受け取ります。

Insert Menu Items

メニューバー、またはメニューバーのサブメニューにメニュー項目を挿入します。



メニュータグは、項目を挿入するサブメニューを指定します。**メニュータグ**を指定しなかった場合、項目は**メニューバー**に挿入されます。

項目名と**項目タグ**は、メニューに挿入する項目を特定します。**項目名**と**項目タグ**のタイプは、文字列の配列（複数の項目を挿入する場合）、または単なる文字列（1つの項目を挿入する場合）のいずれかです。**項目名**と**項目タグ**のいずれか一方だけで配線することもできますが、その場合は名前とタグは同じ値になります。各項目に異なる名前とタグを付けたい場合は、**項目名**と**項目タグ**を別々の値で配線する必要があります。

項目の後は、項目を挿入する場所を指定します。**項目の後**は、既存の項目のタグ（文字列）でも、メニューにおける位置を指す指標（ゼロベースの整数）でもかまいません。メニューの先頭に挿入する場合は、**項目の後**に0より小さい数字を配線します。メニューの最後に挿入する場合は、メニューの項目の数より大きい数字を配線します。アプリケーション項目は、本章の最後に記載した表6-1のアプリケーションタグを使用して挿入します。区切り線は、`APP_SEPARATOR`というアプリケーションタグを使用し

て挿入します。関数は、挿入されたメニュー項目のタグがメニュー階層で重複しないように、割り当てられたタグに必要に応じて番号を追加します。

項目タイプ出力は、挿入した項目の実際のタグを返します。**メニュータグ**または**項目の後**（タグ）が見つからない場合は、エラーを返します。

次の図に示した例では、**ファイル**および**編集**という最上位の2つのメニューと**テスト**というサブメニューで構成されるメニューツリーが作成されます。

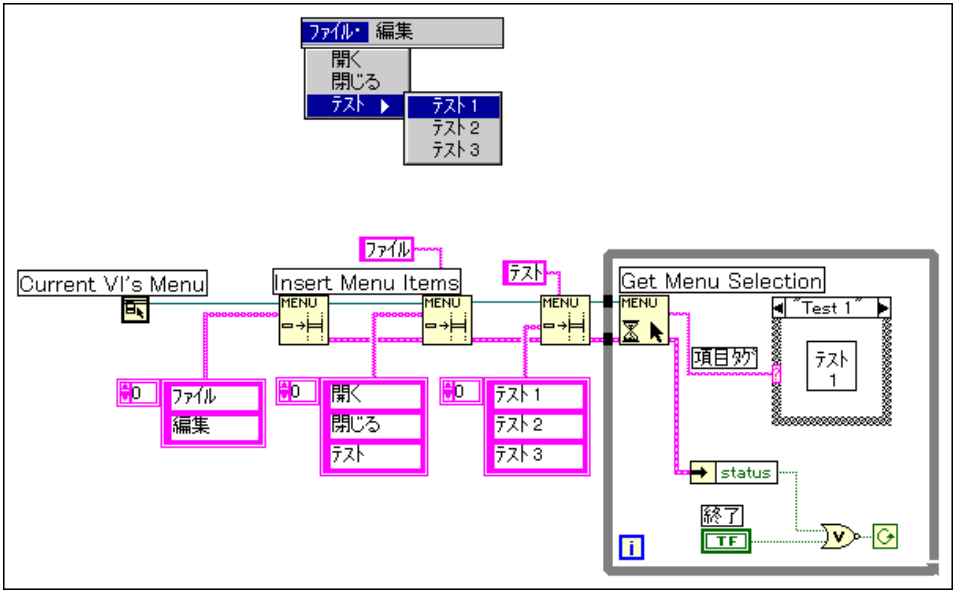
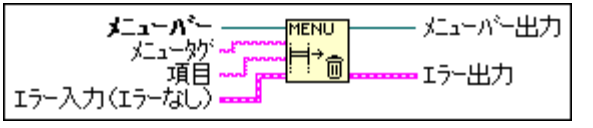


図 6-5 図動的なメニューの挿入

Delete Menu Items

メニューバー、またはメニューバーのサブメニューから項目を削除します。

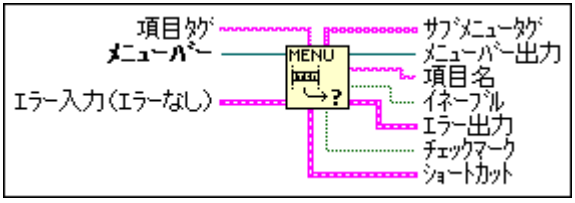


メニュータグが指定されている場合は、**メニュータグ**で指定したサブメニューから項目が削除され、それ以外の場合はメニューバーから項目が削除されます。**メニュータグ**あるいは指定された項目のいずれかが見つからないときは、関数はエラーを返します。

項目は、既存の項目のタグ（文字列）、既存の項目のタグの配列、メニューにおける項目の位置を指す指標（ゼロベースの整数）、または指標の配列のいずれでもかまいません。**項目**を配線しなかった場合は、メニューバーのすべての項目が削除されます。指定した項目のいずれかにサブメニューがある場合は、サブメニューとそのすべての内容が自動的に削除されます。アプリケーション項目は、本章の最後に記載した表 6-1 のアプリケーションタグを使用して削除することができます。区切り線には固有のタグがないため、区切り線は位置を示す指標を使用して削除するのが最もよい方法です。

Get Menu Item Info

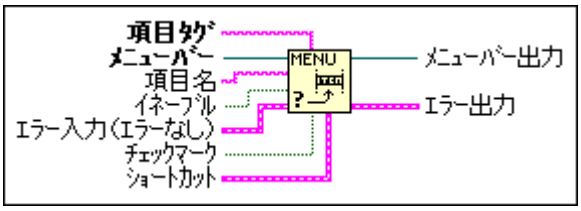
項目タグで指定したメニュー項目の属性を返します。



オプションの属性には、**項目名**（メニューに表示される文字列）、**イネーブル**（False のときは項目をグレーで表示）、**チェックマーク**（項目のとなりにチェックマークを表示するかしないかを指定）、**ショートカット**（キーアクセルレータ）があります。項目にサブメニューがある場合は、**サブメニュータグ**でその項目タグが文字列の配列として返されます。**項目タグ**が配線されていない場合は、メニューバーの項目が返されます。**項目タグ**が無効である場合は、エラーが返されます。

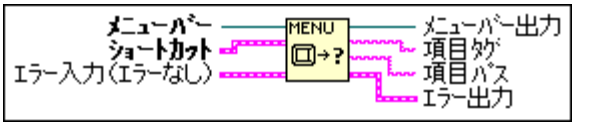
Set Menu Item Info

メニューおよび項目のタグで指定された項目の属性を設定します。項目の属性には、項目名（メニューに表示される文字列）、イネーブル（False のときは項目をグレーで表示）、チェックマーク（項目のとなりにチェックマークを表示するかしないかを指定）、ショートカット（キーアクセルレータ）があります。配線されていない属性は変更されません。項目タグが無効である場合は、エラーが返されます。



Get Menu Shortcut Info

所定のショートカットでアクセスできるメニュー項目を返します。



項目パスは、項目タグをコロン（:）でつないだ文字列です。

ショートカットは、文字列（キー）とブール（シフトキーを含むか否かを指定）で構成されます。

アプリケーション項目タグ

アプリケーション項目は、次の表に示したアプリケーション項目のタグを使用して挿入することができます。

表 6-1 アプリケーション項目タグ

アプリケーション項目	項目タグ
ファイル	APP_FILE
新規	APP_NEW
開く...	APP_OPEN
閉じる	APP_CLOSE
保存	APP_SAVE
別名で保存...	APP_SAVEAS
コピーの保存...	APP_SAVE_A_COPY_AS
オプション付き保存...	APP_SAVE_WITH_OPTIONS
プリンタ設定...	APP_PRINTER_SETUP
文書の印刷...	APP_PRINT_DOCUMENTATION
ウィンドウの印刷...	APP_PRINT_WINDOW
VIライブラリの編集...	APP_EDIT_VI_LIBRARY
テンプレートの編集...	APP_EDIT_TEMPLATE
一括コンパイル...	APP_MASS_COMPILE
CVI FPファイルの変換...	APP_CONVERT_CVI

表 6-1 アプリケーション項目タグ（続き）

アプリケーション項目	項目タグ
終了	APP_EXIT
編集	APP_EDIT
取り消し	APP_UNDO
やり直し	APP_REDO
切り取り	APP_CUT
コピー	APP_COPY
貼り付け	APP_PASTE
消去	APP_CLEAR
ファイルから画像をインポート...	APP_IMPORT_PICT_FROM_FILE
環境設定...	APP_PREFERENCES
ユーザ名...	APP_USER_NAME
パスワードキャッシュ消去	APP_CLEAR_PASSWORD_CACHE
パレットセットを選択	APP_SELECT_PALETTE_SET
制御器と関数パレットの編集...	APP_EDIT_PALETTES
操作	APP_OPERATE
停止	APP_STOP
終了後に印刷	APP_PRINT_AT_COMPLETION
終了後にログ	APP_LOG_AT_COMPLETION
データロギング	APP_DATA_LOGGING
ログ...	APP_LOG
回収...	APP_RETRIEVE
データ一掃...	APP_PURGE_DATA
ファイル連結の変更...	APP_CHANGE_LOG_BINDING
ファイル連結の消去	APP_CLEAR_LOG_BINDING
呼び出されたら中断	APP_SUSPEND_WHEN_CALLED
現在のすべての設定をデフォルト設定にする	APP_REINIT_ALL_TO_DEFAULT
プロジェクト	APP_PROJECT

表 6-1 アプリケーション項目タグ（続き）

アプリケーション項目	項目タグ
User Project VIs *	APP_USER_PROJECT_VIS
VI階層を表示	APP_SHOW_VI_HIER
このVIの発呼者	APP_THIS_VIS_CALLERS
このVIのサブVI	APP_THIS_VIS_SUBVIS
開かれていないサブVI	APP_UNOPENED_SUBVIS
開かれていないType Def	APP_UNOPENED_TYPEDEFS
検索...	APP_FIND
検索結果	APP_SHOW_SRCH_RSLTS
次を検索	APP_FIND_NEXT
前を検索	APP_FIND_PREV
プロフィールウィンドウを表示	APP_SHOW_PROFILE_WINDOW
VIの比較...	APP_COMPARE_VIS
差を表示	APP_SHOW_DIFFERENCES
文字列のエクスポート...	APP_EXPORT_STRINGS
文字列のインポート...	APP_IMPORT_STRINGS
ウィンドウ	APP_WINDOWS
パネルを表示	APP_SHOW_PANEL_DIAGRAM
VI情報...	APP_SHOW_VI_INFO
履歴を表示	APP_SHOW_HISTORY
制御器パレットを表示	APP_SHOW_CTRLS_FCTNS
ツールパレットを表示	APP_SHOW_TOOLS
クリップボートを表示	APP_SHOW_CLIPBOARD
エラーリストを表示	APP_SHOW_ERRORS
左右にならべて表示	APP_TILE_LEFT_RIGHT
上下にならべて表示	APP_TILE_UP_DOWN
全画面表示	APP_FULL_SIZE
Open Windows *	APP_OPEN_WINDOWS

表 6-1 アプリケーション項目タグ（続き）

アプリケーション項目	項目タグ
ヘルプ	APP_HELP
ヘルプを表示	APP_SHOW_HELP
ヘルプをロック	APP_LOCK_HELP
シンプルヘルプ	APP_SIMPLE_HELP
オンラインリファレンス...	APP_ONLINE_REF
VIのオンラインヘルプ	APP_ONLINE_HELP_FOR_VI
Help Files *	APP_HELP_FILES
LabVIEWについて...	APP_ABOUT

7

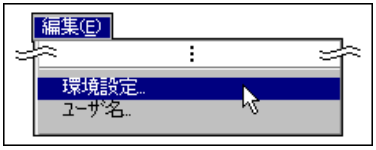
環境をカスタマイズする

この章では、印刷、表示、**取り消し**などの機能に関するエディタの環境設定、および**制御器**パレットや**関数**パレットの内容の変更によって環境をカスタマイズする方法について説明します。

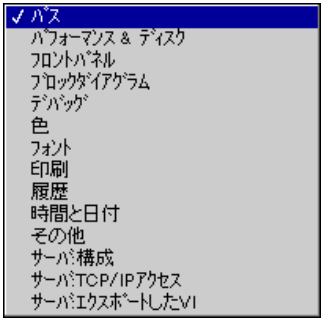
環境を設定する

アプリケーションのカスタマイズおよびエディタの構成には、環境設定ダイアログボックスを使用します。

環境設定ダイアログボックスは、次に示すように**編集→環境設定...**を選択することによって呼び出すことができます。

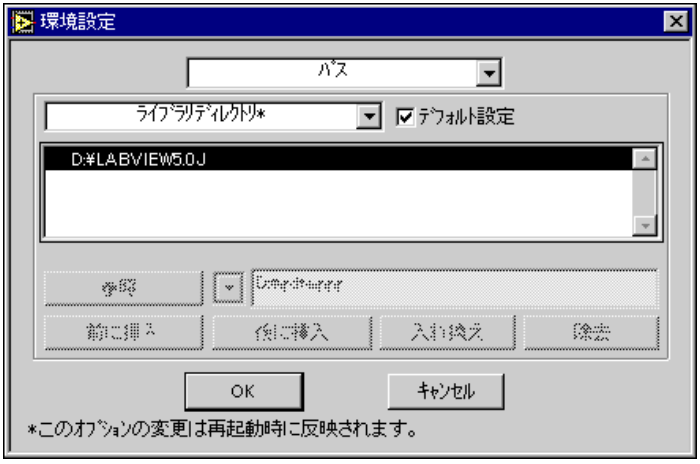


環境設定のさまざまなカテゴリーの選択には、次に示すようにダイアログボックスの一番上にあるプルダウンメニューを使用します。



パスの環境設定

VIを検索する際に検索するディレクトリ、一時ファイルに対して使用するパス、およびライブラリディレクトリを指定することができます。パスが選択されていない場合は、環境設定ダイアログボックスのプルダウンメニューで**パス**を選択し、次の図で示すダイアログボックスを呼び出します。



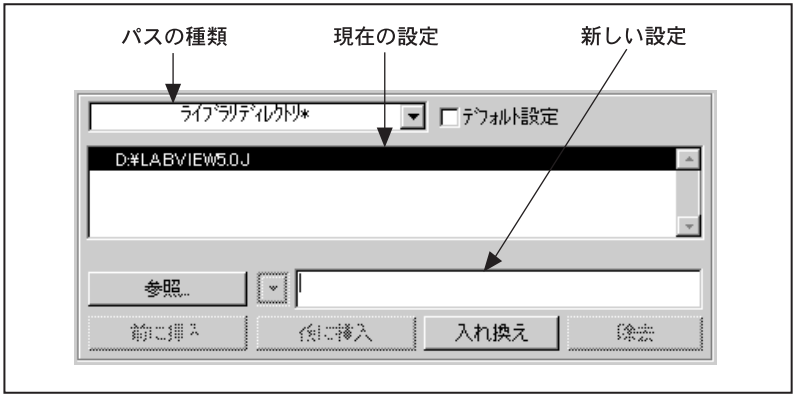
☞ **注**　**パス名のフォーマット（コロン、スラッシュ、円マークの使用方法）は、プラットフォームによって若干異なります。**

パスの環境設定のダイアログボックスには、次の図に示すようなもう1つのプルダウンメニューがあります。このメニューでは、表示あるいは編集したいパスの種類を選択します。

表示されるダイアログボックスでは多くの項目がグレーで表示され、使用できない状態になっていますが、これはGがデフォルトパスに設定されているためです。これらの環境設定を変更したい場合は、次の図で示すように**デフォルト設定**のチェックボックスの選択を解除します。

ライブラリディレクトリ、テンポラリディレクトリ、デフォルトディレクトリ、およびメニューディレクトリ

ライブラリディレクトリ、**テンポラリディレクトリ**、**デフォルトディレクトリ**、および**メニューディレクトリ**は、それぞれが1つのディレクトリ（フォルダ）です。これらのいずれかのパスを編集する場合は、新しいパスを入力して既存のパスと差し替えるか、またはファイルダイアログボックスを呼び出してパスを選択するかのどちらかを選択できます。これを次の図に示します。



ライブラリディレクトリは、`vi.lib`が保存されているディレクトリ、およびユーザが提供するライブラリディレクトリの絶対パス名を表示します。デフォルトは、Gの開発環境が保存されているディレクトリです。

テンポラリディレクトリは、一時ファイルを保存するディレクトリの絶対パス名を表示します。このディレクトリは、次に示すようにプラットフォームによって異なります。

- **（Windows）**デフォルトは、Gの開発環境が保存されているディレクトリです。
- **（Macintosh）**システム7では、デフォルトはハードドライブの最上位に位置する`Temporary Items`と呼ばれる見えないフォルダです。Appleは、一時ファイルをこのディレクトリに保存することを推奨しています。開発環境がクラッシュすると、再起動した際に一時ファイルがゴミ箱に移されます。
- **（UNIX）**デフォルトは、`/tmp`ディレクトリです。

デフォルトディレクトリは、ファイルダイアログボックスが最初に表示する**デフォルトディレクトリ**の絶対パス名を表示します。**関数→ファイルI/O→ファイル定数**パレットのデフォルトディレクトリ関数もこの値を返します。デフォルトは、現在の作業ディレクトリです。デフォルトディレクトリへの変更は直ちに実行されます。

メニューディレクトリは、**制御器**パレットおよび**関数**パレットの構成を記述したパレットメニューファイルが保存されているディレクトリのパスを表示します。デフォルトでは、**ライブラリディレクトリ**内の**メニューディレクトリ**になります。通常はこのディレクトリを変更する必要はありませんが、ネットワークインストールを行い、個々のユーザがそれぞれ独自のメニューセットを必要とする場合は変更することができます。

☞ **注**　**ライブラリディレクトリ、テンポラリディレクトリ、デフォルトディレクトリ、およびメニューディレクトリの設定の変更は、いずれもＧの開発環境を再起動してはじめて有効になります。**

VI 検索パス

VI 検索パスは、Gの開発環境が予想した場所で見つからなかったサブVIや制御器、あるいは外部のサブルーチンを検索する場合にのみ使用されます。VI 検索パスを使用すると、検索するパスリストを作成することができます。検索パスを編集する際には、他のパスよりも多くのオプションを使用でき、特定の場所に新しい項目を追加したり、パスを削除したり、**特殊**パスのリストからパスを選択したりすることができます。

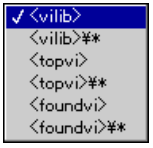
リストには、Gの開発環境が検索するパスが検索順に表示されます。新しいディレクトリを追加する場合は、まず最初に、他のディレクトリとの関係において、Gの開発環境にそのディレクトリをいつ検索させるかを決定します。隣接するディレクトリをリストから選択します。次に、**参照...**ボタンを使用してファイルダイアログボックスを呼び出すか、または特殊なプルダウンメニューを使用して特殊パスのリストを画面に表示したうえで、検索するディレクトリを選択します。これらのオプションの横にある文字列制御器でこのパス編集または入力します。最後に、**前に挿入**、**後に挿入**、または**入れ換え**を使用してパスをリストに追加します。

パスを選択すると、通常Gはそのディレクトリを検索しますが、そのディレクトリのサブディレクトリは検索しません。検索を階層的に行うためには、新しいパス項目として * を追加します。

各プラットフォームでの例を次に示します。

- **（Windows）**C:¥VIs¥ というディレクトリを再帰的に検索するには、[C:¥VIs¥*] とタイプします。
- **（Macintosh）**ハードディスクでVIsというフォルダを再帰的に検索するには、[HD:VIs:*] とタイプします。
- **（UNIX）**/usr/home/gregg/vis を再帰的に検索するには、[/usr/home/gregg/vis/*] とタイプします。

いくつかの特殊ディレクトリのなかから選択する場合は、**参照...** ボタンの右にある特殊なプルダウンメニューを使用します。



これらのディレクトリは下記の通りです。

- <vilib>というパス入力は、**ライブラリディレクトリ**の中のvi.libというディレクトリを指すシンボリックパスです。
- <topvi>は、開いている最上位のVIが保存されているディレクトリを指します。
- <foundvi>は、VIをロードするたびにLabVIEWが作成するディレクトリのリストを指します。検索時には、LabVIEW がサブ VI を検出するディレクトリ、あるいはユーザがマニュアル（手作業）で選択したディレクトリがこのリストに追加されます。ディレクトリを移動したり、ディレクトリの名前を変更したあとで呼び出すVIを開く場合は、そのロードのためにそのディレクトリを一回だけ検索する必要があるため、このシンボリックパスを使用してください。

パスを削除するには、**除去**ボタンを使用します。削除したパスは、リストの別の位置に再び挿入する場合に備えて、文字列ボックスに保存されます。

☞ **注**　**VI検索パス**への切り替えは、次回アプリケーションを起動したときに有効になります。

パフォーマンスとディスクの環境設定

編集作業によっては、ディスクで一時ファイルを使用する必要のある場合があります。たとえば、VI を保存すると、VI はまず一時ファイルに保存されます。保存が正しく実行されると、新しいファイルが元のファイルと差し替えられます。これにはかなりのディスク容量が必要となる場合もあります。問題を避けるため、起動時には一時ディレクトリの使用可能なディスク容量がチェックされます。ディスクの空き容量が 500 KB 未満のときには、プログラムは警告を発し、起動を中止します。空き容量が 2 MB 未満のときには、警告は発しますが起動は続行します。これらの警告は、環境設定のプルダウンメニューからアクセスするパフォーマンス&ディスクダイアログボックスを使用してオフにしたり、警告を発するレベルを変更したりすることができます。

パフォーマンスとディスクの環境設定ダイアログボックスを次の図に示します。Windows 3.1 ではメニュー項目が 1 つしか使用できませんが、Macintosh ではこの他のオプションが使用できます。

このダイアログボックスのオプションは、下記の通りです。

メモリをなるべく早く解放する — 個々のVIの実行が終了するたびにその VI のメモリを解放します。サブ VI は実行後直ちにメモリを解放するため、このオプションを使用すると一部のアプリケーションではメモリの使用効率を上げることができます。ただし、G がメモリを割り当てたり解放したりする回数が多くなるためパフォーマンスが低下し、場合によってはメモリが分割され過ぎるおそれがあります。

（Windows 3.1）デフォルトタイマを使用する — Gのタイミング関数は、内蔵の Windows タイマを使用します。このタイマのデフォルトの分解能は 55 ms です（1 秒間に 18 回増分）。これは、G の Tick Count、Wait、および

Wait Until Next ms Multiple VIが55 msの分解能しかないことを意味します。ただし、DAQ VIはDAQボードの内蔵タイマを使用して集録速度を制御するため、DAQ VIではより正確なタイミング速度を指定できる点に注意してください。

Windowsの内蔵タイマの分解能を1 msに上げることにより、Tick Count、Wait、およびWait Until Next ms Multiple VIの分解能を1 msにまで上げることができます。タイマの分解能を上げると、ソフトウェアのタイミングループが一層正確になります。

状況によっては、分解能を55 msのままで使用した方が望ましい場合があります。分解能を上げると、タイマのチックの割り込み回数が毎秒18回から毎秒1,000回に増加します。ほかの割り込み中心の操作を実行する場合、PCの割り込み負荷の処理能力を超えてしまい、PCがロックしたりクラッシュしたりするおそれがあります。

DAQボードとPCメモリ間のデータ転送にDMAではなくプログラムI/Oを使用したDAQ操作は、割り込み中心の操作です。高速の転送速度で次のような操作を実行しようとする場合は、タイマのチックの分解能を1 msに変更しないようにしてください。

- PC-LPM-16を使用したバッファアナログ入力
- AT-MIO-16、MC-MIO-16、またはLabシリーズのボードを使用したバッファを使用したアナログ出力
- DIO-24、DIO-96、またはLabシリーズのボードを使用したバッファを使用したデジタルI/O
- DAQボードを使用し、`WDAQCONF`でDMAをオフにしたバッファを使用したアナログ入力または出力

386 25 MHzマシンで30 kサンプル/秒以上の転送速度を使用すると、1 msのタイマ分解能では割り込み負荷の問題が発生します。この問題は、486 33 MHzマシンで75 ～ 80 kサンプル/秒以上の転送速度を使用した場合、あるいは486 50 MHzマシンで100 kサンプル/秒以上の転送速度を使用した場合にも発生する可能性があります。

Windows 95/NT対応しているGのタイミング関数のタイマ分解能は、Windows 95やWindows NTでの最大分解能である1 msです。低速のマシンでは分解能がこれよりも低い場合があります。

（Macintosh）実行中にメモリを圧縮する — 約30秒に1回メモリを圧縮するようGに指示します。圧縮は、割り当てられたメモリを1カ所に移動してディスクの分割を少なくします。このオプションの欠点は、圧縮に時間がかかるため、メモリの圧縮中に一時的に性能が低下するおそれがある点です。

（Macintosh）協力レベル — このオプションを使用すると、Gとバックグラウンドで実行している他のアプリケーションとの連動状況を制御することができます。このオプションを低に設定すると、Gが他のアプリケーションと時間を共有する回数が少なくなります。低に設定するとGのパフォーマンスは向上しますが、同時に実行している他のアプリケーションのパフォーマンスは低下します。高に設定した場合は、Gのパフォーマンスは低下しますが、他のアプリケーションにはより多くの実行時間が与えられます。

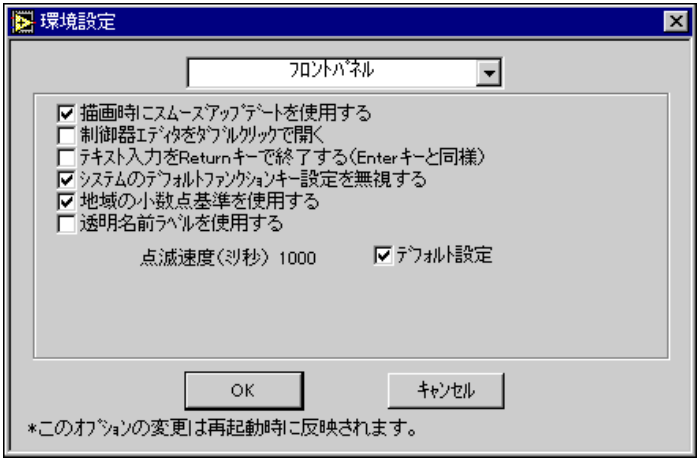
起動時に空きディスク容量をチェック — Gの起動時に一時ディレクトリ内の使用可能なディスク容量をチェックします。500 KB未満であれば中止、2,000 KB未満であれば警告という2つのチェックを、Gにデフォルトで行うかどうかを指定することができます。

マルチスレッドで実行 — このオプションは、マルチスレッドをサポートしているシステムでの操作にのみ使用できます。このオプションを無効にしておくと、実行システムは1つのスレッドしかないときと同じように動作し、ユーザインタフェースを使用した対話とVIの実行がいずれも同じスレッドの中で発生します。マルチスレッドの実行システムでは正しく動作しないVIがある場合は、互換性を考慮してこの方法を使用することがあります。

☞ **注**　**起動時に空きディスク容量をチェック、デフォルトタイマを使用する、およびマルチスレッドで実行への変更は、Gを再起動したあとに有効になります。実行中にメモリを圧縮する、メモリをなるべく早く解放する、および協力レベルの各オプションの変更は、直ちに有効になります。**

フロントパネルの環境設定

フロントパネルの環境設定ダイアログボックスを次の図に示します。



このダイアログボックス内のオプションは下記の通りです。

制御器エディタをダブルクリックで開く ― フロントパネルの制御器や表示器の外観をカスタマイズするための制御器エディタウィンドウを、簡単に呼び出せるようにします。

テキスト入力を Return キーで終了する（Enter キーと同様） ― テンキーパッドの<Enter>キーと同じように、英数字キーボードの<Return>キーでもテキストの入力を終了できるようにします。このオプションを選択すると、<Ctrl-Enter>**（Windows）**、<option-return>**（Macintosh）**、<meta-Return>**（Sun）**、または<Alt-Return>**（HP-UX）**を押すことによって新たに行を挿入することができます。

（Windows、Macintosh）システムのデフォルトファンクションキー設定を無視する ― オペレーティングシステムがデフォルトとして、システムを対象とした一部のファンクションキーを使用する場合があります。たとえば、PCでF10を押す動作は<Break>キーを押すのと同じです。Macintoshでは、F1 ～ F4 のキーはそれぞれ**切り取り、コピー、貼り付け、消去**として扱われます。このオーバーライドオプションを有効にすると、ファンクションキーはシステムを対象としたキーとしてではなく、標準のファンクションキーとしてGに渡されます。

地域の小数点基準を使用する ― ピリオドの代わりにシステム用の小数点を使用します。たとえば、多くの国ではコンマが小数点として使用されます。G を独自のシステム構成の中で使用したいときは、このオプションを有効にします。どんな場合にもかならずピリオドを小数点として使用したい場合は、このオプションを無効にしておきます。

描画時にスムーズアップデートを使用する ― Gは、スムーズアップデートを無効にした状態で制御器を更新するときには、制御器の内容を消去して新しい値を描画します。その場合、古い値の消去と差し替えが行われるために画面のちらつきが目立つようになります。スムーズアップデートを使用すると、G は画像の一部を消去せずに、画面の外のバッファにデータを描画してからその画像を画面にコピーするため、画像の消去による画面のちらつきの発生を防ぐことができます。ただし、スクリーンの外に描画するためのバッファを維持することが必要になるため、パフォーマンスが低下するおそれがあります。

透明名前ラベルを使用する ― 透明なラベルを使用するようソフトウェアに指示します。

（Sun）Support numeric keypad on Sun keyboards ― Sun のキーボード（キーパッド、矢印キー、ヘルプキーを含む）のサポートを有効にします。Sun 以外のキーボードを使用している場合は、このオプションは有効にしないでください（X ターミナル、あるいは X ソフトを実行している PC を使用している場合など）。

点滅速度 ― フロントパネルオブジェクトの点滅速度を、デフォルトの1,000ミリ秒、またはユーザが入力したその他の値に設定します。点滅は、属性ノードによってオンになる基本的な属性です。詳しくは、「第22章　属性ノード」を参照してください。

☞ **注**　**フロントパネルの環境設定ダイアログボックスで変更した内容は、直ちに有効になります。**

ブロックダイアグラムの環境設定

ブロックダイアグラムの環境設定ダイアログボックスを次の図に示します。

このダイアログボックスのオプションは下記の通りです。

ワイヤの接点でドットを表示する ― ワイヤが複数の接続先に分岐している箇所にドットを表示します。それにより、分岐点とワイヤが単に交差しているだけの点の区別が容易になります。

端子のヒントラベルを表示する ― カーソルを関数やサブVI内の端子に合わせたときに、端子の下にパラメータ名を表示します。

サブVIのヒントラベルに制御器のキャプションを使用 ― VIのヘルプを表示したりVIの端子のヒントラベルを表示する際に、制御器の名前でなく制御器のキャプションを使用します。

配線ガイドを表示する ― カーソルをブロックダイアグラムのノードに合わせたときに、各パラメータのデータタイプとデータの方向を示すワイヤスタブを表示します。

ドロップ時にサブ VI 名を表示する — VI をブロックダイアグラム上にドロップしたときに、サブ VI にもとの VI の名前でラベルを付け、そのラベルを表示するようソフトウェアに指示します。

透明名前ラベルを使用する — 透明なラベルを使用するようソフトウェアに指示します。

関数パレット上でウィンドウタイトルを使用 — 関数パレットで VI のパレット名を表示する際に、VI 名ではなくウィンドウタイトルを使用します。

最大取り消し回数 — 取り消しおよびやり直しが可能な動作の数を設定します。値をゼロに設定すると、取り消しは不可能になります。各 VI にそれぞれの取り消しのインスタンスが記録されるため、入力した値はその VI だけに適用されます。この値を小さい値に設定することにより、メモリの使用量を節約できます。詳しくは、本章の「取り消し」の項を参照してください。

☞ **注**　**ブロックダイアグラムの環境設定ダイアログボックスで変更した内容は、直ちに有効になります。**

デバッグの環境設定

デバッグの環境設定ダイアログボックスを次の図に示します。

このダイアログボックスのオプションは下記の通りです。

実行ハイライトのデータバブルを表示する — ワイヤに沿ってバブルを描画し、実行の流れを動画で表示します。動きを少なくして速やかにデバッグを行いたいときは、この機能を無効にします。

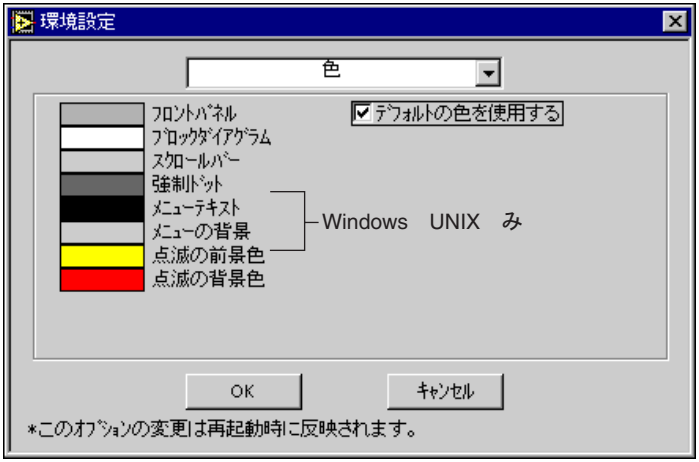
実行ハイライトの自動プローブを表示する ― スカラ値を自動的に検出し、値をダイアグラム上に表示します。そのために表示にぶれが生じる場合は、この機能を無効にします。

デフォルトとして警告をエラーボックス内に表示する ― エラーリストダイアログボックスにエラーのほかに警告も表示します。警告はブロックダイアグラム内の潜在的な問題を示すもので、VIが正しくないことを意味するものではありません。

☞ **注**　**デバッグの環境設定ダイアログボックスで変更した内容は、直ちに有効になります。**

色の環境設定

色の環境設定ダイアログボックスを次の図に示します。



このダイアログボックスでは、Gがさまざまなアイテムに使用する色を変更することができます。**デフォルトの色を使用する**のチェックボックスを選択しない場合は、任意の四角形をクリックして色を変更することができます。オプションは下記の通りです。

フロントパネル ― 新しいVIのフロントパネルの色を選択します。この色の変更は古いVIには適用されません。

ブロックダイアグラム ― 新しいVIのブロックダイアグラムの色を選択します。この色の変更は古いVIには適用されません。

スクロールバー ― スクロールバーの色を選択します。この色は現在開いているVIにのみ適用されます。

強制ドット ― 数値データの強制を示すドットの色を選択します。この色は現在開いているVIにのみ適用されます。

（Windows、UNIX）メニューテキスト ― メニューで使用されるテキストの色を選択します。

（Windows、UNIX）メニューの背景 ― メニューで使用される背景の色を選択します。

点滅の前景色 ― 点滅オブジェクトの前景色を選択します。点滅状態の点滅オブジェクトにのみ適用されます。点滅は、属性ノードでオンになる基本属性です。詳しくは、「第22章 属性ノード」を参照してください。

点滅の背景色 ― 点滅オブジェクトの背景色を選択します。点滅状態の点滅オブジェクトにのみ適用されます。

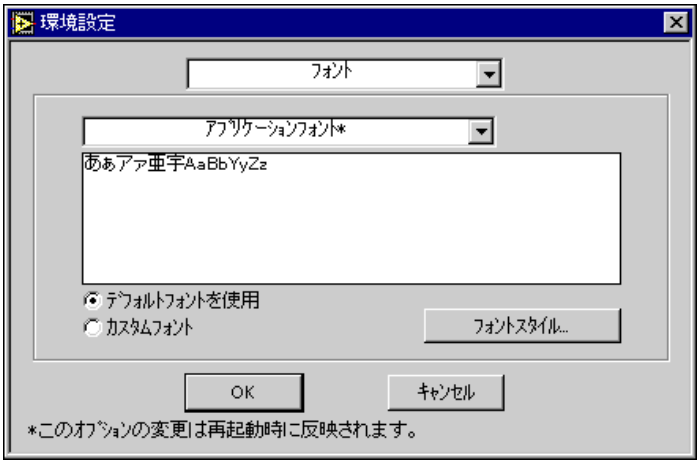
デフォルトの色を使用する ― このダイアログボックス内に表示されるアイテムに対してデフォルト設定の色を使用します。これらの色を1つでも変更したいときは、このオプションを無効にする必要があります。

（UNIX）Provided extra colors ― 216色パレットと125色パレットのいずれかを選択します。大きなパレットを使用すると、256色しかサポートしていないシステムではウィンドウを切り替える際に点滅が目立つようになります。

☞ **注**　色の環境設定ダイアログボックスのオプションの変更は、直ちに有効になります。

フォントの環境設定

フォントの環境設定ダイアログボックスでは、アプリケーションフォント、システムフォント、およびダイアログフォントというあらかじめ定義された3種類のフォントを変更することができます。テキストボックスの上のポップアップメニューから、変更したいフォントのカテゴリーを選択します。**アプリケーションフォント**を選択したときのダイアログボックスを次の図に示します。



Gは、インタフェースの特定の部分に3種類のフォントを使用します。フォントは、次に示すようにプラットフォームに応じてあらかじめ定義されています。

デフォルトフォントを使用 ― 上記で述べたように、それぞれのプラットフォームに応じてGが定義したデフォルトのフォントを使用します。

カスタムフォント ― このチェックボックスは、**フォントスタイル**...オプションを通じてフォントの特性を変更すると自動的にチェックされます。

フォントスタイル... ― フォントの特性を変更するためのダイアログボックスを表示します。

☞ **注**　**フォントの環境設定ダイアログボックスのオプションの変更は、直ちに有効にはなりません。ソフトウェアを再起動する必要があります。**

フォントをあらかじめ定義する方法、およびフォントの移植性については、「第29章 移植性およびローカル化について」の「解像度とフォント」の項を参照してください。

印刷の環境設定

印刷の環境設定ダイアログボックスを次に示します。オプションのなかには特定のプラットフォームでしか使用できないものもありますので注意してください。

（Windows、Macintosh）標準印刷 — VIの印刷データ（フロントパネル、ダイアグラム、アイコン、その他）をフォーマットし、標準の描画コマンドを使用してそれらのデータを印刷します。ファイルに印刷したいとき（プリンタドライバがファイルへの印刷をサポートしている場合）、プリンタがPostScript をサポートしていないとき、あるいはG 言語ソフトウェアでなくプリンタドライバでPostScriptの変換を実行したいときは、このオプションを使用します。

（Windowsのみ）ディザリングを許可 — 描画コマンドを作成する際にプリンタのディザリングを使用します。プリンタディザリングは白黒印刷に適用され、領域を白と黒だけで印刷する代わりにグレースケールを追加します。このオプションは、プリンタのタイプに関係なく使用できます。

PostScript印刷 — VIの印刷データをPostScript（.ps）フォーマットに変換し、PostScript データとしてプリンタに送出します。PostScript プリンタとPostScript プリンタドライバを使用すると、プリントアウトはグラフィックイメージになります。プリンタやプリンタドライバが PostScript 印刷をサポートしていない場合は、このメニュー項目を選択しないでください。

PostScriptレベル2 — 接続されているプリンタが**PostScriptレベル2**をサポートしていることをソフトウェアに知らせます。このボックスが選択されていると、G は **PostScript レベル 2** のコードをプリンタに送出します。このオプションは、プリンタが**PostScript レベル2**をサポートしている場合にのみ選択してください。

（Windows、UNIX）ビットマップ印刷 ― ビットマップを作成してそのページのすべてのデータをビットマップに描画し、そのビットマップをプリンタに送ります。この方法は、ほかの 2 つの方法よりも印刷に時間がかかり、出力の解像度も PostScript 印刷ほど高くありませんが、標準の印刷方法よりもテキストやフォントを正確に再現することができます。ビットマップイメージはカラープリンタではカラーで印刷され、白黒プリンタでは白黒で印刷されます。

（Windowsのみ）カラー/グレースケール印刷 ― このオプションは、プリンタドライバから受け取った深度設定を無視して 8 BPP（8 ビット / ピクセル）のカラーあるいはグレースケールの出力データをプリンタに送ります。ほとんどのプリンタは色の処理能力を正しく通知しますが、レーザープリンタドライバの中には実際に 8 BPP を処理できる場合でもビット深度を 1 として通知するものもあります。

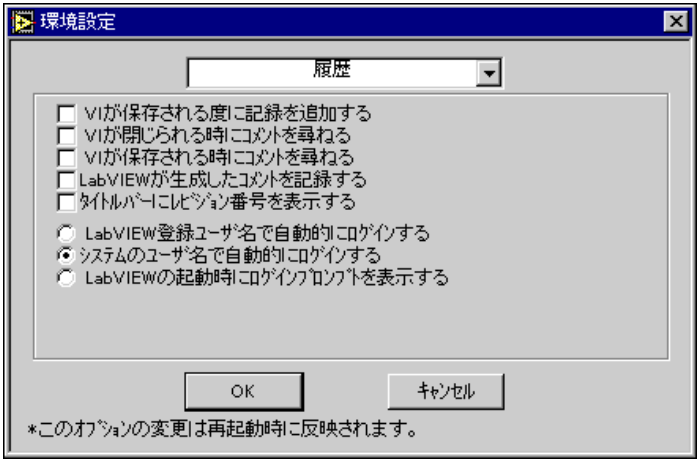
余白 ― プリントアウトの絶対余白は、ミリメートルまたはインチ単位で設定することができます。4 つの余白（上、下、左、右）は、いずれも個別に設定することができます。余白幅は、プリンタの物理的な仕様によって制限されます。プリンタで許容される値よりも小さく余白を設定した場合は、プリンタの許容最小余白が実際の余白幅になります。環境設定ダイアログボックスの余白設定は、新たに作成または変換される VI に適用されます。特定の VI の余白を設定したいときは、**VI 設定**を使用します。

☞ **注**　**印刷の環境設定ダイアログボックスのオプションの変更は、直ちに有効になります。**

履歴の環境設定

「第 27 章　アプリケーションを管理する」で説明しますが、それぞれの VI にはその VI の開発履歴を表示する履歴ウィンドウがあります。履歴の環境設定ダイアログボックスでは、新しい VI の履歴ウィンドウのデフォルト設定を選択することができます。

履歴の環境設定のダイアログボックスを次の図に示します。



履歴ダイアログボックスは、2つのグループに分けられます。最初のグループには、新しいVIの履歴ウィンドウにデータがいつどのように追加されるかを示す5つのオプションがあります。

履歴の環境設定ダイアログボックスにある2つめのオプショングループは、LabVIEWまたはBridgeVIEWへの記録方法に関するものです。このグループは、VIの履歴ウィンドウに入力されるコメントのヘッダに挿入する名前を指定します。

履歴ウィンドウのデータに関するオプションは、**VI設定→文書作成**ダイアログボックスを通してVIに対して**個別に**指定します。オプションに対するアクセス方法についての詳細は、「第6章　VIおよびサブVIをセットアップする」の「文書作成オプション」の項を参照してください。

このダイアログボックスのオプションは下記の通りです。

VIが保存される度に記録を追加する — VIを保存するたびにVIの履歴にデータを追加します。履歴ウィンドウの**コメント**ボックスに**コメント**を入力しなかった場合は、ヘッダのみが履歴に追加されます。ヘッダは、レビジョン番号（このダイアログボックスのさらに下にあるレビジョン番号オプションを選択した場合）、日付と時間、およびVIの名前で構成されます。

VIが保存される時にコメントを尋ねる — 前回の保存以降に変更した内容に関するコメントを表示する履歴ウィンドウを開きます。このオプションは、編集中ではなく変更を終えた後でコメントを入力したい場合に使用すると便利です。このオプションを選択していない状態で**保存**を選択すると、保存が完了するまでVIの履歴を変更することはできなくなります。

VIが閉じられる時にコメントを尋ねる ― 上記のオプションとよく似ていますが、VI を保存するたびにではなく、VI を終了するときにだけコメントのプロンプトを表示します。VI をロードした後で VI を変更した場合は、すでにその変更を保存していても、プロンプトが表示されます。

☞ **注**　**VI の履歴しか変更しなかった場合は、VI を保存する場合も終了する場合もコメントのプロンプトは表示されません。**

LabVIEWが生成したコメントを記録する ― 特定のイベントが発生した場合に、履歴ウィンドウにコメントを入力するようエディタに指示します。コメントを自動入力させるイベントとしては、LabVIEW または BridgeVIEW の新バージョンへの変換、サブ VI の変更、VI の名前またはパスの変更があります。

タイトルバーにレビジョン番号を表示する ― 履歴ウィンドウのヘッダにレビジョン番号を追加します。番号はゼロからスタートし、VI を保存するたびに 1 つずつ増分されます。ただし、VI の履歴のみの変更では番号は増分されません。

ログインオプションは下記の通りです。

（WindowsおよびMacintoshのみ）LabVIEW登録ユーザ名で自動的にログインする ― アプリケーションに登録されたユーザ名を使用します。

LabVIEWの起動時にログインプロンプトを表示する ― アプリケーションの起動時に、ユーザ名を入力するためのプロンプトを表示します。ユーザ名は、**編集→ユーザ名**を選択することによりいつでも変更することができます。プロンプトは、これまでにこのオプションを使用して LabVIEW に入力したすべてのユーザ名を表示します。

☞ **注**　**これは LabVIEW だけの機能です。BridgeVIEW には、編集→ユーザ名を表示しない独自のログインセキュリティシステムがあります。**

前の図で示したプルダウンメニューを使用することで、名前を入力しなくてもWindowsやMacintoshでは別のユーザ名（システムユーザ名またはアプリケーションへの登録名）に、UNIX ではシステムユーザ名に変更することができます。また、名前を入力してリストに表示させることもできます。

（Windows NT、ファイル共有システムを備えたMacintosh、およびUNIX）システムのユーザ名で自動的にログインする ― ユーザ名が定義されていることを前提として、現在のユーザのシステムへのログイン名を使用するようソフトウェアに指示します。

☞ **注**　履歴の環境設定ダイアログボックスのオプションの変更は、直ちに有効になります。

時間と日付の環境設定

時間と日付の環境設定ダイアログボックスを次の図に示します。

このダイアログボックスのオプションは、新しい制御器および表示器のデジタル表示部の時間と日付のデフォルト表示を制御します。「第9章　数値制御器と数値表示器」の「デジタル表示のフォーマットと精度」の項で述べるように、このダイアログボックスのデフォルト設定は個々のデジタル表示ごとに無効にすることができます。

このダイアログボックスのオプションは下記の通りです。

デフォルト日付形式 ― デジタル表示で月、日、年のどれを最初に表示するかを指定します。

デフォルト時間形式 ― デジタル表示で時間を12時間表示にするか24時間表示にするかを指定します。

日付の区切り ― デジタル表示で月、日、年（デフォルト日付形式オプションで指定した順序で）の区切りに使用する文字を指定します。

時間の区切り ― デジタル表示で時間と分の区切りに使用する文字を指定します。

☞ **注**　時間と日付の環境設定ダイアログボックスのオプションの変更は、直ちに有効になります。

その他の環境設定

その他の環境設定ダイアログボックスを次の図に示します。

このダイアログボックスのオプションは下記の通りです。

ヒントラベルを表示する ― ヒントラベルを表示する/しないを切り替えます。

（Windows、Macintosh）ネイティブファイルダイアログを使用する ― オペレーティングシステム固有のファイルダイアログボックスを、(次のパラグラフで説明する一部の機能を除いて)マシンにある他のアプリケーションのファイルダイアログボックスとよく似た形で表示します。このオプションを選択しなかった場合、G 言語ソフトウェアはプラットフォームに依存しない G 固有のファイルダイアログボックスを使用します。G 固有のファイルダイアログボックスには、最近使用したパスのリストの表示や、VI を VI ライブラリに保存する際の手順の簡略化など、いくつかの便利な付加機能があります。

（Windows、Macintosh）ドロップスルークリックを許可する ― マウスで 1 回クリックするだけでアクティブでないウィンドウのオブジェクトをアクティブにし、選択できるようにGを設定します。これにより、マウスをクリックする回数を 2 回から 1 回に減らすことができます（通常は、最初のクリックがウィンドウをアクティブにし、2 回目のクリックがそのウィンドウでのクリックとして機能します)。

ホットメニューを使用する — このオプションを使用すると、左のマウスボタンを押し続けなくてもメニューの中を移動することができます。人間工学に基づいたこの機能は、手の負担を軽減します。

自動定数ラベルを表示する — 自動作成される定数のラベルを作成する/しないを切り替えます。このオプションを設定すると、定数を作成した端子名が自動的にラベルに表示されます。

実行モードでVIを開く — VIを編集モードでなく実行モードで開きます。

起動時にナビゲーションダイアログをスキップ — 共通のダイアログボックスに移動するための初期ウィンドウをスキップして、名前の付いていないVIを開きます。

☞ **注**　**その他の環境設定ダイアログボックスのオプションの変更は、直ちに有効になります。**

サーバ：構成

サーバ：構成ダイアログボックスを次の図に示します。

これらのオプションは、他のアプリケーションがVIサーバにアクセスする際にTCP/IPまたはActiveXのどちらのプロトコルを使用するかを指定します。TCP/IPを選択した場合は、クライアントのアプリケーションがサーバに接続する際に使用する**ポート**番号を指定する必要があります。他のアプリケーションがTCP/IPを使用して接続できるようにした場合は、どのインターネットホストがサーバにアクセスできるかを構成するのが最も好ましい方法です。詳しくは、本章の「サーバ：TCP/IPアクセス」の項を参照してください。VIサーバのActiveXインタフェースについての詳細は、「第21章　VIサーバ」を参照してください。

サーバ：構成では、VIサーバにアクセスするアプリケーションが下記のどのサーバリソースを使用できるようにするかも指定することができます。

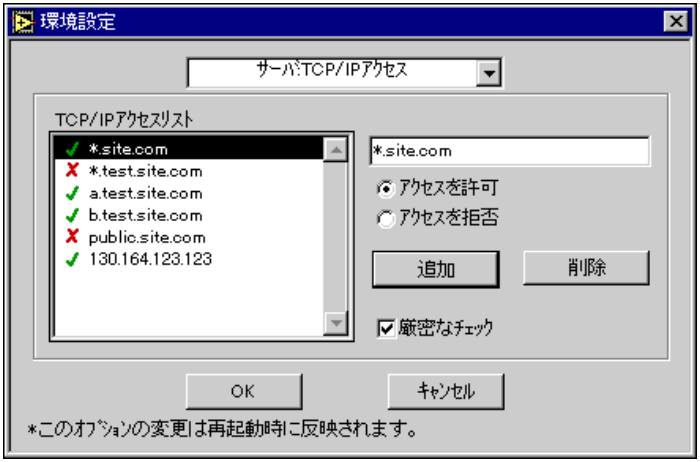
- **VIコールを許可**は、サーバ上でエキスポートしたVIをアプリケーションが呼び出せるようにします。他のアプリケーションがVIにアクセスするのを許可する場合は、エクスポートしたVIの設定ページでエキスポートするVIを指定するのが最も好ましい方法です。詳しくは、本章の「サーバ：エクスポートしたVI」の項を参照してください。
- **VIメソッドとプロパティを許可**は、サーバ上でアプリケーションがVIのプロパティを読み込んだり設定したりするのを許可します。上記と同様、エクスポートしたVI環境設定のページでVIを指定することができます。詳しくは、本章の「サーバ：エクスポートしたVI」の項を参照してください。
- **アプリケーションメソッドとプロパティを許可**は、アプリケーションがサーバのプロパティを読み込んだり設定したりするのを許可します。

サーバ設定のデフォルトは、ActiveXが有効でTCP/IPが無効です。また、**VIコール**は有効ですが、**VIメソッドとプロパティ**と**アプリケーションメソッドとプロパティ**は無効になります。

サーバ：TCP/IPアクセス

リモートのアプリケーションがTCP/IPプロトコルを使用してVIサーバにアクセスできるようにする場合は、サーバにアクセスできるインターネットホストを指定するのが最も好ましい方法です。

サーバ：TCP/IPアクセスダイアログボックスを次の図に示します。



サーバ：TCP/IP アクセスを使用すると、VI サーバにアクセスするクライアントを指定することができます。TCP/IP アクセスリストは、サーバへのアクセス権を持つクライアントとアクセス権を持たないクライアントを示します。項目を変更する場合は、リストで項目を選択して TCP/IP アクセスリストの右のボックスに入力します。クライアントにサーバへのアクセス権を付与する場合は、**アクセスを許可**のラジオボタンをクリックします。アクセス権を拒否する場合は、**アクセスを拒否**のラジオボタンをクリックします。現在選択されている項目の後ろに新しい項目を挿入するには、**追加**ボタンをクリックします。現在選択されている項目を削除するには、**除去**ボタンをクリックします。TCP/IP アクセスリスト内の項目の位置を変更するには、クリックアンドドラッグによってその項目を移動します。アドレスからのアクセスが許可されている項目の横にはチェックマークが表示されます。アドレスからのアクセスが許可されていない項目の横には *X* マークが表示されます。項目の構文が正しくない場合は、マークは表示されません。

クライアントがサーバへの接続を確立しようとすると、サーバは TCP/IP アクセスリスト内の項目をチェックし、そのクライアントにアクセス権があるかないかを判断します。リストでクライアントのアドレスと一致する項目が見つかった場合、サーバはアクセスを許可するか拒否するかをその項目の設定に基づいて決定します。ほかにもクライアントのアドレスと一致する項目がある場合は、前述した項目の設定ではなく、それらの項目の設定に基づいてアクセスを許可するかどうか決定します。たとえば、上に示した図では、リストは（* というワイルドカードを使用して）.test.site.com で終わるすべてのアドレスにはアクセス権が認められていないことを示していますが、a.test.site.com と b.test.site.com にはアクセス権が与えられます。クライアントのアドレスと一致する項目が存在しない場合は、アクセスは拒否されます。ワイルドカードの * およびアドレスが一致する項目のアクセス許可についての詳細は、表 7-1 を参照してください。

130.164.123.123 といったインターネット（IP）アドレスは、www.natinst.com のように 1 つまたは複数のドメイン名を持っている場合があります。ドメイン名からそれに対応する IP アドレスへの変換は、**ネームレゾリューション**と呼ばれます。IP アドレスからドメイン名への変換は、**名前の検索**と呼ばれます。

名前の検索およびネームレゾリューションは、インターネット上のドメインネームシステム（DNS）サーバにアクセスするシステムコールを通じて行われます。システムが DNS サーバにアクセスできない場合やアドレス名が無効な場合は、名前の検索やネームレゾリューションは正しく実行されません。リストの項目に IP アドレスに分解できないドメイン名が含まれていると、ネームレゾリューションエラーが発生します。リストの項目に *.natinst.com のようにドメイン名の一部しか含まれていないときは、

名前の検索エラーが発生し、クライアントのIPアドレスの検索は正しく実行されません。

厳密なチェックは、ネームレゾリューションエラーや検索エラーのためにクライアントのIPアドレスと比較できないアクセスリストの項目を、どのように処理するかを指定します。厳密なチェックを有効にしてある場合は、TCP/IP アクセスリストでアクセス権が付与されていない項目はネームレゾリューションエラーに遭遇するとクライアントのIPアドレスと一致しているものとして処理されます。**厳密なチェック**が無効の場合、アクセスリストの項目はネームレゾリューションエラーに遭遇すると無視されます。

インターネットホストのアドレスを指定するためには、そのドメイン名またはIPアドレスを入力します。インターネットホストのアドレスを指定する際には、ワイルドカード * を使用します。たとえば、domain.com というドメイン内にあるすべてのホストを指定する場合は、*.domain.com と入力します。また、最初の2つのオクテットが130.164 であるサブネットのすべてのホストを指定する場合は、130.164.* と入力します。* だけを入力した場合は、すべてのアドレスと一致するものとみなされます。

次の表は、TCP/IPアクセスリストの項目の例を示したものです。

表 7-1 サーバ：TCP/IP アクセス

アクセス文字列	一致の対象
*	すべてのホスト
test.site.com	test.site.com というドメイン名のホスト
*.site.com	*.site.com というドメイン名で終わるすべてのホスト
130.164.123.123	130.164.123.123 というIPアドレスを持つホスト
130.164.123.*	IP アドレスが 130.164.123 で始まるすべてのホスト

先に示した図では、site.com というドメインにあるホストのうち、test.site.com というドメインにあるホストを除くすべてのホストがサーバへのアクセス権を持っています。さらに、a.test.site.com、b.test.site.com、および130.164.123.123 というホストもサーバへのアクセス権を持っています。ただし、public.site.com というホストは site.com というドメインにあってもアクセス権は与えられていません。

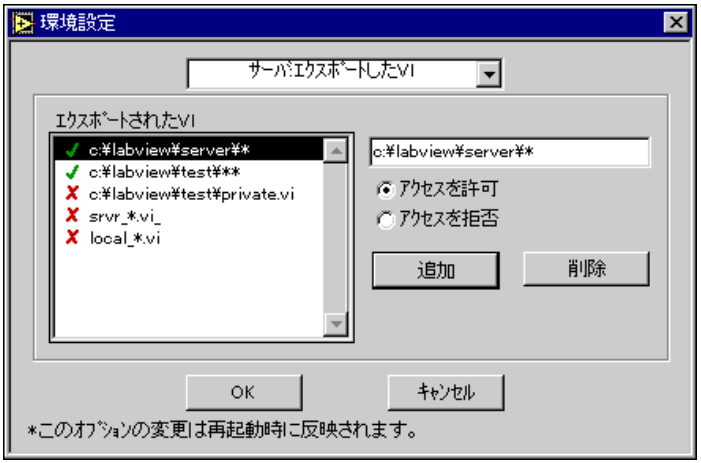
TCP/IPアクセスのデフォルト設定では、サーバマシン上のクライアントにのみアクセス権が付与されます。

☞ **注**　VIサーバをDNSサーバへのアクセス権を持たないシステムで実行する場合は、TCP/IPアクセスリストのドメイン名は使用しないでください。ドメイン名やIPアドレスの解決は正しく実行されず、システムの処理速度が低下します。パフォーマンスの低下を防ぐため、一致頻度の高い項目はTCP/IPアクセスリストの下の方に入力するようにしてください。

サーバ：エクスポートしたVI

リモートのアプリケーションにVIサーバ上のVIへのアクセスを許可する場合は、それらのアプリケーションがアクセスできるVIを指定するのが最も良い方法です。

サーバ：エクスポートしたVIダイアログボックスを次の図に示します。



サーバ：エクスポートしたVIを使用すると、他のアプリケーションがVIサーバを通じてアクセスできるVIを指定することができます。エクスポートしたVIのリストは、エキスポートするVIを指定します。項目を変更する場合は、リストで項目を選択し、エクスポートしたVIリストの右にあるテキストボックスに入力します。リモートのコンピュータがそのVIにアクセスできるようにするかしないようにするかは、**アクセスを許可**または**アクセスを拒否**ボタンをクリックして指定します。現在選択されている項目の下に新しい項目を追加するには、**追加**ボタンをクリックします。現在選択されている項目を削除するには、**削除**ボタンをクリックします。エクスポートしたVIリスト内の項目の位置を変更するには、クリックアンドドラッグしてその項目を移動します。VIへのアクセスが許可されている項目の横にはチェックマークが表示されます。VIへのアクセスが許可されていない項目の横にはXマークが表示されます。項目の構文が正しくない場合は、マークは表示されません。

リストの各項目は、VI の名前または VI のパスを記述し、ワイルドカード文字を含む場合もあります。パスのセパレータを含む項目は VI のパスと比較されますが、パスのセパレータを含まない項目は VI の名前とだけ比較されます。リモートのクライアントが VI にアクセスしようとすると、サーバはエクスポートした VI リストをチェックし、要求された VI へのアクセスを許可すべきかどうかを判断します。要求された VI と一致する項目がリスト内に見つかると、サーバはその VI へのアクセスを許可するかどうかを項目の設定内容に基づいて決定します。ほかにも VI と一致する項目がある場合は、前述の項目の設定ではなくその項目の設定に基づいてアクセス権の有無を決定します。リストに要求された VI と一致する VI が存在しない場合は、その VI へのアクセスは拒否されます。

エクスポートした VI のリストでは、ワイルドカードを使用して 1 つの項目で複数の VI を指定することができます。使用できるワイルドカードの種類は下記の通りです。

「?」　パスセパレータ以外の 1 個のあらゆる文字と一致するものとみなされます。

「*」　パスセパレータ以外の 0 個または 1 個以上のあらゆる文字と一致するものとみなされます。

「**」　パスセパレータも含めて 0 個以上のあらゆる文字と一致するものとみなされます。

一致する VI 名にワイルドカード文字を入れて指定したいときは、「¥」**（Macintosh および UNIX）**または「\」**（Windows）**を使用してワイルドカード文字をエスケープする、つまりその文字がワイルドカードでないことをソフトウェアに示す必要があります。

次の表は、エクスポートした VI リストの項目の例を示したものです。これらの例では、UNIX のパスセパレータを使用しています。

表 7-2 サーバ：エクスポートした VI リストの項目

VI のアクセス文字列	一致の対象
*	すべての VI
/usr/labview/*	/usr/labriew/ というディレクトリ内のすべての VI。
/usr/labview/**	/usr/labview/ というディレクトリおよびそのすべてのサブディレクトリ内にあるすべての VI。
test.vi	Test.VI という名前のすべての VI。
OK¥?	OK? という名前のすべての VI。

前の図では、c:¥labview¥server というディレクトリに入っているすべてのVIがエキスポートされます。同様に、c:¥labview¥test というディレクトリおよびそのすべてのサブディレクトリに入っているc:¥labview¥test¥private.vi以外のすべてのVIもエキスポートされます。さらに、srvr_ で始まり .vi で終わるすべての VI もエキスポートされます。local_ で始まり .vi で終わるVIは、それらがc:¥labview¥serverディレクトリに入っているVIであってもエキスポートされません。

エクスポートしたVIのデフォルト設定では、すべてのVIへのアクセスが許可されます。

環境設定の保存方法

通常は、必要なことがらは環境設定ダイアログボックスが処理してくれるため、環境設定のデータを手作業で編集したり、データの正確なフォーマットを知る必要はありません。環境設定は、次で述べるように各プラットフォームごとに別々に保存されます。

（Windows）LabVIEWユーザは、環境設定のデータをLabVIEWディレクトリのLabVIEW.INIファイルに保存します。ファイルのフォーマットは、他の .INI ファイル（WIN.INI ファイルなど）とよく似ています。このファイルは、[LabVIEW] というセクションマーカで始まります。

その後には、offscreenUpdates=True といったさまざまな変数とその値が続きます。LABVIEW.INI で構成値が定義されていない場合は、LabVIEW はWIN.INI ファイルに [LabVIEW] セクションが存在するかどうかを調べ、存在する場合はそのファイルの構成値をチェックします。

BridgeVIEW ユーザは、環境設定のデータを BridgeVIEW ディレクトリのBridgeVIEW.INIファイルに保存します。このファイルのフォーマットは、他の .INI ファイルとよく似ています。このファイルは、[BridgeVIEW] というセクションマーカで始まります。

☞ **注**　**WIN.INI ファイルを参照するのは、Windows 3.1 の場合のみです。BridgeVIEW ユーザにはこのデータが適用されません。**

環境設定ファイルは、LabVIEW またはBridgeVIEWの起動時にコマンドラインで指定することもできます。たとえば、labview.ini の代わりにlvrcという名前のファイルを使用する場合は、labview -pref lvrcとタイプします。

（Macintosh）デフォルトでは、LabVIEWはLabVIEW Preferencesと呼ばれるシステムフォルダの中の初期設定フォルダにテキストファイルを作成します。

LabVIEW の環境設定ファイルは、必要に応じて LabVIEW フォルダにコピーすることができます。LabVIEW は、起動時に LabVIEW フォルダで環境設定ファイルを探します。見つからない場合は初期設定フォルダを探し、そこでも見つからない場合は初期設定フォルダに新しくファイルを作成します。デスクトップと LabVIEW フォルダの間で環境設定ファイルを移動することにより、複数のユーザが使用できるように複数の環境設定を作成することができます。

（UNIX）環境設定データは、`.labviewrc` という名前のホームディレクトリのテキストファイルに保存されます。環境設定ダイアログボックスからパラメータを変更すると、そのデータはこのファイルに書き込まれます。以下で、保存フォーマットおよび環境設定データを探す場合のルールについてさらに詳しく説明します。

環境設定のエントリは、**環境設定の名前**、コロン、および値で構成されます。環境設定の名前は実行可能名で、後ろにピリオド（.）とトークンが続きます。LabVIEW が環境設定の名前を検索する際には、大文字と小文字は区別されます。環境設定の値は、シングルまたはダブルのクォーテーションマークで囲むことができます。たとえば、デフォルトである 2 倍の精度、および特定のディレクトリを再帰的に検索する検索パスを使用する場合は、環境設定エントリを次のように指定します。

```
labview.defPrecision : double

labview.viSearchPath : "/usr/lib/labview/*"
```

LabVIEW は、アプリケーションのディレクトリで `labviewrc` という名前のファイルでも環境設定を検索します。たとえば、LabVIEW ファイルを `/opt/labview` にインストールした場合は、`/opt/labriew/labviewrc` とホームディレクトリの `.labviewrc` ファイルから環境設定を読み込みます。

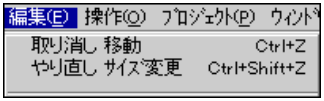
`.labview` ファイルのデータは、`labviewrc` ファイルの同じ項目のデータよりも優先します。このグローバルな環境設定ファイルは、すべてのユーザが共通に使用する VI の検索パスなどのデータを保存するために使用することができます。

環境設定ファイルは、LabVIEW の起動時にコマンドラインで指定することもできます。たとえば、`lvrc` というファイルを使用したい場合は、`[labview -pref lvrc]` とタイプします。

注意　**この場合でも、`.labviewrc` ファイルが読み込まれます。また、この例では環境設定ダイアログボックスで変更した内容が `lvrc` ファイルに書き込まれる点にも注意してください。**

取り消し

編集→取り消しや**編集→やり直し**を使用すると、直前に実行した動作を取り消したり、取り消した動作を再実行することができます。**取り消し**のキーコマンドは<Ctrl-Z>（Macintoshでは<Cmd-Z>）、**やり直し**のキーコマンドは<Ctrl-Shift-Z>（Macintoshでは<Cmd-Shift-Z>）です。**編集**メニューのオプションである**取り消し**と**やり直し**には、取り消しや再実行の対象となる動作に関する簡単な（1語または2語の）説明が表示されます。次の図では、**サイズの変更の取り消し**を選択するとその項目が元のサイズに戻ります。



取り消せる動作あるいは再実行できる動作の数は、**環境設定→ブロックダイアグラム→最大取り消し回数**で設定します。動作の数をゼロに設定した場合は、取り消しとやり直しは使用不能になります。VIの取り消し可能な動作の数の設定についての詳細は、本章の「環境を設定する」の「ブロックダイアグラムの環境設定」の項を参照してください。

各VIにはそれぞれの取り消しのインスタンスが記録されるため、**最大取り消し回数**で入力した値はそのVIにだけ適用されます。この値を小さい値に設定することにより、メモリの使用量を節約できます。

VIの発呼者は、VIがそのインタフェースを変更すると自動的に更新します。その場合、取り消しに関する発呼者の情報は破棄されます。このような動作が発生するのは、下記のいずれかの操作を実行した場合です。

- VIのコネクタペーンのパターンを変更したとき、またはコネクタペーンの制御器のデータタイプを変更したとき。
- Type defの外観を変更したとき（厳密である場合のみ）、データタイプまたは厳密度を変更したとき、またはカスタム制御器のType defを変更したとき。
- グローバルVIの制御器のデータタイプまたは名前を変更したとき、またはグローバルVIの制御器を追加または削除したとき。

制御器パレットおよび関数パレットをカスタマイズする

Gには、下記のいずれかの方法で制御器パレットや関数パレットをカスタマイズするための強力なツールのセットが用意されています。

- user.libまたはinstr.libディレクトリに保存して、パレットに独自のVIや制御器を追加する。
- **編集→制御器と関数パレットの編集**オプションを使用してパレットに独自のVIや制御器を追加する
- 特定のディレクトリのファイルを追加または削除するとパレットが自動的に変更されるようにする
- 組み込まれたパレットを配置し直し、使用頻度の高い関数にさらにアクセスし易くする
- あるユーザに対しては一部の関数を表示せず、別のユーザに対してはすべての機能を表示するというように、ユーザごとに異なるビューを設定する
- WindowsとMacintoshでは、フロントパネルがアクティブウィンドウのとき**関数**パレットは表示されず、ブロックダイアグラムがアクティブウィンドウのときには**制御器**パレットが表示されません。同時にすべてのパレットが表示されるようにするには、環境設定ファイルを編集してShowAllPalettes = Trueという行を追加します。

user.libおよびinstr.libへVIおよび制御器を追加する

制御器パレットや**関数**パレットに新しい項目を追加する最も簡単な方法は、それらをuser.libディレクトリの内部に保存する方法です。Gを再起動すると、**関数**パレットの**ユーザライブラリ**サブパレットには、各ディレクトリのサブパレット、user.libの.libまたは.mnuファイルおよびuser.libの各ファイルへのエントリが含まれます。

関数パレットの**計測器ドライバ**パレットは、instr.libと対応しています。このディレクトリ内に**計測器ドライバ**を入れて、パレットからアクセスし易くすることもできます。

☞ **注**　vi.libディレクトリには、ソフトウェアをアップグレードしたときに上書きされるナショナルインスツルメンツ提供のファイルが含まれているため、vi.libにはユーザファイルを保存しないようにしてください。

パレットメニューの設定と変更

制御器パレットと**関数**パレットのデータは、アプリケーションディレクトリのmenusディレクトリに保存されます。menusディレクトリには、作成またはインストールした各パレットメニューに対応するディレクトリが含まれます。ネットワーク上のコンピュータから実行する場合は、個々のmenusディレクトリを各ユーザごとに定義する必要がある場合があります。そのような場合は、環境設定ファイルのmenusDirを別のパス（すなわちユーザの環境設定ファイルごとに固有のパス）に設定する行を追加することができます。

この方法を使用すると、他の人へのパレットメニューの転送が容易になります。パレットメニューのコピーを他の人に渡す場合は、menusディレクトリからパレットメニューのディレクトリのコピーを渡します。そのコピーを相手のmenusディレクトリに書き込んでシステムを再起動すると、相手はそのパレットメニューを使用できるようになります。

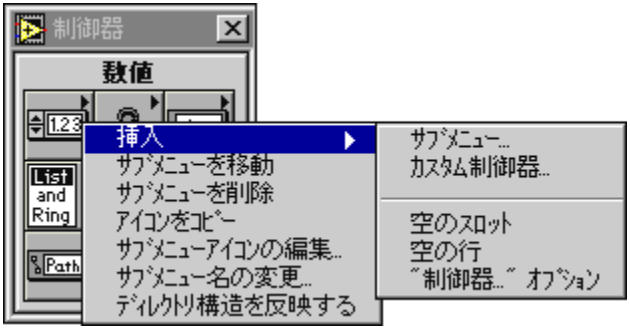
別のパレットメニューに切り替える場合は、**編集→パレットセットを選択**を選択したのち、**メニュー選択**リングから見たいパレットメニューを選択します。

独自のパレットメニューの作成方法については、次の「パレットエディタ」の項を参照してください。パレットメニューの構成方法については、本章の「パレットメニューの環境」の項を参照してください。

パレットエディタ

制御器パレットや**関数**パレットのレイアウトや内容をさらに綿密に制御したいときは、**編集→パレットセットを選択**オプションを使用します。このオプションを選択するとパレットエディタが起動し、パレット編集ダイアログボックスが表示されます。

このエディタでは、オブジェクトを別の場所にドラッグしてパレットの内容を変更することができます。オブジェクトを削除、カスタマイズ、あるいは挿入したいときは、次の図で示すようにパレットをポップアップするか、またはサブパレット内のオブジェクトをポップアップします。



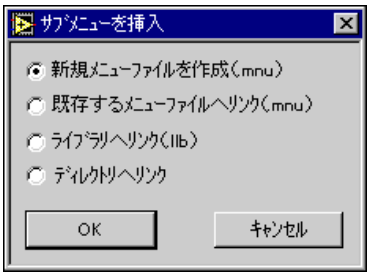
ポップアップメニューのオプションを使用すると、**ユーザライブラリ**メニュー（user.libに対応）または**計測器I/O**メニュー（instr.libに対応）内のあらゆるものを変更することができます。最上位の**制御器**パレットや**関数**パレット、あるいはあらかじめ定義されているその他のメニューを編集したいときは、最初にパレット編集ダイアログボックスの**メニュー選択**リングから**新しい設定...**を選択して新しいパレットメニューを作成する必要があります。新しい設定を選択したあとでデフォルトのメニューセットの一部であるメニューに対して行った変更は、menusディレクトリ内のパレットメニューのディレクトリにコピーするまでは有効にはなりません。組み込まれているパレットにはこのように保護機能が用意されているため、ユーザはデフォルトのパレットメニューを壊さずにパレットを試してみることができます。

サブパレットの新しい行または列に新しいオブジェクトを追加したいときは、サブパレットの右端または下側の容量をポップアップします。あるいは、オブジェクトをパレットの右または下のスペースにドラッグして新しい行または列を作成することもできます。

サブパレットを作成する

パレットを追加する際には、パレットを新しい場所に移動したり、サブパレットのアイコンを編集したり、パレットエディタを使用してパレットの名前を変更することができます。

パレットを全く最初から作成したり、user.libやvi.libに入っていないパレットを作成したいときは、パレットエディタのポップアップメニューで**挿入→サブメニュー...**を選択します。このオプションを選択すると、次のようなダイアログボックスが表示されます。



サブメニューのデータは、VI ライブラリまたは .mnu ファイルに保存することができます。.LLB ファイルまたはメニューファイルには、**関数**パレットと**制御器**パレットを 1 つずつ保存することができます。

新規メニューファイルを作成は、新規の空のパレットを挿入する場合に選択します。パレットの名前をつけ、パレットをファイルに保存するよう指示するプロンプトが表示されます。ファイル名には、それがパレットメニューであることがわかるようにできるだけ .mnu という拡張子を付けるようにしてください。また、.mnu ファイルはなるべくなら menus ディレクトリ内のパレットメニューのディレクトリか、またはそのメニューに関係の深い制御器や VI が入っているのと同じディレクトリに保存するようにしてください。

既存のメニューファイルへリンクは、その制御器パレットまたは関数パレットのファイルに追加したい既存のパレットがある場合に選択します。

ディレクトリへリンクは、ディレクトリ内のすべてのファイルを対象とした項目を含むパレットを作成したいときに選択します。このオプションを選択すると、そのディレクトリ内のそれぞれのサブディレクトリ、VI ディレクトリ、および .mnu ファイルに対して再帰的にサブパレットが作成されます。これらのパレットは、選択したディレクトリに新しいファイルを追加したり選択したディレクトリからファイルを削除すると、自動的に更新されます。サブパレットの自動更新のオン／オフは、ポップアップメニューの**ディレクトリ構造を反映する**オプションを選択することによって切り替えます。このメニュー項目は、それぞれのサブディレクトリ内にパレットメニューファイルを作成します。

ライブラリへリンクは、VI ライブラリの一部を構成する**制御器**パレットや**関数**パレットにリンクしたいときに選択します。上記の設定と同様、ライブラリにファイルを追加するとライブラリのパレットは自動的に更新されます。

パレットの中では、VI、関数、およびサブパレットを自由に組み合わせることができる点に注目してください。また、パレットには別の場所にある VI を入れることもできます。

サブパレットを移動する

サブパレットはクリックすると開いてしまうため、ドラッグして移動することはできません。サブパレットを移動するには、そのポップアップメニューから**サブメニューを移動**の項目を選択します。<Shift>キーを押しながらサブパレットをクリックするショートカットを使用すると、サブパレットを開かずにドラッグすることができます。

パレットメニューの環境

.mnu ファイルと VI ライブラリはいずれもバイナリファイルで、**制御器**パレットと**関数**パレットを 1 つずつ入れることができます。また、どちらのタイプのファイルにも**制御器**パレットと**関数**パレットのアイコンが入っています。作成する各サブメニューは、それぞれ別々のファイルに保存する必要があります。

パレットメニューを選択すると、G は menus ディレクトリで 1 つのディレクトリを探します。G は、最上位の**制御器**パレットと**関数**パレットをそのディレクトリの root.mnu ファイルから作成します。最上位のパレットはその後、そのパレット内のそれぞれのサブパレットの .mnu ファイルや VI ライブラリファイルとリンクされます。

ディレクトリにリンクするためには、最初にそのディレクトリを検索してそこに dir.mnu ファイルが存在するかどうかをチェックします。dir.mnu ファイルが見つかった場合は、その .mnu ファイルがそのディレクトリのサブパレットとして使用されます。dir.mnu ファイルが見つからなかった場合は、アプリケーションがディレクトリの内容に基づいて dir.mnu ファイルを作成します。各 VI（または制御器）ごとにエントリが作成されます。アプリケーションは、各サブディレクトリ、.mnu ファイル、または VI ライブラリごとにサブパレットを作成します。

VI ライブラリにファイルを追加したり、VI ライブラリからファイルを削除すると、G は自動的にパレットの内容を更新します。あるいは、ディレクトリの内容に基づいて更新するよう .mnu ファイルを設定することもできます。この設定を変更するためには、サブパレットアイコンのポップアップメニューから**ディレクトリ構造を反映する**を選択します。.mnu ファイルを反映したいディレクトリに保存しておくのも 1 つの賢明な方法ですが、実際には別のディレクトリを選択することも可能です。つまり、特定のディレクトリの内容を反映させるだけでなく、同じパレットに別のディレクトリに入っている VI や制御器を追加することもできます。また、ポップアップメニューから**項目を削除**を選択してパレットから VI を削除することもできます。

第II部

フロントパネルオブジェクト

第II部では、フロントパネルのオブジェクト、制御器および表示器、およびActiveXの制御器について説明します。

「第II部 フロントパネルオブジェクト」は、下記の章で構成されています。

「第8章 フロントパネルオブジェクトの概要」では、フロントパネルと、その2つのコンポーネントである制御器と表示器について説明します。また、他のプログラムからグラフィックを取り込んで制御器で使用する方法についても説明します。

「第9章 数値制御器と数値表示器」では、さまざまなスタイルの数値制御器や数値表示器の作成、操作、および構成方法について説明します。

「第10章 ブール制御器とブール表示器」では、ブール制御器やブール表示器の作成、操作、および構成方法について説明します。

「第11章 文字列制御器と文字列表示器」では、文字列制御器や文字列表示器、およびテーブルの使用方法について説明します。これらのオブジェクトには、**制御器→文字列と表**パレットを使用してアクセスできます。

「第12章 パスとRefnumの制御器および表示器」では、**制御器→パスとRefnum**パレットにより呼び出されるファイルパスの制御器およびrefnumsの使用方法について説明します。

「第13章 リストとリングの制御器および表示器」では、**制御器→リストとリング**パレットにより呼び出されるリストボックスとリングの制御器および表示器について説明します。

「第14章 配列とクラスタの制御器 および表示器」では、配列とクラスタの使用方法について説明します。配列やクラスタには、**制御器→配列とクラスタ**パレットによりアクセスします。

「第15章 グラフとチャートの制御器および表示器」では、**制御器→グラフ**パレットでのグラフやチャートの表示器の作成および使用方法について説明します。

「第16章　ActiveX制御器」では、Gベースのソフトウェアと他のアプリケーションとの対話能力を高めるActiveXコンテナの機能について説明します。

8

フロントパネルオブジェクトの概要

この章では、フロントパネル、およびそのコンポーネントである制御器と表示器について説明します。また、他のプログラムから制御器で使用するために図形をインポートする方法についても説明します。

フロントパネルを構築する

フロントパネルの制御器と表示器は、VI の対話方式の入力端子と出力端子です。制御器は VI にデータを渡すために使用し、表示器は VI が生成したデータを表示します。この項では、すべての制御器および表示器に共通のいくつかの編集オプションについて説明します。

フロントパネルの**制御器**パレットを次に示します。

パレットのいずれかのボックスにカーソルを合わせると、パレットウィンドウの上の欄に制御器グループの名前が表示されます。

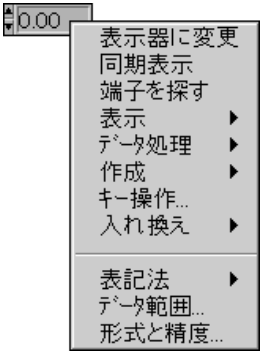
パレットに表示される制御器のタイプは下記の通りです（左から右、上から下の順）。

- **数値制御器** — 数量の入力および表示に使用します。
- **ブール制御器** — TRUE/FALSE の値の入力および表示に使用します。
- **文字列と表の制御器** — テキストの入力および表示に使用します。
- **リストとリングの制御器** — オプションリストの表示、およびオプションリストからの選択に使用します。

- **配列とクラスタの制御器** — データをグループ化します。
- **グラフ制御器** — 数値データをチャートやグラフで表します。
- **パスと Refnum の制御器** — ファイルのパスの入力や表示に使用したり、VIから別のVIにRefnumを渡すために使用します。
- **装飾制御器** — 図形を追加してフロントパネルをカスタマイズするために使用します。これらのオブジェクトは単に装飾を目的としたもので、データは表示しません。
- **ユーザ制御器** — カスタマイズした制御器を`user.lib`ファイルに自動的に保存します。
- ActiveX — ActiveXに対するサポートを強化します。これらのオブジェクトにはActiveXコンテナやActiveXバリアントが含まれます。
- **制御器を選択** ... — 独自にデザインしたカスタム制御器を選択するために使用します。

フロントパネルの制御器と表示器のオプション

VI の編集中にフロントパネルの制御器または表示器をポップアップすると、次の図に示すようなメニューが表示されます。ポップアップメニューの線より上に表示されるオプションは、制御器および表示器すべてに共通です。線よりも下に表示される（**表記法**、**データ範囲**...、**形式と精度**...など）オプションは、オブジェクトによって変わります。



制御器パレットのオブジェクトは、標準的な使用目的に応じて制御器または表示器として構成されます。たとえば、**ブール**パレットからトグルスイッチ（切り替えスイッチ）を選択すると、フロントパネル上に制御器として表示されます。これは、トグルスイッチが通常は入力装置として使用されるためです。また、LEDを選択した場合は、フロントパネル上に表示器として表示されます。これは、LEDが通常は出力装置として使用されるためです。ただし、どの制御器および表示器も、オブジェクトのポップアップメニューで**制御器に変更**あるいは**表示器に変更**コマンドを選択することによ

り、制御器を表示器に、あるいは表示器を制御器に構成し直すことができます。**数値**パレットにはデジタル制御器とデジタル表示器の両方が用意されていますが、その理由はこの両者がいずれも頻繁に使用されるためです。**文字列**パレットにも、文字列制御器と文字列表示器が用意されています。

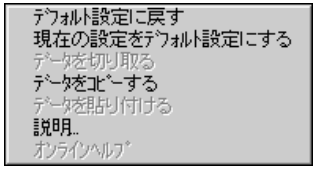
制御器および表示器のポップアップメニューの**端子を探す**項目は、その制御器または表示器のブロックダイアグラム端子をハイライトで表示します。この項目は、複雑なブロックダイアグラム上で特定のオブジェクトを見分けるのに便利です。

作成→属性ノードを選択すると、オブジェクトの属性ノードを作成します。属性ノードは、オブジェクトのさまざまなプロパティをプログラム上で制御するために使用します。

表示のサブメニューは、ネームラベルなどの表示をするかしないかを選択できる制御器部品のリストを表示します。

VIの編集中は、制御器のポップアップメニューに**データ処理**というサブメニューが表示されます。このメニューの項目を使用すると、制御器の内容の切り取り、コピー、貼り付けのほか、制御器をデフォルト値に設定したり、制御器の現在の値をその制御器のデフォルト値として設定すること、さらに制御器の記述を読み込んだりや変更することができます（VIの編集中は、データを表示器に貼り付けることが可能です）。複雑な制御器のなかには、これ以外のオプションを持つものもあります。たとえば、配列では値の範囲をコピーするオプションや、配列の最後の要素を表示するためのオプションがあります。

次の図は、数値制御器の編集モードの**データ処理**サブメニューを示したものです。VIの実行中に使用できるのは、制御器のポップアップメニューのこのサブメニューのみです。



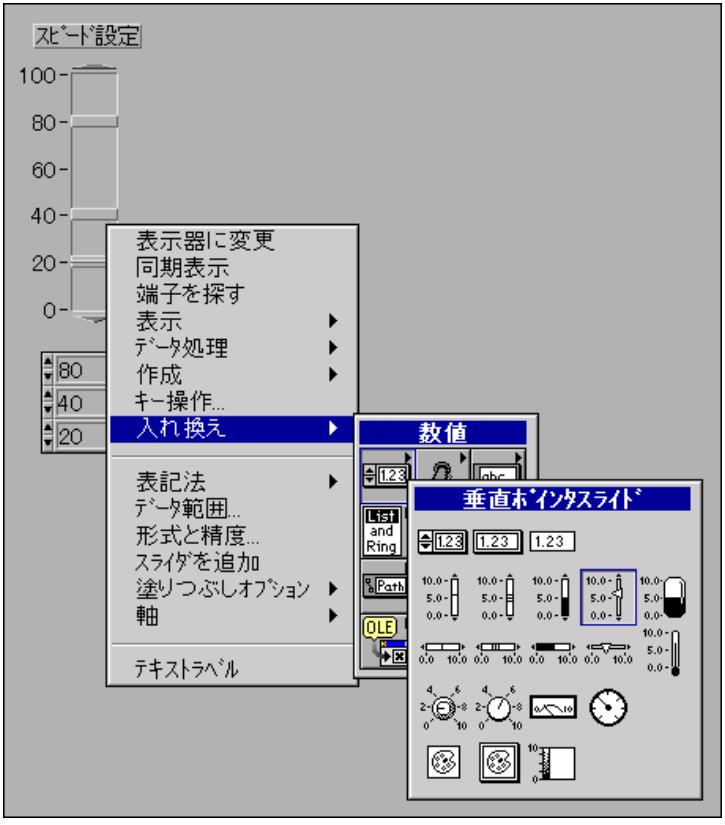
VIの実行中に制御器をポップアップして変更できるのは、制御器の値だけです。デフォルト値や記述など制御器のほとんどの特性は、VIの実行中は変更することはできません。

制御器を差し替える

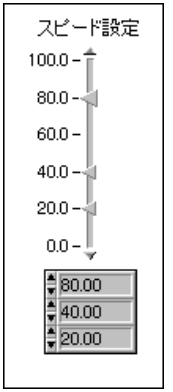
オブジェクトのポップアップメニューにある**入れ換え**の項目には、フロントパネル上の現在の制御器または表示器と差し替えたい制御器または表示器を選択するための**制御器**パレットが表示されます。

別のスタイルの制御器に変更する際に、それまでその制御器に対して選択してきた編集オプションを無効にしたくないときは、**入れ換え**を使用します。ポップアップメニューから**入れ換え**を選択すると、制御器の名前、記述、デフォルト値、データの流れの方向（制御器または表示器）、色、サイズなど、もとの制御器に関する情報が可能な限り保持されます。ただし、新しい制御器ではその制御器固有のデータタイプが使用されます。制御器の端子やローカル変数からのワイヤは、ブロックダイアグラムに接続されたままになります。

差し替える制御器と差し替えられる制御器が似ているほど、多くの特性を保持することができます。次の図では、スライドを別のスタイルのスライドに差し替える例を示します。



結果は次のようになります。



新しいスライドの高さ、スケール、値、名前、記述などは、もとのスライドと同じです。

仮にスライドをポップアップして文字列と差し替えた場合は、スライドと文字列に共通する項目が少ないため、名前、記述、データの流れの方向だけしか保持されません。

制御器を差し替えるもう 1 つの方法として、制御器をクリップボードにコピーする方法があります。この方法では、**入れ換え**のポップアップメニューは使用せず、もとの制御器の特性も保持されませんが、ブロックダイアグラムや VI の接続は保持されます。まず、新しい制御器をクリップボードにコピーします。次に、位置決めツールで差し替えたい古い制御器を選択し、さらに**編集→貼り付け**を選択します。古い制御器が破棄され、新しい制御器に差し替えられます。

制御器のキー操作オプション

フロントパネルのすべての制御器には、**キー操作...** の項目があります。この項目は、キーボードのキーコンビネーションと制御器とを関連付けるために使用します。VI の実行中にユーザがこのキーコンビネーションを入力すると、G はユーザがその制御器をクリックしたときと同じように動作します。関連付けられた制御器が操作の対象になります。制御器がテキスト制御器の場合には、その制御器内のテキストがすべてハイライトで表示され、編集可能な状態になります。また、制御器がブール制御器の場合には、ボタンの状態が切り替わります。

☞ **注**　**表示器にはデータを入力できないため、キー操作...の項目が使用できません。**

また、**キー操作...**は、実行中にユーザが制御器から制御器へと次々と移動するとき、特定の制御器を含めるかどうかを指定するのにも使用できます。

キー操作...を使用すると、ファンクションキーをパネルの動作を制御するさまざまなボタンと関連付けることができます。これを使用してVIに対してダイアログボックスと同じように動作するデフォルトボタンを定義することにより、<Enter>キー**（Windows および HP-UX）**または<Return>キー**（Macintosh および Sun）**を押す操作には、デフォルトボタンをクリックするのと同じ意味を持たせることができます。

ダイアログボックスボタンに <Enter> キーまたは <Return> キーを連結させると、次の図に示すように、ボタンの表示には自動的に太い枠が付けられます。

<Enter>または<Return>キーについて理解しておく必要のある重要な点は、以下の二点です。

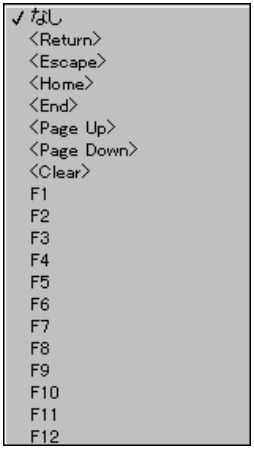
1. 制御器と<Enter>または<Return>キーを連結すると、フロントパネルの制御器はいずれも改行コマンドを受け取れなくなります。そのため、フロントパネル上の文字列はすべて1行で表記されます。
2. 制御器から制御器に移動する場合、ボタンは特別な方法で処理されます。<Enter>または<Return>キーが別の制御器のキー操作設定として定義されている場合でも、あるボタンに移動して <Enter> または <Return>キーを押すとそのボタンの状態が切り替わります。

同じキーコンビネーションを同じフロントパネルの複数の制御器に割り当てることはできません。

キー操作...オプションを選択すると、次のようなダイアログボックスが表示されます。

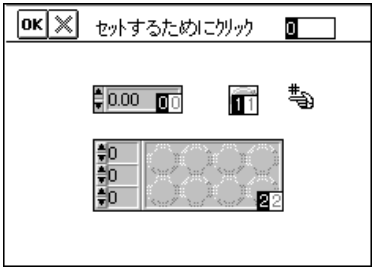
制御器に割り当てるキーは、このダイアログボックスの左上のリングを使用して選択します。キーは1つのキーで指定することも、または<Shift>キーなどの修正キーと組み合わせて指定することもできます。修正キーは、<Shift>キー以外にも<Alt>**（Windows および HP-UX）**、<command>**（Macintosh）**、または<meta>**（Sun）**キーといったメニューキーを選択することができます。このリングのオプションは、次の図に示す通りです。

キー操作ダイアログボックスの右上には、現在関連付けられているキーを表示する現在のキー定義というリストがあります（前の図を参照）。



パネル順序オプション

フロントパネル上の制御器や表示器には、フロントパネル上の位置とは無関係に、パネル順序と呼ばれる論理的な順序付けが存在します。フロントパネル上で最初に作成した制御器または表示器がエレメント0、2番目の制御器または表示器がエレメント1、などとなります。制御器や表示器を削除すると、パネル順序は自動的に修正されます。パネル順序を変更する場合は、**編集→パネル順序...**を選択します。フロントパネルの外観が変更されますが、これは次の図に示すようにパネル順序の編集モードになったためです。



制御器および表示器の上の白いボックスは、パネル順序内のそれらの現在の位置を示します。黒いボックスは、パネル順序における制御器または表示器の新しい位置を示します。パネル順序カーソルで要素をクリックすると、パネル順序内における要素の位置がツールバーに表示されている番号に設定されます。この番号を変更する場合は、ツールバーで新しい番号を入力します。パネル順序を正しく設定した後、**入力**ボタンをクリックして設定を確定し、パネル順序の編集モードを終了します。古いパネル順序に戻してパネル順序の編集モードを終了したいときは、X ボタンをクリックします。

X ボタン

パネル順序は、フロントパネルのロギングで生成されるデータログファイルのレコードに記録される制御器や表示器の順序を決定します。

ダイアログボックスの制御器をカスタマイズする

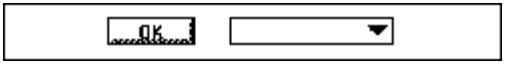
ダイアログリング

ダイアログボックスの制御器には、ダイアログリング（数値）、ダイアログボタン（ブール）、ダイアログチェックボックス、およびダイアログラジオボタンがあります。

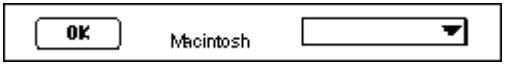
ダイアログリング

ダイアログの制御器は、使用するプラットフォームによってその形状が異なります。どの制御器も、プラットフォーム固有の色と形で表示されます。

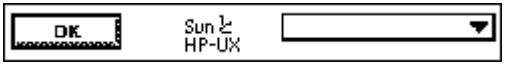
（Windows）ダイアログボタンは、立体的な四角いグレーのボタンとして表示されます。ダイアログリングは、次の図に示すような白黒の二次元の長方形として表示されます。



（Macintosh）ダイアログボタンは、平面的な白黒の楕円として表示されます。ダイアログリングは、次の図に示すように影の付いた白黒の長方形として表示されます。



（UNIX）ダイアログボタンは、立体的なグレーの四角いボタンとして表示されます。ダイアログリングは、次の図に示すように影の付いた平面的な白黒の長方形として表示されます。ダイアログボタンやダイアログリングに色を付けることはできません。



これらの制御器は形状が変化するため、G を実行するコンピュータに応じた制御器を使用して VI を作成することができます。これらの制御器を、チェックボックスとラジオボタン、ブール、単純な数値制御器、単純な文字列、ダイアログのフォントと合わせて使用することにより、VI を使用するコンピュータに合わせて形状が変化する VI を作成することができます。**VI 設定**オプションを使用してメニューバーやスクロールバー、および制御器が表示されないようにすると、そのコンピュータの標準のダイアログボックスとよく似た形状の VI を作成することができます。

インポートしたグラフィックスを使用して制御器をカスタマイズする

他のプログラムからグラフィックスをインポートし、背景の画像、リング制御器の項目、あるいは他の制御器の一部として使用することができます。これらの制御器でのグラフィックスの使用方法についての詳細は、「第 13 章　リストとリングの制御器および表示器」、および「第 24 章　カスタム制御器と Type Def」を参照してください。

画像を使用するためには、まず最初に BridgeVIEW または LabVIEW で使用するクリップボードに画像をロードする必要があります。これには、次に示すようにプラットフォームによって 1 つまたは 2 つの方法があります。

（Windows）ペイントプログラムからWindowsのクリップボードに画像を直接コピーしてG言語のソフトウェアに切り替えることが可能な場合は、画像はG言語のクリップボードに自動的に取り込まれます。その場合は、ファイルシステムから直接画像ファイルをドラッグするか、またはLabVIEWの**編集**メニューの**ファイルから画像をインポート**...の項目を使用してGのクリップボードに画像ファイルを取り込みます。Windows 3.1では、BMP、CLP、EMF、GIF、JPG、LZW、PCX、PNG、TARGA、TIFF、WMF、およびWPGファイルに対しては後者の方法を使用します。Windows 95やWindows NTでは、BMP、CLP、EMF、JPG、PNG、およびWMFファイルを使用します。

Windows 95やWindows NTでは、ビットマップやメタファイルを直接LabVIEWやBridgeVIEWのパネルにドラッグアンドドロップすることができます。

（Macintosh）ペイントプログラムからMacintoshのクリップボードに画像を直接コピーしてG言語のソフトウェアに切り替えることが可能な場合は、画像はG言語のクリップボードに自動的に取り込まれます。

（UNIX）`xwd`コマンドを使用して作成したX Window Dump（XWD）タイプの画像、あるいはJPGまたはPNG画像を、UNIXのEditメニューのImport Picture from File...の項目を使用して取り込みます。

（すべてのプラットフォーム）G言語ソフトウェアのクリップボードに画像が存在する場合は、その画像をフロントパネル上で静止画像として貼り付けることができます。あるいは、ポップアップメニューの**画像をインポート**項目や制御器エディタの**画像をインポート**項目を使用することもできます。

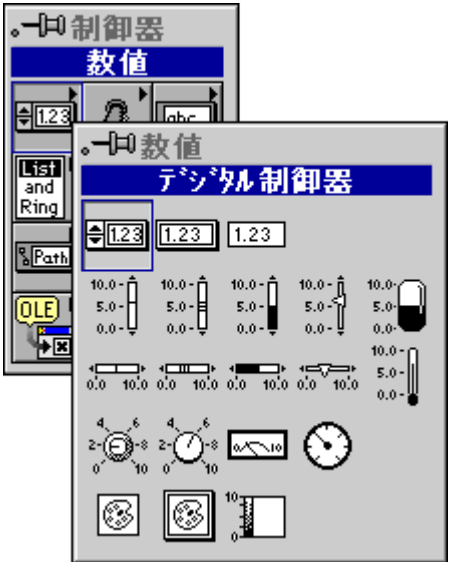
他のプログラムから画像を取り込む方法の例は、`examples¥general¥controls¥custom.llb`を参照してください。

9

数値制御器と数値表示器

この章では、さまざまなスタイルの数値制御器および数値表示器の作成、操作、および構成方法について説明します。

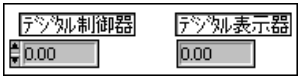
パレットで**制御器→数値**を選択すると、次の図に示すように制御器と表示器の新しいパレットが表示されます。



制御器および表示器には、デジタル、スライド、回転、リング、列挙、カラーボックス、カラーランプ制御器があります。

デジタル制御器とデジタル表示器

デジタルの数値制御器および数値表示器を次の図に示します。



デジタル数値は、数値データを入力あるいは表示するための最も簡単な手段です。

操作ツール

表示されている値を増加させたり減少させたりするためには、操作ツールと下記のいずれかの方法を使用します。

- デジタル表示ウィンドウの内側をクリックし、キーボードから数字を入力します。
- デジタル制御器またはデジタル表示器の増分矢印または減分矢印をクリックします。
- 変更したい数字の右側にカーソルを置き、キーボードの上向き矢印または下向き矢印を押します。

入力ボタン

ツールバーに表示される**入力**ボタンは、テンキーパッドの<Enter>キーを押すか、表示ウィンドウの外側をクリックするか、または**入力**ボタンをクリックするまでは古い値が新しい値に置き換わらないことをユーザに示すためのものです。このような確定動作を間に入れることにより、VIの実行中に入力途中の値が入力値とみなされるのを防ぐことができます。これは、たとえばデジタル表示の値を135に変更する場合に、135と入力し終わる前にVIが途中の1あるいは13を入力値として受け取らないようにするためです。このルールは、増分矢印や減分矢印を使用して変更した値には適用されません。

数値制御器は、10進数、小数点、+、-、大文字または小文字のe、用語のInf（無限）およびNaN（Not a Number＝数字でない）という語、絶対時間フォーマットでは/と:、および大文字または小文字のamとpmだけを受け付けます。選択した表記法の制限を超えると、数字が自動的に本来の限界値に設定されます。たとえば、バイト整数用に設定されている制御器に1234という値を入力すると、自動的に127という値に設定されます。また、NaNをaNNと入力したり、InfをIfnと入力するなど、数値でない値を誤って入力した場合は、その値は無視され、前の値が使用されます。

通常、増分ボタンは1つのケタの数字だけを変更します。そのケタの数字を9より大きくしたり、0より小さくしようとすると、となりのケタの数字が変化します。増加と減少は、自動的に反復実行されます。増分ボタンをクリックしてマウスボタンを押したままにすると、表示されている値が繰り返し増加します。<Shift>キーを押しながら増分ボタンを押すと、1回の増分の幅が最初は1つずつ、次は10ずつ、その次は100ずつというように段階的に大きくなります。また、限界値に近づくと増分の幅が小さくなり、値が限界値に達すると通常の増分幅になります。

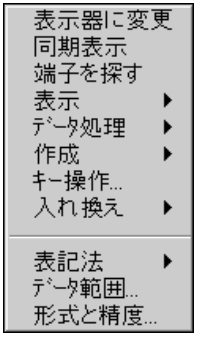
時間と日付のデジタル制御器では、個々の相対時間の要素に加え、それぞれの要素（秒、分、時、日、月、年）も増加または減少させることができます。

デジタル表示の1の位（ケタ）以外の数字を増加させるには、操作ツールを使用して挿入ポイントを変更したい位の右に移動します。数字の変更が終わると、再び1の位が増分の対象になります。

制御器では、数字が大きすぎてデジタル表示に収まりきらなくなる場合があります。そのような場合は、制御器を横に伸ばしてサイズを変更すると数字全部を見ることができます。

デジタル数値のオプション

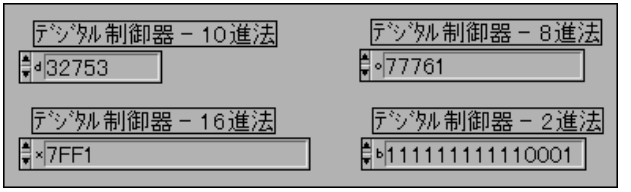
デジタルの数値のデフォルト値は、そのポップアップメニューを使用して変更することができます。デジタル数値のポップアップメニューを次に示します。



整数を他の基数で表示する

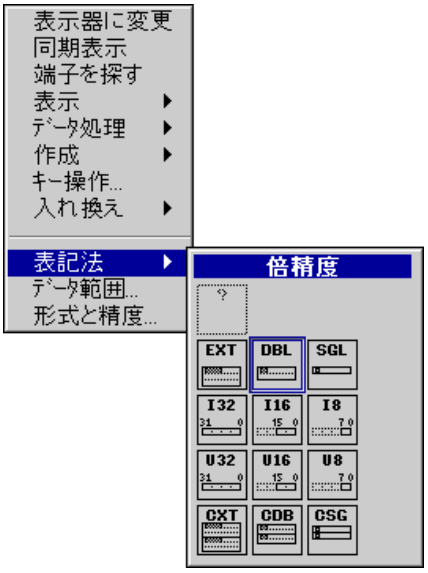
符号の付いた整数や符号の付いていない整数データは、10進数だけでなく16進数、8進数、あるいは2進数でも表示することができます。表記法を変更するには、数値のポップアップメニューで**表示→基数**を選択します。次に示すように、数値表示のボックスに d が表示されます。

d をクリックすると前図に示したメニューが表示されます。次の図は、32,753という数字を各基数の表記法で表したものです。



数値の表記法を変更する

デジタルの数値制御器または数値表示器では、12 通りの表記法の中から選択することができます。制御器または表示器のポップアップメニューの**表記法**の項目を使用して、32 ビット単精度（SGL）、64 ビット倍精度（DBL）、拡張精度（EXT）浮動小数点数、または、符号付き（I8）あるいは符号なし（U8）のバイト整数（8 ビット）、符号付き（I16）あるいは符号なし（U16）のワード整数（16 ビット）、符号付き（I32）あるいは符号なし（U32）の倍長整数（32 ビット）の 6 種類の整数表記法のいずれかに変更します。また、拡張精度複素数（CSG）、倍精度複素数（CDB）、または単精度（CSG）浮動小数点複素数を選択することもできます。これらの選択項目を次の図に示します。



数値制御器と数値表示器の範囲オプション

それぞれの表記法には、本来の範囲の最小値と最大値があります。たとえば、符号付きバイト整数の値は–128～127の範囲に制限されますが、浮動小数点数の範囲は次の表に示す通りです。

表 9-1 浮動小数点数の範囲オプション

精度	単精度	倍精度	拡張（プラットフォームにより異なる）
最小の正数	1.5E–45	5.0E–24	1.9E–4951
最大の正数	3.4E38	1.7E308	1.1E4932
最小の負数	–1.5E- 45	–5.0E–324	–1.9E–4951
最大の負数	–3.4E38	–1.7E308	–1.1E4932

☞ **注**　**G は表 9-1 浮動小数点数の範囲オプションで示した範囲内で処理することができますが、テキストフォーマットで G が表記および表示できる拡張浮動小数点数の範囲は ±9.9999999999999999E999 です。HP-UX、Concurrent、および Power Macintosh では、拡張浮動小数点数の範囲は倍精度の数字と同じです。**

ポップアップメニューの**データ範囲...**項目を使用すると、これらの本来の範囲内で新たに限界を設定するすることができます。次のダイアログボックスが表示されます。

数値の範囲をチェックする

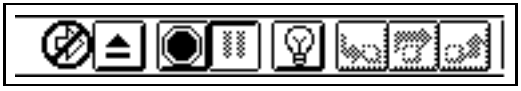
さらに、中間値の増分の幅を一定量に制限することもできます。たとえば、ワード正数の増分の幅を10に制限したり、単精度浮動小数点数の増分の幅を0.25に制限したりすることができます。値の範囲や増分幅を制限する場合は、VIやオペレータが範囲や増分幅を超えて値を設定しようとしたときの対応策も決定する必要があります。その選択肢として、次のような手段があります。

無視　G は無効な値を変更せず、フラグも設定しません。増加矢印または減少矢印をクリックすると値は設定した増分幅で変更されますが、値が最大値や最小値を超えることはありません。

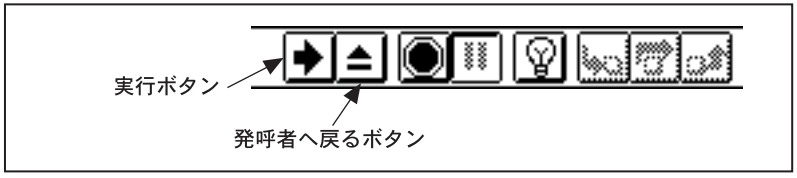
強制　Gは無効な値を最も近い有効値に自動的に設定します。たとえば、最小値が3、最大値が10で、増分幅が2であれば、有効な値は3、5、7、9、10となります。したがって、値0は3、値6は7、値100は10に強制的に設定されます。

中断　Gは値が無効なときはVIやサブVIの実行を中断します。無効な値で中断するように設定すると、G はフロントパネルを開いてエラーを表示する必要が生じた場合に備えて、フロントパネルのすべてのデータをメモリに保存します。

値が無効な制御器を含む最上位のVIは、実行できません。サブVIを実行する前（サブVIを実行しようとしたとき）に制御器の値が無効であった場合は、VIは中断されます。VIのフロントパネルが開き（またはアクティブウィンドウになり）、無効な制御器が赤枠（白黒モニタの場合は太い黒枠）で囲まれます。中断されたサブVIのブロックダイアグラムのツールバーは、次のように表示されます。

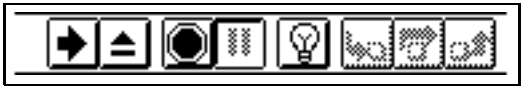


実行を再開するためには、制御器を有効な値に設定する必要があります。すべての制御器が有効な値になると、ツールバーは次の図のようになります。**実行**ボタンをクリックして実行を再開します。



サブVIの実行中に制御器または表示器の値が無効になると、ブレークポイントがある場合と同じように実行は一時的に停止します。VIのフロントパネルが開き（またはアクティブウィンドウになり）、現在無効な表示器および制御器が赤枠（白黒モニタの場合は太い黒枠）で囲まれます。中断したVIの実行が終了したときに1つでも無効な値が存在すると、次の図で示すように**発呼者へ戻る**ボタンは使用不能になります。

呼び出し側のVIへ戻るためには、表示器を有効な値に設定する必要があります。また、有効な出力値を生成するように制御器の値を変更したのち、**実行**ボタンを押して再度サブVIを実行することもできます。すべての表示器の値が有効になると、ツールバーは次の図のようになります。**発呼者へ戻る**ボタンをクリックして実行を再開します。

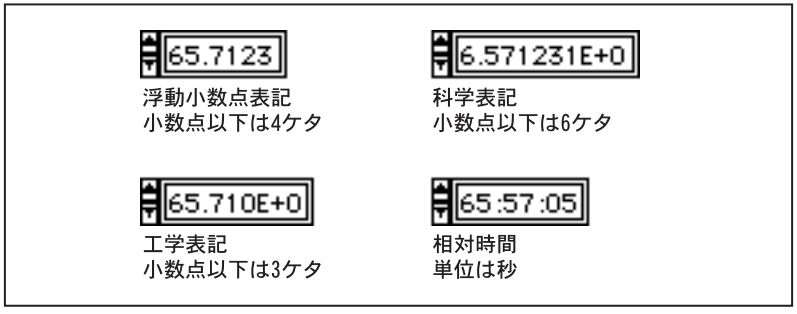


デジタル表示のフォーマットと精度

デジタル表示のフォーマットは、数値または時間と日付を選択することができます。数値には、浮動小数点表記、科学表記、工学表記、または秒単位の相対時間のいずれかを選択できます。その精度、すなわち小数点の右側に表示される桁数は、0ケタから20ケタまでの間で選択することができます。選択した精度は値の表示にのみ適用され、内部の精度は表記法によって異なります。

デジタル表示のこれらのパラメータを変更するには、表示のポップアップメニューから**形式と精度 ...** を選択します。次の図のような数値フォーマットのダイアログボックスが開きます。

ダイアログボックスの上の欄には、選択の内容に応じて例が表示されます。デジタル制御器のフォーマットと精度の設定の例を次の図に示します。



絶対時間および（または）日付のフォーマットを設定するには、ダイアログボックスの上にある形式リングから**時間と日付**を選択します。ダイアログボックスが次の図に示すような表示に変わります。

形式は、時間と日付のいずれか、または両方に対して設定できます。時間または日付のどちらか一方だけを入力すると、指定しなかった部分のデータには推測値が適用されます。時間を入力しなかった場合は、12:00 a.m. という推測時間が適用されます。日付を入力しなかった場合は、前の日付の値が適用されます。日付の形式でない制御器に対して日付を入力すると、環境設定ダイアログボックスでの設定に基づいて月、日、年の順序が決定されます。年に対して2ケタの値を入力した場合は、その値が38未満のときは21世紀の年を、38以上のときは20世紀の年を表すものとみなされます。

絶対時間は時間と日付の文字列として表示されますが、ソフトウェアの内部では協定世界時（= UTC。これまでグリニッジ標準時に相当）の1904年1月1日午前12:00から現在までの経過秒数で表されます。これらのデータは、ソフトウェアの内部で追跡されます。

☞ **注**　**絶対時間形式の制御器では、時間と日付のどちらか一方だけを入力することも、その両方を入力することもできます。日付に推測値を使用したくない場合は、相対時間を使用してください。**

選択を行うたびに、ダイアログボックスの右上部分の例の表示が変わることに注意してください。

日付と時間の有効範囲は、次に示すようにコンピュータのプラットフォームによって異なります。

（Windows）1970 年 1 月 2 日午前 12:00 ～ 2106 年 2 月 4 日午前 12:00

（Windows95/NT）1970 年 1 月 2 日午前 12:00 ～ 2038 年 1 月 17 日午前 12:00

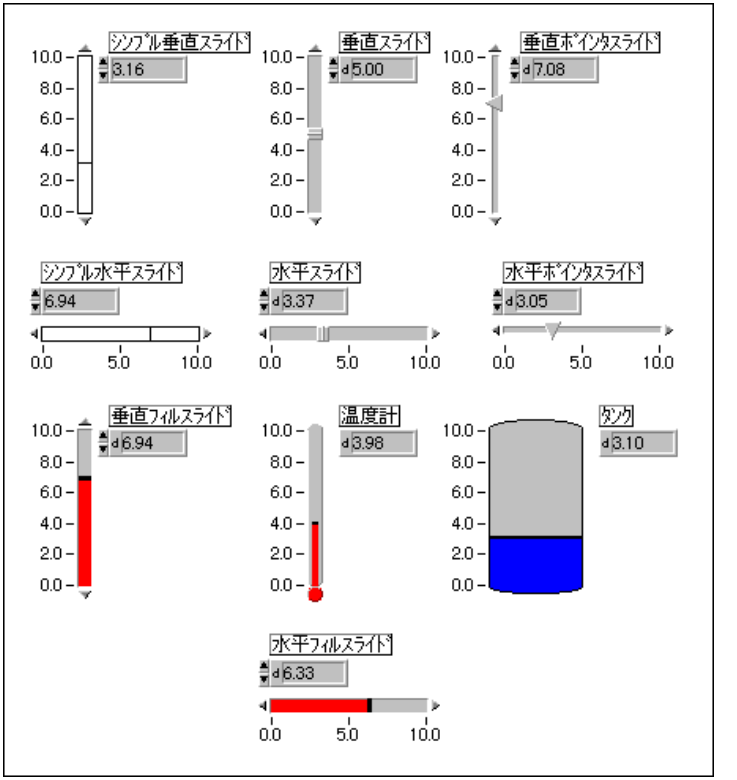
（Macintosh）1904 年 1 月 2 日午前 12:00 ～ 2040 年 1 月 2 日午前 12:00

（UNIX）1901 年 12 月 15 日午前 12:00 ～ 2038 年 1 月 17 日午前 12:00

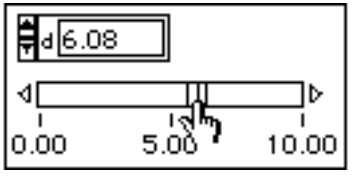
上記の範囲は、時間帯や夏時間の使用により最長で数日間延びる場合があります。

スライドの数値制御器と数値表示器

スライド制御器およびスライド表示器を次の図に示します。

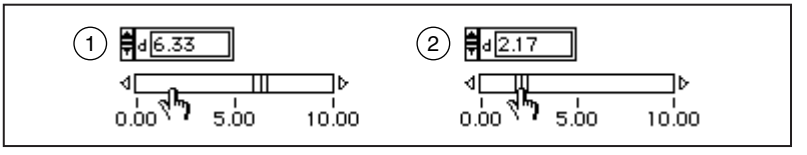


それぞれのスライドにはデジタル表示があります。本章の「デジタル制御器とデジタル表示器」の項で述べたように、デジタル表示はスライド制御器にデータを入力するために使用します。スライダ、スライドの外枠、スケールなどの部品や、値を入力したり変更するための増分ボタンには、操作ツールを使用します。スライダは、移動することによって制御器の値を表示する部品です。ハウジングは静止部品で、スライダはこのハウジング上を移動します。スケールはスライダの値を示し、増分ボタンはハウジングの両側にある小さな三角形です。次の図はスライダの例を示します。

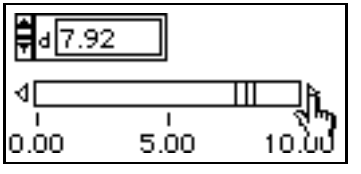


スライダは､操作ツールを使用して新しい位置に移動することができます。VI の実行中に位置を変更すると、VI が制御器の値を読み取る頻度によってはその中間値がプログラムに渡される場合があります。

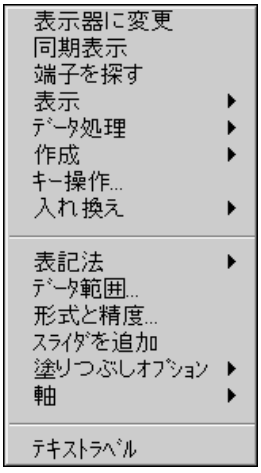
また、次の図に示すように、スライド上の任意の位置をクリックしてスライダをその位置に素早く移動させることもできます。この場合、中間値はプログラムには渡されません。



増分ボタンのあるスライドを使用している場合は、次の図に示すように、増分ボタンをクリックしてスライダを矢印の方向にゆっくり移動させることができます。スライダを速く動かしたいときは、<shift>キーを押しながら増分ボタンをクリックします。中間値は、プログラムに渡される可能性があります。



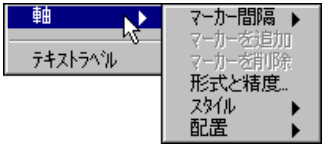
デジタル数値と同様、スライドのポップアップメニューには**表記法、データ範囲...**、および**形式と精度**の項目があります。これらの項目は、スライドが複素数を表示できない点以外はデジタル表示の項目と同じです。スライドには、これら以外のオプションもあります。スライドのポップアップメニューを次に示します。



ポップアップメニューの**表示→デジタル表示**項目は、スライドのデジタル表示を表示するかどうかを制御します。

スライドのスケール

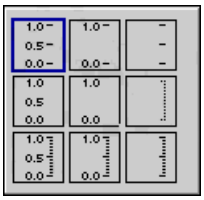
スケールのサブメニュー項目は、スケールに対してのみ適用されます。次の図はスケールのポップアップメニューを示します。



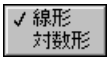
マーカ間隔オプションについては、本章の「スケールマーカ」の項で説明します。

形式と精度オプションの機能は、本章の「デジタル表示のフォーマットと精度」で説明した通りです。

スタイル項目は、次の図で示すパレットを表示します。チックマークやスケール値のないスケールを表示したり、スケールそのものを隠すこともできます。



配置項目は、線形スケールや対数形スケールの目盛りの間隔指定オプションを表示します。



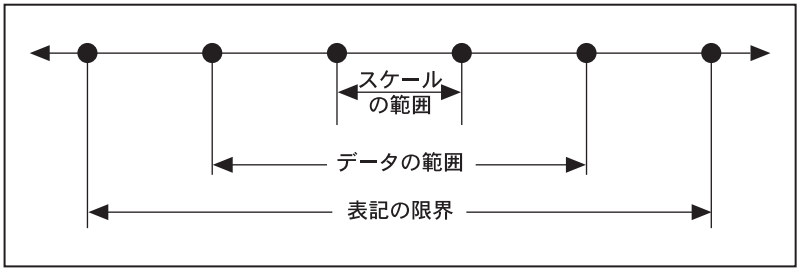
対数形の目盛り間隔を変更したときにスケールの下限値が0以下になっていると、限界値は自動的に正の数になり、ほかの値もそれに応じて変更されます。スケールオプションはマッピング機能も含め、**データ範囲**のポップアップオプションで変更したスライドのデータ範囲の値とは無関係である点に注意してください。データを対数形の値に制限したいときは、データの範囲を0以下の値が含まれないように変更します。

スケールマーカ

数値制御器や数値表示器のスケールには、マーカの位置に相当する値を示すラベルである2つ以上のマーカがあります。

スケールの限界値を変更する

外側の2つのマーカは、スケールの限界を示します。これらのマーカは、かならずしも範囲の限界と一致している必要はありませんが、制御器や表示器のサブセットしても使用できます。たとえば、符号付き16ビット整数の制御器または表示器のデフォルトは–32,768から32,767の範囲ですが、データを–1,000から1,000の範囲に設定し、スケールの限界を0と500に設定することもできます。

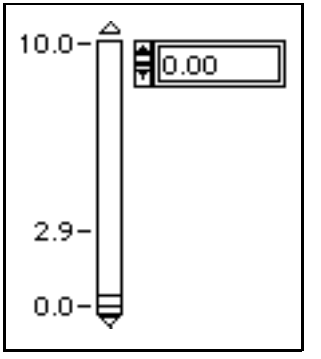


スケールの最大値、最小値、および増分幅は、操作ツールまたはラベリングツールを使用して5通りの方法で対話形式で変更することができます。スケールをプログラム上で変更したいときは、属性ノードを使用します。属性ノードを使用したプログラム上でのスケールの変更方法についての詳細は、「第22章 属性ノード」を参照してください。

- 最大値の表示部に新しい値を入力した場合、最小値は変化せず、増分幅が自動的に再計算されます。
- 最小値の表示部に新しい値を入力した場合、最大値は変化せず、増分幅が自動的に再計算されます。
- 最大値の表示部に現在の最小値を入力すると、スケールが反転し、それまでの最小値が最大値に、それまでの最大値が最小値になります。増分幅は再計算されます。
- 中間値マーカに値を入力すると、その値から最小値を引いた値が増分幅になります。
- スライドのサイズを変更すると、マーカが重ならないように増分幅が調節されます。

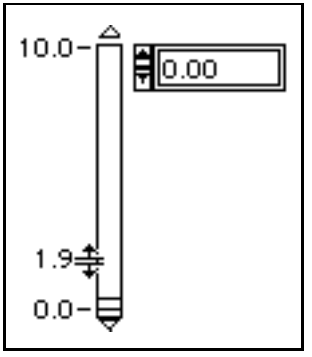
スケールマーカを不均等な間隔で配置する

デフォルトでは、スケールマーカは等間隔で配置されます。ただし、必要に応じてスケールの内側にあるマーカの位置を正確に指定することができます。これは、スライド上でいくつかの特定のポイント（設定ポイントや閾値など）を示すのに便利です。マーカを不均等な間隔で配置したいときは、スケールのポップアップメニューから**スケール→マーカ間隔→任意**を選択します。次に、スライドをポップアップして**スケール→マーカを追加**を選択するか、またはマーカをポップアップして**マーカを追加**を選択します。次の図で示すように、マーカが任意の位置に表示されます。



任意マーカを削除するには、スライドをポップアップして**スケール→マーカを削除**を選択するか、またはマーカをポップアップして**マーカを削除**を選択します。

マーカが作成されたら、マーカに値を入力することができます。マーカは、入力した値に相当する位置に自動的に移動します。マーカの位置は、スケールのデータポイントの横にあるチックと呼ばれる小さな線をドラッグして移動することもできます。その場合は、操作ツールをチックの上に移動します。次の図で示すように、カーソルが変化してチックを移動できることを示します。



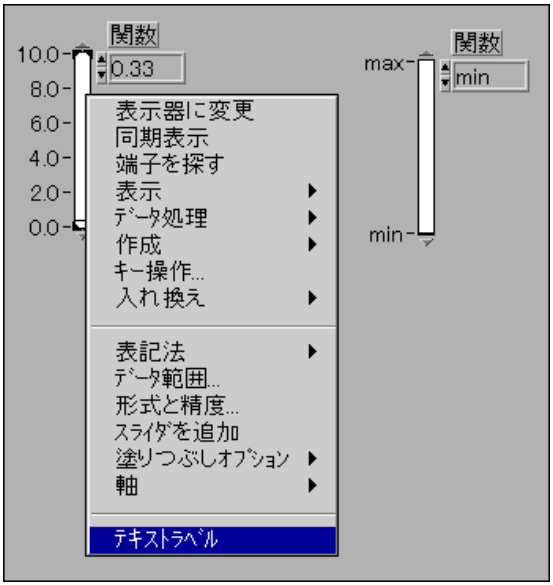
均等マーカモードでは、チックをドラッグするとマーカの間隔が変化します。任意マーカモードでは、チックをドラッグすると現在のマーカだけが移動し、その他のマーカは移動しません。<Ctrl>**（Windows）**、<option>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**キーを押しながらドラッグすると、新たなマーカが作成されます（**スケール→スタイル**ダイアログボックスでチックが表示されないように設定されているときは、チックをドラッグすることはできません）。

任意マーカは、2 つのエンドマーカではなく内側のマーカのみに影響します。現在の範囲内に任意マーカが表示されていないときは、スケールは一時的に均等マーカに戻ります。

テキストスケール

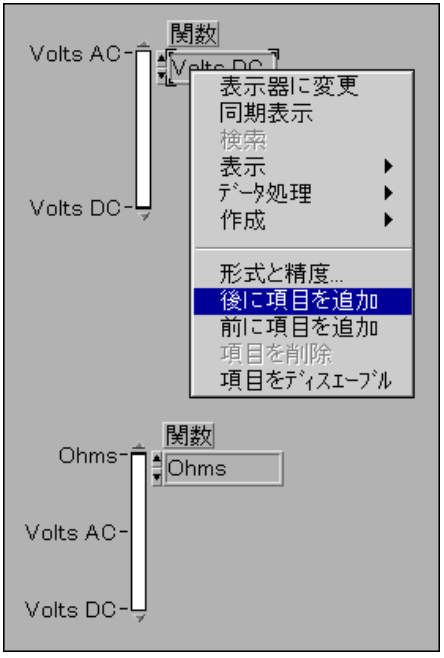
数値スケールに対しては、テキストラベルを使用することもできます。ラベルの利点は、文字列を整数値と関連付けられる点にあります。このような構成は、相互に排他的なオプションを選択するのに便利です。このオプションを使用する場合は、スライドのポップアップメニューから**テキストラベル**を選択します。minおよびmaxというデフォルトのテキストラベルを含むスライド制御器が表示され、必要なラベルを直ちに入力することができます。各ラベルを入力したあとで新たなラベルを作成するには、<Shift-Enter>**（Windows** および **HP-UX）**または<Shift-Return>**（Macintosh** および**Sun）**キーを押します。最後のラベルの入力が終わってから、テンキーパッドの<Enter>キーを押します。

ラベリングツールは、テキスト表示や実際のスライド上の min ラベルと max ラベルの編集にも使用できます。



作成するテキストラベルにどんな数値が関連付けられているかを知りたいときは、テキスト表示をポップアップして**表示→デジタル表示**を選択します。これらの値は常に 0 から始まり（垂直スライドではいちばん下、水平スライドでは左または右）、各テキストラベルごとに 1 ずつ増加します。

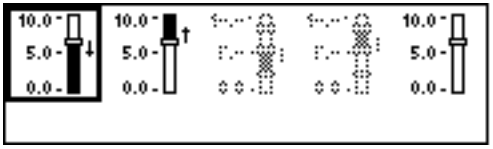
新しいラベルを作成する場合は、次の図に示すようにテキスト表示のポップアップメニューの**後に項目を追加**または**前に項目を追加**を使用します。



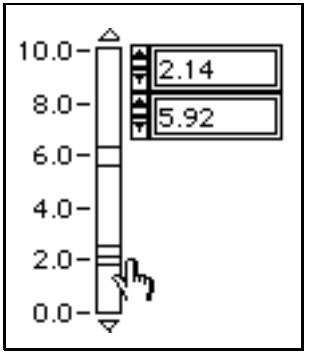
既存の項目を編集している場合は、<Shift-Enter> を押して新しい項目に進むこともできます。

塗りつぶしたスライドと複数値のスライド

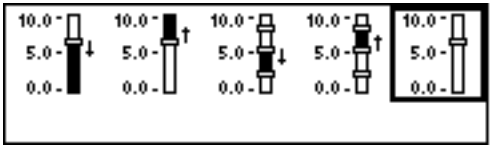
数値パレットには、最小値からスライダ値までの部分を塗りつぶして表示するように構成された 4 つの制御器があります。4 つの制御器とは、垂直スライド、水平スライド、タンク、および温度計です。ポップアップメニューの**フィルオプション**項目を使用すると、すべてのスライドを塗りつぶしスライドに、塗りつぶしスライドを通常のスライドに切り替えることができます。通常は、次の図に示すように、最小値からスライダの位置までを塗りつぶす、最大値からスライダの位置までを塗りつぶす、塗りつぶしは使用しない、という 3 つの選択肢があります。



また、1つのスライドに複数の値を表示することもできます。その場合は、ポップアップメニューで**スライダを追加**を選択します。次の図に示すように、新しいスライダと新しいデジタル表示が表示されます。



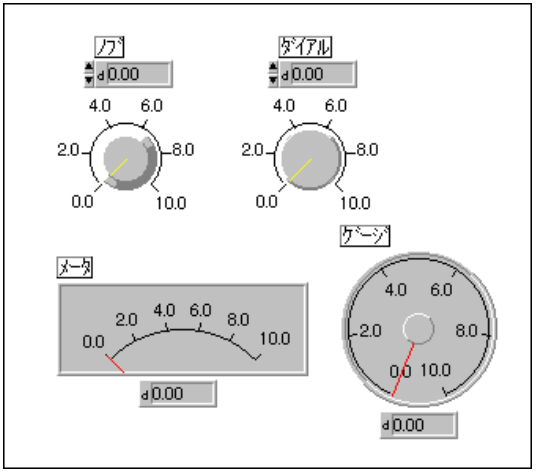
このとき、**フィルオプション**パレットにはさらに2つのオプション、すなわち**上の値まで塗りつぶし**と**下の値まで塗りつぶし**が表示されます。これらのオプションは、アクティブなスライダに対して適用されます。次の図に、これら5つのすべてのオプションを示します。



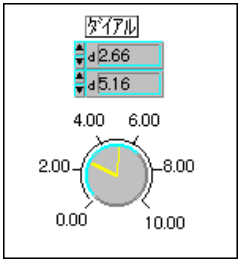
スライドの制御器および表示器のサンプルについては、`examples¥general¥controls¥alarms1d.llb`を参照してください。

回転数値制御器と回転数値表示器

回転数値制御器と回転数値表示器を次の図に示します。



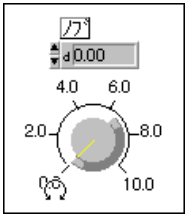
回転オブジェクトのオプションは、ほとんどがスライドのオプションと同じです。回転オブジェクトのスライダ（指針）は、スライド式ではなく回転式ですが、操作方法はスライドの操作方法と同じです。



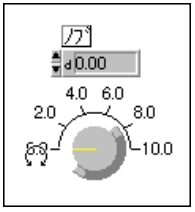
回転制御器は、線形制御器と同様、複数の値を表示することができます（前図参照）。新しい値を追加する場合は、スライドに新しい値を追加する場合と同じように、ポップアップメニューから**指針を追加**を選択します。

位置決めツールをスケールの上に移動すると、ツールが回転カーソルに変わります。スケールの外側の限界値をクリックしてドラッグすると、スケールの周りのスケールアーチが変化します。スケールアーチを回転したいときは、スケールの内側のマーカをクリックしてドラッグします。この場合、変わるのは開始角と終了角だけで、スケールの範囲は変化しません。この操作の手順を次の図に示します。<Shift> キーを押しながら操作すると、スケールアーチが45度刻みで回転します。

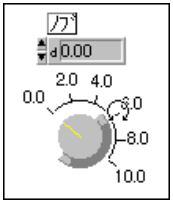
カーソルをツマミのスケールの上に移動します。カーソルが、両端に矢印が付いた馬蹄形に変わります。



外側のマーカをドラッグすると、スケールアーチの長さが変化します。



内側のマーカをドラッグすると、スケールアーチが回転します。



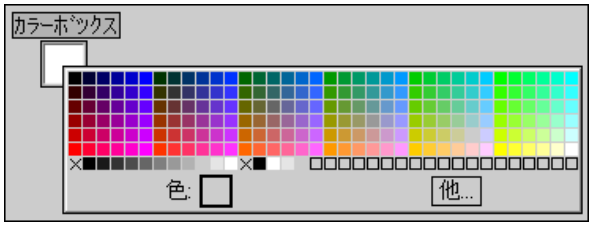
回転スケールの目盛りも、線形スケールの場合と同様操作ツールでドラッグすることができます。また、目盛り間隔も、線形スケールの場合と同じように不均等にすることができます。詳しくは、本章の「スケールマーカを不均等な間隔で配置する」の項を参照してください。

リング制御器およびリング表示器についての詳細は、「第13章　リストとリングの制御器および表示器」を参照してください。

カラーボックス

カラーボックスは、指定された値に相当する色を表示します。色の値は、RRGGBB という形式の 16 進数で表されます。この数字は、最初の 2 ケタが赤の値を、次の 2 ケタが緑の値を、最後の 2 ケタが青の値をそれぞれ制御します。次の図に例を示します。

カラーボックスの色を設定するためには、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、次の図に示すような**カラー**パレットを呼び出します。選択したい色の上でマウスボタンを放すと、その色を選択することができます。

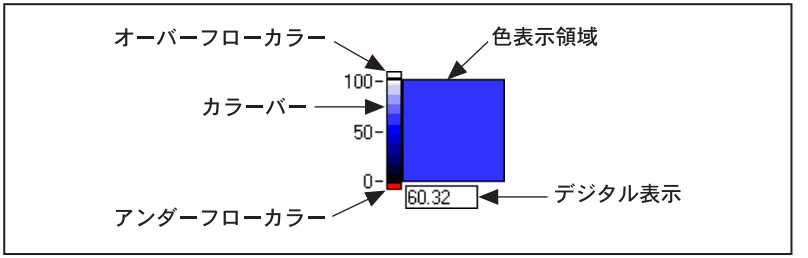


カラーボックスは、おもにフロントパネルで表示器として使用されます。カラーボックスの色を設定する最も簡単な方法は、ブロックダイアグラムウィンドウの**関数→数値→その他の数値定数**パレットのカラーボックス定数を使用する方法です。ブロックダイアグラム上で、カラーボックス定数をカラーボックス端子に配線します。操作ツールまたは色付けツールでカラーボックス定数をクリックすると、カラーボックスを設定する際に使用するのと同じ**カラー**パレットが表示されます。Case ストラクチャの内部で一連のカラーボックス定数を使用すると、フロントパネル上でカラーボックス表示器をそれぞれの状態を示すさまざまな色に変更することができます。

カラーパレットの左下隅に表示される小さい T は、透明を表します。

カラーランプ

カラーランプは、色を使用してその数値を表示します。カラーバーは、最低2つ以上の任意マーカによって構成されます。それぞれの任意マーカは、数値と、その数値に相当する表示色で構成され、入力値が変わると表示される色がその値に相当する色に変わります。次の図で1つの例を示します。



これらの組み合わせは、カラーバーの任意マーカを使用して作成します。任意マーカは、それぞれが1つの値を指定し、その値に対応する色を持っています。マーカの色を変更する場合は、ランプではなくマーカをポップアップします。そして、好みに応じてマーカを追加したり、値を変更したり、色を設定することができます。

指定したマーカ間の値に対して中間色を表示するかしないかは、カラーランプのポップアップメニューの**色の補間**項目を使用して指定します。**色の補間**を無効にした場合は、色は現在の値以下の最も高いレベルの色に設定されます。

色の配列は、常にマーカ値ごとに記憶されます。ランプのスケールは、配列の最大値と最小値に対応しています。スケールの最大値または最小値が変わると、色の配列のレベルが新しい値の間で自動的に再配分されます。

カラーバーには、表示 / 非表示を指定できるさまざまなオプションがあります。これらのオプションには、単位ラベル、デジタル表示、ランプがあります。

この制御器のランプの構成要素は、スケールの上と下にそれぞれ1つの追加色を持っています。これらの追加色は、オーバーフローまたはアンダーフローが発生したときに表示する色を選択するために使用します。制御器の値は、カラーバーの最大値より大きい、または最小値よりも小さくなります。これらの領域を操作ツールでクリックし、**カラー**パレットでオーバーフローの色とアンダーフローの色を選択します。

カラーランプとカラーボックスは、いずれも数値を 1 つの色として表示するために使用されます。カラーランプでは、任意の範囲の色に任意の範囲の値を割り当てることができます。一方、カラーボックスには数値の赤、緑、青の各成分で指定した色だけが表示され、与えられた値に対応する色を変更することはできません。

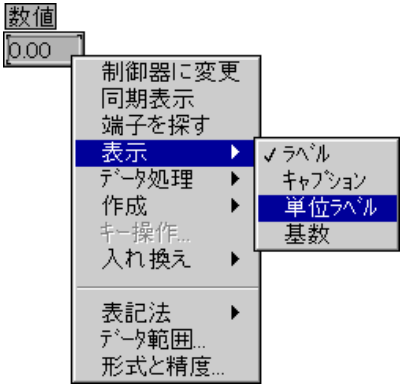
☞ **注　カラーランプではモニタの処理能力に応じた数だけ色を使用できます。モニタで使用できる色よりも多く色を使用することはできません。**

カラーバーを使用すると、強度グラフ制御器および強度チャート制御器のカラーテーブルを指定することができます。これらの制御器についての詳細は、「第15章 グラフとチャートの制御器および表示器」を参照してください。

単位の種類

どのような数値制御器でも、その制御器に応じた物理的な単位（メートル、キロメートル / 秒など）を使用することができます。ただし、単位を使用する数値制御器のデータタイプは、浮動小数点数に限られます。

制御器の単位の表示や変更には、単位ラベルと呼ばれる独立した付属ラベルが使用されます。このラベルは、次の図で示すように、ポップアップメニューで項目を選択することによって表示することができます。



単位ラベルが表示されたら、メートルは m、フィートは f、秒は s というように単位を標準的な省略形で入力することができます。

使用できる単位がわからないときは、とりあえず簡単な単位（たとえば m）を入力し、その単位をポップアップして**単位...**を選択します。ダイアログボックスに使用可能な単位に関するデータを表示し、このダイアログボックスを使用して先に入力した単位を最適な単位に変更します。

次に示した数値制御器の単位は、メートル/秒に設定されています。

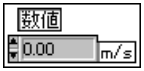


表 9-1 から表 9-5 は、単位機能で使用できるすべての単位を示します。

表 9-2 基本単位

数量名	単　位	略　号
平面角	ラジアン	rad
立体角	ステラジアン	sr
時間	秒	s
長さ	メートル	m
質量	グラム	g
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd

表 9-3 特別な名前を持つ派生単位

数量名	単　位	略　号
周波数	ヘルツ	Hz
力	ニュートン	N
圧力	パスカル	Pa
エネルギー	ジュール	J
電力	ワット	W
電荷	クーロン	C
電圧	ボルト	V
静電容量	ファラド	F
電気抵抗	オーム	Ohm
コンダクタンス	シーメンス	S
磁束	ウェーバ	Wb

表 9-3 特別な名前を持つ派生単位（続き）

数量名	単　位	略　号
磁束密度	テスラ	T
インダクタンス	ヘンリー	H
光束	ルーメン	lm
照度	ルックス	lx
摂氏温度	摂氏度	degC
活量	ベクレル	Bq
吸収線量	グレイ	Gy
等価線量	シーベルト	Sv

表 9-4 SI 単位と併用する単位

数量名	単　位	略　号
時間	分	min
時間	時	h
時間	日	d
平面角	度	deg
平面角	分	’
平面角	秒	"
容積	リットル	l
質量	立法トン	t
面積	ヘクタール	ha
エネルギー	電子ボルト	eV
質量	単一原子量単位	u

表 9-5 CGS単位

数量名	単　位	略　号
面積	バーン	b
力	ダイン	dyn
エネルギー	エルグ	erg
圧力	バール	bar

表 9-6 その他の単位

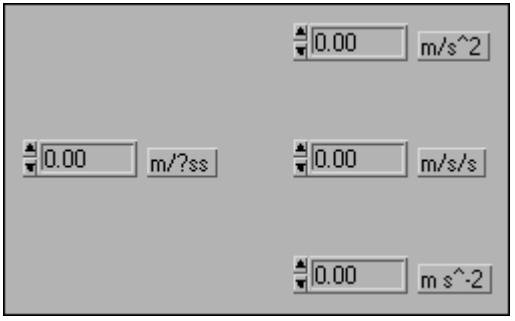
数量名	単　位	略　号
華氏温度	華氏度	degF
摂氏温度	摂氏度	Cdeg
華氏温度差	華氏度	Fdeg
長さ	フィート	ft
長さ	インチ	in
長さ	マイル	mi
面積	エーカー	acre
圧力	気圧	atm
エネルギー	カロリー	cal
エネルギー	英国温度単位	Btu

単位を入力する

単位ラベルに無効な単位を入力すると、次の図で示すようにラベルに無効な単位であることを示す？マークが表示されます。

単位はかならず正しい略号で入力してください。誤った略号を入力すると、次の図で示すように？マークが表示されます。

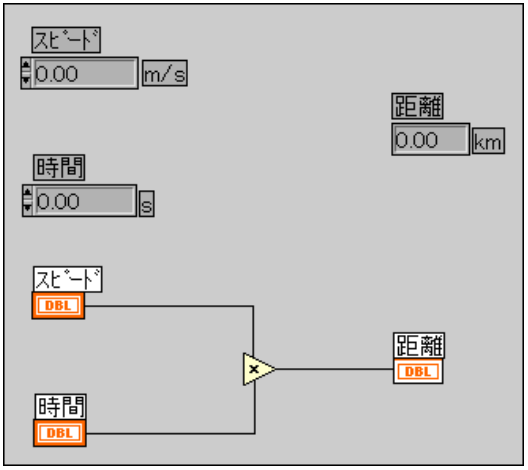
また、解釈が曖昧な単位は入力できない点にも注意してください。次の図では、m/ss に？が表示されていますが、これはm/ss が m/s2 であるのか (m/s)*s であるのかが不明確なためです。このような曖昧さを避けるためには、次の図で示すように単位を別のかたちで入力する必要があります。



チャートやグラフに対しては、そのチャートあるいはグラフが対応する単位を持つオブジェクトに配線しない限り、単位を指定することはできません。配線すれば、そのチャートまたはグラフが接続により獲得した単位を表示することができます。この単位を変更することは可能ですが、異なる物理量の測定単位に変更することはできません。たとえば、チャートへの入力値の単位がmi である場合、チャートの単位をft やin、あるいはmに変更することはできますが、N、Hz、min、acre、あるいはAには変更することはできません

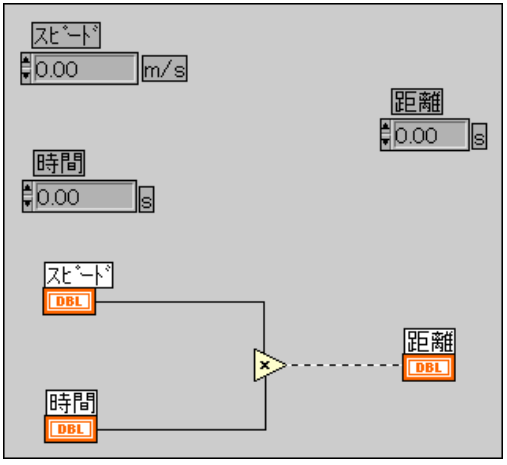
単位および単位の種類に関する厳密なチェック

次の図に示すように、単位を持つソースに配線されたワイヤは、互換性のある単位を持つ接続先にのみ配線することができます。



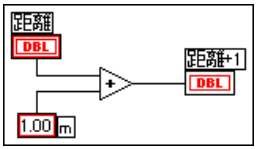
前の図の場合、表示器に対して指定された単位がキロメートルであったため、距離表示は自動的にキロメートルに換算されます。

次の図に示すように、単位に互換性がない信号は接続できません。次の図に示した破線のワイヤのポップアップメニューで**エラーリスト**を選択すると、エラーリストウィンドウに［ワイヤ： 単位が競合しています］というエラーが表示されます。



関数のなかには、単位が定義されておらず、単位を持つ信号に対して使用できないものもあります。たとえば、Increment という関数は単位が定義されていません。長さの単位を使用している場合、Increment は 1 メートル、1 キロメートル、1 フィートのどれを加算するのかわかりません。このような曖昧さのため、Increment のような関数は単位を持つデータに使用することはできません。

このような問題を回避する方法の 1 つに、ブロックダイアグラム上でで適切な単位を持つ数値定数と加算関数を使用して独自の Increment 単位関数を作成する方法があります。



☞ **注**　　**フォーミュラノードでは単位は使用できません。**

多形性単位

波形の RMS の値を計算する VI を作成したい場合、波形の単位を定義する必要があります。電圧波形、電流波形、温度波形などの波形は、それぞれ個別の VI が必要になります。1 つの VI が入力で受け取る単位に関係なく同じ計算を実行できるように、G 言語ソフトウェアには多形性単位という機能が用意されています。

多形性単位は、$x と入力することによって作成します。ここで x は数字です（例：$1）。これを、実際の単位のプレースホルダーとみなします。VI が呼び出されると、G は渡された単位をその VI で発生するすべての $x に置き換えます。$x には、かならず基本単位で表される単位（フィートやインチではなくメートル、メートル/秒など）が渡されます。

多形性単位は、固有の単位として扱われます。別の単位に変換することはできませんが、他の単位と同じようにダイアグラム全体に伝播されます。多形性単位を同じく $1 という略号を持つ表示器に接続すると、単位が一致し、VI のコンパイルが可能になります。

$1 は、他の単位と同様、組み合わせて使用することもできます。たとえば、入力に 3 秒を乗じたのち表示器に接続した場合、表示器の単位は $1 x でなければなりません。表示器の単位がそれ以外の単位であるときは、ブロックダイアグラムは不良ワイヤを表示します。

複数の多形性単位を使用する必要がある場合は、$2 や $3 などの略号を使用します。

多形性単位を持つサブVIを呼び出すと、その入力で受け取った単位に基づいて出力単位を計算します。たとえば、多形性単位$1と$2を使用して2つの入力を持つVIを作成し、そのVIが$1 $2/sという形の出力を生成するものとします。サブVIが、m/sの入力が$1入力に、kgの入力が$2入力に渡されるように接続されていると、出力単位はkg m/s^2として計算されます。

さらに、別のVIに$1と$1/sという2つの入力があり、出力が$1^2として計算されるものとします。このVIが、m/sの入力が$1入力に、m/s^2の入力が$1/s入力に渡されるように接続されていると、出力単位はm^2/s^2として計算されます。一方、このVIが、mの入力が$1入力に、kgの入力が$1/s入力に渡されるように接続されていると、そのサブVIの呼び出しは中断します。入力の1つは単位矛盾として宣言され、出力は（可能な場合には）もう1つの入力から計算されます。それぞれの単位は区別されるため、多形的なVIは多形的なサブVIを持つことができます。

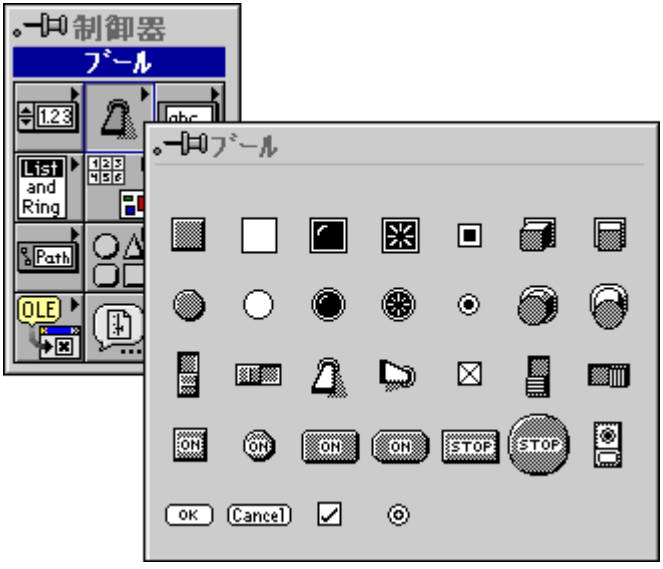
10

ブール制御器とブール表示器

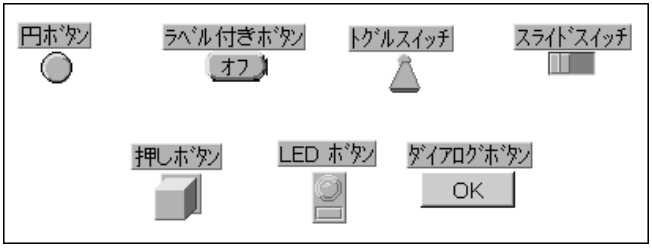
この章では、ブール制御器とブール表示器の作成、操作、および構成の方法について説明します。

ブール制御器とブール表示器の作成および操作

ブール制御器やブール表示器には、TRUE と FALSE（真と偽）という 2 つの値があります。ブール制御器とブール表示器は、次の図で示すように**制御器→ブール**パレットに用意されています。



機械的なプッシュボタン、切り替えスイッチ、スライドスイッチの機能を模倣したブール制御器を次の図に示します。



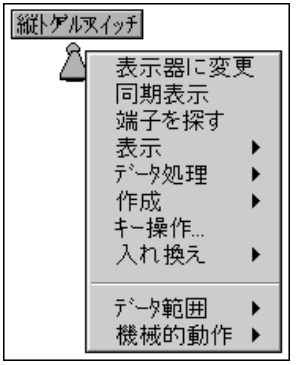
☞ **注**　**ダイアログのボタン、ダイアログのチェックマーク、およびダイアログのラジオボタンの表示方法は、プラットフォームによって異なります。それぞれのプラットフォームでのダイアログの制御器についての詳細は、「第8章　フロントパネルオブジェクトの概要」の「ダイアログボックスの制御器をカスタマイズする」の項を参照してください。**

LEDやライトを模倣したブール表示器を次の図に示します。

ブール制御器を操作ツールでクリックすると、TRUE（オン）とFALSE（オフ）の状態が切り替わります。表示器は出力専用であるため、実行モードでは表示器をクリックしても何も変化しません。編集モードでは、制御器と表示器の両方を操作することができます。

ブール制御器とブール表示器を構成する

それぞれのブール制御器あるいはブール表示器には、そのポップアップメニューにいくつかの項目が用意されています。ブールオブジェクトのポップアップメニューを次の図に示します。



ポップアップメニューの線より上に表示される項目は、制御器および表示器すべてに共通です。これらの項目については、「第8章　フロントパネルオブジェクトの概要」の「フロントパネルの制御器と表示器のオプション」の項で説明しています。また、**データ範囲**と**機械的動作**については、本章の後の項で説明します。

ブールのラベルを変更する

ブールパレットには、テキストを表示するいくつかの制御器があります。これらは**ラベル付きブール**と呼ばれます。次の図に示すように、ボタンはTRUEの状態のときには［オン］と表示し、FALSEの状態のときには［オフ］と表示します。ボタンを操作ツールでクリックすると、制御器は逆の状態に切り替わります。編集モードでは、ラベリングツールを使用していずれかの状態にあるテキストを編集します（たとえばオンを YES に変更）。

デフォルトでは、テキストはボタンの中央に表示されます。テキストを移動したいときは、ポップアップメニューから**テキストを切り放す**を選択します。

次に、位置決めツールを使用してテキストの位置を変えるか、または**テキストを中心に設定**を選択します。ブールのテキストを選択したのち、ツールバーのフォントリングを使用してテキストのフォント、サイズ、および色を変更します。また、**表示**のサブメニューでブールテキスト項目の設定を切り替えることにより、**ブールテキスト**をどちらの状態でも非表示にすることができます。

パレットにラベル付きブールとして表示されないブール制御器のデフォルトではラベルが付いていません。ただし、**表示**のサブメニューから**ブールテキスト**を選択してラベルを付けることができます。これらのブールのブールテキストは、移動することができます。テキストは、ポップアップメニューで**テキストを中心に設定**を選択しない限りボタンの中央に固定されることはありません。

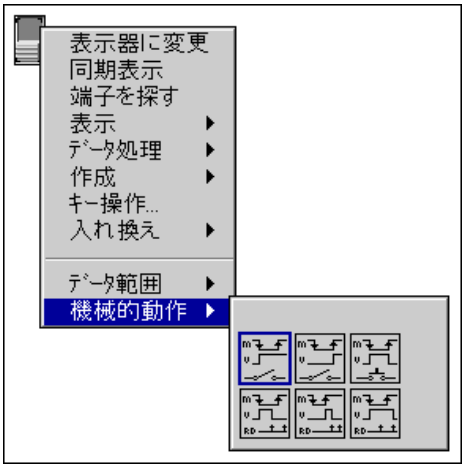
ネームラベルとブールテキストの両方ではなくどちらか一方のフォントだけを変更したいときは、変更したいアイテムをラベリングツールで選択したのち、フォントリングのを使用して必要な変更を行います。

ブールデータの範囲をチェックする

ブール値のエラーを検出することができます。ブールが常に TRUE になることが予想されるときは、ブールのポップアップメニューのサブメニューから**データ範囲→ False の場合中断**を選択します。ブールが常に FALSE になることが予想されるときは、**データ範囲→ True の場合中断**を選択することでエラーを検出することができます。予想外のブール値が発生した場合は、数値範囲の**中断**項目の説明の通り、VI は実行の前または後に中断します。

ブール制御器の機械的動作を構成する

ブール制御器には、6 種類の機械的動作があります。次の図に示すように、ポップアップメニューの**機械的動作**パレットからアプリケーションに適した動作を選択します。これらのパレットの中にある文字のうち、M は制御器を操作する際のマウスボタンの動作を表します。V は制御器の出力値を表し、RD は VI が制御器を読み取るタイミングを表します。



押されたらスイッチ動作は、電灯のスイッチと同じように、操作ツールで制御器をクリックするたびに制御器の値を変更します。この動作は、VIが制御器を読み取る頻度による影響は受けません。

放されたらスイッチ動作は、制御器図形の境界線内でマウスをクリックしてマウスボタンを放した後でのみ制御器の値を変更します。この動作は、VIが制御器を読み取る頻度による影響は受けません。

放されるまでスイッチ動作は、制御器をクリックしたときに制御器の値を変更し、マウスボタンを放すまで新しい値を保持します。（ドアのベルの動作と似ており）マウスボタンを放すと制御器はもとの値に戻ります。この動作は、VIが制御器を読み取る頻度による影響は受けません。

押されたらラッチ動作は、制御器をクリックしたときに制御器の値を変更し、VIが制御器を1回読み取るまでその値を保持します。VIが制御器を1回読み取ると、そのときにマウスボタンを押していてもいなくても制御器はデフォルト値に戻ります。この動作は回路ブレーカの動作とよく似ており、Whileループを停止させたり、あるいは制御器を設定するたびにVIに一回だけ何らかの動作をさせるのに便利です。

放されたらラッチ動作は、制御器の図形の境界線内でマウスボタンを放した後でのみ制御器の値を変更します。VIが制御器を1回読み取ると、制御器はもとの値に戻ります。この動作は、ダイアログボックスボタンやシステムボタンと同じように機能します。

放されるまでラッチ動作は、制御器をクリックしたときに制御器の値を変更し、VIが制御器を1回読み取るか、またはマウスボタンを放すまで（どちらか最後に発生する動作まで）その値を保持します。

ブール制御器とブール表示器のサンプルについては、examples¥general¥controls¥booleans.llb を参照してください。

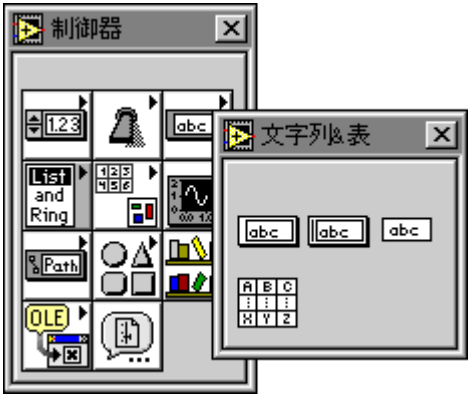
インポートした画像を使用してブールをカスタマイズする

任意のブール制御器またはブール表示器の TRUE および FALSE の画像を取り込むことにより、ブールの独自のスタイルを設計することができます。この手順については、「第 24 章　カスタム制御器と Type Def」の「制御器エディタを使用する」の項を参照してください。

11

文字列制御器と文字列表示器

この章では、文字列制御器と文字列表示器、および表の使用方法について説明します。これらのオブジェクトには、次の図に示す**制御器→文字列 & 表**パレットによりアクセスすることができます。



文字列制御器と文字列表示器を使用する

文字列制御器と文字列表示器を次の図に示します。

文字列制御器のテキストの入力や編集には、操作ツールまたはラベリングツールを使用します。デフォルトでは、新しいテキストや変更したテキストは、テンキーパッドの<Enter>キーを押すか、**ツール**パレットの**入力**ボタンを押すか、または制御器の外側をクリックして編集セッションを終了するまでダイアグラムには渡されません。ただし、ポップアップメニューから**タイプ中に値を更新項目**を選択すると、文字を入力するたびに制御器の値が更新されます。英数字キーボードの<Return>キーを押すと、改行文字が入力されます。

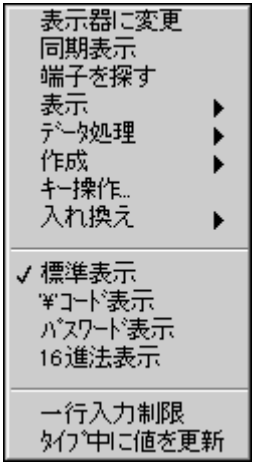
テキストが文字列ウィンドウの右端に達すると、ワードラップ機能によりスペース文字やタブ文字などの、自然に切れる位置でテキストが次の行に送られます。

☞ **注**　VI の実行中は、<Tab> キーを使用して次の制御器に移動することができます。VI の編集中に <Tab> キーを押すと、別のツールに変わります。文字列中にタブ文字を入力したいときは、文字列のポップアップメニューで '¥' コード表示項目を選択して、¥t とタイプします。文字列中に改行を入力するときは、英数字キーボードで <Enter>（**Windows** および **HP-UX**）または <Return>（**Macintosh** および **Sun**）キーを押すか、または文字列のポップアップメニューから '¥' コード表示項目を選択して、¥n とタイプします。文字列中に改行を入力したいときは、文字列のポップアップメニューから '¥' コード表示項目を選択して、¥r とタイプします。'¥' コードの全リストについては、本章の「表示タイプ」の「表 11-1　G の '¥' コード」を参照してください。

文字列の操作については、**オンラインリファレンス→G プログラミング言語→ G 関数と VI リファレンス→文字列関数**のトピックを参照してください。また、examples¥general¥strings.llb に入っているサンプルも参照してください。

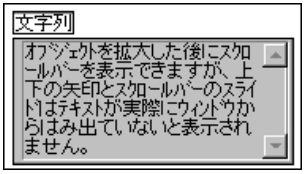
文字列制御器と文字列表示器のメニュー項目

文字列には、次の図に示す文字列のポップアップメニューからアクセスする特殊な機能があります。



スクロールバーメニューの項目

文字列のポップアップメニューの**表示**のサブメニューの**スクロールバー**項目は、文字列制御器をスクロールバーに収まるサイズまで拡大しない限り使用できません。この項目を選択すると、次の図に示すように文字列制御器または文字列表示器上にスクロールバーが表示され、文字列制御器に表示されていないテキストも表示できるようになります。また、この項目を使用すると、テキストの量が多い文字列制御器がフロントパネル上で占めるスペースを最小限に抑えることもできます。スクロールバーがグレーで表示されている場合、文字列の高さを制御器よりも高くすると、スクロールバーが使用できるようになります。



表示タイプ

文字列のポップアップメニューの中央にある各項目を使用すると、文字列ウィンドウに通常通りにデータを表示する、印刷不可能な文字を¥コードに置き換えて表示する、パスワードモードで表示する（それぞれの文字の代わりに * 記号を使用する）、16 進文字で表示する、のいずれかを選択することができます。

標準表示

メニューの**標準表示**項目を選択すると、すべての文字がキーボードから入力した通りに表示されます（ただし、<Tab>キーや<Esc>キーのような表示不可能な文字で作成した特殊文字は除きます）。

円コード（‘¥’）表示

文字列のポップアップメニューから **‘¥’ コード表示**を選択すると、円コード（¥）のすぐ後の文字を表示不可能な文字とみなすようソフトウェアに指示することができます。次の表で、G がこれらのコードをどのように解釈するかを示します。

表 11-1 G の ‘¥’ コード

コード	G の解釈
¥00 - ¥FF	8 ビット文字の 16 進数の値。必ず大文字を使用。
¥b	バックスペース（ASCII BS。\08 と等価）
¥f	用紙送り（ASCII FF。\0C と等価）
¥n	改行（ASCII LF。\0A と等価）
¥r	復帰（ASCII CR。\0D と等価）
¥t	タブ（ASCII HT。\09 と等価）
¥s	スペース（\20 と等価）
¥¥	円コード（ASCII\。\5C と等価）

16 進数文字には大文字を使用し、用紙送りやバックスペースなどの特殊文字には小文字を使用してください。そうすることで、`¥BFare` という文字列を**16進数のBF**の後に**are**という単語が続いているものとみなし、`¥bFare` と `¥bfare` という文字列はバックスペースの後に**Fare**と**fare**という単語が続いているものとみなすよう G に命令することができます。`¥Bfare` という文字列の `¥B` はバックスペースコードではなく、また、`¥Bf` も有効な 16 進コードではありません。このような場合に、¥ コードの後に有効な 16 進コードが一部しかない場合は、G は ¥ コードの後に **0** という文字が続いているものとみなし、`¥B` を **16 進数の 0B** として読み込みます。また、¥ コードの後に有効な 16 進数文字が存在しないときは、G は ¥ コード文字を認識しません。

表示不可能な文字のなかには、**‘¥’ コード表示**を選択しているかいないかに関係なく、改行文字のようにキーボードから文字列制御器に入力できる文字もあります。ただし、表示ウィンドウにテキストが存在するときに ¥ コードモードを有効にすると、G は表示不可能な文字および ¥ コードも表示されるように画面を描画しなおします。

仮に、モードを標準表示に設定して次のような文字列を入力したとします。

```
left
¥right¥3F
```

left の後の改行文字とその後の¥コード文字は、¥コード形式では¥n¥¥と表示されるため、¥コードモードを有効にすると次のような文字列が表示されます。

```
left¥n¥¥right¥¥3F
```

次に、'¥' **コード表示**を選択して次のような文字列を入力したとします。

```
left

¥right¥3F
```

このモードを無効にすると、Gは最初は改行文字として解釈していた¥rを今度は印刷するために、次のような文字列が表示されます。¥3F は疑問符（?）を表す特殊コードで、次のように表示されます。

```
left

ight?
```

ここで再度 '¥'**コード表示**を選択すると、次のような文字列が表示されます。

```
left¥n¥¥right?
```

表示器でも動作は同じです。

これらの例では、モードを切り替えても文字列中のデータそのものに変わりはありません。特定の文字の表示方法だけが変わります。

モードは、プログラムをデバッグしたり、印刷不可能な文字を計測器やシリアルポート、あるいはその他のデバイスに送出したりするのに便利です。

パスワード表示

パスワード表示項目を使用すると、文字列制御器は入力した個々の文字を * で表示します。ただし、ブロックダイアグラムから文字列のデータを読み取る際には、実際にユーザが入力したデータを読み取ります。制御器からデータをコピーする際には、* 文字だけがコピーされます。

16進表示

文字列を英数字ではなく 16 進数の文字で表示するためには、**16 進表示**項目を使用します。'¥' **コード表示**と同様、**16進法表示**もデバッグや計測器と通信を行うのに便利な項目です。

入力を一行に制限

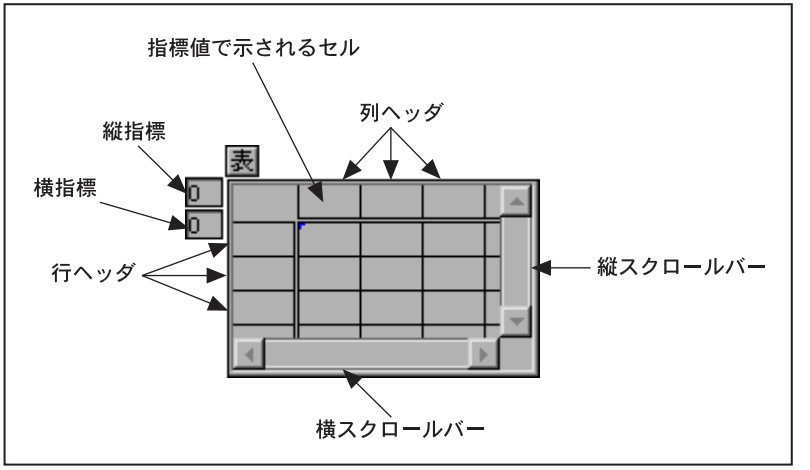
一行入力制限項目は、文字列の入力中に改行文字が入力されないようにします。

入力中の値の更新

ユーザが入力ボタンをクリックした時点あるいはその他の方法で編集作業を終了した時点ではなく、文字を入力するたびに制御器の値が変更されるようにするには、タイプ中に値を更新項目を選択します。この機能は、入力が正確かどうかをチェックしたり、入力を制限したり、あるいはユーザにフィードバックする際に便利です。たとえば、入力を英数字だけに制限する文字列制御器を作成することもできます。

表

表とは、文字列の二次元配列のことです。次の図は、そのすべての機能を表示した表の例を示したものです。



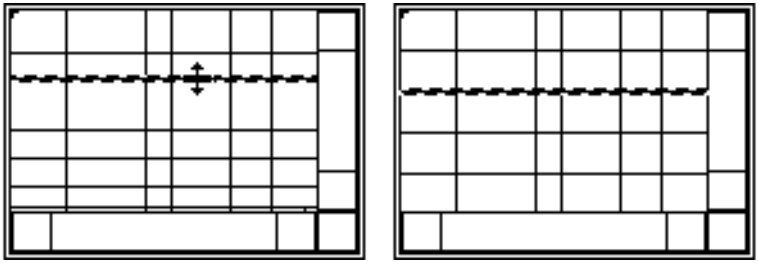
表には行ヘッダと列ヘッダがあり、二重線によってデータと区切られています。フロントパネルに表を配置する際には、ヘッダを入力します。ヘッダは、操作ツールまたはラベリングツールを使用して変更することができます。また、属性ノードを使用してヘッダの更新や読み込みを行うことができます。

指標表示は、表の左上の隅にどのセルが表示されるかを示します。これらの指標は、配列の場合と同じように操作することができます。

表、行、および列のサイズを変更する

サイズ変更ツールを使用すると、どの隅からでも表のサイズを変更することができます。表の行や列は、位置決めツールでそのいずれかの境界線をドラッグして変更することができます。境界線をドラッグできる位置にツールを正しく合わせると、次の図に示すいずれかのドラッグカーソルが表示されます。境界線をクリックし、ドラッグして行または列のサイズを変更します。

<Shift>キーを押しながら境界線をドラッグすると、複数の行あるいは列を同じサイズに変更することができます。サイズを変更したい行または列が、選択した表の領域（青または太い線で囲まれた領域）内にある場合、その表のすべての行または列が同じサイズになります。そのため、次の左の図では高さが不均等な行も、<Shift>キーを使用することで右の図のように高さが均等になります。



データ表の入力と選択

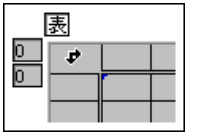
キーボードを使用すると、表に素早くデータを入力することができます。操作ツールまたはラベリングツールでセルの内側をクリックして、データを入力します。

英数字キーボードの<Return>キーは、テキストを入力してカーソルを下のセルに移動します。テンキーパッドの<Enter>キーは、データを入力して入力を終了します。<Shift>キーを押しながら矢印キーを押すと、入力カーソルが隣りのセルに移動します。

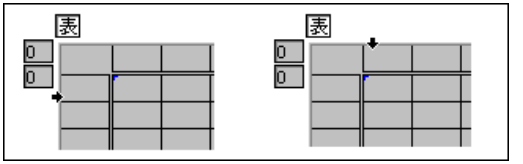
任意のセルを選択するには、操作ツールを使用するか、またはセルをダブルクリックします。<Shift>キーを押しながらクリックし、ドラッグすると選択範囲を拡張することができます。

表の現在の内容の外側に移動すると、表をスクロールしながら選択範囲を拡張します。選択されたセルの周りには、選択されていることを示す境界線が表示されます。表のポップアップメニューの**選択スクロール**項目を使用すると、スクロール機能の有効/無効を切り替えることができます。

また、表のすべてのデータを選択したり、1 つの行または列のすべてのデータを選択することもできます。表のすべてのデータを選択するには、操作ツールを表の左上隅に移動します。次の図に示すように、ツールを正しい位置に移動すると 2 つの矢印を持つ特殊なカーソルが表示されます。その部分をクリックしてデータを選択します。



一行または一列のデータ全体を選択するには、操作ツールを行の左端または列の上端に移動します。この場合も、ツールを正しい位置に移動すると次の図に示すような特殊な矢印カーソルが表示されます。マウスボタンをクリックして、行の右端あるいは列の一番下までドラッグします。

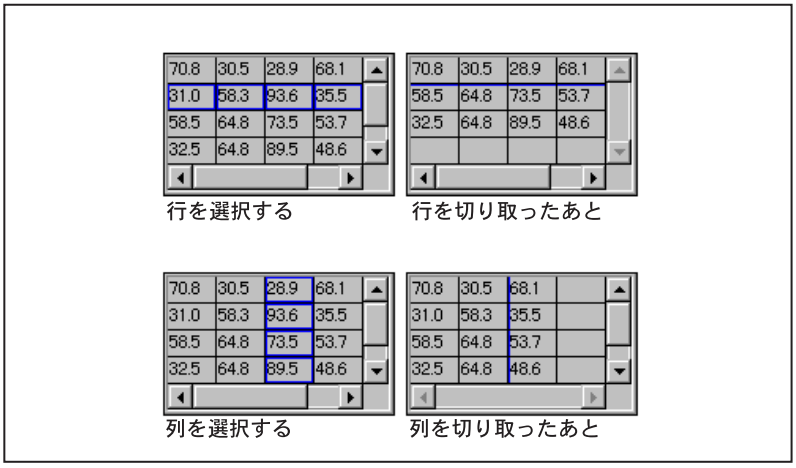


ポップアップメニューの**データ処理**サブメニューの**データをコピーする**、**データを切り取る**、**データを貼り付ける**などの項目を使用すると、データのコピー、切り取り、および貼り付けを行うことができます。

データを一行（または複数行）分切り取ると、次の上段の図で示すように、切り取った行の下にある行すべてが上に移動します。一列（または複数列）のデータを切り取ると、次の下段の図で示すように、その列より右にあるすべての列が左に移動します。

行または列の一部を切り取ると、表からはその行または列の全体が削除されますが、クリップボードには選択した部分だけがコピーされます。

最初に対象を選択せずに貼り付けを行うと、必要に応じて行または列が表に追加されます。



選択したデータの領域を表示するには、**データ処理→選択項目を表示**を選択または選択を解除してそのアイテムの有効/無効を切り替えます。

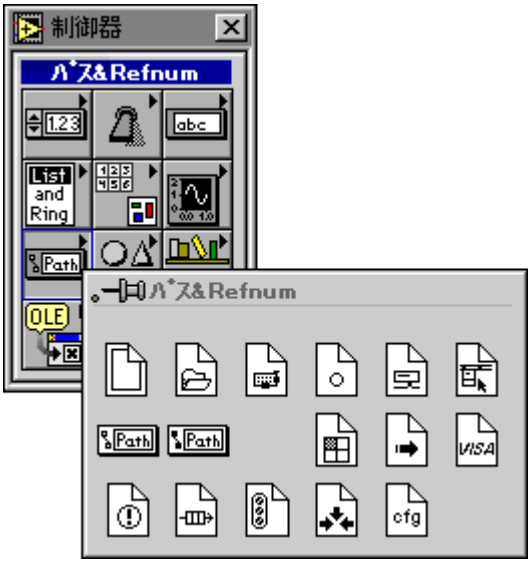
表の行ヘッダと列ヘッダは、表のデータの一部ではありません。表のヘッダは独立した 1 つのデータであり、属性ノードを使用して読み込んだり設定したりすることができます。

ブロックダイアグラムに接続されている限り、表は文字列の 2 次元配列です。そのため、文字列関数は表を操作することができます。これらの関数の使用に関しての詳細は**オンラインリファレンス→G プログラミング言語→G 関数と VI リファレンス→文字列関数**のトピックを参照してください。

12

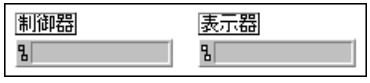
パスと Refnum の制御器および表示器

この章では、次の図に示す**制御器→パス & Refnum** パレットで使用できるファイルパスの制御器と refnum の使用方法について説明します。



パス制御器とパス表示器

パス制御器とパス表示器を次の図に示します。

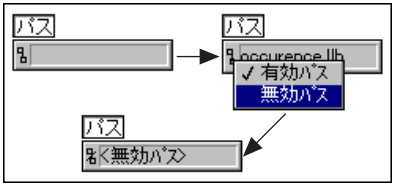



パス記号

パスの制御器と表示器は、各プラットフォームの標準的な構文を使用して、ファイルシステム内のファイルやディレクトリの場所を入力したり表示するために使用します。パスを返すはずの関数が正しく実行されなかった場合、その関数は無効なパスを返します。パス制御器またはパス表示器に無効なパスが表示されると、有効パス記号が無効パス記号に変わり、テキストフィールドにパスが無効であることを示す<無効パス>が表示されます。


無効パス記号

パス記号をクリックしてメニューから**無効パス**項目を選択すると、パス制御器の値を有効パスから無効パスに変更することができます。これを次の図で示します。同様に、無効パス記号をクリックしてメニューから有効パス項目を選択すると、パス制御器あるいはパス表示器の値を無効パスから**有効なパス**に変更することができます。



この**無効パス**という値をデフォルト値として使用すると、VI 上のパスを制御することができます。この方法により、入力されていないパスが検出できます。検出を行うには、ダイアログボックスを使用してパスを選択します。

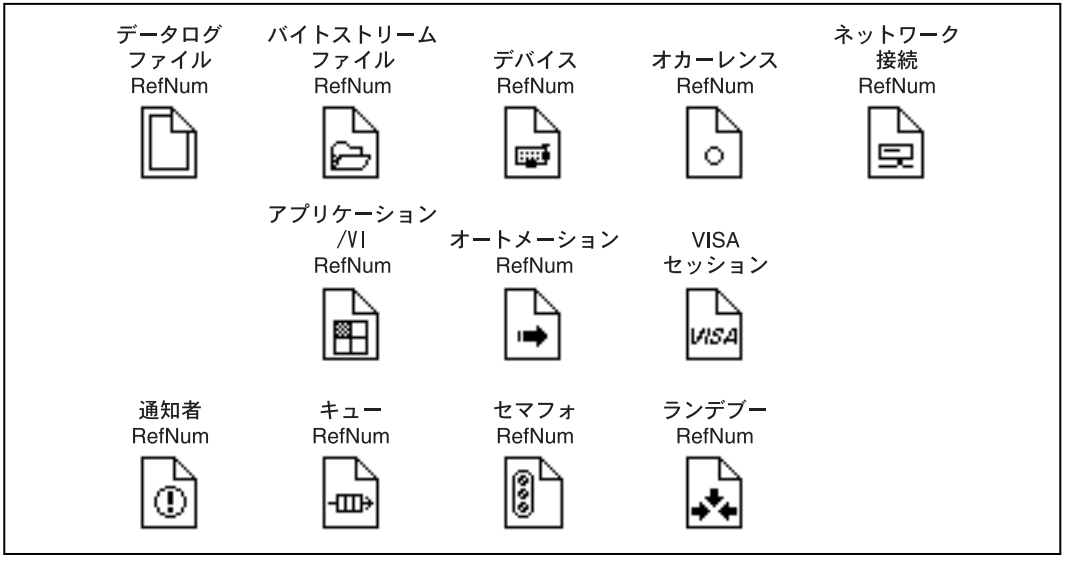
パス制御器の空のパスは、制御器では空の文字列として表示されます。Windows プラットフォーム上でこの空のパスがファイル I/O 関数（特に File/Directory Info VI 関数と List Directory VI 関数）に接続されているときは、空のパスはコンピュータにマッピングされたドライブのリストを参照します。Macintosh プラットフォームでは、空のパスはデスクトップ上のファイルを参照します。UNIX プラットフォームでは、空のパスはルートディレクトリを参照します。

Refnum 制御器と Refnum 表示器

パス & Refnum パレットには、次の図に示すようにいくつかの refnum 制御器が用意されています。refnum とは、ファイルやデバイス，ときには他のマシンへのネットワーク接続などのオブジェクトを識別するための固有 ID です。refnums の多くは、I/O の操作の対象となるオブジェクト（たとえば Read File 関数で読み取るファイルなど）を指定するために使用されます。refnum 制御器は VI に refnum を渡し、refnum 表示器は VI から受け取った refnum を渡します。それぞれの refnum 制御器には、それに対応する表示器が存在します。refnum 表示器を作成するには、まず最初に refnum 制

御器を作成し、次にその制御器のポップアップメニューから**表示器に変更**を選択します。

次の図に、さまざまな種類の refnum を示します。



ファイルを開く際には、開きたいファイルのパスを指定する必要があります。refnum が返され、開くファイルが特定されます。この **refnum** は、開いたファイルに関して以降行われるすべての操作に使用されます。この refnum は、ファイルを開くたびに毎回そのファイルを特定するために生成される固有番号と考えることができます。ファイルを閉じると、refnum はファイルから切り離されます。前の図に示す**制御器→パス** & Refnum パレットにある refnum について以下で説明します。

ファイルの refnum には、データログファイル用とバイトストリーム用の 2 種類の refnum があります。

- **データログファイル** RefNum ― データログファイルは固有の構造を持っているため、データログファイル RefNum は refnum だけでなく、発呼者 VI に対する（または発呼者 VI からの）ファイルタイプに関する説明も渡します。データログファイル RefNum は、クラスタと同様サイズを変更することができます。制御器は、ファイルの構造を定義する refnum の内部に配置します。番号を含むファイルに対しては番号を含むデータログ refnum を作成します。ファイル内のレコードが一組の番号を含む場合は、refnum の内部にクラスタを配置して、クラスタの内部に 2 つの数値制御器を配置します。

- **バイトストリームファイル RefNum** — バイトストリームファイル RefNum は、テキストファイルまたはバイナリファイルいずれかのバイトストリームファイルに対して適用されます。通常この refnum は、ある VI の中でファイルを開くかまたは作成し、そのファイルに対する I/O を別の VI で実行したいときに使用します。I/O を実行する VI のフロントパネルには refnum 制御器が、ファイルを開くまたは作成する VI のフロントパネルには refnum 表示器が必要になります。

その他の refnum には項目がないため、簡単に使用できます。

- **デバイス RefNum** — デバイス RefNum は、Macintosh 専用のデバイス I/O 関数で使用されます。この refnum は、1 つの VI のデバイスでファイルを開き、別の VI のデバイスで I/O を実行したい場合に使用されます。

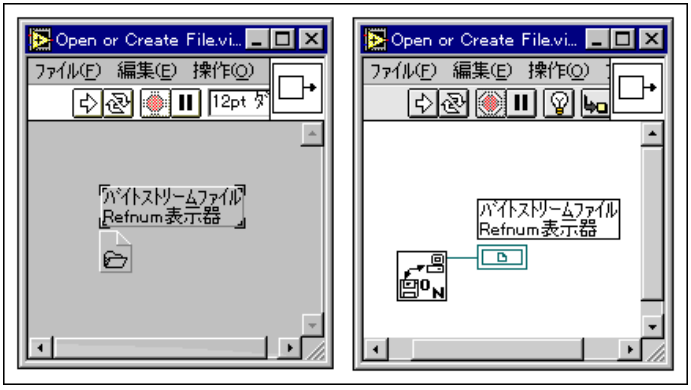
☞ **注**　**フロントパネル上でデバイス RefNum 制御器が必要になることはほとんどありません。これらの制御器は、G のシリアルデバイスドライバにアクセスするために** `vi.lib` **の計測器 I/O VI で使用されます。デバイス RefNum の制御器が必要となるのは、Macintosh 用の特殊な用途の G デバイスドライバを作成する場合に限られます。**

- **オカーレンス RefNum** — オカーレンス RefNum は、オカーレンス関数で使用されます。この refnum は、ある VI でオカーレンスを生成し、別の VI でオカーレンスを設定したりあるいは待つ場合です。
- **ネットワーク接続 RefNum** — ネットワーク接続 RefNum は、TCP/IP VIs で使用されます。通常この refnum は、ある VI でネットワーク接続を確立し、別の VI でそのネットワーク接続に対する I/O を実行したい場合に使用されます。
- **アプリケーションまたは VI RefNum** — アプリケーションまたは VI refnum 制御器は、VI Server 関数で使用されます。この refunum の制御器は、LabVIEW アプリケーションまたは LabVIEW 内の VI へのリファレンスを開き、そのリファレンスをパラメータとして別の VI に渡したいときに使用されます。これらのいずれかの refnums をいずれかの VI サーバ関数に渡すことにより、アプリケーションおよび特定の VI の動作を制御することができます。refnum 制御器のサブメニューで **VI サーバクラスを選択**からアプリケーション refnum、VI refnum、または厳密に類別化された VI refnum のいずれかを指定することができます。厳密に類別化された VI refnum は、VI のコネクタペーンを含むデータタイプ情報を保有しているため、Call By Reference Node 関数を使用して動的にロードされる VI を呼び出すために使用することができます。VI は、使用したいコネクタペーンの入っているディスクから **VI サーバクラスを選択→参照...** を使用して選択することができます。また、VI アイコン / コネクタペーンまたはサブ VI アイコンをダイアグラムまたは階層ウィンドウから VI refnum 制御器にドラッグアンドドロップして、厳密に類別化された VI refnum の制御器のタイプを指定することもできます。

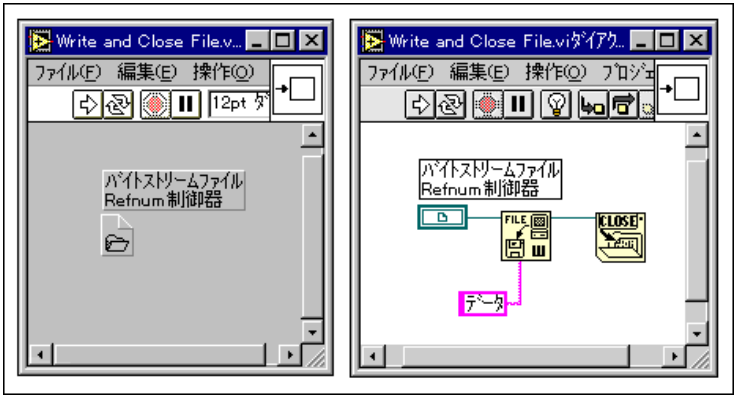
通常VI refnum制御器が1つのVIから別のVIにVI refnumを渡すのに対し、厳密に類別化されたVI refnumの制御器は、厳密に類別化されたVIへリファレンスを受け渡す際にOpen VI Reference関数に渡すタイプ指定子としても必要になります。その場合、refnum制御器の値は重要ではありません。関数ではタイプだけが使用されます。

- **オートメーション RefNum** ― オートメーション RefNum は、オートメーションオブジェクトへのリファレンスです。
- **VISAセッション** ― VISAセッションは、VISAリソースへのリファレンスです。
- **ノーティファイアRefNum** ― ノーティファイアRefNumは、通知VIに対して使用されます。通常は、1つのVIで通知者（ノーティファイア）を作成し、別のVIで通知待ちや通知の送出を行いたい場合に使用します。
- **キュー RefNum** ― キューRefNumは、キューVIに対して使用されます。通常は、1つのVIでキューを作成し、別のVIでキュー要素の挿入や削除を行いたい場合に使用します。
- **セマフォRefNum** ― セマフォRefNumは、セマフォVIに対して使用されます。通常は、1つのVIでセマフォを作成し、別のVIでそのセマフォを使用したり解放したいときに使用します。
- **ランデブーRefNum** ― ランデブーRefNumは、ランデブーVIに対して使用されます。通常は、1つのVIでランデブーを作成し、別のVIでランデブー待ちをしたいときに使用します。

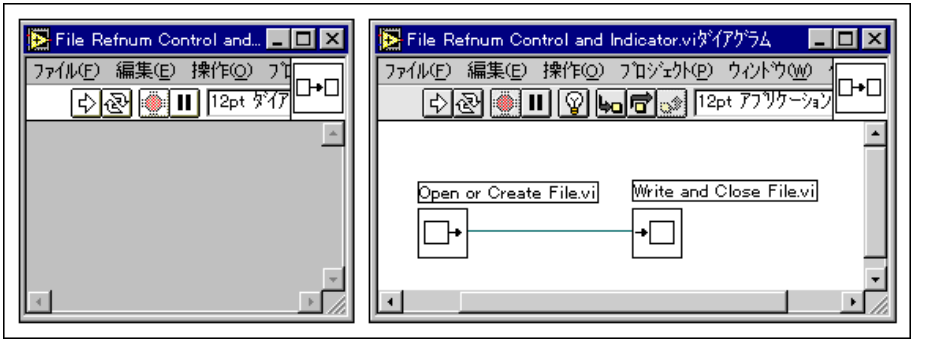
次の図は、ファイルを開くかまたは作成し、別のVIでそのファイルにアクセスするために使用できるバイトストリームファイルrefnum表示器によりバイトストリームファイルrefnumを渡すVIのフロントパネルとブロックダイアグラムを示したものです。



次の図は、バイトストリームファイル refnum 制御器により VI に渡されたバイトストリームファイル refnum で指定されたファイルにデータを書き込み、そのファイルを閉じる VI のフロントパネルとブロックダイアグラムを示したものです。



次の図は、1 つの VI が開いたまたは作成したファイルにアクセスするために使用されるバイトストリームファイル refnum を、そのファイルにデータを書き込み、そのファイルを閉じる別のサブ VI に渡す VI のフロントパネルとブロックダイアグラムを示したものです。



☞ **注**　VI に refnum を渡す場合、あるいは VI から refnum を渡す場合には、refnum 制御器または refnum 表示器を使用してください。

13

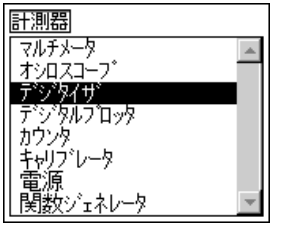
リストとリングの制御器および表示器

この章では、次の図で示す**制御器→リスト & リング**パレットを通じてアクセスするリストボックスとリングの制御器および表示器について説明します。

リスト & リングパレットには、テキストリング、メニューリング、ダイアログリング、画像リング、テキストと画像リング、一項目選択リストボックス、多項目選択リストボックス、および列挙タイプの 8 つの制御器があります。この章では、リング制御器、リストボックス制御器、数値タイプ制御器の順に説明します。リストボックス制御器およびリング制御器の属性についての詳細は「第22章 属性ノード」を参照してください。

リストボックス制御器

リストボックス制御器は、項目のリストを表示します。次の図に例を示します。リストボックス制御器には、すなわち 1 つの項目だけを選択できる一項目選択タイプと、2つ以上の項目を選択できる多項目選択タイプの2つのタイプがあります。



リストには、各項目をテキストで表示する以外に、各項目の横にいくつかの異なる記号のなかから1つの記号を表示することができます (たとえば、**保存**ダイアログボックスで表示されるディレクトリとファイルの記号が異なるのと同じです)。また、個々の項目を選択不能にしたり、項目と項目の間に区切り線を付けることもできます。現在選択されている項目が単数なのか、または複数であるのかは、制御器の値を読み取ることによって検出します。属性ノードを使用すると、ユーザがどの項目をダブルクリックしたのかを検出することもできます。さらに、属性ノードを使用して項目の文字列や記号を設定したり、項目を選択不能にするかしないかを設定することもできます。

項目のリストを作成する

リストボックスをフロントパネルに配置した時点では、リストボックスにはまだ項目が入っていません。項目のリストを作成するには、編集モードでラベリングツールを使用して項目を入力してするか、または実行モードで属性ノードを使用します。1 つの項目の入力したら、そのつど改行を入力します。VI を作成する際に設定したい項目があらかじめ決まっている場合は、編集モードでその項目を入力します。内容の設定が実行時にしかできない場合 (たとえばファイルのリストなど) は、属性ノードを使用します。

リストボックスの項目を選択する

リストボックスの選択項目には、マウスを使用する、矢印キーを使用する、あるいは選択したい項目の名前の一部をタイプする、の3通りの方法があります。

- マウス — 操作ツールを使用して項目を選択します。多項目選択のリストボックスでは、2つめ以降の項目は<Shift>キーを押しながらマウスボタンをクリックして選択します。
- 矢印キー — 矢印キーを使用して項目を選択します。現在ハイライトで表示されている項目を選択項目リストに追加するには、スペースバーを押します。現在選択されている項目の選択を解除するには、矢印キーを使用して選択を解除したい項目に移動したのち、スペースバーを押します。
- 選択したい項目の名前の一部をタイプする — 項目の名前の一部をタイプして項目を選択します。左または右の矢印キーを使用して、タイプした文字と一致する前または次の項目に移動することもできます。

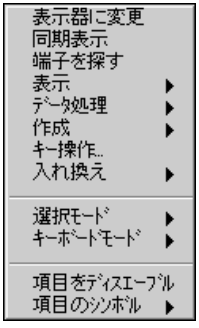
リストボックスのデータタイプ

一項目選択のリストボックスのデータタイプはInt32です。一項目選択のリストボックスのデータ値は、現在選択されている項目を表す番号で、最初の項目の番号は0になります。項目が選択されていないときは、値は–1になります。

多項目選択のリストボックスはInt32の配列で、配列中の値は現在選択されている項目を表します。項目が選択されていないときは、値は空の配列になります。

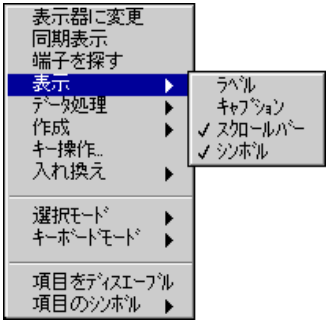
リストボックスのポップアップメニュー項目

リストボックスのポップアップメニューには、次の図に示す項目が表示されます。この項では、リストボックス固有の項目について説明します。



表示

次の図のように、ポップアップメニューの**表示**から表示したいリストボックスのコンポーネントを選択します。ラベル、スクロールバー、および記号などは、それぞれ表示するしないを指定することができます。

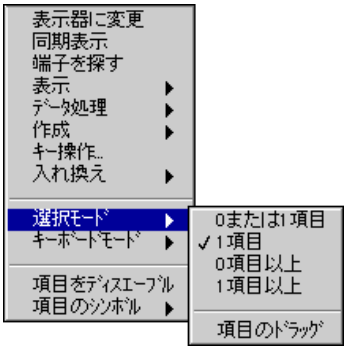


表示→記号を選択すると、リストボックスのリストの左側に記号の表示スペースが追加されます。その例を次の図に示します。最初は、どの項目にも記号は表示されません。記号を追加するには、**項目の記号**または**属性ノード**を使用します。

選択モード

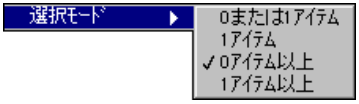
ポップアップメニューの**選択モード**は、1 度に選択できる項目の数を指定するために使用します。

一項目選択のリストボックスの選択モード項目は、次の図に示すように、リストボックスを 0 または 1 つの選択、あるいは 1 つの選択のいずれかをサポートするよう設定します。



1 項目（デフォルト設定）を選択した場合は、かならずリストボックスの 1 つの項目が選択（ハイライト表示）されます。別の項目をクリックすると、その項目が新たに選択（ハイライト表示）され、前の項目の選択が解除されます。**0 または 1 項目**を選択した場合も操作方法は同じですが、<Shift> キーを押しながら現在選択されている項目をクリックしてその項目の選択を解除し、どの項目も選択されていない状態にすることができます。**項目のドラッグ**を使用すると、リストボックスの項目をマウスでドラッグして移動することができます。

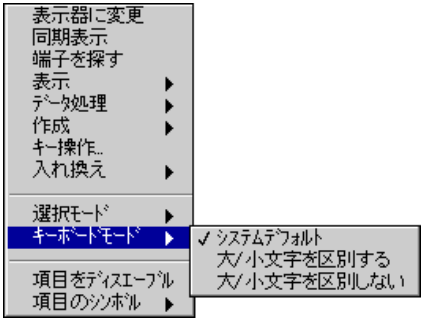
多項目選択のリストボックスの場合は、一項目選択のリストボックスの選択モード項目のほかに、次の図に示すように複数の項目を選択するように指定することもできます。



多項目選択のリストボックスのデフォルト設定の選択モードは、**0 項目以上**（0 個以上の項目の選択が可能）です。また、**1 項目以上**を選択すると、少なくとも 1 つの項目を選択しなければならないように指定されます。

キーボードモード

先頭の文字をタイプして項目を選択する際に、大文字小文字の区別をどのように処理するかを指定するためには、次の図に示すようにポップアップメニューの**キーボードモード**を使用します。



大 / 小文字を区別するを選択した場合は、選択したい項目のテキストの通りに大文字と小文字を使い分けて入力した場合にのみ、その項目が選択されます。**大 / 小文字を区別しない**を選択した場合は、大文字と小文字の区別は無視されます。**システムデフォルト**（リストボックスのデフォルト設定）を選択した場合は、各プラットフォームでの他のリストボックスと同じ動作になります。Windows と Macintosh のリストボックスでは、大文字と小文字を使い分ける必要はありません。UNIX のリストボックスでは、大文字と小文字の使い分けが必要です。

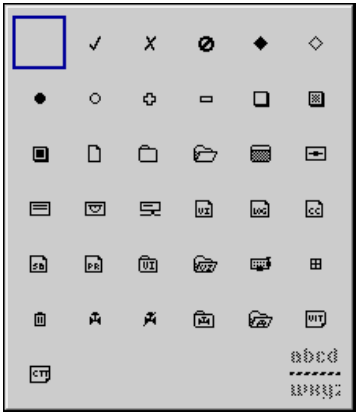
項目をディスエーブル

リストボックスの項目は、その項目をポップアップし、ポップアップメニューで**項目をディスエーブル**を選択することにより、選択不能にすることができます。項目を選択可能にするには、その項目をポップアップして**項目をイネーブル**を選択します。

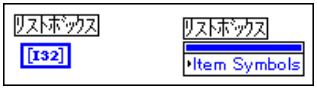
項目をプログラム上で選択不能または選択可能にするためには、次の図に示す Disabled Items の属性ノード（項目の配列）を使用します。

項目の記号と区切り線

項目をポップアップし、**項目の記号**サブメニューからいずれかの項目を選択することにより、リストボックスの項目に対して記号を設定することができます。右下の隅にある項目を選択すると、リストの項目が区切り線に差し替えられます。



リストの項目の記号をプログラム上で設定するためには、次の図に示すItem Symbolsの属性ノード（項目の記号の配列）を使用します。

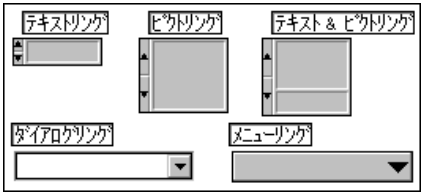


項目の記号は、**関数→数値→その他の数値定数**パレットのListbox Symbol Ring定数を使用して指定することができます。

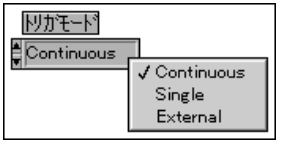
項目の記号に–1という値を使用すると、リストボックスにはその項目の名前の代わりにグレーの区切り線が表示されます。

リング制御器

リングは、数値を文字列または画像（またはその両方）と関連付ける特殊な数値オブジェクトです。次の図に、さまざまな種類のリングを示します。



リングは、特にトリガモードのような互いに相容れない項目の選択に便利です。たとえば、次に示すようにContinuous、Single、あるいはExternalのいずれかのトリガを選択できるようにしたいときなどに使用します。



前の例では、`Trigger Mode`というラベルに3つのモードデスクリプタがあり、それぞれのデスクリプタはリングのテキスト表示領域に順次表示されます。Gは、各項目を環状リストの形式に並び替え、1度に1項目だけが表示されるようにします。各項目は0から*n*–1までの値を持っています（*n*は項目の数。ここでは3）。ただし、リングの値は、その数値データの範囲内のどんな値でもかまいません。リングは、2以上の値に対しては最後の項目（上記の例では`External`）を表示し、0以下の値に対しては最初の項目（上記の例では`Continuous`）を表示します。

ポップアップして**表示→デジタル表示**を選択すると、リングの現在の項目に関連付けられている数値を表示することができます。

ユーザは、このリングを使用すると簡単なリストから項目を選択することができ、その際個々の項目の値を覚える必要はありません。選択した項目の値はブロックダイアグラムに渡され、ブロックダイアグラムでは選択された項目を実行するCaseストラクチャ（条件コード）のケースを選択するといったことが可能になります。

リング制御器では、2通りの方法で項目を選択することができます。次の項目または前の項目に移動するには、増分ボタンを使用します。その際、マウスボタンを押し続けるとリストの項目が次々と表示されます。あるいは、操作ツールでリングをクリックしてから、選択したい項目を表示されたメニューから選択することにより、項目を直接選択することもできます。

ただし、メニューリングの外観や動作がプルダウンメニューと同じになるので、増分ボタンは表示されません。この場合は、リングをクリックし、表示されたメニューから項目を選択することになります。

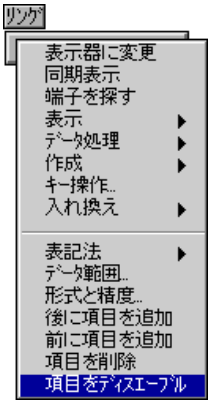
リングにテキスト項目を追加する

新しいテキ文字列には、値が0の1つの項目と、空の文字列を含む表示領域が存在します。リングのテキスト表示領域には、ラベルの場合と同様、ラベリングツールを使用してテキストを入力したり、編集したりすることができます。入力を終了する場合は、<Enter>キー（**Windows**および**HP-UX**）または<Return>（**Macintosh**および**Sun**）キーを押すか、あるいはテキスト表示領域の外側をクリックします。テキ文字列のポップアップメニューの**後に項目を追加**は、現在の項目の後に新しい空の項目を作成します。**前に項目を追加**は、現在の項目の前に新しい項目を挿入します。たとえば、項目0の編集中に**後に項目を追加**を選択すると、項目1が作成され、新しい項目のテキストの入力が可能になります。また、1つの項目を入力したあとで<Shift-Enter>（**Windows**および**HP-UX**）または<Shift-Return>（**Macintosh**および**Sun**）を押しても、新しい項目に進むことができます。

どのリングでも、項目を新しく挿入すると、挿入ポイントより後ろの項目の値はそれぞれ1つずつ繰り上げられます。たとえば、項目4の後に項目を1つ挿入すると、挿入した項目は項目5になり、それまでの項目5が項目6に、項目6は項目7になります。また、項目4の前に項目を1つ挿入すると、挿入した項目が項目4になり、それまでの項目4が項目5に、項目5は項目6になります。

項目を削除する場合は、リングのポップアップメニューの**項目を削除**コマンドを使用します。項目追加コマンドと同様、この場合も項目の数値は自動的に調節されます。

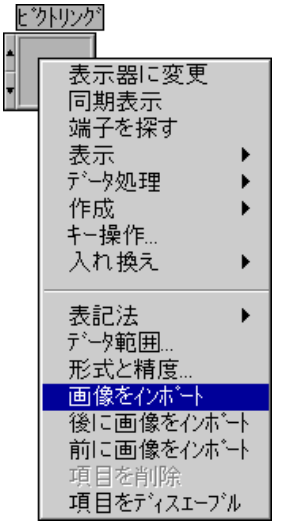
編集中にリングの項目を選択不能にしたいときは、選択不能にしたい項目を選択します。次に、リング制御器をポップアップし、次の図に示すように**項目をディスエーブル**を選択します。



編集中に選択不能にした項目を選択可能にしたいときは、ポップアップして**項目をイネーブル**を選択します。リングの項目をプログラム上で選択不能または選択可能にしたいときは、次の図に示す**項目をディスエーブル**の属性ノード（指標の配列）を使用します。

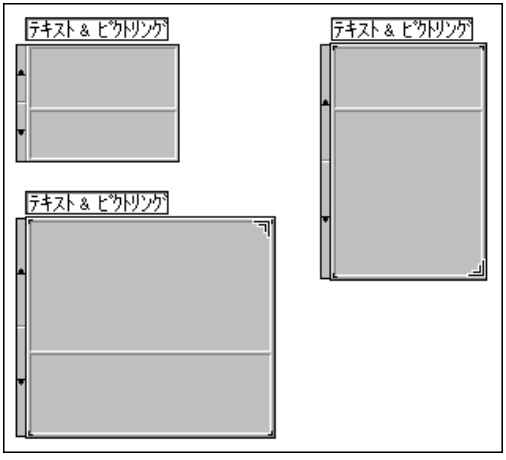
リングに画像項目を追加する

新しい画像リングあるいはテキストと画像リングには、空の画像表示領域を持つ項目が 1 つあります。画像を画像リングにインポートするためには、あらかじめ画像をクリップボードにコピーしておく必要があります。クリップボードに画像をコピーしたら、画像リングをポップアップして**画像をインポート**を選択します。新たな画像を追加したいときは、画像をクリップボードにコピーしたのち、次の図に示すように画像リングをポップアップして**前に画像をインポート**または**後に画像をインポート**を選択します。



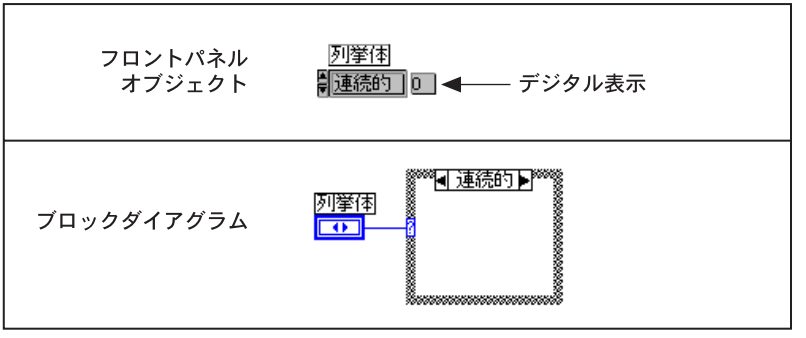
テキストと画像リングのサイズやテキストを変更する

他の制御器と同様に、テキストと画像リングもサイズを拡大してスペースを増やすことができます。また、サイズを縮小することもできます。テキストと画像リングの下のどちらか一方の角を使用してサイズを変更すると、テキスト表示領域の高さが変わります。テキストと画像リングの上のどちらか一方の角を使用してサイズを変更すると、画像表示領域の高さが変わります。この2つのサイズ変更機能を次の図で示します。



列挙体制御器

列挙体制御器は、テキストリング制御器とよく似ています。ただし、列挙体のデータがCaseストラクチャに接続されている場合は、ケースは数字ではなく文字列、すなわち項目のテキストを表示します。列挙体のデータタイプは符号なしのバイト整数、符号なしの2バイト整数、または符号なしの4バイト整数のいずれかで、**表記法**パレットから選択できます。次の図に例を示します。



リング制御器の代わりに列挙体を使用する利点は、サブVIのコネクタペーン上で列挙体を使用し、ポップアップして定数、制御器、または表示器を作成することによって、同じ文字列を持つ列挙体を作成できることです。

列挙体に項目を入力する場合は、リングに項目を入力する場合と同様、**後に項目を追加**または**前に項目を追加**を使用します。または、ラベリングツールを使用することもできます。新しい項目を入力するためには、<Shift-Enter>（**Windows** および **HP-UX**）または <Shift-Return>（**Macintosh** および **Sun**）を押します。項目の入力が終了したら、列挙体の外側の任意の場所をクリックします。

列挙体をCaseストラクチャに接続する場合、そのCaseストラクチャには列挙体のそれぞれの項目を処理するためのケースがなければなりません。Caseストラクチャにデフォルトのケースがある場合、デフォルトのケースは他のケースによって明示的に処理されない項目を処理します。

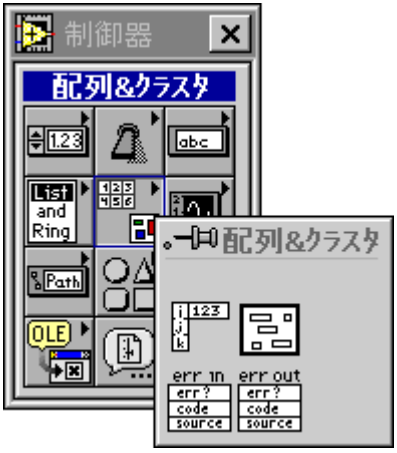
IncrementおよびDecrement以外のすべての算術演算は、列挙体を符号なしの数値と同じように処理します。Incrementは、最後の列挙を最初に増分し、Decrementは最初の列挙を最後に減分します。それ以外の演算では、列挙体は符号なしの値として扱われます。符号付きの整数を強制的に列挙に接続する場合は、負の数字が最初の列挙に接続され、範囲外の正の数字が最後の列挙に接続されます。符号なしの整数は、範囲外の場合、常に最後の列挙に接続されます。

数値を列挙体表示器に接続した場合は、数字は最も近い列挙項目に変換され、範囲外の数字は上記の方法で処理されます。列挙体制御器を数値に接続した場合は、値は列挙体の指標になります。列挙体を列挙体表示器に接続するためには、列挙項目が一致している必要がありますが、表示器には列挙体の項目以外の項目が含まれていてもかまいません。

14

配列とクラスタの制御器および表示器

この章では、配列とクラスタの使用方法について説明します。配列とクラスタには、次の図で示すように**制御器→配列＆クラスタ**パレットからアクセスできます。



`examples¥general¥arrays.llb`のサンプルを参照してください。

ヘルプ→オンラインリファレンス→関数と VI リファレンスを使用すると、G の関数の説明を参照することができます。G の関数の多くは、スカラだけでなく配列にも使用できます。「配列**の関数**」と「クラスタ**の関数**」のトピックでは、配列およびクラスタの操作を目的とした専用の関数について説明しています。

配列

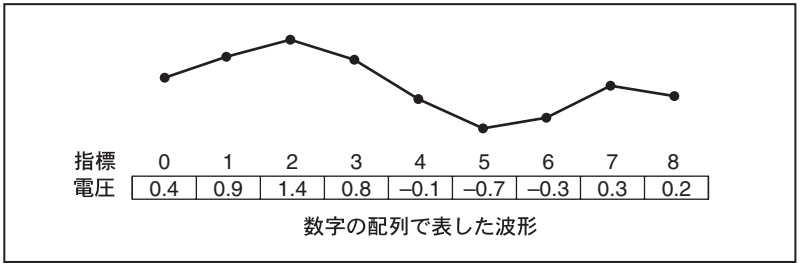
配列とは、さまざまなタイプのデータ要素を集めた固定サイズの集合体であるクラスタとは異なり、同じタイプのデータ要素を集めた可変サイズの集合体です。配列の要素には**指標**が付いていて、配列の指標を指定することにより配列の個々の要素にアクセスすることができます。**指標**は0から始まります。すなわち、次の表で示すように、*n* を配列中の要素の数とすると、指標の範囲は0から*n*–1となります。

指標	要素数が11個の配列
0	Melissa
1	Greg
2	Gregg
3	Don
4	Duncan
5	Thad
6	Dean
7	Stepan
8	Kate
9	Mary
10	Mark

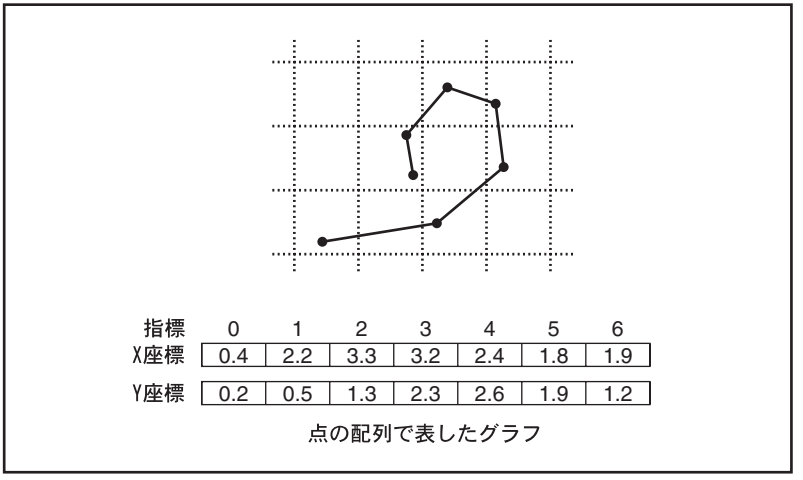
配列の簡単な例として、名前のリストがあります。このようなリストは、Gでは次図のように文字列の配列として表されます。

弘美	恵美	和子	幸雄	直之	健哉	寛	茂	康史	美智子	薫
文字列配列の名前リスト										

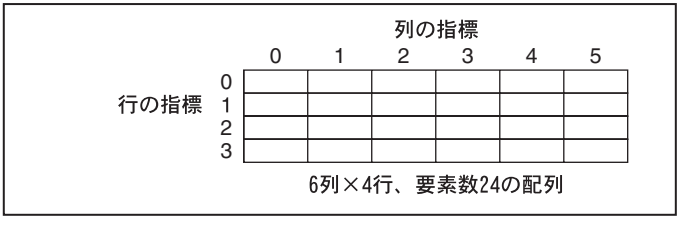
別の例として、次の図のような数値の配列として表される波形があります。この配列では、連続した個々の時間点における電圧値が連続した個々の要素になります。



さらに複雑な例として、座標の配列として表されるグラフがあります。次の図に示すように、各座標はX座標とY座標を表す2つの数字で定義されます。

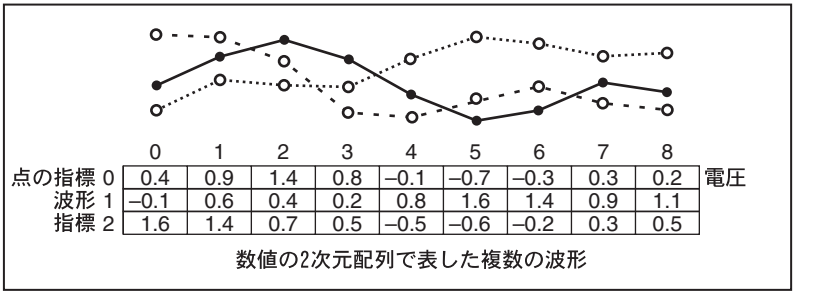


前に示した例は、いずれも**1次元**（1D）の配列です。**2次元**（2D）の配列では、要素を特定するために列の指標と行の指標の2つの指標が必要になります。これらの指標はいずれも0から始まります。この場合、配列はN列×M行の配列と呼ばれ、次の図に示すように要素の数はN × M個になります。



簡単な例として、チェスのボードがあります。チェスのボードには8つの列と8つの行があり、合計64個のマスがあります。それぞれのマスには1つのチェスの駒を置くことができますが、何もなくてもかまいません。チェスのボードは、文字列の2次元配列として表すことができます。個々の文字列は、ボード上のそれぞれのマスに置かれている駒の名前（マスに駒がないときは空白文字列）になります。このほかにも、よく目にする例としてカレンダーや電車の時刻表、テレビの画像などがあります。テレビの画像は、画面上の各点の輝度を表す数字の2次元配列として表されます。コンピュータのユーザがよく目にする例としては、数字の行と列、公式、テキストを使用する表計算プログラムがあります。

1次元配列の例は、いずれも2次元配列に一般化することができます。次の図は、いくつかの波形を数字の2次元配列として表したものです。行の指標は波形を特定し、列の指標は波形上の点を特定します。



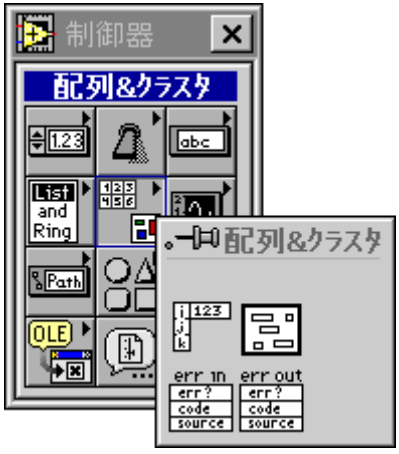
数値の2次元配列で表した複数の波形

配列の次元の数は任意ですが、要素を特定するためには各次元ごとに1つの指標が必要になります。配列の要素のデータタイプは、配列を含むクラスタや、さまざまなタイプを含むクラスタでもかまいません。配列の要素に配列を使用することはできません。代わりに、多次元配列や、配列を含むクラスタで構成される配列を使用してください。

よく似たデータの集合体を処理する場合は、配列の使用を検討してみる必要があります。計算やI/O関数を繰り返し実行する際には、配列が役立つ場合があります。Gには配列の関数やVIが多数用意されているため、配列を使用することでアプリケーションを小型化、高速化、簡略化できます。

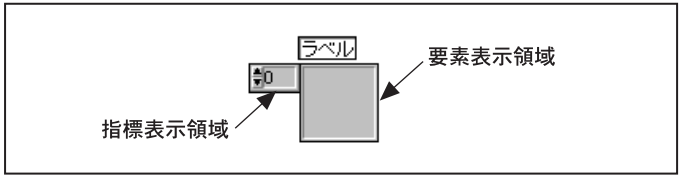
配列制御器を作成する

配列制御器を作成する場合は、次の図に示すように、まず最初に**制御器→配列＆クラスタ**から配列を選択します。



配列制御器や配列表示器は、前図で示した**配列＆クラスタ**パレットから選択した**配列シェル**を、有効な**要素**（数値、ブール、文字列、パス、refnum、またはクラスタ）と組み合わせることによって作成します。要素に別の配列やチャートを使用することはできません。また、要素がグラフのときは、最上層に配列ではなくクラスタを含むグラフデータタイプだけが有効となります。

配列＆クラスタパレットから配列を選択すると、次の図に示すようにフロントパネルに配列シェルが配置されます。



新しい配列シェルには、**指標表示**領域が 1 つ、**要素表示**領域が 1 つ、およびラベル（任意）があります。

配列のタイプを定義する方法は 2 通りあります。使用したいタイプの制御器または表示器を要素表示ウィンドウにドラッグするか、あるいはフロントパネル上でポップアップし、制御器を選択し、それを要素の表示領域にドラッグすることによって制御器または表示器を直接配置します。どちらの場合も、挿入した制御器または表示器によって空白の表示領域が埋められます。

たとえば、ブールの配列を定義したいときは、図 14-1 から図 14-4 で示すように、ポップアップしてブール制御器を選択し、それを要素の表示領域にドラッグします。

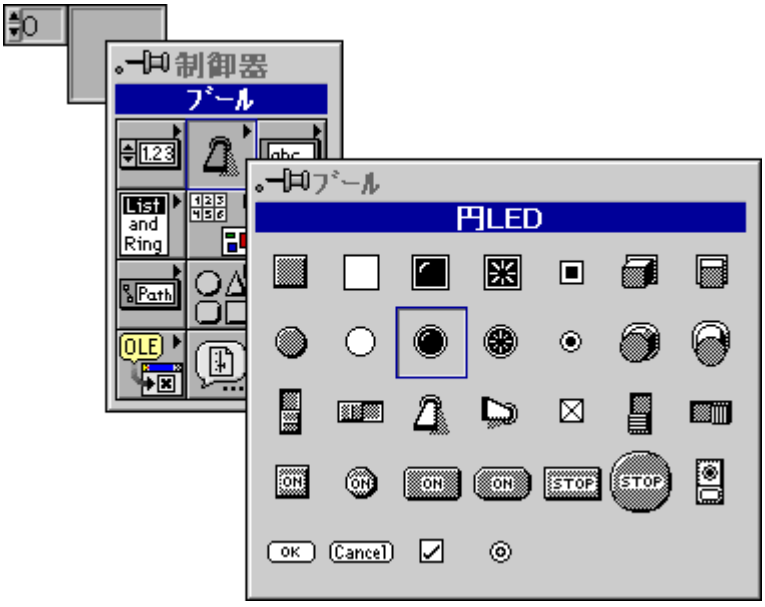


図 14-1 要素の表示領域をポップアップしてブール制御器を選択する。

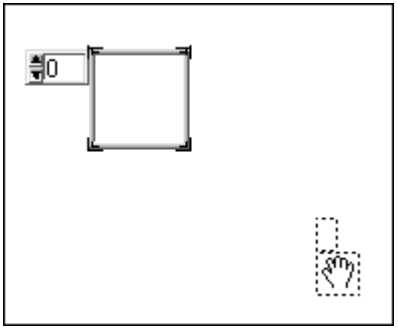


図 14-2 マウスボタンを放す。メニューが消え、カーソルがウィンドウスクロールのアイコンに変わる。

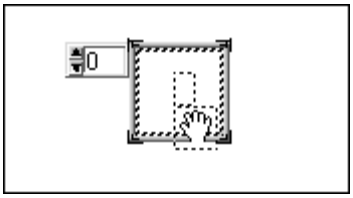


図 14-3 ウィンドウスクロールアイコンを要素の表示領域にドラッグする。

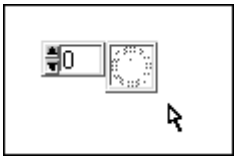
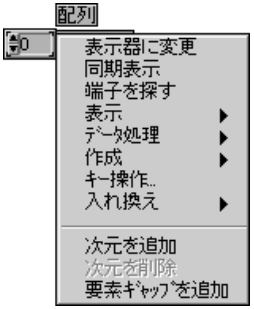


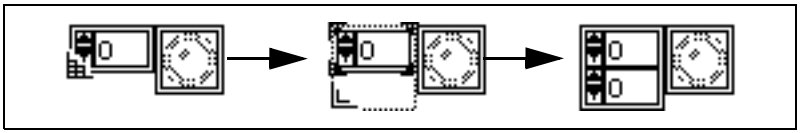
図 14-4 マウスボタンをクリックすると、表示領域に制御器が配置される。

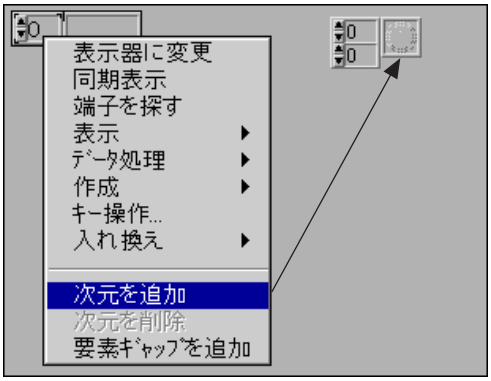
指標表示のポップアップメニューを次の図に示します。



配列の次元

新しい配列には、次元が 1 つと指標表示領域が 1 つあります。指標の表示領域は垂直方向に拡大したり、指標の表示領域のポップアップメニューで**次元を追加**オプションを選択することができます。また、指標の表示領域を垂直方向に縮小したり、**次元を削除**を選択して次元を削除することもできます。次元を追加すると、そのつど新たな指標の表示領域が表示されます。指標の表示領域のサイズの 2 通りの変更方法を次の図で示します。



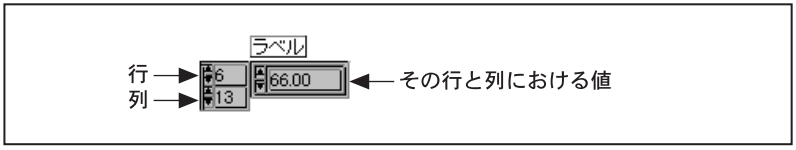


配列の次元の数は、データの中のある項目を特定する際にいくつの識別子を使用するかを決定するものです。たとえば、本の中のある単語を特定する場合には、192 ページの 28 行目の 6 番目の単語という言い方をします。このとき、192、28、6 という数字が単語を特定するための指標であるとすると、この本は単語の 3 次元（3D）配列とみなすことができます。また、図書館で一冊の本を探すためには、どの階のどの列のどのブックケースのどの本棚のどの位置にあるかを特定する必要があります。これを配列で表すと、5 つの次元を使用することになります。したがって、図書館の中でのある単語の場所を表すためには、8 つの指標が必要になります。

配列に次元を追加すると、ワイヤの形や、厚み、色が変わる点に注意してください。詳しくは、「第18 章　ブロックダイアグラムを配線する」を参照してください。

配列の指標表示

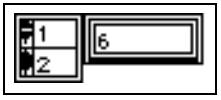
2次元の配列を行列として考えると、上の表示部が行の指標、下の表示部が列の指標になります。次の図に示すように、右側の合成表示は指定された位置の値を表示します。



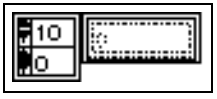
たとえば、次の図に示すような値を持つ3つの行と4つの列からなる配列があるものとします。

0	1	2	3
4	5	6	7
8	9	10	11

行と列の指標はそれぞれ0を基準とし、最初の列は列0、2番目の列は列1というように指標が付けられます。次の図で示すように指標の表示を行1、列2に変えると、6という値が表示されます。



範囲外の値を表示しようとすると、値の表示領域はグレーに変わり、現在値が定義されていないことを示します。たとえば、上に示した配列の行10の要素を表示しようとすると、次の図に示すように値の表示領域がグレーで表示されます。

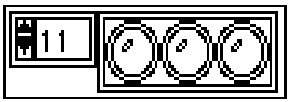


配列を単一要素形式または表形式で表示する

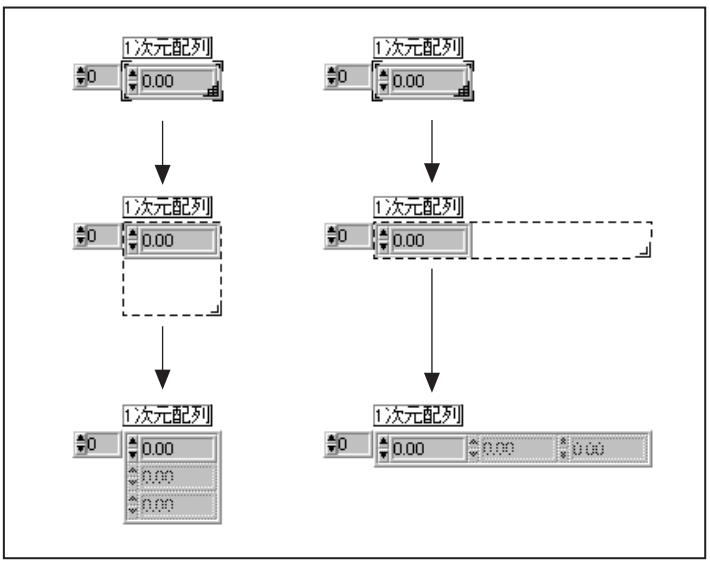
配列サイズ変更ツール

通常のサイズ変更記号

新しい配列は、単一要素形式で表示されます。配列は、単一要素の値、すなわち指標表示が示す値を表示します。配列は、要素を取り囲む 4 つの角のいずれかを使用して配列シェルのサイズを変更することにより、要素の表として表示することもできます。配列サイズ変更ツールは、配列シェルのサイズ変更ハンドルの上にあるときは通常のサイズ変更ツールとは若干異なる形をしていますが、サイズ変更の操作を開始すると通常のサイズ変更の記号に変わります。

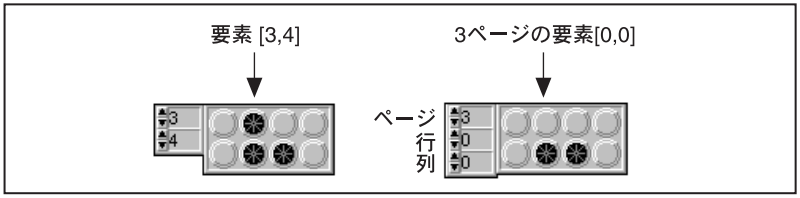


次の図は、1 次元配列のサイズを垂直方向または水平方向に変更してより多くの要素を同時に表示できるようにする方法を示したものです。

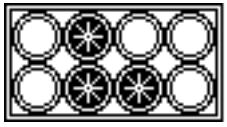


指標表示領域には、表の左または左上隅にある要素の指標が表示されます。指標を変更すると、大きな配列または多次元配列の別の部分を表示することができます。1 次元配列の場合は、指標は最も左に表示されている要素の列を示します。2 次元配列の場合は、下の 2 つの指標が左上に表示されている要素の座標を示します。次の左の図では、この要素は[3,4] となっています。

3 次元の配列を、文字（列）の行（行）で構成された複数のページからなる一冊の本とみなすと、表には 1 つのページのすべてまたは一部が表示されます。次の右の図は、配列の 3 ページ目の最初の 2 行の最初の 4 つの列を示しています。



配列の指標を表示しないようにしたいときは、外側の枠をポップアップし、ポップアップメニューで**表示→指標を表示**をオフに切り替えます。次の図は、指標を表示しない表形式の配列の例を示したものです。



配列を表形式の配列とは異なる向きで表示したいときは、フロントパネル上でクラスタの中に要素を表示し、配列中でそれらの要素を処理します。それには、**オンラインリファレンス→関数と VI リファレンス→クラスタ関数**トピックで説明した Array To Cluster 関数および Cluster To Array 関数を使用します。

配列を操作する

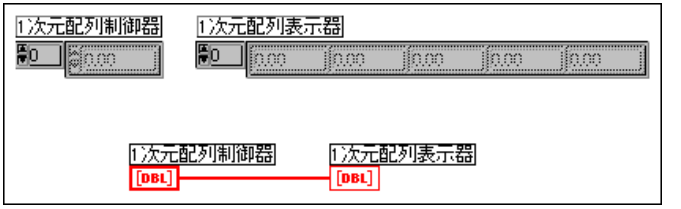
配列の要素は、通常の制御器と同様に操作することができます。指標の表示は、デジタル制御器を操作する場合と同じ方法で操作することができます。配列制御器の要素の値を設定するには、指標の表示を操作して要素をウィンドウに表示したのち、要素の値を設定します。

配列のデフォルトサイズとデフォルト値

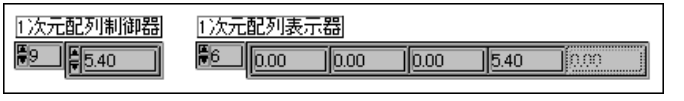
配列のサイズは、要素の数を固定することによって制限することはできません。ただし、配列制御器のデフォルト値を設定した場合は、デフォルトのサイズも設定することができます。**デフォルトサイズを必要以上に大きく設定しないようにしてください。**配列のデフォルトサイズを大きく設定すると、すべてのデフォルトデータが VI とともに保存されるため、ファイルのサイズも大きくなります。

要素を含まない新しい配列シェルは、**定義されていません**。このような配列はデータタイプも要素もないため、プログラムで使用することはできません。配列に制御器を配置すると、その配列は配置した制御器と同じタイプの**空の配列**（長さは 0）になります。この配列は要素も定義されていませんが、データタイプが定義されているため VI で使用することは可能です。配列制御器はすべての要素をグレーで表示し、要素が定義されていないことを示します。

次の図は、5 つの要素が表示された表形式の配列表示器に接続された単一要素の配列制御器を示したものです。最初の図は、両方の配列が空白である（要素の値が定義されていない）ことを示しています。

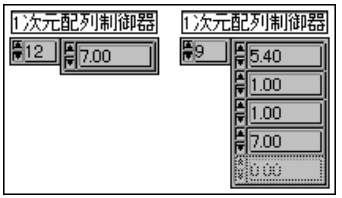


配列制御器の指標表示を 9 に設定し、指標 9 の要素の値を 5.4 に設定すると、配列中のデータ（要素）の数は 10 個（0 ～ 9）に増えます。要素 9 の値は、上で設定した値すなわち 5.4 になります。その他の要素には、デジタル制御器のデフォルト値である 0.00 が割り当てられます。ここで、シェルまたは配列制御器のポップアップメニューで**データ処理→現在の設定をデフォルト設定にする**を選択すると、配列のデフォルトデータは指定された 10 個の要素で構成されます。VI を実行した後は、表示器に同じ値が表示されます。



要素に対して**データ処理→デフォルト設定に戻す**を選択すると、要素はデフォルト値にリセットされます。

ここで、数値要素のデフォルト値を 0.0 から 1.0 に変更してみましょう。デフォルト値を変更するには、操作ツールで表示器の値を変更したのち、要素をポップアップして**現在の設定をデフォルト設定にする**を選択します。さらに、ここで要素 12 の値を 7.0 に変更すると、配列の要素の数は 13 個になります。最初の 10 個の要素の値は変わりませんが、要素 10 と要素 11 の値はデフォルト値の 1.0 になり、要素 12 の値は指定した 7.0 という値になります。この例を次の図に示します。



ここで、シェルまたは指標のポップアップメニューから**デフォルト設定に戻す**を選択すると、上の図で示したように配列の要素は再び 10 個に戻ります。

配列のサイズを [N_i x N_j x N_k ...] から [M_i x M_j x M_k ...] に増やすためには、最後の新たな要素 [M_i–1, M_j–1, M_k–1 ...] に値を割り当てます。サイズを小さくするためには、指標配列のポップアップメニューから**データ処理→配列を空にする**コマンドを実行したのち配列および値を設定するか、または次の項で述べるように配列中の不要なグループを選択した後ポップアップメニューから**データ処理→データを切り取る**を選択します。

配列の要素

実行モードでは、<Tab> キーを使用してフロントパネルの制御器間を移動したり、配列の要素間を移動することができます。最初は、<Tab> キーを押すとフロントパネルの制御器間を移動します。これを、特定の配列の要素間を移動できるようにしたいときは、まず最初に <Tab> キーでその配列に移動します。次に、<Ctrl> **(Windows)**、<command> **(Macintosh)**、<meta> **(Sun)**、または <Alt> **(HP-UX)** キーと下向きの矢印キーを利用して配列の中に移動します。それ以降は、<Tab> キーを使用して配列の要素間または配列の指標間を順番に移動することができます。再び <Tab> キーで制御器間を移動できるようにしたいときは、<Ctrl> **(Windows)**、<command> **(Macintosh)**、<meta> **(Sun)**、または <Alt> **(HP-UX)** キーと上向き矢印キーを使用します。

配列のサイズを検出する

配列制御器や配列表示器のサイズを知りたいときは、指標配列のポップアップメニューから**データ処理→最後の配列要素を表示**を選択します。

配列の移動とサイズの変更

配列を移動する場合は、必ずシェルの境界線または指標表示をクリックしてから配列をドラッグします。配列のシェルと要素は互いに固定されていないため､配列中の要素をドラッグすると要素が配列から切り離されます。シェルをドラッグするつもりが誤って要素をドラッグしてしまった場合は、要素をドラッグしてシェルに戻すか、要素をウィンドウの境界線の外側までドラッグするか､あるいは<Esc>キーを押してからマウスボタンを放すことにより、操作を無効にすることができます。サイズ変更の操作を無効にしたいときは、境界線をフロントパネルウィンドウの外側までドラッグしてからマウスボタンを放します。配列が現在選択されている領域の一部である場合は、要素をつかむと配列がドラッグされます。

配列の指標のサイズ変更は、垂直方向の場合は 4 つの角のいずれかを使用し、水平方向の場合は左側の角を使用します。

配列セルを選択する

配列からデータをコピーしたり、配列にデータを貼り付けるためには、まず最初に配列の領域を選択する必要があります。コピーや貼り付けは、下記の手順で行います。

1. データを選択するため、配列の指標をコピーしたいデータセットの最初の要素に設定します。
2. 配列のポップアップメニューから**データ処理→選択項目の開始**を選択します。
3. 配列の指標を、コピーしたいデータセットの最後の要素に設定します。
4. 配列のポップアップメニューから**データ処理→選択項目の終了**を選択します。
5. 最後に、**データ処理→データをコピーする**または**データ処理→データを貼り付ける**を選択して、選択したデータをコピーするか、または貼り付けます。

実行モードでは、配列をポップアップするとこれらのメニュー項目を直接使用することができます。選択の手順は、次の例で詳しく説明します。

配列セルの選択例

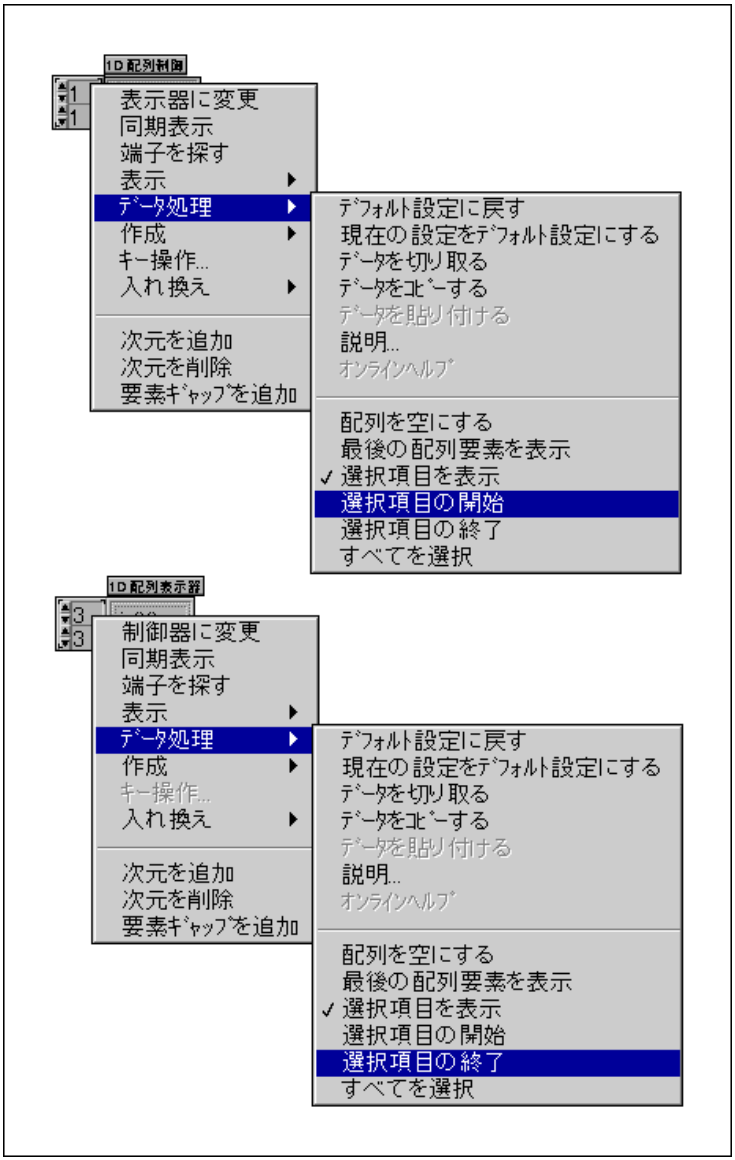
まず最初に、ポップアップメニューで**選択項目を表示**を選択します。選択したセルを目で確認できるようにするためには、**選択項目を表示**をイネーブルに設定する必要があります。

選択したセルの周りに枠が表示されます。ポップアップメニューから**データ処理→要素ギャップを追加**を選択します。セルが細いスペースによって分割されます。かならずしも要素間にスペースを挿入する必要はありませ

んが、スペースを挿入することによって配列の外観が若干変化し、見やすくなります。セルを選択すると、このスペース上に太い枠が表示され、どのセルが選択されているかが示されます。要素間にスペースを挿入しなかった場合は、選択したセルのエッジ上に枠が表示されます。

選択するセルは、配列の指標で定義します。次の例では、2 次元配列を使用しています。

配列の指標を 1,1 に設定します。ポップアップメニューから**選択項目の開始**を選択します。配列の指標を 3,3 に設定します。ポップアップメニューから**選択項目の終了**を選択します。

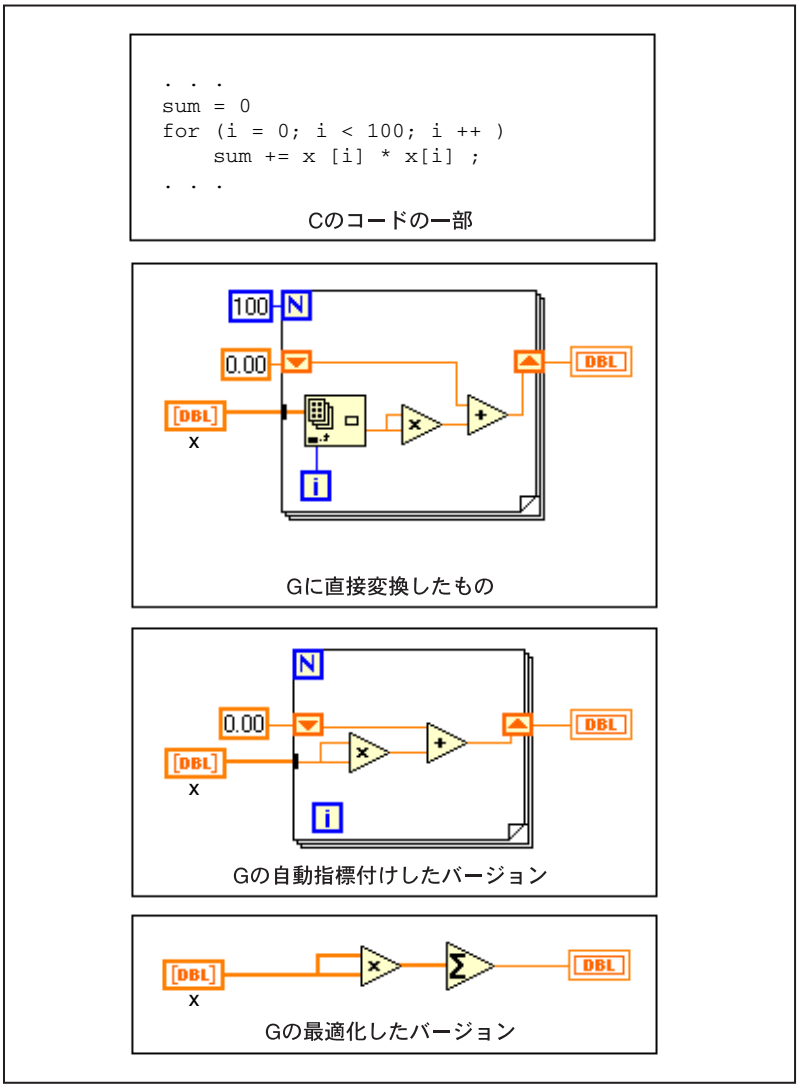


選択範囲には、小さい方の指標番号で指定したセルは含まれますが、大きい方の指標番号で指定したセルは含まれません。1,1 から 3,3 までのセルを選択したいときは、指標を 4,4 に設定してから**選択項目の終了**を選択します。選択した項目を切り取ると、選択範囲の上下左右の行と列のセットが削除されますが、クリップボードには実際に選択した領域だけがコピーされる点に注意してください。

セルを選択したら、別のセルに貼り付けるデータを切り取るか、またはコピーします。すると、枠の一部だけがハイライト表示のままになります。これが挿入ポイントになり、配列中に残されます。この枠を隠したいときは、ポップアップメニューから**選択項目を表示**の選択を解除します。この枠は、**選択項目を表示**の選択を解除しなくても、選択を空白にすれば消すことができます。それには、すべての次元の指標を 0 に設定したのち、**選択項目の開始**と**選択項目の終了**を選択します。それにより、各次元の選択範囲が 0 ～ 0 に設定され、フロントパネルに最初に配列を配置したときと同じ状態になります。

G の配列と他のシステムの配列

大部分のプログラミング言語が配列を使用しますが、G ほど多くの配列関数を組み込んでいる言語はほとんどありません。通常、他の言語では要素の抽出や差し替えといった基本的な配列操作しか用意されていないため、複雑な操作はユーザが構築する必要があります。G にはこれらの基本演算子が用意されているため、別の言語のプログラムを直接 G の VI にマッピングすることができます。ただし、一般的には G の高度な配列関数を使用して再設計した方が、より簡単で小さなダイアグラムを作成することができます。次の例は、配列の各要素の 2 乗を合計する C プログラムの一部、それを G のブロックダイアグラムに直接変換したもの、および変換したものよりも効率的な 2 つの再設計バージョンを示したものです。

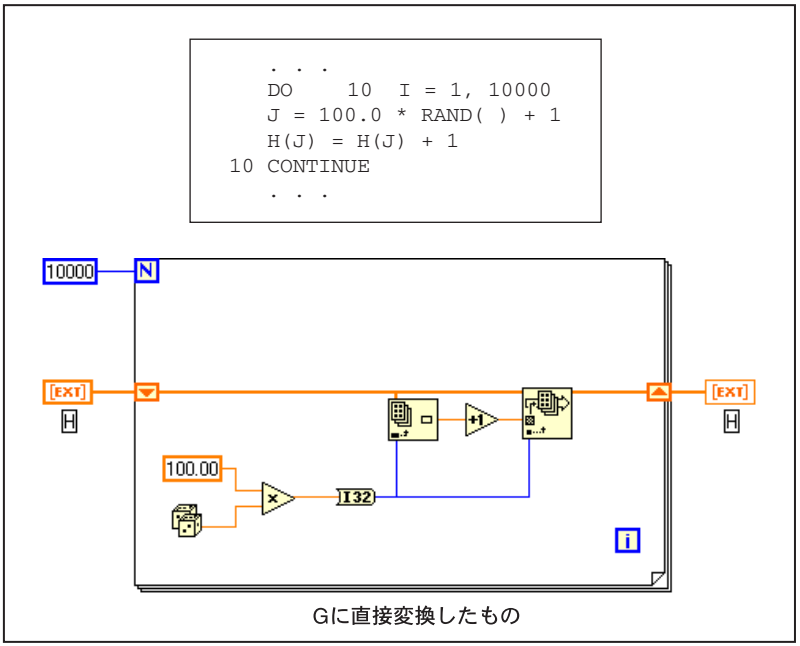


Index Array 関数の
アイコン

要素を抽出する指標付け操作は、C では配列名の後の指標を囲む [ ] で表します。G では、この操作は左記の Index Array 関数のアイコンで表します。配列は左上の端子に配続され、指標は左下の端子に配続されます。また、配列の要素の値は右の端子に配続されます。

次の例は、配列の要素を差し替える基本操作を示したものです。FORTRAN コードの一部は乱数のヒストグラムを生成（乱数ジェネレータは 0 ～ 1 の値を生成）しますが、これは G バージョンも同じです。FORTRAN の J の

計算にある1の加算は無視してください。FORTRANの場合、配列は指標が0ではなく1から始まるため、加算が必要です。



Gに直接変換したもの

Replace Array Element
関数のアイコン

要素の挿入あるいは差し替えのための指標付け操作は、FORTRANでは配列名の後の等号の左にある指標を囲む（ ）で表されます。Gでは、この操作は左記のReplace Array Element関数のアイコンで表されます。配列は左上の端子に接続され、差し替え値は左中央の端子に接続されます。また、指標は左下の端子に接続され、更新後の配列は右の端子から接続されます。

ほとんどのプログラミング言語では、配列を使用する前にまずそのことを宣言する必要があり、ときには配列を初期化することが必要になります。Gでは、フロントパネル上で配列制御器または配列表示器を作成することが、配列を宣言することに相当します。また、その値を定義することが、配列への初期値の割り当てに相当します。

ほとんどの言語では、宣言の一部として配列の最大サイズを示す必要があります。配列のサイズに関する情報は配列を構成するデータの一部とはみなされないため、プログラムの作成者はこのサイズを常に把握している必要があります。Gでは、配列のサイズ情報は自動的に記憶されるため、配列の最大サイズを宣言する必要はありません。

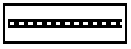
また、Gでは配列の枠外の値は保存できないようになっています。一方、従来のほとんどのプログラミング言語にはこのようなチェック機能がないため、誤ってメモリを破壊してしまうおそれがあります。

クラスタ

Gのクラスタは、番号の付いた1つまたは複数の要素の集合体で、Cやその他の言語のデータ構造とよく似ています。配列とは異なり、クラスタではGのデータタイプを自由に組み合わせて使用することができます。クラスタの要素と配列の要素にはどちらも番号が割り当てられますが、クラスタの要素には、1度に1つの要素の指標を指定するのではなく、一度にクラスタをすべての要素に**分解する**ことによってアクセスします。クラスタは、サイズが固定されている点においても配列とは異なります。配列の場合と同様、クラスタも制御器としても表示器としても使用できますが、クラスタの要素に制御器と表示器の両方を使用することはできません。

クラスタを使用すると、ダイアグラム上の複数の場所に表示される関連データ要素を1つにまとめることができるため、配線の複雑さを軽減でき、サブVIが必要とするコネクタ端子の数も減らすことができます。

ブロックダイアグラム上のほとんどのクラスタは配線パターンが共通していますが、数字のクラスタ（**点**とも呼ばれる）には特殊な配線パターンがあります。



クラスタの共通配線パターン

数字のクラスタの配線パターン

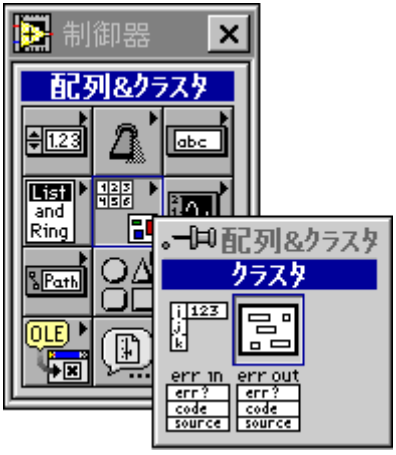
端子は、両方のタイプが同じ場合にのみ接続できます。これは、クラスタの場合、両方のクラスタの要素数が同じで、かつ（クラスタ順位によって決まる）対応する双方の要素のタイプが一致していなければならないことを意味します。表記法が異なる数字には、強制的に同じタイプが適用されます。

クラスタ順位は、クラスタの要素に割り当てられる番号を表す数字です。これについては、本章の「クラスタ要素の順位」の項で説明します。

クラスタを操作する関数についての詳細は、**オンラインリファレンス→関数とVIリファレンス→クラスタ関数**を参照してください。

クラスタを作成する

クラスタ制御器やクラスタ表示器は、ブール、文字列、チャート、グラフのスカラ、配列、あるいは他のクラスタを任意に組み合わせ、クラスタシェルの中に配置することによって作成します。**クラスタシェル**には、次の図に示すように**制御器**パレットからアクセスします。



新しいクラスタシェルには、サイズの変更が可能な枠とラベルがあります。

空白の要素領域をポップアップすると、**制御器**パレットが表示されます。クラスタには**制御器**パレットの任意の要素を配置することができます。既存のオブジェクトをクラスタシェルにドラッグすることもできますが、ポップアップメニューから選択してオブジェクトを作成し、それをドラッグして直接クラスタシェルに配置することもできます。クラスタは、最初にクラスタに配置したオブジェクトのデータの方向(制御器または表示器)を引き継ぐため、そのあとに追加したオブジェクトにもそのデータ方向が適用されます。クラスタのいずれかの要素のポップアップメニューから**制御器に変更**または**表示器に変更**を選択すると、クラスタとそのすべての要素が表示器から制御器に、または制御器から表示器に変更されます。

クラスタ要素の操作と設定

制御器と表示器の構成方法は、デフォルト値の設定とメニュー項目**制御器に変更**および**表示器に変更**を除いては、クラスタの外にある制御器や表示器の構成方法と同じです。

クラスタ要素

実行モードでは、<Tab>キーを使用してフロントパネルの制御器間を移動したり、クラスタの要素間を移動することができます。最初は、<Tab>キーを押すとフロントパネルの制御器間を移動します。これを、特定のクラスタの要素間を移動できるようにしたいときは、まず最初に<Tab>キーでそのクラスタに移動します。次に、<Ctrl>**（Windows）**、<command>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**キーと下向きの矢印キーを使用してクラスタの要素間を移動します。再び<Tab>キーで制御器間を移動できるようにしたいときは、<Ctrl>**（Windows）**、<command>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**キーと上向き矢印キーを使用します。

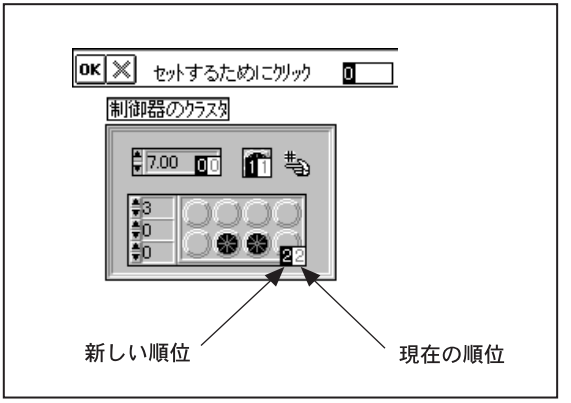
クラスタのデフォルト値

クラスタ全体のデフォルト値を個々の要素の現在の値に設定するには、クラスタの枠をポップアップし、クラスタのポップアップメニューから**データ処理→現在の設定をデフォルト設定**コマンドを選択します。すべての要素をそれぞれのデフォルトの構成に戻すには、クラスタのポップアップメニューから**データ処理→デフォルト設定に戻す**コマンドを選択します。

クラスタの1つの要素のデフォルト値を変更するには、その要素をポップアップし、**現在の設定をデフォルト設定**を選択します。

クラスタ要素の順位

クラスタの要素には、シェル内部の配置に関係のない論理上の順位が割り当てられます。クラスタに最初に挿入したオブジェクトは要素0、2番目に挿入したオブジェクトは要素1という順位になります。要素を削除すると、順位は自動的に調整されます。現在の番号は、クラスタのポップアップメニューから**クラスタ順位 ...** を選択することにより変更することができます。要素の外観は、次の図に示すように変化します。



要素上の白いボックスは、クラスタ内でのその要素の現在の順番を表示します。黒いボックスは、新しい順番を表示します。クラスタ要素の番号設定カーソルで要素をクリックすると、ツールバーに表示された番号がクラスタ内でのその要素の順位として設定されます。この順位を変更するには、そのフィールドに新しい番号を入力し、該当する制御器をクリックします。順位が正しく設定できたら、入力ボタンをクリックし、設定を確定してクラスタの要素順位の編集モードを終了します。前の設定に戻したいときは、×ボタンをクリックします。


入力ボタン

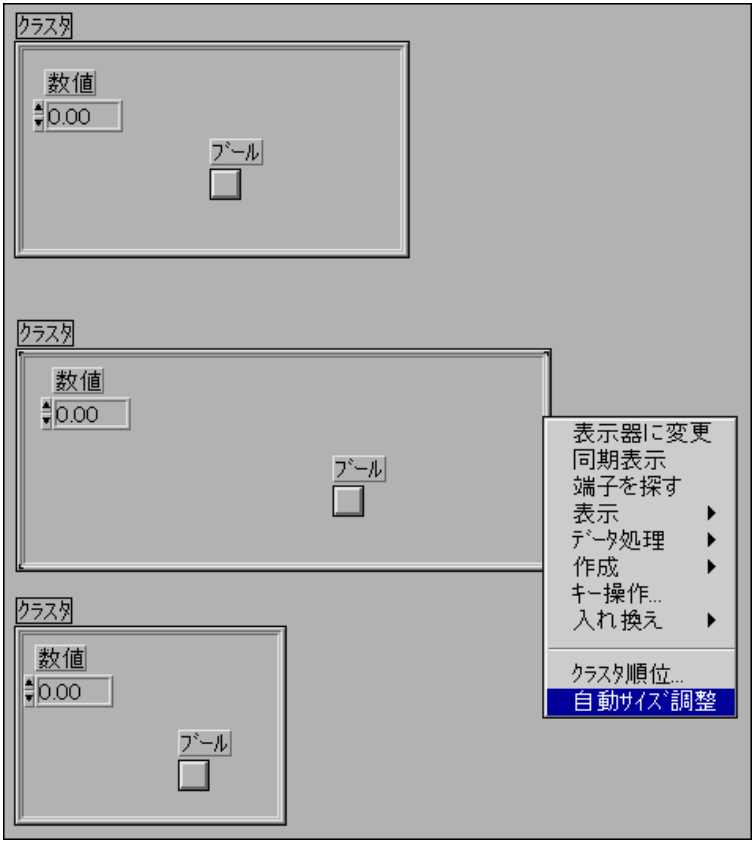

×ボタン

クラスタ要素の順位は、ブロックダイアグラム内の Bundle 関数および Unbundle 関数上に端子として現れる要素の順位を決定します。これらの関数については、詳しくは**オンラインリファレンス→関数とVIリファレンス→クラスタ関数**トピックを参照してください。

クラスタの移動とサイズの変更

クラスタ要素は、クラスタの固定コンポーネントではありません。つまり、シェルの内部においても要素を個別に移動したり、サイズを変更したりできるということです。要素を誤ってシェルの外側にドラッグするのを防ぐため、クラスタを移動したいときはシェルをクリックし、クラスタのサイズを変更したいときはシェルの枠から変更します。クラスタ全体をドラッグするつもりが、誤って要素をドラッグしてしまった場合は、要素をドラッグしてシェルの内側に戻すか、フロントパネルウィンドウの枠の外側までドラッグするか、または <Esc> キーを押したあとでマウスボタンを放すと、操作を無効にすることができます。また、サイズの変更の操作を無効にしたいときは、枠をフロントパネルウィンドウの外側までドラッグするか、<Esc> キーを押してからマウスボタンを放します。

次の図に示すように、ポップアップメニューから**自動サイズ調整**を選択すると、クラスタのサイズを内容が収まる適度な大きさに調整することができます。



クラスタの組み立て

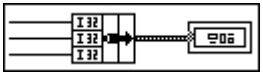
Gには、クラスタの組み立てや作成のための関数が3種類あります。Bundle と Bundle By Name は、個々の要素からクラスタを組み立てたり、個々の要素を同じタイプの要素と差し替えるための関数です。Array To Cluster関数は、配列の要素をクラスタの要素に変換します。

Bundle 関数

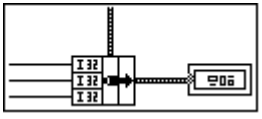

Bundle 関数

Bundle関数は、ブロックダイアグラム上では左側に2つの要素入力端子が付いた状態で表示されます。この関数には、**関数**パレットの**配列＆クラスタ**パレットからアクセスします。アイコンの垂直方向のサイズを変更すると、必要な数だけ端子を作成することができます。一番上の端子に接続する要素は、出力クラスタの要素0になり、2番目の端子に接続する要素は出力クラスタの要素1になります。

各入力端子に配線すると、次の図に示すようにそれまで空白だった端子の中に、接続した要素のデータタイプを表す記号が表示されます。作成した入力はすべて配線する必要があります。



Bundle関数には、次の図に示すように、左側の要素用入力端子のほかに中央のクラスタ用入力端子もあります。この端子を使用すると、既存のクラスタのワイヤの 1 つまたは複数の要素を他の要素に影響を与えずに入れ換えることができます。



この関数を使用してクラスタの要素を入れ換える場合の例については、本章の「クラスタ要素を差し替える」の項を参照してください。

Bundle By Name 関数

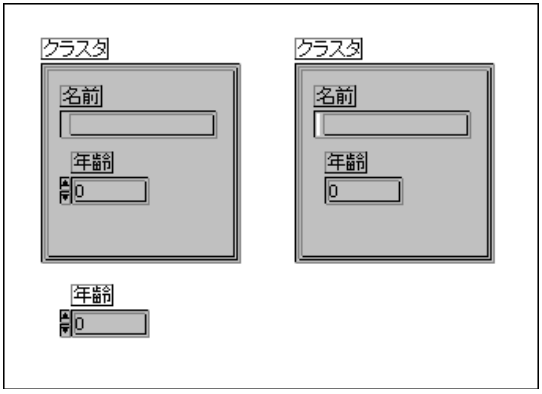

Bundle By Name 関数

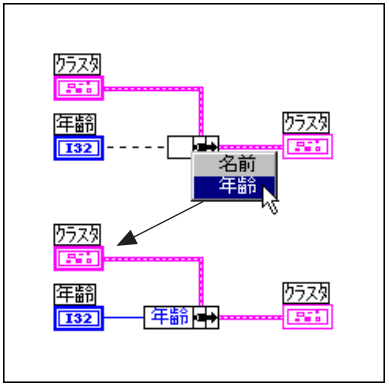
Bundle By Name 関数も**関数→クラスタ**パレットにあり、Bundle 関数とよく似ています。ただし、この関数は要素の順位でフィールドを検索するのではなく、名前でフィールドを検索します。Bundle関数とは異なり、アクセスしたい要素だけを表示することができます。アクセスしたい要素ごとに、関数への入力を追加する必要があります。

名前はクラスタ内での要素の順位を示すものではないため、クラスタも中央の端子に配線する必要があります。Bundle By Name 関数は、名前を使用して要素を入れ換える場合にのみ使用するもので、新しいクラスタを作成するために使用することはできません。ただし、データタイプのクラスタを中央の端子に配線し、さらに新しい値をすべて名前として配線することにより、同じ動作を発生させることができる場合もあります。

たとえば、Name という文字列と Age という番号を含むクラスタがあるものとします。Bundle By Name 関数を配置したら、まず最初に入力クラスタを Bundle By Name 関数の中央の端子に接続する必要があります。

中央の端子を配線したら、Bundle By Name 関数の左のいずれかの端子をポップアップし、変更したい要素を選択します。すると、ソースクラスタ内の要素名のリストが表示されます。このリストで名前を選択したら、新しい値をその端子に接続して、クラスタのその要素に新しい値を割り当てることが可能になります。その例を次の図に示します。

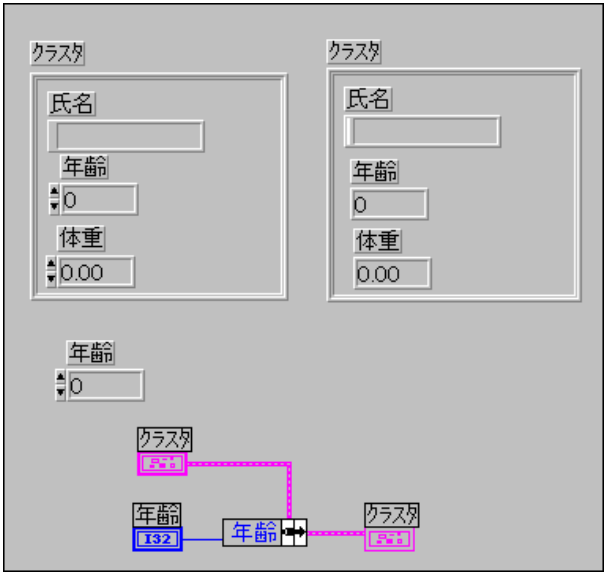




Bundle By Name 関数は、必要な数だけの要素が表示されるようにサイズを変更することができます。これは、最終的なクラスタの要素と同じ数の要素を必要とする Bundle 関数とは異なります。

クラスタにクラスタが含まれている場合は、Bundle By Name 関数の個々の要素をポップアップすると、横に**項目選択**というサブメニューが表示されます。このサブメニューを使用すると、サブクラスタの個々の要素に名前でアクセスしたり、すべての要素を選択したりすることができます。

Bundle By Name 関数は、開発の途中で修正する可能性のあるデータ構造を使用して作業をする場合に役立ちます。新しい要素の追加やクラスタ要素の順位の変更をともなう修正を行う場合でも、名前は引き続き有効であるため、Bundle By Name 関数を変更する必要はありません。たとえば、上に示した例を変更して元のクラスタと接続先のクラスタに新しい要素の体重を追加しても、VI の実行内容に変わりはありません。その例を次の図に示します。



Bundle By Name は、Type Def を使用する大きなアプリケーションで使用すると特に便利です。変更される可能性のあるクラスタを Type Def として定義しておくと、Type Def のファイルを更新することにより、その制御器を使用する VI を新しいデータタイプが映されるように変更することができます。クラスタの要素にアクセスする際に VI が Bundle By Name と Unbundle By Name しか使用しなければ、そのクラスタを使用する VI が破壊されるのを防ぐことができます。

Type Def についての詳細は、「第 24 章　カスタム制御器と Type Def」の項を参照してください。また、本章の「Unbundle By Name 関数」の項も参照してください。

Array To Cluster 関数

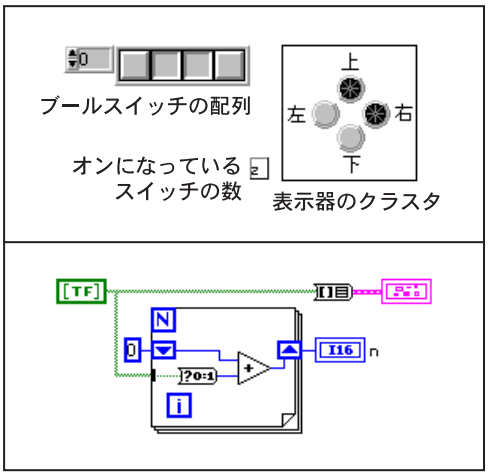

Array To Cluster 関数

Array To Cluster 関数は、1 次元配列の要素をクラスタの要素に変換します。この関数は、**関数**パレットの**クラスタ**パレットまたは**配列**パレットから選択します。フロントパネルのクラスタ表示器内で同じタイプの要素を表示し、かつブロックダイアグラム上で指標値を利用して要素を操作したいときに使用すると便利です。

クラスタの要素の数を構成するためには、関数のポップアップメニューから**クラスタサイズ...** の項目を選択します。そのときに表示されるダイアログボックスを次の図に示します。

この関数は、配列の要素 0 をクラスタの要素 0 に、配列の要素 1 をクラスタの要素 1 にという要領で順次要素を割り当てます。配列の要素の数がクラスタの要素の数よりも多い場合、関数は残りの要素を無視します。配列の要素の数がクラスタの要素の数よりも少ないときは、関数はクラスタ側の余っている要素にその要素のデータタイプのデフォルト値を割り当てます。

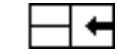
次の図は、データ配列をフロントパネルのクラスタ表示器で表示する方法を示したものです。この方法を使用すると、アプリケーションに合わせてフロントパネル上の配列の要素をアレンジすることができます。代わりに表形式の配列表示器を使用することも可能ですが、その場合は固定表示方式（横方向の表示で一番下の要素が左に表示され、かつ指標表示付き）以外は使用できなくなります。



クラスタの分解

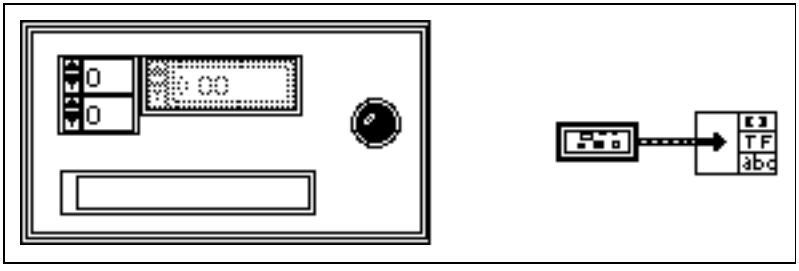
Gには、クラスタを分解するための関数が3つあります。Unbundle関数とUnbundle By Name関数は、クラスタを個々の要素に分解します。Cluster To Array関数は、同じタイプのクラスタをそのタイプの要素の配列に変換します。

Unbundle関数


Unbundle関数

Unbundle関数は、**関数**パレットの**クラスタ**パレットから選択します。この関数は、右側に2つの要素出力端子があります。端子の数は、Bundle関数の場合と同じようにサイズ変更ツールを使用して調節することができます。一番上の端子にはクラスタの要素0が渡され、2番目の端子には要素1が渡されます。

クラスタのワイヤは、正しい数の出力端子を作成するまで壊れたままになります。端子の数が正しい数になると、ワイヤは塗りつぶされます。次の図に示すように、端子の記号には要素のデータタイプが表示されます。

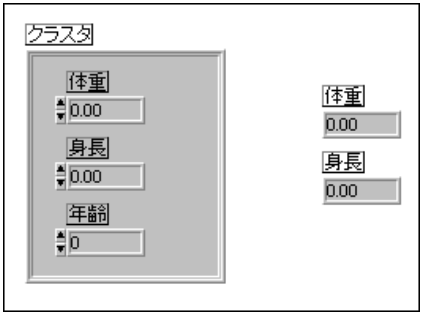


Unbundle By Name 関数

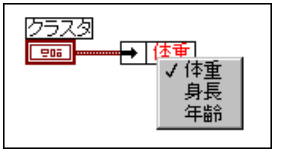

Unbundle By Name 関数

Unbundle By Name 関数も、**関数**パレットの**クラスタ**パレットにあります。この関数は、Unbundle 関数とよく似ています。ただし、要素の順位ではなく名前でフィールドを検索します。Unbundle 関数とは異なり、Unbundle By Name 関数では読み取りたい要素だけを表示することができます。また、かならずしも各クラスタ要素につき出力が 1 つ必要なわけではありません。

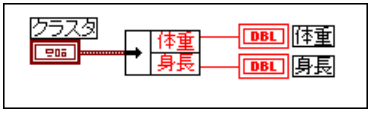
例として、体重、身長、年齢という 3 つの要素からなるクラスタを持つパネルを次に示します。



Unbundle By Name 関数に接続したら、次の図に示すように、関数の出力端子をポップアップして**項目選択**を選択し、クラスタの要素の名前から読み取りたい要素を選択することができます。



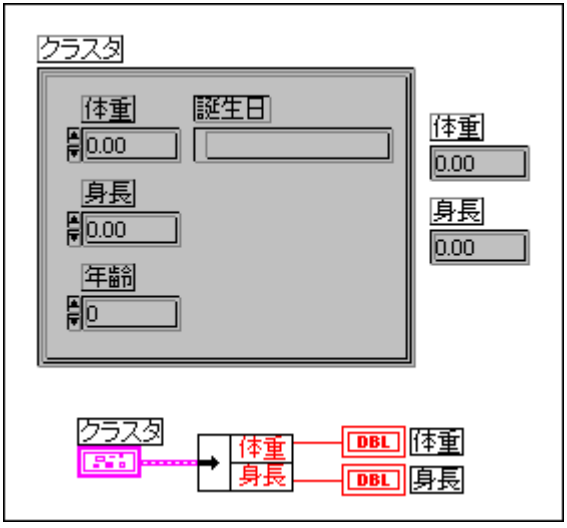
次の図に示すように、他の要素を読み取りたいときは関数のサイズを変更します。ただし、かならずしもすべての要素を読み取る必要はないこと、および要素は任意の順序で読み取ることができることを覚えておいてください。



Unbundle 関数に比べて Unbundle By Name が有利な点は、クラスタのデータ構造とそれほど緊密に結びついていない点です。Bundle 関数では、かならずその要素の数と同じ数の端子が必要になります。Unbundle 関数に接続したクラスタに要素を追加したり要素を削除したりすると、ワイヤが破壊

されてしまいます。一方、Unbundle By Name 関数では、ダイアグラム上で要素を参照していない限り、クラスタに要素を追加したりクラスタから要素を削除したりしても VI が破壊されることはありません。

たとえば、前に示したクラスタでは、誕生日の文字列を追加してもダイアグラムが有効であることに変わりはありません。その例を次の図に示します。



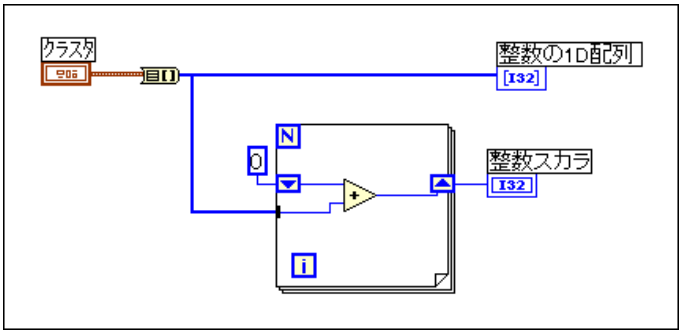
Bundle By Name 関数と同様 Unbundle By Name 関数は、特に Type Def が関与した大きなアプリケーションにおいて使用すると便利です。詳しくは、本章の「Bundle By Name 関数」の項、および「第 24 章　カスタム制御器と Type Def」の項を参照してください。

Cluster To Array 関数


Cluster To Array 関数

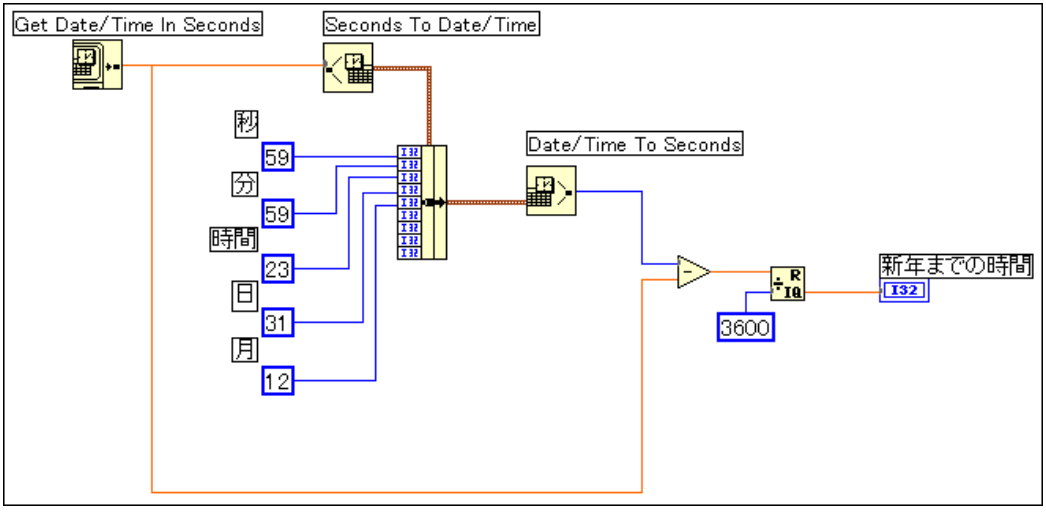
Cluster To Array 関数は、**関数**パレットの**変換**パレットにあります。この関数は、クラスタの要素を 1 次元配列の要素に変換します。ただし、クラスタ要素はすべて同じタイプでなければなりません。配列には、クラスタの要素と同じ数の要素が含まれます。

Cluster To Array 関数は、次の図に示すように、いくつかのフロントパネル制御器をクラスタ内である一定の順序にグループ化し、さらにそれらの制御器をブロックダイアグラム上で配列として処理したいときに使用すると便利です。



クラスタ要素を差し替える

ときには、クラスタの 1 つまたは 2 つの要素の値を他の要素に影響を与えずに差し替えたり変更したりすることが必要になる場合があります。そのような場合は、クラスタを分解し、変更したくない要素も他の要素の差し替え値とともに直接Bundle関数に接続します。次の例は、それよりもさらに便利な方法を示したものです。この例では、日付と時間のクラスタを使用して新年までの時間数を計算します。Bundle関数のクラスタの入力端子（中央の端子）はすでに配給されているため、あとは差し替えが必要な要素の入力端子だけを配給するだけで済みます。



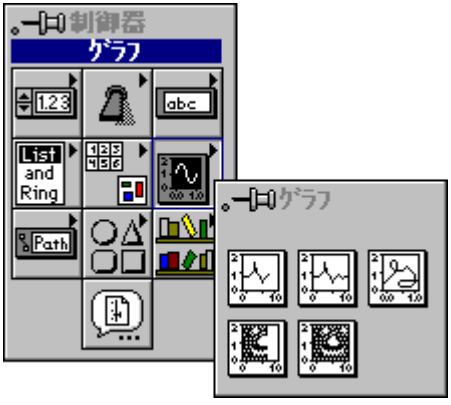
15

グラフとチャートの制御器および表示器

この章では、**制御器→グラフ**パレットにあるグラフとチャートの表示器の作成方法および使用方法について説明します。

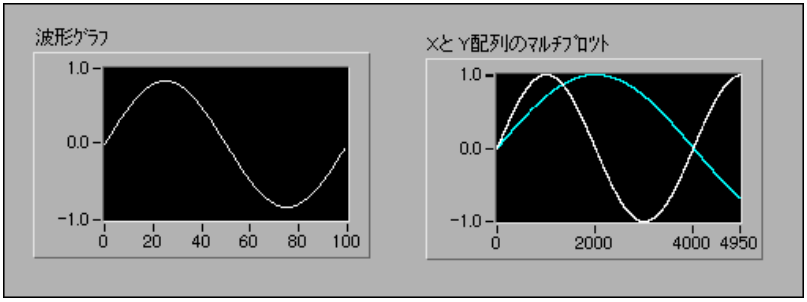
グラフ表示器は、1つまたは複数の**プロット**の2次元（2D）表示器です。グラフはデータを受け取り、データをブロックとしてプロットします。チャートもプロットを表示しますが、ポイントごとまたは配列ごとにデータを受け取り、表示を更新します。その際、一定数の過去のポイントのデータが表示用バッファに記憶されます。この章の説明とともに、グラフのチャートのサンプルも参照してください。サンプルは、`examples¥general¥graphs`に入っています。

Gには、3種類のグラフと2種類のチャートが用意されています。次の図はこれらのグラフとチャートを示したものです。上段は左から波形チャート、波形グラフ、*xy*グラフ、下段は同じく強度チャート、強度グラフの順に並んでいます。



波形グラフと XY グラフを作成する

波形グラフは、等間隔でサンプリングした測定データを表示するためのものです。**XY グラフ**は、一連の点のデータを表示することができます。これらの点は、かならずしも等間隔である必要はありません。波形グラフと *xy* グラフは、**制御器→グラフ**パレットから呼び出せます。これらのグラフの例を次の図に示します。



波形グラフは、*x* 軸上に等間隔で配された点による単一値の関数（たとえば時間の経過にともなって値が変化する波形など）をプロットします。*xy* グラフは、汎用のデカルトグラフ作成オブジェクトで、円形や波形など複数の値を持つ関数のグラフを可変の時間ベースで作成するのに使用できます。どちらのグラフも、任意の数のプロットを表示できます。

これらのグラフはいずれも、複数のデータタイプを受け付けます。そのため、表示の前に必要なデータの処理を最小限に抑えることができます。次の項で、これらのデータタイプについて説明します。

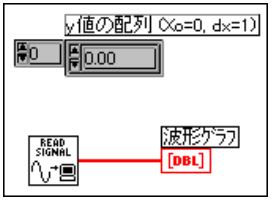
また、グラフのデータタイプに関する項に続く項では、グラフの高度なオプションについて説明します。これらのオプションには、グラフの外観のカスタマイズ、カーソルの表示と操作、グラフの凡例の表示などがあります。

シングルプロットのグラフを作成する

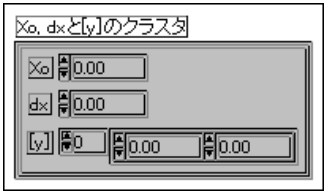
波形グラフのデータタイプ

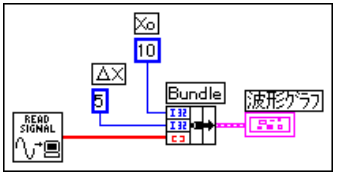
波形グラフでは、シングルプロットのグラフに 2 つのデータタイプを使用できます。以下で、これらのデータタイプについて説明します。

波形グラフで使用できるデータタイプの 1 つは、値の単一配列です。グラフはデータをいくつかの点として捉え、$x = 0$ から始めて 1 つずつ繰り上げていきます。次の図は、このタイプのデータの作成方法を示したものです。



もう 1 つのデータタイプは、x の初期値、Δx の値、および y データの配列で構成されるクラスタです。次の図は、このタイプのデータの作成方法を示したものです。

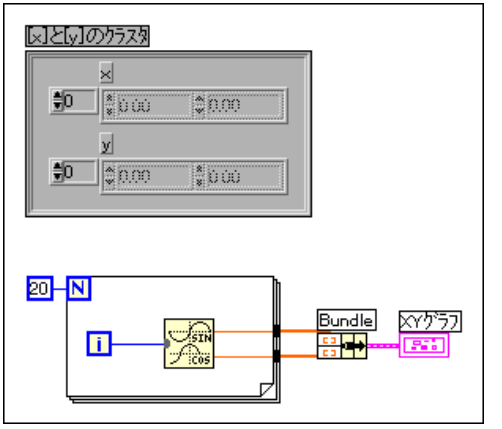




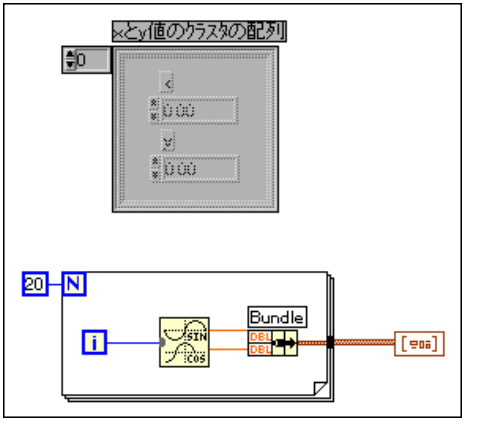
XY グラフのデータタイプ

xy グラフでは、シングルプロットのグラフには 2 つのデータタイプを使用できます。以下で、これらのデータタイプについて説明します。

xy グラフで使用できるデータタイプの 1 つは、*x* の配列と *y* データの配列で構成されるクラスタです。次の図は、このタイプのデータの作成方法を示したものです。



もう 1 つのデータタイプは点の配列です。この場合の点とは、*x* の値と *y* の値からなるクラスタです。次の図は、このタイプのデータの作成方法を示したものです。



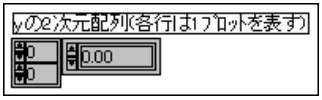
マルチプロットのグラフを作成する

1 つの波形または 1 つの *xy* グラフに複数のプロットを表示することができます。ほとんどの場合は、前の項で説明したデータタイプの配列を使用して 1 つのグラフに表示する複数のプロットを記述します。G では配列の配列は使用できないため、シングルプロットのデータタイプを作成することによって配列の配列が生成される場合には、2 次元配列か、または配列を含むクラスタの配列を使用します。

波形グラフのデータタイプ

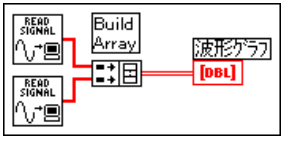
マルチプロットの波形グラフでは、次に示す 5 種類のデータタイプを使用できます。

マルチプロットの波形グラフで使用できるデータタイプの 1 つは、値の 2 次元配列です。この場合、それぞれの行のデータが 1 つのプロットに相当します。グラフはこのデータを点として捉え、*x* = 0 から始めて 1 つずつ繰り上げてゆきます。

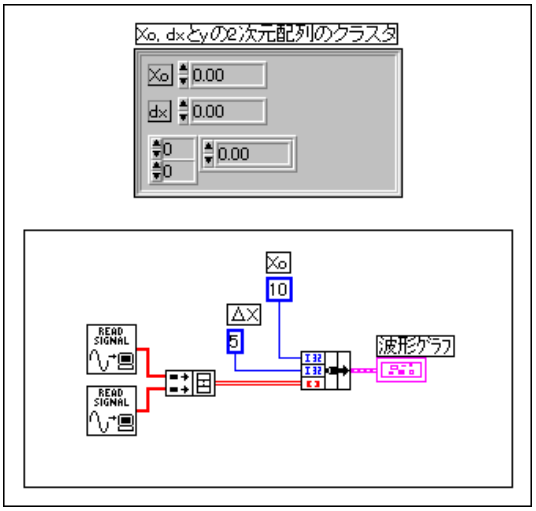


グラフのポップアップメニューから**配列を転置**を選択すると、各列のデータがプロットとして処理されます。データ集録ボードから複数のチャネルをサンプリングする際には、この方法を使用するとデータを各チャネルが別々の列に書き込まれた 2 次元配列として受け取ることができるため、とくに便利です。

次の図は、グラフを使用して 2 つの信号を表示する場合の例を示したものです。ここでは、2 次元配列のそれぞれの行が 1 つの信号に相当します。



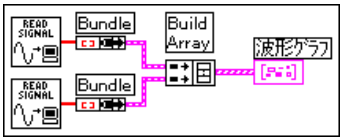
第 2 のデータタイプは、*x* の値、Δ*x* の値、および *y* データの 2 次元配列で構成されるクラスタです。*y* のデータは、先のデータタイプで説明した通りに解釈されます。このデータタイプは、すべて同じ一定の速度でサンプリングした複数の信号を表示するのに便利です。次のダイアグラムは、このタイプのデータの作成方法を示したものです。



第 3 のデータタイプは、*y* データの配列からなるクラスタの配列です。内側の配列はプロットの各点を記述し、外側の配列の各要素はそれぞれのプロットに相当します。

各プロットの要素の数が異なる場合は、2 次元配列の代わりにこのデータ構造を使用します。たとえば、複数のチャネルでデータをサンプリングする際に、サンプリングの時間長がチャネルによって異なる場合にはこのデータ構造を使用します。その理由は、2 次元配列の各行の要素の数は同じでなくてはなりませんが、配列の要素であるクラスタは、要素数の異なる配列を含むことができるためです。

次の図は、2 つの配列から適切なデータ構造を作成する方法を示したものです。

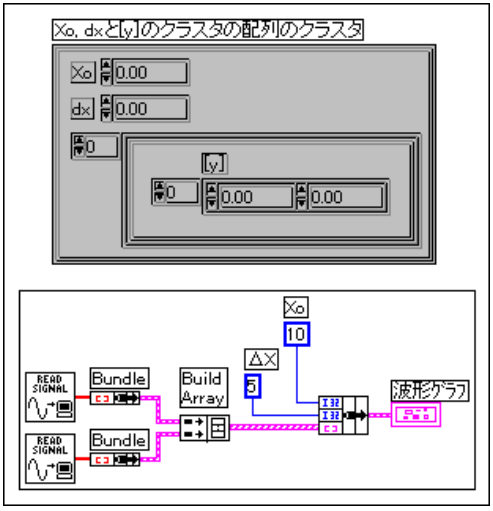


この方法のサンプルについては、examples¥general¥graphs¥gengraph.llb ディレクトリの波形グラフ VI を参照してください。

配列をクラスタからなる配列の要素に変換するもう 1 つの方法として、Build Cluster Array 関数を使用する方法があります。

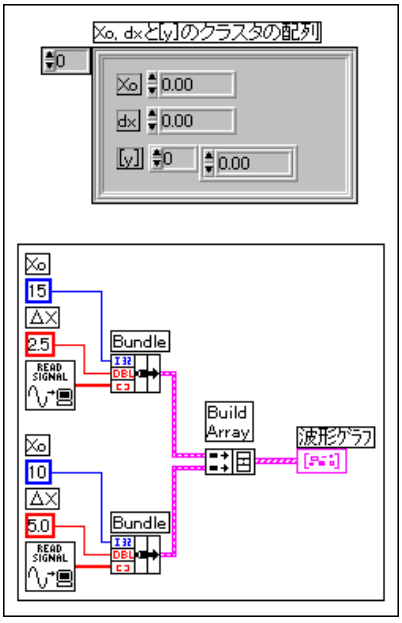
第 4 のデータタイプは、*x* の初期値、Δ*x* の値、および *y* データの配列を含むクラスタの配列で構成されるクラスタです。

各プロットの要素の数が異なる場合は、2 次元配列の代わりにこのデータ構造を使用します。たとえば、複数のチャネルでデータをサンプリングする際に、サンプリングの時間長がチャネルによって異なる場合にはこのデータ構造を使用します。その理由は、2 次元配列の各行のエレメントの数は同じでなくてはなりませんが、配列の要素であるクラスタは、要素数の異なる配列を含むことができるためです。次のダイアグラムは、このタイプのデータの作成方法を示したものです。



配列はBundle関数によってクラスタとしてまとめられ、まとめられたクラスタからBuild Array関数によって配列が構成されている点に注意してください。代わりに、指定した入力のクラスタの配列を作成する Build Cluster Array 関数を使用することもできます。

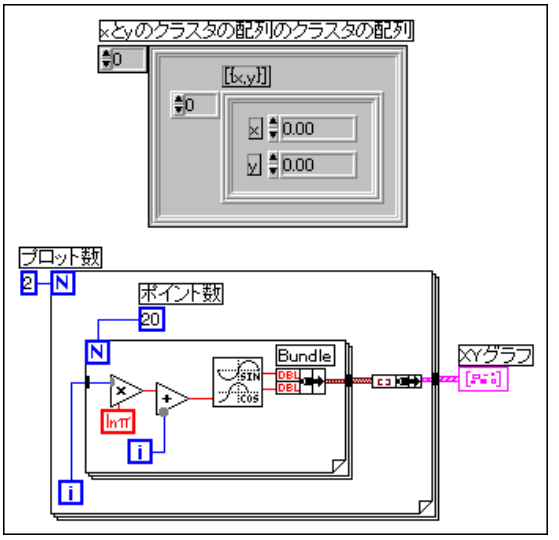
第5のデータタイプは、*x* の初期値、Δ*x* の値、および*y* データの配列からなるクラスタの配列です。これは、マルチプロットの波形グラフで使用されるデータタイプの中で最も一般的なデータタイプです。その理由は、各プロットごとに *x* 軸の開始点と増分幅を個別に設定できることにあります。次のダイアグラムは、このタイプのデータの作成方法を示したものです。



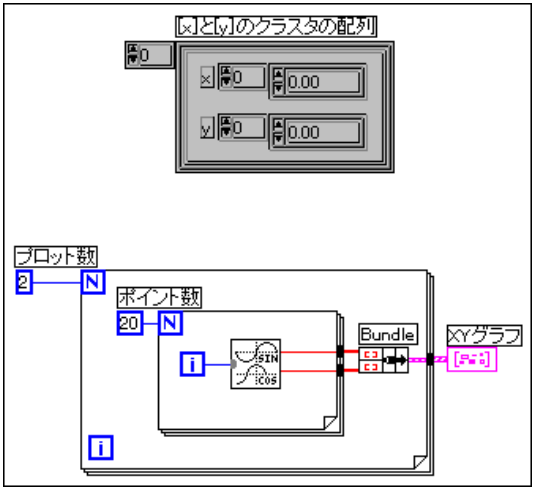
XY グラフのデータタイプ

xy グラフは、次に示すようにマルチプロットの 2 つのデータタイプを受け付けます。これらのデータタイプはいずれも、上で説明したシングルプロットのデータタイプのクラスタの配列です。

最初のデータタイプは、プロットを含むクラスタの配列です。ここでは、点の配列が 1 つのプロットに相当します。1 つの点は、*x* と *y* の値からなるクラスタとして定義されます。次のダイアグラムは、このタイプのデータの作成方法を示したものです。

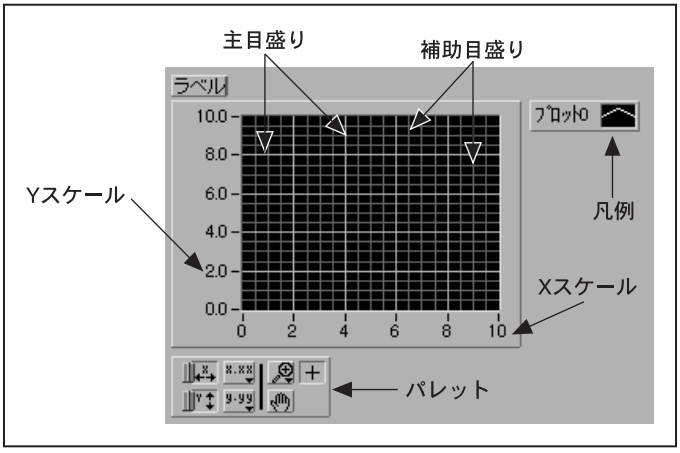


もう 1 つのデータタイプは、プロットの配列です。ここでは、x 配列と y 配列のクラスタがプロットに相当します。次のダイアグラムは、このタイプのデータの作成方法を示したものです。

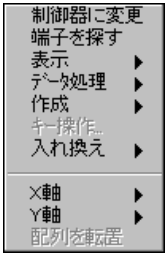


グラフにカスタムオプションを設定する

どちらのグラフにも任意に選択できるパーツがあり、これらのパーツを表示するかしないかはグラフのポップアップメニューの**表示**サブメニューを使用して切り替えることができます。これらの項目には、特定のプロットの色やスタイルを定義するための凡例、VIの実行中にスケールオプションや形式オプションを変更するためのパレット、複数のカーソルを表示するため**カーソル**パレットがあります。次の図は、これらの任意に選択できるコンポーネントのうち**カーソル**パレット以外のすべてのパーツを使用したグラフを示したものです。カーソルパレットについては、「グラフカーソル」の項で示します。



グラフには、データの表示方法をカスタマイズするための多数のオプションが用意されています。グラフのポップアップメニューを次の図に示します。**配列の転置**は、波形グラフでのみ使用できます。



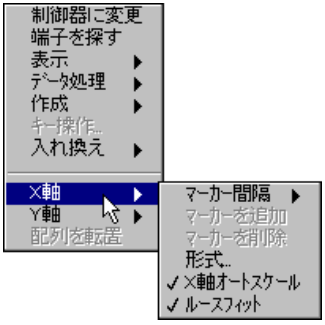
スケールオプション

グラフは、横軸のスケールや縦軸のスケールを自動的に調節し、配線されているデータを反映することができます。この自動スケール機能のオン / オフは、グラフのポップアップメニューの**データ処理**または**X 軸 /Y 軸**のサブメニューの項目**X 軸オートスケール**または**Y 軸オートスケール**を使用して切り替えます。これらの自動スケール機能は、本章の後の方で説明するようにグラフのパレットを使用して制御することもできます。グラフの自動スケール機能は、デフォルトではオンに設定されています。ただし、自動スケール機能を使用するとグラフの実効速度が遅くなる場合があります。

横軸または縦軸のスケールは、他の制御器や表示器の場合と同じように操作ツールを使用して直接変更することができます。

グラフのポップアップメニューの**データ処理**サブメニューには、オフスクリーンバッファを使用して点滅を最小限に抑える**スムーズアップデート**という項目があります。使用しているコンピュータやビデオシステムによっては、この機能を使用するとグラフの実効速度が遅くなる場合があります。

x 軸のスケールと *y* 軸のスケールには、いずれも次の図に示すような項目のサブメニューがあります。



マーカの間隔

デフォルト設定では、*x* 軸と *y* 軸のスケールのマーカ値は等間隔に配置されます。ただし、必要な場合には *x* 軸あるいは *y* 軸のスケール上でマーカの位置を任意に指定することができます。この機能は、グラフ上のいくつかの特定の点（設定点や閾値など）にマーカを設定したい場合に使用すると便利です。

任意の間隔でマーカを設定したいときは、スケールのポップアップメニューで**X 軸**（または**Y 軸**）**→マーカ間隔→任意**を選択します。この選択のあと、操作ツールをチックマーカの上に移動すると、次の図に示すようにカーソルが 2 つの矢印の付いたカーソルに変わります。この状態になっ

たら、<Ctrl>（**Windows**）、<option>（**Macintosh**）、<meta>（**Sun**）、<Alt>（**HP-UX**）キーを押しながら既存のチックマークをドラッグして新しいマーカを作成したり、あるいは既存のチックマークをドラッグしてスケール上の任意の場所に移動することができます。

マーカを追加または削除する場合は、チックマークまたはスケールをポップアップし、**マーカを追加**または**マーカを削除**を選択します。マーカを作成したあとマーカの位置を変更するためには、マーカに数字を入力します。

形式

形式...をクリックすると、次の図に示すような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の項目を選択し、下記のグラフの属性を設定します。

スケールスタイル — この項目は、スケールのメジャーチックマークおよびマイナーチックマークを選択するために使用します。メジャーチックマークは、スケールのラベルに相当する点で、マイナーチックマークはラベルとラベルの間の内側の点です。特定の軸のマーカが表示されるようにしたい場合も、このパレットを使用します。

マッピングモード — スケールメニューにあるこの項目は、線形モードまたは対数モードのどちらを使用してデータをマッピングするかを指定するために使用します。

グリッドオプション — これらの項目をクリックすると、グリッドタイプのパレットが表示されます。このパレットで、グリッドラインを表示しない、メジャーチックマークの位置（スケールのラベルに相当する点）にのみグリッドラインを表示する、メジャーチックマークとマイナーチックマーク（マイナーチックマークはラベル間にある内側の点）の両方にグリッドラインを表示する、のいずれかを選択します。グリッドのポップアップの横にある制御器は、グリッドラインの色を選択するための色のメニューです。

スケール移動子 — スケール移動子は、初期値、波形チャートや波形グラフの点と点の間の間隔、あるいは強度チャートや強度グラフのスケールの点と点の間の間隔を指定するために使用します。これらのスケール移動子は、表示するデータのスケールを変更する際にも使用できます。たとえば、データが –2,048 ～ 2,047 の範囲でサンプリングしたバイナリデータである場合、追加オフセットを 2.5 に、スケール移動子を 5/4095 = 0.001221 に設定すると、データを 0 ～ 5 の適正な電圧値に変換することができます。

形式と精度 — *x* 軸および *y* 軸のスケールの形式（数値、または時間と日付）は、ダイアログボックスの上にあるメニューリングで選択します。

数値形式を選択した場合は、浮動小数点、科学、工学、秒単位の相対時間のどの表記法にするか、10 進数、16 進数、8 進数、2 進数のどの表記法にするか、精度（小数点下のケタ数）を何ケタにするか（0 から 20 ケタまで）を選択することができます。選択した精度は値を表示する場合にのみ適用され、内部の計算の精度が表記法によって決まることに変わりはありません。それぞれの項目の組み合わせを選択すると、ダイアログボックスに例が表示されます。

時間と日付形式を選択した場合は、次の図に示すようにダイアログボックスの内容が変わります。

絶対時間と絶対日付のどちらか一方、または両方の形式を指定することができます。スケールを編集して時間のみまたは日付のみを入力した場合は、指定しなかった方のデータ成分は推測値が使用されます。編集時に時間を入力しなかった場合は、12:00 という値が使用されます。日付を入力しなかった場合は、前の日付の値が使用されます。日付を入力してもスケールが日付形式になっていない場合は、月、日、年の順番は**環境設定**ダイアログボックスの設定に基づいて決定されます。年を 2 ケタの数字で入力した場合、その数字が 38 未満のときは 21 世紀の年とみなされ、38 以上のときは 20 世紀の年とみなされます。

絶対時間は時間と日付の文字列として表示されますが、ソフトウェアの内部では協定世界時（UTC）すなわち旧来のグリニッヂ標準時（GMT）の 1904 年 1 月 1 日午前 12:00 から現在までの経過秒数で表されます。

オプションを選択すると、ダイアログボックスの右上の表示が変わることに注目してください。

時間と日付の有効範囲は、次に示すようにプラットフォームによって異なります。

- **（Windows）** 1970 年 1 月 2 日午前 12:00 ～ 2040 年 1 月 3 日午前 12:00
- **（Windows NT）** 1904 年 1 月 1 日午前 12:00 ～ 2040 年 1 月 3 日午前 12:00
- **（Macintosh）** 1904 年 1 月 2 日午前 12:00 ～ 2040 年 1 月 2 日午前 12:00
- **（UNIX）** 1904 年 1 月 1 日午前 12:00 ～ 2038 年 1 月 17 日午前 12:00

これらの範囲は、時間帯や夏時間を使用しているかどうかにより、最大で数日間延長くなる場合があります。

オートスケール

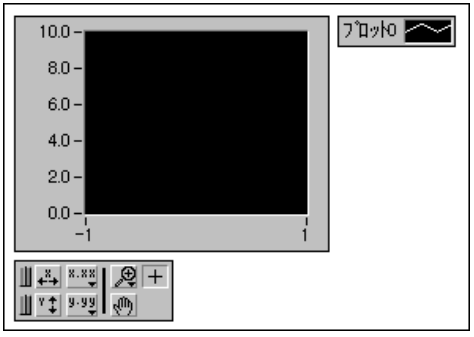
オートスケール（X または Y）は、自動スケールのオン／オフを切り替えるために使用します。

ルースフィット

ルースフィットをオンにすると、エンドマーカはスケールに対して使用している増分幅の倍数になるように調節されます。スケールが正確にデータの範囲に設定されるようにしたいときは、グラフのポップアップメニューで**ルースフィット**をオフにします。

パンオプションとズームオプション

グラフパレットは、フロントパネルにドロップするグラフすべてに組み込まれています。このパレットには、パン（グラフの表示領域をスクロールすること）の制御器とグラフの一部を拡大縮小するための制御器が用意されています。次の図にグラフと**グラフ**パレットを示します。



X 軸オートスケールボタン（左記参照）を押すと、*x* 軸のスケールが自動調節されます。*y* の自動スケールボタンを押すと、*y* 軸のスケールが自動調節されます。どちらか一方のスケールが連続して自動調節されるようにしたいときは、ロックスイッチ（左記参照）をクリックして自動スケールをオンにロックします。

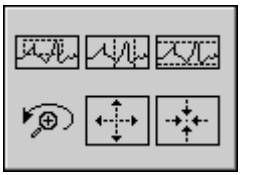
スケール形式ボタン（左記参照）を使用すると、*x* 軸あるいは *y* 軸のスケールマーカのフォーマットを実行時に制御することができます。

残りの 3 つのボタンは、グラフの操作モードを制御するために使用します。

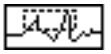
通常は、プラスまたは十字の記号で示される標準の操作モードになります。操作モードでは、グラフ内をクリックしてカーソルを移動することができます。

パンツール（左記参照）を押すとモードが切り替わり、グラフのプロット領域をクリックアンドドラッグしてデータをスクロールできるようになります。

ズームツール（左記参照）を押すと、グラフをズームイン（拡大）またはズームアウト（縮小）することができます。ズームツールをクリックすると、ズームの方法を選択するためのポップアップメニューが表示されます。このメニューを次の図に示します。



以下で、各項目について説明します。



四角形を使用してズームを実行。

四角形を使用して *x* 軸のデータに対してのみズームを実行（*y* 軸のスケールは変化しません）。

四角形を使用して *y* 軸のデータに対してのみズームを実行（*x* 軸のスケールは変化しません）。

最後に実行したズームを取り消し、グラフを前の設定に戻します。

1 つの点を中心にズームイン。1 つの点の位置でマウスボタンを押し続けると、マウスボタンを放すまでグラフが連続的にズームインします。

1 つの点を中心にズームアウト。1 つの点の位置でマウスボタンを押し続けると、マウスボタンを放すまでグラフが連続的にズームアウトします。

☞ **注**　**最後の 2 つのモード（ズームインとズームアウト）では、<Shift> キーを押しながらクリックすると別の方向にズームできます。**

凡例オプション

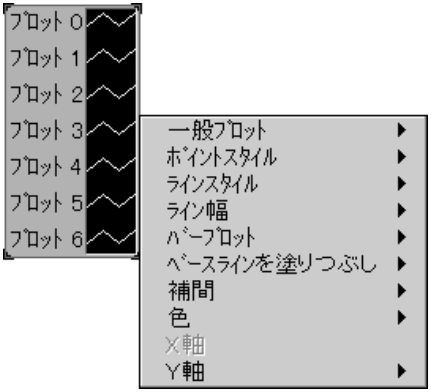
グラフは、カスタムのプロットスタイルを作成しない限り、新しい各プロットに対してデフォルトのスタイルを使用します。マルチプロットのグラフで、特定のプロットに特定の特性を使用したいとき（たとえば 3 番目のプロットの色を青にしたいとき）は、凡例を利用してこれらの特性を設定することができます。凡例は、グラフのポップアップメニューの**表示**サブメニューを使用して表示する／表示しないを選択することができます。また、凡例を使用して各プロットの名前を指定することもできます。

凡例を選択すると、1 つのプロットだけが表示されます。サイズ変更ツールで凡例のいずれかの角をドラッグすると、複数のプロットを作成することができます。プロットの特性を設定すると、グラフは凡例が表示されているいないにかかわらず、これらの特性をプロットに割り当てます。グラフは、凡例で指定されている数より多くのプロットを受け取った場合は、それらのプロットをデフォルトのスタイルで表示します。

グラフの本体を移動すると、それとともに凡例も移動します。ただし、凡例をドラッグすると、グラフに対する凡例の位置移動することができます。ラベルのスペースを広げたいときは凡例の左側でサイズを変更し、プロットサンプルのスペースを広げたいときは凡例の右側でサイズを変更します。

デフォルトでは、各プロットに 0 から順番に数字のラベルが付けられます。このラベルは、ほかのラベルの場合と同じ方法で変更することができます。プロットラベルの右側には、**プロットサンプル**が表示されます。各プロットサンプルに、プロット、ライン、色、およびプロットの点のスタイルを変更するためのポップアップメニューが用意されています。グラフに接続した点の配列は、グラフの凡例で割り当てた特性によって表示されます。

プロットサンプルのポップアップメニューを次の図に示します。



一般プロット項目を使用すると、プロットを 6 種類の一般的なスタイル（分散プロット、バープロット、0 までフィルプロットなど）のいずれかに簡単に設定することができます。このサブパレット内の項目は、ポイント、ライン、およびフィルスタイルを 1 回の操作でに設定することができます。

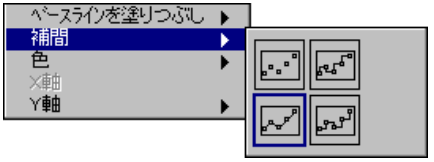
ポイントスタイル、ラインスタイル、ライン幅の各項目は、プロットを区別するためのスタイルを表示します。ライン幅のサブパレットには、デフォルト設定の 1 ピクセルよりも太いいくつかのライン幅のほか、ヘアラインもあります。ヘアラインは、画面上の表示には効果はありませんが、プリンタや印刷モードがヘアライン印刷をサポートしている場合は非常に細い線を印刷することができます。

☞ **注**　Windows では、ワイドペンは実線スタイルでしか使用できません。

バープロット項目には、垂直バー、水平バー、バーを使用しないの 3 つの選択肢があります。

ベースラインを塗りつぶし項目は、ベースラインでどこまで塗りつぶすかを指定します。**ゼロ**は、プロットから 0 で生成されたベースラインまでを塗りつぶします。**無限**は、プロットからグラフの正のエッジまでを塗りつぶします。**負の無限**は、プロットからグラフの負のエッジまでを塗りつぶします。このメニューの下の方にある項目を使用すると、このグラフの塗りつぶしたい特定のプロットを選択することができます。

補間項目は、次の図に示すようなパレットを表示します。このパレットでは、点と点を結ぶ線の描画方法を選択します。最初の項目は線を描画しないため、分散プロットに適しています。左下の項目は、点と点を直線で結びます。点と点を直角のエルボで結ぶ 2 ステップ項目は、ヒストグラム状のプロットを作成するのに適しています。右上の項目は y 軸方向の線を先に描き、右下の項目は x 軸方向の線を先に描きます。



色項目は、プロットの色を選択するためのパレットを表示します。プロットには凡例で色付けツールを使用して色を付けることもでき、また VI の実行中に色を変更することもできます。

Y 軸項目は、グラフの y 軸スケールのリストを表示します。この項目は、重なり合ったチャートの個々のプロットのスケールを指定するために使用します。

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制御器です。チャートとグラフは、チャートが前のデータを記憶できるという点で異なります。記憶する最大データ量は、ユーザが指定することができます。新しいデータは古いデータに追加されるため、現在の値を前のデータと照らし合わせることができます。

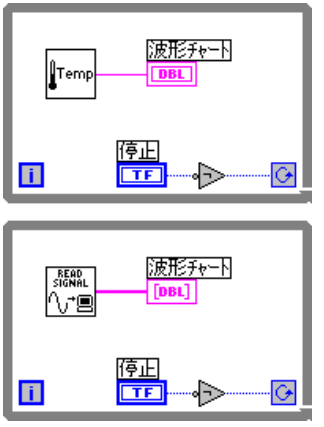
波形チャートのサンプルについては、
`examples¥general¥graphs¥charts.llb` を参照してください。

波形チャートのデータタイプ

チャートには、1 回に 1 つの値だけを渡すことも複数の値を渡すこともできます。波形グラフの場合と同様、個々の値は等間隔波形の一部として扱われ、個々の点は x 軸上で前の点から点 1 つ分だけ間をあけて配置されます。

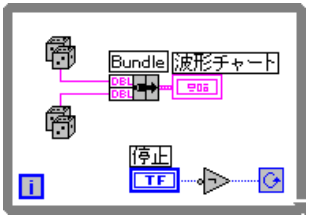
チャートには、1 つのスカラ値だけを渡すことも、複数の値からなる配列を渡すこともできます。チャートは、これらの入力値を 1 つのプロットに対する新しいデータとして扱います。

次のダイアグラムは、各データタイプごとのチャートの使用方法を示したものです。

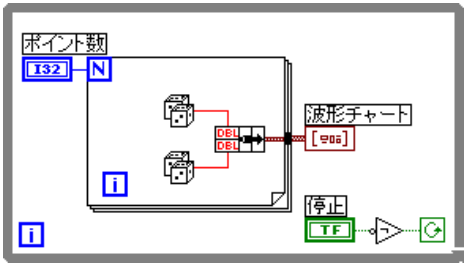


チャートに 1 回に 1 つずつ値を渡すよりも、一度に複数の値を渡した方がチャートを描画し直す回数は少なくなります。

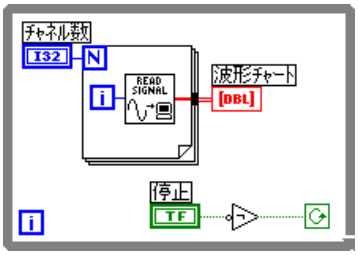
波形チャートに複数のプロットのデータを渡す方法としては、いくつかの方法があります。最初の方法は、データをスカラ数値のクラスタとしてまとめる方法です。この場合、個々の数値が各プロットの 1 つの点を表します。その例を次の図に示します。



一回の更新でプロットの複数の点を渡したいときは、数値のクラスタの配列をチャートに配線することができます。個々の数値は、各プロットの 1 つの点を表します。その例を次の図に示します。

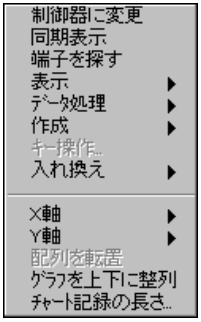


表示したいプロットの数が実行時にならないと決定できない場合、あるいは一回の更新でプロットの複数の点を渡したい場合は、2 次元配列のデータをチャートに渡すことができます。波形グラフの場合と同様、通常は行が各プロットの新しいデータとして処理されます。波形チャートのポップアップメニューの**配列を転置**を使用すると、列を各プロットの新しいデータとして処理させることができます。



波形チャートオプション

チャートは、凡例やパレットなどを含め、グラフとほとんど同じ機能を備えており、これらの機能の使用方法も同じです（詳しくは、本章の「スケールオプション」および「凡例オプション」の項を参照してください）。波形チャートでは、カーソルは使用できません。チャートのポップアップメニューを次の図に示します。



チャートのポップアップメニューの**表示**サブメニューを使用すると、オプション機能のデジタル表示やスクロールバーを表示するかしないかを指定することができます。**デジタル表示**は、プロット中の最新の値を表示します。ダイアグラムからチャートに渡された最後の値が、チャートにとって最新の値になります。デジタル表示は、各プロットに 1 つずつあります。

バッファに記憶されている古い値を見たいときは、スクロールバーを使用して前にプロットした値の範囲まで *x* 軸をスクロールするか、または *x* 軸のスケールの範囲を古いデータが表示されるように変更します。

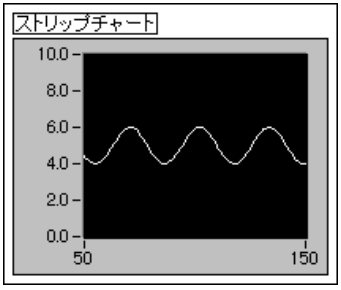
メモリが足りなくなるのを防ぐため、チャートがバッファに記憶できるデータの量には限界があります。このバッファのデフォルトサイズは 1,024 ポイントです。チャートがこの限界に達すると、新しいデータを書き込めるように古いポイントから順次破棄されます。このバッファ長は、チャートのポップアップメニューの**チャート記録の長さ...**項目を使用して変更することができます。

チャートの更新モード

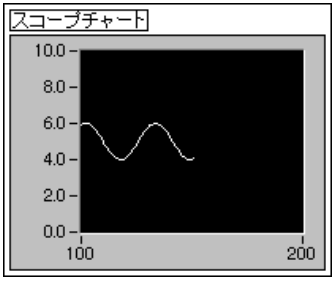
新しいデータを表示に追加する際のチャートの動作方法を変更するためには、次の図に示すように、チャートのポップアップメニューの**データ処理**サブメニューの**更新モード**項目を使用します。

以下で、ストリップチャート、スコープチャート、およびスイープチャートという 3 つの項目について図を示しながら説明します。

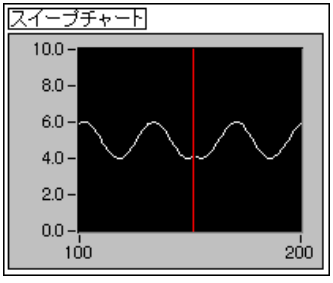
ストリップチャートモードでは、紙テープのストリップチャートレコーダとよく似たスクロール式表示器が表示されます。新しい値を受け取ると、その値が右端に表示され、古い値は左に移動します。



スコープチャートモードの表示は、オシロスコープとよく似ています。新しい値を受け取ると、その値は最後の値の右側に表示されます。プロットが表示領域の右端に達すると、プロットが画面から消え、再度左端からプロットの描画がスタートします。スコープチャートはスクロールに必要な負荷がないため、処理速度はストリップチャートよりも大幅に速くなります。

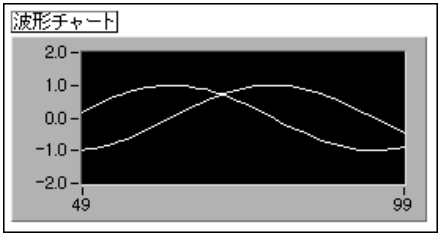


スウィープチャートモードは、スコープチャートとよく似ていますが、データが右端に達しても消えることはありません。そのかわりに、新しいデータの先頭を示す縦の線が表示され、新しいデータが追加されるたびにこの縦の線が表示領域に沿って移動します。

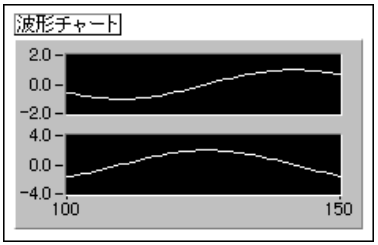


スタックプロットとオーバーレイプロット

デフォルト設定では、チャートは同じグリッド上に複数のグラフを描く場合と同じように、複数のプロットを重ね合わせて表示します。オーバーレイプロットの例を次の図に示します。



一方、チャートのポップアップメニューから**グラフを上下に整列**項目を選択すると、複数のプロットをスケールの異なる y 軸を利用して別々に表示することができます。この方法を使用すると、それぞれのチャートに対して範囲の異なる y 軸スケールを使用することができます。スタックプロットの例を次の図に示します。

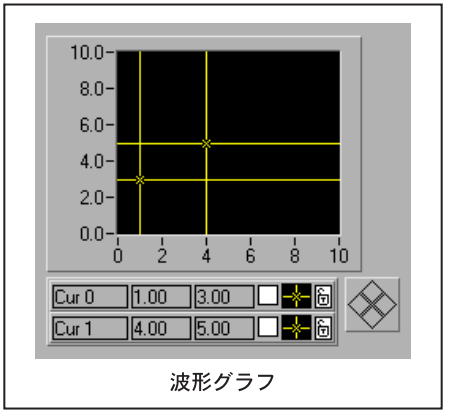


チャートにクラスタとしてデータを入力すると、チャートは正しい数のプロット表示を自動的にスタックします。データを 2 次元配列として入力した場合は、ポップアップメニューの**表示**サブメニューにある凡例を使用して正しい数のプロット表示を作成する必要があります。プロットの数を増やすために凡例の表示を拡大すると、スタックされたプロット表示の数も同じ数に増えます。

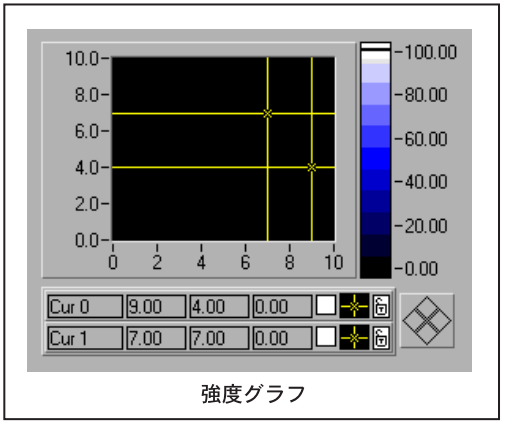
グラフカーソル

それぞれのグラフに、カーソルをグラフに配置するための**カーソル**パレットを表示することができます。そしてプロット上でカーソルにラベルを付け、カーソルをマーカとして使用することができます。カーソルをマーカとして使用する場合は、カーソルをデータプロットに固定してカーソルがデータの後を追うようにします。

次の図は、**カーソル**パレットが表示された波形グラフと強度グラフを示したものです。



波形グラフ

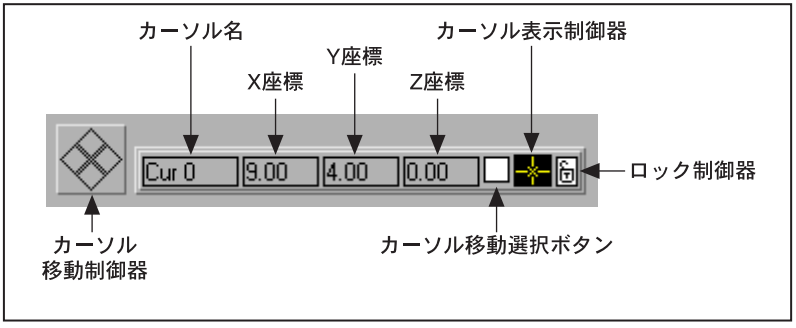


強度グラフ

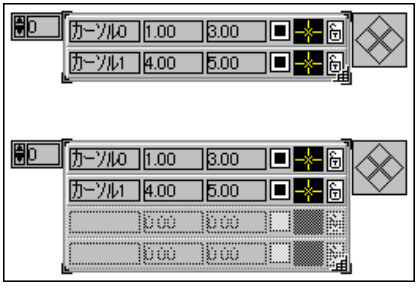
グラフの各カーソルは、下記のパーツを備えています。

- ラベル
- *xy*座標、および該当する場合は*z*座標
- プロットカーソルパッドで移動するためのプロットをマークするボタン
- カーソルの外観を制御するためのボタン
- カーソルをプロットに固定するか、自由に移動できるようにするかを指定するためのボタン

これらのパーツを次の図に示します。

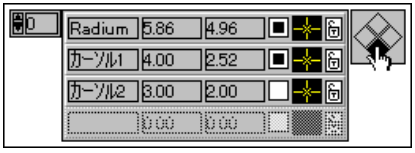


カーソルパレットは、配列と同じように機能します。パレットを拡張して複数のカーソルに関する情報を表示したり、指標制御器を使用して他のカーソルに関する情報をパレットに表示したりすることができます。指標制御器を表示するためには、カーソル表示上でポップアップメニューの**表示**項目を使用します。



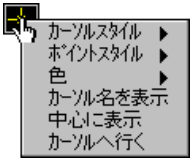
カーソルを消去するためには、**データの処理**のポップアップメニューで**選択項目の開始**と**選択項目の終了**の各項目を選択したのち、同じメニューの**データを切り取る**項目でカーソルを切り取ります。詳しくは、「第14章 配列とクラスタの制御器 および表示器」の「配列セルを選択する」の項を参照してください。

カーソルは、操作ツールを使用してドラッグするか、またはカーソル移動制御器を利用することにより、グラフ上で移動することができます。カーソルをドラッグする場合は、グラフでパンツールやズームツールが選択されていない状態にしておく必要があります。カーソル移動制御器の矢印をクリックすると、選択したすべてのカーソルが指定した方向に移動します。カーソルを選択するためには、操作ツールを使用してグラフ上で移動するか、該当するカーソルの選択ボタンをクリックします。次の例では、上の 2 つのカーソルが真下の方向に移動します。

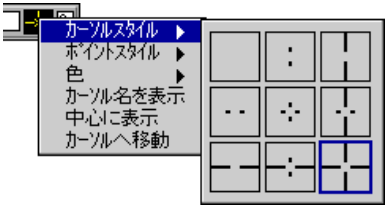


カーソル移動制御器

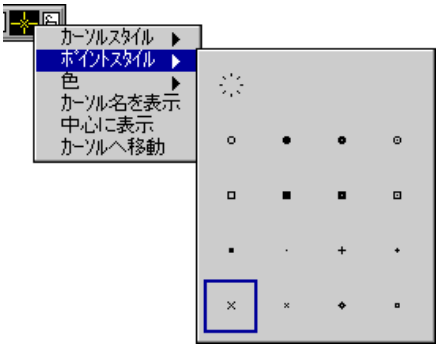
カーソル表示制御器を操作ツールでクリックすると、カーソルの形状やプロット上にカーソル名を表示するかどうかを指定するためのポップアップメニューが表示されます。このポップアップメニューを次の図に示します。



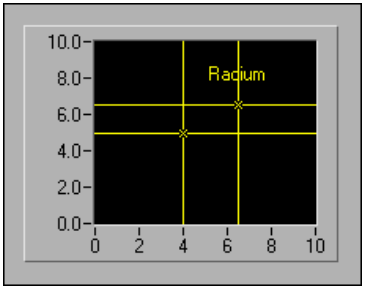
このメニューで、十字カーソルのスタイルを選選択します。十字カーソルには、次の図に示すように無限に延長できる縦線、横線、または十字、中心がカーソル位置に表示される短い縦線、横線、または使用しないといったスタイルがあります。



また、次の図に示すように、カーソル位置を示す点のスタイルやカーソルの色も選択できます。



次の図に示すようにプロット上にカーソル名を表示するためには、このメニューから**カーソル名を表示**項目を選択します。



中心に表示を選択すると、カーソルをグラフの表示領域内に戻すことができます。この項目は、カーソルが表示領域の外に出てしまった場合に使用すると便利です。この項目を選択すると、カーソル位置の（*x*,*y*）座標が変わります。

カーソルへ移動を選択すると、グラフの表示領域がカーソルの位置までスクロールされます。この場合、カーソルの位置は変わらず、選択したカーソルが表示領域に表示されるようにスケールが変化します。表示領域の大きさも変わりません。この機能は、カーソルで印を付けておいたグラフ上の特定の点（最大値や最小値など）を表示領域に表示したいときに使用すると便利です。

ロック解除　ロック

各カーソルの最後のボタンは、カーソルを特定のプロット上にロックしたいときに使用します。ロックボタンをクリックすると、ポップアップメニューが表示されます。このメニューを使用して、カーソルを特定のプロットに固定します。カーソルをプロットに固定すると、ボタンは締まった錠の表示に変わります。このポップアップメニューを次の図に示します。

ドラッグ項目は、マウスでカーソルを移動できるようにするかどうかを指定します。**ドラッグの許可**を選択した場合は、カーソルの移動すなわちドラッグが可能になります。メニューの区切り線より下にある項目は、マウスによるカーソルの移動方法を指定します。**ドラッグの許可**を選択すると、プロット上でカーソルを移動することはできなくなります。

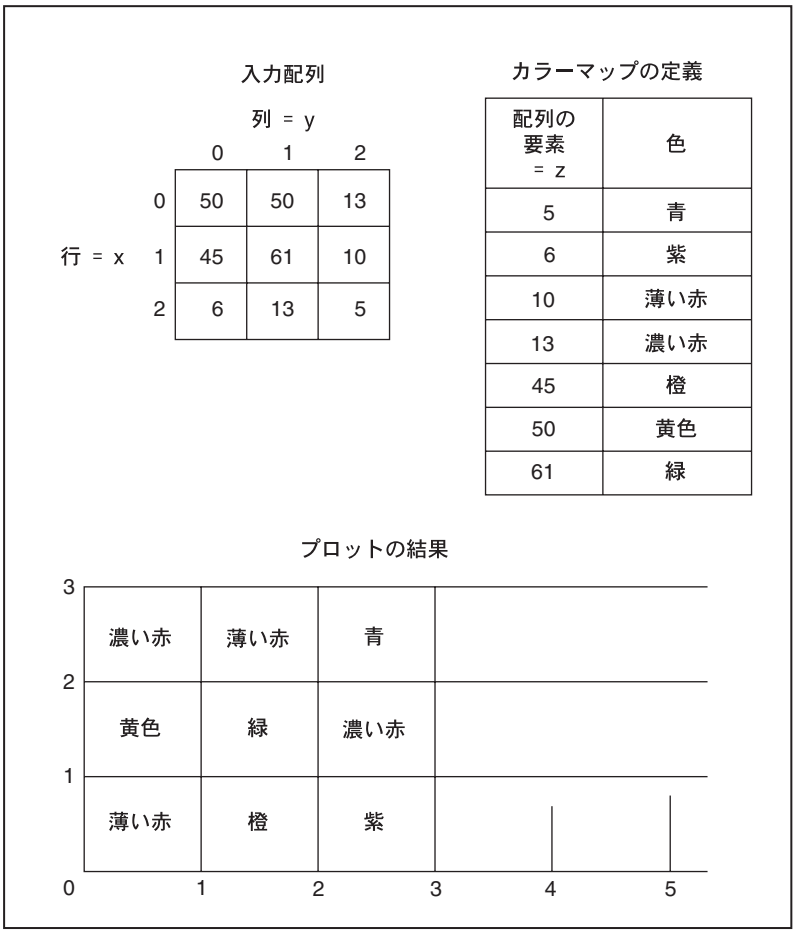
グラフの任意の場所にカーソルを表示あるいは移動できるようにしたいときは、**解放**を選択します。カーソルが常にプロット上の最も近い点にスナップされるようにしたいときは、**ポイントにスナップ**を選択します。**プロットにロック**は、カーソルを特定のプロットにロックしたいときに使用します。**プロットにロック**を最初に選択したときは、カーソルはプロット上の最初の点にロックされます。カーソルのロックを解除し、カーソルを別の場所に移動したあとで**プロットにロック**を選択すると、カーソルは最後にロックしたときの位置に移動します。

このメニューの 2 番目の区切り線の下には、カーソルをロックできるプロットのリスト（例：プロット 0、プロット 1、プロット 2、...）が表示されます。

グラフの属性ノードを使用したプログラム上でのカーソルやマーカの作成、制御、読み込みには、さまざまな項目があります。詳しくは、「第 22 章　属性ノード」を参照してください。

強度チャート

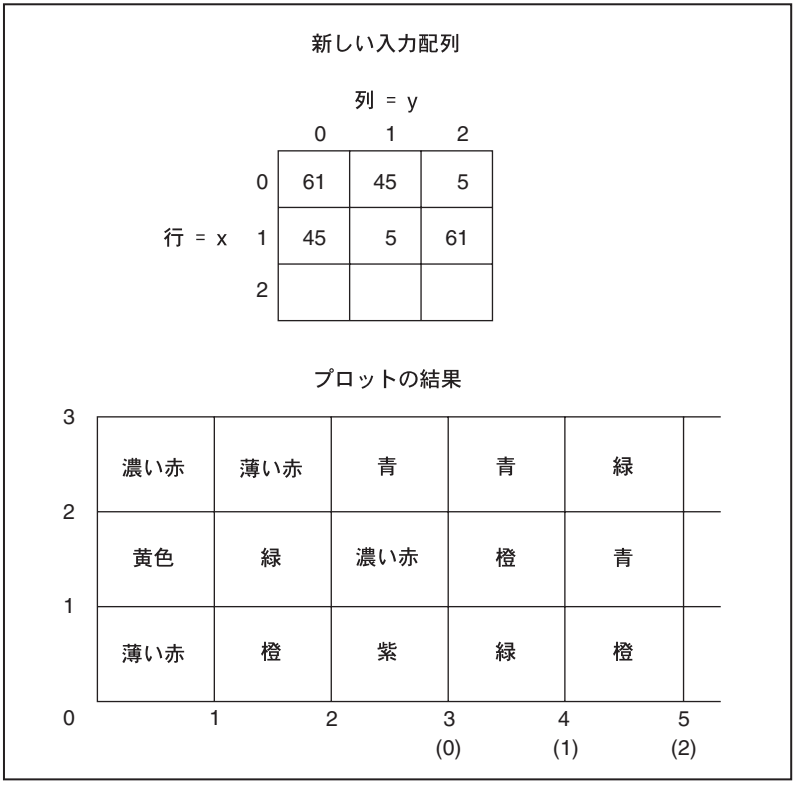
強度チャートは、デカルト平面に色のブロックを配置することによって 3 次元のデータを 2 次元プロット上に表示する手段です。強度チャートは、数字の 2 次元配列を受け付けます。配列中の個々の数字は特定の色を表します。2 次元配列の要素の指標は、この色のプロット上の位置を設定します。次の図は、強度チャートの操作の概念を示したものです。



強度チャートの色は、カラーバーを使用して対話形式で定義するか、またはチャートの属性ノードを使用してプログラム上で定義することができます。数字に色を割り当てる手順については、本章の「色のマッピング」の項で説明します。

配列の指標は、色のブロックの左下の隅に対応します。このブロックには、配列の指標で定義される単位面積があります。強度チャートは、最大256色まで表示できます。

データのブロックのプロットが作成されると、デカルト平面の原点が強度のデータブロックの右側に移動します。強度チャートに新しいデータが渡されると、新しいデータは次の図に示すように古いデータの右側に表示されます。



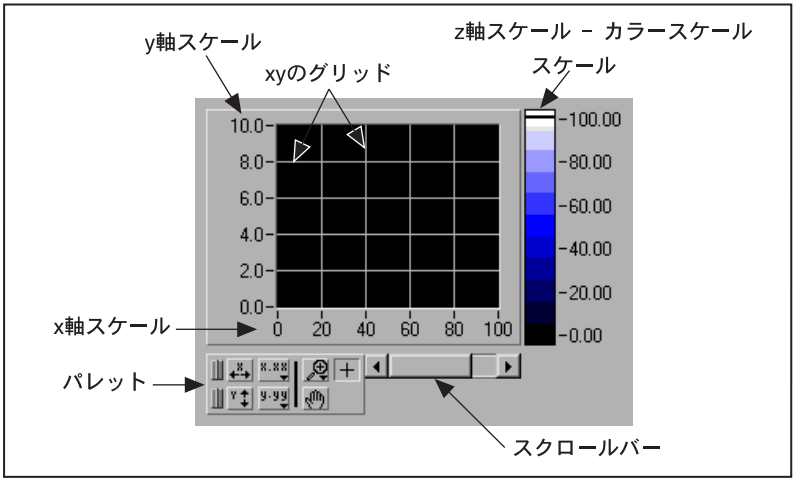
チャート表示器がいっぱいになると、古いデータはチャートの左側にスクロールされます。

強度チャートのサンプルについては、
`examples¥general¥graphs¥intgraph.llb` を参照してください。

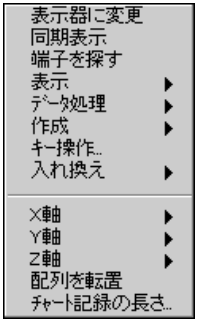
強度チャートのオプション

強度チャートには、他のチャートのオプションパーツも多数あります。そのほとんどは、表示するかしないかをグラフのポップアップメニューの**表示**サブメニューを使用して切り替えることができます。これらのパーツには、VIの実行中にスケールや形式を変更するためのパレットも含まれています。また、強度チャートには3番目の次元（色）があるため、値の範囲や色へのマップはカラーランプ表示器とよく似たスケールで定義されます。

次の図で、これらのオプションパーツをすべて表示したチャートを示します。



強度チャートには、データの表示をカスタマイズするための多数の項目があります。強度チャートのポップアップメニューを次の図に示します。



これらの項目は、そのほとんどが波形チャートのメニューの項目と同じものです。**表示**メニューを使用すると、z 軸スケールのカラーバーを表示するかしないかを指定することができます。**X 軸**メニューと**Y 軸**メニューは、波形チャートの**X 軸**メニューと**Y 軸**メニューと同じです。

強度チャートは、過去の更新データの履歴を記憶します。このバッファは、チャートのポップアップメニューの**チャート記録の長さ...**を選択することにより構成することができます。強度チャートのデフォルトサイズは 128 ポイント（点の数が 128 個）です。強度チャートの表示は、メモリを大量に使用する点に注意してください。たとえば、512 個の点の履歴と 128 個の y の値を持つ単精度のチャートを表示するためには、512 * 128 * 4 バイト（単精度のサイズ）、すなわち 256 キロバイトのメモリが必要になりま

す。強度チャートに大量のデータを表示したいときは、十分な量のメモリを確保してください。

強度チャートは、ストリップチャート、スイープチャート、およびスコープチャートの標準更新モードをサポートします。波形チャートの場合と同様、更新モードは**データ処理**メニューから選択します。

色のマッピング

色のマッピングは、カラーランプの数値制御器の色を定義する場合と同じように対話形式で設定できます。詳しくは、「第 9 章　数値制御器と数値表示器」の「カラーランプ」の項を参照してください。

属性ノードを使用してプログラム上で色を設定する方法は、2 通りあります。その 1 つは、カラースケールの場合と同じ方法で値対色のマッピングを属性ノードに設定する方法です。この方法を使用する場合は、Z Scale Info: Color Array 属性を指定します。この属性は、クラスタの配列で構成され、配列中の個々のクラスタには数値の限界値、およびその値に対応する表示色が含まれます。この方法を使用してカラーテーブルを指定する場合は、Z Scale Info: High Color 属性で上限より上の色を、Z Scale Info: Low Color で下限より下の色を指定することができます。色の総数は 254 色に制限されており、これに上限より上の色と下限より下の色を加えて合計 256 色となります。これより多くの色を指定すると、指定した色を補完することによって 254 色のカラーテーブルが作成されます。

色をプログラム上で設定するもう 1 つの方法は、Color Table 属性を使用してカラーテーブルを指定する方法です。この方法を使用すると、最大 256 の色の配列を指定することができます。チャートに渡されるデータは、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マッピングされます。カラースケールの範囲が 0 ～ 100 の場合、データ中の値 0 が指標 1 に、値 100 が指標 254 にマッピングされ、それより内側にある値は 1 と 254 の間で補完されます。0 未満の値は下限より下の値（指標 0）にマッピングされ、100 より大きい値は上限より上の色（指標 255）にマッピングされます。

☞ **注**　**強度チャート（またはグラフ）に表示できる色および色の数は、使用しているビデオカードが表示できる色および色の数に制限されます。また、使用するモニタによっても表示可能な色の数に制限があります。**

強度グラフ

強度グラフは、古いデータを記憶しない点を除いては、本質的に強度チャートと同じです。強度グラフでは、新しいデータが渡されるたびに新しいデータが古いデータと差し替えられます。

強度グラフのサンプルについては、`examples¥graphs¥intgraph.llb` を参照してください。

強度グラフのデータタイプ

強度グラフは、数字の 2 次元配列を受け付けます。個々の数字は、チャートによって色にマッピングされます。

渡されるデータの行は、チャートの新しい列として表示されます。行を行として表示したいときは、チャートのポップアップメニューの**配列を転置**を使用します。

強度グラフのオプション

強度グラフの機能は、チャートの更新モードがない点を除けば強度チャートとほとんど同じです。データの更新のたびに古いデータが新しいデータに入れ換えられるため、スクロールバーや履歴オプションはありません。

強度グラフでも、他のグラフと同じようにカーソルを使用することができます。個々のカーソルは、グラフ上の指定された点の *x*、*y*、および *z* 座標の値を表示します。グラフ上でのカーソルの操作についての詳細は、本章の「グラフカーソル」の項を参照してください。

色のマッピングの設定方法は、強度チャートの場合と同じです。

16

ActiveX 制御器

この章では、G ベースのソフトウェアとその他のアプリケーションとの対話を強化する Active コンテナの機能について説明します。

ActiveX のフロントパネルの拡張機能

フロントパネルには、次の図のような ActiveX サブパレットがあります。このサブパレットには、2 つの ActiveX コントロールである、ActiveX コンテナ、および ActiveX Variant が含まれています。

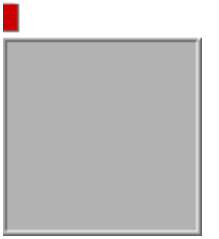
ActiveX Variant の制御器と表示器

ActiveX Variant の制御器と表示器を使用すると、ActiveX Variant のデータをソフトウェアに渡し、ActiveX のクライアントの機能を強化することができます。ブロックダイアグラムに ActiveX Variant の制御器および表示器を配置すると、次のように表示されます。

このフロントパネルオブジェクトは、ActiveX Variant のデータを表示データに変換する際に使用します。

ActiveX コンテナ

ActiveX コンテナを使用すると、ActiveX オブジェクトを VI パネルに組み込み、内蔵制御器と並べて配置することができます。このコンテナは、ActiveX コントロールおよびドキュメントをフロントパネル上に表示するために使用できます。次の図は、フロントパネルに最初にコンテナを配置したときの状態を示したものです。



ActiveX コンテナは、ブロックダイアグラム上にオートメーション refnum 端子として表示されます。この端子を Automation 関数に配線することにより、コンテナに格納されたオブジェクトを制御することが可能になります。そのオブジェクトが Automation インタフェースを備えていない場合は、端子の refnum は無効となるため Automation 関数で使用することはできません。

フロントパネルのコンテナにオブジェクトを挿入するには、ポップアップメニューで**ActiveX オブジェクトの挿入**を選択します。**ActiveX オブジェクトを選択**ダイアログボックスが表示されます。

コンテナに格納できるオブジェクトには、ActiveX ドキュメントと ActiveX コントロールという一般的な 2 つのタイプがあります。

ActiveX ドキュメント — これらのオブジェクトはコンテナオブジェクトに格納でき、ポップアップメニューで**オブジェクトの編集**オプションを選択することにより編集できます。**オブジェクトの編集**を選択すると、オブジェクトを編集するための新しいウィンドウが表示されます。文書の中にはオートメーションインタフェースをサポートしているものもあり、このような文書に対してはダイアグラム上でオートメーション関数を使用することができます。

ActiveXコントロール — これらのオブジェクトは、コンテナオブジェクトに格納した時点でアクティブになり、操作できるようになります。また、オートメーションインタフェースを備えているため、ダイアグラム上で Automation 関数を使用して制御することができます。

オブジェクトは、3 通りの方法でドロップできます。ドロップ方法は、ダイアログボックスの上の選択ボックスに表示されます。

ドキュメントを作成 — システムに登録されたものの中からドキュメントタイプを選択します。

ファイルからオブジェクトを作成 — ファイルシステム内にあるファイルを選択します。オブジェクトは、ファイルにリンクしたままにしておくこともできますが、パネルに静的にコピーすることもできます。

コントロールを作成 — システムに登録されているものの中から ActiveX コントロールを選択します。

2 種類のオブジェクトすなわちコントロールとドキュメントは、コンテナに格納することができます。コントロールやドキュメントを新たに作成することも、既存のドキュメントを挿入することもできます。新たなドキュメントを作成する場合は、**ドキュメントを作成**を選択し（上の図参照）、表示されたリストの中からオブジェクトのタイプを選択します。

既存のドキュメントつまりファイルを挿入する場合は、**ファイルからオブジェクトを作成**を選択します。ダイアログボックスが次の図のように変わります。

参照...を使用して挿入したい文書を探します。**ファイルにリンク**オプションを選択した場合は、フロントパネルオブジェクトが更新される際にドキュメントも更新されます。**ファイルにリンク**を選択しなかった場合は、ドキュメントの静的バージョン（元のファイルと連動しないバージョン）が挿入されます。

既存のコントロールを挿入する場合は、**コントロールを作成**を選択します。ダイアログボックスが次の図のように変わります。

使用可能なコントロールのタイプがシステムに登録されています。

ActiveX パレットを作成する

プロジェクト→ ActiveX コントロールをインポート ... を選択すると、ActiveX コントロールのセットをカスタム制御器に変換してパレットのメニューに追加することができます。

メニュー項目を選択すると、システムのコントロールのリストが表示されます。このリストで、1 つまたは複数のコントロールを選択することができます。カスタム制御器を保存するディレクトリまたは `.llb` ファイルを選択するよう指示するプロンプトが表示されます。デフォルト設定の保存先は、`user.lib` になります。これは、このデレクトリのすべてのファイルが自動的にパレットメニューに表示されるためです。

第III部

ブロックダイアグラムのプログラミング

第III部では、Gでのブロックダイアグラムの作成および操作に必要なコンポーネントについて説明します。

「第III部 ブロックダイアグラムのプログラミング」の各章の内容は下記の通りです。

- 「第17章 ブロックダイアグラムの概要」では、ブロックダイアグラムを作成する際に使用する3つのコンポーネントのうち、端子とノードについて説明します。もう1つのコンポーネントである配線については、「第18章 ブロックダイアグラムを配線する」で説明します。
- 「第18章 ブロックダイアグラムを配線する」では、ブロックダイアグラムの端子をワイヤを利用して接続する方法について説明します。
- 「第19章 ストラクチャ」では、Forループ、Whileループ、Caseストラクチャ、およびシーケンスストラクチャの使用方法について説明します。これらのストラクチャは、**関数→ストラクチャ**パレットに入っています。
- 「第20章 フォーミュラノード」では、ブロックダイアグラムで数学式を実行するためのフォーミュラノードの使用方法について説明します。フォーミュラノードは、**関数→ストラクチャ**パレットから呼び出すことができます。
- 「第21章 VIサーバ」では、VIやアプリケーションのプログラム制御メカニズムについて説明します。また、VIやアプリケーションの遠隔制御の方法についても説明します。
- 「第22章 属性ノード」では、フロントパネル制御器の属性をプログラムを利用して設定したり、読み取ったりするための属性ノードの使用方法について説明します。便利な属性としては、表示色、制御器の表示、リング制御器のメニュー文字列、グラフやチャートのプロットの色、グラフカーソルなどがあります。
- 「第23章 グローバル変数とローカル変数」では、グローバル変数とローカル変数の定義方法および使用方法について説明します。グローバル変数を利用すると、複数のVIから特定の値のグループを簡単に呼び出すことができます。ローカル変数は、同様の操作を単一のVIから行う場合に使用します。

17

ブロックダイアグラムの概要

この章では、ブロックダイアグラムを作成する際に使用する 3 つのコンポーネントのうち、端子とノードについて説明します。もう 1 つのコンポーネントである配線については、「第 18 章　ブロックダイアグラムを配線する」で説明します。

端子とノード

ブロックダイアグラムは、端子、ノードおよびワイヤを使用して作成します。

端子は、ブロックダイアグラムとフロントパネル間、およびブロックダイアグラム上のノード間でのデータの受け渡しに使用されるポートのことです。端子は、関数や VI のアイコンの基本をなすものでもあります。次の図は、左が端子パターン、右がそのアイコンを示したものです。関数または VI の端子を表示するためには、アイコンをポップアップして**表示→端子**を選択します。

ノードは、プログラムの実行要素です。ノードは、従来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相当します。

ワイヤは、入力端子と出力端子の間のデータの通路です。

端子

G には、さまざまなタイプの端子があります。一般には、ワイヤを接続できる場所はいずれも端子です。G には、制御器や表示器の端子、ノードの端子、**定数**、ストラクチャ上の特殊な端子があります。

データを渡す端子（フロントパネルの制御器の端子、ノードの出力端子、定数など）は、**ソース端子**とも呼ばれます。ノードの入力端子やフロントパネルの表示器の端子はデータを受け取るため、**接続先端子**または**シンク端子**とも呼ばれます。

制御器と表示器の端子

値をフロントパネルの制御器に入力すると、VIの実行時にこれらの値が制御器の端子からブロックダイアグラムに渡されます。VIの実行が終了すると、表示器の端子を通じて出力データがブロックダイアグラムからフロントパネル制御器に渡されます。Gの一部の制御器と表示器の端子の記号を、「表 17-1 Gの制御器および表示器の端子の記号」に示します。それぞれの記号は、制御器や表示器のデータタイプを示す絵と色で表され、数値の制御器および表示器の場合は数値の表記法も示されます。制御器の端子は、表示器の端子よりも太い枠で表示されます。端子はそれに対応する制御器または表示器に所属しているため、端子を削除することはできません。制御器や表示器を削除したいときは、フロントパネルから削除します。

配列端子は、1つのデータタイプを角カッコで囲み、配列の要素のデータタイプの色で表示します。

表 17-1 Gの制御器および表示器の端子の記号

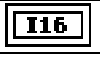
制御器	表示器	説　明	色
		拡張精度浮動小数点	オレンジ
		倍精度浮動小数点	オレンジ
		単精度浮動小数点	オレンジ
		拡張精度複素数浮動小数点	オレンジ
		倍精度複素数浮動小数点	オレンジ
		単精度複素数浮動小数点	オレンジ
		符号なし32ビット整数	青
		符号なし16ビット整数	青
		符号なし8ビット整数	青
		32ビット整数（倍長ワード）	青
		16ビット整数（ワード）	青
		8ビット整数	青
		クラスタ	茶またはピンク
		配列	可変

表 17-1 Gの制御器および表示器の端子の記号（続き）

制御器	表示器	説　明	色
		パス	水色
		Refnum	水色
		ブール	緑
		文字列	ピンク
		列挙体	青
		OLE Variant	紫

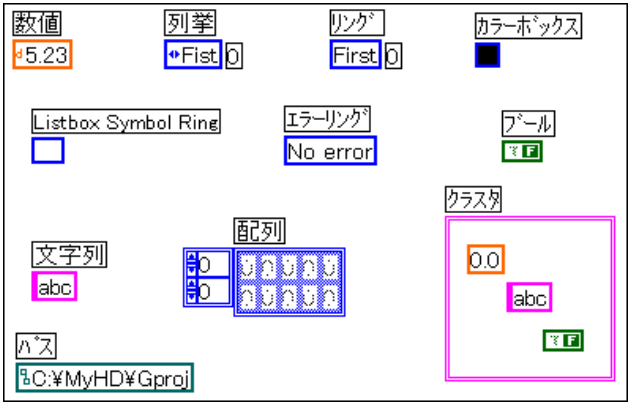
定数

定数はダイアグラム上にある端子で、ブロックダイアグラムに直接データ値を渡します。**ユーザ定義定数**は、プログラムの実行前に定数を編集することによって定義され、実行中にその値を変更することはできません。**ユニバーサル定数**の値は固定値です。

ユーザが定義する定数

ユーザが定義する定数の最も簡単な作成方法は、入力または出力をポップアップして**定数を作成**を選択する方法です。これらの定数は、そのタイプによっては**関数**パレットの中のさまざまなパレットにも含まれています。そのほとんどは、各パレットのいちばん下またはいちばん上にあり、数値、列挙、およびリングの定数はいちばん下の行、すなわち**数値**パレットにあります。カラーボックス定数、リストボックスの記号リング定数、およびエラーリング定数という 3 つの定数は、**数値→その他の数値定数**サブパレットにあります。パス定数は、**ファイル I/O →ファイル定数**サブパレットにあります。

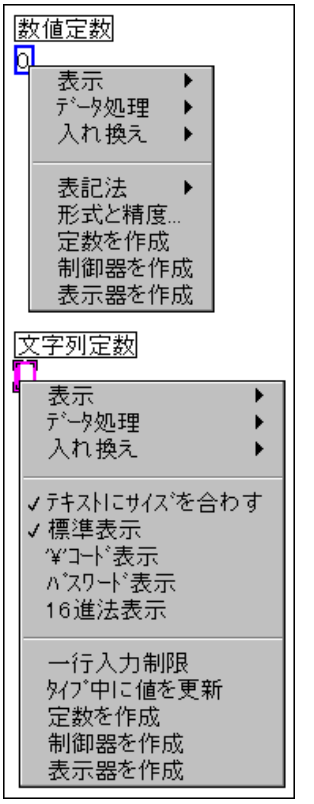
これらの定数を次の図に示します。



定数にラベルを付けるには、定数をポップアップして**表示→ラベル**を選択します。ハイライト表示のテキストボックスが表示されます。これは、ツールを変更せずに直接文字を入力できることを示しています。

ラベルのようにユーザが定義する定数は、入力したデータ長に合わせて自動的にサイズが変更されます。ラベルあるいは文字列定数のサイズや形を変更したいときは、そのポップアップメニューから**テキストにサイズを合わす**を選択すると、定数あるいはラベルはそのデータ長に合わせて自動的にサイズが調節されます。

ユーザが定義する定数の値を設定するには、フロントパネル上のデジタル制御器、ブールのスライドスイッチ、文字列制御器の場合と同じように操作ツールを使用します。**数値定数**や**文字列定数**は、フロントパネルの数値制御器とよく似ており、そのポップアップメニューも次に示すように制御器のポップアップメニューとよく似ています。



矢印キーを使用すると、新しい数値定数または値を選択した数値定数を増分または減分することができます。この方法は特に、1、2、3 といった小さい値を持つ制御器パラメータをプログラミングする場合に便利です。

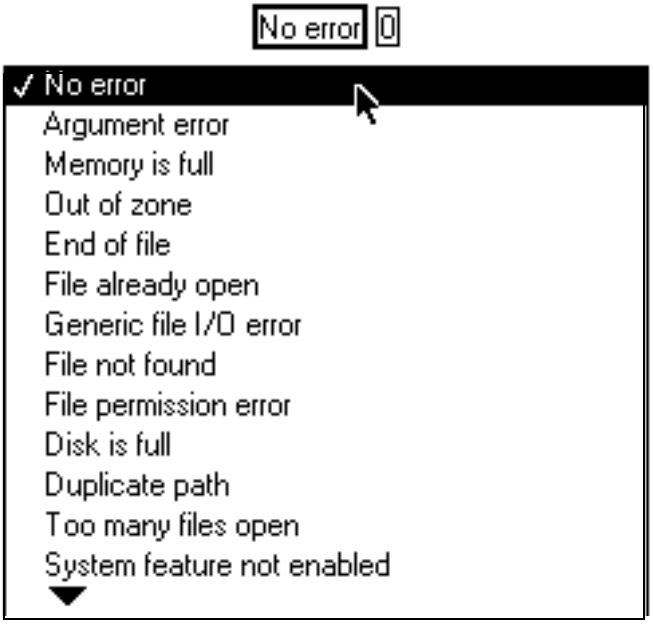
リング定数は、フロントパネルのリング制御器と同じように数字をテキストと関連づけます。リング定数の値は、符号なしの 16 ビット整数です。リング定数の表記法は、複素数以外の数値タイプに変更することができますが、その場合でも値は常に 0 から *n*–1 までの整数になります。

列挙定数は、ニーモニック（整数に関連付けられた文字列）がタイプの一部とみなされる点を除いては、リング定数と同じです。列挙を Case ストラクチャの選択端子に配線した場合は、ケースは従来の数値ではなく列挙のニーモニックに従って名前が付けられます。列挙の数値の表記法は、常に符号なしのバイト、ワード、または倍長になります。

色は、カラーボックス制御器に対して使用する**カラーボックス定数**から選択することができます。カラーボックス制御器の値は、特定の色と相関関係にあります。カラーボックスを設定するには、色付けツールまたは操作ツールでカラーボックスをクリックし、カラーパレットから使用したい色を選択します。

リストボックス記号リング定数は、リストボックス制御器の項目に記号を割り当てるために使用します。

エラーリング定数は、前もって定義するリングです。操作ツールで定数をクリックし、表示されたダイアログボックス（次の図参照）から使用したいエラーメッセージを選択します。このリングは、ブロックダイアグラムをより記述的なものにできるため、ファイル I/O 関数のようにエラーコードやエラークラスタを返す関数に使用すると便利です。たとえば、Open File 関数を使用して存在しないファイルを開こうとすると、関数はエラーコードを返します。このエラーコードの出力値と File Not Found (7) に設定されたエラーリングの設定とを比較することにより、エラー条件をテストすることができます。



パス定数を使用すると、ブロックダイアグラム上で定数のパス値を作成することができます。

パス、文字列、およびカラーボックスの定数はサイズを変更することができますが、ブール、数値、リング、および列挙の定数はサイズを変更することはできません。数値定数は、タイプした入力値に応じた表記を使用し

ます。たとえば、[1.1] とタイプしたとすると、定数はDBLになります。これは、**表記法**ポップアップメニューの**入力値に合せる**の設定によって制御されます。

入力値に合せるの表記法は、新しい値をタイプしたときに数値定数がその表記法を自動的に決定するかしないかを指定します。ブロックダイアグラムの数値定数は、新しく作成する際にデフォルト設定の入力値に合せるに設定されます。たとえば、DBL数値定数に2をタイプした場合、タイプは自動的にI32に変わります。定数の表記法をポップアップメニューを使用して変更した場合は、指定したタイプが変更されないように**入力値に合せる**は自動的にオフになります。入力した値に基づいて自動的にタイプを決定する方法に戻すためには、表記法のポップアップメニューから**入力値に合せる**を選択する必要があります。

数値定数のデフォルトの表記法は、入力した数値が少数点数のときは倍精度の浮動小数点数、整数のときは倍長の整数、複素数のときは倍精度の複素数になります。たとえば、['123'] と入力した場合は表記法は倍長の整数になり、['123.'] と入力した場合は倍精度の浮動小数点数になります。表記法は、定数のポップアップメニューの**表記法**項目を使用して変更することができます。

汎用定数

汎用定数には、汎用数値定数と汎用文字列定数という2つのタイプがあります。

- 汎用数値定数 — πや光の速度など、正確でかつ一般に使用される数学的な値や物理的な値のセット。
- 汎用文字列定数 — 行送りや改行など、一般に使用される表示不可能な文字列のセット。

これらの定数についての詳細は、LabVIEWまたはBridgeVIEWのオンラインリファレンスを参照してください。(**ヘルプ→オンラインリファレンス→関数およびVIリファレンス**)

ノード

ノードは、ブロックダイアグラムの実行要素です。ノードには、**関数**、**サブVI**、**ストラクチャ**、**コードインタフェースノード（CIN）**、**フォーミュラノード**、**属性ノード**の6つのタイプがあります。関数ノードとサブVIノードは、ブロックダイアグラムでの形や役割はよく似ていますが、実際には大きな違いがあります。関数については、本章の「関数」の項で説明します。サブVIについては、「第3章　サブVIを使用する」で説明します。

ストラクチャは、実行順序を制御するGのデータフロープログラミングモデルを補足するものです。本章でも「ストラクチャ」の項で簡単に説明し

ていますが、詳しい説明については「第 19 章　ストラクチャ」を参照してください。

CIN は、ブロックダイアグラムと C や Pascal などの従来のプログラミング言語で作成したコード間のインタフェースです。詳しくは、「LabVIEW Code Interface Reference Manual」を参照してください。このマニュアルは、ソフトウェアのプログラムディスクまたは CD に PDF ファイルの形式で収録されています。

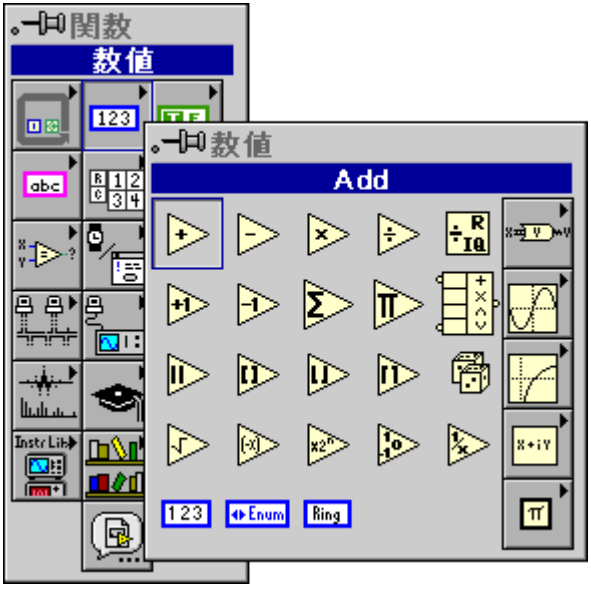
フォーミュラノードは、ブロックダイアグラム上で公式を使用して G の関数を補足します。「第 20 章　フォーミュラノード」で説明します。

属性ノードは、制御器の属性をプログラム上で変更します。詳細は「第 22 章　属性ノード」で説明します。

関数

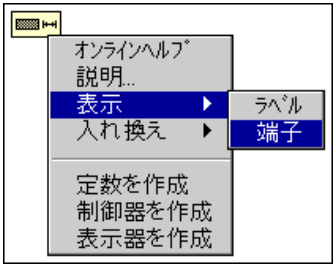
関数は、G に組み込まれた基本ノードです。これらは、数字の加算、ファイルの I/O、文字列のフォーマットなどの基本操作を実行します。関数には、フロントパネルやブロックダイアグラムはありません。関数をコンパイルすると、マシンコードが生成されます。個々の関数についての詳細は、**オンラインリファレンス→関数および VI リファレンス**を参照してください。

関数は、次の図に示すように関数パレットから選択します。



関数を選択すると、ブロックダイアグラムにそのアイコンが表示されます。関数のラベルを表示するには、アイコンをポップアップして**表示→ラベル**を選択します。ラベリングツールでテキストをハイライト表示にし、文字を入力すると、ラベルを変更することができます。関数のラベルは、ブロックダイアグラム上で関数の目的を示すために使用することができます。

ヘルプウィンドウを使用すると、関数の配線方法を確認することができます。あるいは、次の図に示すようにアイコンのポップアップメニューで**表示→端子**を選択すると、端子の正確な位置を知ることができます。

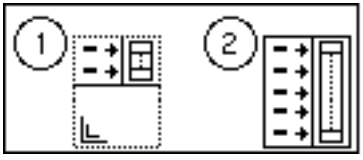


端子が表示されている状態で再度**表示→端子**を選択すると、今度は関数のアイコンを表示されます。

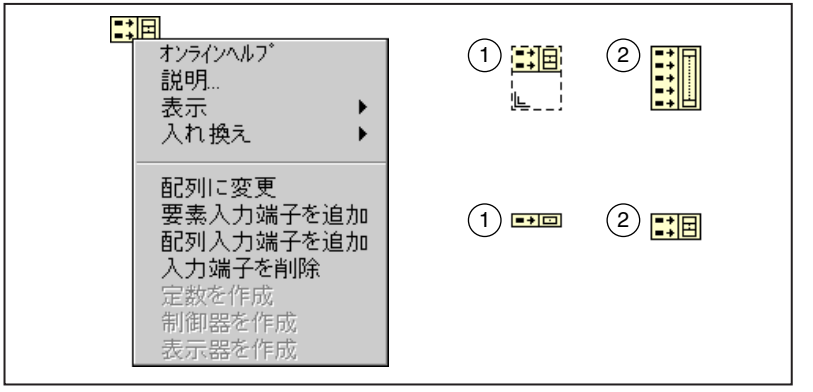
関数に配線する際には、関数の端子の1つに配線します。

関数の中には、端子の数が可変のものもあります。たとえば、3つの要素からなる配列を作成する場合、Build Array関数には3つの入力端子が必要となりますが、10個の要素からなる配列を作成する場合には10個の端子が必要になります。

拡張可能な関数の端子の数を変更する場合は、他のオブジェクトの場合と同様、次の図に示すようにサイズ変更ツールでアイコンのサイズを変更します。関数を拡大したり縮小したりすることは可能ですが、縮小によって配線した端子が消えてしまう場合には縮小することはできません。



端子の数は、次の図に示すように端子のポップアップメニューの**入力端子を追加**コマンドまたは**入力端子を削除**コマンドを使用して変更することもできます。**入力端子を削除**コマンドは、ポップアップした端子を削除し、端子にワイヤが配線されている場合はその配線を解除します。**入力端子を追加**コマンドは、ポップアップした端子のすぐ後に端子を追加します。これらのコマンドのフルネームは、関数によって異なります。



ストラクチャ

プログラムを作成する際には、コードの一部を指定した回数だけ繰り返し実行したり、あるいはある条件が真（TRUE）の間繰り返し実行することが必要になる場合があります。また、存在する条件に応じてコードの別の部分を実行したり、コードをある一定の順序で実行することが必要になる場合もあります。G には、**ストラクチャ**と呼ばれる 4 つの特殊なノードがあります。これらのノードを使用すると、G のデータフローのフレームワークの中で普通なら不可能な方法でコードを実行することも可能になります。

それぞれのストラクチャは、サイズの変更が可能な独自の枠を持っています。この枠の中に、そのストラクチャの特殊なルールに基づいて実行するコードが格納されます。そのため、ストラクチャの枠の中のダイアグラムはサブダイアグラムと呼ばれます。サブダイアグラムは、階層的に構成することができます。

G には、コードを実行するための次のようなストラクチャがあります。

- For ループ — 実行を一定回数繰り返します。
- While ループ — ある条件が TRUE の間サブダイアグラムの実行を繰り返します。

- Caseストラクチャ — 格納された複数のサブダイアグラムの中から、ストラクチャの選択端子に渡された値に応じていずれか 1 つのサブダイアグラムだけを実行します。
- シーケンスストラクチャ — サブダイアグラムの番号順にコードを実行します。

ストラクチャはノードであるため、そのノードを他のノードに接続するための端子を備えています。たとえば、ストラクチャへのデータの入出力を行う端子は、**トンネル**と呼ばれます。トンネルは、ストラクチャの外部および内部からのワイヤに対する可動接続ポイントです。詳しくは、「第 19 章 ストラクチャ」を参照してください。

ブロックダイアグラムオブジェクトの入れ換えと挿入

仮に、本来Decrement関数を使用すべきところにIncrement関数を使用しているものとします。この場合、Increment関数のノードを削除したのち、**関数**パレットから Decrement ノードを選択し、配線し直すことができます。あるいは、Increment ノードのポップアップメニューの**入れ換え**項目を使用することもできます。その場合は、**入れ換え**を選択して**関数**パレットを呼び出し、Decrement 関数を選択します。この方法には、古いノードが新しいノードに入れ換えられ配線の手間が省けるという利点があります。関数は他のどんな関数にも置き換えることができますが、それぞれの関数ノードの端子の数やデータタイプが異なる場合にはワイヤが切断される可能性があります。

また、**入れ換え**は定数を別の定数に置き換えたり、ストラクチャを別のよく似たストラクチャに置き換える場合（たとえば While ループを For ループと置き換える場合など）にも使用できます

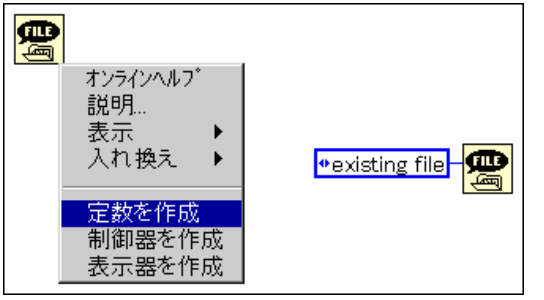
☞ **注　サブ VI で入れ換えを使用する際に、すでにメモリに入っている VI の名前と同じ名前を選択した場合は、置き換えられたノードは選択した VI ではなくすでにメモリに入っている VI を指します。**

ワイヤのポップアップメニューには、**挿入**という項目があります。**挿入**を選択すると**関数**パレットが呼び出されます。このパレットで関数または VI を選択すると、選択したノードはポップアップしたワイヤに接続されます。ただし、ノードに入力端子や出力端子が複数存在する場合は、ワイヤはかならずしも予期した端子に接続されるとは限らないため、配線を入念にチェックする必要があります。

定数、制御器、および表示器を自動的に追加する

メニューから定数、制御器、あるいは表示器を選択し、それらを手作業で端子に配線するかわりに、端子をポップアップして**定数を作成**、**制御器を作成**、または**表示器を作成**を選択すると、正しいデータタイプのオブジェクトを自動的に作成することができます。そうすることが適当である場合には、新しく作成した定数、制御器、または表示器のワイヤは自動的に配線されます。端子がType Defに対応している場合は、Type Defの定数、制御器、または表示器を作成します。Type Defが後で更新されると、制御器、表示器、または定数も更新されます。

たとえば、File Dialog関数の**選択モード**入力の定数が必要な場合は、入力をポップアップして**定数を作成**を選択します。すると、**選択モード**入力に使用可能なすべての値を含む列挙タイプが自動的に作成されます。モードの入力がまだどこにも配線されていない場合は、新しい定数は自動的に配線されます。



定数を作成、**制御器を作成**、または**表示器を作成**を使用すると便利な場所としては、ほかに関数やVIの出力、ワイヤ、定数、フロントパネルの制御器や表示器の端子などがあります。

18

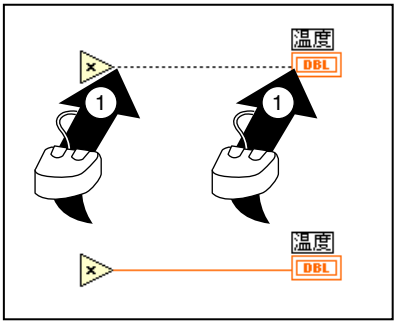
ブロックダイアグラムを配線する

この章では、ブロックダイアグラム上の端子を配線することによって相互に接続する方法について説明します。

配線方法

端子の接続には、配線ツールを使用します。次の図に示すように、カーソルが指すポイントすなわちツールのホットスポットは、ワイヤの巻き取り器から出ているワイヤの先端に位置します。

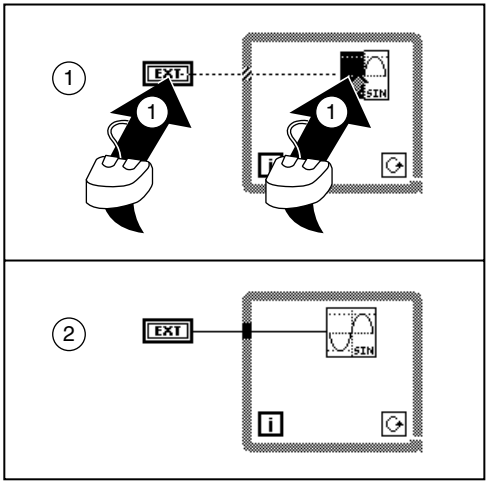
左記の記号はマウスを表します。この章の配線図では、このマウス記号の先端の矢印がクリックする場所を示し、マウスボタンの上に印刷された数字がクリックする回数を示します。



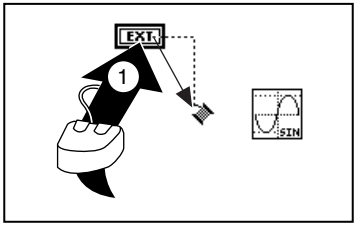
ある端子から別の端子に配線するには、上の図に示すように配線ツールで最初の端子をクリックしたのち、ツールをもう一方の端子に移動してその端子をクリックします。マウスを最初の端子からもう一方の端子に移動する際には、マウスボタンを放します。また、どちらの端子を先にクリックしてもかまいません。配線ツールのホットスポットが端子上に正しく位置していると、端子領域が点滅します。その端子をクリックすると、その端子にワイヤが接続されます。一方の端子にワイヤを接続したのちダイアグラム上でカーソルを動かすと、巻き取り器からワイヤが引き出されるようにして端子とツールの間にワイヤが描画されます。このとき、マウスボタンを押し続ける必要はありません。

既存のワイヤから配線するときは、既存のワイヤを起点または終点として上で述べた手順で配線します。既存のワイヤに新しいワイヤを接続できる正しい位置に配線ツールを移動すると、ワイヤが点滅します。

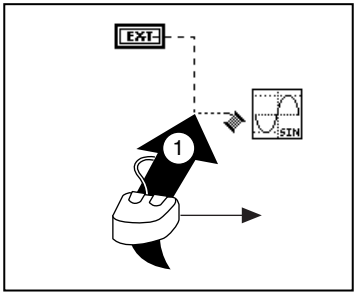
ストラクチャの外側の端子からストラクチャの内側の端子には、基本的な配線手順で直接接続することができます。ストラクチャの枠と交差する箇所には、次の図に示すように自動的にトンネルが作成されます。



ワイヤが端子から垂直または水平のどちらの方向に延びるかは、配線ツールを最初にどちらの方向に動かすかによって決まります。ツールを上または下に動かした場合はワイヤは垂直方向に延び、ツールを左または右に動かした場合はワイヤは水平方向に延びます。端子上でマウスボタンをクリックすると、次の図に示すようにホットスポットの正確な位置とは無関係に端子の中心にワイヤが接続されます。

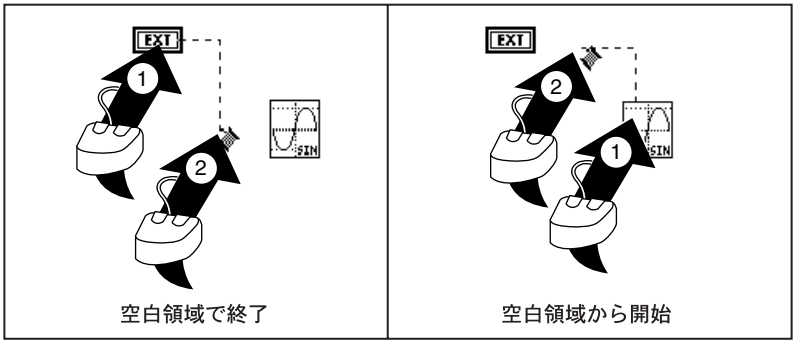


ワイヤは、クリックしなくても1回だけ90度曲げることができます。空白領域をクリックすると、次に示すようにさらに90度方向転換することができます。

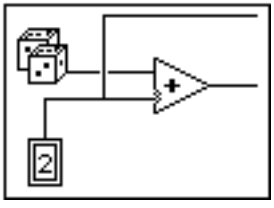


☞ **注**　**ワイヤの方向（垂直方向／水平方向）は、スペースバーを押して切り替えることもできます。最後の方向転換を取り消す場合は、<Ctrl>（Windows）、<option>（Macintosh）、または<中ボタン>（UNIX）を押しながらクリックします。最後に方向転換した位置が最初にクリックした端子またはワイヤである場合は、方向転換を取り消すとワイヤ全体が削除されます。**

次の図に示すように、配線ツールで空白領域をダブルクリックするとワイヤの起点あるいは終点を空白領域に置くことができます。



ワイヤが交差する箇所では、先に接続したワイヤが下にあることを示すために、先に接続したワイヤに次の図のような小さな隙間が挿入されます。

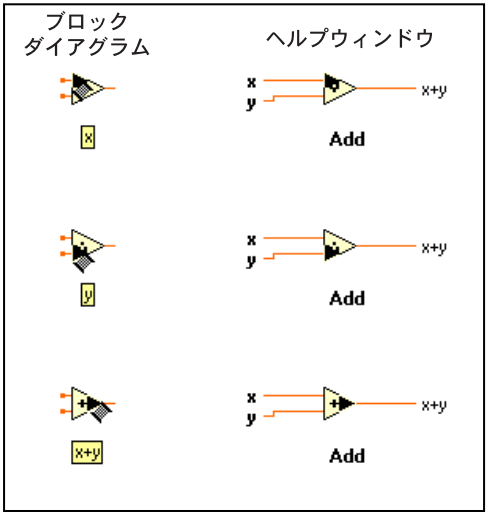


複雑な関数やサブVIの配線に際しては、配線ツールをVIアイコンに近づけたときに表示されるワイヤスタブやヒントラベルに注意することが重要になります。

左記のVIの周りに短く表示されるワイヤスタブは、スタブの形、太さ、色によってそれぞれの端子のデータタイプを示します（詳しくは、『Gクィックリファレンスカード』を参照してください）。スタブの端の点は入力を示します。出力には点がありません。スタブが出ている方向は、わかりやすいなダイアグラムを作成するための望ましい配線の向きを示しています。端子を配線すると、その端子のワイヤスタブは表示されなくなります。

テキストが表示された左記のヒントラベルは、マウスボタンをクリックしたときにワイヤが接続される端子の名前を示します。

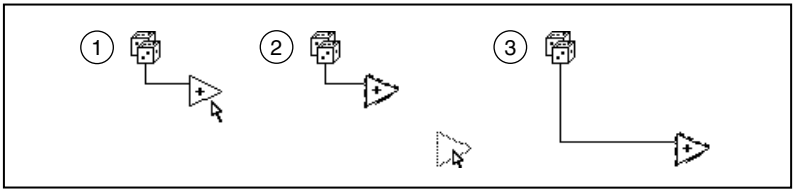
また、各コネクタペーン端子をハイライトで表示するヘルプウィンドウ機能を利用することもできます。この機能を利用すると、ワイヤを接続すべき正確な場所を確認することができます。簡単な例として、Add 関数の3通りの接続を次の図に示します。



☞ **注**　上記のヘルプウィンドウの機能は、拡張が不可能な関数には使用できません。そのような関数の例としては、Array Bundle 関数があります。

ワイヤの延長

配線されたオブジェクトは、選択したオブジェクトを位置決めツールで個別に、あるいはまとめてドラッグすることにより移動できます。選択したオブジェクトに接続されたワイヤは、自動的に延長されます。次の図は、ワイヤの延長機能について示したものです。

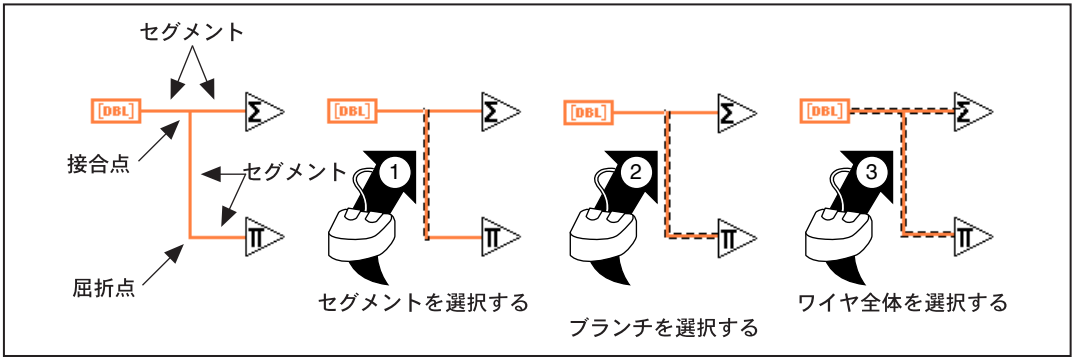


選択したオブジェクトを複製したり、1 つのダイアグラムから別のダイアグラムに（たとえばブロックダイアグラムからストラクチャのサブダイアグラムに）移動する場合、オブジェクトを接続しているワイヤは、選択されていない限り元の場所に残されます。

☞ **注**　**本章の「配線上の問題の解決方法」の項の中の、「未配線の先端」の項で述べるように、ワイヤを延長すると場合によってはワイヤスタブや未配線の先端が生じる場合があります。VI を実行するためには、スタブや未配線の先端は削除しておく必要があります。その最も簡単な方法は、編集→不良ワイヤの削除コマンドを選択する方法です。このコマンドは、ワイヤの不要なループも削除します。**

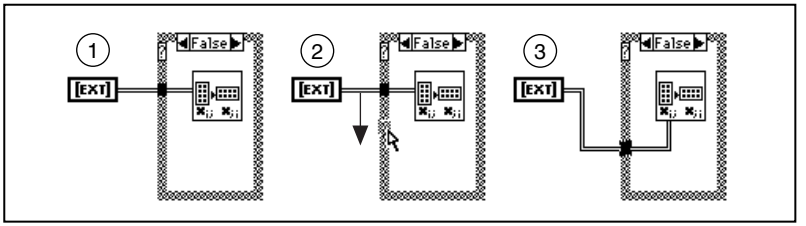
ワイヤの選択、移動、および消去

ワイヤ**セグメント**は、縦または横の1本のワイヤです。3本または4本のワイヤセグメントが結合する点は**接合点**です。**屈折点**は、2本のセグメントが結合する位置です。ワイヤ**ブランチ**には、交点から交点、端子から交点、あるいは間に接合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が含まれます。位置決めツールでワイヤを1回クリックすると、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。<Delete>キーまたは<Backspace>キーを押すと、ワイヤの選択した部分が消去されます。これらの操作を次の図で示します。

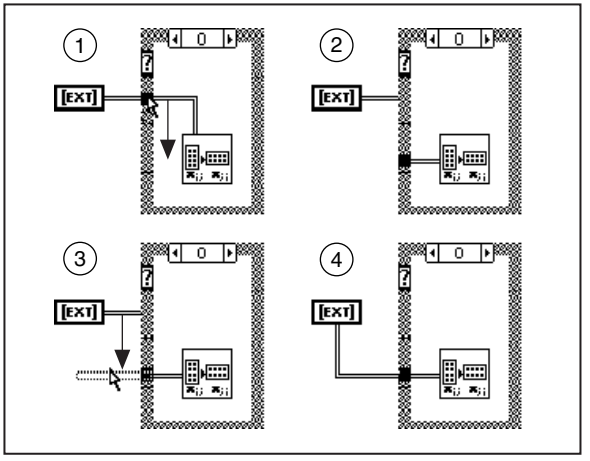


配線したオブジェクトを移動すると、セグメントが余分に生成されたり、誤った場所に配置されたりする場合があります。ワイヤセグメントを移動するには、位置決めツールでドラッグします。1つまたは複数のセグメントを選択し、ドラッグして移動することができます。ワイヤのドラッグの方向を水平または垂直のいずれかに制限するためには、<Shift>キーを使用します。ワイヤを水平または垂直のどちらの方向に変換するかは、最初に移動する方向によって決まります。また、キーボードの矢印キーを押すと、選択したセグメントを1回に1ピクセルずつ移動することができます。選択したセグメントを1回に数ピクセルずつ移動したいときは、<Shift>キーを押しながら矢印キーを押します。隣接する選択されていないセグメントは、変更に応じて自動的に延長されます。一度に複数のセグメントを選択し、ドラッグすることもできます。その場合、これらのセグメントが必ずしも連続したセグメントである必要はありません。

次の図に示すように、トンネルを移動しても通常はトンネルとノードの間の接続は維持されます。



ただし、トンネルを移動するとストラクチャの枠の下に新たなワイヤセグメントが作成される場合があります。次の図に示すように、このセグメントは隠れているために選択したりドラッグしたりすることはできませんが、このセグメントに接続されているセグメントをドラッグすると隠れているセグメントはなくなります。トンネルにどのワイヤが接続されているかはっきりしないときは、ワイヤを3回クリックします。



ループストラクチャの内側と外側でワイヤの一部を同時に選択するためには、まず最初にどちらか一方の側でワイヤの一部を選択し、次に<Shift>キーを押しながらもう一方の側でワイヤの一部を選択します。前に選択したオブジェクトのグループに新たにオブジェクトを追加するには、<Shift>キーを押しながら新たなオブジェクトを選択します。また、ワイヤの両方の部分を囲むようなかたちで選択の四角形をドラッグすることもできます。ストラクチャは、選択の四角形で完全に取り囲まない限り選択されません。その他のノードは、この四角形に接触しているだけで選択されます。

画面に表示されていないオブジェクトへの配線

場合によっては、画面に表示されているオブジェクトと画面に表示されていないオブジェクトを配線する必要があります。たとえば、2 つのオブジェクトが離れすぎていて画面に同時に表示できないような場合です。画面に表示されているオブジェクトと画面に表示されていないオブジェクトを配線するためには、まず最初に画面に表示されているオブジェクトの端子を配線ツールでクリックします。次に、配線ツールをブロックダイアグラムの縁より少し外側までドラッグしてダイアグラムを自動的にスクロールさせ、もう一方のオブジェクトが画面に表示されたら、そのオブジェクトの端子をクリックして配線を完了させます。

配線上の問題の解決方法

不良ワイヤ

ワイヤを誤って接続すると、**壊れたワイヤ**（破線）が表示されることがあります。ときには、壊れたワイヤセグメントが非常に小さかったり、あるいはオブジェクトの陰に隠れていたりして、配線ミスが確認できない場合があります。**実行**ボタンに壊れた矢印（左記参照）が表示されているのにブロックダイアグラム上で問題が見つからない場合は、壊れたワイヤセグメントが隠れている可能性がありますので、**編集→不良ワイヤの削除**を選択します。それによって実行ボタンが正常な矢印（左記参照）に戻れば、問題が除去されたことになります。正常な矢印に戻らない場合は、**壊れた実行**ボタンをクリックしてエラーリストを確認します。

あるワイヤがどうして壊れているかがわからないときは、壊れているワイヤをポップアップし、次に示すポップアップメニューから**エラーリスト**を選択します。**エラーリスト**のダイアログボックスにエラーのリストが表示されたら、いずれかのエラーをクリックします。エラーをクリックするとそのエラーに関係のあるワイヤが選択されるので、そのワイヤを削除または修正します。

詳しくは、「第 4 章　VI とサブ VI の実行およびデバッグ」の「壊れた VI を修正する」および「実行可能な VI をデバッグする」の項を参照してください。

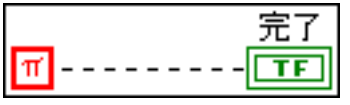
以下に、ワイヤエラーのリストの一部を示します。

- ワイヤ：タイプが競合しています
- ワイヤ：次元数が競合しています
- ワイヤ：要素が競合しています
- ワイヤ：refnumが競合しています
- ワイヤ：クラスが競合しています
- ワイヤ：列挙が競合しています
- ワイヤ：単位が競合しています
- ワイヤ：複数の信号源があります
- ワイヤ：信号源がありません
- ワイヤ：未接続配線があります
- ワイヤ：サイクルを形成しています

ストラクチャの配線についての詳細は、「第19章　ストラクチャ」の「ストラクチャの配線に関する問題」の項を参照してください。

ワイヤタイプの競合

タイプの不一致は、次に示すようにデータタイプの異なる2つのオブジェクト（たとえば数値とブール）を接続した場合に発生します。



次元の競合や要素の競合は、いずれも同じような状況下で発生します。前者は、要素が一致していても次元が一致しない2つの配列を接続した場合に発生し、後者は要素のタイプが異なる2つのクラスタを接続した場合に発生します。

refnumの競合は、種類の異なる2つのrefnumの配線、あるいはレコードタイプの異なる2つのデータログファイルrefnumの配線が原因で発生します。

クラスの競合はクラスの異なる同じ種類の2つのrefnumを配線した場合に発生します。

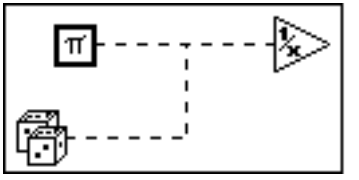
列挙の競合のエラーは、互換性のない2つの列挙を配線した場合に発生します。2つの列挙が互換性を持つのは、両者の文字列がすべて完全に一致する場合、あるいは一方の各文字列がもう一方の各文字列の最初の数文字と完全に一致する場合です。

単位の競合は、互換性のない計量単位を使用している 2 つのオブジェクトを配線した場合に発生します。

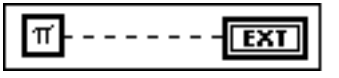
その他のタイプの競合は、いずれも競合エラーになります。これらの問題は一般的に、誤って関数やサブ VI の間違った端子にワイヤを配線した場合に発生します。そのような場合は、ワイヤを選択して削除したうえで正しい端子に接続し直します。この種のエラーは、ヘルプウィンドウを使用することで回避できます。また、エラーによっては、1 つの端子のタイプを変更しなければならなくなる場合もあります。

複数のワイヤソース

1 つのデータソースを複数の接続先に配線することは可能ですが、複数のデータソースを 1 つの配線先に接続することはできません。次に示す例では、ソースを 1 つ切り離す必要があります。

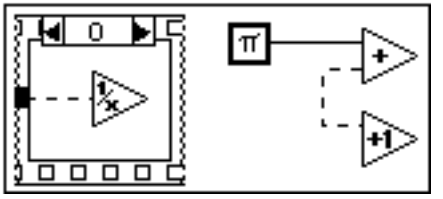


フロントパネルを組み立てる際に、表示器をドロップしようとして制御器をドロップしてしまう場合があります。そのような場合、フロントパネル制御器の端子に出力値を配線しようとすると、複数ソースエラーが表示されます。このエラーを除去するためには、端子をポップアップして**表示器に変更**コマンドを選択します。



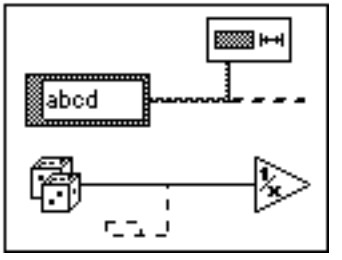
ワイヤソースの不在

次の図に、ソースを持たないワイヤの 2 つの例を示します。その一つは、トンネルから Reciprocal 関数にデータを渡せても、トンネルにデータを渡すソースが存在しないケースです。もう 1 つは、2 つの関数の入力が相互に配線されている一方で、それらの入力のデータソースが存在しないケースです。これらの問題を解決するためにはそれぞれ、トンネルと、Add 関数および Increment 関数にデータソースを配線する必要があります。また、一方が制御器でなければならないときに、2 つのフロントパネル表示器の端子を接続した場合も同じエラーが発生します。このような場合は、一方の端子をポップアップし、**制御器に変更**コマンドを選択します。

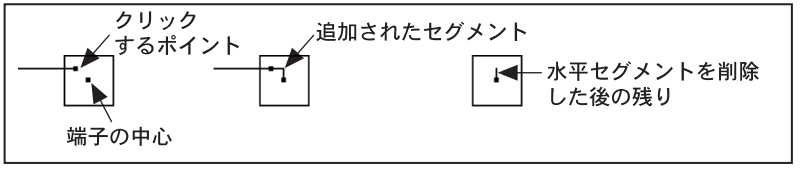


未配線の先端

未配線の先端とは、次の図に示すような端子に接続されていないワイヤブランチです。これらは、ワイヤの延長、配線の取り消し、あるいは配線したオブジェクトの削除によって発生します。未配線の先端を処分するためには、**編集→不良ワイヤの削除**を選択します。



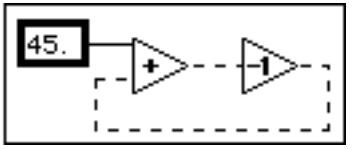
端子に配線する場合、カーソルを正確に端子の中心に合わせてクリックすることはほとんどありません。カーソルが端子の中心からずれていると、クリックした点から端子の中心まで自動的にワイヤセグメントが追加されます。その後、端子へのワイヤを削除すると、未配線の先端が残り、壊れた実行ボタンが表示される原因になる可能性があります。この問題を次の図に示します。



このような事態を回避するためには、かならずワイヤを 3 回クリックして全体を選択してからワイヤを削除します。また、**編集→不良ワイヤの削除**を選択して未配線の先端を削除することもできます。

ワイヤのサイクル接続

ワイヤはサイクルを形成してはなりません。つまり、ワイヤは次に示すような閉鎖ループのアイコンあるいはストラクチャを形成することができません。個々のノードは相手からデータが渡されるのを待って実行されることになるため、実行システムはサイクルを実行することはできません。



繰り返し計算でデータを正しくフィードバックする方法については、「第19章 ストラクチャ」の「シフトレジスタ」の項で説明しています。

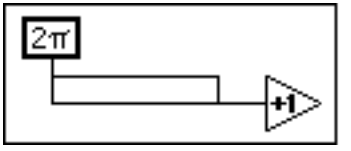
回避すべき配線

以下では、不良ワイヤは生成されなくても、ブロックダイアグラムが読みにくくなったり、実際には実行しないことを実行するような印象を与える状況について説明します。

☞ **注**　**ワイヤに何が接続されているかはっきりしないときは、ワイヤを2回または3回クリックしてブランチまたはワイヤ全体を選択できることを覚えておいてください。**

ワイヤのループ

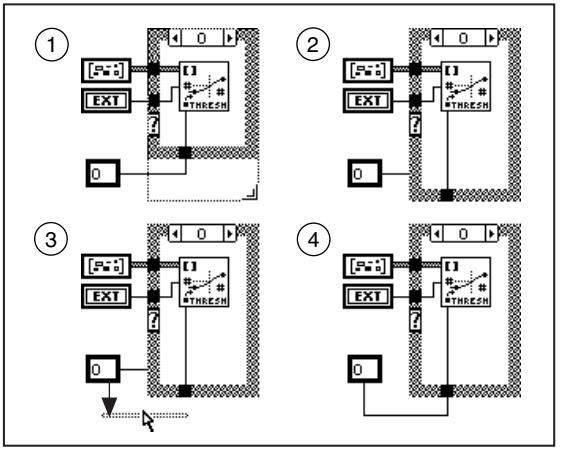
ワイヤのループはエラーではありませんが、ダイアグラムを不必要に混乱させる原因となるため、好ましいデザインとはいえません。いずれかのブランチをダブルクリックしてそのブランチを選択し、削除します。ワイヤのループを次の図に示します。



編集→不良ワイヤの削除を選択すると、これらのループの 1 つのブランチが削除され、配線がすっきりします。

隠れたワイヤセグメント

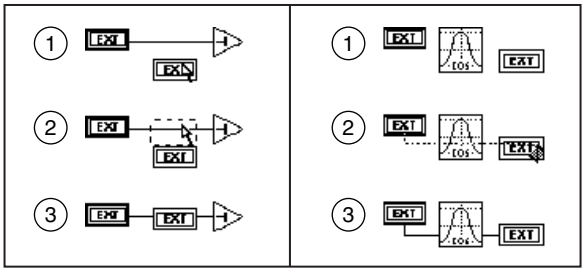
ストラクチャの枠の下やオーバーラップしたオブジェクト間での配線は、一部のワイヤセグメントが隠れてしまうおそれがあるため避けるようにしてください。隠れたワイヤセグメントの例を次の図に示します。



上の図の 1 および 2 で示すように、トンネルを移動したりストラクチャを拡張したりすると、誤って隠れたワイヤセグメントを生成してしまうおそれがあります。上の図の 3 と 4 は、このダイアグラムをよりわかりやすくする方法を示したものです。定数に接続したワイヤセグメントをドラッグすると、隠れているワイヤセグメントを表示することができます。また、<Shift>キーを使用すると、ワイヤのドラッグを制限し、未配線の先端が生成されるのを防ぐことができます。

オブジェクトの下の配線

ワイヤは、クリックしたオブジェクトにだけ接続されます。次の図に示すように、端子やアイコンをワイヤの上にドラッグすると、端子やアイコンには接続されていないのに接続されているように見えてしまいます（左側の図の1、2、3参照）。



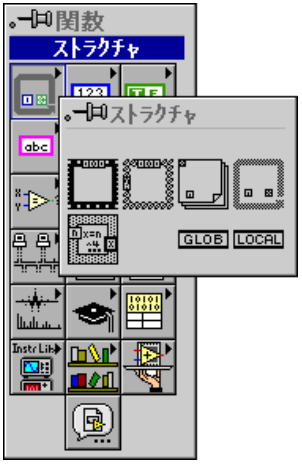
また、図の右側の例で示すように、アイコンや端子を通ってワイヤをドラッグした場合も接続されたように見えますが、実際にはワイヤはアイコンの後ろを通っているにすぎません（右側の図の1、2、3参照）。このような状態は視覚的に錯覚を招くため、避ける必要があります。

この問題についての詳細は、「第4章　VIとサブVIの実行およびデバッグ」の「警告メッセージについて」の項を参照してください。

19

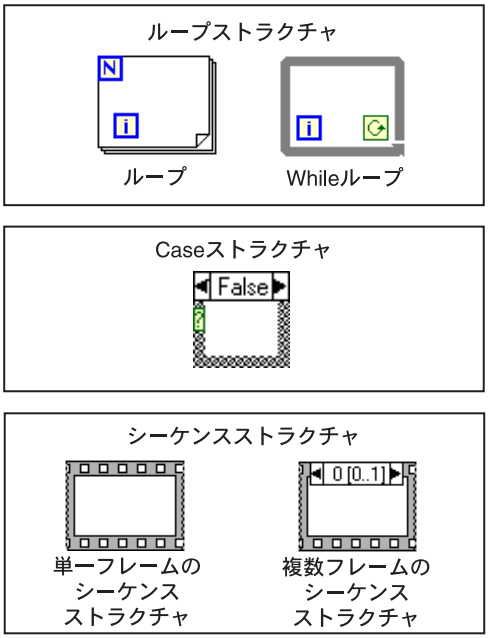
ストラクチャ

この章では、For ループ、While ループ、Case ストラクチャ、およびシーケンスストラクチャの使用方法について説明します。これらのストラクチャは、次の図に示す**関数→ストラクチャ**パレットに入っています。



これらのストラクチャの使用方法については、
`example¥general¥structs.llb` を参照してください。

ストラクチャは、従来のプログラミング言語における制御ストラクチャと同様、ブロックダイアグラムの実行の流れを補足するノードです。次の図に示すように、個々のGストラクチャのアイコンはサイズの変更が可能な長方形で、それぞれ独自の枠を持っています。



ストラクチャは他のノードと同様に、入力データを使用できる場合は自動的に実行し、実行が終了した場合にのみ出力ワイヤにデータを渡します。ただし、個々のストラクチャは、以下で説明するルールに基づいてサブダイアグラムを実行します。

サブダイアグラムは、ストラクチャの枠の内側にあるノード、ワイヤ、および端子の集合体です。For ループと While ループは、それぞれ 1 つのサブダイアグラムを持っています。一方、Case ストラクチャとシーケンスストラクチャは、複数のサブダイアグラムを持つことができますが、トランプの山札と同じように積み重なった状態になっているため、1 度に 1 つのサブダイアグラムしか見ることはできません。サブダイアグラムは、最上位のブロックダイアグラムを作成する場合と同じ方法で作成します。サブダイアグラムには、ブロックダイアグラムの端子、ノード (他のストラクチャを含む)、およびワイヤを格納することができます。

ストラクチャとの間でデータの受け渡しを自動的に行うための端子は、ワイヤがストラクチャの境界線と交差する位置に作成します。このような境界線上の端子は**トンネル**と呼ばれます。トンネルにはかならず、ストラクチャの内側を向いたエッジとストラクチャの外側を向いたエッジが 1 つず

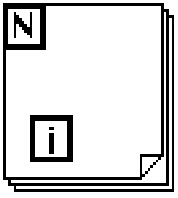
つあります。トンネルは、常にストラクチャの境界線上に位置し、位置決めツールでドラッグすることにより境界線上を移動することができます。

ストラクチャにはこの他に、それぞれのストラクチャに固有の端子もあります。これらの端子については、以下のそれぞれのストラクチャの項で説明します。

Forループストラクチャと Whileループストラクチャ

For ループと While ループは、繰り返しの動作を制御します。For ループは指定された回数だけ動作を繰り返し、While ループは指定された条件が真（TRUE）でなくなるまで動作を繰り返します。

Forループ



繰り返し端子

カウント端子

For ループは、サブダイアグラムを**カウント**回だけ実行します。カウントは、**カウント端子**の値に相当します。回数は、ループの外側からカウント端子の左または上のエッジに値を配線して明示的に設定することもできますが、**自動指標付け**を使用して暗黙的に設定することもできます（詳しくは本章の「自動指標付け」の項参照）。カウント端子の残りのエッジは、カウントに内側からアクセスできるようにループの内側を向いています。

繰り返し端子には、すでに終了した繰り返しの回数が格納されます。1 回目の実行中は値は 0、2 回目の実行中は 1 となり（以下同様）、最終回の実行中は *N–1* となります。**カウント端子と繰り返し端子はいずれも符号付きの倍長整数で、値の範囲は 0 ～ 2^{31}–1 です。**カウント端子に浮動小数点数を配線すると、G は値を丸め、必要な場合は強制的に範囲内の値に設定します。カウント端子に 0 を配線した場合、ループは実行されません。

For ループは、下記の疑似コードと等価です。

```
for i = 0 to N - 1
  Execute subdiagram
```

Whileループ


条件端子


繰り返し端子

Whileループは、**条件端子**に配線したブールがFALSEになるまでサブダイアグラムを実行します。Gは、各回の実行が終了するたびに条件端子の値をチェックし、値がTRUEであれば再度実行します。したがって、ループは少なくとも1回は実行されることになります。条件端子のデフォルト値はFALSEであるため、この端子が配線されていない場合はループは1回しか実行されません。

繰り返し端子には、Forループの場合と同様、すでに実行した回数が格納されます。

Whileループは、下記の疑似コードと等価です。

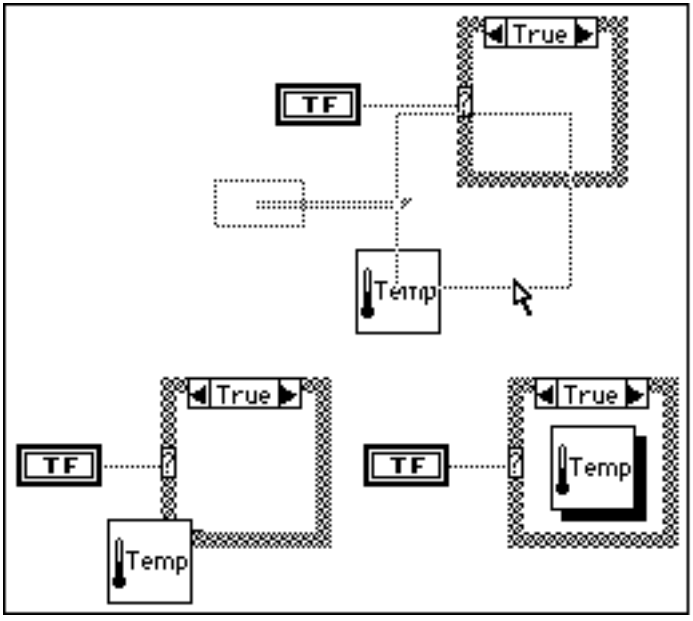
```
Do
    Execute subdiagram (which sets condition)
While condition is TRUE
```

どちらのループストラクチャにも、**シフトレジスタ**と呼ばれる端子を追加することができます。シフトレジスタは、現在の実行から次の回の実行にデータを渡すために使用します。詳しくは、本章の「シフトレジスタ」の項を参照してください。

ストラクチャ内にオブジェクトを配置する

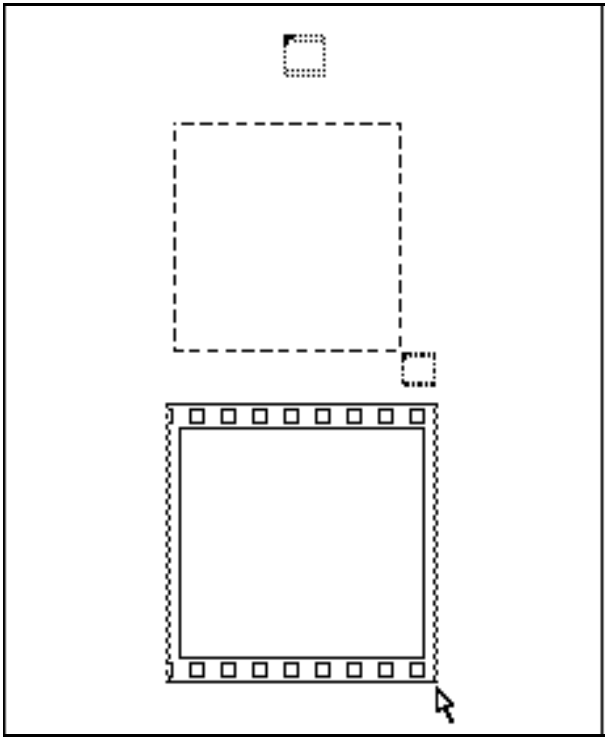
オブジェクトをストラクチャの内側にドラッグするか、またはオブジェクトを囲むストラクチャを作成することにより、ストラクチャ内にオブジェクトを配置することができます。

一方、オブジェクトの上にストラクチャをドラッグしても、ストラクチャ内にオブジェクトを配置することはできません。ストラクチャを移動してそのストラクチャが別のオブジェクトと重なった場合、エディタは前面にあると判断したオブジェクトを表示します。別のオブジェクトを完全に覆うようにしてストラクチャを配置したときに、エディタがそのオブジェクトが最前列のオブジェクトであると判断した場合、そのストラクチャがオブジェクトの内部ではなく上にあることを示すための濃い影が表示されます。それ以外の場合は、オブジェクトはストラクチャによって完全に隠されます。アプリケーションがどのオブジェクトを最前列のオブジェクトとして選択するかについては、「第2章 VIを編集する」の「オブジェクトを配置する」の項を参照してください。両方のケースを次の図に示します。



ブロックダイアグラム上へのストラクチャの配置とサイズの変更

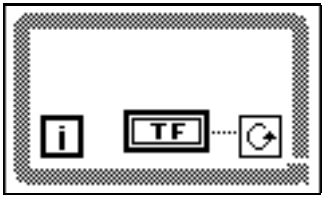
関数→ストラクチャでストラクチャを選択し、ダイアグラム上に配置できる状態になると、カーソルの代わりに小さいストラクチャアイコンが表示されます。このアイコンは、ブロックダイアグラム上に配置することができます。マウスボタンをクリックし、サイズを変更せずにボタンを放した場合は、再度クリックするまでサイズ変更モードのままになります。ダイアグラムをクリックし、ドラッグすると、ストラクチャを任意のサイズに変更することができます。その例を次の図に示します。また、ストラクチャをダイアグラムに配置した**後**でも、サイズ変更カーソルでいずれかの角をクリックし、ドラッグすることによってサイズを変更することができます。



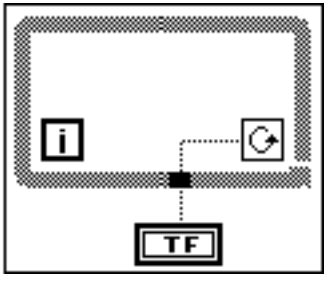
ループ内に端子を配置する

ループの実行を開始する際に、入力によりループにデータが渡されます。出力は、ループの繰り返し実行がすべて終了した場合にのみループからデータを渡します。

毎回の繰り返し実行の際にそのつどループに端子の値をチェックさせたいときは、ループの**内部に**端子を配置する必要があります。たとえば、フロントパネルのブール制御器の端子を While ループの内部に配置し、その端子をループの条件端子に配線すると、ループは繰り返し実行の**たびに**条件端子の値をチェックし、再度実行すべきかどうかを判断します。この While ループは、次の図に示すようにフロントパネルの制御器の値を FALSE に変更することにより、停止させることができます。



ブール制御器の端子を次の図のように While ループの外側に配置した場合は、ループの開始時にブール制御器が TRUE になっていると、ループは無限に繰り返されます。

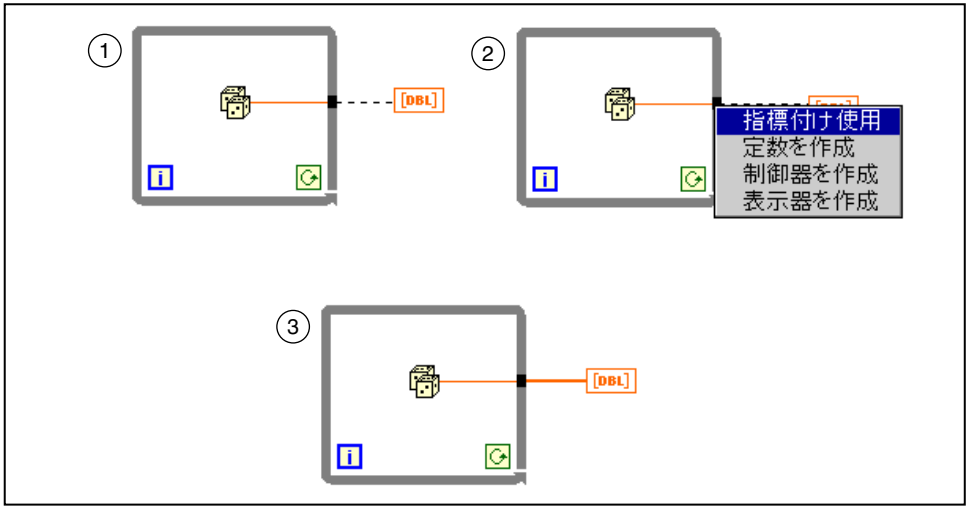


ループの開始時に制御器が TRUE になっていると、フロントパネルの制御器の値を FALSE に変更してもこのループの実行は停止しません。これは、ループが停止し、再度 VI を実行するまでは値が伝送されないためです。誤って無限ループを作成してしまったときは、VI を中断することでループを停止させることができます。実行を中断するには実行停止ボタンをクリックします。

自動指標付け

For ループと While ループのストラクチャは、自動的に配列に指標を付け、その配列を境界に蓄積することができます。これらの機能を総称して**自動指標付け**と呼びます。外部のソースの配列をループの境界上の入力トンネルに配線し、入力トンネルに対して自動指標付けを使用に設定しておくと、配列のコンポーネントは最初のコンポーネントから順に 1 回に 1 つずつループに渡されます。ループは、スカラ要素を 1 次元配列から、1 次元配列を 2 次元配列からというように指標付けします。出力トンネルではこれと逆の動作が発生し、スカラ要素は 1 次元配列に、1 次元配列は 2 次元配列にというように累積されます。

次の図は、自動指標付けを使用したループと使用していないループのトンネルの形を示したものです。ループの境界上でワイヤの次元が変わると、ワイヤは太くなります。



For ループは、配列を順番に処理するのによく使用されます。そのため、配列を For ループに（あるいは For ループから）配線した場合の配列のトンネルの自動指標付けは、デフォルトで使用に設定されます。While ループはこのような目的に使用されることは少ないため、自動指標付けはデフォルトで不使用に設定されています。ループの境界上にあるトンネルの自動指標付けを使用または不使用にするためには、ループの境界上のトンネルをポップアップして**指標付け使用**または**指標付け不使用**を選択する必要があります。

自動指標付けを使用してForループのカウントを設定する

Forループに入る配列に対して自動指標付けを開始すると、カウント値（繰り返し回数）は自動的に配列のサイズに設定されるため、カウント端子に明示的に配線する手間が省けます。ただし、自動指標付けを複数のトンネルに対して開始した場合、あるいはカウント値も明示的に設定した場合は、カウント値は選択した値のうちの最小値になります。たとえば、ループに入る配列が2つあり、コンポーネントの数はそれぞれ10個と20個で、どちらも自動指標付けが使用になっているものとします。この場合、カウント端子に15という値を配線したとしても、ループの実行回数は10回となり、2つめの配列では最初の10個のコンポーネントしか自動指標付けされません。

自動指標付けの出力配列は、ループの各回の実行ごとに新しい出力を受け取ります。したがって、自動指標付けされた出力配列のサイズは、常に繰り返しの回数と同じになります（上の例では10個）。

出力トンネルの自動指標付けが不使用になっていると、ループの最後の回の要素値だけが渡されます。

Whileループで自動指標付けを使用する

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ループと同じ方法で配列に指標を付けます。ただし、Whileループは特定の条件がTRUEである間は繰り返し実行されるため、Whileループの実行回数は配列のサイズによって制限されることはありません。Whileループの指標が入力配列の要素の数を超えると、配列の要素タイプのデフォルト値がループに渡されます。自動指標付けされた出力配列は、Whileループを実行するたびに出力された要素を受け取ります。出力配列のサイズは、Whileループの実行が繰り返される限り増え続けます。

☞ **注**　**繰り返し回数の多いWhileループで自動指標付けを使用すると、メモリが足りなくなる場合があります。Whileループで配列を作成する際にこのような問題を避けるためには、『LabVIEWユーザマニュアル』の「第5章　配列、クラスタ、およびグラフ」を参照してください（LabVIEWをご使用の場合）。**

For ループを 0 回実行する

カウントを 0 に設定すると、For ループはそのダイアグラムをまったく実行しません。For ループから出力されるすべてのスカラデータには、下記のルールが適用されます。

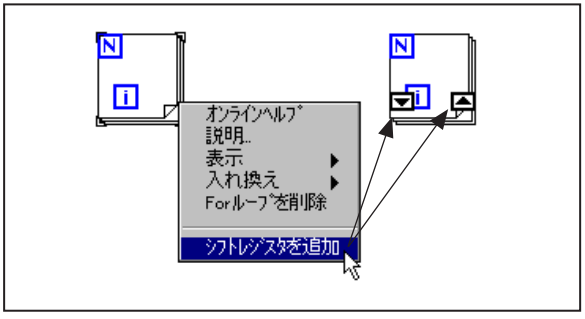
- 出力トンネルで自動指標付けによって作成される出力配列は空になります。
- 初期化されたシフトレジスタからの出力は、初期値になります。詳しくは、本章の「シフトレジスタ」の項を参照してください。
- 指標付けを行わない出力トンネルを通して渡される配列も空になります。
- 指標付けを行わない出力トンネルを通して渡されるすべてのスカラは未定義であるため、これらの値に依存することはできません。

ループのカウントを 0 に設定する、あるいはデフォルトで 0 に設定する方法は 2 通りあります。空の入力配列を自動指標付けするか、あるいは 0 またはマイナスの数字をカウント端子に明示的に配線します。

入力配列を自動指標付けする場合、あるいは変数を使用してループのカウントを設定する場合は、ダイアグラムを分析して 0 カウントが発生するかどうかをチェックし、0 カウントが発生する場合は、ダイアグラムが空の配列を正しく処理できるようにする必要があります。

シフトレジスタ

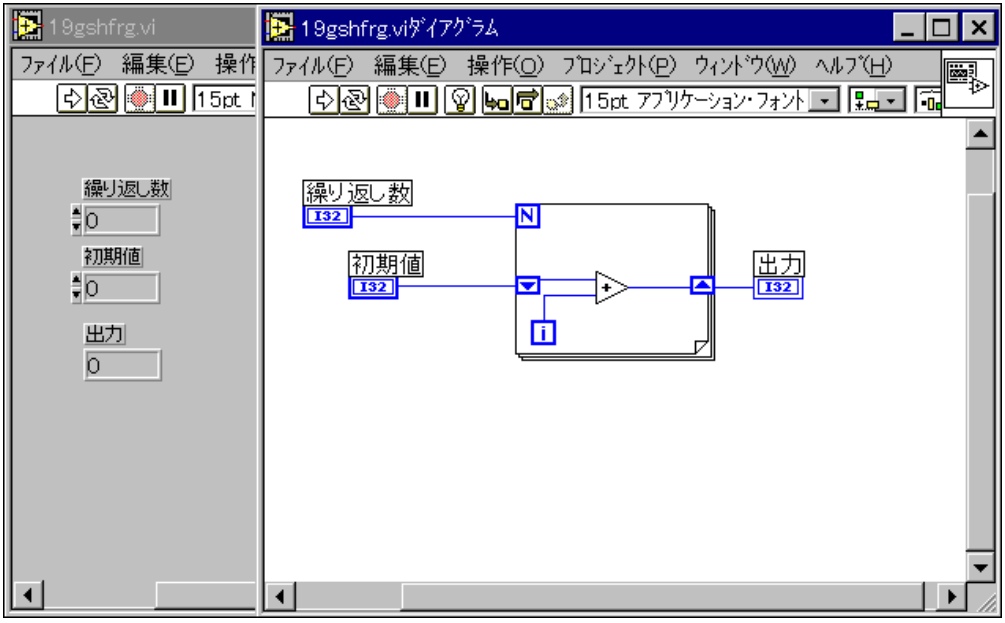
シフトレジスタは、For ループと While ループで使用される変数で、ループの 1 回の実行を完了して次の回に移る際にデータをフィードバック、すなわち転送します。次の図に示すように、ループの境界線のポップアップメニューから**シフトレジスタを追加**を選択すると、縦の境界線上の任意の場所にレジスタを作成することができます。このメニュー項目は、ストラクチャの上のエッジや下のエッジでは使用できません。シフトレジスタは、縦の境界線上に沿ってドラッグすることにより移動できます。



シフトレジスタには、ループの縦の 2 つの境界線上で互いに向かい合う 1 組の端子があります。**右側の端子**、すなわち上向きの矢印を持つボックスは、ループの各回の終了時の値を記憶します。このデータは各回の終了時にシフトされ、次の回の開始時に**左側の端子**、すなわち下向きの矢印を持つボックスに表示されます。シフトレジスタはどのタイプのデータにも使用できますが、各レジスタの端子に配線するデータは同じタイプでなければなりません。1 つのストラクチャ上に複数のシフトレジスタを作成することができます。

シフトレジスタを初期化するためには、ループの外側から左の端子に値を配線します。レジスタを初期化しなかった場合は、ループは最後に実行したときにレジスタに書き込まれた値を使用し、まだ一度もループを実行していない場合はそれぞれのデータタイプのデフォルト値を使用します。動作に一貫性を持たせるには、初期化されたシフトレジスタを使用します。初期化されていないシフトレジスタの使用方法については、LabVIEW をご使用の場合は『LabVIEW ユーザマニュアル』の「第 3 章　ループとチャート」を、BridgeVIEW をご使用の場合は『BridgeVIEW User Manual』の「Chapter 10 Loops and Charts」を参照してください。

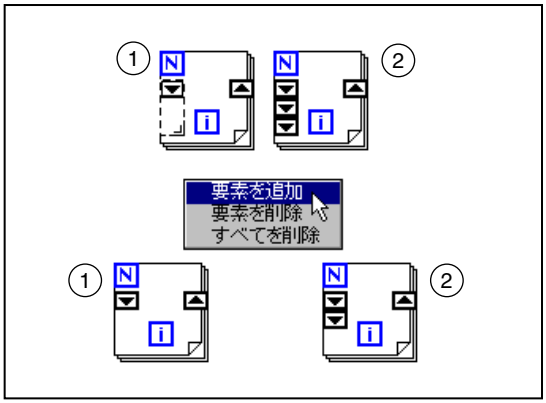
ループの実行が終了すると、シフトレジスタに記憶された最後の値が右側の端子に残されます。右側の端子がループの外に配線されているときは、ループの実行終了時にこの最後の値が渡されます。この特性を次の図に示します。



シフトレジスタは左側に複数の端子を持つことができるため、前の値を複数記憶することができます。


サイズ変更カーソル

特定のシフトレジスタの端子を追加または削除するためには、サイズ変更カーソルを使用して左側の端子のサイズを変更します。あるいは、シフトレジスタのポップアップメニューから**要素を追加**コマンドを選択して、シフトレジスタの左側の端子を追加することもできます。追加された端子は、ポップアップした端子のすぐ下に表示されます。ポップアップした端子を削除したいときは、**要素を削除**コマンドを使用します。両方の方法を次の図に示します。余分に配線した端子を削除するには、シフトレジスタのポップアップメニューから**要素を削除**を選択するのが唯一の方法です。**すべてを削除**を選択すると、シフトレジスタそのものが削除されます。

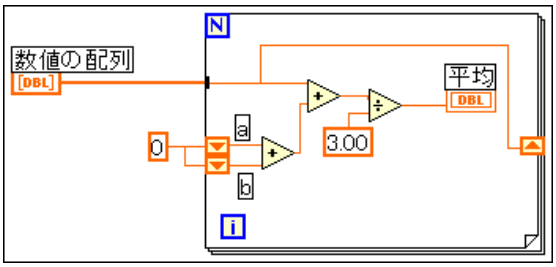


左のいちばん上の端子には、前の回（*i* –1 の回）の値が格納されます。そのすぐ下の端子には、それよりもさらに 1 つ前の回（*i* –2 の回）の値が格納されます（それ以降の端子も同様）。

シフトレジスタの左の端子の 1 つを初期化するためには、左の端子をすべて初期化する必要があります。

次の疑似コードは、G のブロックダイアグラムと等価の 3 つの値を使用した移動平均ルーチンを示します。

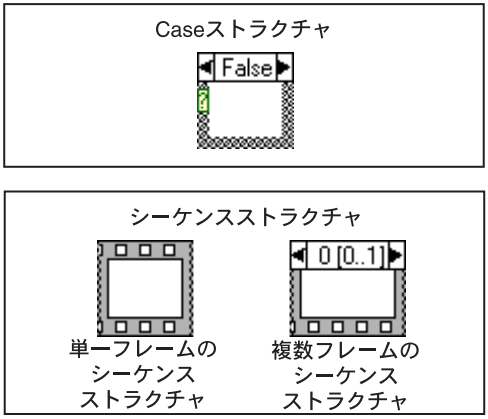
```
a = b = 0
for i = 0 to N - 1
    avg = (x[i] + a + b)/3
    b = a
    a = x[i]
```



シフトレジスタの設定例については、
examples¥general¥structs.llb¥Random Average.vi を参照してください。

Caseストラクチャとシーケンスストラクチャ

Caseストラクチャとシーケンスストラクチャは、いずれも複数のサブダイアグラムを持つことができますが、1度に1つしか見ることができません。各ストラクチャの境界線の上には**サブダイアグラム表示ウィンドウ**があり、ウィンドウの中央には**ダイアグラム識別子**が、両側には減分ボタンと増分ボタンがあります。Caseストラクチャの場合は、ダイアグラム識別子は現在表示されているサブダイアグラムを実行する場合の値を表示します。シーケンスストラクチャの場合は、ダイアグラム識別子は現在どのサブダイアグラムが表示されているかを示します。ダイアグラム識別子が数値のときは、その後にストラクチャに入っているサブダイアグラムの数の最小値と最大値を示すダイアグラム識別子範囲が続きます。



減分ボタン（左）をクリックすると前のサブダイアグラムが表示され、増分ボタン（右）をクリックすると次のサブダイアグラムが表示されます。最初のサブダイアグラムで減分ボタンをクリックすると最後のサブダイアグラムが表示され、最後のサブダイアグラムで増分ボタンをクリックすると最初のサブダイアグラムが表示されます。表示ウィンドウのその他の使用方法については、本章の「Caseストラクチャとシーケンスストラクチャ」の項で説明します。

Caseストラクチャのサンプルについては、`examples¥general¥structs.llb¥SquareRoot.vi` を参照してください。

Caseストラクチャ

Caseストラクチャは、1つのサブダイアグラム、すなわちケースを選択し、選択子と呼ばれる入力値に基づいて実行します。Caseストラクチャを使用すると、次のことが可能になります。

- ケースに対応する選択子の値のリストおよび範囲を指定できます。
- 整数、ブール、文字列、または列挙タイプを選択子として使用できます。
- デフォルトのケース（または動作）を指定できます。
- 最初の選択子の値に基づいてケースをソートします。

ブロックダイアグラムにCaseストラクチャを配置する場合は、Caseストラクチャの選択子ラベルに選択子の値を直接入力します。また、ラベリングツールを使用して選択子の値を編集することもできます。値は、1つだけ指定することも、ケースを選択する値のリストや範囲を指定することもできます。リストを指定する場合は、–1,0,5,10というようにそれぞれの値をコンマで区切ります。10から20までの範囲を指定する場合は、10..20というかたちで指定します（この場合10と20は範囲に含まれます）。また、..0（0以下のあらゆる数字）や100..（100以上のあらゆる数字）のように、一方の側が開放の範囲も使用できます。リストと範囲は、..6,7..10,12,13,14というように組み合わせることもできます。重複した範囲を含む選択子を入力した場合は、Caseストラクチャはより簡潔なかたちで選択子を表示し直します。上に示した例は、..10,12..14と表示されます。

また、ケース選択子には文字列や列挙の値を使用することもできます。これらの値は、[**"red"**]、[**"green"**]、[**"blue"**]というように必ずクォーテーションマークで囲って表示されます。ただし、コンマや記号[**".."**]を含まない文字列や列挙を入力する場合は、クォーテーションマークを入力する必要はありません。

注　選択子アイコンに配線したオブジェクトと異なるタイプの選択子値を入力した場合は、値が赤で表示され、VI を実行できなくなります。また、浮動小数点計算にともなうラウンドオフエラーの発生を避けるため、LabVIEW ではケースの選択子ラベルに浮動小数点数を使用できないようになっています。ケースに浮動小数点タイプを配線した場合は、旧バージョンの LabVIEW と同様、タイプは最も近い整数に丸められます。Case 選択子に 1.2 というように浮動小数点数を入力すると、値は赤で表示されます。

次に示す Case は数値タイプであるため、文字列のセレクタ値は赤で表示されます。

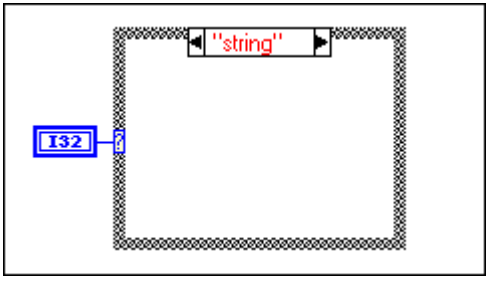


図 19-1 Case ストラクチャのタイプの不一致

さらにケースを追加する場合は、本章の「サブダイアグラムを追加する」の項に記載した手順に従ってください。

注　通常、他の言語の Case 文はケース値が範囲外の値であるときはケースを実行しません。G では、範囲外の値を処理するデフォルトのケースを含めるか、または使用可能なすべての入力値のリストを指定する必要があります。

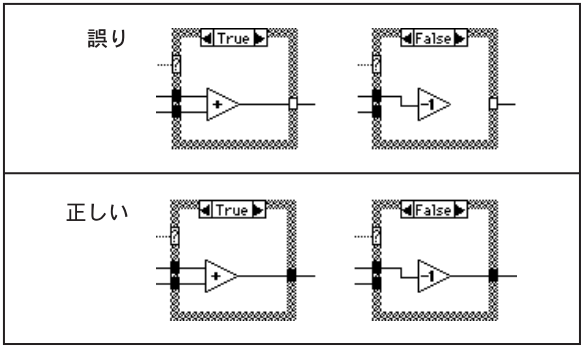
選択子は、左の境界線上の任意の場所に配置することができますが、必ず選択子を配線する必要があります。選択子はデータタイプに合わせて自動的に調節されます。選択子に配線した値を数値からブールに変更すると、ケース 0 とケース 1 は FALSE と TRUE に変更されます。ほかにケースが存在する場合（2 ～ n）でも、それらのケースは破棄されません。これは、データタイプの変更が不測のものであった場合に備えるためです。ただし、これらの余分なケースはストラクチャを実行する前に削除する必要があります。

このことは、列挙型を選択子に配線したときに列挙項目より多くのケースが存在する場合にもあてはまります。このようなケースのダイアグラム識別子は、これらのケースを削除しなければストラクチャを実行できないことを示すために、赤の数値識別子として表示されます。

ケース選択子の値を別のタイプに変換する際には、次の点に注意してください。たとえば23という数値を文字列に変換する場合は、文字列の値は23になります。文字列を数値に変換する場合は、数字を表す選択子の文字列だけが数値に変換されます。その他の値は文字列のままになります。数値をブールに変換する場合は、0はFalseに、また1はTrueに変換され、他の値は文字列になります。

すべての入力端子（トンネルおよび選択端子）のデータは、すべてのケースで使用できます。ケースは、入力データを使用したり出力データを渡したりする必要はありませんが、いずれかのケースが出力データを渡す場合は、**すべてのケースが出力データを渡す必要があります。**次の図の上の例のように、すべてのケースから出力トンネルにデータが配線されていない場合は、トンネルは白で表示され、**実行**ボタンに壊れた矢印が表示されます。

次の図の次の例のように、すべてのケースからトンネルにデータが渡される場合は、トンネルは黒で表示され、実行ボタンは壊れていない状態で表示されます。



Caseストラクチャのポップアップメニューには、次の図に示すようにブール以外のケースのための項目も含まれています。

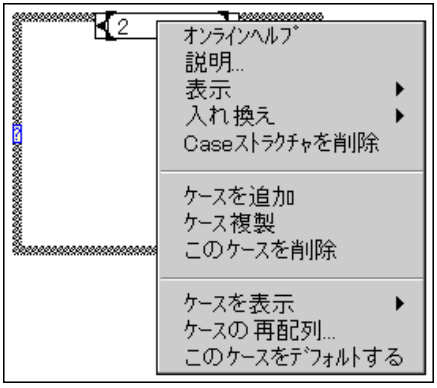


図 19-2 Caseストラクチャのポップアップメニュー項目

ケースを追加オプションは、表示されているケースの後ろにケースを追加します。表示されているケースの前にケースを追加することはできません。ケースの順序は、**ケースの再配列...**を選択することによって変更できます。**ケースの再配列...**を選択すると、次のようなダイアログボックスが表示されます。

図 19-3 ケースの再配列ダイアログボックス

ソートボタンは、最初の選択子の値に基づいて選択子の値をソートします。選択子の位置を変更する場合は、移動したい選択子の値をクリックし、移動したい場所までドラッグします。**完全な選択子文字列**は、選択したケース選択子が長すぎて**ケースリスト**に収まりきらない場合にケース選択子全体を表示します。このダイアログボックスでは、どの制御器に関しても**ヘルプ**ボタンをクリックして文脈対応のヘルプを呼び出すことができます。

☞ **注**　**この機能では、Case ストラクチャに表示されるケースの順序だけが変更されます。Case ストラクチャの実行時の動作には影響を与えません。**

ポップアップメニューの**このケースをデフォルトにチェックする**という項目は、選択子の値がCase ストラクチャに登録されていないときに実行する特定のケースを指定します。Case ストラクチャの上の選択子の値の中には、Default という単語が表示されます。Case ストラクチャに考えられるすべての選択子の値に対するセレクタが含まれていない場合は、Case ストラクチャに対してデフォルトのケースを指定する必要があります。

Case のサブダイアグラムの追加、移動、および削除は、Case ストラクチャでもシーケンスストラクチャでも同じであるため、以下のシーケンスストラクチャに関する項の次に説明します。

シーケンスストラクチャ

シーケンスストラクチャはフィルムのフレームのような形をしており、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**で構成されます。シーケンスストラクチャを使用したVIのサンプルについては、`examples¥general¥structs.llb¥TimingTemplate.vi` を参照してください。

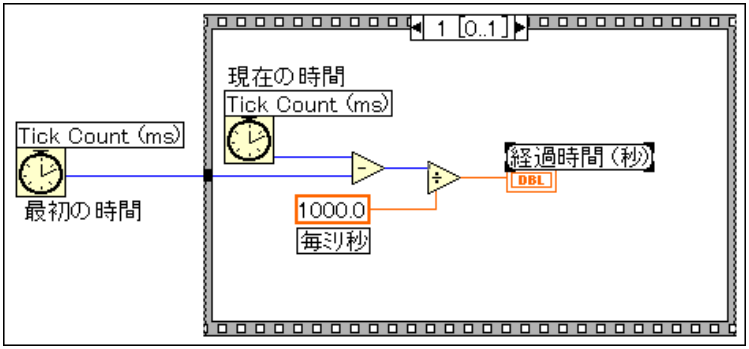
プログラムの個々の要素を順番に並べることによってプログラムの実行順序を決定することを、**制御フロー**といいます。BASIC、C、およびその他のほとんどのプログラミング言語は、文はプログラム中で現れる順番に実行されるため、これらの言語には独自の制御フローがあります。シーケンスストラクチャは、データの依存性が十分でないときに制御フローを確保するための手段です。シーケンスストラクチャは、フレーム 0、フレーム 1、フレーム 2 という順に最後のフレームまで実行します。データは、最後のフレームの実行が終了してはじめてストラクチャから外に渡されます。

個々のフレーム内では、ブロックダイアグラムの他の部分と同様、ノードの実行順序はデータの依存性によって決定されます。

シーケンスストラクチャは、データの依存性のないノードの実行順序を制御するために使用します。他のノードから直接または間接的にデータを受け取るノードは、他方のノードに対してデータの依存性があるため、必ず他方のノードの実行が終了してから実行されます。データの依存性が存在

する場合や、実行順序が重要でない場合は、シーケンスストラクチャを使用する必要はありません。

このような状況は、関数が使用する時間を決定する場合によく発生します。Tick Count 関数は、現在の時間をミリ秒で返します。この時間は関数を実行するためには重要でないため、時間の測定と関数との間にはデータの依存性は存在しません。次の図に示すように、シーケンスストラクチャの左にある Tick Count 関数は、シーケンスストラクチャの前に実行されます。次に、関数はフレーム 0 を実行し、フレーム 1 で経過時間の計算が行われます。シーケンスストラクチャは、正しい実行順序を強制します（フレーム 0 は表示されていません）。



シーケンスストラクチャの出力トンネルは、Case ストラクチャとは異なり、データソースを 1 つしか持つことができません。出力はどのフレームからも渡すことができますが、データがストラクチャから外に出るのは個々のフレームの実行が終了したときではなく、ストラクチャ全体の実行が終了したときである点に注意してください。入力トンネルのデータは、Case ストラクチャの場合と同様すべてのフレームで使用できます。

1 つのフレームからそれより後のフレームにデータを渡すためには、**シーケンスローカル**と呼ばれる端子を使用します。シーケンスローカルを獲得するためには、ストラクチャの境界線のポップアップメニューから**シーケンスローカルを追加**を選択します。この項目は、シーケンスローカルやサブダイアグラムの表示ウィンドウ上でポップアップした場合は使用できません。端子は、空いている場所であれば境界線上のどこにでもドラッグすることができます。端子を削除する場合は、シーケンスローカルのポップアップメニューの**削除**コマンドを使用します。

データソースを持つフレームのシーケンスローカル端子には、外側を向いた矢印が表示されます。それより後のフレームの端子には、端子がそのフレームのソースであることを示す内側を向いた矢印が表示されます。ソースフレームより前のフレームでは、シーケンスローカルは使用できず、グレーの四角形で表示されます。次の図にシーケンスローカルの端子を示します。

Caseストラクチャとシーケンスストラクチャを編集する

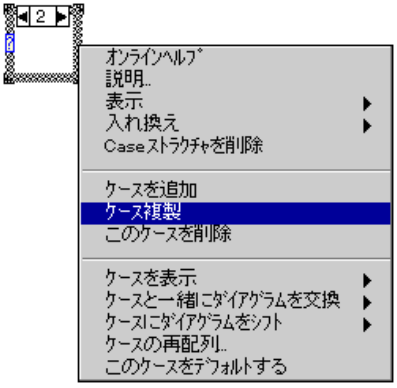
Case ストラクチャとシーケンスストラクチャでは編集や操作の方法が同じであるため、以下の例では Case ストラクチャとそのポップアップメニューだけを示します。シーケンスストラクチャについては、ケースという言葉を**フレーム**という言葉に差し替えてお読みください。新しい Case ストラクチャには 2 つのケースがありますが、1 つのケースに編集することもできます。新しいシーケンスストラクチャのフレームは 1 つです。

サブダイアグラム間を移動する

1 つ上または 1 つ下のサブダイアグラムを見る最も簡単な方法は、表示ウィンドウの減分ボタンまたは増分ボタンを使用する方法です。いくつかのサブダイアグラムを飛び越えて別のサブダイアグラムに移動したいときは、次の図に示すように、サブダイアグラムの識別子をクリックし、ポップアップメニューから見たいサブダイアグラムを選択します。あるいは、境界線のポップアップメニューの**ケースを表示**コマンドを使用することもできます。

サブダイアグラムを追加する

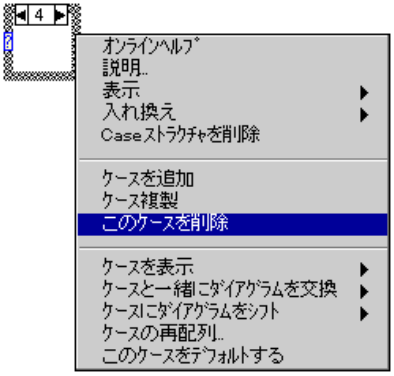
次に示すように、ストラクチャの境界線のポップアップメニューまたはダイアグラムの識別子のポップアップメニューから**ケース複製**を選択すると、現在表示されているサブダイアグラムの後ろにそのコピーが追加されます。



シーケンスストラクチャにサブダイアグラムを追加または削除した場合は、エディタはそれに合わせてダイアグラム識別子を自動的に調節します。Caseストラクチャの場合には、サブダイアグラムの順番はプログラムの実行には無関係ですが、ポップアップメニューの**ケースの再配列**...項目を使用して変更することができます。

サブダイアグラムを削除する

表示されているサブダイアグラムを削除する場合は、次に示すようにストラクチャの境界線のポップアップメニューから**ケースを削除**を選択します。このコマンドは、サブダイアグラムが 1 つしか存在しない場合には使用できません。

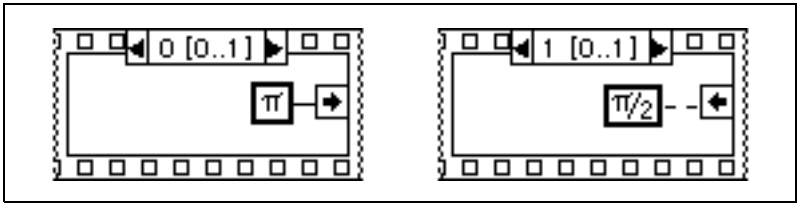


ストラクチャの配線に関する問題

以下の項では、ストラクチャの間違った接続について説明します。

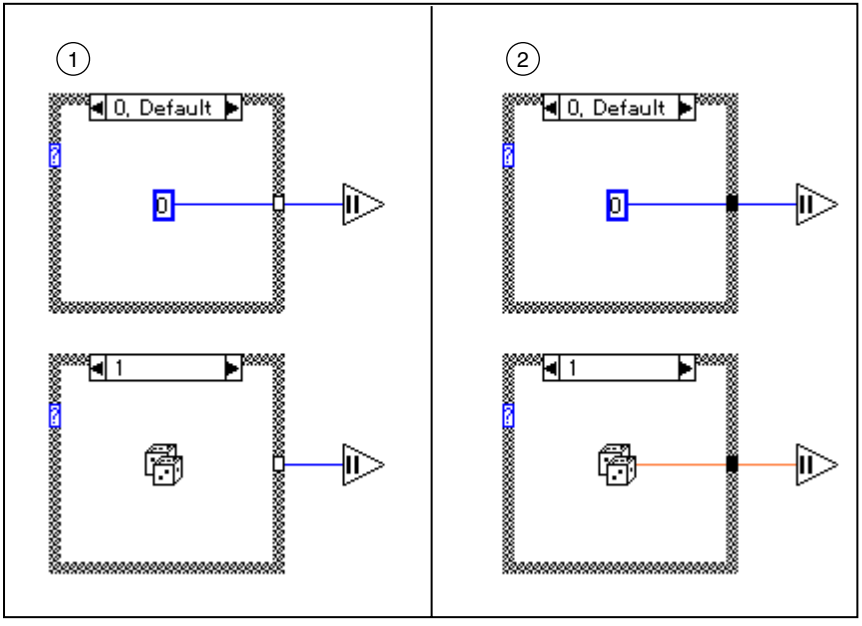
シーケンスローカルに複数の値を代入する

シーケンスストラクチャのローカル変数に対しては、1 つのフレームでしか値を代入することはできませんが、その値はそれ以降のフレームで使用することができます。次の左の図は、フレーム 0 のシーケンスローカルに値 π が代入されていることを示しています。フレーム 1 でこの同じローカル変数に別の値を代入すると、不良ワイヤが表示されます。このエラーは、複数信号源エラーの一つです。



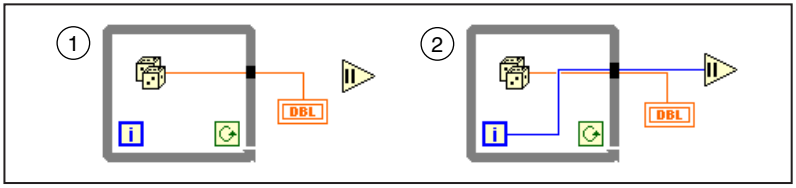
Caseストラクチャの一部のケースのトンネルへの配線もれ

Case ストラクチャからストラクチャの外のオブジェクトに配線する場合、次の 1 の例で示すように、そのオブジェクトにすべてのケースのソースを接続しないと不良トンネルになります。この状態で実行すると、少なくとも 1 つのケースはデータ値を生成しないことになるため、このエラーは一種の信号源のないエラーです。この場合は、次の 2 の例で示すように、すべてのケースをトンネルに配線して問題を取り除いてください。これは、1 つのケースだけが実行され、Case ストラクチャを実行するたびに出力値が1つしか生成されないため、複数信号源のルール違反にはなりません。

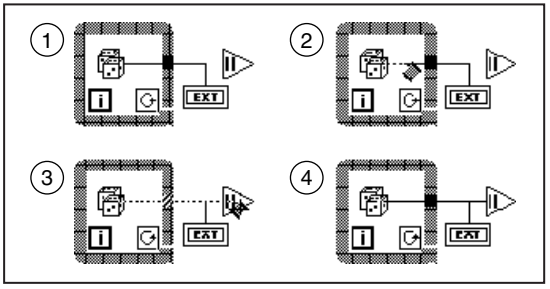


トンネルを重ね合わせる

G では配線によってトンネルが作成されるため、ときにはトンネルが重なってしまう場合があります。トンネルが重なってもダイアグラムの実行には影響はありませんが、編集作業が面倒になるため、トンネルは重ねないようにしてください。トンネルの重なりが発生した場合は、一方のトンネルをドラッグしてもう一方も見えるようにしてください。次の例を参照してください。



どちらのトンネルが上にあるかを見分けるのは容易ではありません。重なっているトンネルの一方に配線しようとしてミスを犯す場合がありますが、いつでも不良ワイヤを削除して配線し直すことができます。

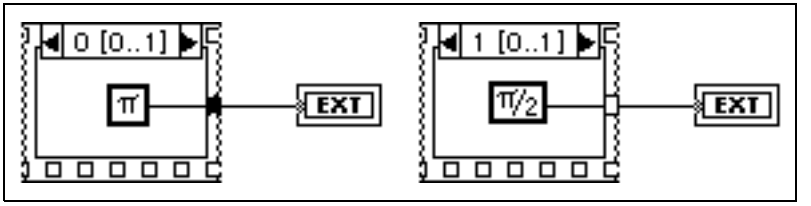


ストラクチャの中のオブジェクトから外のオブジェクトに配線しようとしたときに、すでにそのようなワイヤが存在する場合は、前図に示すように再度ストラクチャの内側から**配線することは避け**、2 番目のワイヤはトンネルを起点にして配線するようにしてください。この例では、2 つのトンネルが重なっていても問題は起こりません。しかし、これがCase ストラクチャである場合は、重なり合った 2 つの不良トンネルがそれぞれのケースに配線されているように見えることがあります。そのような場合は、トンネルからすべてのワイヤを削除してトンネルを消すことにより、いつでも正しく配線し直すことができます。

VI のトンネルが完全に重なり合っている場合は、**警告を表示**を選択するとエラーリストウィンドウに警告が表示されます。これらの問題についての詳細は、「第 4 章　VI とサブ VI の実行およびデバッグ」の「警告メッセージについて」の項を参照してください。

シーケンスストラクチャの複数のフレームから配線する

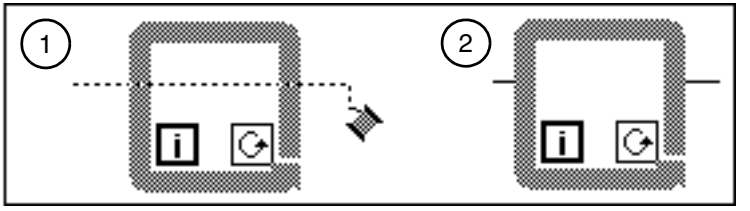
次の図は、複数信号源エラーのもう 1 つの例を示したものです。シーケンスストラクチャの 2 つのフレームが、同じトンネルに値を代入しようとしています。トンネルは、エラーを示す白で表示されています。



ストラクチャの内側ではなく下側を通して配線する

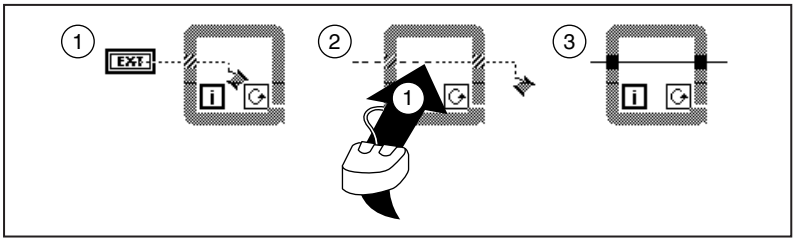
ストラクチャの内側を通して配線するためには、次に示すようにストラクチャの内側または境界線のいずれかをクリックする必要があります。

ストラクチャの内側も境界線もクリックしなかった場合は、次に示すようにワイヤはストラクチャの下側を通って配線されます。



配線ツールがストラクチャの左側の境界線と交差すると、トンネルがハイライトで表示され、マウスボタンをクリックするとそこにトンネルが作成されることを示します。マウスボタンをクリックせずにそのままストラクチャを通ってストラクチャの右側の境界線に触れる位置までドラッグすると、右側の境界線上にもう 1 つのトンネルがハイライトで表示されます。さらに、そのままマウスボタンをクリックせずに、配線ツールをストラクチャの右側の境界線の外側までドラッグすると、2 つのトンネルは消え、ワイヤはストラクチャの内側ではなく下側を通って配線されます。

次の図を確認してください。



ただし、ストラクチャの内側でワイヤの方向を転換した場合は、配線ツールを右側の境界線の外側までドラッグしてもワイヤはストラクチャの内側を通って配線されます。

ストラクチャの内容を消去せずにストラクチャを削除する

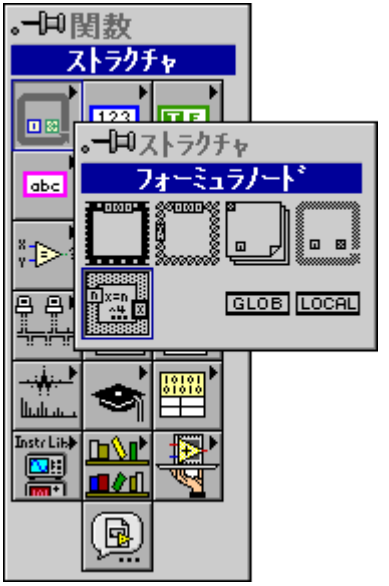
ストラクチャ（For ループ、While ループ、Case ストラクチャ、シーケンスストラクチャ）の内容を削除せずに、ストラクチャを削除することができます。これらのオブジェクトのいずれかをポップアップすると、ストラクチャを削除するための項目が表示されます。For ループと While ループの場合は、ループの内容がその下のダイアグラムにコピーされます。また、トンネルによって接続されたワイヤはいずれも自動的に相互に接続されます。

Case ストラクチャまたはシーケンスストラクチャの場合は、ストラクチャを削除すると現在のフレームまたはケースだけが残され、その他のフレームやケースはすべて削除されます。この場合、隠れているフレームやケースが失われることを示す警告が表示され、操作を取り消すことができます。

20

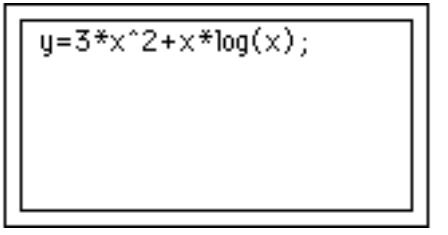
フォーミュラノード

この章では、ブロックダイアグラムでフォーミュラノードを使用して数学式を実行する方法について説明します。フォーミュラノードは、**関数→ストラクチャ**パレットから呼び出すことができます。



フォーミュラノードを使用する

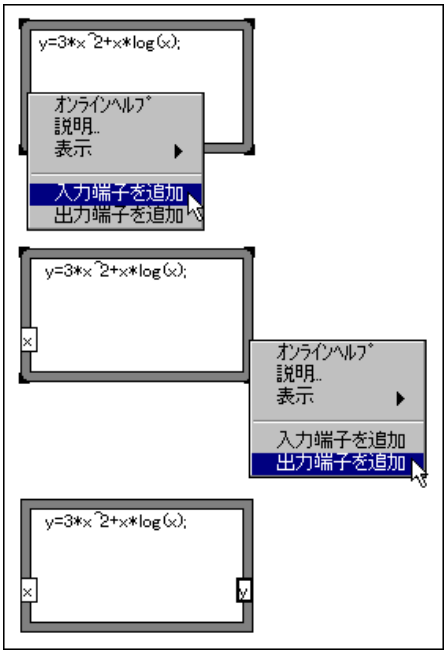
フォーミュラノードはサイズの変更が可能なボックスで、4 つのストラクチャ（For ループ、While ループ、Case ストラクチャ、シーケンスストラクチャ）とよく似ています。ただし、フォーミュラノードは次の例のようにセミコロンで区切られた公式文を格納するもので、サブダイアグラムを格納するものではありません。



公式文では、ほとんどのテキストベースのプログラミング言語が算術式に使用しているのと同じような構文を使用します。コメントを追加する場合は、スラッシュとアスタリスクでコメントを囲みます（例：`/*comment*/`）。

フォーミュラノードを使用した VI のサンプルについては、`examples¥general¥structs.llb¥Equations.vi` を参照してください。

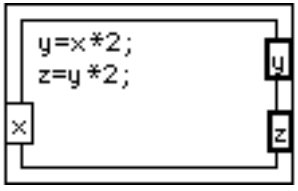
フォーミュラノードの境界線のポップアップメニューには、次の図に示すように入力変数や出力変数を追加するための項目が用意されています。



出力変数は、太枠で区別されます。

フォーミュラノードで使用する変数や公式の数には、制限がありません。また、2 つの入力あるいは 2 つの出力に同じ名前を付けることはできません。ただし、出力に入力と同じ名前を使用することは可能です。

フォーミュラノードで計算に使用する変数は、すべて入力または出力として宣言する必要があります。中間値（ノードへの入力からノードからの最終的な出力までの間に実行される計算の結果に代入される値）は、必ず出力として宣言する必要があります。ただし、中間値をフォーミュラノードの外にあるノードに配線する必要はありません。次の図では、変数 *y* の値は外部のノードに接続されていませんが、変数 *y* と変数 *z* はともに出力として宣言する必要があります。



次の図に示すように、ポップアップメニューから**出力に変更**を選択することにより、入力を出力に変更することができます。

次に示すように、ポップアップメニューから**入力に変更**を選択することにより、出力を入力に変更することができます。

変数はすべて浮動小数点数値スカラで、精度はコンピュータの構成によって決まります。変数に単位を使用することはできません。公式中に使用する入力変数は、すべて配線する必要があります。配線されたすべての出力変数は、少なくとも 1 つの式で定義する必要があります。すなわち、これらの出力変数は、等号の左側に位置していなければなりません。出力変数を等号の右側の式の中で使用することはできますが、G ではその出力変数が前の式で定義されているかどうかはチェックされません。代入が式の中で行われる場合は、中の式の値が代入される値になります。

例　$x = \sin(y = \mathrm{pi}/3)$

$\frac{\pi}{3}$ を y に代入したのち、$\sin\left(\frac{\pi}{3}\right)$ を x に代入します。

構文エラーが発生した場合は、**壊れた実行**ボタンをクリックしてエラーリストを参照します。リストでは、フォーミュラノードが公式の一部に # の記号を表示し、フォーミュラボックスがエラーを検出したことを示します。

フォーミュラノードの関数および演算子

関数名前は、すべて小文字でなければなりません。表 20-1 に、フォーミュラノード関数の名前を示します。

表 20-1 フォーミュラノード関数

関　数	対応するG関数の名前	説　明
abs(*x*)	Absolute Value	*x* の絶対値を返します。
acos(*x*)	Inverse Cosine	ラジアン単位で*x*のアークコサインを求めます。
acosh(*x*)	Inverse Hyperbolic Cosine	ラジアン単位で*x*のハイパボリックアークコサインを求めます。
asin(*x*)	Inverse Sine	ラジアン単位で*x*のアークサイン求めます。
asinh(*x*)	Inverse Hyperbolic Sine	ラジアン単位で*x*のハイパボリックアークサインを求めます。
atan(*x*)	Inverse Tangent	ラジアン単位で *x* のアークタンジェントを求めます。
atanh(*x*)	Inverse Hyperbolic Tangent	ラジアン単位で*x*のハイパボリックアークタンジェントを求めます。
ceil(*x*)	Round to +Infinity	*x* を次に大きい整数に丸めます（最小整数は*x*以上）。
cos(*x*)	Cosine	ラジアン単位で*x*のコサインを求めます。
cosh(*x*)	Hyperbolic Cosine	ラジアン単位で*x*のハイパボリックコサインを求めます。
cot(*x*)	Cotangent	ラジアン単位で*x*のコタンジェントを求めます（1/tan(*x*)）。
csc(*x*)	Cosecant	ラジアン単位で*x*のコセカントを求めます（1/sin(*x*)）。
exp(*x*)	Exponential	*e* を*x*乗した値を求めます。
expm1(*x*)	Exponential (Arg) – 1	*e* の *x* 乗から 1 を引いた値を求めます（$e^x - 1$）。

表 20-1 フォーミュラノード関数（続き）

関　数	対応するG関数の名前	説　明
floor(*x*)	Round to –Infinity	*x* を次に小さい整数に丸めます（最大整数は*x*以下）。
getexp(*x*)	Mantissa & Exponent	*x* の指数部を返します。
getman(*x*)	Mantissa & Exponent	*x* の仮数部を返します。
int(*x*)	Round To Nearest integer	引数を最も近い偶数の整数に丸めます。
intrz(*x*)	Round Toward 0	*x* を *x* と 0 の間の最も近い整数に丸めます。
ln(*x*)	Natural Logarithm	*x* の自然対数を求めます（底は *e*）。
lnp1(*x*)	Natural Logarithm (Arg + 1)	(*x* + 1) の自然対数を求めます。
log(*x*)	Logarithm Base 10	*x* の対数を求めます（底は10）。
log2(*x*)	Logarithm Base 2	*x* の対数を求めます（底は2）。
max(*x*,*y*)	Max & Min	*x* と *y* を比較して大きい方の値を返します。
min(*x*,*y*)	Max & Min	*x* と *y* を比較して小さい方の値を返します。
mod(*x*,*y*)	Quotient & Remainder	商が負の方向に丸められる場合、*x*/*y* の余りを求めます。
rand( )	Random Number (0–1)	0～1（0と1は含まない）の浮動小数点数を求めます。
rem(*x*,*y*)	Remainder	商が最も近い整数に丸められる場合、*x*/*y* の余りを求めます。
sec(*x*)	Secant	*x* ラジアンのセカントを計算します（1/cos(*x*)）。
sign(*x*)	Sign	*x* が 0 より大きいときは1、*x* が 0 と等しいときは 0、*x* が 0 より小さいときは–1 を返します。
sin(*x*)	Sine	*x* ラジアンのサインを求めます。
sinc(*x*)	Sinc	*x* のサインを *x* ラジアンで割った値を求めます（sin(*x*)/*x*）。

表 20-1 フォーミュラノード関数（続き）

関　数	対応するG関数の名前	説　明
sinh(*x*)	Hyperbolic Sine	*x* のハイパボリックサインをラジアン単位で求めます。
sqrt(*x*)	Square Root	*x* の平方根を求めます。
tan(*x*)	Tangent	*x* のタンジェントをラジアン単位で求めます。
tanh(*x*)	Hyperbolic Tangent	*x* のハイパボリックタンジェントをラジアン単位で求めます。
x^*y*	x^y	*x* の *y* 乗の値を求めます。

フォーミュラノードの構文

フォーミュラノードの構文は、バッカスーナウア（Bacus-Naur）表記法を使用して要約すると次のようになります。角カッコはオプション項目であることを示します。

```
<assignlst> := <outputvar> = <aexpr> ; [ <assignlst> ]
<aexpr>:=<expr> | <outputvar> = <aexpr>
<expr>:=<expr> <binaryoperator> <expr>
    |<unaryoperator> <expr>
    |<expr> ? <expr> : <expr>
    |( <expr>)
    |<inputvar>
    |<outputvar>
    |<const>
    |<function> ( <arglist> )
<binaryoperator>:= + | - | * | / | ^ | != | ==
    | > | < | >= | <= | && | ||
<unaryoperator>:= + | - | !
<arglist>:=<aexpr>[ , <arglist> ]
<const>:=pi | <number>
```

演算子を優先順位の高い順に並べると次のようになります。

=	代入
? :	条件
||	論理和
&&	論理積
!= ==	等しくない、等しい
< > <= >=	その他の関係（より小さい、より大きい、以下、以上）
+ -	加算、減算
* /	乗算、除算
+ - !	単項式（正、負、論理否定）
^	指数

指数および代入の演算子は右連結です（右から左にグループ化します）。その他のバイナリ演算子はすべて左連結です。TRUEの数値は1、FALSEの数値は0です（出力の場合）。0の論理値はFALSE、0以外の値の論理値はTRUEです。

条件式<lexpr> ? <texpr>: <fexpr>の論理値は、<lexpr>の論理値がTRUEのときは<texpr>、それ以外の場合は<fexpr>です。

フォーミュラノードエラー

表20-2にフォーミュラノードで検出されるエラーの一覧を示します。

表 20-2 フォーミュラノードエラー

エラーメッセージ	エラーメッセージの意味
構文エラー	演算子の使用ミス、その他。
不良トークン	文字が認識できない。
出力変数が必要です	入力変数に代入できない。
出力変数が欠けています	存在しない出力変数に代入しようとした。
変数が欠けています	参照しようとした入力変数または出力変数が存在しない。
引数が足りません	関数の引数が不十分。
引数が多すぎます	関数の引数が多すぎる。
引数リストに終端がありません	引数リストの右カッコが表示される前に公式が終了されている。
左括弧が欠けています	関数名の後ろに引数リストが存在しない。
右括弧が欠けています	対応するすべての右カッコが表示される前に公式が終了されている。
コロンが欠けています	3進の条件演算子の不正な使用。
セミコロンが欠けています	公式文がセミコロンで終了していない。
等号が欠けています	公式文が適切な代入値でない。

21

VIサーバ

この章では、VIやアプリケーションのプログラム制御のメカニズムについて説明します。また、VIやアプリケーションの遠隔制御の方法についても説明します。

Gの多くの機能には、VIサーバを通してプログラム上でアクセスすることができます。これらの機能には、Windows 95/NTプラットフォームのどのActiveXクライアントからもアクセスすることができます。また、一部のG言語関数を使用すると、あらゆるプラットフォームでVIサーバのすべての機能にアクセスすることが可能になります。さらに、これらの関数を使用すると、BridgeVIEWやLabVIEWのローカルバージョンと同様、TCP/IPネットワーク上のリモートバージョンにおいてもVIサーバのほとんどすべての操作の実行が可能になります。ActiveXのサーバおよびクライアント機能についての詳細は、『LabVIEW ユーザマニュアル』または『BridgeVIEW User Manual』、およびソフトウェアの**オンラインリファレンス**を参照してください。

VIサーバを使用する

以下に、VIサーバの使用によって可能になるいくつかのタスクを示します。

- VIをアプリケーションに静的にリンクせずに動的にメモリにロードでき、また、通常のサブVIを呼び出す場合と同じように、VIのサーバ関数を使用してそれらのVIを呼び出すことが可能になります。この機能は、大規模なアプリケーションを使用する際にメモリを節約したり、起動時間を短縮するのに役立ちます。構成に関する一連のダイアログボックスなど、アプリケーションのあまり頻繁に使用しない部分を必要なときにだけロード、実行されるようにしておくと、アプリケーションをメモリにロードするための時間だけでなく、アプリケーションによるメモリの消費量も節約することができます。ユーザが操作を終えた後は、VIを解放して再度メモリを使用可能な状態に戻すことができます。
- VIインタフェースのプログラム制御が可能になります。たとえば、動的にVIウィンドウの場所を決定したり、パネルの特定の場所が見られるようにパネルをスクロールしたり、あるいはパネルウィンドウを閉じたり開いたりすることが必要になる場合があります。VIのフロントパネルのこれらのプロパティは、いずれもVIサーバを通して制御することができます。

- BridgeVIEWやLabVIEWからインターネットを通して呼び出せる関数をエクスポートするためのサーバアプリケーションを、簡単に作成することが可能になります。たとえば、遠隔地のデータを集録し、記録するデータ集録アプリケーションを使用し、ローカルマシンからそのデータをサンプリングする事が必要になる場合があります。その場合、簡単な環境設定を行い、ある種のVIをインターネットを通して呼び出せるようにすることで、サブVIを呼び出すのと同じくらい簡単に最新のデータを転送することが可能になります。ネットワークの詳細はすべてVIサーバによって処理されるため、サーバのクライアントを実行しているプラットフォームの種類には関係なく通信できます。
- VIのプロパティをプログラム上で変更し、変更したデータをディスクに保存することが可能になります。たとえば、アプリケーションを開発する際には、デバッグ機能や実行時のポップアップメニューを使用でき、スクロールバーが表示され、かつウィンドウのサイズを変更できるようにVIを構成する一方で、アプリケーションを配布する際にはこれらの機能をオフにし、さらにVIの最入実行、VIを実行するための実行システムの種類、あるいはVIのヘルプファイルへのパスといった他のある種の属性を正しく設定することが必要になる場合があります。このような属性は、すべてVIサーバを介してプログラム上で設定、参照できるため、VIを編集するためのアプリケーションを作成すれば、VI設定ダイアログボックスによってすべてのVIを1つずつを設定する必要がなくなります。
- アプリケーション用のプラグインアーキテクチャを作成することにより、アプリケーションをユーザに配布した後でもアプリケーションに機能を追加することが可能になります。たとえば、データにフィルタをかけるVIのセットがあり、それらがすべて同じパラメータを使用するものとします。これらのフィルタが1つのプラグインディレクトリから動的にロードされるように設計したアプリケーションに、フィルタセットの一部のフィルタだけを組み込んだ状態で出荷すれば、後で新しいフィルタのVIをプラグインディレクトリにロードするだけで簡単にフィルタを追加できるため、ユーザに幅広いオプションを提供することができます。

VIサーバの機能

VIサーバの機能には、2つの主要なクラスのオブジェクトすなわちアプリケーションオブジェクトとVIオブジェクトのリファレンスを通してアクセスします。これらのいずれか一方のオブジェクトのリファレンスを作成したら、そのリファレンスをそのオブジェクトに対して実行する関数に渡します。作業が終わったら、リファレンスを閉じます。このプログラミングのルールは、ファイルI/Oおよびネットワークのリファレンスとよく似ています。

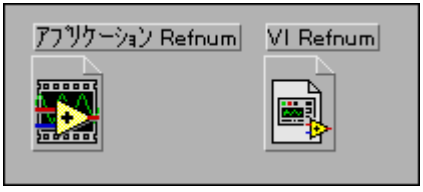
アプリケーションとVIのリファレンスで重要な点は、ネットワークにおける透過性です。つまり、オブジェクトのリファレンスは、リモートのマシンでもユーザ自身のマシンで開くのと同じ方法で開くことができます。リモートのオブジェクトのリファレンスを開いたら、一部の制約を除いてローカルオブジェクトとまったく同じ方法でそのリモートオブジェクトを処理することができます。リモートオブジェクトの操作の場合、ネットワークを通した操作に関する情報の送信および結果の受信は、VIサーバが行います。プログラムは、操作がリモートであるかローカルであるかに関係なく見た目にはまったく同じように機能します。

Gアプリケーションのリファレンスを使用すると、アプリケーションを実行しているプラットフォーム、BridgeVIEWやLabVIEWのバージョン、現在メモリに入っているすべてのVIのリストなど、Gの環境に関する情報を入手することができます。また、現在のユーザの名前や他のアプリケーションにエクスポートしたVIのリストなどの情報を設定することもできます。

VIへのリファレンスがある場合は、メモリにそのリファレンスをロードします。リファレンスをロードすると、VIはそのリファレンスが閉じられるまでメモリに常駐します。ただし、VIに対して同時に複数のリファレンスが開かれている場合もあります。そのような場合は、VIはそのVIに対するすべてのリファレンスが閉じられるまでメモリに常駐します。VIのリファレンスを使用すると、フロントパネルウィンドウの位置などの動的プロパティだけでなく、**VI設定**で使用できるVIのあらゆるプロパティを入手および設定することができます。また、VIをプログラム上で印刷したり、別の場所に保存したり、あるいは翻訳を目的として文字列をエクスポートしたりインポートしたりすることもできます。

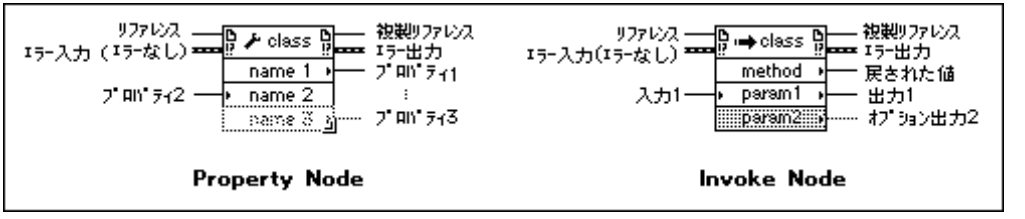
アプリケーションリファレンスとVIリファレンス

ほとんどのアプリケーションでは、VIへのリファレンスはそれぞれOpen Application Reference関数とOpen VI Reference関数を使用して作成します。リファレンスをフロントパネルに表示したいときは、**制御器**パレットから**パス & Refnum**を選択し、アプリケーションまたはVIのRefnumを選択します。このrefnumのデータタイプは、デフォルトではApplication Refnumと表示されます。これをVI Refnumに変更したい場合は、refnumをポップアップしてVIサーバクラスを**選択→Virtual Instrument**を選択します。



アプリケーションリファレンスおよびVIリファレンスに対してプロパティノードやインボークノードを使用する

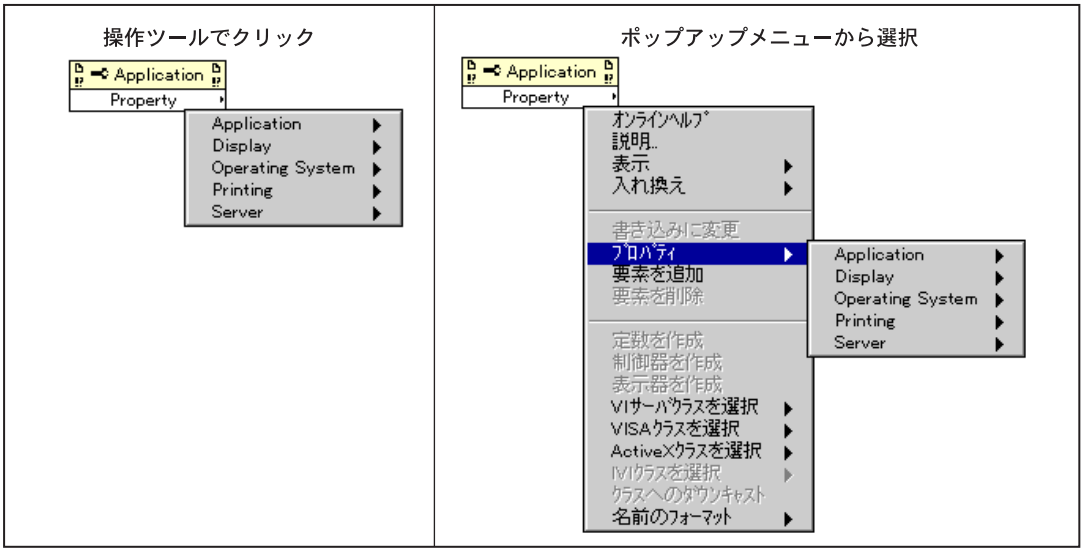
アプリケーションリファレンスやVIリファレンスに対する操作の多くは、プロパティノードまたはインボークノードを通してのみ実行できます。この2つのノードは、上の部分に入力と出力が2つずつ、その下に各種の入力および出力があるという点でよく似ています。



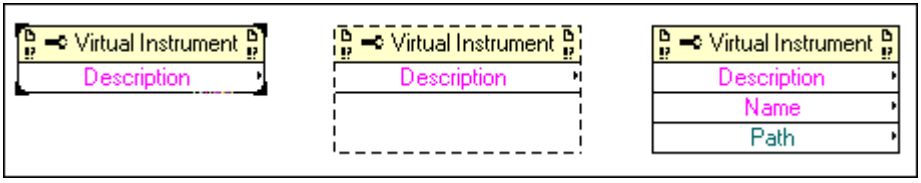
リファレンスはアプリケーションまたはVIで、**複数リファレンス**は他の関数に値を渡すために使用されるリファレンスのコピーです。**エラー入力**は標準のエラークラスタで、エラーの発生の有無を示すデータ、エラーコード、およびエラーの説明で構成されます。**エラー入力**にエラーが存在する場合は、ノードは何も実行せず、単にエラーだけを**エラー出力**に渡します。**エラー入力**にエラーが存在せず、ノードの実行中にエラーが発生した場合は、**エラー出力**にはそのエラーに関する情報が含まれます。

どちらのノードも、リファレンスの入力に関しては多形性を有しています。アプリケーションまたはVIのリファレンスをこの入力に配線すると、ノードには自動的にそのデータタイプが適用され、そのデータタイプに該当する操作だけが使用可能になります。

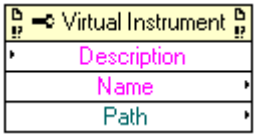
プロパティノードを使用すると、アプリケーションまたはVI上でのさまざまなプロパティの入手（読み取り）や設定（書き込み）が可能になります。プロパティは、ノードのポップアップメニューから操作ツールでプロパティ端子をクリックするか、または端子をポップアップすることによって選択できます。



1 つのノードを使用して複数のプロパティを入手または設定することができます。プロパティノードを拡大すると、新しい端子が追加されます。

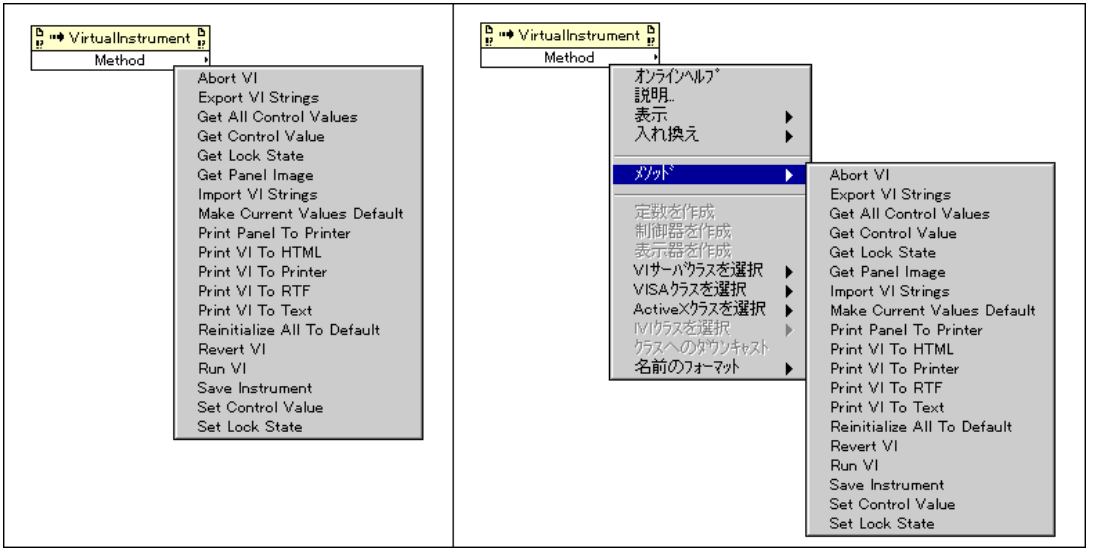


次の図に示すように、1 つのプロパティノードで複数のプロパティを読み込んだり書き込んだりすることができます。読み取りの対象となるプロパティは右側の小さい矢印で示され、書き込みの対象となるプロパティは左側の小さい矢印で表示されます。プロパティを読み込むか書き込むかは、ポップアップメニューで**読み取りに変更**または**書き込みに変更**を選択することによって指定します。

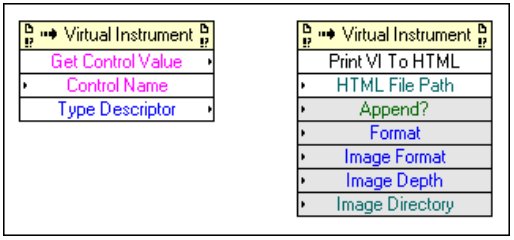


ノードは上から下の順に実行されます。ノードの途中でエラーが発生した場合は、残りのプロパティは無視され、エラーが返されます。エラー文字列には、どのプロパティでエラーが発生したかを示す情報が含まれます。

インボークノードを使用すると、アプリケーションまたはVIに対して動作、すなわちメソッドを実行することができます。プロパティノードとは異なり、1つのインボークノードはアプリケーションまたはVIに対して1つのメソッドしか実行できません。実行可能なメソッドには、操作ツールでメソッド端子をクリックするか、またはプロパティノードの場合と同じように端子をポップアップすることによってアクセスできます。



メソッドの名前は、必ずパラメータリストの最初の端子の名前になります。関数が値を返す場合は、その結果がメソッド端子の値になります。値を返すメソッドには、たとえばGet Control Valueなどがあります。関数が値を返さない場合（Print VI To HTMLなど）は、メソッド端子は値を持たないことになります。



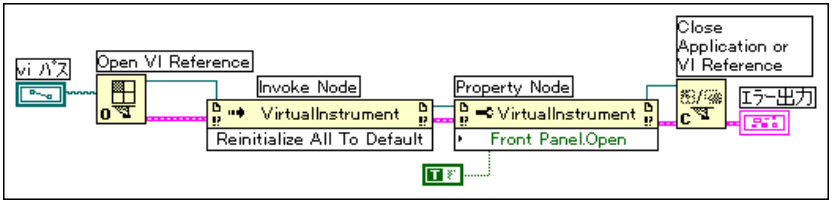
インボークノードではパラメータが上から順に表示され、オプションのパラメータは下にグレーで表示されます。上の図では、HTML File Pathは必須パラメータであるのに対し、Append?、Format、Image Format、Image Depth、およびImage Directoryはいずれもオプションのパラメータです。

VIクラスとアプリケーションクラスのプロパティおよびメソッドの例

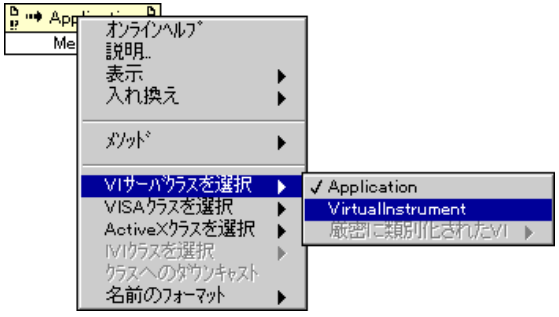
以下の項では、一般的なVIおよびアプリケーションのクラスの使用方法について説明します。VIやアプリケーションのクラスプロパティやメソッドのその他の使用例については、`examples¥general¥viserver`を参照してください。

VIクラスのプロパティとメソッドを操作する

VIのプロパティは、VIでのメソッドの実行とは無関係に設定することができます。アプリケーションによっては、VIのプロパティへのアクセスとVIでのメソッドの実行の両方が必要になる場合もあります。次のダイアグラムでは、VIのフロントパネルオブジェクトがそれぞれのデフォルト値に初期化されてからフロントパネルが開かれ、これらのデフォルト値が表示されます。



VI上でプロパティやメソッドにアクセスする前に、Open VI Referenceを実行してそのVIへのリファレンスを作成する必要があります。VI上でメソッドを呼び出すためには、Invoke Nodeを使用します。Open VI ReferenceからInvoke Nodeにワイヤを接続すると、VIのすべてのクラスのメソッドにアクセスできるようになります。また、ノードをポップアップし、**VIサーバクラスを選択→Virtual Instrument**を選択することによってもVIクラスのメソッドにアクセスすることができます。

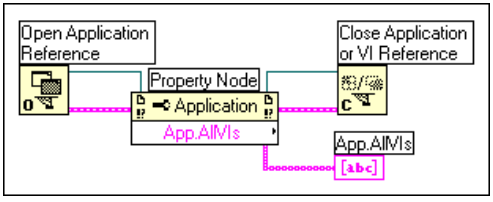


プロパティノードは、インボークノードと同じように動作します。プロパティノードにVIリファレンスを接続すると、VIクラスのすべてのプロパティにアクセスできます。また、ノードをポップアップして**VIサーバクラスを選択→**Virtual Instrumentを選択することもできます。

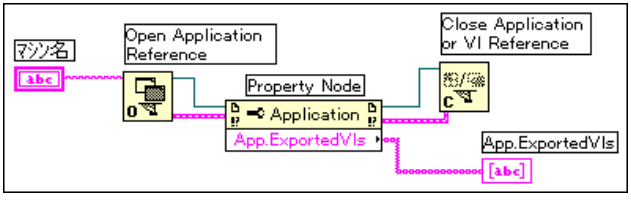
プロパティ値を設定する場合は、ポップアップして矢印が左側にあることを確認したのち、書き込みに変更を選択します。エラーが発生していないか常にチェックしてください。インボークノードとプロパティノードは、実行前にエラーが発生していると実行されません。これらのノードの1つでエラーが発生すると、エラーが発生したプロパティまたはメソッドがエラーメッセージの中に示されます。

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

ローカルまたはリモートのLabVIEWでのプロパティの設定や読み取りは、LabVIEWでのメソッドの実行に関係なく行うことができます。次の図では、ローカルマシンのメモリに入っているVIがフロントパネル上で文字列配列中に表示されています。



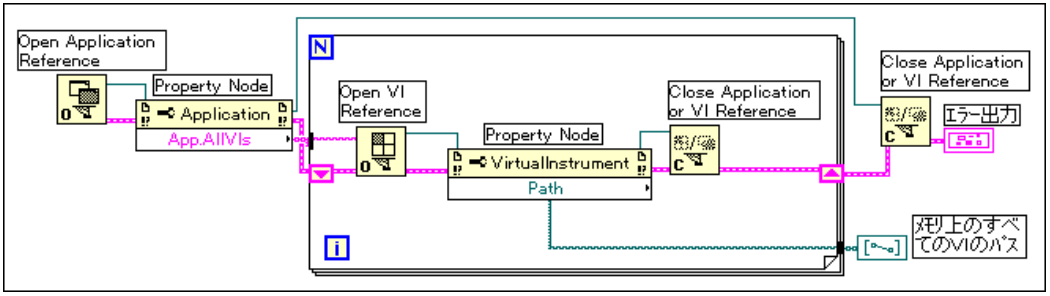
リモートマシンのメモリに入っているVIを検索するためには、次の図に示すように文字列制御器をマシンの名前の入力に配線し、マシンのIPアドレスの数値またはドメイン名を入力します。また、前図で使用しているAll VIs in Memoryという属性はLabVIEWおよびBridgeVIEWの旧バージョンにしか適用されないため、属性をExported VIs in Memoryに変更する必要があります。



常に、エラーが発生していないかチェックしてください。プロパティノードは、実行する前にエラーが発生していると実行されません。プロパティノードでエラーが発生した場合は、エラーが発生した場所がエラーメッセージの中に示されます。

VIクラスと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

VIクラスとアプリケーションクラスのプロパティおよびメソッドは、別々に使用することができます。アプリケーションによっては、両方のクラスからプロパティやメソッドにアクセスしなければならない場合があります。次のブロックダイアグラムでは、ローカルマシンのメモリに入っているVIが決定され、これらのVIのそれぞれのパスがフロントパネルに表示されています。メモリ内のVIをすべて見つけるには、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VIのそれぞれのパスを決定するためには、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、メモリに入っているVIの数によって決まります。VIリファレンスはメモリ内のそれぞれのVIに対して必要となるため、Open VIとClose VIはForループの内部で使用する必要があります。アプリケーションのリファレンスは、すべてのVIのパスを読み取ってから閉じるようにしてください。



厳密に類別化されたVI Refnum

厳密に類別化されたVI Refnumは、動的にVIを呼び出すアプリケーションでのみ使用されます。そのようなアプリケーションでは、2つのケースで厳密に類別化されたVI Refnumを使用します。その1つの、おそらく最も一般的なケースは、厳密に類別化されたVI Refnumをサブ VIにパラメータとして渡す場合です。このような場合は、厳密に類別化されたVI Refnumの制御器をVIのコネクタペーンに接続し、refnum端子をCall By Reference Nodeの入力に配線します。refnumの値は、この場合には呼び出すVIを決定するために使用されます。

もう1つのケースは、厳密に類別化されたVI Refnumを Open VI Reference関数のタイプ指定子の入力に配線した場合です。このようなケースでは、制御器の値は無視され、refnumのタイプだけがこの関数によって使用され

ます。refnumのタイプは、開いているVIが厳密に類別化されたVI Refnumのコネクタペーンと同じコネクタペーンを持っているかどうかを決定するために使用されます。

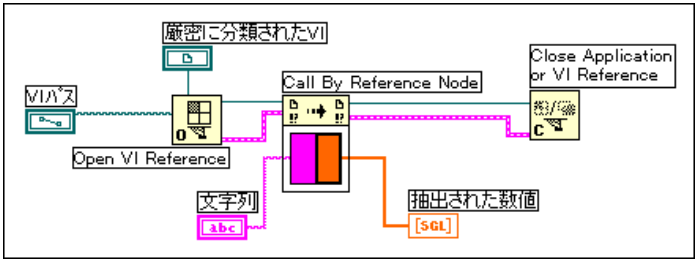
厳密に類別化されたVI Refnumを作成するためには、**制御器→パス & Refnum→アプリケーションまたはVI Refnum**を選択してVIのrefnumをフロントパネル上にドロップします。ポップアップして**VIサーバクラスを選択→参照...**を選択します。VIを指定するためのダイアログボックスが表示されます。

ただし、厳密に類別化されたVI Refnumに対してVIを指定した場合でも、コネクタペーンの情報だけが保存される点に注意してください。このことは、rerfnumとVIの間に永久的な関係が確立されるわけではないことを意味します。特に、VIのコネクタペーンを選択することと、選択したVIのリファレンスを入手することを混同しないようにしてください。特定のVIは、Open VI Reference関数で**VIパス**の入力を通して指定します。

厳密に類別化されたVI Refnumに対してコネクタペーンを選択すると、そのコネクタペーンはVIのrefnumの**VIサーバクラスを選択→厳密に類別化されたVI**サブメニューに表示されます。ただし、BridgeVIEWあるいはLabVIEWを終了すると、次回アプリケーションを立ち上げたときにこれらのコネクタペーンは表示されません。

厳密に類別化されたVI Refnumの例

厳密に類別化されたVI Refnumを使用する唯一のケースは、動的にVIを呼び出す場合に限られます。次のブロックダイアグラムでその例を示します。



VIパスと**厳密に類別化されたVI Refnum**をOpen VI Reference関数に配線すると、Open VI Reference関数をCall By Reference Nodeに接続すること

が可能になります。コネクタペーンは、Call By Refernce Nodeに自動的に表示されます。それにより、任意の入力値と出力の表示器をコネクタペーンに配線できるようになります。Call By Rerference Nodeは、指定された任意の入力値を使用してVIリファレンスの入力で指定されたVIを呼び出します。このノードは、すべてのサブVIを1度にメモリにロードしたくないときに使用すると便利です。

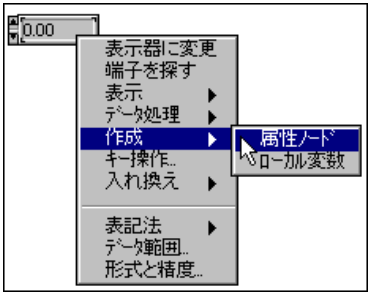
22

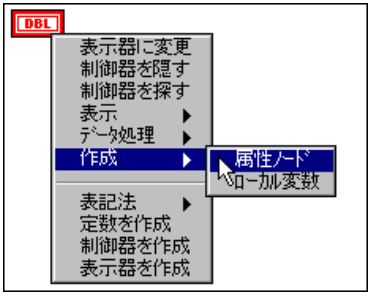
属性ノード

この章では、属性ノードを使用してプログラム上でフロントパネル制御器の属性の設定および読み取りを行う方法について説明します。便利な属性としては、表示色、制御器の可視性、リング制御器のメニューの文字列、グラフやチャートのプロットの色、グラフカーソルがあります。

属性ノードを作成する

属性ノードは、フロントパネル制御器のポップアップメニューまたは制御器の端子から**作成→属性ノード**項目を選択して作成します。



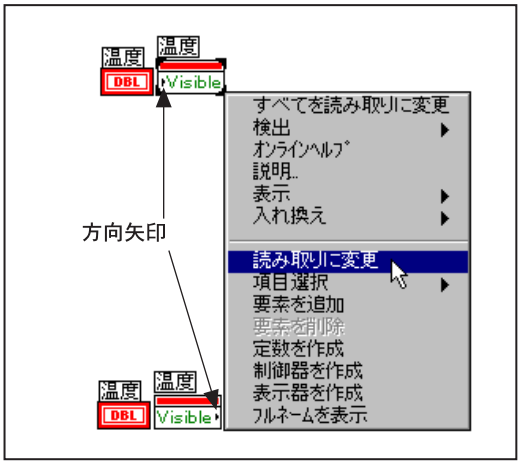


この項目を選択すると、次の図に示すように制御器の端子の近くのダイアグラム上に新たなノードが作成されます。制御器にラベルが存在する場合は、制御器のラベルが属性ノードの最初のラベルとして使用されます。このラベルは、ノードを作成した後で変更することができます。

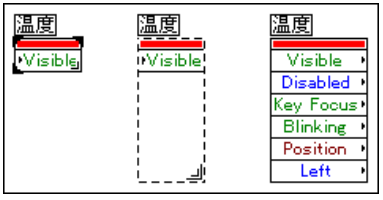
次の図に示すように、属性の端子をポップアップして**項目選択**を選択すると、制御器から書き込みや読み取りのできる属性のメニューが表示されます。操作ツールで属性ノードをクリックすると、属性のリストへのショートカットを使用することができます。

属性の読み取り、書き込みのどちらを実行するかは、次の図に示すように属性ノードのポップアップメニューから**読み取りに変更**または**書き込みに変更**のいずれかを選択して指定します。

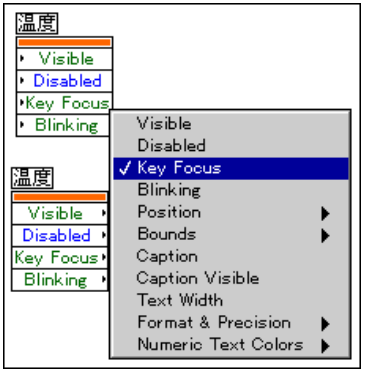
端子の左側に小さい矢印がある場合は、属性を書き込むことができます。端子の右側に矢印がある場合は、属性を読み取ることができます。



属性ノードを拡張すると、同じノードで複数のノードの読み取りまたは書き込みを行うことができます。ノードを拡張すると、新しい端子が追加されます。属性ノードは、上から下の順に実行されます。

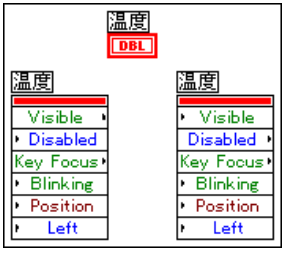


端子をある属性に関連づけるためには、操作ツールで端子をクリックし、属性ノードのポップアップメニューから属性を選択します。

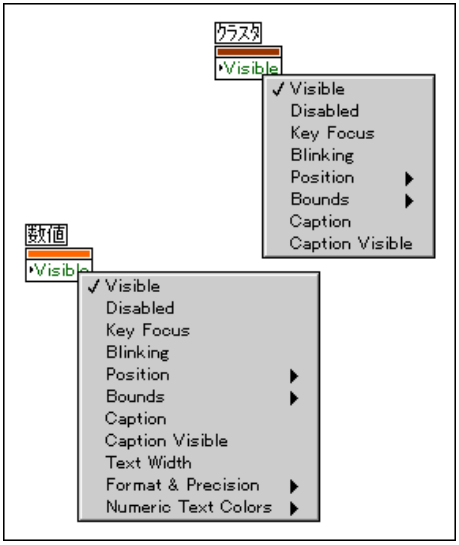


既存のノードの複製を作成するか、または再度**作成→属性ノード**項目を選択することにより、属性ノードの複製を作成することができます。既存のノードの複製を作成するためには、位置決めツールでそのノードをクリックし、<Ctrl>**（Winows）**、<option>**（Macintosh）**、<meta>**（Sun）**、または<Alt>**（HP-UX）**キーを押しながらドラッグします。ただし、**編集**メニューコマンドを使用して属性のコピーおよび貼り付けを行った場合は、属性ノードおよびその属性ノードが参照する制御器の両方のコピーが作成されます。

属性ノードの個々のコピーは、読み取り用の端子と書き込み用の端子をそれぞれ別個に持つことができます。

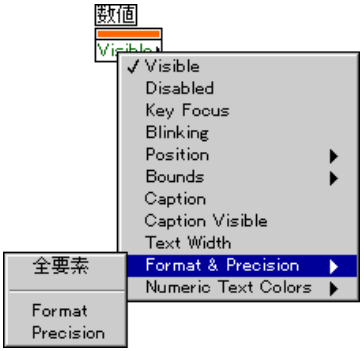


クラスタ、およびクラスタ内の数値制御器の属性を次の図に示します。



制御器の中には、グラフのように読み取ったり設定したりできる端子を数多く持つものもあります。これらの属性のうち、あるものはいくつかのカテゴリにまとめられ、サブメニューにそのリストが表示されます（たとえば数値制御器の場合の Format & Precision カテゴリー）。サブメニューか

ら**全要素**オプションを選択すると、すべての属性を一度に設定することができます。また、特定の属性を個別に選択して1つまたは複数の属性を別々に設定することもできます。次の図に、1つの例として数値制御器のFormat & Precisionオプションを示します。



属性ノードを作成すると、端子および制御器のポップアップメニューの**制御器を探す**および**端子を探す**という項目が、属性ノードを検索するためのサブメニューに変わります。同様に、属性ノードにはその属性ノードが関連付けられている制御器や端子を検索するためのオプションがあります。

属性ノードの使用方法の例については、
`example¥general¥atribute.llb`を参照してください。

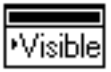
属性ノードのヘルプを使用する

ヘルプウィンドウと**オンラインリファレンス**（**ヘルプ**メニュー）は、属性ノードの使用に際して非常に役に立つツールです。これらを利用すると、説明や、データタイプ、属性に対して使用できる値を検索することができます。詳しくは、「第1章　Gプログラミングの概要」の「属性のヘルプ」の項を参照してください。

基本属性

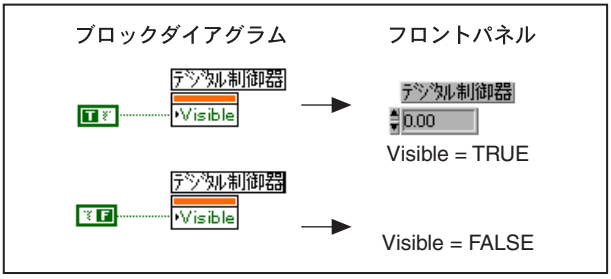
アプリケーションのさまざまなフロントパネルオブジェクトを変更するための属性は数多くあります。この項では、ほとんどすべてのフロントパネルオブジェクトに共通のVisible、Disabled、Key Focus、Blinking、Position、およびBoundsについて説明します。

Visible



フロントパネルオブジェクトのVisibleは、可視性属性を使用して読み取ったり書き込んだりすることができます。関連付けられたオブジェクトは、この属性がTRUEのときは表示され、FALSEのときは表示されません。

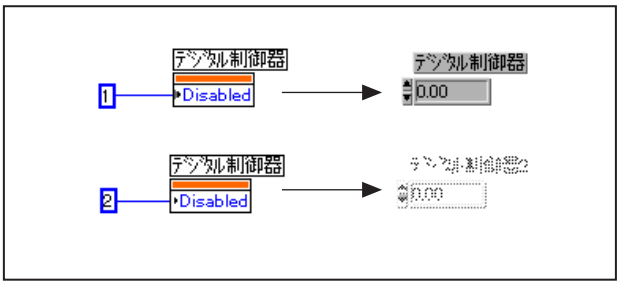
次の図で、デジタル制御器は見えないように設定されています。ブール値をTRUEにすると、この制御器は図のようにまた見えるようになります。



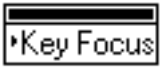
Disabled

ユーザがオブジェクトにアクセスできるようにするかできないようにするかは、Disabledを使用して制御します。値が0のときは、オブジェクトは有効になり、ユーザはそのオブジェクトを操作できます。値が1のときはオブジェクトは無効になり、操作が禁止されます。値が2のときは、オブジェクトは無効になり、グレーで表示されます。

デジタル制御器にユーザがアクセスできないようにすることができます。最初の例では、制御器は無効になっても外観は変わりません。2番目の例では、デジタル制御器はユーザによるアクセスが禁止されているため、グレーで表示されています。

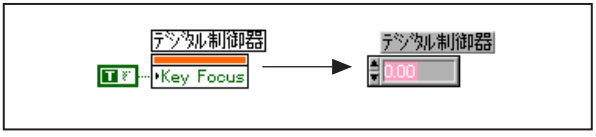


Key Focus



Key Focus を使用すると、制御器をキーフォーカスにしたり、制御器が現在キーフォーカスになっているかどうかをチェックすることができます。キーフォーカスの状態にある制御器は、<Tab>キーでその制御器に移動して制御器をアクティブにしたときと同じように反応します。ほとんどの制御器では、キーボードで値をタイプして制御器に値を入力することができます。また、実行モードで<Tab>キーを押すか、または制御器に関連付けられたホットキー（**キー操作**項目を使用して割り当てます）を押してフロントパネル上でキーフォーカスを設定することができます。

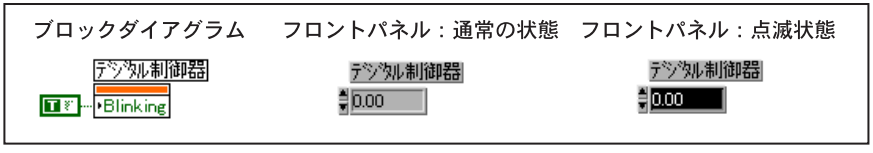
デジタル制御器をキーフォーカスにすると、マウスで制御器を選択しなくても制御器に値を入力することができます。



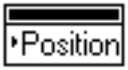
Blinking

Blinking を使用すると、オブジェクトの点滅状態の読み取りや設定が可能になります。この属性を TRUE に設定した場合は、フロントパネルオブジェクトが点滅します。点滅速度や色は、**環境設定**ダイアログボックスで設定します。この属性を FALSE に設定すると、オブジェクトの点滅は停止します。

次の図では、フロントパネル表示器が点滅するように設定されています。

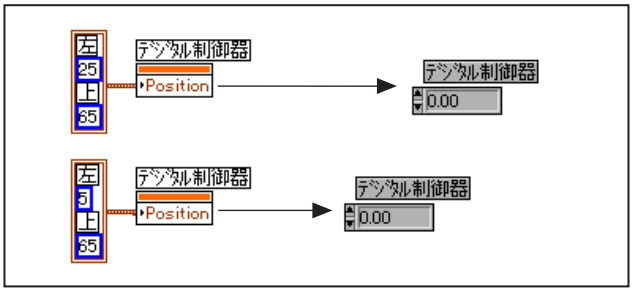


Position

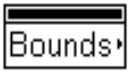


フロントパネル上のオブジェクトの左上隅の位置の設定や読み取りは、Position を使用して行うことができます。位置は、パネルウィンドウの原点に対する相対的な距離（ピクセル数）によって決定されますが、最初は、ウィンドウの左上隅が原点になります。ウィンドウをスクロールすると、原点は移動します。この属性は、符号なしの 2 つの倍長整数からなるクラスタで構成されます。クラスタの最初の項目（Left）は、パネルウィンドウの原点に対する制御器の左端の相対位置、クラスタの 2 番目の項目（Top）は、同じく原点に対する制御器の上端の相対位置を表します。

次の図に示すように、位置属性ノードを実行することにより、フロントパネル上のデジタル制御器の位置が変化します。



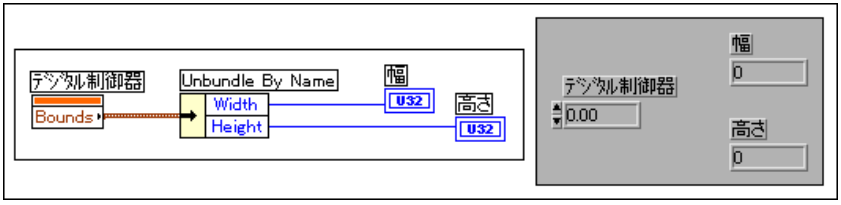
Bounds（読み取り専用）



Bounds は、フロントパネル上のオブジェクトの境界をピクセル単位で読み取ります。値には、制御器、および制御器のあらゆる部品（ラベル、凡例、スケールなど）が含まれます。この属性は、符号なしの 2 つの倍長整数からなるクラスタで構成されます。クラスタの最初の項目（Width）は、オブジェクトの幅を、クラスタの 2 番目の項目（Height）はオブジェクトの高さを表します（単位はいずれもピクセル）。

Bounds は、**読み取り専用**の属性です。フロントパネル上の制御器や表示器のサイズを**変更するものではありません**。グラフやチャートにプロット領域サイズ属性があるように、ほとんどのオブジェクトには他にも属性があります。この属性は、すべての部品を含む制御器全体のサイズを読み取ることができるため、他の制御器を現在作業している制御器に対して相対的に配置することができます。

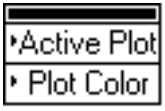
デジタル制御器の境界は、次の図に示すようにして読み取ることができます。



制御器あるいは表示器に固有の属性の例

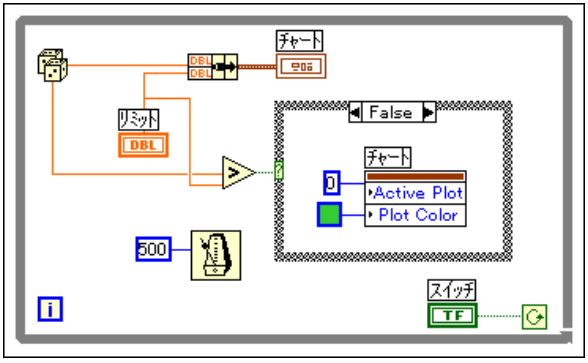
以下の項では、属性ノードの一般的な使用方法について説明します。属性ノードのその他の使用方法については、`general` ディレクトリを参照してください。

チャートのプロットの色を変更する



左記の属性は、アクティブなプロット（今後のトレース固有属性の設定または読み取りの対象となるトレース）、およびアクティブなプロットの色の設定や読み取りを行います。Active Plot の値は、マルチプロットチャートのどのプロットをアクティブにするかを示す整数です。Plot Color の値は、使用したい色を表す整数です。Plot Color には、属性リストから Plot Info→Plot Color を選択することによってアクセスできます。

この属性ノードは、Active Plot で指定されたプロットの色を変更します。次の例では、値がユーザにより指定された限界を超えると乱数プロットの色が変わります。ここでは、プロットの色が変わる前にアクティブプロットが指定されている点に注意してください。



ブール属性の文字列を設定する

このブール属性は、ブール制御器のラベルの設定または読み取りを行います。入力は、最大 4 つの文字列（それぞれ False、True、True トラッキング、False トラッキングの状態に相当）からなる配列です。

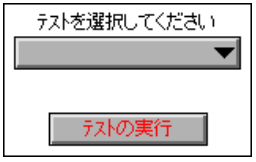
- True と False：　ブールのオンとオフの状態
- True と False のトラッキング：　ブールの一方の状態からもう一方の状態への一時的な移行のレベル。True トラッキングは、ブールが True から False に変わる場合の移行です。False トラッキングは、ブール属性が変更されたときの False から True への移行です。トラッキングは、機械的な解放されたらスイッチおよび解放されたらラッチの動作をともなうブールにのみ適用されます。これらの機械的動作は、マウスボタンを放すまで移行状態を保ちます。移行状態にあるときは、True トラッキングおよび False トラッキングで設定されたテキスト文字列が表示されます。

ブール制御器の表示文字列は、**停止**、**実行**、**停止？**、**実行？**という文字列に設定することができます。

リング制御器の文字列を設定する

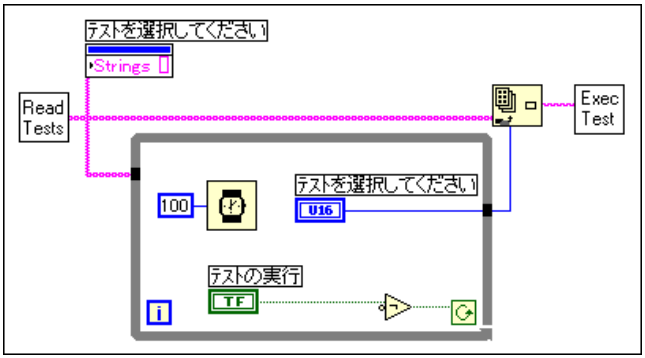
リング制御器は、現在選択されている項目の数値を格納するポップアップメニュー制御器です。この制御器を使用すると、ユーザに対してオプションのリストを提示することができます。実行時までオプションを決定できない場合は、属性ノードを使用してオプションを設定することができます。リング制御器についての詳細は、「第 13 章　リストとリングの制御器および表示器」を参照してください。

次の例では、テストのリストを表示するリング制御器のあるパネルがユーザに提示されます。ユーザはテストを選択し、**テストの実行**ボタンをクリックしてプロセスを続行します。



次の図に示したダイアグラムは、ファイルから有効なテストのリストを読み取り、そのリストを文字列の配列の形でリング制御器の属性ノードに渡します。ダイアグラムはループに入り、ユーザが**テストの実行**ボタンをクリックするのを待ちます。それにより、ユーザにはリング制御器からテストを選択したり、他の制御器のための情報を入力する機会が与えられます。

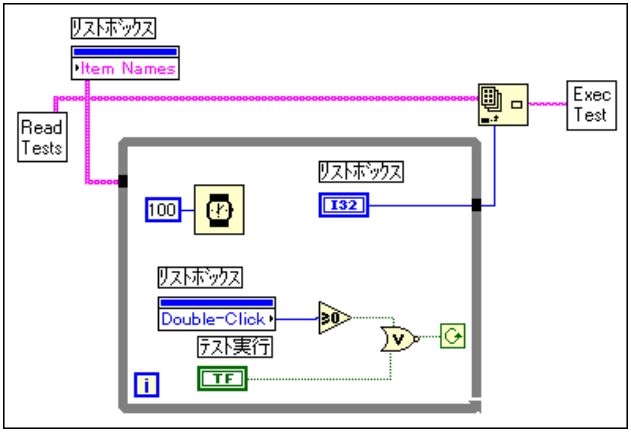
ユーザがテストを選択すると、制御器の数値に対応する文字列が読み取られ、そのテストを実行するVIに渡されます。



ダブルクリックされたリストボックスの項目を使用する

Double-Click は、リストボックスに固有の読み取り専用の属性です。この属性は、フロントパネル上のリストボックスのどの項目がダブルクリックされたかを示します。Double-Click の値は、項目をクリックするか、あるいは項目を選択したあとで<Enter>（**Windows** または **UNIX**）または<Return>（**Macintosh** および **Sun**）キーを押すことによって設定されます。Double-Click の値は、何もダブルクリックしなかった場合は–1になります。この値は、属性ノードを使用して読み取ったあとは–1にリセットされます。また、別の項目を選択したり、他の属性を設定した場合にも–1にリセットされます。

次の図は、実行するテストをDouble-Click ノードを使用して決定する場合の例を示したものです。



テストの実行ボタンをクリックするか、または項目をクリックすると、ループは停止します。項目をダブルクリックした場合は、Double-Click は –1 ではなく項目番号を返します。

オプションを選択的にユーザに表示する

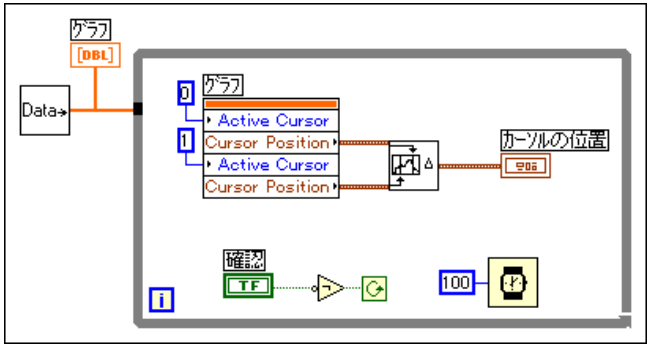
ユーザが選択を行う際には、ユーザに対して他のオプションを提示することが必要になる場合があります。これには3通りのケースが考えられます。

1. 1 つのオプションでポップアップサブ VI を使用します。ユーザに提示するオプションを持ったサブ VI を作成することができます。サブ VI を作成する際に、**VI 設定→実行オプション**の**呼び出されたらフロントパネルを表示する**および**元に閉じてあったら閉じる**を使用することにより、呼び出し時にこれらのいずれかのサブ VI が開くようにすることができます。
2. オプションを提示するもう 1 つの方法は、属性ノードの `Visible` オプションを利用して制御器を選択的に表示したり隠したりする方法です。
3. オプションを提示する 3 番目の方法は、属性ノードの `Disabled` オプションを利用して制御器を選択的に無効にする方法です。制御器が無効になると、ユーザは値を変更することはできません。

プログラム上でグラフカーソルを読み込む

属性ノードを使用すると、マルチプロットグラフの 1 つのプロットの情報や、複数のスライダを持つスライドの 1 つのスライダにアクセスすることができます。ただし、その場合には、操作の対象となる項目を指定する必要があります。ダイアグラムからアクセスする前にアクティブにしておく必要がある属性のよい例として、グラフの複数のカーソルがあります。

次のブロックダイアグラムは、データをグラフで表示し、グラフカーソルの位置をプログラム上で読み取る VI を示したものです。



Whileループを通過する際、VIはまず最初にカーソル0をアクティブにし、その数値を読み取ります。次に、カーソル1をアクティブにしてその値を読み取ります。さらに、計算を実行し、カーソルの選択に関するデータをフロントパネルに表示します。確認ボタンを押すと、VIはループを終了してカーソルの位置を確定します。

グラフカーソルをプログラム上で読み取るアプリケーションについては、`example¥general¥graphs¥zoom.llb`を参照してください。

23

グローバル変数とローカル変数

この章では、グローバル変数とローカル変数の定義方法および使用方法について説明します。グローバル変数は、複数のVIから簡単に、共有したい値にアクセスするために使用します。ローカル変数は、1つのVIの中で同様の目的に使用します。

グローバル変数とローカル変数は、Gの中でも高度な概念です。これらの変数は、この章の内容を十分理解した上で使用してください。

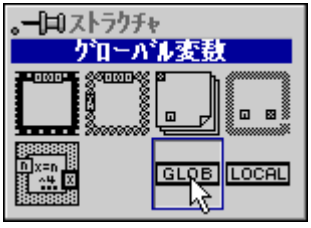
グローバル変数とローカル変数の使用方法の例については、`examples¥general¥globals.llb`および`examples¥general¥locals.llb`を参照してください。

グローバル変数

グローバル変数は、組み込まれたGオブジェクトです。グローバル変数を作成すると、特殊なVIが自動的に作成されます。そのVIに含まれるグローバル変数のデータタイプを定義するフロントパネル制御器を、このVIに追加します。

複数のグローバル変数を作成する方法は2通りあります。それぞれが1つのグローバルを持つ複数のVIを作成する方法と、1つのグローバル変数用フロントパネル上で複数の制御器を定義して、複数のグローバル変数を持つ1つのVIを作成する方法です。ただし、関連のある変数をグループにまとめることができるため、複数のグローバル変数を持つ1つのVIを作成する方法の方が効率的です。

グローバル変数は、**関数→ストラクチャ**パレットでグローバル変数を選択し、ダイアグラム上に配置することによって作成します。



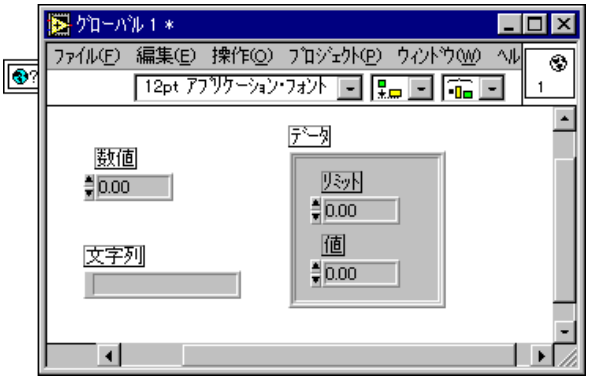

グローバルノード

ブロックダイアグラム上に、グローバル変数のノードが表示されます。

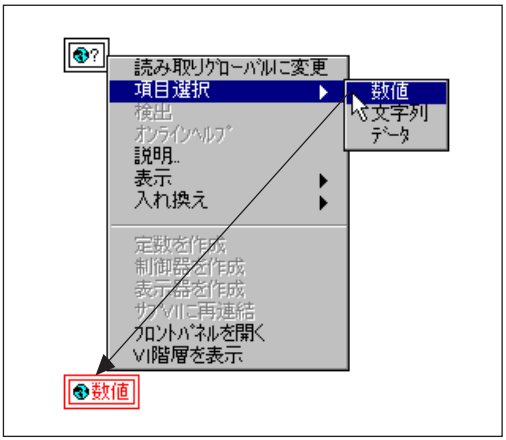
ノードのフロントパネルを開くためには、ノードをダブルクリックします。このパネルは、1 つまたは複数のグローバル変数のデータタイプを定義するために使用します。

☞ **注**　**グローバル変数は名前で検索する必要があるため、それぞれの制御器には名前（ラベル）をつけてください。グローバル変数のデータタイプを定義したら、グローバル VI を保存します。**

次の図で、3 つのグローバル変数（数字、文字列、および 2 つの値からなるクラスタ）を持つフロントパネルの例を示します。



ブロックダイアグラム上にグローバル変数を配置し、そのフロントパネルを定義すると、ノードがそのフロントパネルと関連付けられます。1 つのグローバル変数の VI で複数のグローバル変数を定義することができるため、ノードからどのグローバル変数にアクセスするかを選択する必要があります。次の図に示すように、ノードをポップアップし、**項目選択**メニューから名前で項目を選択してグローバル変数を選択します。あるいは、操作ツールでノードをクリックしたのち、項目を選択します。



グローバル変数は、書き込むことも読み取ることも可能です。グローバル変数に書き込むことはグローバル変数の値が変わることを意味し、グローバル変数から読み取ることはグローバル変数をデータソースとして使用することを意味します。グローバル変数の書き込みまたは読み取りが必要な場合は、グローバル変数のポップアップメニューから**書き込みグローバルに変更**または**読み取りグローバルに変更**項目を選択します。

グローバル変数は、メモリ中のVIを使用して書き込んだり読み取ったりすることができます。その場合、グローバル変数が意図に反して変更されないように、読み取るVIや書き込むVIがアプリケーションのどこに位置しているかを把握しておくことが重要です。Gプログラムは、同時に多数の動作が進行する場合もあるため、さまざまな並行ダイアグラムからグローバル変数にいつアクセスするかを把握することは必ずしも容易なことではありません。グローバル変数のような共有のリソースに対するアクセスの競合をともなう問題は、マルチスレッドのソフトウェアではより顕著になります。正当な理由でグローバル変数を使用するユーザも多数存在しますが、無差別な使用は容易にデバッグできない状況を招くおそれがあるため、使用には注意が必要です。

グローバル変数のVIを保存したあとは、**関数→VIを選択...**を使用してそのグローバル変数を他のVIに配置することができます。グローバル変数のVIをファイルダイアログボックスから選択すると、ダイアグラム上にグローバル変数ノードが配置されます。また、グローバル変数に対しては、階層ウィンドウから複製、コピーと貼り付け、あるいはドラッグとコピーの操作を実行することもできます。

ローカル変数

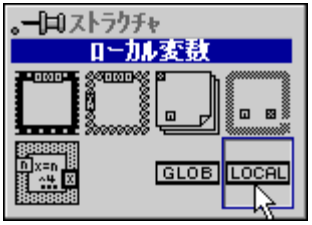
ローカル変数を使用すると、VIのフロントパネル上の1つの制御器または表示器への書き込みや読み取りが可能になります。ローカル変数への書き込みは、結果的には端子にデータを渡すのと同じですが、相手が制御器であってもデータを渡すことができ、相手が表示器であってもデータを読み取ることができます。また、特定のフロントパネル制御器に対し、書き込みモードの変数や読み込みモードの変数を取り混ぜた任意の数のローカル変数のリファレンスを使用することができます。

実質的には、ローカル変数のリファレンスを使用することによってフロントパネル制御器を入力および出力の両方として使用することが可能になります。

☞ **注　ローカル変数に対しては、フロントパネルオブジェクトと関連のあるラベルを選択する必要があります。フロントパネルオブジェクトと無関係のローカル変数は、正しく機能しません。**

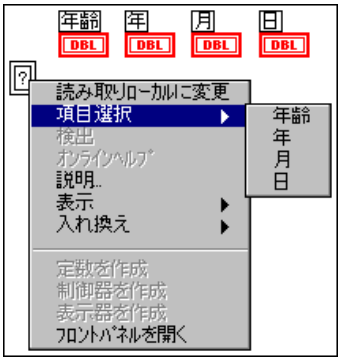
ローカル変数
アイコン

ローカル変数を作成する最も簡単な方法は、フロントパネル制御器または端子をポップアップして**作成→ローカル変数**を選択する方法です。ローカル変数は、自動的にダイアグラム上に表示されます。ローカル変数を作成するもう1つの方法として、次に示すように**関数→ストラクチャ**からローカル変数を選択する方法もあります。

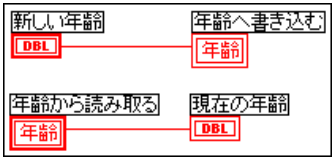


ローカル変数
ノード

グローバル変数とよく似たノードが表示されます。ノードをポップアップするか、または操作ツールでクリックすると、読み取りたい制御器あるいは設定したい制御器を、次の図に示すように最上層のフロントパネル制御器のリストから選択することができます。また、**書き込みローカルに変更**または**読み取りローカルに変更**項目を選択することにより、制御器に書き込むのか制御器から読み取るのかを指定することができます。



次の図は、それぞれが読み取りまたは書き込みという異なる意味を持つ複数のローカル変数が、どのように 1 つの制御器にアクセスするかを示しています。このダイアグラムでは、age という変数を一回は書き込みに、もう一回は読み取りに使用しています。



前の例で示すように、ローカル変数やグローバル変数は思い通りの結果が得られるような順番になるよう注意してください。順序を正しく設定しなかった場合は、"Write" to age が "Read" from age より前に発生するという保証はありません。

第IV部

上級トピック

第IV部は、仮想計測器の作成に使用されるGの上級機能およびテクニックに関する内容から構成されています。

第IV部 上級トピックの各章の内容は下記の通りです。

- 「第24章 カスタム制御器とType Def」では、カスタム制御器とType Defについて説明します。
- 「第25章 他の言語で書かれたコードを呼び出す」では、他の言語で書かれたコードのさまざまな呼び出し方法について説明します。
- 「第26章 Gの実行システムについて」では、VIのマルチタスク処理と実行について説明します。
- 「第27章 アプリケーションを管理する」では、Gアプリケーションのファイルの管理方法について説明します。
- 「第28章 パフォーマンスについて」は、3つのセクションに分かれています。第1のセクションでは、VIの実行時間に関するデータを表示し、シングルスレッド、マルチスレッド、およびマルチプロセッサのアプリケーションをモニタするパフォーマンスプロフィールについて説明します。第2のセクションでは、実行時の速度に影響を及ぼす要素について説明します。第3のセクションでは、メモリの使用に影響を及ぼす要素について説明します。
- 「第29章 移植性およびローカル化について」では、プラットフォーム間でのVIの移植およびVIのローカル化に関する問題について説明します。

24

カスタム制御器とType Def

この章では、カスタム制御器とType Defについて説明します。

フロントパネルの制御器や表示器は、アプリケーションにより適した形にカスタマイズすることができます。たとえば、スイッチがオンのときに閉鎖値を表示し、オフのときに開放値を表示するブールスイッチや、スケールが左ではなく右にあるスライド制御器、テキスト項目や画像項目があらかじめ定義されているリング制御器などを作成することができます。

カスタマイズした制御器あるいは表示器は、VIと同じようにディレクトリやVIライブラリに保存することができます。保存した制御器は、他のフロントパネルでも使用することができます。また、カスタム制御器のアイコンを作成し、その制御器の名前やアイコンを**制御器**パレットに表示することもできます。

VIのさまざまな場所で同じ制御器を必要とする場合は、その制御器のType Def（タイプ定義）と呼ばれるマスターコピーを作成することができます。このType Defを修正すると、そのType Defを使用しているすべてのVIが自動的に修正されます。

カスタム制御器は、ブロックダイアグラム上でも使用することができます。その場合は、その制御器と同じデータタイプを持つ定数が作成されます。Type Defをブロックダイアグラム上で使用した場合、Type Defを修正すると生成される定数が自動的に更新されます。

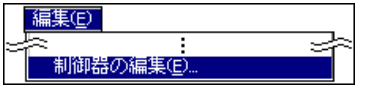
以下の項では、制御器のこのようなカスタマイズの方法について説明します。

カスタム制御器

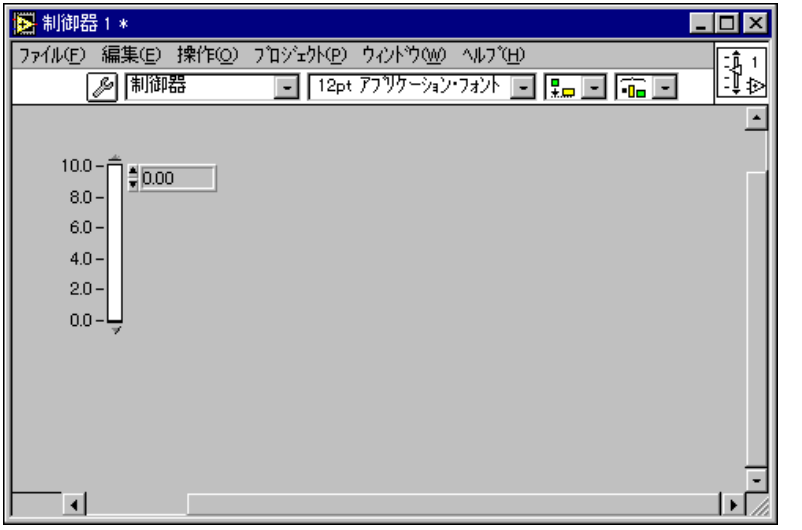
制御器エディタを使用する

制御器のカスタマイズは、必ず編集モードで行います。フロントパネルに、作成したい制御器に最もよく似た制御器を配置します。たとえば、スケールが右側に付いたスライドを作成したいときは、まず最初に縦のスライドをフロントパネルに配置します。

位置決めツールでスライド制御器を選択したのち、さらに**編集→制御器の編集...**を選択します。このオプションは、制御器を選択した場合にのみ選択できます。フロントパネルでは、一回に1つの制御器しか編集できません。



制御器のコピーを表示するウィンドウが開きます。このウィンドウ（次の図参照）は、制御器エディタと呼ばれ、ウィンドウのタイトルは最初は制御器*N*と表示されます。このタイトルは制御器エディタのウィンドウに割り当てられた名前で、制御器を保存して永久名を割り当てるまではこの名前が使用されます。



制御器エディタのウィンドウはフロントパネルと同じような形をしていますが、1つの制御器の編集や保存にのみ使用されるもので、ダイアグラムがないため、実行することはできません。


編集モード


カスタマイズモード

制御器エディタには編集モードとカスタマイズモードがあり、現在のモードはツールバーに表示されるボタン（左記参照）で示されます。制御器エディタを最初に開いたときは、編集モードになります。編集モードでは、編集モードのフロントパネルで作業する場合と同じように、制御器のサイズや色を変更したり、ポップアップメニューからオプションを選択したりすることができます。カスタマイズモードでは、制御器の部品に個別に変更を加えることができます。カスタマイズモードについては、この章で後ほど詳しく説明します。

制御器の編集が終了すると、制御器エディタを開いたときに作業していたフロントパネル上でその制御器をもとの制御器と差し替えることができます。また、別の制御器で使用するために保存しておくこともできます。

カスタム制御器から変更を適用する

もとのフロントパネル制御器を新しいカスタム制御器に差し替える準備ができたら、制御器エディタのメインメニューから**ファイル→変更を適用する**を選択します。

カスタム制御器をもとのフロントパネルだけで使用する場合は、制御器を保存せずに制御器エディタを終了することができます。もとのVIは、必ずカスタム制御器に差し替えた上で保存するようにしてください。このカスタム制御器を将来フロントパネルで使用したい場合は、本章の「カスタム制御器を保存する」の項で説明する方法で保存する必要があります。

変更を適用するオプションは、制御器を修正した後でのみ使用できます。更新すべき制御器が存在しない場合は、**変更を適用する**は使用できません。これは、もとの制御器を削除または差し替えた場合や、もとのフロントパネルを閉じた場合、あるいは先に保存したカスタム制御器を**ファイル→開く**で開いた場合などです。

有効なカスタム制御器

not-OK ボタン

制御器エディタに制御器が2つ以上ある場合は、not-OK ボタンが表示されます。有効なカスタム制御器は 1 つでなければなりません。ただし、これは他の制御器のクラスタでもかまいません。not-OK ボタンは、制御器をクラスタや配列に出し入れすると一時的に表示されます。エラーの説明を見たいときは、not-OK ボタンをクリックします。制御器エディタに制御器が2つ以上ある場合は、エラーメッセージは次のようになります。

フロントパネル上にカスタム制御器に属さない余分な制御器があります。

カスタム制御器を保存する

カスタム制御器を他のフロントパネルで使用したいときは、制御器エディタのメインメニューから**ファイル→別名で保存...**を選択します。制御器は、VI を保存する場合と同様、ディレクトリまたは VI ライブラリに保存します。ディレクトリや VI ライブラリには、制御器または VI、あるいはその両方が格納されます。

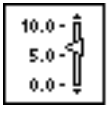
制御器に対する修正を保存せずに制御器エディタを終了すると、制御器を保存するかどうかを尋ねるダイアログボックスが表示されます。

カスタム制御器を使用する

カスタム制御器を保存したら、VI のフロントパネルから**制御器→制御器を選択...**を選択してフロントパネル上でその制御器を使用することができます。また、VI のブロックダイアグラムで**関数→ VI を選択 ...** を選択してブロックダイアグラム上で使用することもできます。カスタム制御器をブロックダイアグラム上で使用する場合は、カスタム制御器と同じデータタイプの定数を作成します。

制御器パレットへのカスタム制御器の追加についての詳細は、「第7章　環境をカスタマイズする」の「制御器パレットおよび関数パレットをカスタマイズする」の項を参照してください。

アイコンを作成する



制御器アイコン

制御器をあとで**制御器**パレットに追加する場合、あるいは制御器が Type Def である場合は、制御器を保存する前に制御器を表すアイコンを作成します。制御器エディタウィンドウの右上にあるアイコンボックスをダブルクリックするか、またはポップアップして**アイコンの編集 ...** を選択して制御器のアイコンを作成します。このアイコンは、**制御器**パレットでこの制御器を表し、制御器が Type Def である場合は階層ウィンドウで制御器を表します。詳しくは、本章の「Type Def」の項、および「第3章　サブ VI を使用する」の「階層ウィンドウを使用する」の項を参照してください。

カスタム制御器の独立性

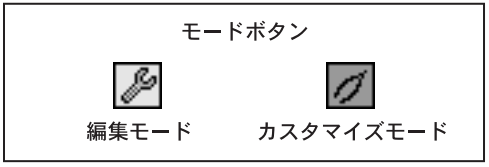
保存したどのカスタム制御器も、**ファイル→開く**により開くことができます。カスタム制御器は、必ず制御器エディタウィンドウ内で開きます。

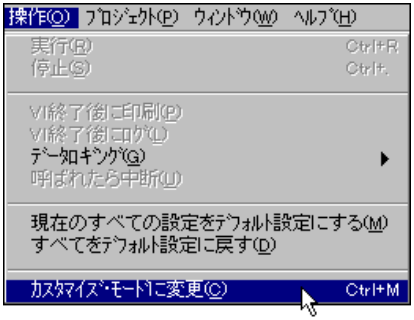
開いたカスタム制御器に変更を加えても、その制御器を使用しているVIには影響を及ぼしません。フロントパネル上でカスタム制御器を使用する場合、カスタム制御器のそのコピーはそのカスタム制御器が保存されているファイルやVIライブラリにはリンクされません。制御器の個々のコピーはそれぞれ独立したコピーです。

ただし、さまざまなVIのフロントパネルやブロックダイアグラム上のカスタム制御器と、その制御器のマスターコピーとの間にはリンクを作成することができます。リンクを作成するためには、カスタム制御器をType Defまたは厳密Type Defとして保存する必要があります。それにより、制御器のマスターコピーに対する変更を、その制御器を使用しているすべてのVIに反映させることができます。詳しくは、本章の「Type Def」の項を参照してください。

カスタマイズモードオプション

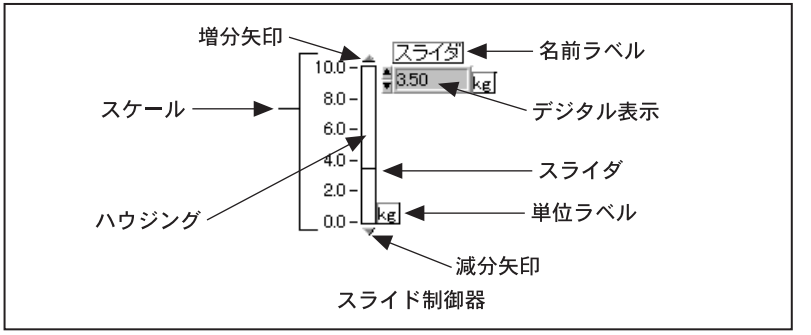
カスタマイズモードの制御器エディタでは、制御器をさらに幅広く変更することができいます。編集モードとカスタマイズモードを切り替えるには、次の図に示すように、制御器エディタのツールバーのモードボタンをクリックするか、または制御器エディタをアクティブウィンドウにして**操作**メニューから**カスタマイズモードに変更**または**編集モードに変更**を選択します。





独立した部品

どの制御器も、さらに小さい部品で構成されます。たとえばスライド制御器は、スケール、ハウジング、スライダ、増分矢印、減分矢印、デジタル表示、ネームラベルで構成されます。スライドの各部品を次の図に示します。



スライド制御器

制御器エディタをカスタマイズモードに切り替えると、制御器の各部品は独立した部品になります。個々の部品は、他の部品に影響を及ぼすことなく変更することができます。たとえば、位置決めツールでスライドのスケールをクリックし、ドラッグすると、スケールだけが移動します。ツールバーの**整列**リングや**間隔**リングを使用して、部品を選択して並べたり、分散させたり、あるいはツールバーの**再順序**リングを使用して部品の層の順番を入れ替えたりすることができます。カスタマイズモードでは、編集モードでは表示されない名前ラベルやデジタル制御器の基数などの部品も含め、制御器のすべての部品が表示されます。

制御器の部品はそれぞれ独立しているため、カスタマイズモードでは制御器の値を操作したり変更したりすることはできません。操作ツールは使用不能になっている点に注意してください。また、ブロックダイアグラムを使用したり、制御器をコネクタペーンに接続したりすることはないため、制御器エディタでは配線ツールも常に使用不能になります。

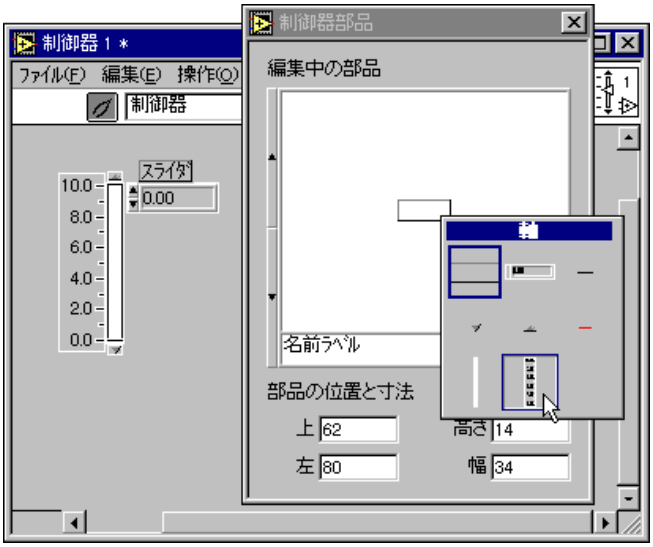
制御器エディタの部品ウィンドウ

制御器の部品のサイズや位置を確認したいときは、**ウインドウ→部品ウインドウを表示**を選択します。制御器の部品を識別するための浮動ウィンドウが開き、各部品の正確な位置とサイズが表示されます。部品ウィンドウの**現在の部品**の表示部には、制御器エディタウィンドウで現在選択されている部品の画像と名前が表示されます。現在の部品の表示部をクリックすると、すべての部品のメニューを表示することができます。

また、現在の部品の表示部の増分矢印または減分矢印をクリックすると、制御器の別の部品をスクロールで表示させることができます。現在の部品

の表示部に表示される部品を変更すると、制御器エディタウィンドウの制御器でその部品が選択されます。制御器エディタウィンドウで制御器の別の部品を選択、変更、またはポップアップすると、現在の部品の表示部に表示される部品も変更されます。

次の図は、左が制御器エディタウィンドウ、右が制御器部品ウィンドウを示したものです。現在の部品はスライドの名前ラベルで、制御器エディタウィンドウでもこの部品が選択されています。制御器部品ウィンドウには、現在の部品の表示部をクリックしたときに表示される部品のメニューが表示されます。ここに示した例は、名前ラベルを現在の部品として表示していますが、これからスケール部品に切り替えようとしているところを示しています。



制御器部品ウィンドウは、現在の部品の正確な位置とサイズを表示します。これらの値は、ピクセル単位の値です。制御器エディタで部品の位置やサイズを変更すると、部品ウィンドウに表示される位置やサイズが更新されます。また、制御器部品ウィンドウに位置やサイズの値を直接入力して制御器エディタウィンドウにおける部品の位置やサイズを変更することもできます。2つの部品を同じサイズにしたり、1つの部品を別の部品と並べて配置したいときには、この方法を使用すると便利です。前の図では部品ウィンドウにスライドの名前ラベルの位置とサイズが表示されていますが、ラベルの左上角のピクセル座標は（74,84）で、ラベルの高さは15ピクセル、幅は64ピクセルになっています。

別のウィンドウに切り替えると、制御器部品ウィンドウは画面から消えます。制御器エディタに戻ると、再度制御器部品ウィンドウが表示されます。

さまざまな部品のカスタマイズモードのポップアップメニュー

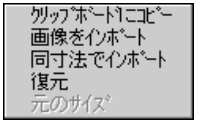
カスタマイズモードでは、制御器のポップアップメニュー全体が個々の部品のポップアップメニューに置き換えられます。部品をポップアップすると、編集モードで使用できるオプションと、カスタマイズモードでしか使用できないオプションが表示されます。ポップアップメニューは部品によって異なります。カスタマイズできる部品には3つ基本のタイプがあります。

- 装飾部品。最も一般的な部品で、スライドのハウジング、スライダ、増分矢印、減分矢印などがあります。装飾部品は画像を表示します。
- テキスト部品。スライドの名前ラベルなどがあります。テキスト部品は、背景の画像（通常は単純な長方形）とテキストで構成されます。
- 部品としての制御器。スライドのデジタル表示に使用される数値制御器などがあります。つまみ、メーター、チャートもデジタル表示に数値制御器を使用します。制御器の中には、さらに複雑なものもあります。たとえば、グラフはクラスタの配列をカーソル表示部品として使用します。

以下の項では、さまざまな部品とそのポップアップメニューオプションについてさらに詳しく説明します。

装飾部品

装飾部品には、ユーザとの対話や表示の手段はありません。次の図は、スライドのハウジングなどの装飾部品のポップアップメニューを示したものです。装飾部品をポップアップするためには、カスタマイズモードになっていなければなりません。また、制御器部品ウィンドウの部品の画像ではなく、部品そのものをポップアップする必要があります。



以下で、ポップアップメニューのオプションについて説明します。

- **クリップボードにコピー** — 部品の画像をクリップボードにコピーします。スライドのハウジングに対して**クリップボードにコピー**を選択すると、縦に細長い挿入長方形の画像がクリップボードにコピーされます。このクリップボードの画像は、任意のフロントパネルに貼り付けたり、**画像をインポート**を使用して別の部品に画像としてインポートすることができます。これらの画像は、**制御器**パレットの**装飾体**と同じです。

ハウジングの長方形のような単純な図形が他の部品で必要な場合、ペイントプログラムでこれらの図形を作成するかわりに他の部品からコピーした画像を使用する方がいくつかの点で便利です。サイズを変更する際には、既存の部品からコピーした画像や装飾の方がペイントプログラムで作成した画像よりもきれいに見えます。たとえば、ペイントプログラムで描画した長方形は、均一にしか拡大できないため、面積を拡大すると枠も太くなります。一方、スライドのハウジングのように部品からコピーした長方形は、サイズを拡大しても枠の太さは変わりません。

もう1つの利点は、組み込み部品がカラーモニタでも白黒モニタでも基本的には同じように表示される点です。

さらに、部品からコピーした画像や装飾には、色付けツールを使用して色を付けることができます。別のソースからインポートした画像では、色もその画像を定義する1つの要素であるため、インポートしたときの色を変えることはできません。

- **画像をインポート** — 装飾部品の現在の画像をクリップボードに入っている画像に差し替えます。このオプションは、制御器の外観を個別にカスタマイズしたいときに使用します。たとえば、ブールスイッチに対して開閉バルブの画像をインポートすることができます。クリップボードに現在の画像が入っていないときは、**画像をインポート**は使用できません。

  制御器エディタには、ブール制御器に画像をインポートするためのショートカットがあります。編集モードのときには、ブールのポップアップメニューから**画像をインポート**→True または**画像をインポート**→False を選択することができます。それにより、画像は通常の状態、および対応する遷移状態の両方にインポートされます。遷移のステータスについて詳細は、本章の「複数の画像を持つ装飾部品」の項を参照してください。

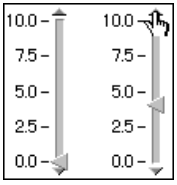
  また、カスタマイズモードでは、遷移のステータスに対して異なる画像をインポートすることができます。これを行うためには、ボタンをポップアップしたのち、**画像項目**を使用して3つめの画像に変更します。クリップボードにTrue→Falseの画像を入れた状態で、再度ポップアップして画像インポートを選択します。4つめ（False→True）の画像に対してもこの操作を繰り返します

- **同寸法でインポート** — 部品の元のサイズを変えずに、クリップボードの画像を縮小または拡大して現在の画像を差し替えます。クリップボードに現在の画像が入っていないときは、**同寸法でインポート**は使用できません。

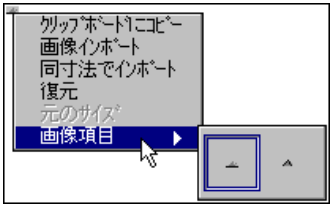
- **復元** — 部品を元の形に戻します。**復元**では部品の位置は変更しません。フロントパネルから**制御器を編集**を選択して制御器エディタを開いている場合は、部品は最初にフロントパネル上にあったときの形に戻ります。**ファイル→開く** ... を選択して制御器エディタを開いた場合は、**復元**は使用できません。
- **元のサイズ** — 部品の画像を元のサイズに戻します。この機能は、他のアプリケーションからインポートした後でサイズを変更した画像に対して使用します。これらの画像の中には、サイズを変更する元の状態より劣って見えるものもあり、そのような場合は元のサイズに戻すことが必要になる場合があります。画像をインポートしていない場合は、**元のサイズ**は使用できません。

複数の画像を持つ装飾部品

装飾部品の中には、それぞれ異なる状況で表示される複数の画像を持つものもあります。このような画像はいずれもサイズが同じで、同じ色を使用します。たとえば、スライドの増分矢印は三角形ですが、通常は背景から若干浮き上がって見える形で表示されます。一方、スライドの値を増分するために操作ツールでクリックする際には、この三角形がへこんだ形で表示されます。次の図に、増分矢印の操作時の2つの画像を表示します。



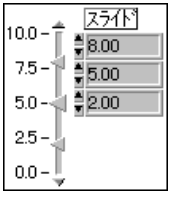
次の図に示すように、複数の画像を持つ装飾部品のポップアップメニューには、**画像項目**というオプションがあります。



画像項目は、その装飾部品の画像をすべて表示します。現在表示されている項目は、太い枠で囲まれます。画像をインポートすると、現在の画像のみが変更されます。他の画像に対して画像をインポートする場合は、まずその画像を選択してから新しい画像をインポートします。

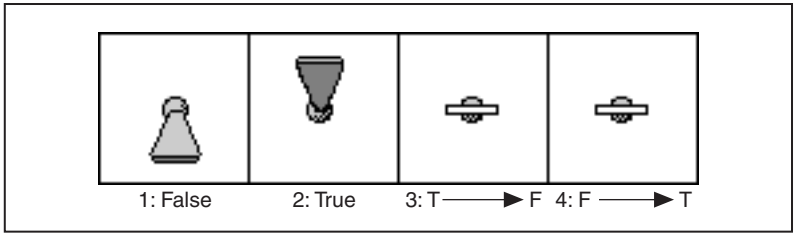
独自の画像を持つ装飾部品

装飾部品で複数の画像を使用する場合、それぞれの画像に異なる色やサイズを使用することができます。たとえばスライドでは、複数値スライドのどのスライダがアクティブであるかを示すためにサイズの異なる2つの画像を使用します。次の例では、スライドの真ん中のスライダがアクティブであることを大きな三角形を用いて示しています。



ブールスイッチも複数の画像を使用します。それぞれの画像には、異なる色やサイズを使用することができます。1つのブールスイッチには異なる4つの画像があります。第1の画像はFALSEの状態を示し、第2の画像はTRUEの状態を示します。第3と第4の画像は、ブール制御器の機械的動作を**放されたらスイッチ**または**放されたらラッチ**のいずれかに設定するために使用します。

マウスボタンを放すまでは、これらの機械的動作によってブール値が変更されることはありません。ボタンをクリックしてからボタンを放すまでの間は、ブールは第3または第4の画像を遷移状態として表示します。第3の画像はTRUEからFALSEへの遷移状態を示し、第4の画像はFALSEからTRUEへの遷移状態を示します。次の図に示したトグルスイッチでは第3と第4の画像は同じになっていますが、必ずしもすべてのケースでこのようになるとは限りません。

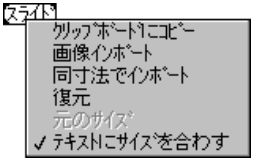


装飾部品で複数の画像を使用できる場合、その部品のポップアップメニューには次の図に示すように**独自のサイズ**というオプションがあります。

独自のサイズは、カスタマイズモードでのみ使用できるオプションで、装飾部品の複数の画像のうち 1 つの画像だけを移動したりサイズを変更したいときに使用します。このオプションは通常はチェックされないため、装飾部品の現在の画像を移動すると他の画像も同じ距離だけ移動し、サイズを変更すると他の画像も同じ割合でサイズが変更されます。

テキスト部品

テキスト部品は、テキストの付いた画像です。名前ラベルのようなテキスト部品のポップアップメニューには、装飾部品のポップアップメニューと同じオプションがいくつかあります。それ以外のオプションは、フロントパネルの編集モードのテキストのポップアップメニューと同じです。テキスト部品のポップアップメニューの例を次の図に示します。



部品ウィンドウには、テキスト部品のテキストではなく背景の画像だけが表示されます。カスタマイズできるのは、テキストではなく背景の画像です。

部品としての制御器

制御器は、別の制御器を部品として使用することができます。その一般的な例は、スライド、ノブ、メーター、チャートのデジタル表示です。デジタル制御器が別の制御器の一部として使用される点を除いては、デジタル表示と通常のフロントパネルのデジタル制御器との間に違いはありません。

デジタル表示もいくつかの部品で構成されています。制御器エディタでもとの制御器を編集する場合、デジタル表示は 1 つの部品として機能するた

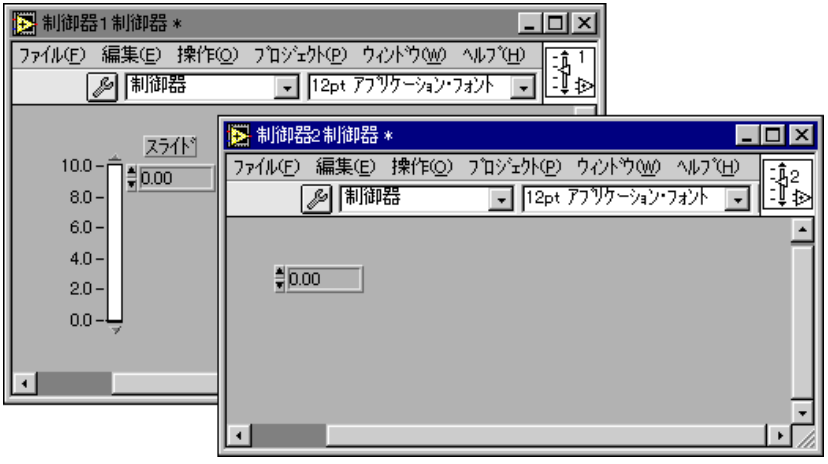
め、デジタル表示の部品を個別に変更したり移動したりすることはできません。ただし、デジタル表示に対して制御器エディタを開き、そこでデジタル表示をカスタマイズすることは可能です。

別の制御器の部品としての制御器をカスタマイズするためには、その制御器に対して制御器エディタを開きます。編集モードでメインの制御器から部品を個別に選択できる場合は、もとのフロントパネルから部品に対して直接制御器エディタウィンドウを開くことができます。たとえば、デジタル表示はスライド制御器から個別に選択することができます。制御器エディタウィンドウを開いたら、**編集→制御器を編集...**を選択することができます。

編集モードにある場合は、いつでもメインの制御器の制御器エディタウィンドウから部品に対して制御器エディタウィンドウを開くことができます。制御器エディタで部品を選択し、さらに**編集→制御器を編集...**を選択します。このように、制御器エディタは階層的に開くことができますが、大部分の制御器では最上位においてのみ他の制御器を部品として使用します。ただし、複雑な制御器を部品として使用し、さらにそれらの制御器が他の制御器を部品として使用するグラフは例外です。

現在カスタマイズしているメインの制御器に対して2つめの制御器エディタを開くことはできません。

次の図は、左がスライドの制御器エディタ、右がデジタル表示の制御器エディタを示したものです。編集モードで制御器を選択できる場合は、制御器エディタウィンドウを階層的に開く際に必ずしもカスタマイズモードになっている必要はありません。



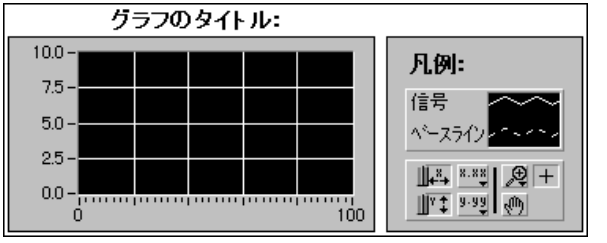
カスタム制御器への装飾部品の追加

制御器エディタでカスタム制御器を作成する際には、装飾部品やテキスト部品を追加して制御器をさらに幅広く変更することができます。

クリップボードから画像あるいはテキストを貼り付けたら、ラベリングツールを使用してラベルを作成するか、または**制御器→装飾体**から画像を選択します。すると、その画像あるいはテキストが制御器の一部となり、制御器をフロントパネル上に配置する際に制御器とともに表示されます。この操作は、制御器エディタの編集モードでもカスタマイズモードでも行うことができます。新しい部品は、他の部品と同じように移動したり、サイズを変更したり、あるいは層の順序を変更することができます。追加した部品は、カスタマイズモードの制御器部品ウィンドウに装飾部品として表示されます。

制御器エディタでは、装飾部品を削除することもできます。

次の図は、**グラフのタイトル**ラベル、**凡例**ラベル、および凡例部品を囲むボックスを装飾部品として使用したカスタムグラフを示したものです。



別のフロントパネルでカスタム制御器を使用する場合、追加した装飾部品のサイズを変更することはできますが、装飾部品を移動することはできません。

カスタム制御器に関する注意事項

カスタム制御器の作成に際しては、いくつかの点に注意する必要があります。

- あるプラットフォームで作成した画像は、別のプラットフォームにロードすると多少違って見えることがあります（これは、画像リングにインポートした画像や、フロントパネルの背景として使用する画像にもあてはまります）。たとえば、不規則な形をした画像や透明な背景は、別のプラットフォームでは白ベタの背景として表示される場合があります。「第29章　移植性およびローカル化について」の「画像」の項を参照してください。

- 制御器エディタで変更できるのは制御器の外観だけです。制御器の動作を変更することはできません。これは、2つのことを意味します。
  - 制御器のデータの表示方法を変更することはできません。
  - 制御器を編集する際、特に制御器のサイズを変更する際に、制御器の動作を変更することはできません。

    たとえば、リング制御器の高さを高くすると、増分矢印や減分矢印の高さも高くなります。増分矢印と減分矢印を移動してリングの下に並べて配置した場合、リングの高さを高くするとこれらの矢印の高さも高くなり、奇妙な形になります。
- カスタム制御器は、外観に問題がないように見えても異常な動作をする場合があります。制御器が思い通りの形で表示されても、その不規則な編集動作が好ましくない場合は、次の「Type Def」の項に記載した厳密なタイプ定義に関する説明を読み、編集がどのように制限されるかを理解してください。

Type Def

Type Defは、制御器のマスターコピーです。マスターコピーすなわちType Defは、制御器エディタを使用して作成します。Type Defは、多数のVIで同じ種類の制御器を使用する場合に便利です。制御器をType Defとして保存しておくと、そのType Defをすべての VIで使用することができます。あとでその制御器を変更する際には、その制御器を使用している個々のVIを変更しなくても、1つのType Defファイルの内容を更新するだけで済みます。

一般Type Def：データタイプの一致

Type Defは、制御器のデータタイプを制御器を使用するすべての場所で同じになるように強制します。多くの場所で同じデータタイプの制御器を使用したいとき、およびそのデータタイプを使用しているすべての場所で自動的に変更したいときは、Type Defを使用します。たとえば、倍精度のデジタル制御器からType Defを作成したあと、そのType Defを多くの異なるVIで使用し、さらにそのあとでType Defを16ビット整数の制御器に変更するものとします。Type Defを変更すると、そのType Defを使用しているすべてのVIを自動的に更新することができます。この場合、自動更新されないように特に指定したVIだけが自動更新の対象外となります。詳しくは、本章の「Type Defを使用する」の項を参照してください。

また、2つの整数と文字列からなるクラスタのようなType Defも使用することができます。このType Defを2つの整数と2つの文字列からなるクラスタに変更すると、このクラスタを使用しているすべての場所でType Defを更新することができます。

データタイプがマスターコピーと一致する限り、Type Def は異なる名前、説明、デフォルト値、サイズ、色、さらに異なるスタイルの制御器（スライドのかわりとしてのノブ）を使用することもできます。

厳密 Type Def：すべてのことがらの一致

Type Def は、制御器を使用しているすべての場所で、制御器のデータタイプだけでなくそのサイズ、色、外観も含め、制御器に関するほとんど**すべてについて**同じになるように強制します。これを、**厳密 Type Def** といいます。

制御器に関するデータのうち、厳密 Type Def のマスターコピーと違っていてもかまわないのは名前、説明、およびデフォルト値に限られます。ここで、1 つの例として、赤いフレームを持つ倍精度のデジタル制御器の厳密 Type Def を作成するものと仮定します。その場合、一般 Type Def と同様、厳密 Type Def を整数に変更するとこの Type Def を使用しているすべての VI が自動的に更新されます。ただし、一般 Type Def とは異なり、たとえば赤いフレームを青に変えるなどしてType Defにそれ以外の変更を加えた場合にも、その Type Def を使用しているすべての VI の更新が必要となります。また、厳密 Type Def を使用している VI の自動更新を禁止することはできません。

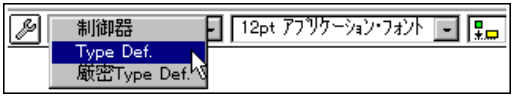
☞ **注**　**属性ノードの多くは、厳密 Type Def には使用できません。厳密 Type Def に使用できる属性は、Visible、disabled、Key Focus、Blinking、Position、Bounds など、制御器の外観に関するものに限られます。**

ブロックダイアグラムにおける Type Def

Type Def をブロックダイアグラム上で使用する場合、その外観は制御器や表示器ではなく定数になります。したがって、厳密 Type Def のコピーはブロックダイアグラム上では一般 Type Def のコピーと同じように動作します。すなわち、厳密 Type Def のデータタイプが変更された場合にのみ自動更新されます。

Type Def を作成する

Type Def は、次の図に示すように、制御器エディタウィンドウのツールバーでリングを設定することによって作成します。思い通りに制御器を設定し、制御器エディタウィンドウで**ファイル→保存**を選択します。



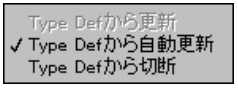
保存したType Defは、いずれも**ファイル→開く ...** を選択することによって開くことができます。Type Defは、常に制御器エディタウィンドウに表示されます。Type Defに対する変更は、そのType Defを使用しているすべてのVIに適用されます。

Type Defを使用する

一般Type Defや厳密Type Defは、カスタム制御器と同様、VIのフロントパネルまたはブロックダイアグラム上に配置します。フロントパネルやブロックダイアグラム上のType Defは、他の制御器や定数と同じように編集あるいは操作することができます。

☞ **注　フロントパネル上の厳密Type Defは、名前、説明、デフォルト値の変更以外の編集はできません。**

次の図に示すように、ポップアップメニューのType Defオプションを見ると制御器がType Defであるかどうかがわかります。厳密Type Defは、フロントパネルやブロックダイアグラムで編集できないこと、およびポップアップメニューでほとんどのオプションが欠けていることで識別できます。



VIは、フロントパネルまたはブロックダイアグラムで使用するそれぞれのType Defについて、そのType Defが保存されているファイルやVIライブラリとの接続を維持します。この接続は、フロントパネルまたはブロックダイアグラムにType Defを配置したのちそのType Defを選択し、さらに**編集→制御器を編集 ...** を選択したときの動作によって確認できます。開かれる制御器エディタには保存したType Defが表示され、その名前も通常の制御器*N*ではなく指定した名前になります。

Type Defを更新する

Gの開発環境は、Type Defを使用するすべての場所でデータタイプが同じになるようにします。また、厳密Type Defに関するすべての点について、それを使用するすべてのフロントパネルで同じになるようにします。不正な一般Type Defや厳密Type Defは、それがフロントパネル上にある場合はファイルまたはVIライブラリに保存されているのと同じコピーに差し替え、それがブロックダイアグラム上にある場合はファイルまたはVIに保存されている制御器と一致するデータタイプの定数と差し替えることにより、自動的に修正することができます。

フロントパネル上でType Defのコピーの色やサイズを変更する際には、この自動更新機能を使用しない方が望ましい場合もあります（変更をそのフロントパネルに限定したい場合など）。そのような場合は、フロントパネル上でType Defをポップアップして**Type Defから自動更新**オプションをオフにすることができます。必要に応じてこのType Defを自動更新する代わりに、壊れた矢印の実行ボタンが表示され、フロントパネル上のType Defは使用できなくになります。Type Defは、ポップアップメニューから**Type Defから更新**を選択するか、またはデータタイプをType Defと一致するように変更して修正するまで実行することはできません。厳密Type Defは常に自動更新されるため、厳密Type Defのポップアップメニューには**Type Defから自動更新**オプションはありません。

Type Defを使用する際には、Type Defに異なるデフォルト値を割り当てることができます。ただし、Type Defのデータタイプが変更されると、すべてのデフォルト値がマスターコピーから更新されるため、数値を文字列に変更するような場合には古いデフォルト値を新しいデータタイプに変更することができません。それ以外の場合は、それぞれのデフォルト値が保持されます。

定数を作成、**制御器を作成**、または**表示器を作成**を使用してType Defを配線した場合は、Type Defはマスターコピーから更新されます。**定数を作成**、**制御器を作成**、および**表示器を作成**についての詳細は、「第17章　ブロックダイアグラムの概要」を参照してください。

Type Defを検索する

VIは個々のType Defへの接続を維持する必要があるため、Type Defを使用しているVIを実行するにはそのType Defが保存されているファイルやVIライブラリが存在しなければなりません。VIを開いたときにVIが必要とするType Defが見つからない場合は、VIで使用しているそのType Defのコピーは使用不能になり、壊れた矢印の実行ボタンが表示されます。この問題を解消するためには、正しいType Defを探し出してそれを開くか、または使用不能になっているコピーをポップアップして**Type Defから切断**を選択する必要があります。Type Defからの接続が切断されると、そのコピーはデータタイプに対する制約が解除され、通常の制御器または定数になります。再度この接続を確立するためには、Type Defを探し出して制御器をそのType Defと入れ換える必要があります。

クラスタのType Def

クラスタをType Defまたは厳密Type Defとして使用する場合は、ブロックダイアグラム上でBundle関数やUnbundle関数ではなくBundle By Name関数やUnbundle By Name関数を使用してクラスタの要素にアクセスする方が賢明です。これらの関数は、クラスタ要素を順番ではなく名前で参照するため、Type Defであるクラスタの要素の順番を変更したり要素を追加したりしても参照には影響しません。Bundle By NameまたはUnbundle By Nameで参照する要素を削除した場合は、ブロックダイアグラムを変更する必要があります。これらの関数についての詳細は、「第14章 配列とクラスタの制御器 および表示器」を参照してください。

25

他の言語で書かれたコードを呼び出す

この章では、他の言語で書かれたコードのさまざまな呼び出し方法について説明します。このような方法としては、プラットフォーム固有のプロトコルを使用する方法や、VIへのリンクを目的として書かれたコードを呼び出すコードインタフェースノードを作成し、Call Library関数ノードを使用してWindowsのダイナミックリンクライブラリ（.DLLファイル）、Macintoshのコードフラグメント、UNIXの共有ライブラリを呼び出す方法があります。また、LabWindows/CVI Function Panelコンバータを使用してLabWindows/CVIで書かれた計測器ドライバを変換する方法もあります。

VIから他のアプリケーションを実行する

他のアプリケーションは、VIから実行することができます。ただし、Windows、UNIX、Macintoshではその方法が異なります。

（Windows、UNIX）System Execを使用してVIから他のアプリケーションを実行します。**関数→通信**パレットの単純なSystem Exec VIを使用して、VIからコマンドラインを実行することができます。コマンドラインには、起動しようとしているアプリケーションがサポートしているものであればどんな引数でも使用できます。

TCP/IP（あるいはWindowsではDDE）を介してアプリケーションにアクセスできる場合は、データやコマンドをそのアプリケーションに渡せる可能性があります。アプリケーションの通信機能の詳細については、アプリケーションの参考資料を参照してください。LabVIEWをご使用の場合は、他のアプリケーションへのデータ転送用通信VIの使用方法については、『LabVIEW ユーザマニュアル』の「第20章　通信の概要」、および「第21章　TCPとUDP」も参照してください。

（Macintosh）Apple Event VIを使用してVIから他のアプリケーションを実行します。Apple EventはMacintosh固有のプロトコルで、アプリケーションはこのプロトコルを通して相互に通信します。これらのイベントは、アプリケーション間でのコマンドの送信に使用できます。また、他のアプリケーションを起動するために使用することもできます。Gでの Apple Event VIを使用した他のアプリケーションのさまざまな起動方法および制御方法に関する詳細については、LabVIEWをご使用の場合は『LabVIEW ユーザマニュアル』の「第21章　TCPとUDP」を参照してください。

Call Library関数を使用する

標準のほとんどの共有ライブラリ（Windowsではダイナミックリンクライブラリすなわち DLL、Macintosh ではコードフラグメント、UNIX では共有ライブラリ）は、Call Library関数ノードを使用して呼び出すことができます。Call Library関数ノードには、多数のデータタイプや呼び出しの規格があります。このノードを使用すると、ほとんどの標準ライブラリや固有に作成したライブラリから関数を呼び出すことができます。

Call Library 関数ノードは、呼び出したいノードがすでに存在する場合や、WindowsのDLL、Macintoshのコードフラグメント、あるいはUNIXの共有ライブラリの作成方法をよく知っている場合に最も適しています。ライブラリは、複数の開発環境に共通した標準フォーマットを使用するため、ほとんどすべての開発環境でGからの呼び出しが可能なライブラリを作成することができます。コンパイラで標準の共有ライブラリを作成できるかどうかについては、コンパイラのマニュアルを参照してください。

このノードについての詳細は、この章の「Call Library関数」の項を参照してください。

マルチスレッドのオペレーティングシステムでは、ダイナミックリンクライブラリ（.DLL ファイル）や共有ライブラリに対する複数の呼び出しを同時に実行することができます。デフォルトでは、すべてのCall Libraryノード(旧バージョンのノードも含みます)がユーザインタフェーススレッドで実行されます。Call Library ノードを再入実行可能として構成する際には、呼び出される関数が複数のスレッドで同時に実行できるようにしておく必要があります。詳しくは、「第26章　Gの実行システムについて」の「マルチスレッド処理」の項を参照してください。

コードインタフェースノードを使用する

最高のパフォーマンスが要求されるアプリケーション、あるいはCコードに任意のデータ構造を渡す必要があるアプリケーションに対しては、コードインタフェースノード（CIN）を作成することができます。CIN を使用すると、G言語のVIへのリンクを目的として作成されたコードを呼び出すことができます。

CINは、GからCコードを呼び出すためのごく一般的な方法です。CINとの間で、任意の複雑なデータ構造をやりとりすることができます。また、データ構造はGに保存されているのと同じフォーマットでCINに渡されるため、CIN を使用することによってより高いパフォーマンスが得られる場合もあります。

ただし、このようなレベルのパフォーマンスを確保するためには、CIN を作成できなくてはなりません。それには、Cの開発に関する十分な知識と、

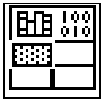
必要なCINを作成するための十分な時間が要求されます。また、CINはGと密接な関係にあるため、使用できるコンパイラも限られます。

コードインタフェースノードの作成方法に関しては、LabVIEWをご使用の場合はソフトウェアのプログラムディスクまたはCDにPDFファイル形式で収録されている『LabVIEW Code Interface Reference Manual』を参照してください。

Call Library関数

Call Library関数を使用すると、16ビットのWindows 3.1 DLL、32ビットのWindows 95/NT DLL、Macintoshのコードフラグメント、あるいはUNIXの共有ライブラリの関数を直接呼び出すことができます。

（Macintosh）Call Library関数は、Macintoshのコードフラグメントマネージャ（CFM）を使用します。これは、あらゆるPowerMacマシンにおいて標準になっています。680x0のMacintoshコンピュータは、CFMという拡張子を使用します。また、680x0のMacintoshコンピュータでは、Call Libraryノードは引数が可変の関数を呼び出すことはできません。Macintoshの共有ライブラリは、他のプラットフォームとは異なる動作をします。1つのファイルには複数のコードフラグメントを書き込むことができ、それぞれのフラグメントに名前を付けることができます。

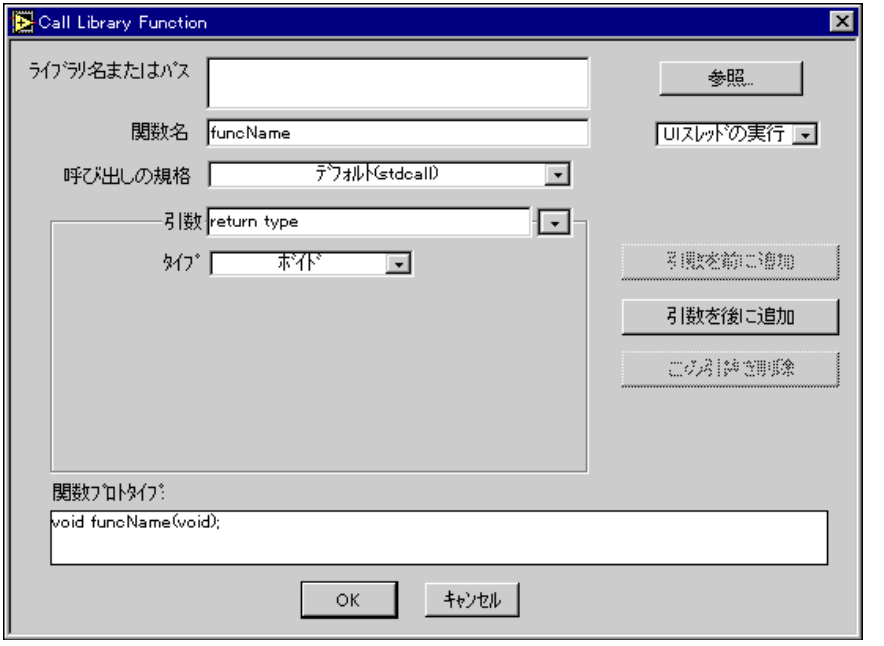


左記のCall Library関数は、**関数→上級**パレットに入っています。

Call Library関数をダブルクリックするか、またはこの関数のノードのポップアップメニューから**設定 ...**を選択すると、ライブラリ、関数、引数、およびノードの戻り値を指定するためのダイアログボックスが表示されます。また、Windowsでの呼び出しに関する規格も表示されます。OKボタンをクリックすると、正しい数の端子がノードに自動的に追加され、各端子が正しいデータタイプに設定されます。

関数の戻り値は、ノードの最も上に位置する1組の端子の右側の端子に返されます。戻り値がない場合は、この1組の端子は使用されません。これより下の端子のそれぞれのペアは、関数引数リスト中の引数と対応しています。値は、左側の端子に書き込むことによって関数に渡されます。引数の値は、関数を呼び出したあとで右側の端子から配線することによって読み取ります。

Call Library Function ダイアログボックスを次の図に示します。



ダイアログボックスで項目を選択すると、選択した関数の C のプロトタイプが下の関数プロトタイプと呼ばれる下部の表示器に表示されます。

☞ **注**　C++ で作成およびコンパイルされた共有ライブラリの呼び出しは、完全にテストされていないため正しく実行できない場合があります。

Windows 3.1 では、Call Library 関数を使用して最大 29 個の 4 バイト引数を渡すことができます。値で渡される倍精度の浮動小数点引数は、サイズが 8 バイトであるため 2 つの引数としてカウントされます。したがって、値で渡される倍精度の浮動小数点引数の数は最大 14 個に制限されます。

Windows 3.1 では、同じ DLL 内にあり、かつ異なる引数を持つ同じ関数を呼び出す Call Library ノードを 2 つ以上メモリに格納することはできません。

Windows 3.1 では、DLL は 16 ビットでなければなりません。Windows 95/NT では、DLL は 32 ビットでなければなりません。Windows 95/NT から 16 ビットの DLL を呼び出すためには、32 ビットの DLL としてコンパイルするか、またはサンク DLL を作成する必要があります。サンク DLL についての詳細は、Microsoft 社のマニュアルを参照してください。

呼び出しに関する規格（Windows）

関数の呼び出しに関する規格は、呼び出しに関する規格のリングで選択します。デフォルトの呼び出しの規格は、Windows 3.1 ではPascal、Windows 95/NT ではStdcall になります。このデフォルト設定は、ほとんどのDLL に使用される呼び出しの規格に対応しています。一方、C の呼び出し規格を使用する方法もあります。どの呼び出し規格が適しているかについては、呼び出すDLL のマニュアルを参照してください。

引数リスト

最初は、Call Library 関数には引数がなく、戻り値はVoid です。関数に引数を追加するためには、**引数を前に追加**または**引数を後に追加**ボタンをクリックします。引数を削除する場合は、**この引数を削除**ボタンをクリックします。

引数リングを使用すると、さまざまな引数や戻り値を選択することができます。選択した引数の名前は、よりわかりやすい名前に変更することができます。引数の名前は呼び出しには影響しませんが、出力ワイヤに転送されます。わかりやすい名前を使用すると、引数の切り替えが容易になります。

それぞれの引数のタイプは、タイプリングを使用して指定します。戻り値のタイプは、関数に戻り値がないことを示すVoid、**数値**、または**文字列**のいずれかに限られます。

引数に対しては、**数値**、**配列**、**文字列**、Adapt to Type、のいずれかを選択できます。

タイプリングで項目を選択すると、そのデータタイプに関する詳細やライブラリ関数へのデータの渡し方を指定するための新たな項目が表示されます。ライブラリによって必要とされるデータタイプが異なるため、Call Library **ノード**にはこれらのデータタイプに対応したさまざまな項目があります。どのデータタイプを使用すべきかについては、呼び出すライブラリのマニュアルを参照してください。

- Void ― このタイプは戻り値に対してのみ使用できます。この項目は、引数に対しては使用できません。戻り値を持たない関数に対しては、このタイプを使用します。
- **数値** ― 数値のデータタイプに対しては、データタイプリングを使用して正確な数値タイプを指定する必要があります。数値タイプには下記の種類があります。
  - 符号付きおよび符号なしの8ビット、16ビット、または32ビット整数
  - 4バイトの単精度の数
  - 8バイトの倍精度の数

拡張精度の数や複素数を使用することはできません。標準ライブラリでは、通常これらの数は使用しません。

また、値または値のポインタのどちらを渡すかについても、形式リングで指定する必要があります。

☞ **注**　Windows 3.1では、戻り値に単精度あるいは倍精度のデータタイプを使用することはできません。DLLが単精度の数や倍精度の数を返すための標準手段は存在しません。浮動小数点の処理方法は、各コンパイラによって異なります。単精度または倍精度の数を返す必要がある場合は、データが戻り値としてではなく引数として返されるようにする必要があります。

- **配列** — 配列のデータタイプ(数値のデータタイプと同じ項目を使用)、次元の数、配列を渡す際に使用する形式を指定することができます。**形式**項目は、配列データのポインタを渡すのか、LabVIEW**配列ハンドル**を渡すのかを指定するために使用します。**配列データポインタ**を選択した場合は、配列の次元を別々の引数として渡します。

☞ **注意**　Windows 3.1では、Call Library**ノード**は配列に対して配列データポインタ形式だけを使用します。また、Windows 3.1では、Huge**ポインタ**を使用してデータが渡されるように指定することもできます。Huge**ポインタ**は、配列で64**キロバイト**を超えるデータを渡す必要がある場合に使用できます。この項目は、呼び出すDLLがデータのポインタとしてHuge**ポインタ**を想定している場合にのみオンにします。通常のポインタを想定している関数にHuge**ポインタ**を渡すと、アプリケーションがクラッシュする恐れがあります。

⚠ **注意**　配列のサイズをreallocなどのシステム関数を使用して変更すること避けてください。システムがクラッシュする恐れがあります。

- **文字列** — 文字列に対しては文字列フォーマットを指定できます。文字列フォーマットにはC、Pascal、Gがあります。

⚠ **注**　文字列フォーマットは、ライブラリ関数が想定する文字列のタイプに基づいて選択してください。ほとんどの標準ライブラリは、Cの文字列（後ろにNull文字が付いた文字列）、またはPascalの文字列（バイト長から始まる文字列）を想定しています。呼び出すライブラリ関数が特にG専用に作成されたものである場合は、LabVIEW文字列ハンドル形式を使用することができます。LabVIEW文字列ハンドルは、バイト長を表す4バイト値とそれに続く文字列データのポインタを示すポインタです。

☞ **注**　Windows 3.1では、LabVIEW文字列ハンドル形式を使用したり、文字列を返したりすることはできません。戻り値のタイプとして文字列をサポートしているプラットフォームでは、文字列は直ちにバッファにコピーされます。この文字列は、割り当てを解除することはできません。

注意　**文字列のサイズをreallocなどのシステム関数を使用して変更することは避けてください。システムがクラッシュするおそれがあります。**

Adapt to Type ― G言語の任意のデータタイプをDLLに渡せるようにします。データタイプは、CIN に渡す場合と同じ方法で渡されます。これは、以下のことを意味します。

- スカラはリファレンスによって渡されます（スカラのポインタがライブラリに渡されます）。
- 配列および文字列は、ハンドル（データのポインタへのポインタ）として渡されます。詳しくは、ソフトウェアのプログラムディスクまたはCDにPDFファイル形式で収録されている『Code Interface Reference Manual』を参照してください。
- クラスタはリファレンスによって渡されます。
- 配列あるいはクラスタのスカラ要素は一列です。たとえば、数値を含むクラスタは、数値を含むデータ構造のポインタとして渡されます。
- 配列内のクラスタは一列です。
- クラスタ内の文字列および配列はハンドルで参照されます。

注　Windows 3.1 では、**Adapt to Type** は使用できません。

他のデータタイプを想定している関数を呼び出す

ときには、G では使用しないデータタイプを想定している関数に遭遇する場合があります。たとえば、Call Library 関数は、数値以外のデータからなる任意クラスタや配列を渡すために使用することはできません。

データタイプによっては、送信したいデータのバイナリイメージを収めた文字列またはバイトの配列を作成することによりデータを渡せる場合もあります。バイナリデータは、データ要素を文字列にタイプキャストし、それらをつなぎ合わせることによって作成することができます。

もう 1 つの方法では、G で使用するデータタイプ、およびライブラリ関数が想定しているデータ構造を構築するための引数を受け付けるライブラリ関数を作成したうえで、そのライブラリ関数を呼び出します。

最後の手段として、上記の代わりにコードインタフェースを作成する方法があります。コードインタフェースノードは任意のデータ構造を受け取ることができますが、G でのデータ受け渡し方法について理解する必要があるため、この方法をマスターするには多少時間がかかります。

LabWindows/CVIの関数パネルコンバータ

☞ **注**　この機能は、Macintoshでは使用できません。

LabWindows/CVIの関数パネルコンバータは、LabWindows/CVIで作成した計測器ドライバの変換プロセスを自動化し、これらのドライバをGで使用できるようにします。LabWindows/CVI計測器ドライバは、Cのソースファイルとヘッダファイル、コンパイルしたコードを格納するダイナミックリンクライブラリ（DLL）**（Windows）**または共有ライブラリ**（UNIX）**、および関数パネル（FP）ファイルと呼ばれるLabWindows/CVI固有のファイルで構成されます。LabWindows/CVIで関数パネルファイルを使用する目的は、ユーザがCの関数の呼び出し文を編集する際に、ポップアップウィンドウで値を入力して引数や戻り値を指定できるようにすることにあります。

LabWindows/CVIの関数パネルコンバータは、CVI FPファイルのデータを使用してタイプ、データの表記法、およびフロントパネル上での制御器の配置を決定し、個々の関数をVIに変換します。生成された各VIは、そのブロックダイアグラムのCall Library関数を使用して正しいドライバライブラリの正しいCルーチンを呼び出します。

多くの計測器ドライバは、ライブラリ関数を呼び出さなくても完全にダイアグラム上で作成することができますが、場合によってはLabWindows/CVIの関数パネルコンバータを使用することでいくつかのメリットが得られます。特定の計測器用のG ドライバがなく、LabWindows/CVIでドライバが存在する場合は、このツールを使用することで比較的容易にそのCVIドライバを使用することができます。ただし、ユーザにとってはGのダイアグラムで構成された真のGドライバの方が見やすく、修正も簡単であること、およびG環境においても複数のタスクを確実に処理できることを考えれば、可能な場合には真のGドライバを選択した方が賢明です。ライブラリの呼び出しは同期であるため、平行して実行しているVIはいずれも呼び出し中は一時中断されます。

☞ **注**　CVIのすべての.DLLファイルは、実行時のサポートを必要とします。変換プロセスで作成されたVIを使用するためには、CVIランタイムエンジンをインストールしてください。

変換プロセス

ファイル→ CVI FP ファイルの変換 ... を選択すると、LabWindows/CVI の関数パネルファイルを選択するためのダイアログボックスが表示されます。FP ファイルを選択すると、次のような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で、新しい VI の保存先、および変換したいドライバ関数を指定します。

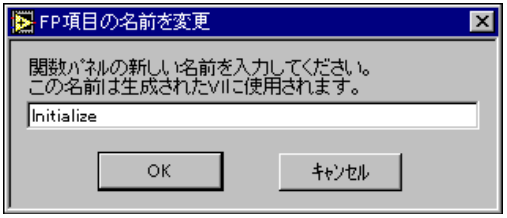
保存先のディレクトリあるいは VI ライブラリは、いちばん上のテキストボックスで指定します。プログラムが提示する保存先が表示されていますが、パスを変更すれば新しい VI を任意の場所に保存することができます。**参照 ...** ボタンを使用すると、ファイルダイアログボックスを使用して通常の方法で保存先を指定することができます。

計測器接頭辞のテキストボックスには、関数パネルファイルが表示する計測器の接頭辞が表示されます。この接頭辞は、CVI が関数パネルツリー中のすべての C 関数の名前の頭に追加します。同様に、コンバータも生成する VI の名前の頭にこの接頭辞を追加します。DLL に基づいてプログラムが提示する接頭辞が表示されていますが、この文字列は必要に応じて変更または削除することができます。

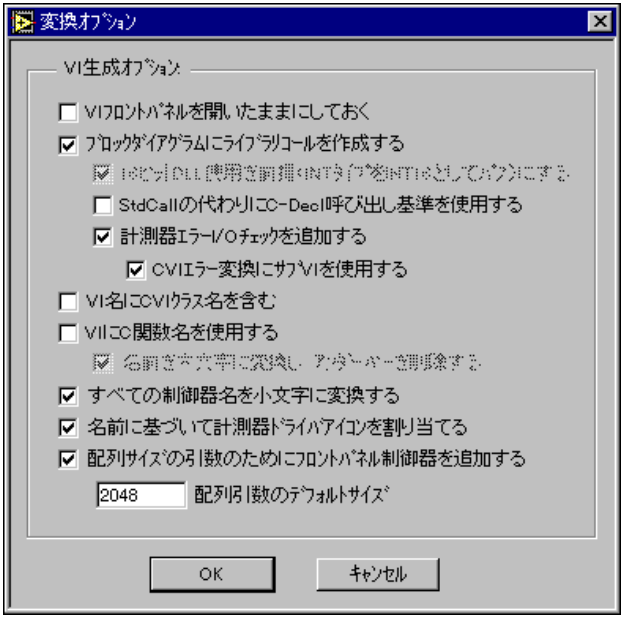
ダイアログボックスのその他の部分は、変換する関数パネル項目の選択に関する部分です。リストボックスには、関数パネルツリーのすべての項目が LabWindows/CVI と同じように、クラスに応じて階層的にインデントをつけて表示されます。クラスの名前も表示されますが、グレーで表示されるため選択することはできません。最初は、関数パネルファイルで検出されたすべてのノードが選択されます。チェックマークはその項目が選択さ

れていることを示します。選択されていない項目にはチェックマークは表示されません。また、長方形はその項目がクラスの名前であることを示します。

項目をダブルクリックすると、選択のステータスが切り替わります。**すべてを選択**ボタンと**すべての選択を解除**ボタンは、選択の補助手段として使用します。また、項目を1回クリックして反転表示のバーをその項目に移動したのち、**名前の変更**をクリックすると、FP項目の名前を変更するための次のようなダイアログボックスが表示されます。



オプション...ボタンをクリックすると、次のようなダイアログボックスが表示されます。



変換オプションは下記の通りです。

VIフロントパネルを開いたままにしておく ― コンバータは変換終了時に個々のVIをディスクに保存したあともVIを破棄せず、フロントパネルを開いたままVIをメモリに残します（デフォルト設定はオフ）。

ブロックダイアグラムにライブラリコールを作成する ― コンバータはCall Library関数ノードを各VIのブロックダイアグラム上に配置し、フロントパネルのすべての端子を正しく配線します。このオプションがオンになっていないときは、フロントパネルだけが作成され、ブロックダイアグラムには何も配置されません（デフォルト設定はオン）。

16ビットDLL使用を前提にする ―**（Windows 3.1）**コンバータはCall Library関数の引数を記述する記述子を作成する際に、LabWindows/CVIの関数パネルから渡された整数タイプをINT32ではなくINT16として処理します。これは、Borlandのような他社のコンパイラで作成したDLLを呼び出す際に必要になります（デフォルト設定はオン）。

計測器エラーI/Oチェックを追加する ― コンバータは作成された各VIのブロックダイアグラム上にエラー処理コードをドロップします。このオプションを選択した場合は、エラー入力とエラー出力のクラスタが、フロントパネル上のLabWindows/CVIの関数パネルに由来する他のすべての制御器および表示器の下にドロップされます。フロントパネルの端子とブロックダイアグラムのCall Library関数はCaseストラクチャに格納され、ストラクチャはエラー入力のステータスフィールドがエラーを示すFALSEの場合にのみ実行されます（デフォルト設定はオン）。

CVIエラー変換にサブVIを使用する ― 各VIのブロックダイアグラムにサブVIをドロップし、LabWindows/CVIスタイルのエラーコードをGeneral Error Handlerに渡せるようにGスタイルのエラークラスタにマッピングします。Call Library関数のノード、エラー入力クラスタ、および現在のVIの名前からの整数の戻り値は、サブVIに渡されます。サブVIによってエラーが検出されると、エラーはステータスをTRUEに設定してエラー出力に渡されます。警告が検出された場合は、警告がエラー出力に渡されますが、ステータスはFALSEに設定されます。それ以外の場合は、エラー入力がエラー出力に渡されます。このオプションが設定されていないときは、サブVIはドロップされず、ブロックダイアグラムはCall Library関数の戻り値が0未満の場合にのみエラー出力でエラーが示されるように作成されます（デフォルト設定はオン）。

VI名にCVIクラス名を含む ― 個々のVIの名前を、各計測器ドライバオプションに対する関数パネルの名前またはC関数の名前に基づいて自動的に作成します。このオプションがチェックされていないときは、自動的に生成された関数の名前の頭に、LabWindows/CVIの関数パネルに関連したクラス名が追加されます。

VIにC関数名を使用する ― 通常は、コンバータは生成するVIの名前を、関数パネル項目の名前に計測器の接頭辞と.viを直接追加することによって作成します。ただし、LabWindows/CVIでは関数パネルツリーの「葉」が固有名であるとは限らないため、この方法ではそれぞれの項目の名前が必ずしも固有名になるとは限りません。このオプションを設定すると、コンバータはVIの名前をFP項目に対応する実際のC関数の名前を使用して作成するため、名前を固有名にすることができます。

名前の重複が原因で発生する問題を避けるため、コンバータは表示する関数パネル項目のリストを作成する際にすべての項目が固有であるかどうかをチェックします。固有でない項目はヌル記号で示され、ダブルクリックしても選択できなくなります。名前の重複が原因で発生する問題を避けるため、コンバータはこのような名前の競合を検出した場合、最初にダイアログボックスを開く際にワンボタンの警告を表示し、**VIにC関数名を使用する**オプションをオンに設定します。関数パネルの名前を使用したい場合は、このオプションを手動でオフにし、名前の競合している各項目の名前を変更することができます。名前を変更した項目は、**VIにC関数名を使用する**オプションが変更された場合でもユーザが割り当てた名前を保持します。

名前を大文字に変換し、下線を削除する ― Cの関数名を使用して作成された名前は、関数パネル項目の名前に由来する名前に比べて「見劣りがする」ため、このオプションは頭文字を大文字に変え、下線をスペースに置き換えて「見やすく」します。それでも、略号を拡張することはできないため、FPの名前ほど見栄えはよくありませんが、重複しているFP項目名を個別に変更したくない場合にはこれは不可欠な処理です。

すべての制御器名を小文字に変換する ― VXI **Plug & Play**規格に適合するように、制御器の名前を小文字に変換します。

名前に基づいて計測器ドライバアイコンを割り当てる ― 生成されたVIの名前に基づいてVIにアイコンを割り当てます。コンバータは、関数名の中で初期化、閉じる、セルフテスト、リセット、構成、測定といったキーワードを検索し、それに見合ったアイコンを使用します。キーワードが見つからなかった場合は、デフォルトのアイコンが使用されます。また、アイコンの左上には最大7文字の計測器の接頭辞が表示されます。

配列引数のデフォルトサイズ ― 計測器ドライバのDLL関数が配列を出力する際には、関数が配列を書き込むためのメモリがあらかじめ割り当てられ、DLLに渡されている必要があります。このオプションを使用すると、配列に割り当てるデフォルトサイズを要素の数で指定することができます。Call Library関数で配列を引数として使用している場合は、コンバータはInitialize Array関数をドロップしてノードに渡す配列を作成します。このような配列の初期サイズは、**配列引数のデフォルトサイズ**オプションによって指定されます。このような構造を含むVIを容易に検出し、特殊なケースを個別に処理できるように、.outファイルに警告が生成されます。

メインのLabWindows/CVI関数パネルコンバータダイアログボックスで、OKを選択します。すると、変換中のFPファイルに対応するライブラリを選択するためのファイルダイアログボックスが表示されます（Create Library Callオプションが設定されている場合）。このダイアログボックスをキャンセルしても、Call Library関数ノードでライブラリのパスが指定されないままになるだけで、変換が中止されることはありません。

OKをクリックすると、作業ステータスのダイアログボックスに、コンバータが作成した新しい個々のVIの名前が表示されます。*prefix*.outという名前のログファイルが作成され、作成されたすべてのVI、および変換中に発生したすべての警告あるいはエラーが記録されます。警告やエラーが発生した場合は、変換終了時にこのファイルをチェックするよう指示するメッセージが表示されます。

26

Gの実行システムについて

この章では、VIのマルチタスク処理と実行について説明します。

Gの実行システムでは、複数のVIを同時に実行することができます。また、1つのVIで複数の並列分岐を使用し、それぞれの分岐を同時に実行することもできます。

通常の使用に際しては、マルチタスク処理の方法について詳しく知っている必要はありません。ダイアグラムの各部分を並列な実行とみなし、複数のVIが並列に実行されるものとみなすことができます。実行システムのマルチタスク処理機能を使用すると、他のVIを実行しながら別のVIを編集したりシングルステップで実行したりすることができます。また、誤って無限ループを持つVIを作成した場合でも、それによってコンピュータがロックしたり他のVIの実行が妨げられたりすることはありません。

時間の割り当てが重要な意味を持つ一部のアプリケーションでは、マルチタスク処理システムがどのように機能するかをよく理解することが重要になります。以下の項では、この問題について説明します。

マルチタスク処理の概要

ほとんどのコンピュータには、プロセッサが1つしかありません。これは、一度に1つのタスクしか実行できないことを意味します。マルチタスク処理では、1つのタスクを短時間実行したあと、別のタスクを実行することによって実現されます。このとき、個々のタスクに割り当てられた時間が非常に短ければ、見た目には複数のタスクが同時に実行されているように映ります。

Windows 3.1とMacintoshのオペレーティングシステムは、主として協調マルチタスク処理という方法を使用します。この方法では、個々のプログラムを他のタスクに対して自ら定期的に譲歩するように作成する必要があります。大部分のアプリケーションは、ユーザとの対話等のイベントの待機にかなりの時間を費やすため、通常は他のタスクを実行する機会は数多くあります。しかし、あるタスクが長時間にわたって時間を占有していると、他のタスクに実行の機会が与えられません。したがって、すべてのプログラムをこの点に配慮して注意深く作成しない限り、マルチタスク処理は正しく機能しません。

実際には、印刷、ウィンドウのドラッグ、ファイルの保存といった一般的なタスクを実行する場合、ほとんどのプログラムでは協調的マルチタスク処理は十分に実践されません。

多くのオペレーティングシステムは、プリエンプティブマルチタスク処理と呼ばれる別の形のマルチタスク処理を使用します。プリエンプティブマルチタスク処理では、オペレーティングシステムがタスクの切り替えやスケジューリングを処理します。個々のタスクには、限られた長さの実行時間が割り当てられます。与えられたタスクの時間が経過すると、1 つのタスクが強制的に中断され、別のタスクが実行を開始します。この切り替えは、開発者側で特別なコードを作成しなくても任意の時点で処理されます。

マルチスレッド処理

通常、アプリケーションがタスクを実行する必要がある場合、問題を解決する方法はいくつか存在します。1 つは、一度に 1 つずつステップを処理しながらタスクを完了する方法です。一方、ほとんどのタスクは、論理的には並列実行が可能なより小さいタスクに分けることができます。たとえば、文書を印刷する場合には、印刷をバックグランドで実行しながら別の文書を編集することができます。マルチスレッドという用語は、マルチタスク処理を 1 つのアプリケーションに適用し、そのアプリケーションを並列実行が可能なより小さいタスクに分割することを意味します。

協調的マルチタスク処理と同じように、アプリケーション中で協調的マルチスレッド処理を実践するプログラムを設計することができます。個々のスレッドは、他のスレッドやタスクを実行できるように定期的に譲歩するように作成する必要があります。マルチスレッドアプリケーションのすべてのスレッドは、1 つのプログラムの一部として作成されるため、開発者はタスク間の協力をより幅広く制御することができます。ただし、アプリケーション全体を通してこれを実践することは難しく、協調的マルチスレッド処理方式のアプリケーションでは、潜在的な並列性が多少損なわれる傾向があります。

オペレーティングシステムの中には、プリエンプティブマルチスレッド処理のためのメカニズムを備えているものもあります。オペレーティングシステムは、プリエンプティブマルチタスク処理システムで異なるプログラムを管理するのと全く同じ方法で、プログラムのスレッド間のコンテキストの切り替えを管理します。そのため、プログラムのアクティブなスレッドにプロセッサを効率的に配分できる可能性があります。

Gの実行システム

Gの実行システムは、協調的マルチスレッド処理だけでなく、プリエンプティブマルチスレッドをサポートするオペレーティングシステムでは、プリエンプティブマルチスレッドも使用します。使用されるスレッドの数は、プリエンプティブマルチスレッド機能を備えたシステムにおいても限りがあるため、場合によっては再びコオペラティブマルチスレッドを使用することになります。

（Windows 95/NT、Solaris 2、およびPowerMAX）アプリケーションはマルチスレッドです。実行システムは、スレッドを使用してVIのプリエンプティブマルチタスク処理を処理します。ただし、使用できるスレッドの数には限りがあるため、高度に並列化されたアプリケーションに対しては、スレッドを使い切った後は協調的マルチタスク処理に戻ります。また、アプリケーションと他のタスクとの間のプリエンプティブマルチタスク処理は、オペレーティングシステムが管理します。

（MacintoshおよびWindows 3.1）アプリケーションはシングルスレッドです。実行システムは、独自のスケジューリングシステムを使用してVIの協調的マルチタスク処理を行います。アプリケーションは、定期的に短時間ずつ譲歩することによって、他のアプリケーションとの間で協力的マルチタスク処理を実践します。

（HP-UXおよびSolaris 1）アプリケーションはシングルスレッドです。実行システムは、独自のスケジューリングシステムを使用してVIの協調的マルチタスク処理を行います。アプリケーションと他のタスクとの間のプリエンプティブマルチタスク処理は、オペレーティングシステムが管理します。

基本的な実行システム

以下の説明は、シングルスレッドのアプリケーションとマルチスレッドのアプリケーションの両方にあてはまります。

実行システムは、アクティブなタスクの待ち行列を保持します。たとえば、並列で実行される3つのループがある場合、1つのタスクを実行している間は他の2つのタスクは待ち行列の中で待機します。すべてのタスクが同じ優先順位を持つものと仮定すると、まず1つのタスクを一定の時間だけ実行します。次に、そのタスクを待ち行列の最後尾に戻し、次のタスクを一定時間実行します。タスクの実行が終了すると、実行システムはそのタスクを待ち行列から削除します。

実行システムは、生成されたVIコードを呼び出し、待ち行列の最初の項目を実行します。ある時点で、生成されたVIコードは実行システムをチェックし、実行すべき別のタスクがあるかどうかをチェックします。別のタスクがなければ、そのVIのコードが引き続き実行されます。

シングルスレッドアプリケーションでのユーザインタフェースの管理

実行システムは、VIを実行するだけでなく、ユーザインタフェースとの対話の調整を行う必要があります。ユーザがボタンをクリックしたり、ウィンドウを移動したり、あるいはスライド制御器の値を変更したりしたときは、実行システムはその動作を管理しながらバックグランドで引き続きVIを実行します。

シングルスレッドの実行システムは、ユーザのオペレーションに対する応答とVIの実行を交互に処理することでマルチタスク処理を実践します。実行がVIから実行システムに返されると、実行システムは処理すべきユーザインタフェースイベントが存在するかどうかをチェックします。イベントが存在しない場合は、VIに戻るか、または待ち行列から次のタスクを取り出します。

ボタンを押したり、メニューをプルダウンすると、その動作が完了するまでにしばらく時間がかかることがあります。実行システムは、ユーザが制御器やメニューを通して行う操作に対する応答とVIの実行を交互に処理することで、バックグランドでVIを引き続き実行します。

マルチスレッドアプリケーションと複数の実行システム

マルチスレッドバージョンには、複数の実行システムがあります。**VI設定→実行オプション**の**優先実行システム**を使用すると、6つの異なる実行システムのいずれかにVIを割り当てることができます。

選択できる実行システムは、下記の通りです。

- ユーザインタフェース
- 標準
- 計測器I/O
- データ集録
- 他1
- 他2

複数の実行システムを持つことの目的は、他のVIとは切り離して実行する必要のあるVIのために、おおまかな仕切りを確保するためです。デフォルトでは、VIはユーザインタフェースとは別のスレッドで実行する標準実行システムで実行されます。計測器I/O実行システムは、VISA、GPIB、およびシリアルI/Oが他のVIに干渉するのを防ぐために用意されたものです。同様に、データ集録実行システムはデータ集録用のVIのためのものです。

ユーザインタフェース実行システムは、マルチスレッドバージョンでもシングルスレッドバージョンの場合とまったく同じように動作します。ユーザインタフェースシステムは、ユーザインタフェースを管理する役割を果たします。VIはユーザインタフェースのスレッドで実行できますが、実行システムは協調的マルチタスク処理とユーザインタフェースイベントに対する応答を交互に処理します。

他の実行システムにはいずれも、そのシステム固有の待ち行列があります。これらの実行システムには、ユーザインタフェースを管理する役割はありません。これらの待ち行列に入っているVIのいずれかが制御器を更新する必要がある場合、VIはユーザインタフェースのスレッドにその役割を渡します。

また、ユーザインタフェース以外の各実行システムには、待ち行列からVIを実行するための2つのスレッドがあります。各スレッドがタスクを実行するため、たとえばVIがCINを呼び出す際には、2番目のスレッドがその実行システムの内部で引き続き他のVIを実行します。個々の実行システムのスレッドの数は限られているため、スレッドがいっぱいになるとシングルスレッドシステムの場合と同じようにタスクの実行は保留されます。

作成したVIを標準実行システムに設定したままにしてもそれらのVIを正しく実行できる場合は、別の実行システムに設定することも検討する必要があります。たとえば、計測器ドライバを作成する場合は、計測器I/O実行システムを使用するようにVIを設定することが必要になります。

VIを標準実行システムに設定したままにする場合でも、ユーザインタフェースをその固有のスレッドに切り離せることは、マルチスレッドの1つの重要な利点です。ユーザインタフェースで実行されるあらゆる動作（フ

ロントパネル上への描画、マウスクリックへの応答など）は、ブロックダイアグラムのコードの実行時間を犠牲にすることなく実行できます。グラフに大量の情報を表示する場合でも、それによってブロックダイアグラムのコードの実行が妨げられることはありません。同様に、長い演算ルーチンの実行によって、マウスクリックやキーボードからのデータ入力に対するユーザインタフェースの応答が妨げられることもありません。

複数のプロセッサを搭載したコンピュータは、マルチスレッドによってさらに多くのメリットを得ることができます。プロセッサが 1 つしかないシステムでは、オペレーティングシステムがスレッドを先制（プリエンプティブ）し、プロセッサ上の各スレッドに時間を配分します。複数のプロセッサを搭載したコンピュータでは、複数のプロセッサで同時にスレッドを実行できるため、複数の動作の真の同時実行が可能になります。

計測器 I/O、データ集録といった名前は、これらのシステムで使用するタスクのタイプを区別するために提案された名前にすぎません。I/O や DAQ は他のシステムでも実行できますが、このような名前を使用することはアプリケーションを分割したり、その構造を理解するのに役立ちます。

他 1 と他 2 は、専用のスレッドを必要とするその他のタスクがアプリケーションに存在する場合に使用できます。

同期ノードとブロッキングノード

すでに述べたように、ノードには同期のものもあり、これらのノードと他のノードとの間でマルチスレッドを実践することはできません。つまり、マルチスレッドバージョンでは同期ノードが最後まで実行され、同期ノードを実行しているスレッドはそのタスクが完了するまで占有されることを意味します。

コードインタフェースノード（CIN）、DLL 呼び出し、およびすべての演算関数は、同期で実行されます。ほとんどの解析用 VI や DAQ VI は CIN が含まれているため、同期で実行されます。たとえば、高速フーリエ変換（FFT）は、FFT の実行時間の長さには関係なく、それを実行するスレッドがなくても最後まで実行されます。

それ以外のノードは、ほとんどが非同期です。ストラクチャ、I/O 関数、タイミング関数、およびサブ VI は、非同期で実行されます。

Wait、Wait on Occurrence、Dialog Box、GPIB の各関数、VISA 関数、およびシリアル VI は、タスクが終了するまで待ちますが、スレッドを保留せずに待つことができます。これらのタスクは、そのタスクが完了するまでの間待ち行列から外されます。タスクが終了すると（たとえば、Dialog Box 関数が表示したダイアログボックスでユーザがボタンを押すと）、タスクは待ち行列の最後尾に格納されます。

タスクに優先順位を割り当てる

並行なタスクに優先順位を付ける方法は2通りあります。1つは、**VI設定**で優先順位を設定する方法です。もう1つは、Wait関数を戦略的に使用する方法です。

通常は、VIのデフォルトの優先順位を変更する必要はありません。優先順位を使用して実行順序を制御すると、意図した結果が得られない場合があります。優先順位の使用方法を誤ると、順位の低いタスクが完全に疎外されるおそれがあります。

Wait関数による優先順位の割り当て

Wait関数を使用すると、それほど重要でないタスクの実行回数を減らすことができます。たとえば、複数の並列のループがあり、そのうちのいくつかのループだけを回数を多く実行したいときは、優先順位の低いタスクにWait関数を挿入することにより、他のタスクにより多くの時間を割り当てることができます。

本章の「同期ノードとブロッキングノード」の項で述べたように、ダイアグラムが待機している場合は、他のタスクを実行できるようにそのダイアグラムは待ち行列から完全に外されます。

ユーザインタフェースをポーリングするループは、待機を使用するのに特に適しています。100 ～ 200 ミリ秒程度の待機は、実際には認識できるほどの時間ではありませんが、他のタスクを効率的に処理できるようにアプリケーションを解放することができます。また、オペレーティングシステムを解放し、他のスレッドや他のアプリケーションにより多くの時間を割り当てることもできます。

VI設定ダイアログボックスでの優先順位の割り当て

VIの優先順位は、**VI設定**ダイアログボックスの**優先順位**というメニュー項目を使用して設定します。

優先順位には6つのレベルがあり、それらを順位の低いものから順に並べると次のようになります。

- バックグラウンド（最下位）
- 低（通常）
- 中
- 高
- 最高（最上位）
- サブルーチン

最初の5つのレベルは動作の面ではよく似ています（最下位から最上位）が、「サブルーチン」レベルにはさらにいくつかの特性があります。以下の説明は、「サブルーチン」レベル以外のすべてのレベルにあてはまります。

ユーザインタフェーススレッドとシングルスレッド実行システムにおける優先順位

ユーザインタフェーススレッドの内部では、優先順位はシングルスレッドシステムでもマルチスレッドシステムでも同じように処理されます。以下では、これらのスレッド構成において優先順位がどのように処理されるかについて説明します。

シングルスレッドシステム、およびマルチスレッドシステムのユーザインタフェーススレッドでは、上で述べた実行システムの待ち行列には複数のエントリポイントがあります。優先順位の高いVIは、待ち行列中では優先順位の低いVIの前に格納されます。優先順位の高いタスクを実行中に、待ち行列にそれよりも優先順位の低いタスクしか存在しない場合は、優先順位の高いVIが継続して実行されます。たとえば、実行の待ち行列にそれぞれの優先順位を持つ2つのVIがある場合、「最高」レベルのVIが両方の実行が終了するまで独占的に実行時間を共有し、次に「高」レベルのVIが両方の実行が終了するまで独占的に時間を共有する、という方法で処理されます。

ただし、優先順位の高いVIがWait（ms）関数のように待機する関数を呼び出す場合は例外です（待機する非同期関数のリストについては、本章の「同期ノードとブロッキングノード」の項を参照のこと）。この場合、待機またはI/Oが終了するまで優先順位の高いVIが待ち行列から削除され、他のタスク（優先順位の低いVIなど）に実行するよう指示します。待機あるいはI/Oが終了すると、実行システムは保留されていたタスクを待ち行列の優先順位の低いタスクの前に再度挿入します。

また、優先順位の高いVIが優先順位の低いサブVIを呼び出すと、そのサブVIは呼び出されている間、呼び出し側のVIと同じ優先順位まで押し上げられます。したがって、VIが呼び出すサブVIの優先順位のレベルを変更して順位を押し上げる必要はありません。

優先順位の高いタスクに優先的に時間を割り当てると、優先順位の低いタスクのための時間がすぐに足りなくなるため、優先順位の使用には注意が必要です。優先順位の高いタスクが長時間実行されるように設計されている場合は、優先順位の高いタスクが定期的に待機またはI/Oを実行して待ち行列から削除されない限り、優先順位の低いタスクは実行されません。優先順位を使用する場合は、ダイアグラム中の優先順位の低い部分に待機を追加して、時間を解放することを検討してみる必要があります。

その他の実行システムにおけるマルチスレッドシステム優先順位

VI 設定の優先実行システムという実行システムメニュー項目に対応する6つの実行システムについては、この章ですでに説明しました。実際には、これら6つのそれぞれのカテゴリーには各優先順位レベル（サブルーチンレベルは除く）ごとに別々の実行システムが存在します（ユーザインタフェース実行システムは除く）。優先順位の付いたこれらの各実行システムには、そのシステム自身の待ち行列と、その待ち行列のダイアグラムを処理するための1つの専用スレッドがあります。

6つの実行システムではなく、優先順位とは無関係のユーザインタフェースシステム用の実行システムが1つと、他のシステム用の実行システムが20あります（5つのシステム×4つの優先順位レベル）。

これらの各実行システムのスレッドには、等級付けに基づいてオペレーティングシステムの優先順位レベルが割り当てられます。これは、通常の実行においては、優先順位の低いタスクよりも優先順位の高いタスクにより多くの時間が与えられることを意味します。上記のユーザインタフェーススレッドにおける優先順位と同様、優先順位の高いタスクが定期的に待機せずに長時間実行されると、優先順位の低いタスクに時間が割り当てられなくなる可能性があります。

オペレーティングシステムの中には、優先順位の低いタスクに時間が割り当てられなくなるのを防ぐために、順位の低いタスクの優先順位を強制的に押し上げるものもあります。したがって、順位押し上げ機能を備えたオペレーティングシステムでは、優先順位の高いタスクが継続して実行することを要求している場合でも、優先順位の低いタスクに定期的に実行の機会が与えられます。ただし、この機能はオペレーティングシステムによって異なり、また、一部のオペレーティングシステムではタスクの優先順位や順位押し上げ機能の内容がユーザによって調整される場合もあるため、この機能に完全に依存することはできません。

ユーザインタフェースシステムには、専用のスレッドは1つしかありません。このスレッドは、他の実行システムの「通常」レベルの優先順位と同じオペレーティングシステム優先順位に設定されます。したがって、VIを「通常より上」レベルの標準実行システムで実行されるように設定すると、ユーザインタフェースに実行時間が配分されなくなる可能性があります。これは、ユーザインタフェースの応答が遅れたり、無応答になる原因となります。同様に、VIを「バックグラウンド」レベルで実行されるように設定した場合は、ユーザインタフェーススレッドよりも低い優先順位で実行されます。

VIが優先順位の低いサブVIを呼び出す場合には、上で述べたユーザインタフェースでの場合と同様、呼び出し中は実行システムがサブVIの優先順

位を呼び出し側の VI と同じレベルに押し上げます。したがって、VI の設定でVIとそのサブVIの両方に同じ実行システムが割り当てられていると、優先順位の低いサブ VI への呼び出しは同じ優先順位の実行システムで実行されます。一方、VI が優先順位の高いサブ VI を呼び出した場合は、サブ VI への呼び出しは優先順位の高い別の実行システムで実行されます。

「サブルーチン」レベル

VIを「サブルーチン」レベル（最上位のレベル）に割り当てた場合は、動作は多少異なります。「サブルーチン」レベルは、VI をできるだけ効率的に実行できるようにすることを基本概念としています。コンパイラは、「サブルーチン」レベルの VI を他の VI と時間を共有しないようにコンパイルします。

VIは、「サブルーチン」レベルの優先順位で実行する際、そのVIを実行するスレッドの制御権を効率的に獲得します（呼び出し側と同じスレッドで実行されます）。このスレッドでは、この「サブルーチン」レベルの VI の実行が終了するまでは、他のVIはそれが同じ「サブルーチン」レベルに設定されたものであっても実行することはできません。このことは、シングルスレッドの実行システムでは、他のどんなVIも実行することはできないことを意味します。ユーザインタフェース実行システム以外の実行システムでは、サブルーチンを実行しているスレッドは他のVIを処理することはできませんが、その実行システムのもう 1 つのスレッドは、他の実行システムとともにVIを実行することができます。

サブルーチン VI が他の VI と時間を共有しないだけでなく、サブルーチン VI の実行はサブルーチンが呼び出されたときにフロントパネルの制御器や表示器が更新されないように処理されます。したがって、サブVIのフロントパネルを見るだけではその実行を確認することはできません。

サブルーチンVIは、他のサブルーチンVIを呼び出すことはができますが、他の優先順位の VI を呼び出すことはできません。「サブルーチン」レベルは、単純な演算を実行するサブVIのオーバーヘッドを最小限に抑えたい場合に使用します。

また、サブルーチンは他の実行待ち行列と対話するようには設計されていないため、通常サブルーチンを待ち行列から削除する関数を呼び出すことはできません。すなわち、Wait 関数、I/O 関数、またはダイアログボックス関数を呼び出すことはできません。

サブルーチンには、優先順位の高いアプリケーションで役に立つもう 1 つの機能があります。サブVIをポップアップしてビジー状態ならばサブルーチンコールしないを選択すると、実行システムはサブルーチンが別のスレッドで実行中のときには呼び出しを省略します。この機能は、タイムクリティカルなループでサブルーチンが実行する操作を安全に省略し、かつサブ VI の実行が終了するのを待つことによる遅延を回避したい場合にも

有効に活用できます。サブVIの実行を省略すると、サブVIの出力はすべてそのデータタイプのデフォルト値に設定されます。それにより、数値の出力は0に設定され、文字列および配列の出力はなくなり、ブールはFalseに設定されます。ただし、これはサブVIのフロントパネルの制御器のデフォルト値ではなく、タイプのデフォルトである点に注意してください。サブルーチンが実際に実行されたかどうかを確認したい場合は、サブルーチンを正しく実行されたときにはTrueを返すように設計する必要があります。正しく実行されなかった場合は、デフォルト値であるFalseが返されます。

再入実行

通常の状況では、実行システムは同じサブVIへの複数の呼び出しを同時に実行することはできません。再入実行不可能なサブVIを複数の場所から呼び出そうとすると、1つの呼び出しが実行され、他の呼び出しはそれが終わるまで待機します。（VI設定を使用して）VIを再入実行に設定すると、呼び出しの個々のインスタンスはそのインスタンスの情報ステータスを保持します。その場合、実行システムは複数の場所から同時に同じサブVIを実行することができます。再入実行は、次のような場合に使用すると便利です。

- 指定された時間だけ、またはタイムアウトになるまでVIが待機する場合
- VIのデータを共有したい場合に使用するグローバル変数とは違い、同じVIの複数のインスタンスの間でVIのデータを共有しない場合

再入実行は、**VI設定**ダイアログボックスの実行オプションを使用して開始します。**再入実行**を選択した場合は、一部のメニュー項目は使用不能になります。

再入実行を選択した場合は、下記の項目は使用できないことに注意してください。

- ロードされたらフロントパネルを表示する
- 呼び出されたらフロントパネルを表示する
- 実行のハイライト
- シングルステップ

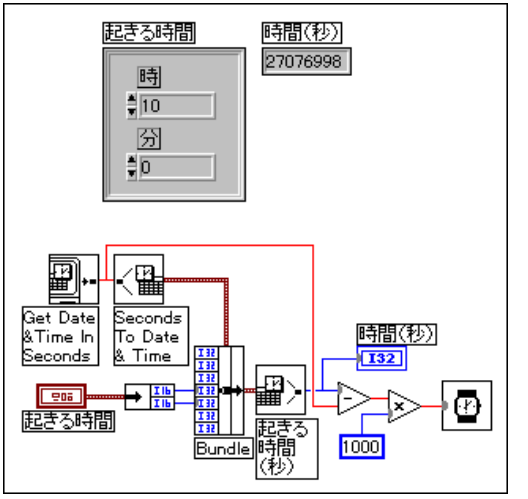
これらのメニュー項目が使用不能になるのは、そのときどきの状態が表示されるように、サブVIが個々の呼び出しごとにデータのコピーや実行状態を切り替える必要があるためです。

再入実行使用例

以下の2つの項で説明する例は、再入実行VIを示したものです。

待機するVIを使用する

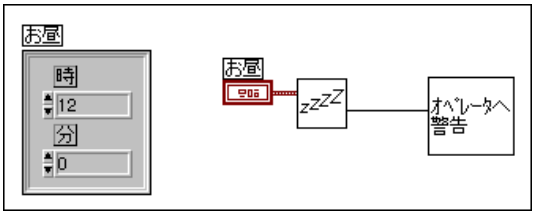
次の再入実行の例は、時間と分を入力として使用し、その時間になるまで待機するSnoozeと呼ばれるVIを示したものです。これを複数の場所で同時に使用したい場合は、VIを再入実行にする必要があります。



Get Date/Time In Seconds関数は現在の時刻を秒で読み込み、Seconds to Date/Timeはその値を時間値のクラスタ（年、月、日、時、分、秒、曜日）に変換します。Bundle関数は、現在の時間と分を、フロントパネルの、起きる時間クラスタ制御器に格納されている、同じ日のもっと遅い時刻を表す値と差し替えます。調整されたレコードは秒に変換され、さらに現在の時刻と後の時刻との時間差である秒数に1,000が掛けられてミリ秒に換算されます。得られた値はWait関数に渡されます。

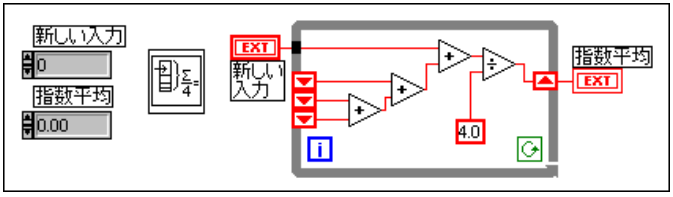
LunchとBreakという2つのVIは、Snoozeをサブ VIとして使用します。次の図にそのフロントパネルとブロックダイアグラムを示すLunchのVIは、正午になるとフロントパネルをポップアップし、昼食の時間になったことをオペレータに知らせます。BreakのVIは午前10時にパネルをポップアップし、休憩時間になったことをオペレータに知らせます。BreakのVIは、ポップアップサブVIが異なる点を除いてはLunchのVIと同じです。

LunchとBreakを並列で実行するためには、Snoozeが再入実行可能でなければなりません。再入実行不可能でない場合は、Lunchを最初に実行すると、Breakは Snoozeが目覚める正午まで待機するため、本来の時間より2時間遅れてしまうことになります。



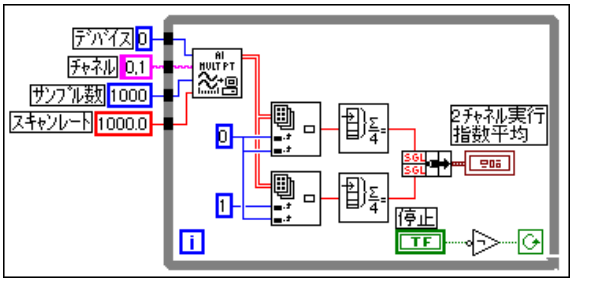
データの共有を目的としない保存VIを使用する

再入実行を必要とするもう1つのケースは、データを保存するサブVIに対して複数の呼び出しを実行する場合です。ここで、4つのデータポイントの移動指数平均を計算するExpAvgというサブVIを作成する場合について考えてみます。ExpAvgは、過去の値を記憶するため、3つの左端子を持つ初期化されないシフトレジスタを使用します。初期化されないレジスタについては、LabVIEWをご使用の場合は『LabVIEWユーザマニュアル』の「第3章　ループとチャート」を参照してください。



BridgeVIEWをご使用の場合は、『BridgeVIEW User Manual』の「第10章　ループとチャート」を参照してください。

次に、VIがExpAvgを使用して2つのデータ集録チャンネルの移動平均を計算する場合について考えます。たとえば、あるプロセスの2つのデータポイントの電圧を監視し、その移動指数平均を帯グラフで表示するものとします。ダイアグラムには、2つのExpAvgノードが存在します。呼び出しは、1つはチャンネル0に対して、1つはチャンネル1に対してというように交互に実行します。また、チャンネル0を最初に実行するものとします。このとき、ExpAvgが再入実行不可能だと、チャンネル1の呼び出しではチャンネル0の呼び出しで得られた平均が使用され、チャンネル0の呼び出しではチャンネル1の呼び出しで得られた平均が使用されます。ExpAvgを再入実行可能にすると、データを共有する危険をともなわずにそれぞれの呼び出しを別々に実行することができます。

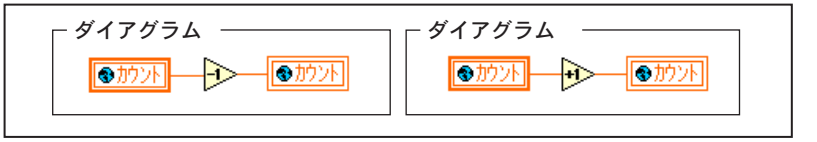


グローバル変数、ローカル変数、および外部リソースへのアクセスの同期化

実行システムは複数のタスクを並列で実行する可能性があるため、グローバル変数、ローカル変数、およびリソースが正しい順序で使用されるようにしておく必要があります。

競合状態

競合状態は、並列で実行される 2 つ以上のコードが、共有する同じリソースの値（一般的にはグローバルまたはローカルの変数）を変更する場合に発生します。次の 2 つの図は、競合条件の例を示したものです。



ダイアグラム 1 はカウントの値を順次繰り下げ、ダイアグラム 2 はカウントの値を順次繰り上げます。2 つのダイアグラムの間にはデータの依存関係はないため、これら 2 つのダイアグラムの実行は次のような順序で発生することもあります（カウントの初期値は4であるものとします）。

ダイアグラム 1 :　カウントを読み取ります（4）。
ダイアグラム 2 :　カウントを読み取ります（4）。
ダイアグラム 1 :　4 を繰り下げ、カウントに書き込みます（カウントは 5 になります）。
ダイアグラム 2 :　4 を繰り上げ、カウントに書き込みます（カウントは 3 になります）。

2 つのダイアグラムを実行するとカウントが繰り上げられたのち繰り下げられ、実際には元の値に戻るものと予想していても、競合状態が発生して一方の操作しか有効にならない場合があります。

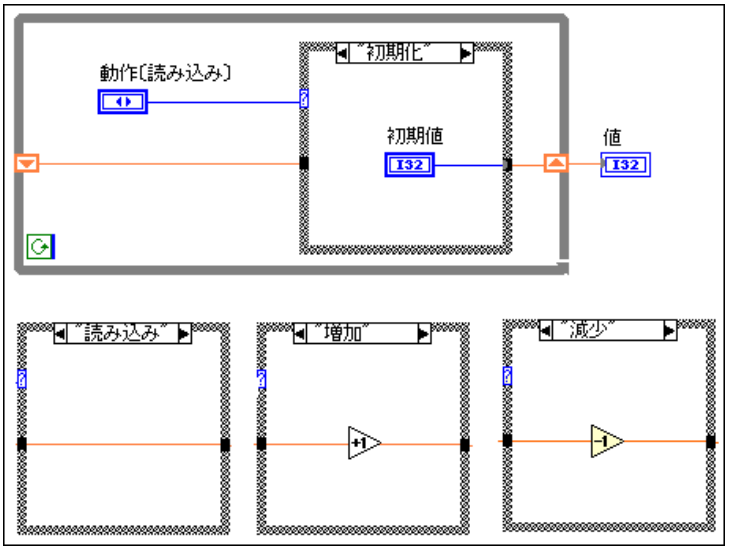
競合状態の発生を防ぐ方法はいくつかあります。最も簡単なのは、グローバル変数が変更される場所をアプリケーション全体を介して一カ所だけにする方法です。たとえば、前の例では、グローバル変数のカウントを繰り上げる、あるいは繰り下げるダイアグラムは、すべてChange Countという同じサブVIを呼び出します。これらのダイアグラムは、ブール引数を渡すことにより、グローバル変数が繰り上げられたか繰り下げられたかを示します。

再入実行不可能なVIは一度に1つの呼び出し側VIのためにしか実行されず、またグローバル変数を変更するVIはアプリケーションを介してChange Countだでけあるため、競合状態は回避されます。

☞ **注**　**シングルスレッドの環境では、「サブルーチン」レベルのVIを使用して競合状態を発生させずにグローバル変数の読み取り、修正、書き込みを行うことができます。これは、「サブルーチン」レベルのVIは実行スレッドを他のどのVIとも共有しないためです。マルチスレッドの環境では、別のスレッドで実行している別のVIが同時にグローバル変数にアクセスする可能性があるため、「サブルーチン」レベルを使用した場合でもグローバル変数への独占的アクセスは保証されません。**

関数的グローバル変数

グローバル変数に関連した競合状態を回避するもう1つの方法として、グローバル変数を一切使用せず、代わりに初期化されないシフトレジスタを持つループを使用するVIを使用して、グローバルデータを記憶する方法があります。このようなVIは、一般的には関数的グローバル変数、あるいはLabVIEW 2タイプのGlobal変数（LabVIEW 2ではグローバル変数は存在せず、専ら関数的グローバル変数を使用していました）と呼ばれます。関数的グローバル変数には、VIがどの関数を実行するかを指定するための動作入力引数があります。VIは、Whileループ中で初期化されないシフトレジスタを使用して操作の結果を記憶します。次の図は、単純なカウントのグローバル変数を使用する関数的グローバル変数のダイアグラムです。この例における動作は、**初期化**、**読み取り**、**繰り上げ**、および**繰り下げ**です。



VIが呼び出されるたびに、ループ中のダイアグラムは正確に1回だけ実行されます。**動作**引数に基づき、ループ中のケースはシフトレジスタの値を初期化、変更しない、繰り上げる、繰り下げるのいずれかの動作を実行します。

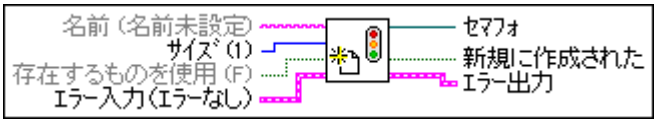
関数的グローバル変数は、前の例で示したように簡単なグローバル変数として使用することもできますが、スタックや待ち行列のバッファといったさらに複雑なデータストラクチャを使用する場合には特に便利です。また、関数的グローバル変数は、グローバル変数で表すことのできないファイル、計測器、あるいはDAQカードといったグローバルリソースへのアクセスを防ぐためにも使用できます。

セマフォ

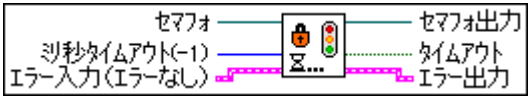
関数的グローバル変数VIは自分が持っているデータを一度に1つの発呼者に対してしか変更させないため、関数的グローバル変数を使用すると、グローバルリソースを処理する際の同期の問題を容易に解決することができます。関数的グローバル変数の1つの欠点は、保有しているリソースを新しい方法で修正しようとしたときに、グローバル変数のVIのダイアグラムを変更し、新たな動作を追加する必要がある点です。したがって、グローバルリソースの使用方法が頻繁に変わるような一部のアプリケーションでは、不便を生じる場合があります。そのような場合は、セマフォを使用してグローバルリソースへのアクセスを防ぐようにアプリケーションを設計する必要があります。

セマフォは共有リソースへのアクセスを防ぐために使用されるオブジェクトで、Mutex としても知られています。共有リソースにアクセスするコードは、しばしば重要な部分と言われています。一般的には、共通のセマフォで保護された重要セクションに入るためには、一度に 1 つのタスクしか必要としません。ただし、セマフォが複数のタスク（あらかじめ定義された数まで）が重要セクションに入るのを許可することは可能です。

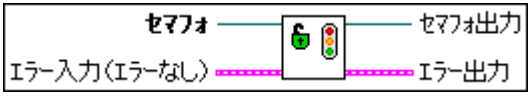
新しいセマフォは、Create Semaphore VI（**関数→上級→同期→セマフォ**パレット）で作成します。VI は、入力をセマフォの初期サイズとして使用します。このサイズにより、そのセマフォを同時に使用できるタスクの数が決定します。タスクがセマフォを使用して実行を開始すると、そのつどセマフォのサイズが 1 つずつ繰り下げられます。サイズが 0 になると、そのセマフォを使用しようとしたタスクは、別のタスクがそのセマフォを解放するまで待つ必要があります。



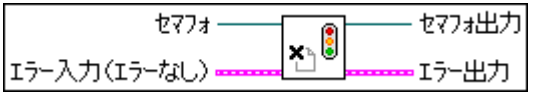
タスクは、Acquire Semaphore VI を呼び出すことによってセマフォを使用したいことを知らせます。セマフォのサイズが 0 より大きい場合は、VI は直ちにタイムアウト = FALSE を返し、タスクが実行されます。セマフォのサイズが 0 のときは、タスクはセマフォを使用できるようになるまで待つか、または VI が指定された時間だけ延期されるのを待ちます。VI がタイムアウト = TRUE を返したときは、セマフォを獲得できなかったことを意味し、タスクは重要セクションを実行しません。VI がタイムアウト =FALSE を返したときは、セマフォを獲得できたことを意味します。



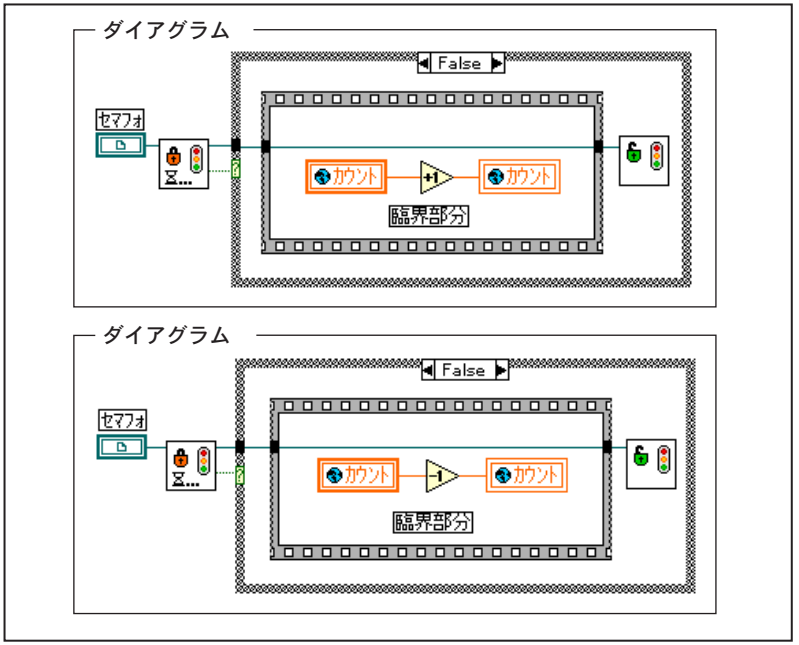
タスクは、セマフォを獲得してその重要セクションの実行を終了すると、Release Semaphore VI を呼び出してセマフォを解放します。セマフォを解放する際に、別のタスクがそのセマフォを獲得するのを待っている場合は、最初のタスクは引き続き実行を継続することができます。それ以外の場合は、セマフォのサイズが 1 つ繰り上げられます。



セマフォが不要になると、Destroy Semaphore VIを呼び出して破壊されます。そのセマフォに対して待機しているAcquire Semaphore VIが存在する場合は、それらのVIはエラーとタイムアウト=TRUEを直ちに返します。



次の例は、(グローバル変数の繰り上げや繰り下げを行う) 重要なセクションを保護するためのセマフォの使用方法を示したものです。セマフォはサイズ1をCreate Semaphore VIに渡すことによって作成されています。



重要セクションを実行しようとしている各ダイアグラムは、まず最初にAcquire Semaphore VIを呼び出す必要があります。セマフォが使用中のとき（サイズが0のとき）は、VIはセマフォが使用可能になるまで待機します。Acquire Semaphore VIからセマフォが獲得されたことを示すタイムアウト=FALSEが返されると、ダイアグラムはFALSEのケースの実行を開始します。ダイアグラムの重要セクション（シーケンスフレーム）の実行が終了すると、セマフォが解放され、待機している別のダイアグラムが実行を開始できるようになります。

その他の同期関数

セマフォ VI のほかにも、並列タスクを同期化するためのさまざまな関数やVIがあります。これらの関数は、いずれも**関数→上級→同期→セマフォ**パレットに入っています。

オカーレンス関数 ― オカーレンス関数は、別のタスクから通知があるまで 1 つまたは複数の関数を待機させるための手段を提供します。Generate Occurrence 関数は、Wait on Occurrence または Set Occurrence 関数に渡すオカーレンスを作成するために使用します。

ノーティフィケーション VI ― オカーレンスとよく似ていますが、通知を送出する際にメッセージを添付します。複数のVIが同じ通知を待っているときは、それらすべてのVIがメッセージを受け取ります。また、オカーレンスとは異なり、通知を取り消すことができます。

キュー VI ― 待ち行列タスクは、通知と同様他のタスクにメッセージを送出するために使用します。ただし、リスナがメッセージを検索する際に古いメッセージから先に受け取れるように、複数のメッセージを待ち行列に入れることができます。また、待ち行列VIは、リスナが1つしか存在しない場合に最も有効に活用できます。リスナが待ち行列から要素を削除すると、他のリスナーはその要素を見ることはできなくなります。

VIは、`enqueue.vi` を使用して待ち行列にメッセージを追加することができます。待ち行列から最も古いメッセージを取り出すためには、`dequeue.vi` を使用します。Create Queue を使用して待ち行列を作成する際には、待ち行列のサイズを制限するかどうかを指定します。サイズの制限された待ち行列を作成して、その容量を超えるデータを待ち行列に入れようとすると、`enqueue.vi` の操作はリスナが `dequeue.vi` を使用してデータを取り出し、要素を追加できるようになるまで実行が延期されます。

ランデブー VI ― ランデブーVIは、複数の並列タスクをある共通のポイントで同期化する必要がある場合に使用します。個々のタスクは、ランデブーポイントに到達するといったんそこで待機し、待機中のタスクの数が指定された数に達した時点で、すべてのタスクが一斉に実行を再開します。

実行システムおよび優先順位の使用に関する一般的提言

以下は、この章で説明した実行システムオプションの使用に関する一般的な提言です。

知っておくべき重要な点は、これらの提言の中には非常に複雑に見える項目もありますが、ほとんどのアプリケーションでは優先順位のレベルを使用したり実行システムを交換したりする必要はないという点です。実行システムは、VI間のマルチタスク処理を自動的に処理します。

デフォルトでは、あらゆるVIは標準実行システムで「通常」レベルの優先順位で実行されるように設定されます。マルチスレッドシステムでは、ユーザインタフェースの動作は専用のスレッドによって処理されるため、VIはユーザインタフェースの対話とは切り離されます。シングルスレッドのシステムを使用している場合でも、実行システムはユーザインタフェースの対話の処理とVIの実行とを交互に処理するため、同様の結果が得られます。

一般的に、実行の優先順位付けをするための最も簡単かつ安全な方法は、Wait関数を使用してアプリケーション中の優先順位の低いループの実行を遅らせる方法です。100 ～ 200 ミリ秒程度の遅れはユーザにはほとんど感知されないため、この方法は特にユーザインタフェースVIに使用するのに適しています。

優先順位を使用する場合は、慎重に使用する必要があります。一定時間動作する優先順位の高いVIを設計する場合、それほど優先順位の高くない部分ではそれらのVIに待機を追加し、優先順位の低いタスクと時間を共有するようにする必要があります。

他のタスクによって操作される可能性のあるグローバル変数、ローカル変数、およびその他の外部リソースを操作する場合も、注意が必要です。本章で先に説明した同期化を使用して、これらのリソースへのアクセスを防止するようにしてください。

シングルスレッドシステムやマルチスレッドシステムで優先順位の使用方法をどのように設計しても、VIが同じであればどちらのシステムでも同様の結果になります。ただし、タイミングにはわずかな違いが発生する場合があります。異なるオペレーティングシステムを使用しているユーザにVIを配布する場合は、これらの条件でアプリケーションを試してみることができます。マルチスレッドシステムを使用している場合は、**環境設定**ダイアログボックスの**パフォーマンス**項目の**マルチスレッドで実行**をオフにすることにより、シングルスレッドバージョンと同じように動作させることができます。

27

アプリケーションを管理する

この章では、G アプリケーションでのファイルの管理方法について説明します。

VI ライブラリを使用したファイルの整理

VI ファイルは、いくつかのグループに分類し、各グループをそれぞれ VI ライブラリ（ルールとして VI ライブラリのファイル名には .llb 拡張子を使用してます）として保存することができます。あるいは、個々の VI を個別のファイルとして保存し、それらのファイルをディレクトリに格納することもできます。2 つの方法の比較については、「第 2 章　VI を編集する」の「VI を保存する」項を参照してください。

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 ライブラリに分けることもできます。その場合、上位 VI をまとめて 1 つの VI ライブラリに保存し、その他の VI ライブラリは関数で分類した VI を保存できるようにセットアップします。ファイルをこのように整理すると、VI の管理が容易になります。

VI を複数のライブラリに分けて保存するもう 1 つの理由は、VI ライブラリのサイズが大きくなると、ファイルの保存に時間がかかるためです。VI を VI ライブラリに保存すると、ライブラリが一時ディレクトリにコピーされ、VI が圧縮されて VI ライブラリに保存され、さらに VI ライブラリが再び元のディレクトリにコピーされます。このコピープロセスは、オリジナルファイルの不慮の破壊を防ぐのに役立ちますが、VI を保存するのにも時間がかかります。

時間を短縮するためには、サイズが 1MB を超えるような VI ライブラリを作成しないようにするのが賢明です。ライブラリのサイズには実質的な制限はなく、VI ライブラリのサイズの違いによって VI ライブラリから VI をロードする時間に目立った差が出るわけではありません。ただし、VI ライブラリに VI を保存する場合、ライブラリのサイズが大きくなるとその分時間もかかります。

VI を保存する際の時間を短縮するもう 1 つの方法は、VI が VI ライブラリに入っているかどうかに関係なく、一時ディレクトリを VI と同じドライブ（UNIX ではパーティション）に作成する方法です。コピー元とコピー先が同じドライブ（あるいはパーティション）にあると、ファイルをより速くコピーすることができます。

VIライブラリを使用する場合は、メニューオプションから**ファイル→VIライブラリの編集...**を使用して特定のVIを最上位に設定することにより、最上位のVIを強調することができます。それにより、最上位のVIをファイルダイアログボックスのファイルリストの先頭に表示します。ディレクトリの場合にはこれに相当するオプションは存在しませんが、サブVIをサブディレクトリに入れることで最上位のVIを強調することができます。

ファイルをバックアップする

事故は絶対に発生しないとは限りません。ハードドライブはクラッシュし、ファイルシステムは破壊します。事故に備える最良の手段は、頻繁に、できれば1日1回はVIを別のドライブまたはテープにコピーしてバックアップコピーを作成することです。

事故の一例として、ユーザが数週間分の仕事のデータの入っているVIライブラリを誤って破壊してしまうケースが考えられます。たとえば、VIライブラリを別のデータファイルの保存先として指定すると、その時点でVIライブラリに別のデータを上書きしてしまうことになります。このような場合、バックアップが残っていればVIライブラリの大部分のデータは回復することができます。

VIを配布する

VIを他のマシンあるいは他のユーザに配布する準備ができたら、編集可能なブロックダイアグラムのソースコードを配布するのか、ブロックダイアグラムを隠すあるいは削除するのかについて検討します。VIは、ブロックダイアグラムなしでも保存することができます。ブロックダイアグラムを除外した場合は、ファイルのサイズが小さくて済み、ソースコードが誤って他人に変更されるのを防ぐことができる一方で、ユーザがVIを別のプラットフォームに移動したり、将来の開発環境に合わせてアップグレードすることはできなくなります。移動やアップグレードの機能が重要な要素となる場合は、パスワードでダイアグラムを保護する方法も検討してみる必要があります。その場合、ソースコードは引き続き使用できますが、パスワードを入力しない限り表示したり変更したりすることはできません。

オプション付き保存ダイアログボックスを使用すると、ダイアグラムをVIから削除することができます。ダイアグラムなしでVIを保存した場合は、他のユーザがVIを変更することはできません。その場合は、元のバージョンを上書きしないよう注意が必要です。

☞ **注**　**ダイアグラムのないVIは、アプリケーションの新バージョンや他のプラットフォーム用に変換することはできません。変換している間、VIをコンパイルし直す必要があります。ダイアグラムのないVIは、コンパイルし直すことはできません。**

また、ユーザがVIを実行する環境についても検討してみる必要があります。すなわち、エンドユーザに開発システムを提供するのか、ランタイムアプリケーションを提供するのかという点です。

アプリケーションは、VIがユーザ側で変更されないようにしたい場合に適しています。アプリケーションに含まれるメニューは簡略化されているため、ユーザはVIを編集したり、ブロックダイアグラムを見たりすることはできません。

アプリケーションは、LabVIEW のアプリケーションビルダライブラリを使用して作成されます。これらのライブラリを使用してアプリケーションを作成する場合は、次の点について考慮する必要があります。

- アプリケーションにVIライブラリを内蔵する必要があるか？
- アプリケーションに**開く**をメニュー項目として入れるか？
- アプリケーションに**終了**をメニュー項目として入れるか？

アプリケーションにVIライブラリを内蔵することを選択した場合は、ライブラリはランタイムエンジンと組み合わされて1つのファイルに収められます。このファイルを実行すると、ライブラリに入っているすべての最上位VIが自動的に開かれます。VIライブラリを内蔵しない場合は、VIはそのプラットフォーム用の開発環境とともに保存されているとみなされるため、アプリケーションは起動時にどのVIでも開くことができます。

メニュー項目の**開く**を有効にすると、アプリケーションにVIライブラリが内蔵されているかどうかには関係なく、アプリケーションを使用してファイルシステムのあらゆるVIを開き、実行することができます。VIライブラリを内蔵した場合は、ユーザが開発システムを持っている場合でも、ユーザや顧客がソースVIにアクセスできない完全に独立したアプリケーションを作成することができます。

同じ顧客に複数のVIのセットを提供する場合は、メニュー項目の**開く**を有効にした1つのランタイムアプリケーションの方が複数のアプリケーションを内蔵するよりも効果的です。これは、内蔵されたそれぞれのアプリケーションに約2～3MBのランタイムコードが含まれるためです。

ランタイムアプリケーションを作成するためには、別売のアプリケーションビルダライブラリが必要になります。LabVIEW をご使用の場合のアプリケーションの作成方法についての詳細はアプリケーションビルダのソフトウェアに添付されている『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。

☞ **注**　**BridgeVIEW の実行システムは、複数の別々のパートで構成されているため、アプリケーションビルダは BridgeVIEW のランタイムアプリケーションの作成には使用できません。**

パスワードによるVIの保護

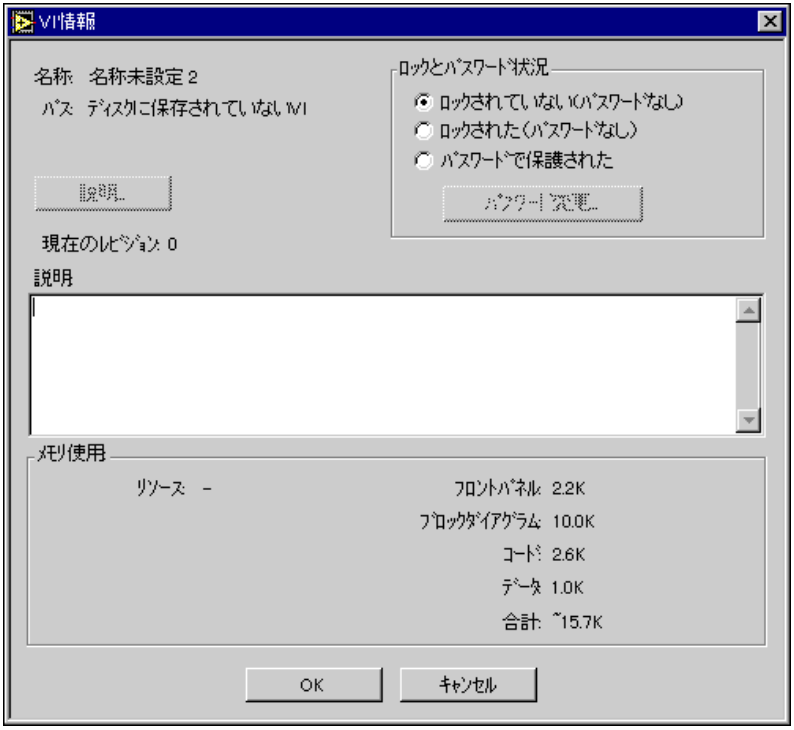
VIを他のユーザに配布する際には、自分が作成したダイアグラムのコードを他のユーザが見たり、他のユーザがVIを編集したりできないようにすることができます。

VIはダイアグラムなしで保存することができますが、その場合は、VIを別のプラットフォームやG言語アプリケーションの新しいバージョンにインポートする際に、VIをコンパイルし直すことはできなくなります。

パスワードでVIを保護すると、他のユーザがブロックダイアグラムを見たり編集したりできないようにすることができます。その場合、ユーザがパスワードで保護されたブロックダイアグラムを見たり、VIを編集するためには、VIの正しいパスワードを入力することが必要になります。

パスワードでVIを保護するためには、**ウインドウ→VI情報を表示...**を選択するか、または**ファイル→オプション付き保存**を選択します。

次の図に示す**VI情報**ダイアログボックスを使用して、個々のVIにパスワードを設定します。



以下では、**VI情報**ダイアログボックスにあるVIを保護するための3つのオプションについて説明します。

ロックされていない — VIが編集可能で、パスワードで保護されないことを示します。ロックされている(パスワード不使用)からこのオプションに切り替えると、VIのロックが直ちに解除されます。ロックされている(パスワード使用)からこのオプションに切り替えたときに、VIのパスワードがまだメモリに入っていない場合は、パスワードを入力するよう指示するプロンプトが表示されます。

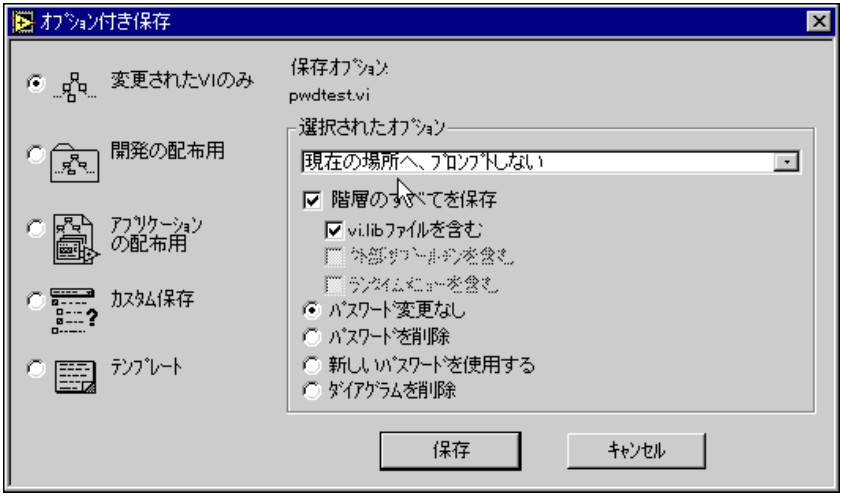
ロックされている(パスワード不使用) — VIは編集されないようにロックされますが、実行したりダイアグラムを見たりすることはできます。ロックされていないからこのオプションに切り替える場合、切り替えは直ちに完了します。一方、ロックされている(パスワード使用)からこのオプション切り替える際にパスワードがまだメモリに入っていない場合は、パスワードを入力する必要があります。

ロックされている(パスワード使用) — この状態は、VIがパスワードで保護されることを意味します。パスワードによるVIの保護はVIのロックとよく似ていますが、パスワードを入力しないとダイアグラムにアクセスすることはできません。この状態から別の状態に切り替える際にパスワードがまだメモリに入っていない場合は、パスワードを入力するよう指示するプロンプトが表示されます。

パスワードについての詳細は、次の「オプション付き保存ダイアログボックス」の項を参照してください。

オプション付き保存ダイアログボックス

オプション付き保存ダイアログボックスを使用すると、配布用のVIの階層全体を簡単に保存することができます。このダイアログボックスを使用すると、vi.libのVIを除く階層全体を選択して保存したり、階層中のVIが参照する外部のすべてのサブルーチンを保存したりできるようになります。**オプション付き保存**ダイアログボックスを次の図に示します。



オプション付き保存ダイアログボックスの左にあるオプションを選択することにより、既成の保存オプションを選択することができます。オプションを選択すると、現在選択されている保存方法が右側の領域に表示されます。この保存方法は、ダイアログボックスの右側の部分にあるオプションを選択してカスタマイズすることができます。

- **変更されたVIのみ** — 最前列のVIおよびそのサブVIに対するすべての変更を保存します。このオプションは、変更をすばやく保存したいときに便利です。
- **開発の配布用** — vi.libに属さないすべてのVI、制御器、および外部のサブルーチンを1つの場所（ディレクトリまたはライブラリ）に保存します。他の開発環境にはそれぞれのvi.libがあるため、このオプションを使用すると他のG開発システムに転送したい階層を簡単に保存することができます。
- **アプリケーションの配布用** — vi.libに属するものも含め、すべてのVI、制御器、および外部のサブルーチンを1つの場所に保存し、すべてのVIからダイアグラムを削除します。ライブラリを内蔵したランタイムアプリケーションを作成したい場合は、このオプションを使用します。実際にランタイムアプリケーションを作成するためには、別売のアプリケーションビルダが必要になります。

- **カスタム保存** — 既成のオプションでは条件が満たされない場合に、ダイアログボックスの**選択されたオプション**領域で特定のオプションを選択します。
- **テンプレート** — VIを.vifというファイル拡張子を持つVIテンプレートとして保存します。テンプレートVIをロードすると、テンプレートのコピーが作成されます。このコピーは、新しい名前を付けて保存する必要があります。

VI **情報**ダイアログボックスの**パスワード変更** ... ボタンは、VI にパスワードを設定していない場合には使用できません。パスワードを変更する際にパスワードがまだメモリに入っていない場合は、パスワードの入力を指示するプロンプトが表示されます。

前の図に示した**オプション付き保存**ダイアログボックスでは、1つのVIまたは複数のVIにパスワードを設定することができます。以下で、**オプション付き保存**ダイアログボックスのロックおよびパスワードのステータスについて説明します。

- **パスワード変更なし** — このオプションを選択した場合、何も変更されません。
- **パスワードを削除** — このオプションを選択すると、まだ保存していないすべてのVIをパスワードによる保護なしで保存します。VIを保存する際に、パスワードで保護されているVIに遭遇した場合は、ダイアログボックスはそのVIまたはVIのグループごとにそれぞれのパスワードを入力するよう指示します（必ずしも1つのVIに1つのパスワードが必要なわけではありません）。パスワードを削除するためには、それぞれの VI または VI のグループのパスワードを知っている必要があります。
- **新しいパスワードを使用** — このオプションを選択すると、VI がパスワードで保護されます。VI を保存する際に、すでにパスワードが設定されているVIに遭遇した場合は、そのVIまたはVIのグループごとにそれぞれのパスワードを入力するよう指示します（必ずしも1つの VI に1つのパスワードが必要なわけではありません）。VIまたはVIのグループに新しいパスワードを設定するためには、それぞれのVIまたはVIのグループのパスワードを知っている必要があります。
- **ダイアグラムを削除** — このオプションは、VIをダイアグラムなしで保存するため、VIが使用するメモリを節約することができます。VIが使用するメモリは少なくて済みますが、VI をソフトウェアの別のバージョンや別のプラットフォームに移すことはできません。

複数の開発者によるアプリケーションの設計

1つのプロジェクトに複数の開発者が参加する場合、多少時間はかかっても設計段階でそれぞれの役割を決めておくと、プロジェクトをスムーズに進めることができます。また、複数の開発者が存在する環境で役に立つソースコード管理ツールを備えたプロフェッショナルG開発ツールキットの使用も検討してみる価値があります。

作業は、アプリケーションの最上層の設計を検討し、アウトラインを作成することから始めます。アプリケーションの主要なコンポーネントについては、**スタブVI**を作成します。スタブVIは、サブVIのプロトタイプです。入力と出力を備えてはいますが、完全なものではありません。将来VIを開発する際に、機能を追加するためのプレースホルダとして使用されます。LabVIEWユーザ用のスタブVIの作成については、『LabVIEWユーザマニュアル』の「第28章　プログラムの設計」を参照してください。BridgeVIEWをご使用の場合は、『BridgeVIEW User Manual』の「Chapter 15, Program Design」を参照してください。

思い通りの機能を備えたスタブVIが作成できたら、個々の開発者がどのコンポーネントを担当するかを決定します。作業を容易にするため、主要なスタブVIをそれぞれ別々のディレクトリあるいはVIライブラリに保存し、その担当の開発者が管理するようにするのも1つの方法です。

マスタコピーを管理する

最良の方法は、プロジェクトのVIの**マスタ**コピーを1つのコンピュータ上に保存する方法です。チェックイン／チェックアウトのシステムを確立しておくと、VIの変更を管理することができます。このシステムでは、開発者は図書館から本を借りる場合と同じようにVIのコピーを借り受けます（チェックアウト）。その間、他の開発者は貸し出し中のファイルに触れることはできません。ファイルを借り受けた開発者は、VIに変更を加えたのち、VIを返却します（チェックイン）。

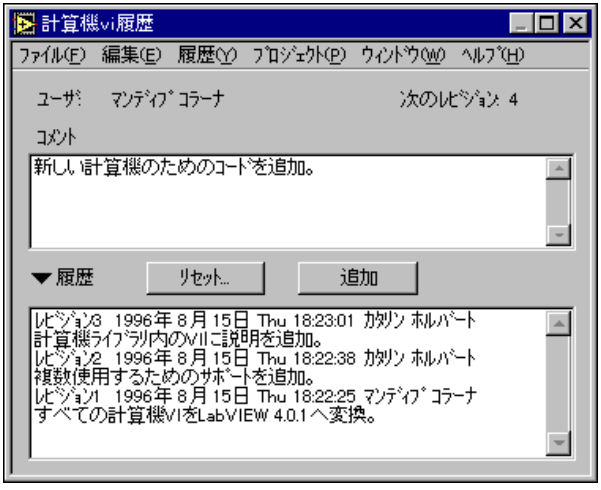
プロフェッショナルG開発ツールキットには、このようなファイルの共有およびチェックイン・チェックアウトを容易にするためのソースコード管理（SCC）ツールが含まれています。また、VIを比較し、その違いをチェックするためのユーティリティも含まれています。このユーティリティは、VIの各バージョンにどのような変更が加えられているかをチェックするのには非常に有効です。

VIの履歴ウィンドウ

VIには、フロントパネルやブロックダイアグラムだけでなく、VIの開発履歴（バージョン番号その他）を表示する履歴ウィンドウがあります。VIを変更した個々のユーザは、変更の内容をこの履歴に記録することができます。この機能を使用すると、VIに対する変更を追跡することができます。ただし、VIの履歴には、2つのVIを比較してその違いを検出するための手段は用意されていません（VIの比較ユーティリティについては、プロフェッショナルG開発ツールキットの説明を参照してください）。Gでは、履歴はVIの一部として保存されます。履歴は、VIを変更したユーザが入力した一連のコメントで構成されます。

☞ **注**　**VI設定→ドキュメントまたは編集→環境設定→履歴のダイアログボックスでLabVIEWが生成したコメントを記録するオプションを選択しない限り、VIの履歴にコメントが自動的に作成されることはありません。VIに変更を加えたユーザは、必ずそのデータを入力して履歴の内容を更新する必要があります。**

履歴ウィンドウは、**ウインドウ→履歴を表示**を選択するか、またはそれに対応するキーボードコマンドの<Ctrl-y>**（Windows）**、<command-y>**（Macintosh）**、<meta-y>**（Sun）**、または<Alt-y>**（HP-UX）**を押すことにより、VIの編集時に見ることができます。次の図に履歴ウィンドウの見本を示します。

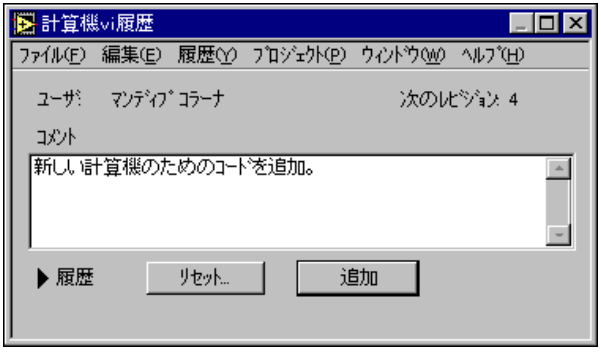


VIを編集する際には、ウィンドウ上段のコメントボックスに重要な変更に関する説明を入力することができます。変更ができたら、**追加**ボタンをクリックしてコメントを履歴に追加します。VIを保存する際にコメントボックスにコメントが存在する場合は、コメントが履歴に自動的に追加されます。追加したコメントは、永久に履歴の一部になります。履歴は書き直す

ことができないため、コメントを追加する前にコメントの内容をチェックする必要があります。

ウィンドウの下にあるボックスには、VIの履歴が表示されます。ダイアログボックスは、履歴の個々のコメントに対し、VIのレビジョン番号、日時、変更者の名前を含むヘッダを表示します。この履歴に追加された3つのコメントを見ることができ、それぞれのコメントはヘッダとともに表示されます。前の図は単なる見本として示したものであるため短いコメントしか表示されていませんが、実際にVIを編集する際には任意の長さのコメントを作成することができます。

履歴ウィンドウは、履歴を表示できるだけの大きさがありますが、**履歴**の矢印（ラベルの左の小さな黒い三角）をクリックすると履歴を隠すことができます。履歴を隠すと、ウィンドウのサイズが小さくなり、VIを編集する際に邪魔にならなくなります。再度履歴を表示したいときは、同じ矢印をクリックすると履歴が表示されます。次の図に例を示します。



また、ウィンドウのサイズを変更して履歴ボックスやコメントボックスのサイズを変更することもできます。

レビジョン番号

レビジョン番号もVIの履歴の一部で、VIが変更されているかどうかを知るための1つの手段です（変更に関するコメントが入力されている場合は、どのように変更されたかも知ることができます）。レビジョン番号は0から始まり、VIを保存するたびに1つずつ繰り上がります。現在のレビジョン番号（ディスクに入っているレビジョンの番号）は、**VI情報を表示...**ダイアログボックスに表示されます。また、**編集→環境設定→履歴**のダイアログボックスでVIのタイトルバーにレビジョン番号を表示するオプションがチェックされている場合は、タイトルバーにも表示されます。

履歴ウィンドウやVIのウィンドウに表示される番号は、次のレビジョンの番号、すなわちVIを保存する際にディスクに保存される番号が表示されま

す。この番号は、現在のレビジョン番号に 1 を加えた数になります。履歴にコメントを追加する際には、コメントのヘッダには次のレビジョン番号が使用されます。たとえば、前に示したサンプル画面の**追加**ボタンをクリックした場合、履歴に追加されるコメントのレビジョン番号は4になります。VIを保存する場合は、現在のレビジョン番号は4になり、変更に関するコメントにも同じ番号が付けられます。

☞ **注**　**変更がVIの履歴に関する変更だけの場合は、VIを保存してもレビジョン番号は繰り上がりません。**

レビジョン番号とコメントが一致しないことがよくありますが、これはVIを保存する際にコメントを入力しなかったためです。これは、別に問題ではありません。実際には、レビジョン番号が便利である理由は、それが独立している点にあります。たとえば、他のユーザからVIのコピーを入手したとします。そしてその後、コピーを入手したあとにそのユーザがVIを更新しているかどうかをチェックする必要が生じたとします。その場合、そのユーザがコメントを付けずにVIを変更していたとしても、レビジョン番号を見れば更新されているかどうかが一目でわかります。

履歴情報をリセットする

コメントボックスの下には、**リセット**というラベルの付いたボタンがあります。**リセット**を押すと、履歴が消去され、レビジョン番号が 0 にリセットされます。この機能は、VI をコピーし、その VI を新しい状態（履歴なしの状態）で使用したいときに便利です。

履歴は、厳密には開発者用のツールであるため、VIのブロックダイアグラムを削除した場合は履歴は自動的に削除されます。履歴ウィンドウは、ランタイムバージョンのアプリケーションでは使用することはできませんが、ブロックダイアグラムを削除したVIでもVI情報ダイアログボックスでレビジョン番号を見ることは可能です。ブロックダイアグラムを削除する前に**リセット**ボタンを使用してVIの履歴をリセットすると、レビジョン番号を削除することができます。

履歴情報を印刷する

VIを印刷する際に、文書の印刷ダイアログボックスで**文書のすべて**を選択すると、履歴とレビジョン番号も印刷することができます。履歴情報だけが印刷されるように印刷のフォーマットを変更したい場合は、**カスタムの設定**ボタンを使用すると印刷の内容を正確に指定することができます。また、文書の印刷ダイアログボックスで出力先リングの**標準テキストファイル**を選択すると、履歴情報をファイルにエクスポートすることができます。

VI設定および環境設定ダイアログボックスの関連オプションを設定する

VI設定ダイアログボックスと**環境設定**ダイアログボックスには、VIの履歴の動作を指定するためのオプションが用意されています。

VI設定ダイアログボックスのオプションを使用すると、履歴情報を自動的に記録するかどうか、および履歴情報をいつ記録するかを、各VIごとに設定することができます。これらの設定に関しては、「第6章　VIおよびサブVIをセットアップする」の「文書作成オプション」の項を参照してください。

環境設定ダイアログボックスのオプションを使用すると、新しいVIの履歴をデフォルトでどのように設定するかを指定することができます。また、このダイアログボックスでは、履歴情報に記録するユーザ名を指定することもできます。これらの設定については、「第7章　環境をカスタマイズする」の「履歴の環境設定」の項を参照してください。

28

パフォーマンスについて

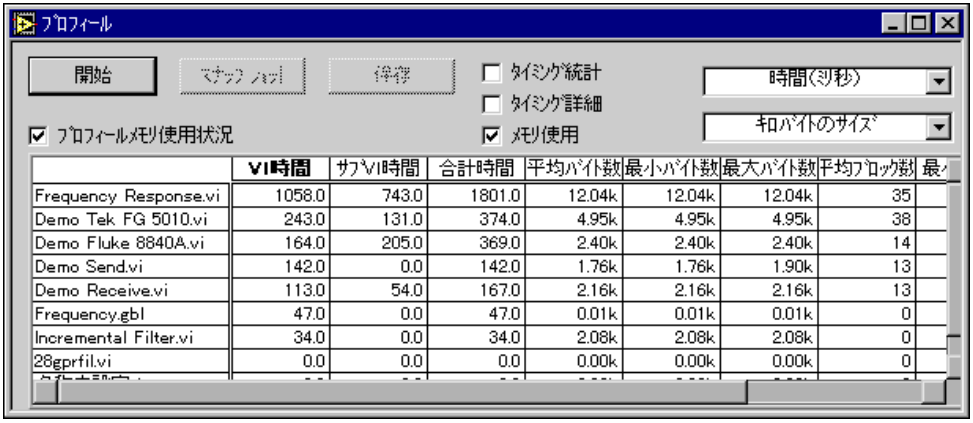
この章は、3つのセクションに分かれています。最初のセクションでは、VIの実行時間に関するデータを表示し、シングルスレッド、マルチスレッド、およびマルチプロセッサアプリケーションを監視するパフォーマンスプロフィールについて説明します。2番目のセクションでは、実行時の速度に影響を及ぼす要素について説明します。3番目のセクションでは、メモリの使用量に影響を及ぼす要素について説明します。

VIのパフォーマンスのプロフィール

VIのパフォーマンスプロフィールは、アプリケーションによりメモリがどのように使用されているかだけでなく、アプリケーションの時間を消費する場所も特定できる強力なツールです。これらのデータは、最も時間を要するVIを最適活用するために、アプリケーションのホットスポットを探し出す際にきわめて貴重な情報となります。パフォーマンスプロフィールは、システム内の各VIの時間とメモリの使用状況を対話形式の表で表示します。表の各行には、特定のVIの情報が含まれています。各VIが使用する時間は、いくつかのカテゴリーに分類され、統計データとしてまとめられます。パフォーマンスプロフィールウィンドウは、VIの1回の実行あたりの最大、最小、平均の各時間を計算します。

この対話形式の表を使用すると、これらのデータのすべてまたは一部を表示したり、カテゴリーごとに分類したり、特定のVIから呼び出されたときのサブVIのパフォーマンスデータを見たりすることができます。

プロフィールウィンドウにアクセスするためには、**プロジェクト→プロフィールウィンドウを表示**を選択します。次の図に使用時のウィンドウの見本を示します。



	VI時間	サブVI時間	合計時間	平均バイト数	最小バイト数	最大バイト数	平均ブロック数	最
Frequency Response.vi	1058.0	743.0	1801.0	12.04k	12.04k	12.04k	35	
Demo Tek FG 5010.vi	243.0	131.0	374.0	4.95k	4.95k	4.95k	38	
Demo Fluke 8840A.vi	164.0	205.0	369.0	2.40k	2.40k	2.40k	14	
Demo Send.vi	142.0	0.0	142.0	1.76k	1.76k	1.90k	13	
Demo Receive.vi	113.0	54.0	167.0	2.16k	2.16k	2.16k	13	
Frequency.gbl	47.0	0.0	47.0	0.01k	0.01k	0.01k	0	
Incremental Filter.vi	34.0	0.0	34.0	2.08k	2.08k	2.08k	0	
28gprfil.vi	0.0	0.0	0.0	0.00k	0.00k	0.00k	0	

このウィンドウに関しては、注意すべき点がいくつかあります。1つは、メモリの使用に関する情報の収集がオプションである点です。これは、情報収集を実行することによってVIの実行時間にかなりのオーバーヘッドが付加される可能性があるためです。このデータを収集するかしないかは、プロフィールの開始ボタンを押す前に**プロフィールメモリ使用状況**のチェックボックスを正しく設定して指定する必要があります。このチェックボックスの設定は、プロフィールのセッションを開始したあとで変更することはできません。プロフィールウィンドウでは、次のボタンを使用できます。

- **開始** ― パフォーマンスデータの収集を可能にします。プロフィールのセッションは、アプリケーションを実行していない状態で開始するのが最適です。そうすることにより、アプリケーションの部分的な実行ではなく完全な実行のデータを測定することができます。

- **スナップショット** ― 現在使用可能なデータを表示します。メモリ内のすべてのVIのデータを集め、表形式で表示します。

- **保存** ― 現在表示されているデータを、テキストをタブで区切ったスプレッドシート形式でディスクに保存します。このデータは、スプレッドシートアプリケーションやVIを使用して見ることができます。

結果を表示する

データは、一部だけを表に表示することもできます。一部の基本的なデータは常に表示されますが、統計値、詳細、およびメモリの使用状況（有効の場合のみ）といったデータを表示するかしないかは、プロフィールウィンドウの該当するチェックボタンを使用して選択することができます。

グローバル VI に関しては、パフォーマンスデータも表示されます。ただし、下記の各カテゴリーの項で説明するように、このデータは多少異なる解釈が必要となる場合があります。

特定の VI から呼び出されたときのサブ VI のパフォーマンスデータは、表中のその VI の名前をダブルクリックすることによって呼び出すことができます。VI の名前をダブルクリックすると、VI の名前のすぐ下にその VI の各サブ VI のデータを示す新たな行が 1 行ずつ表示されます。グローバル VI の名前をダブルクリックした場合は、そのグローバル VI のフロントパネルの各制御器に対して新たな行が 1 行ずつ表示されます。

表中の行のデータは、列のヘッダをクリックすることによってソートすることができます。現在のソート欄は、ヘッダタイトルが太字で表示されます。

VI の時間データは、必ずしも VI が実行するまでに要した時間と一致するとは限りません。その理由は、マルチスレッドの実行システムでは 2 つ以上の VI が交互に実行される可能性があるためです。また、どの VI にも起因しないある程度のオーバーヘッドも発生します。このようなオーバーヘッドには、ユーザがダイアログボックスに応答するまでの時間、ブロックダイアグラムの Wait 関数が消費する時間、あるいはマウスのクリックをチェックするために消費される時間などがあります。

パフォーマンスデータの最初の 3 行に常に表示される基本データは、下記の項目で構成されます。

- **VI時間** — このVIのコードを実際に実行し、そのデータを表示するために要した合計時間でもあり、また、ユーザが VI のフロントパネル制御器（存在する場合のみ）と対話する際に要した時間でもあります。この合計時間は、次に説明する**タイミング詳細**の欄に内訳が示されます。グローバル VI の場合は、この時間はそのグローバル VI のすべての制御器との間でデータをコピーするのに要した時間の合計時間になります。グローバル VI の個々の制御器の時間情報は、グローバル VI の名前をダブルクリックすることによって見ることができます。
- **サブVI時間** — このVIのすべてのサブVIが要した時間。この時間は、この VI が呼び出すすべてのサブ VI、およびこれらのサブ VI が呼び出すサブ VI の VI 時間（上記参照）などを合計した時間です。
- **合計時間** — 上記 2 つのカテゴリの合計。合計時間を計算します。

時間データ

タイミング統計のチェックボックスがチェックされていると、表に下記の欄が表示されます。

- **実行回数** — このVIの実行を完了した回数。グローバルVIの場合は、そのいずれかの制御器にアクセスした合計回数になります。
- **平均** — このVIの1回の実行に要した時間の平均値。この値は、VI時間を単純に実行回数で割った値です。
- **最短** — VIを1回実行するのに要した時間の最小値。
- **最長** — VIを1回実行するのに要した時間の最大値。

タイミング詳細のチェックボックスがチェックされていると、VIが要した合計時間のいくつかのカテゴリーの内訳を見ることができます。ユーザインタフェースを多く使用しているVIでは、これらの内訳をチェックすることでどの操作に最も時間がかかったかを知ることができます。

- **ダイアグラム** — このVIのダイアグラムに対して生成されたコードだけを実行するのに要した時間。
- **ディスプレイ（表示時間）** — このVIのフロントパネル制御器をダイアグラムから渡された新しい値に更新するのに要した時間。
- **描画（描画時間）** — このVIのフロントパネルを描画するのに要した時間。描画時間には、下記が含まれます。
  - ウィンドウがすでに開いているとき、または別のウィンドウの後ろに隠れていたウィンドウを前面に表示したときにフロントパネルを描画するのに要した時間。
  - 概念的には表示時間、すなわちダイアグラムから渡された新しい値を表示するのに要した時間であっても、制御器が透明であったり重なり合ったりしているために発生する時間。このような制御器は、ダイアグラムから新しいデータを受け取ると、それらが画面上で占有している領域内のすべてのものを正しい順序で描画し直せるように、それらの領域を無効にする必要があります。その他の制御器は、ダイアグラムから新しいデータを受け取った時点で直ちにフロントパネル上で描画できます。領域を無効にして再描画する場合はオーバーヘッドが増大します。オーバーヘッドの大部分（ただし全部ではありません）は、描画時間として表示されます。
- **トラッキング** — ユーザがこのVIのフロントパネルと対話する際のマウスの動きを追跡するのに要した時間。この値は、グラフのズームインやズームアウト、ポップアップメニューからの項目の選択、制御器のテキストの選択および入力など、操作の種類によっては重要な意味を持ちます。
- **ローカル** — フロントパネルのローカル変数への、あるいはローカル変数からのデータのコピーに要した時間。ユーザの経験では、大量かつ

複雑なデータが関与している場合は、この時間が特に重要な意味を持つことが明らかにされています。

メモリデータ

メモリ使用のチェックボックスがチェックされていると、VIによるメモリの使用状況を見ることができます（メモリ使用のチェックボックスは、プロファイリングのセッションを開始する前に**プロフィールメモリ使用状況**のチェックボックスを選択しておかないと使用できません）。これらの値は、VI用のデータスペースが使用するメモリ量を示すもので、すべてのVIに必要なサポートデータ構造は含まれません。VI用のデータスペースには、フロントパネルの制御器が明示的に使用するデータだけでなく、コンパイラが暗黙的に作成する一時バッファも含まれます。

メモリのサイズはVIの実行終了後に測定されますが、この値は正確な総使用量を反映しない場合もあります。たとえば、VIが実行中に大きな配列を作成した場合でも、実行終了前にその配列のサイズが小さくなっていると途中の大きな配列のサイズはこの値には反映されません。

この部分には、使用したバイト数に関連したデータと、使用したブロック数に関連したデータの2つのデータセットが表示されます。ブロックとは、1つのデータを記憶するために使用される連続したメモリセグメントを意味します。たとえば、整数の配列は長さにおいては複数バイトである場合もありますが、1つのブロックしか占有しません。実行システムは、配列、文字列、パス、および（Picture Control Toolkitからの）画像に対しては、独立したブロックを使用します。アプリケーションのメモリヒープに多数のブロックがあると、(実行だけでなく) 全体的なパフォーマンスの低下を招くおそれがあります。

メモリ使用のカテゴリーは下記の通りです。

- **平均バイト** — このVIのデータスペースが1回の実行で使用したバイト数の平均値。
- **最小バイト** — このVIのデータスペースが1回の実行で使用したバイト数の最小値。
- **最大バイト** — このVIのデータスペースが1回の実行で使用したバイト数の最大値。
- **平均ブロック** — このVIのデータスペースが1回の実行で使用したブロック数の平均値。
- **最小ブロック** — このVIのデータスペースが1回の実行で使用したブロック数の最小値。
- **最大ブロック** — このVIのデータスペースが1回の実行で使用したブロック数の最大値。

VIの実行速度

コンパイルでは、通常は非常に高速で実行されるコードが生成されますが、時間が重要な意味を持つアプリケーションの場合には、可能な限りの手段を尽くしてVIから最大限のパフォーマンスを引き出すことが要求されます。この項では、実行速度に影響を及ぼす要素について説明するとともに、最大限のパフォーマンスを得るのに役立ついくつかのプログラミングテクニックを紹介します。

パフォーマンス低下の原因を特定するためには、下記の項目をチェックします。

- 入出力（ファイル、GPIB、データ集録、ネットワーキング）
- 画面表示（大規模な制御器、制御器の重なり、過剰表示）
- メモリの管理（配列や文字列の非効率的な使用、非効率的なデータ構造）

他にも、実行のオーバーヘッドやサブVIの呼び出しのオーバーヘッドといった要素もパフォーマンスに影響する場合がありますが、これらの要素の影響は通常は非常に小さいため、速度低下の最大要因となることはありません。

入出力

通常、入出力呼び出しは相当な量のオーバーヘッドを発生させます。入出力呼び出しが演算操作の実行よりもはるかに多くの時間を消費することも少なくありません。たとえば、簡単なシリアルポートの読み込みで数ミリ秒のオーバーヘッドが発生する場合もあります。シリアルポートを使用するアプリケーションでは、この程度のオーバーヘッドは当然起こります。このようなオーバーヘッドの原因は、入出力呼び出しにオペレーティングシステムの複数の層を通したデータ転送が関与していることにあります。

過剰なオーバーヘッドに対処する最良の方法は、入出力呼び出しの回数を減らすことです。何度も入出力呼び出しを実行してデータを少しずつ転送するのではなく、1回の呼び出しで大量のデータを転送できるようにアプリケーションを作成することができれば、パフォーマンスを向上させることができます。

たとえば、データ集録（NI-DAQ）VIを作成する場合、データを読み取る方法としていくつかの選択肢が考えられます。AI Sample Channel VIのようなシングルポイントのデータ転送関数を使用する方法もあれば、AI Acquire Waveform VIといったマルチポイントのデータ転送関数を使用する方法もあります。100個のポイントでデータを集録する必要がある場合は、タイミングを確保するためのWait関数を使用してAI Sample Channel

VIをループで使用します。あるいは、100個のポイントが必要なことを示す入力を備えたAI Acquire Waveform VIを使用することもできます。

AI Acquire Waveform VIを使用した方が、より高速で正確なサンプリング速度を実現できますが、それは、AI Acquire Waveform VIがデータのサンプリングをハードウェアタイマを使用して管理するためです。また、AI Acquire Waveform VIは、オーバーヘッドがAI Sample Channel VIの一回の呼び出しのオーバーヘッドとほぼ同じであるにもかかわらず、はるかに多くのデータを転送することができます。

画面表示

フロントパネル上の制御器を頻繁に更新することは、アプリケーションの中でも最も時間を消費する操作の1つです。特に、グラフやチャートといった複雑な表示を使用する場合は、大量の時間を消費します。

ほとんどの制御器は効率的であるため、このオーバーヘッドは最小限に押さえられます。新しいデータを受け取ったときに、それらが古いデータと同じであった場合、制御器は表示を更新しません。ただし、グラフやチャートは例外です。

表示の更新の速度が問題となる場合、問題を解決するための最良の方法は使用する制御器の数を減らし、かつ表示の内容をできるだけ単純化することです。グラフやチャートの場合は、オートスケール、スケールマーカー、およびグリッドをオフにすることによって表示速度を上げることができます。

他のオブジェクトと重なり合っている制御器では、表示速度が著しく低下します。その理由は、制御器の一部が隠れていると、画面のその部分を再描画するのにより多くの作業を実行しなければならないためです。制御器が重なり合っている場合は、環境設定の**スムーズアップデート**をオンにしていない限りちらつきがより顕著になります。

他の種類のI/Oと同様、制御器の表示には一定のオーバーヘッドが存在します。表示器には、チャートなどある種の制御器を使用して一度に複数のポイントを渡すことができます。一度に多くのデータをチャートに渡すようにすると、作成したチャートの更新回数を減らすことができます。したがって、個々のポイントを受け取るたびに表示するのではなく、チャートのデータを配列として収集し一度に複数のポイントを表示するようにすれば、データの表示速度を大幅に上げることができます。

実行中はフロントパネルが閉じるサブVIを設計する場合は、表示のオーバーヘッドを気にする必要はありません。フロントパネルが閉じていれば、制御器の描画のオーバーヘッドは発生せず、グラフが配列よりも多くの時間を消費するということもありません。

制御器と表示器には同期の表示ポップアップがあり、このポップアップは、マルチスレッドのシステムにおいて更新を延期するかどうかを制御します。シングルスレッドの実行では、この項目は機能しません。ただし、シングルスレッドバージョンのVIの内部におけるこの項目のオン／オフの設定によって、これらのVIをマルチスレッドシステムにロードしたときの更新の動作が違ってきます。

デフォルトでは、この項目はオフに設定されます。オフとは、実行システムがフロントパネルの制御器や表示器にデータを渡す際に、制御器または表示器にデータを渡したのち、直ちに実行を続行できることを意味します。その後のある時点で、ユーザインタフェースシステムは制御器または表示器が更新を必要としていることを通知したのち、画面を更新して新しいデータを表示します。実行システムが短時間に連続して何度も制御器を更新しようとすると、途中の更新画面を見られなくなる場合があります。

多くのアプリケーションでは、これによりユーザに表示する内容に影響を及ぼさずに実行速度を大幅に上げることができます。たとえば、この項目をオフにした場合でも、1秒間に何百回もブールを更新することができます。これは人間が実際に認識できる限界を超えています。非同期の表示では、ユーザインタフェーススレッドによって自動的に更新の回数が減らされ、実行システムはVIの実行により多くの時間を費やすことができます。

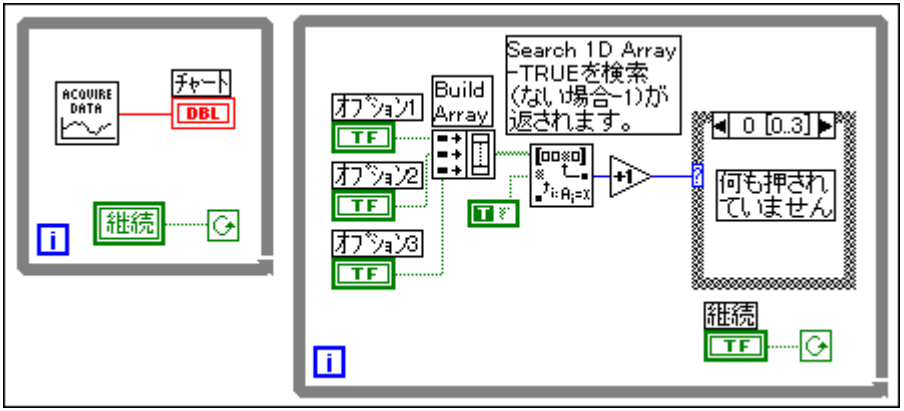
このような表示方法を使用したくない場合は、同期表示をオンにすることができます。

その他の問題

パラレルダイアグラム

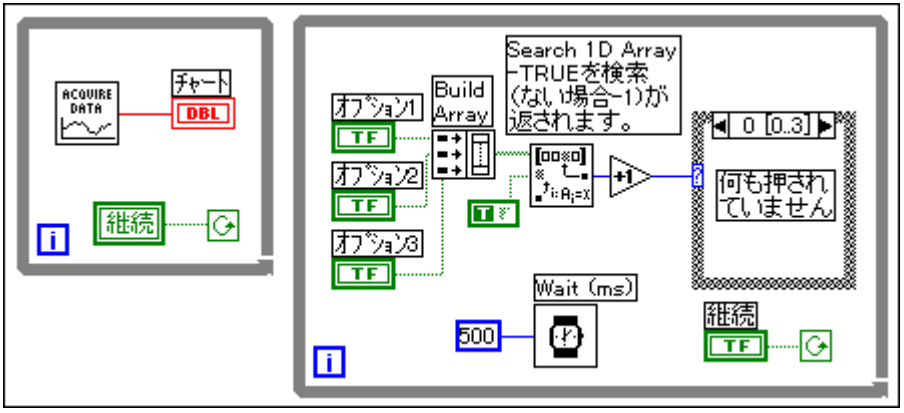
複数のダイアグラムを並列で実行する場合、実行システムは定期的に各ダイアグラムを切り替えます。これらのうちの一部のループが他のループよりも重要度が低い場合には、Wait関数を使用して重要度の低いループが使用する時間を減らします。

例として、次のダイアグラムについて考えてみます。



2つの並列ループがあります。1つはデータを集録するループで、できる限り頻繁に実行する必要があります。もう1つのループは、ユーザによる入力をモニタします。この2つのループは、プログラムの構造上どちらも同じ量の時間が割り当てられます。ユーザの動作をモニタするループは、1秒間に数回実行する機会が与えられます。

実際には、ボタンをモニタするループを0.5秒ごとに1回だけ、あるいはそれ以下の割合で実行されるようにしても、通常不都合はありません。ユーザインタフェースのループでWait (ms)関数を呼び出すことにより、もう一方のループにかなり多くの時間を割り当てることができます。



サブVIのオーバーヘッド

サブVIを呼び出す際には、その呼び出しに伴うある程度のオーバーヘッドが発生します。このオーバーヘッドは、特にミリ秒から10ミリ秒の単位で発生するI/Oのオーバーヘッドや表示のオーバーヘッドに比べると非常に小さなものです（規模は10マイクロ秒の単位）。

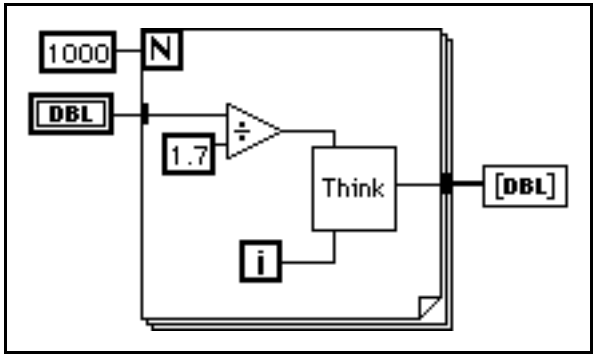
ただし、このオーバーヘッドは場合によっては累積することもあります。たとえば、ループ中でサブVIを10,000回呼び出す場合、このオーバーヘッドはかなりの時間長になります。このようなケースでは、ループをサブVIに内蔵できるかどうかも検討してみる必要があります。

もう1つの選択肢として、特定のサブVIを（VI設定の実行オプションにある**優先順位**項目を使用して）サブルーチンに変更する方法があります。VIがサブルーチンとして設定されていると、実行システムはサブVIを呼び出すためのオーバーヘッドを最小限に抑えます。ただし、それにはいくつかのデメリットもあります。サブルーチンは、フロントパネルのデータを表示することはできません（Gでは、データをサブルーチンのフロントパネル制御器にコピーしたり、フロントパネル制御器からコピーしたりすることはできません）。また、タイミング関数やダイアログボックス関数を含むことはできず、他のVIとの間でのマルテタスク処理も行いません。サブルーチンは、短くてかつ頻繁に実行されるタスクであるため、通常はユーザの介入を必要としないVIとともに使用するのに最も適しています。詳しくは、「第26章 Gの実行システムについて」を参照してください。

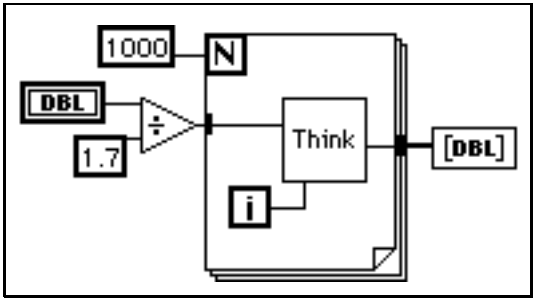
ループでの不必要な計算

毎回同じ結果を生成するような計算は、ループに入れるのを避ける必要があります。そのような計算はループの外に出し、結果をループに渡すようにします。

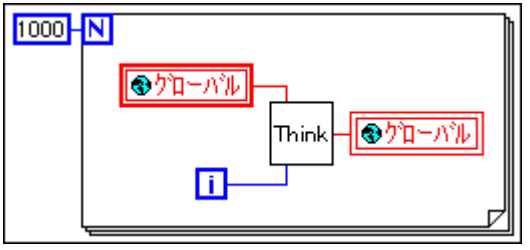
たとえば、次のダイアグラムを見てください。



ループの中にある割り算の結果は、毎回同じになります。したがって、次の図に示すように割り算をループの外に出すことによって、パフォーマンスを向上させることができます。

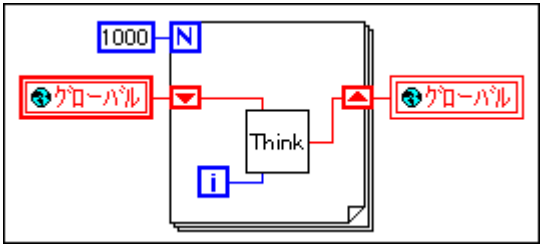


次に、次の図に示すダイアグラムを見てください。

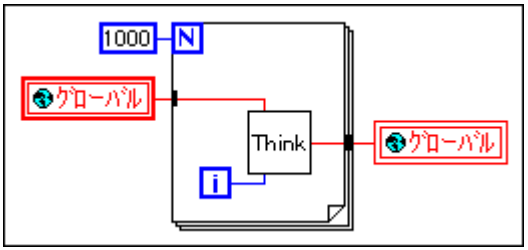


グローバル変数の値が、このループ内で並列に実行される別のダイアグラムあるいはVIによって変更されないことがわかっている場合、このダイアグラムでは、毎回ループを実行するたびにグローバル変数の読み取りやグローバル変数の書き込みを行うことは時間の無駄になります。

このループ内で別のダイアグラムがグローバル変数の読み取りや書き込みを行う必要がない場合は、代わりに次の図のようなダイアグラムを使用することができます。



ここでは、シフトレジスタがサブVIからループの次の回に新しいデータを渡さなければならない点に注意してください。次のダイアグラムは、初心者が犯しやすいミスを示したものです。ここにはシフトレジスタが存在しないため、サブVIの結果を新しい入力値として再びサブVIに渡すことができません。



VIによるメモリの使用

Gでは、従来のプログラミング言語においてプログラマが処理していたことをGが処理します。従来の言語における最も厄介な問題の1つに、メモリの使用の管理があります。従来の言語では、メモリを使用する前のメモリの割り当て、および使用後のメモリの解放をプログラマが管理する必要があります。またプログラマは、誤って最初に割り当てたメモリ領域を超えて書き込まないように特に注意する必要があります。メモリを割り当てていない、あるいは割り当てたメモリが十分でないといった問題は、従来の文字ベースの言語でプログラマが犯す最も重大なミスの1つです。メモリの割り当て不足も、デバックが難しい問題の1つです。

Gの自動メモリ処理により、メモリ管理の問題はほとんど解消されます。Gでは、変数の割り当てや、変数への、あるいは変数からの値の代入は行う必要はありません。かわりに、データの移行を表す配線を通してダイアグラムを作成します。

データを生成する関数は、そのデータを記憶するためのメモリの割り当てを処理します。データが不要になると、そのメモリは解放されます。配列や文字列に新しいデータを追加する際には、その新しいデータを管理するための十分なメモリが自動的に割り当てられます。

この自動メモリ処理は、Gの最も便利な点の1つです。ただし、この処理は自動的に実行されるため、その発生時期を人為的に制御できる度合いは低くなります。プログラムで大量のデータセットを使用する場合は、メモリの割り当てがいつ行われるかをある程度知っておくことが重要になります。そのような基本原理がわかっていると、プログラムが使用するメモリを大幅に減らすことができます。また、メモリの割り当てやデータのコピーにはかなりの時間を要するため、メモリの使用を最小限に抑える方法について理解することは、VIの実行速度を上げるのにも役立ちます。

仮想メモリ

メモリ量が制限されたマシンを使用する場合は、仮想メモリを使用してアプリケーションが使用できるメモリ量を増やすことを検討してみる必要があります。仮想メモリはオペレーティングシステムの機能の一種で、ディスクスペースをRAMとして使用することを意味します。大容量の仮想メモリを割り当てると、アプリケーションはそれを通常は記憶領域として使用可能なメモリとみなします。

アプリケーションにとっては、メモリが真のRAMであるか仮想メモリであるかは問題ではありません。メモリが仮想メモリであるという事実は、オペレーティングシステムによって隠されます。大きな違いは速度にあります。仮想メモリを使用した場合、オペレーティングシステムによって随時メモリからディスクに、あるいはディスクからメモリに切り替えられるため、パフォーマンスの低下がより顕著になる場合があります。仮想メモリを使用すると大規模なアプリケーションの実行が可能になりますが、時間にして大きな制約があるアプリケーションに使用するのには適しません。

Macintoshのメモリ

Macintoshでアプリケーションを起動する際には、システムがメモリの1つのブロックをそのアプリケーションに割り当て、そこからあらゆるメモリが割り当てられます。VIをロードする際には、VIのコンポーネントがメモリにロードされます。同様に、VIを実行する際には、そのVIが操作するすべてのメモリがこの1つのブロックのメモリから割り当てられます。

起動時にシステムが割り当てるメモリ量は、Finderから**ファイル→情報を見る**コマンドを使用して指定します。ただし、アプリケーションがメモリを使い果たした場合、このメモリプールのサイズを変更することはできません。したがって、この引数は実際的な値に設定する必要があります。16MBマシンを使用している場合は、アプリケーションを並列で実行することを検討してみる必要があります。他のアプリケーションを実行する予定がない場合は、メモリの環境設定をできるだけ大きな値に設定します。

VIコンポーネント用メモリの管理

VIには、4つの主要なコンポーネントがあります。

- フロントパネル
- ブロックダイアグラム
- コード（ダイアグラムからコンパイルしたマシンコード）
- データ（制御器および表示器の値、デフォルトデータ、ダイアグラムの定数データなど）

VI をロードする際には、その VI のフロントパネル、コード（プラットフォームに適合する場合）、およびデータがメモリにロードされます。プラットフォームを変更したり、サブVIに対するインタフェースを変更したためにVIをコンパイルし直す必要がある場合は、ダイアグラムもメモリにロードされます。

VIは、そのサブVIのコードやデータスペースもメモリにロードします。場合によっては、一部のサブ VI のフロントパネルもメモリにロードされます。それは、たとえば属性ノードがフロントパネル制御器のステータス情報を操作するという理由で、サブVIが属性ノードを使用している場合などです。サブ VI およびサブ VI のパネルについての詳細は、この章で後ほど説明します。

VIのコンポーネントの構成で重要な点は、VIの一部をサブVIに変換する場合でも、使用するメモリ量がそれより大幅に増えることは通常ないという点です。サブ VI を持たない 1 つの大きな VI を作成すると、その最上位のVIのフロントパネル、コード、およびデータだけでメモリを使い果たしてしまいます。一方、VIをサブVIに分割すると、最上位のVIのコードは小さくなり、サブ VI のコードとデータをメモリに入れることができます。場合によっては、実行時のメモリの使用量が少なくなることもあります。この概念については、本章の「メモリの使用状況を監視する」の項で説明します。

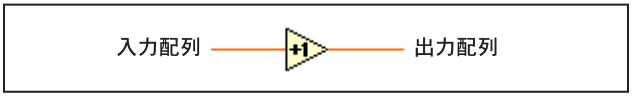
また、大きな VI は編集するにも時間がかかります。エディタは小さい VI をより効率的に処理できるため、VI をサブ VI に分割するとこのような問題は発生しません。さらに、VIの構成を階層化するほど維持や読み取りも容易になります。

☞ **注**　**VIのフロントパネルやブロックダイアグラムが画面よりもはるかに大きい場合は、サブVIに分割すると見やすくなります。**

データフローのプログラミングおよびデータバッファ

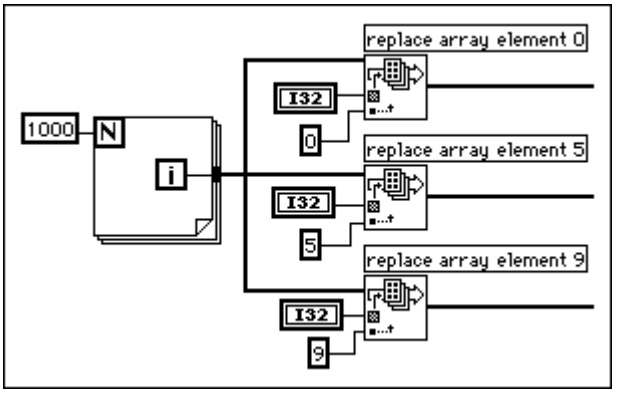
データフローをプログラミングする場合、通常変数は使用しません。データフローのモデルは通常ノードを、データ入力を消費し、データ出力を生成するものとして記述します。このモデルをそのまま使用すると、大量のメモリを使用し、かつパフォーマンスの低いアプリケーションが作成される可能性があります。また、すべての関数は、出力を渡すそれぞれの相手にデータのコピーを作成することになります。G コンパイラは、メモリが再使用できるようになる時期を特定し、出力を渡す相手をチェックして個々の端子に対してコピーを作成する必要があるかどうかを判断することにより、モデルを最適な方法で使用します。

たとえば、コンパイラをより従来的な方法で使用した場合、次のダイアグラムはデータメモリのブロックを入力と出力にそれぞれ 1 つずつ、計 2 ブロック使用します。



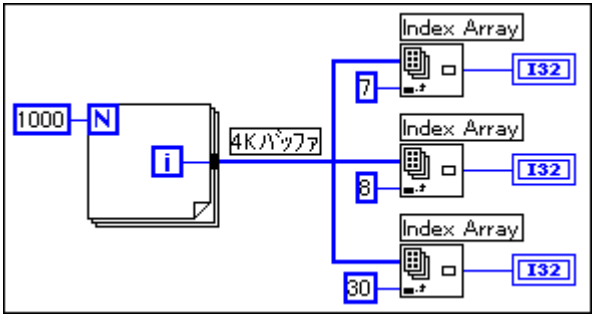
入力配列と出力配列の要素の数は同じで、両方の配列のデータタイプも同じです。ここで、入ってくる配列をデータバッファとして考えます。コンパイラは、出力に対して新しいバッファを作成する代わりに、入力バッファを再使用します。そうすると、実行時にメモリの割り当てを行う必要がなくなるため、メモリを節約できるだけでなく、同時に実行速度も上げることができます。

ただし、次の図に示すように、コンパイラはいつもメモリバッファを再使用できるとは限りません。



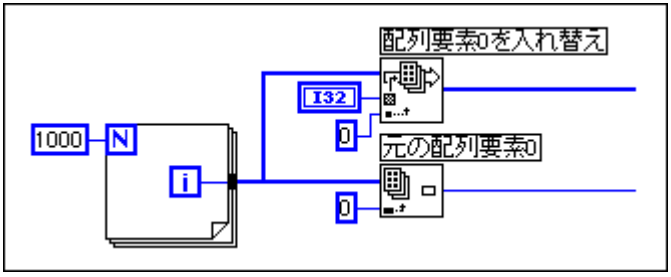
1つのデータソースを複数の送り先に渡すものとします。送り先では、データを修正して配列を生成しようとします。この場合、コンパイラは2つの関数用のデータバッファを作成し、配列データをバッファにコピーします。このように、関数の1つは入力配列を再使用しますが、他の関数はそうはしません。このダイアグラムは、約12KB（オリジナルの配列用の4 KBと、2つの追加バッファ用にそれぞれ4 KB）を使用します。

今度は、次のダイアグラムを見てください。



上記と同様、入力は3つの関数に分岐しています。ただし、ここではどの関数もデータの修正を必要としていません。データを複数の場所に送ると、送り先ではデータを修正せずに読み取ります。Gは、データのコピーは作成しません。このダイアグラムは、約4KBのデータを使用します。

最後に、次のダイアグラムについて考えてみます。



ここでは、入力は2つの関数に分岐しており、その一方の関数はデータを修正しようとします。2つの関数の間に、依存性はありません。したがって、Replace Array Element関数がデータを安全に修正できるように、データコピーを少なくとも1つ作成する必要があることが予測できます。ただし、コンパイラは、データを読み取る関数を最初に実行し、データを修正する関数を最後に実行するように関数の実行スケジュールを作成します。このように、Replace Array Element関数は配列のコピーを生成せず、入力配列バッファを再使用します。ノードの実行順序が重要になる場合は、シーケンスを使用するか、または一方のノードの出力をもう一方のノードの入力として使用することにより、順序を明示的に指定します。

実際は、コンパイラによるダイアグラムの分析は完璧ではありません。ときには、コンパイラはダイアグラムのメモリを再使用するための最適な方法を特定できない場合もあります。

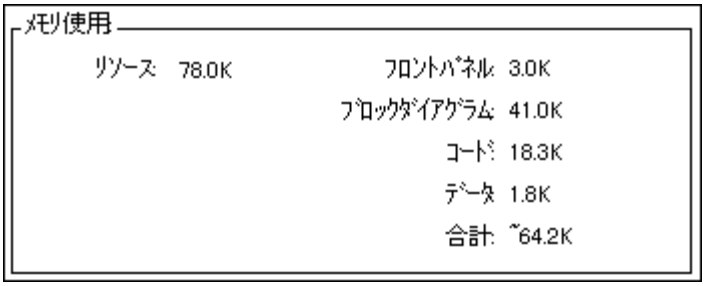
メモリの使用状況を監視する

メモリの使用状況を特定する方法はいくつかあります。

ヘルプ→ LabVIEW について ...（LabVIEW を使用している場合）または Help → About BridgeVIEW...（BridgeVIEW を使用している場合）を選択すると、アプリケーションが使用する総メモリ量をまとめた統計データを見ることができます。このメモリには、アプリケーションが使用するメモリだけでなく、VI のためのメモリも含まれます。VI のセットを実行する前、および実行した後にこのメモリ量をチェックすることにより、VIが使用するおおよそのメモリ量を把握することができます。

パフォーマンスプロフィールでは、VIによる動的なメモリの使用状況を知ることができます。パフォーマンスプロフィールには、それぞれのVIが1回の実行で使用するバイト数とブロック数の最小値、最大値、および平均値が記録されます。詳しくは、本章冒頭の「VI のパフォーマンスのプロフィール」の項を参照してください。

次の図に示すように、**ウインドウ→VI 情報を表示 ...** を使用すると、あるVIが使用するメモリの内訳を見ることができます。この情報の左の欄にはディスクの使用量が表示され、右の欄にはVIのさまざまなコンポーネントが現在使用しているRAMの量が表示されます。ただし、これらのデータにはサブVIが使用するメモリ量は含まれません。



メモリの使用量を知る第4の方法は、Memory MonitorVIというVIを使用する方法です(このVIは`examples`ディレクトリ内の`memmon.llb`に入っています)。このVIは、VI Server関数を使用してメモリ内のすべてのVIのメモリ使用量を特定します。

メモリを効率的に使用するための規格

前項の説明では、コンパイラがメモリを知的な方法で再使用する点が主なポイントでした。コンパイラがメモリを再使用できる場合とそうでない場合についての規格はかなり複雑なため、本章の最後で説明します。実際には、メモリを効率的に使用するVIの作成には、次の規格が役立ちます。

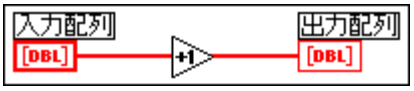
- VIをサブVIに分割しても、メモリの使用に悪影響を及ぼすことは通常ありません。実際には、サブVIを実行しないときには実行システムがサブVIのデータメモリの返還を要求するため、多くの場合メモリの使用効率が向上します。
- スカラ値のコピーに関して特に注意する必要はありません。多くのスカラを使用しない限りメモリの使用に悪影響を及ぼしません。
- 配列や文字列を使用する際に、グローバル変数やローカル変数を過度に使用しないように注意します。グローバル変数やローカル変数の読み取りは、変数のデータのコピーが生成される原因となります。
- 必要な場合以外は、開いているフロントパネル上に長大な配列や文字列を表示しないようにします。開いているフロントパネル上の表示器は、表示しているデータのコピーを保持します。
- サブVIのフロントパネルを表示しないときは、サブVI上に使用しない属性ノードを放置しないようにします。属性ノードが存在すると、サブVIのフロントパネルがメモリ内に残されるため、メモリが不必要に消費される原因になります。

- 時間やメモリの使用が重要視されるVIでは、“入力が範囲外の場合中断する機能”を使用しないようにします。サブVIのフロントパネルは範囲チェックを行うためにロードすることが必要になり、サブVIの制御器や表示器のデータのコピーが作成されることになります。
- ダイアグラムを設計する際には、入力のサイズと出力のサイズが異なる部分に注意します。たとえば、Build Array関数やConcatenate Strings関数を使用して配列や文字列のサイズを頻繁に増やすような場所では、データのコピーが作成されることになります。
- 配列には一貫したデータタイプを使用し、サブVIや関数にデータを渡す際に強制ドットがないか注意します。データタイプを変更すると、実行システムがデータのコピーを作成します。
- 複雑で階層的なデータタイプは使用しないようにします（たとえば長大な配列や文字列を含むクラスタの配列や、長大な配列や文字列クラスタ）。メモリの使用量が増え、パフォーマンスが低下する原因になります。データタイプの設計方法についての詳細は、本章の「効率的なデータ構造を作成する」の項を参照してください。

フロントパネルのメモリに関する問題

フロントパネルが開いているとき、制御器や表示器はそれぞれが表示するデータの専用コピーを保持します。

次の図は、フロントパネルの制御器と表示器を追加した増分関数を示したものです。



VIを実行すると、フロントパネル制御器のデータがダイアグラムに渡されます。増分関数は入力バッファを再使用します。表示器は、表示の目的でデータのコピーを作成します。このようにして、バッファのコピーが3個存在することになります。

このようにフロントパネル制御器によってデータが保護されている場合は、ユーザは制御器にデータを入力したり、関連するVIを実行したり、後ろのノードにデータが渡された際の制御器のデータの変化を見たりすることはできません。同様に、表示器においても、新しいデータを受け取るまでは前の内容が確実に表示されるようにデータが保護されます。

サブVIでは、制御器および表示器を入力および出力として使用することができます。次のような場合には、実行システムがサブVIの制御器および表示器のデータのコピーを作成します。

- フロントパネルがメモリに入っている ― これは以下の理由によって発生します。
  - フロントパネルが開いている。
  - VIを変更した後、まだ保存していない(VIのすべてのコンポーネントはVIが保存されるまではメモリに残されます)。
  - パネルがデータの印刷を使用する。
  - ダイアグラムが属性ノードを使用する。
- VIがローカル変数を使用する。
- パネルがデータロギングを使用する。
- 制御器が“範囲外の場合中断する機能”を使用する。

上記の理由の中にはわかりにくいものもあるため、詳しい説明が必要になります。

理由の 1 つは、チャートの履歴などの属性に関与しています。属性ノードがパネルを開いていないサブVIのチャートの履歴を読み込むためには、制御器または表示器が受け取ったデータを表示する必要があります。このような属性は他にも多数あるため、サブ VI が属性ノードを使用する場合は、実行システムがサブVIのパネルをメモリに残したままにします。

フロントパネルがフロントパネルデータロギングまたはデータ印刷を使用する場合は、制御器および表示器はそれぞれのデータのコピーを保持します。また、データ印刷でパネルを印刷できるように、パネルはメモリに残されます。

VIが“範囲外の場合中断する機能”を使用する場合は、フロントパネルのすべての制御器および表示器との間でデータがコピーされます。いずれかのデータが範囲を超えた場合にフロントパネルを表示できるように、フロントパネルの値が保持されます。メモリの使用量や速度が重要な問題となるような場合は、"このチェック機能"は使用しないようにしてください。

VI設定またはサブVI設定を使用して、サブVIが呼び出されたときにそのフロントパネルが表示されるように設定されていると、そのサブVIは呼び出されたときにフロントパネルがメモリにロードされる点に注意してください。**元に閉じてあったら閉じる**項目が設定されている場合は、サブVIの実行が終了した時点でパネルはメモリから削除されます。

サブVIはデータメモリを再使用できます

通常、サブVIはそのダイアグラムが最上位にコピーされているのと同じ要領で、発呼者のデータバッファを簡単に使用することができます。ほとんどのケースでは、ダイアグラムの一部をサブVIに変換する際により多くのメモリが必要になることはありません。ただし、特殊な表示条件を持つVIについては、前の項で述べたようにフロントパネルや制御器用の追加メモリが必要になる場合があります。

ローカル変数はデータメモリを再使用できません

サブVIを作成する際には、サブVIとの間でのデータの受け渡し方法を定義するコネクタペーンを作成します。コネクタペーンに接続された端子から渡されるデータバッファは、発呼者VIからのデータバッファを再使用することができます。ローカル変数はそれができないため、ローカル変数を読み取る際には代わりに制御器のデータ用のバッファを作成します。

ダイアグラム上のある場所から別の場所にローカル変数を使用して大量のデータを転送する場合、通常はそれだけ多くのメモリを使用することになるため、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遅くなります。

グローバル変数は常にデータのコピーを保持する

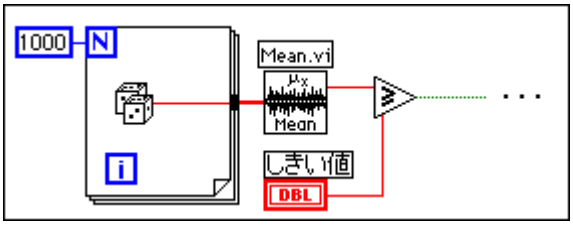
グローバル変数を読み取る際は、そのグローバル変数に記憶されたデータのコピーを作成します。

長大な配列や文字列を操作する場合、グローバル変数を操作するためにかなりの時間とメモリを要する場合があります。これは、配列の1つの要素だけを修正して配列全体を保存するような場合には、特に非効率的です。ダイアグラム上の複数の場所でグローバル変数を読み取る場合は、複数のメモリバッファを作成することが必要になる場合があります。

そのような場合のメモリの使用量を最小限に抑える1つの方法として、初期化されないシフトレジスタをサブVIで使用してデータを記憶する方法があります。これは、グローバル変数の簡潔さとシフトレジスタの効率性を活用した方法です。詳しくは、本章の「ケーススタディ2：データタイプの混在するグローバルテーブル」の項を参照してください。

メモリの再割り当てのタイミングを理解する

次のダイアグラムについて考えてみましょう。



Mean VI の実行後は、データの配列は不要になります。大きなダイアグラムでは、どの時点でデータが不要になるかを特定することは非常に面倒であるため、ある VI の実行中はその VI のデータバッファは解放されません。

Macintosh では、実行システムのメモリが足りない場合、実行システムは実行中でない VI が使用するデータバッファを解放します。実行システムは、フロントパネルの制御器、表示器、グローバル変数、あるいは初期化されないシフトレジスタが使用するメモリは解放しません。

次に、上記と同じ VI をより大きな VI に対するサブ VI として考えてみましょう。データの配列は、サブ VI でのみ作成され、使用されます。Macintosh では、サブ VI が実行中でなく、実行システムのメモリが足りない場合には、実行システムはサブ VI のデータを解放します。このようなケースでは、サブ VI を使用することによってメモリの使用量を節約することができます。

Windows や UNIX のプラットフォームでは、VI を閉じメモリから削除するまで、通常データバッファは解放されません。これらのプラットフォームでは、メモリはオペレーティングシステムから必要に応じて割り当てられ、仮想メモリが正しく機能します。メモリが断片化されている場合は、アプリケーションは実際に使用するよりも多くのメモリを使用しているように見えることがあります。メモリの割り当ておよび解放に際しては、アプリケーションはメモリの使用領域を統合し、使用していないブロックをオペレーティングシステムに返します。

どのプラットフォームにおいても、オプションとして“メモリをなるべく早く解放する”を使用することができます。この項目がオンになっていると、サブ VI は実行が終了すると直ちにメモリを解放します。これは、メモリの使用効率を上げるのには役立ちますが、一方でパフォーマンスの大幅な低下を招くおそれがあります。

出力が入力バッファを再使用するタイミングを決定する

出力のサイズとデータタイプが入力と同じで、かつその入力が他の場所で必要とされない場合は、出力は入力バッファを再使用することができます。上で述べたように、入力が他の場所で使用される場合でも、コンパイラや実行システムが入力を出力バッファとして使用できるように、コードの実行順序を設定できるケースもあります。ただし、そのようなケースでは規格が複雑になるため、それらに頼ることはできません。

一貫したデータタイプの使用

入力のデータタイプが出力のデータタイプと異なる場合は、出力は入力を再使用することはできません。たとえば、32 ビット整数を 16 ビット整数に追加する場合、16 ビット整数が 32 ビット整数に変換されることを示す強制ドットが表示されます。32 ビット整数の入力は、他のすべての条件（32 ビット整数を別の場所で再使用しないなどの条件）を満たすものと想定される場合は、出力バッファとして使用することができます。

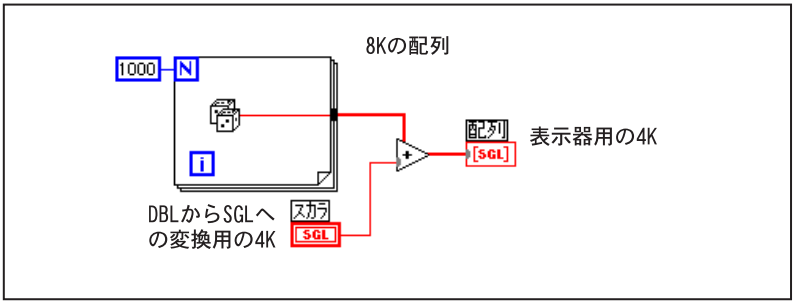
また、サブ VI の強制ドットや多くの関数は、データタイプが変換されることを暗に示しています。通常、コンパイラは変換されたデータに対して新しいバッファを作成します。

メモリの使用量を最小限に抑えるためには、できるだけ一貫して同じデータタイプを使用する必要があります。そうすることで、データのサイズを大きくし、データのコピーの数を減らすことができます。また、一貫したデータタイプを使用することにより、データタイプを再使用するタイミングをコンパイラが柔軟に決定できるようになります。

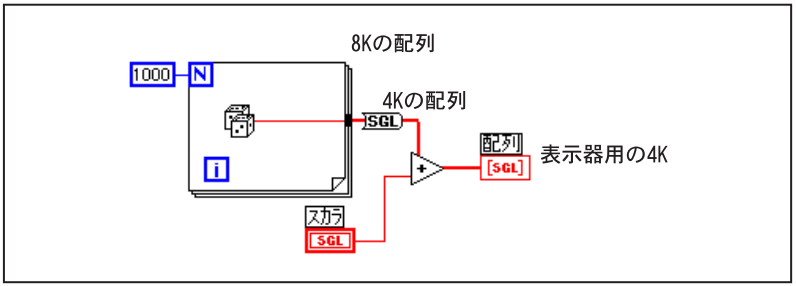
アプリケーションによっては、小さいデータタイプの使用を検討することが必要になる場合があります。たとえば、8 バイト倍精度の数字の代わりに 4 バイト単精度の数字を使用することが必要になることもあります。ただし、不必要な変換を避けるため、呼び出されるサブ VI がどんなデータタイプを必要としているかを慎重に検討する必要があります。

正しいタイプのデータの生成方法

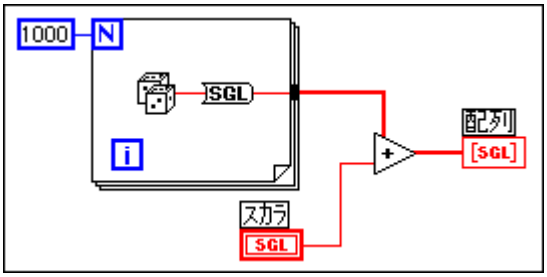
次に示す例を見てください。この例では、1,000 個の乱数値の配列が作成され、スカラに追加されます。Add 関数に強制ドットが表示されているのは、乱数データが倍精度であるのに対し、スカラが単精度であるためです。スカラは、追加に先立って倍精度に変更されます。さらに、生成されたデータは表示器に渡されます。このダイアグラムは、16 KB のメモリを使用します。



次の図は、倍精度の乱数の配列を単精度の乱数の配列に変換することで問題を解決しようとしていますが、これは正しい解決方法ではありません。使用するメモリ量は、前に示した例と変わりません。



最も良い解決方法は、次の図に示すように、乱数を作成時に単精度に変換してから配列を作成する方法です。この方法を使用すると、大きなデータバッファのデータタイプを変換する必要はなくなります。

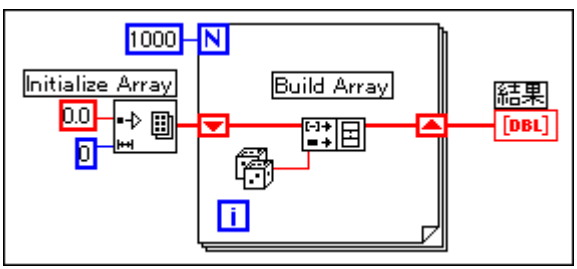


データサイズの頻繁な変更を避ける

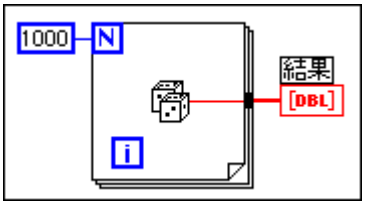
出力のサイズが入力サイズと異なる場合、出力は入力のデータバッファを使用しません。配列や文字列のサイズを大きくしたり小さくしたりするBuild Array、String Concatenate、あるいはArray Subsetといった関数の場合は、このケースに相当します。配列や文字列を使用する場合は、プログラムが使用するメモリ量が増え、またデータを頻繁にコピーすることによって実行速度が低下する原因となるため、これらの関数を頻繁に使用することは避ける必要があります。

例1：配列を作成する

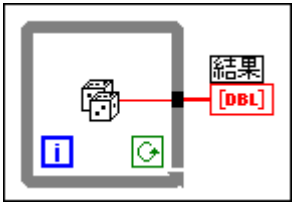
ここでは、データの配列を作成するために使用される次のダイアグラムを例にとって考えてみましょう。このダイアグラムは、新しい要素を結合するBuild Arrayをループ内で定期的に呼び出すことによって配列を作成します。Build Arrayは、入力配列を再使用しません。その代わり、VIは新しい配列を格納できるように出力バッファのサイズをたえず変更しながら、古い配列から新しい配列にデータをコピーします。その結果、実行速度は低下し、特にループの実行回数が多くなるほど速度の低下が顕著になります。



ループの各回の実行ごとに配列に値を追加する場合は、ループのエッジで自動指標付けを使用することによって最大のパフォーマンスを実現することができます。For ループでは、VI は（N に配線された値に基づいて）配列のサイズを前もって特定できるため、バッファのサイズを 1 回変更するだけで済みます。



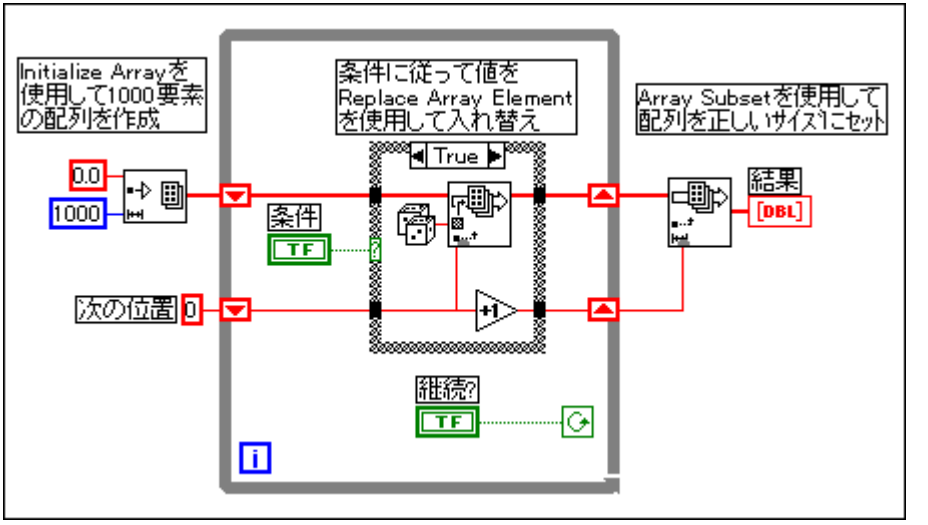
While ループでは、配列の最終的なサイズを特定できないため、自動指標付けはそれほど効率的であるとはいえません。ただし、While ループの自動指標付けでは、出力配列のサイズを大幅に拡張しておけば毎回出力配列のサイズを変更する必要はなくなります。出力配列のサイズは、ループが終了すると適切なサイズに調整されます。While ループの自動指標付けのパフォーマンスは、For ループの自動指標付けとほとんど同じです。



自動指標付けは、ループの各回の実行でそのつど配列に値を追加することを想定しています。一定の条件が満たされた場合にのみ値が配列に追加されるようにしたいときに、配列のサイズの上限を特定できる場合は、あらかじめ配列を割り当てたうえで、Replace Array Element を使用して配列に値を格納する方法を使用できます。

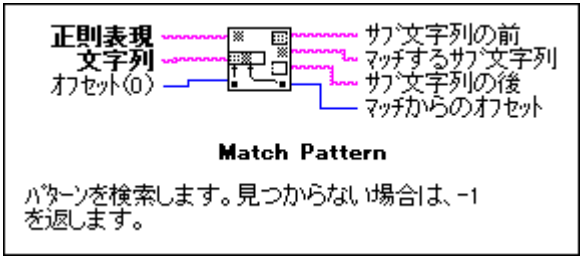
配列にすべて値を格納し終えたら、配列を適切なサイズに変更します。この方法では、配列を一回作成するだけで済み、Replace Array Element は入力バッファを出力バッファとして使用することができます。この場合のパフォーマンスは、自動指標付けを使用したループのパフォーマンスと非常に似ています。。この方法を使用する場合、Replace Array Element はユーザに代わって配列のサイズを変更することはできないため、値を入れ替える配列のサイズを、最終的なデータを格納できるように十分な大きさに設定しておく必要があります。

このプロセスの例を次の図に示します。



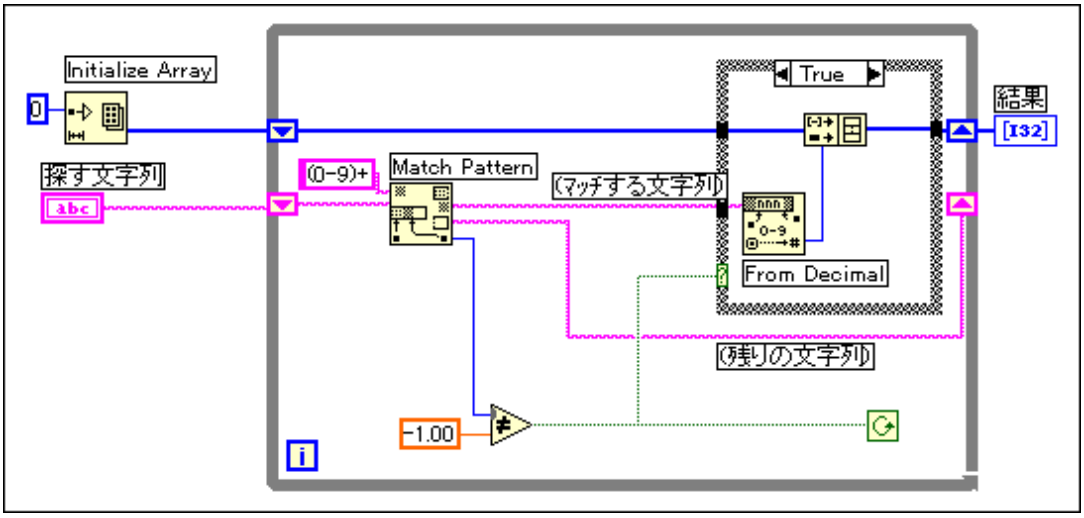
例2：文字列中で一致するパターンを検索する

Match Pattern関数を使用すると、文字列中で一致するパターンを検索することができます。ただし、関数の使用方法によっては、不必要な文字列のデータバッファが作成され、パフォーマンスの低下を招くおそれがあります。次の図は、Match Patternのヘルプウィンドウを示したものです。



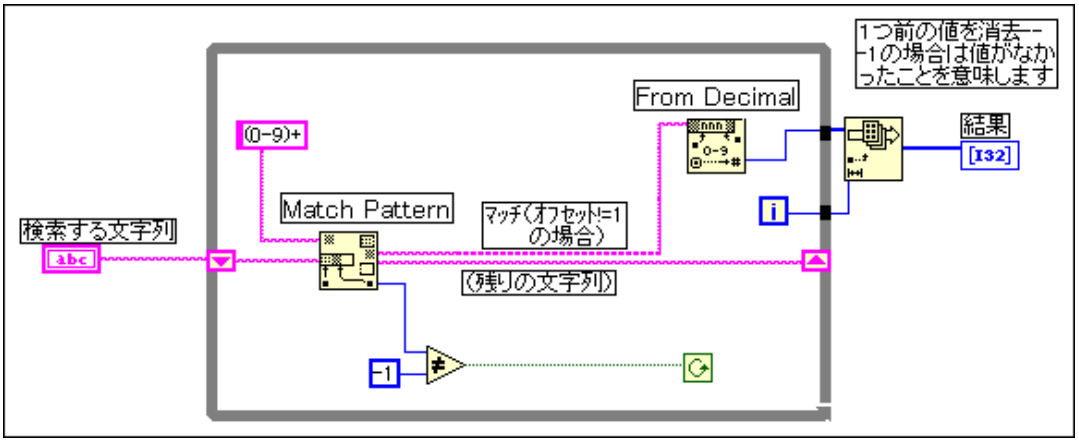
文字列中で整数を検索する場合、この関数への通常の入力式として[0-9]+を使用します。文字列に含まれるすべての整数の配列を作成するためには、ループを使用して、オフセット値が–1になるまで繰り返しMatch Patternを実行します。

次のダイアグラムは、文字列中に存在するすべての整数をスキャンする1つの方法を示したものです。このダイアグラムは、まず最初に空白の配列を作成し、ループの毎回の実行のたびに残りの文字列で数値パターンを検索します。パターンが見つかると（オフセットは–1ではない場合）、ダイアグラムはBuild Arrayを使用してその数字を数字の配列に追加します。文字列に残っている値がなくなると、Match Patternは–1を返し、ダイアグラムの実行が終了します。

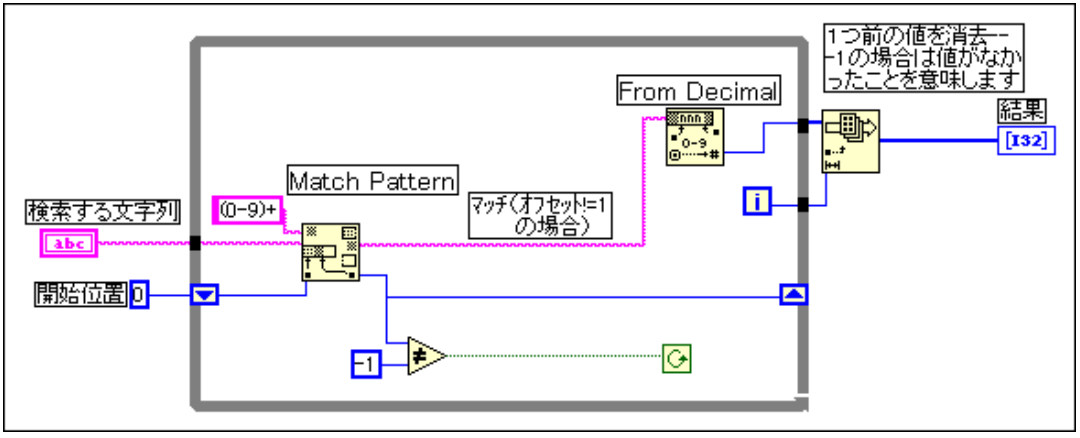


このダイアグラムの1つの問題は、ループ内でBuild Arrayを使用して新しい値を前の値に連結するという点です。代わりに自動指標付けを使用すると、ループのエッジに値を積み重ねることができます。ただし、Match Patternが該当する文字を検出できなかった場合は、ループの最後の実行で生成された不要な値が1つ余分に配列に追加される点に注意してください。この問題は、配列のサブセットを使用して余分な値を削除することで解決できます。これを次の図に示します。



このダイアグラムのもう 1 つの問題は、ループを実行するたびに残りの文字列の不必要なコピーが作成される点です。Match Pattern には、検索をどこから開始するかを指定するのに使用できる入力があります。前回の実行のオフセットを記憶していれば、その数字を使用して次の実行でどこから検索を開始するかを指定することができます。この方法を次の図に示します。

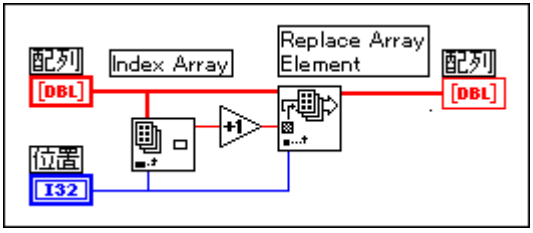


効率的なデータ構造を作成する

前の項での考察から明らかになった点の 1 つは、長大な配列や文字列を含むクラスタの配列、あるいは長大な配列や文字列を含むクラスタのような階層的なデータ構造は、効率的に操作することができないという点です。この項では、その理由を述べるとともに、より効率的なデータタイプを選択する方法について説明します。

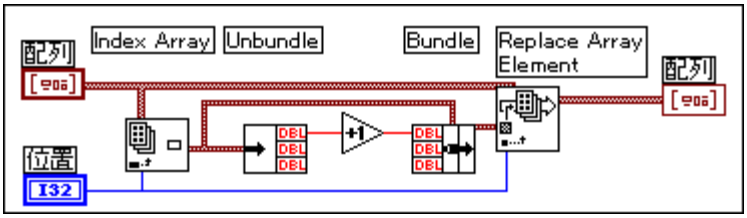
複雑なデータ構造における問題は、データ構造中の要素は、その要素のコピーを作成せずにアクセスしたり変更したりすることが難しいという点です。たとえば、要素そのものが配列であったり文字列であったりする場合のように、これらの要素のサイズが大きい場合は、余分なコピーを作成することによってメモリの使用量が増え、またメモリをコピーする時間も余分にかかります。

通常、データタイプがスカラ値である場合はかなり効率的に処理することができます。同様に、小さな文字列や、スカラを要素として使用している小さな配列は、効率的に処理することができます。スカラの配列の場合、配列中の値を繰り上げるために実行すべき操作をコードで示すと次のようになります。



このコードが効率的な理由は、配列全体のコピーを余分に作成する必要がない点にあります。また、Index Array 関数によって生成される要素が、作成や処理が容易なスカラであることもその理由の 1 つです。

このことは、クラスタがスカラだけで構成されているものと仮定すれば、クラスタの配列にもあてはまります。次のダイアグラムでは、Unbundle 関数や Bundle 関数を使用する必要があるため、要素の操作はやや複雑になります。ただし、クラスタが小さければ（スカラはわずかなメモリしか使用しません）、クラスタの要素にアクセスしてそれらの要素を元のクラスタに戻す場合でも、大きなオーバーヘッドは発生しません。



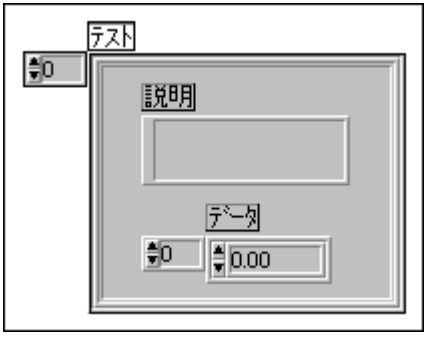
クラスタの配列において個々のクラスタが長大なサブ配列や文字列で構成されている場合、このようなクラスタの要素に指標を付けたり要素の値を変更すると、メモリや速度の面で多大な犠牲を払うことになります。

配列全体を通して要素に指標を付ける際には、その要素のコピーが作成されます。すなわち、クラスタ、およびそれに相当するサブ配列あるいは文字列のコピーが作成されます。文字列や配列のサイズは可変であるため、コピーのプロセスには、文字列やサブ配列を実際にコピーする際のオーバーヘッドに加え、適正なサイズの文字列やサブ配列を作成するためのメモリを割り当てる作業も加わります。このような操作は、実行する回数が少なければそれほど重要な問題とはなりません。ただし、アプリケーションがこのデータ構造に集中的に何度もアクセスするような場合は、メモリの消費量や実行のオーバーヘッドが急速に増加するおそれがあります。

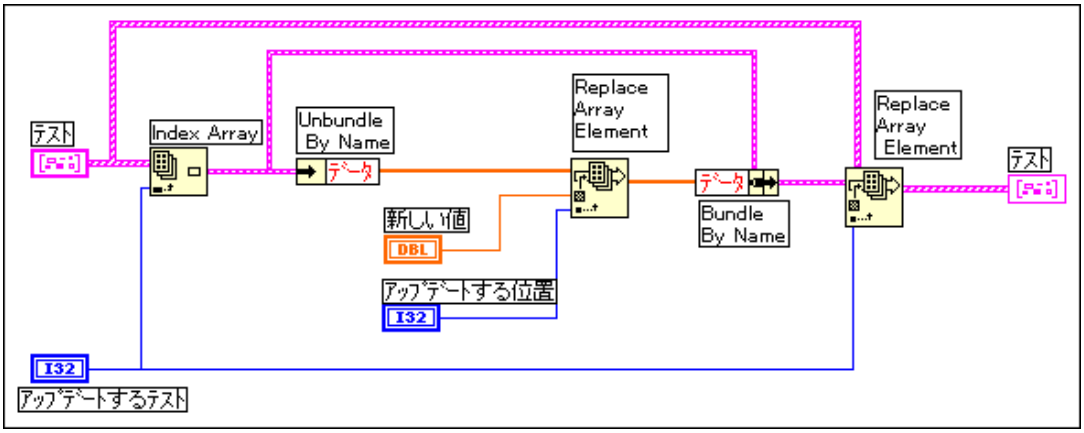
このような問題の解決策としては、別のデータ表記法を使用する方法があります。以下では、異なるアプリケーションを使用する 3 つのケースを例にとり、それぞれのケースに最も適したデータ構造を紹介します。

ケーススタディ 1：複雑なデータタイプを避ける

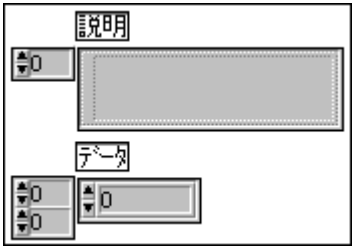
ここでは、複数のテストの結果を記録するアプリケーションについて考えます。テスト結果のデータは、テストを記述した文字列と、テスト結果を格納した配列で構成されるものとします。このようなデータを記憶するのに使用できる 1 つのデータタイプの例を次の図に示します。



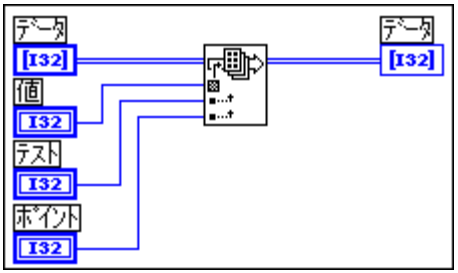
配列中の要素を変更するためには、配列全体を通して要素に指標を付ける必要があります。次に、そのクラスタに対して要素を分解し、配列にアクセスします。さらに、要素を入れ換え、得られた配列をクラスタに保存します。最後に、得られたクラスタを元の配列に保存します。その例を次の図に示します。



分解／指標付けの各段階において、そのデータのコピーが生成される可能性があります。ここでは、コピーは必ずしも生成されるわけではないことに注意してください。データをコピーすると、時間もメモリも消費されます。この場合の解決策としては、データ構造をできるだけ平坦にする（階層的でなくする）方法があります。たとえば、ここではデータ構造を 2 つの配列に分解することができます。最初の配列は、文字列の配列です。もう 1 つの配列は、それぞれの行に 1 つのテストの結果を記憶した 2 次元配列です。分解した結果を次の図に示します。



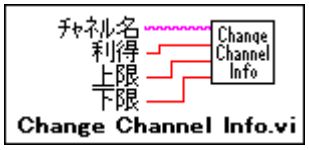
このようなデータ構造にすると、次の図に示すように、Replace Array Element関数を使用して配列の要素を直接入れ換えることができます。



ケーススタディ２：データタイプの混在するグローバルテーブル

ここでは、データをテーブル（表）として記憶するアプリケーションについて考えます。このアプリケーションでは、データにグローバルな形でアクセスするものとします。このようなテーブルには、ゲイン、電圧の上限および下限、チャネル名など、計測器の設定データを格納することができます。

アプリケーション全体を通してデータにアクセスできるようにするためには、テーブルのデータにアクセスするためのサブVIのセット（Change Channel Info VIやRemove Channel Info VIなど）を作成することを検討してみる必要があります。



以下の項では、これらのVIの3通りの使用方法を示します。

わかりやすい方法

この関数のセットを使用する場合、テーブルのデータ構造としていくつかの種類が考えられます。まず最初に考えられるのは、クラスタの配列を含むグローバル変数を使用する方法です。それぞれのクラスタには、ゲイン、上限、下限、チャネル名を格納します。

このようなデータ構造は、前の項でも説明したように、データにアクセスするためには何段階かの指標付けと分解が必要となるため、効率的に操作することは困難です。また、このデータ構造は複数の断片的なデータの集合体であるため、Search 1D Array関数を使用してチャネルを検索することはできません。Search 1D Arrayは、クラスタの配列で特定のクラスタを検索するために使用することはできますが、1つのクラスタ要素と一致する要素の検索には使用できません。

代替方法1

前で示した例と同じように、データを2つの別々の配列に分ける方法です。一方の配列にはチャネル名を入れます。もう一方の配列にはチャネルのデータを入れます。チャネル名の配列の中で個々のチャネル名に付けた指標を使用して、もう一方の配列でそのチャネルのデータを検索することができます。

この場合、文字列の配列がデータとは別になっているため、Search 1D Array関数を使用してチャネルを検索することができます。

実質的には、Change Channel Info VIを使用して1,000個のチャンネルからなる配列を作成する場合、この方法は前の方法より約2倍の速さになります。ただし、他にもパフォーマンスに影響を及ぼすオーバーヘッドが存在するため、この程度では著しく改善されたとはいえません。

本章の「VIによるメモリの使用」項で述べたように、ローカル変数およびグローバル変数の過度の使用には注意が必要です。グローバル変数から読み取る際には、グローバル変数のデータのコピーが生成されます。そのため、配列の要素にアクセスするたびに、配列の完全なコピーが生成されることになります。次の例では、このようなオーバーヘッドを回避するためのさらに効率的な方法を紹介します。

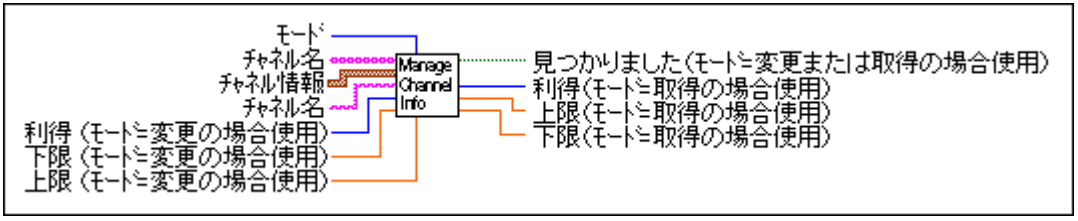
代替方法2

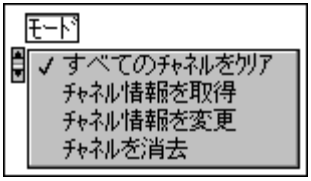
グローバルなデータを保存するもう1つの方法は、初期化されないシフトレジスタを使用する方法です。本来、初期値を配線していなければ、シフトレジスタはいつ呼び出してもその値を保持します。初期化されないシフトレジスタについての知識がない方は、LabVIEW をご使用の場合は『LabVIEW ユーザマニュアル』の「第3章 ループとチャート」をお読みになったうえで先にお進みください。また、BridgeVIEWをご使用の場合は、

『BridgeVIEW User Manual』の「Chapter 10, Loops and Charts」をお読みください。

G コンパイラは、シフトレジスタへのアクセスを効率的に処理します。シフトレジスタの値を読み取る場合、必ずしもデータのコピーが生成されるとは限りません。実際には、シフトレジスタに記憶されている配列に指標を付けたり、配列全体の余分なコピーを生成することなく配列の値を変更したり更新することさえできます。シフトレジスタを使用する場合の問題は、シフトレジスタのデータにはシフトレジスタを含むVIしかアクセスできない点です。ただし、シフトレジスタはモジュール性において非常に優れた面があります。

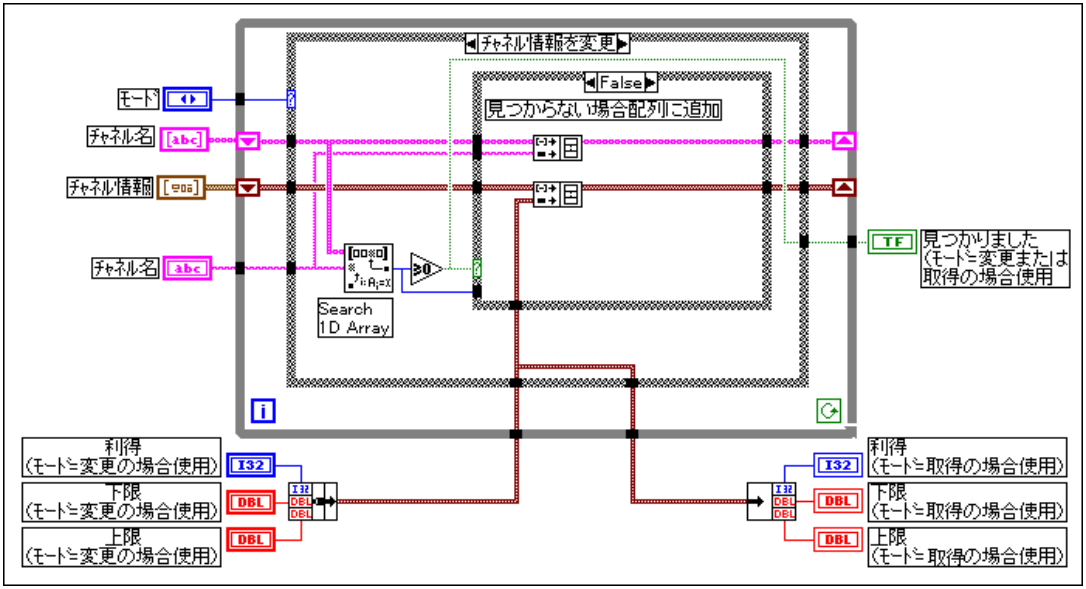
この場合にできることは、次の図に示すように、チャネルを読み取るのか、変更するのか、削除するのか、あるいはすべてのチャネルのデータを 0 にするのかを指定するためのモード入力を備えた1つのサブVIを作成することです。





サブVIには、2つのシフトレジスタを持つWhileループがあります。一方のシフトレジスタはチャネルデータを記憶し、もう一方はチャネル名を記憶します。この 2 つのシフトレジスタは、いずれも初期化されません。次に、Whileループの内部に、モード入力に接続したCaseストラクチャを配置します。モードの値により、シフトレジスタのデータを読み取ったり、変更したりすることができます。

次の図は、これらの異なる 3 つのモードを処理するインタフェースを備えたサブ VI の概要を示したものです。Change Channel Info のコードだけが示されています。



要素の数が1,000個の場合、この方法ではスピードが前の方法の2倍、最初の方法の4倍になります。

ケーススタディ 3：静的な文字列のグローバルテーブル

前の例では、テーブルに複数のデータタイプが混在し、かつ頻繁にテーブルの内容が変更されるアプリケーションについて説明しました。一方、多くのアプリケーションには、いったん作成された後はほとんど変更されることのないテーブルもあります。テーブルはスプレッドシートファイルから読み取られる場合もあります。テーブルは、いったんメモリに読み取られると、主にデータの検索用に使用されます。

ここでは、Initialize Table From File と Get Record From Table という 2 つの関数を使用する方法について説明します。

表を組み込むための 1 つの方法は、文字列の 2 次元配列を使用する方法です。この場合、コンパイラは文字列配列中の個々の文字列を、別々のメモリブロックに保存する点に注意する必要があります。文字列の数が多い場合（たとえば 5,000 個以上あるような場合）は、メモリマネージャの負担が大きくなります。そのため、個々のオブジェクトの数が多くなると、パフォーマンスの顕著な低下を招く可能性があります。

大きな表を記憶するもう 1 つの方法として、テーブルを 1 つの文字列として読み取る方法があります。この場合、文字列中の各レコードのオフセットを格納するためのもう 1 つの配列を作成します。それによってデータの構造が変化し、何千もの小さなメモリブロックの代わりに、1 つの大きなメモリブロック（文字列）ともう 1 つの小さなメモリブロック（オフセットの配列）を使用することになります。

この方法は、設計が複雑ですが、大きなテーブルをより速く処理することができます。

29

移植性およびローカル化について

この章では、プラットフォーム間でのVIの転送、およびローカル化について説明します。

移植可能なVIと移植不可能なVI

VIは、アプリケーションのバージョンが同じであれば、アプリケーションを実行するあらゆるプラットフォーム間での移植が可能です。CINやプラットフォーム固有の機能（DDEなど）を使用しているVIは、移植できません。そのようなVIは、移植をしてもVIが破損します。

Gは、あらゆるプラットフォームで同じファイルフォーマットを使用します。VIは、ディスクまたはネットワークを使用して1つのシステムから別のシステムに転送することができます。

ファイルを新しいシステムに転送したら、VIを開くことができます。アプリケーションは、VIが別のプラットフォームからのものであることを検知し、現在のプロセッサ用の正しいコマンドを使用できるようにVIをコンパイルし直します。別のプラットフォーム用にフォーマットされたディスクを使用してVIを転送する場合は、ディスクを読み込むためのユーティリティプログラム（Macintoshの Apple File Exchangeなど）が必要になる場合があります。

次のようなVIは移植できません。

- `vi.lib`ディレクトリに入れて配布されたVI。配布される各コピーに含まれるvi.libはそのコピー専用の`vi.lib`であるため、`vi.lib`のVIは他のプラットフォームに移動しないでください。
- CINを含むVI。CINが別のプラットフォームからのものである場合、[オブジェクトコードがロードされていません]というエラーメッセージが表示されます。CINのソースコードがプラットフォームから独立したかたちで作成されている場合は、別のプラットフォーム上でCINをコンパイルし直し、移植したVIにリンクすることができます。
- Call Library関数を含むVI。VIを移植することはできますが、同じ名前のライブラリが見つからない場合は破損されます。
- プラットフォーム固有の通信用 VI（Macintoshの Apple EventsやWindowsのDDEなど）。
- Windows用のIn Portユーティリティ VIとOut Portユーティリティ VI、およびMacintosh用のPeekユーティリティVIとPokeユーティリティVI。

プラットフォーム間での移植

プラットフォーム間での移植を容易にするためにできることはいくつかあります。移植に際しては、ファイル名、区切り文字、解像度やフォントの違い、ラベルの重複、画像フォーマットの違いが問題になります。

考慮すべき点の1つに、ファイル名があります。DOSやWindowsの3.xバージョン、およびWindows NTのFATボリュームでは、ファイル名は最大8文字までで、これに3文字の拡張子を追加することができます。拡張子は、一般的には.viを使用します。Macintoshでは、ファイル名は31文字まで使用できます。Windows 95やWindows NT、およびUNIXでは、ファイル名は拡張子.viを含めて255文字まで使用できます。

VIは、短い名前で保存するか、またはVIライブラリに保存するようにすると、あとの作業が楽になります。VIライブラリは1個のファイルに相当し、複数のVIを保存することができます。ライブラリの名前は各プラットフォームの条件を満たしている必要がありますが、ライブラリ中のVIにはプラットフォームに関係なく最大255文字までの名前を付けることができます。このように、VIライブラリはファイルシステムにほとんど依存しないため、VIの転送にはライブラリフォーマットが最も適しています。VIライブラリの作成に関しては、詳しくは「第2章　VIを編集する」の「VIを保存する」の項を参照してください。

ファイル名に関するもう1つの問題は、UNIXでは大文字と小文字を使い分ける必要があり、それ以外のプラットフォームではその必要がないという点です。たとえば、VIを名前で検索する場合には、名前で大文字を使用している箇所にはかならず大文字を入力する必要があります。

区切り文字

複数のプラットフォームで使用するVIを作成する場合、ファイル名にプラットフォーム固有のパス区切り文字［¥、/、:］を使用しないようにする必要があります。また、特殊文字のなかには、ファイルシステムによって異なる解釈のされ方をするものがあるため、ファイル名に特殊文字を使用することを避ける必要があります。たとえば、UNIXでは隠しファイルの名前はピリオドで始まります。

解像度とフォント

移植に際しては、画面の解像度やフォントがプラットフォームによって異なるという点も問題になります。フォントはプラットフォームによって異なる可能性があるため、VIを移植した場合はVIが正しく表示されるように新しいフォントを選択し直すことが必要になることがあります。VIの設計に際しては、プラットフォーム間で最良の状態でマッピングされるフォ

ントとして、アプリケーションフォント、システムフォント、およびダイアログフォントという3種類のフォントがある点に留意してください。

あらかじめ定義されたこれらのフォントに対してマッピングされる実際のフォントは下記の通りです。

- アプリケーションフォントはデフォルトのフォントです。このフォントは、**制御器**パレット、**関数**パレット、および新しい制御器で使用されます。
  - **（Windows）**アメリカ版のWindowsは、通常はArialフォントを使用します。サイズはビデオドライバの設定によって異なります。これは、大きいフォントや小さいフォントを使用するためにビデオドライバの解像度を高く設定する場合があるためです。日本語版のWindows では、アプリケーションは Windows がプログラムマネージャのファイル名に対して使用するフォントを使用します。
  - **（Macintosh）**アプリケーションは、Finderでファイル名に使用されるのと同じフォントをアプリケーションフォントとして使用します。たとえば、アメリカ版の Macintosh システムではアプリケーションは Geneva を使用するのに対し、日本語版の Macintosh システムではアプリケーションは Osaka を使用します。
  - **（UNIX）**アプリケーションはデフォルト設定ではHelveticaを使用します。
- システムフォントはメニューに使用されるフォントです。
  - **（Windows）**アプリケーションはHelvetica を使用します。サイズはビデオドライバによって異なります。
  - **（Macintosh）**アプリケーションは、アメリカのシステムソフトウェアでは通常 Chicago を使用し、日本語のシステムソフトウェアでは Osaka を使用します。
  - **（UNIX）**アプリケーションは、通常は Helvetica をシステムフォントとして使用します。
- ダイアログフォントは、ダイアログボックスのテキストに使用されるフォントです。
  - **（Windows）**アメリカ版のWindows では、アプリケーションフォントの太字版がダイアグラムフォントとして使用されます。日本語版の Windows では、ダイアログフォントはシステムフォントと同じです。
  - **（Macintosh）**アプリケーションは、システムフォントと同じフォントを使用します。
  - **（UNIX）**アプリケーションは、通常は Helvetica をダイアグラムフォントとして使用します。

上記のいずれかのフォントを使用しているVIを別のプラットフォームに移植すると、アプリケーションはそのフォントを移植したプラットフォーム上のよく似たフォントにマッピングします。

あらかじめ定義されたフォントのかわりにGenevaやNew Yorkといった特定のフォントを選択した場合、新しいプラットフォームではフォントのサイズが変更される可能性があります。これは、使用可能なフォントや画面の解像度がプラットフォームによって異なるためです。Macintoshで Geneva またはNewYorkを選択した場合でも、SunやHP-UXではアプリケーションがそれに対応できないためfixedというフォントが使用されます。認識不能なフォントを使用したVIは、Windowsに移植しても新しいフォントに正しくマッピングされない場合があります。

ナショナルインスツルメンツでは、テキストの一部にあらかじめ定義されたフォントを使用している場合はそのテキストサイズを変更しないよう推奨します。フォントをデフォルトサイズ以外のサイズに変更した状態でVIを別のプラットフォームに移植すると、アプリケーションはそのフォントを新しいサイズに適合させようとしますが、予め指定された画面の解像度によっては正しく表示されないことがあります。たとえば、サイズが10（デフォルトより1ピクセル大きいサイズ）のアプリケーションフォントは、Macintosh（Geneva 10）では問題なく表示されますが、高解像のビデオドライバを使用しているWindowsではデフォルトよりも3ピクセル小さくなり、非常に小さく表示されます。

ラベルの重なり

VIを新しいプラットフォームに移植すると、フォントの大小によって制御器やラベルのサイズが変わる場合があります。Gは、ラベルが制御器に重ならないように、ラベルを制御器から離します。また、どのラベルや定数にも**テキストにサイズを合わす**と呼ばれるデフォルト属性があります。最初にラベルまたは定数を作成する際にはこの属性が設定され、オブジェクト内のすべてのテキストが表示されるようにオブジェクトの大きさが必要に応じて変更されます。

オブジェクトのサイズを手動で変更すると、アプリケーションはこの属性をオフにします（ポップアップメニューのチェックマークが消えます）。**テキストにサイズを合わす**がオフになっていると、オブジェクトの境界サイズは変更されることはないため、アプリケーションは必要に応じてテキストの一部をカットします。別のシステムあるいはプラットフォームに移植したときにテキストがカットされないようにしたいときは、ラベルや定数のこの属性をオンに設定しておく必要があります。

SunやHPのモニタのほとんどは、PCやMacintoshのモニタよりもかなり大きく、高い解像度を使用しています。SunやHP-UXをご使用の場合は、フロントパネルをあまり大きくしない方がスムーズに移植できます。

最良の結果を得るためには、制御器が重ならないように、余白を残しておく必要があります。ラベルが一部でも他のオブジェクトに重なっていると、フォントを拡大したときに制御器と重なるおそれがあります。

画像

最も基本的なタイプの画像は、画像の個々のピクセルの色を指定する一連の値、すなわちビットマップを持っています。より複雑な画像になると、画像を描画するたびに実行されるいくつかのコマンドが含まれている場合もあります。描画コマンドを含む画像は、描画プログラム、あるいはグラフィックスアプリケーションの描画層で作成されます。ビットマップ方式の画像は、ペイントプログラム、あるいはグラフィックスアプリケーションのペイント層で作成されます。

ビットマップは、あらゆるプラットフォームに共通の画像記憶フォーマットです。フロントパネルでビットマップ画像を使用している場合は、VIを別のプラットフォームにロードしても画像は同じように表示されます。一方、描画コマンドを含む画像には、たとえばクリッピングやパターンの塗りつぶしコマンドのような他のプラットフォームではサポートされないコマンドが含まれている可能性があります。このような画像を移植すると、別のプラットフォームでは奇妙に表示される場合があります。画像を別のプラットフォームで使用したいときは、VIがそのプラットフォームでどのように表示されるかチェックする必要があります。

画像は、描画プログラムやグラフィックスアプリケーションの描画層を使用して作成できなくはありません。ただし、画像をより確実に移植したいときは、最終的な画像をペイントプログラムやグラフィックスアプリケーションのペイント層に貼り付けてからインポートするようにしてください。

（Windows 95、Windows NT、Macintosh）Windows 95やWindows NTの一部のアプリケーション、およびMacintoshの多くのアプリケーションでは、長方形以外の画像を切り取ったりコピーしたりすることができます。たとえば、Macintoshでは、左に示したようなLassoツールを使用して円や三角形、あるいは音符のようなより複雑な形の輪郭を選択することができます。他のプラットフォームでは、このような変則的な図形を長方形の白い背景の上に描画される場合があります。Windows 95やWindows NTで変則的な形の画像や適切にサイズ調整される画像が必要であれば、拡張メタファイルをサポートするアプリケーションを探してください。

VIのローカル化

フロントパネル上の文字列を他言語に変換するためには、まず最初にそれらの文字列をタグ付きのテキストファイルにエクスポートし、そのテキストファイルを翻訳したのち、再びLabVIEWのパネルにインポートします。詳しくは、「第5章　VIの印刷および文書作成」を参照してください。タグ付きのテキストファイルを翻訳する場合、VIウィンドウのタイトルを次の図に示すような方法で翻訳することもできますが、VI設定ダイアログボックスを使用して対話方式でウィンドウのタイトルを変更することもできます。詳しくは、本章の「VIウィンドウタイトルを編集する」の項、および「第6章　VIおよびサブVIをセットアップする」の「ウィンドウオプション」の項を参照してください。

図 29-1 VIのローカル化例

フロントパネルのすべてのオブジェクトには、キャプションが付いています。G言語エンジンはラベルを使用してオブジェクトを識別するため、ラベルを翻訳することはできません。ラベルをダイアグラムに影響を与えずに変更することはできませんが、キャプションを翻訳してラベルを表示しないようにすることは可能です。詳しくは、「第2章　VIを編集する」の「G環境」の項を参照してください。

フロントパネルの文字列をローカル化できるだけでなく、数字を文字列に変換する際に各言語固有の小数点区切りを使用することもできます。詳しくは、本章の「小数点区切りとしてのピリオドとカンマ」の項を参照してください。Format Date/Time String関数は、指定されたフォーマットで日付と時間を表示します。

VI文字列のインポートとエクスポート

VI文字列のエクスポート／インポートツールは、VIのフロントパネルに含まれるローカル化が可能なすべての文字列を、VI文字列ファイルと呼ばれるタグ付きのテキストファイルに書き込みます。**プロジェクト→文字列のエクスポート**または**プロジェクト→文字列のインポート**を選択したのち、ファイルダイアログボックスでエクスポートまたはインポートしたいテキストファイルを選択します。認識不可能なタグが付いているファイルやタグが付いていないファイルは、インポートされません。

ローカル化が可能な文字列は下記の通りです。

- VIウィンドウのタイトルおよび説明
- オブジェクトのキャプションおよび説明
- フリーラベル
- デフォルトデータ（文字列、テーブル、パス、配列のデフォルトデータ）
- プライベートなデータ（リストボックスの名前、テーブルの行や列のヘッダ、グラフのプロット名、グラフカーソルの名前）

文字列をエクスポートあるいはインポートすると、その実行中に発生したすべてのエラーを記録したログファイルも生成されます。

VI文字列ファイルの構文

VI文字列ファイルのフォーマットは、HTMLファイルとよく似ています。個々の要素にはスタートタグとエンドタグが付けられます。スタートタグは<で始まって>で終わり、エンドタグは</ではじまって>で終わります。ホワイトスペース（空白）文字は、テキスト中で使用しているもの以外は無視されます。<はタグの頭を示す文字として使用するため、テキスト中の「より大きい」を表す記号にはかわりに<<を使用し、「より小さい」を表す記号にはかわりに>>を使用する必要があります。ダブルクォーテーションマークは、"" に置き換えられます。また、行末文字は、<CR>、<CRLF>、または<LF>と表記され、それぞれ復帰文字、復帰改行文字、改行文字として扱われます。このフォーマットは、マシンで読み込めるようにするためのものであるため、読みにくくても心配する必要はありません。タグを変更したり削除したりした場合は、ソフトウェアにファイルをインポートする際にエラーが検出されます。

表29-1は、VIのタグタイプと構文を示したものです。

表29-1 VIのタグの記述

VIのタグタイプ	VIのタグ構文
[VI string file]	<VI [vi attributes] > [vi info] </VI>
[vi attributes]	syntaxVersion=1 lvVersion=nnn revision=nnn name="text"
[vi info]	[vi title] [description] [content]
[vi title]	<TITLE>text</TITLE> | <TITLE><NO_TITLE></TITLE>
[description]	<DESC>text</DESC>
[content]	<CONTENT>[objects]</CONTENT>

VIの属性と属性の間はスペースで区切ります。属性名とその後ろの等号の間、および等号と属性値の間にはスペースは入りません。

例：

```
<VI syntaxVersion=1 LVversion=4502007 revision=10
name="AO Generate Waveform.vi">

<TITLE>AO Generate Waveform.vi</TITLE>

<DESC>This VI generates a timed, simple-buffered
waveform for the given output channel at the specified
update rate.</DESC>

<CONTENT>

.....

</CONTENT>

</VI>
```

表29-2は、フロントパネルの内容について記述したタグであるフリーラベル、オブジェクト所有のラベル、キャプション、および属性を示したものです。

表 29-2 フロントパネルの内容

タグタイプの内容	タグ構文の内容
[content]	<CONTENT>[objects]</CONTENT>
[objects]	([control] | [label]) *
[control]	<CONTROL [control attributes]> [control info] </CONTROL>
[label]	<LABEL>[style text] </LABEL>
[style text]	<STEXT>text with font information </STEXT>

<STEXT> と </STEXT> の間には、フォント設定を入力することができます。フォント情報は、<FONT name="font name" size='3' style='BIUSO' color=00ff00> というフォーマットで記述されます。フォント属性は任意の順番で入力できます。フォント設定は他の要素とは異なり、エンドタグがありません。たとえば、"Bold label" というテキストからなるキャプションは、次のように記述されます。

```
<LABEL><STEXT><FONT name="times new roman" size=12
style='B'>Bold <FONT style='I'>label</STEXT></LABEL>
```

フォントは、あらかじめ定義されたフォント（アプリケーションフォント、ダイアログフォント、またはシステムフォント）のうちの１つを指定するための predef として定義することができます

表 29-3 は、[control] を記述するタグを示したものです。

表 29-3 [control] のタグ

タグタイプの内容	タグ構文の内容
[control]	<CONTROL [control attributes]> [control info] </CONTROL>
[control attributes]	ID=xxx type="Boolean" name="switch"
[control info]	[description] [parts] [privData section] [defData section] [content]
[parts]	<PARTS> [part]*</PARTS>
[part]	<PART [part attributes]> [part info] </PART>

表 29-3 [control]のタグ（続き）

タグタイプの内容	タグ構文の内容
[part attributes]	partID=nnn partOrder=nnn
[part info]	[control] | [label] | [multiLabel]

次に、"Ring" というキャプションとLoad、Unload、開く、閉じるというオプションを使用したリング制御器の記述の例を示します。

```
<CONTROL ID=87 type="Ring" name="RING control">
<DESC>ring control</DESC>
<PARTS>
<PART ID=12 order=0 type="Ring
Text"><MLABEL><STRINGS><STRING>Load</STRING><STRING>Unl
oad</STRING><STRING>Open</STRING><STRING>Close</STRING>
</STRINGS></MLABEL></PART>
<PART ID=82 order=0 type="Caption"><LABEL><STEXT><FONT
color=FF0033 size=12>RING</STEXT></LABEL></PART>
</PARTS>
</CONTROL>
```

上記のMLABEL（複数のラベルを意味するmultilabelの省略形）というタグは、リング制御器のオプションを表す文字列またはブールボタンの文字列、すなわち４つのそれぞれの状態を表す文字列を指定するために使用しています。次に、MLABELタグの構文を使用した一般的な記述を示します。

```
[multiLabel]<MLABEL> [mlabel info] </MLABEL>
[mlabel info][font][strings]
```

表29-4は、文字列、表、配列、およびパスのデフォルトデータを記述するタグを示したものです。

表 29-4 文字列のデフォルトデータ

タグタイプの内容	タグ構文の内容
[defData section]	<DEFAULT> [defData] </DEFAULT>
[defData]	[str def] | [table def] | [arr data] | [path data]
[str def]	[string] | <SAME_AS_TEXT>
[table def]	[strings]
[arr data]	<ARRAY nElems=n> [arr element data] </ARRAY>
[arr element data]	clust data] | [str data] | [non-str data]
[str data]	[string]
[non-str data]	<NON_STRING>
[clust data]	<CLUSTER nElems=n> [clust element data] </CLUSTER>
[clust element data]	[clust data] | [str data] | [non-str data] | [arr data] | [path data]
[path data]	<PATH type =”absolute”> a<SEP> SYSTEM </PATH>

[arr data]については、<ARRAY>タグの後ろにn個の[arr element data]を入力する必要があります。同様に、[clust data]についてもn個の[clust element data]が必要になります。

文字列制御器のデフォルトデータには、<SAME_AS_TEXT>という特殊なタグを使用します。このタグは、文字列のデフォルトデータが文字列のpartsListのkStrTextIDラベルと同じであることを示します。このタグを使用すると、kStrTextIDラベルと文字列のデフォルトデータの両方を指定するために同じテキストを繰り返し入力しなくても済みます。

パス制御器のデフォルトデータについては、スタートタグの<PATH>にパスのタイプを指定する属性を入力することができます。使用可能な属性値は、"absolute"、"relative"、"not-a-path"、および"unc"です。<PATH>と</PATH>に挟まれたパスセグメントの区切りには、<SEP>タグ

を使用します。たとえば、Windowsプラットフォームでは、c:¥windows¥temp¥temp.txtという絶対パスは次のように記述されます。

```
<PATH type="absolute">c<SEP>windows<SEP>temp<SEP>temp.txt</PATH>
```

表29-5は、リストボックスの項目名、行や列のヘッダ、表のセルに使用するフォント、グラフのプロットやカーソルの名前といったプライベートなデータを記述するタグを示したものです。

表 29-5 表のセル、およびグラフのプロット名とカーソル名のタグ記述

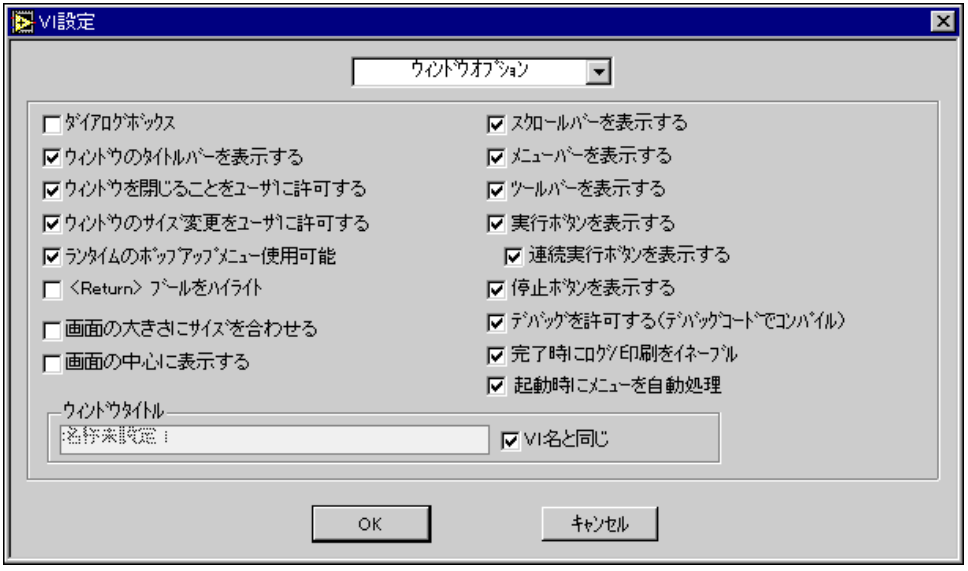
タグタイプの内容	タグ構文の内容
[privData section]	<PRIV> [privData] </PRIV>
[privData]	([items] | [col header] | [row header] | [cell fonts] | [plots] | [cursors])
[items]	<ITEMS> [string]* </ITEMS>
[col header]	<COL_HEADER> [string]* </COL_HEADER>
[row header]	<ROW_HEADER> [string]* </ROW_HEADER>
[cell fonts]	<CELL_FONTS> [cell font]* </CELL_FONTS>
[plots]	<PLOTS> [string]* </PLOTS>
[cursors]	<CURSORS> [string]* </CURSORS>
[cell font]	[row# col#][font]
[font]	<FONT name="font name" size='x' style='BIUSO' color=000000>

[strings]と[string]のフォーマットは次のようになります。

```
[strings]<STRINGS> [string]* </STRINGS>
[string]<STRING> text </STRING>
```

VIウィンドウタイトルを編集する

VIウィンドウタイトルは、VIのファイル名よりもさらに内容をよく表す名前に変更することができます。この機能は、ローカル化したVIに対して重要な意味を持ちます。VIウィンドウのタイトルは他言語に変換でき、ファイルシステムにおける命名条件に拘束されず、ウィンドウを呼び出すVIによって認識されます。VIの編集中にVIウィンドウのタイトルを変更するためには、次の図に示すようにいちばん上のリングで**VI設定**の**ウインドウオプション**を選択します。



ウィンドウタイトルを変更するためには、**VI名と同じの選択**を解除し、使用したいタイトル名をタイプします。VIウィンドウの名前をプログラム的に変更する方法については、「第21章　VIサーバ」を参照してください。

小数点区切りとしてのピリオドとカンマ

システムの小数点区切りの使用方法、すなわちピリオドの使用法は、**編集→環境設定**ダイアログボックスのドロップダウンメニューから**フロントパネル**を選択することにより指定できます。また、下記の関数を使用すると、数字を文字列に変更する際に（および文字列を数字に変更する際に）システムに強制的にピリオドを小数点区切りとして使用させることができます。

- To Engineering
- To Fractional
- To Exponential
- From Exponential/Fract/Eng

Format Date/Time String 関数

日付と時間の表示方法は、Format Date/Time String 関数を使用して指定することができます。この関数については、詳しくは『オンラインリファレンス』**ヘルプ→オンラインリファレンス**を選択）を参照してください。

G 言語アプリケーション間の移植

ブロックダイアグラムを含む LabVIEW の VI は、LabVIEW と BridgeVIEW で G 言語のバージョンに互換性がある場合には、BridgeVIEW に変換することができます。BridgeVIEW から LabVIEW への変換も可能ですが、すべての VI を変換できるわけではありません。

LabVIEW から BridgeVIEW への変換

VI を LabVIEW から BridgeVIEW に変換する際に注意すべき最も重要な点は、VI を作成する際に使用した G プログラミング言語のバージョンです。次の表をご覧ください。LabVIEW 3.x ～ LabVIEW 4.x で作成したすべての VI は、VI からダイアグラムを削除していない限り BridgeVIEW のどのバージョンにも変換できます。LabVIEW 5.0.x で作成した VI は、BridgeVIEW 2.0 以降のバージョンにのみ変換できます。

LabVIEW の バージョン	G の バージョン	BridgeVIEW のバージョン	G の バージョン
4.0.1までのすべてのバージョン	4.0.1までのすべてのバージョン	1.0	4.1
4.1	4.0.2	1.0.1	4.1.1
4.1.1	4.0.3	1.1	4.1.2
5.0	5.0	2.0	5.0

BridgeVIEW から LabVIEW への変換

LabVIEW で BridgeVIEW の VI をロードし実行できるかどうかは、２つの要素によって決まります。１つは、G 言語のバージョンの互換性です。G 4.0.x は G 4.1.x 以降のバージョンで作成した VI をロードすることはできないため、現時点では BridgeVIEW の VI は LabVIEW 5.0 以降のバージョンでロードする必要があります。VI を BridgeVIEW から LabVIEW に変換できるか否かを決定付けるもう１つの要素は、VI に BridgeVIEW 固有の VI が含まれているかどうかという点です。BridgeVIEW 固有の機能を使用した BridgeVIEW VI は、LabVIEW にロードできない場合や LabVIEW で正しく動作しない場合があります。

たとえば、タグデータタイプは BridgeVIEW に固有のものです。タグデータタイプを使用したり BridgeVIEW エンジンにアクセスする VI は、LabVIEW では正しく動作しません。

BridgeVIEW で後に LabVIEW に変換する VI を作成する場合は、Basic G または LabVIEW パレットセットを使用する必要があります。このパレットセットには、LabVIEW のパレットセットと同じ関数が表示されます。これらのパレットの関数のみを使用して作成した VI は、BridgeVIEW から LabVIEW に移植できます。

A

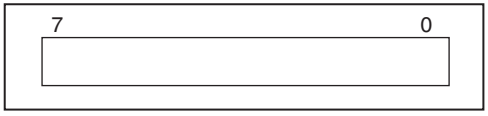
データ記憶形式

この付録は、データの記憶形式について説明したものです。この付録には、コードインタフェースノード（CIN）を使用するユーザや、ファイルI/O関数を使用してファイルの読み込みや書き込みを行うユーザなど、上級ユーザにとって特に役立つ内容が記載されています。この付録では、メモリへのデータの記憶方法、タイプデスクリプタとデータの記憶との関係、およびディスクにファイルを保存するためのデータの平坦化方法について説明します。

フロントパネルの制御器および表示器のデータ形式

ブール

ブールは、8 ビット値として記憶されます。値が 0 のときは、ブールはFALSE になります。0 以外の値はTRUE を表します。

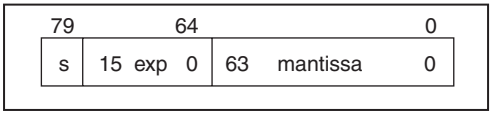


数値

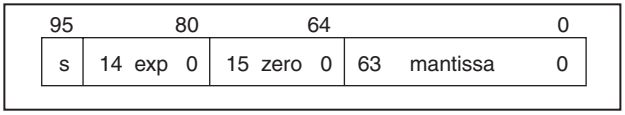
拡張精度

拡張精度の数字は、ディスクに保存する際には各プラットフォーム共通の128 ビット形式で記憶されます。この形式は、UNIX のインメモリフォーマットと同じです。メモリ内でのサイズと精度は、プラット形式によって異なります。

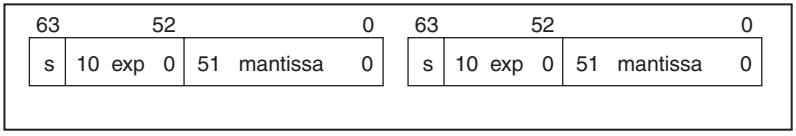
（Windows）拡張精度の浮動小数点数は、80 ビット形式（80287 拡張精度形式）です。



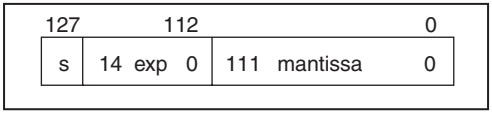
（68K Macintosh）拡張精度の浮動小数点数は、96ビット形式（MC68881-MC68882 拡張精度形式）です。



（Power Macintosh）拡張精度の浮動小数点数は、2 つの倍精度浮動小数点数を組み合わせた Apple double-double 形式で表されます



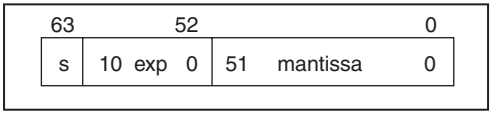
（Sun）拡張精度の浮動小数点数は、128 ビット形式です。



（HP-UX）拡張精度の浮動小数点数は、次の図に示すように倍精度浮動小数点数として表されます。

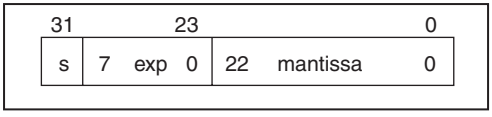
倍精度

倍精度浮動小数点数は、64 ビット IEEE 倍精度形式です（形式デフォルト）。



単精度

単精度浮動小数点数は、32 ビット IEEE 単精度形式です。



倍長整数

倍長整数は、符号付きまたは符号なしの 32 ビット形式です。

ワード整数

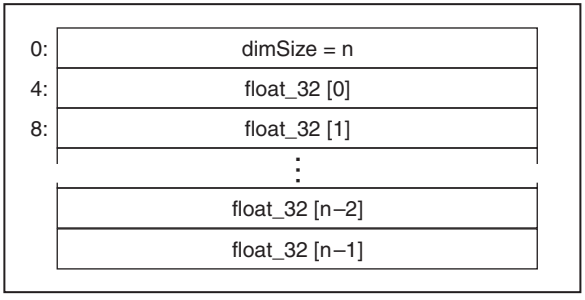
ワード整数は、符号付きまたは符号なしの 16 ビット形式です。

バイト整数

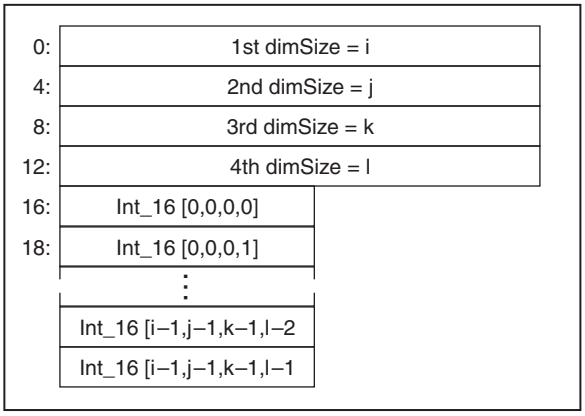
バイト整数は、符号付きまたは符号なしの 8 ビット形式です。

配列

配列は、配列の各次元のサイズを示す倍長整数の後ろにデータが続く形で保存されます。プラットフォームによってはデータの配置に関する制約があるため、データの最初の要素の位置を調節する目的で次元のサイズを表すバイトの後ろに数バイト分のパッド文字が挿入される場合があります。LabVIEW をご使用の場合は、詳しくは『LabVIEW Code Interface Reference Manual』の「Chapter 2, CIN Parameter Passing」の「Alignment Considerations」を参照してください。このマニュアルは、ソフトウェアのプログラムディスクまたは CD にポータブルドキュメント形式（PDF）文書の形で収められています。次の例は、単精度浮動小数点数の 1 次元配列を示したものです。左側の 10 進数は、メモリ上の配列の各データ成分の開始位置に相当するバイトのオフセットを示しています。

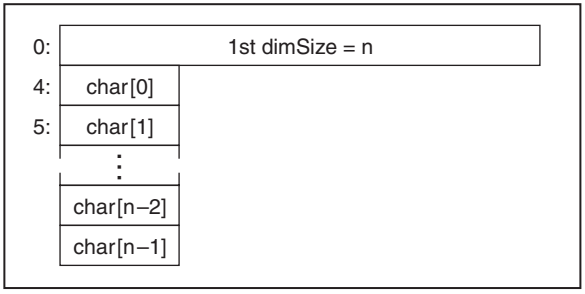


次の図は、ワード整数の4次元配列を示したものです。



文字列

文字列は、バイト整数（8ビット文字）の1次元配列として記憶されます。



パス

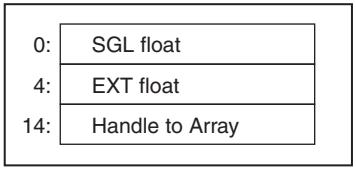
パスは、パスのタイプ、パスのコンポーネントの数を示すワード整数、パスの各コンポーネントという順序で記憶されます。パスのタイプは、絶対パスの場合は0、相対パスの場合は1になります。それ以外のパスタイプ値は、パスが無効であることを意味します。パスの各コンポーネントはパスカル文字列（P文字列）で表され、最初のバイトにはP文字列のバイト長（バイト長を示すバイトは含みません）が書き込まれます。

クラスタ

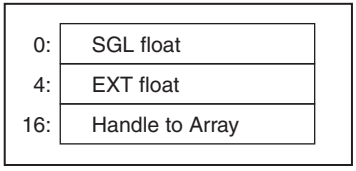
クラスタには、さまざまなデータタイプの要素がクラスタ順位に基づいて記憶されます。スカラデータは直接クラスタに記憶されます。配列、文字列、ハンドル、およびパスは、間接的に記憶されます。クラスタは、実際にデータが記憶されるメモリ領域を指すハンドルを記憶します。プラットフォームによってはデータの配置に関する制約があるため、データの最初の要素の位置を調節する目的で次元のサイズを表すバイトの後ろに数バイト分のパッド文字が挿入される場合があります。LabVIEW をご使用の場合は、詳しくは『LabVIEW Code Interface Reference Manual』の「Chapter 2, CIN Parameter Passing」の「Alignment Considerations」の項を参照してください。このマニュアルは、ソフトウェアのプログラムディスクまたは CD にポータブルドキュメント形式（PDF）文書のかたちで収められています。

次の図は、単精度の浮動小数点数、拡張精度の浮動小数点数、および符号なしワード整数の 1 次元配列のハンドルの順序で含むクラスタを示したものです。詳しくは、「第 14 章　配列とクラスタの制御器 および表示器」を参照してください。

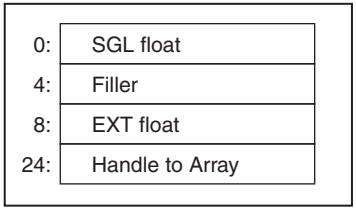
- **（Windows）**



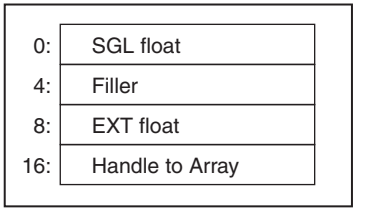
- **（Macintosh）**



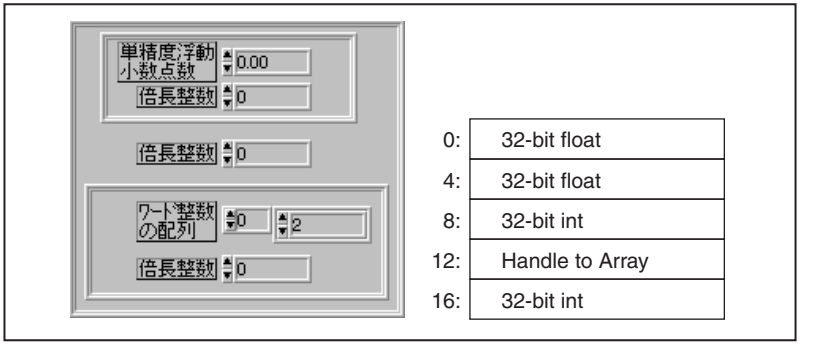
- **（Sun）**



- **（HP-UX）**



次の例では、埋め込みクラスタは間接的に記憶されません。データは、サブクラスタに埋め込まれていないものとして埋め込みクラスタ内に直接記憶されます。配列、文字列、およびハンドルだけが間接的に記憶されます。



タイプデスクリプタ

ブロックダイアグラムの個々のワイヤや端子には、それぞれのデータタイプがあります。これらのデータタイプは、タイプデスクリプタと呼ばれるメモリ内構造体を使用して記憶されます。このデスクリプタはワード整数の文字列で、G のあらゆるデータを記述することができます。数値は、別途記載した場合を除いては16進数で書き込まれます。

タイプデスクリプタの共通形式は、`<length> <type code>` です。

タイプデスクリプタのなかには、タイプコードの後ろに追加情報を持つものもあります。配列やクラスタは他のデータタイプを含むため、データタイプは構造体または集合体になります。たとえば、クラスタタイプには、クラスタの個々の要素のタイプに関する追加情報が含まれます。

タイプデスクリプタの最初のワード（16ビット）は、かならずそのタイプデスクリプタ自身のバイト長（バイト長を示すワードを含む）を表します。2つめのワード（16ビット）は、タイプコードです。タイプコードの上位

バイトは、内部で使用するために予約されています。2 つのタイプデスクリプタが等価であるかどうかを比較する際には、このバイトは無視してください。すなわち、タイプコードの上位バイトが一致していなくても、2 つのディスクリプタを等価であるとみなすことができます。

タイプコードは、次の表に示すように、単精度の浮動小数点数あるいは拡張精度の浮動小数点数といった実際のタイプ情報をコード化します。これらのタイプコード値は、今後のバージョンでは変更される可能性があります。タイプデスクリプタの欄の *xx* は、値が予約されていることを表しています。これらは無視してもかまいません。

データタイプ

次の表は、スカラの数値タイプと非数値タイプ、タイプコード、およびタイプデスクリプタを示したものです。

表 A-1 スカラ数値データタイプ

データタイプ	タイプコード（16 進数）	タイプデスクリプタ（16 進数）
バイト整数	01	0004 *xx*01
ワード整数	02	0004 *xx*02
倍長整数	03	0004 *xx*03
符号なしのバイト整数	05	0004 *xx*05
符号なしのワード整数	06	0004 *xx*06
符号なしの倍長整数	07	0004 *xx*07
単精度の浮動小数点数	09	0004 *xx*09
倍精度の浮動小数点数	0A	0004 *xx*0A
拡張精度の浮動小数点数	0B	0004 *xx*0B
単精度の複素数浮動小数点数	0C	0004 *xx*0C
倍精度の複素数浮動小数点数	0D	0004 *xx*0D
拡張精度の複素数浮動小数点数	0E	0004 *xx*0E
列挙体バイト整数	15	<nn> *xx*15 <k> <k pstrs>
列挙体ワード整数	16	<nn> *xx*16 <k> <k pstrs>
列挙体倍長整数	17	<nn> *xx*17 <k> <k pstrs>
単精度の物理量	19	<nn> *xx*19 <k> <k base-exp>

表 A-1 スカラ数値データタイプ（続き）

データタイプ	タイプコード（16進数）	タイプデスクリプタ（16進数）
倍精度の物理量	1A	<nn> *xx*1A <k> <k base-exp>
拡張精度の物理量	1B	<nn> *xx*1B <k> <k base-exp>
単精度の複素数物理量	1C	<nn> *xx*1C <k> <k base-exp>
倍精度の複素数物理量	1D	<nn> *xx*1D <k> <k base-exp>
拡張精度の複素数物理量	1E	<nn> *xx*1E <k> <k base-exp>

表 A-2 非数値データタイプ

データタイプ	タイプコード（16進数）	タイプデスクリプタ（16進数）
ブール	21	0004 *xx*21
文字列	30	0008 *xx*30 <len>
ハンドル	31	0006 *xx*31 <kind>
パス	32	0008 *xx*32 <len>
画像	33	0008 *xx*33 <len>
配列	40	<nn> *xx*40 <k> <k dims> <element type descriptor>
クラスタ	x50	<nn> *xx*50 <k> <k element type descriptors>

タイプデスクリプタのサイズフィールドの最小値は4です（表 A-1 参照）。ただし、どのタイプデスクリプタも名前（パスカル文字列）を追加することができ、その場合はサイズは名前の分だけ長くなります（2 の倍数に丸められます）。

文字列、パス、画像のデータタイプは、長さは32ビット（配列の次元のサイズと同様）ですが、現時点でコード化されている値は可変サイズを表す FFFFFFFF （-1）だけです。現時点では、文字列、パス、および画像は、いずれも可変サイズです。実際の長さはデータとともに記憶されます。

配列およびクラスタのデータタイプは、どちらも固有のタイプコードを持っている点に注意してください。この 2 つのデータタイプは、そのデータタイプの要素に関する情報のほかに、次元（配列）または構成要素の数（クラスタ）に関する追加情報をともないます。

次の例は、am、fm、fm stereo という項目の数値化されたバイト整数を示したもので、それぞれの文字群が16ビットワードを表します。ダブルクオーテーションマークで囲まれたスペース（" "）は、ASCII のスペースを表します。

```
0016 0015 0003 02a m02 fm 09f m" " st er eo
```

0016 は総データ長が22バイトであることを示します。0015 は数値化されたバイト整数であることを示します。0003 は3つの項目があることを示します。

次の例は、m/s という単位を持つ倍精度の物理量を示したもので、それぞれのグループが16ビットワードを表します。

```
000E 001A 0002 0002 FFFF 0003 0001
```

000E は、総データ長が14バイトであることを示します。0x1A は、データが単位倍精度の数字であることを示します。0002 は、基数と指数の組み合せが2つあることを示します。0002 は、秒の基指標です。FFFF (-1) は、秒の指数です。0003 は、メートルの基指標です。0001 は、メートルの指数です。

☞ **注**　**システムの内部では、あらゆる物理量は実際の表示単位とは無関係に基準単位で記憶されます。表 9-2 にラジアンからカンデラまでの9種類の基準単位を示しましたが、これらの基準単位はそれぞれ0から8までの指標で表されます。**

配列

配列のタイプコードは0x40です。タイプコードのすぐ後ろには、配列の次元の数を示すワードが続きます。その後ろに、各次元ごとに次元のサイズすなわち要素数を示す倍長整数が続きます。最後に、次元のサイズの後ろに要素数のタイプデスクリプタが書き込まれます。要素のタイプは、配列以外であればどのタイプでもかまいません。次元のサイズには、FFFFFFFF (-1) という値を使用できます。この値は、配列の次元のサイズが可変であることを意味します。現時点では、配列はすべて可変サイズになります。実際の次元のサイズはデータとともに記憶され、その値は常に 0 以上となります。次に、倍精度の浮動小数点数からなる 1 次元配列のタイプデスクリプタを示します。

```
000E 0040 0001 FFFF FFFF 0004 000A
```

000E は、タイプデスクリプタの要素を含めたタイプデスクリプタ全体のバイト長です。配列のサイズは可変であるため、次元のサイズは FFFFFFFF となります。要素のタイプデスクリプタ（0004 000A）は、同タイプのスカラの場合と同様に表される点に注意してください。

次に、ブールの2次元配列のタイプデスクリプタの例を示します。

```
0012 0040 0002 FFFF FFFF FFFF FFFF 0004 0021
```

クラスタ

クラスタのタイプコードは0x50です。タイプコードのすぐ後ろには、クラスタの項目の数を示すワードが続きます。ワードの後ろには、**クラスタ順位**中の各構成要素のタイプデスクリプタが続きます。たとえば、符号付きワード整数と符号なし倍長整数の 2 つの整数からなるクラスタについて考えてみます。

```
000E 0050 0002 0004 0002 0004 0007
```

000Eは、構成要素のタイプデスクリプタも含めたタイプデスクリプタのバイト長を示します。

次に、マルチプロットグラフのタイプデスクリプタを示します（数値タイプは可変です）。

```
0028 0040 0001 FFFF FFFF...1次元配列
001E 0050 0001...構成要素が1つのクラスタ
0018 0040 0001 FFFF FFFF...1次元配列
000E 0050 0002...構成要素が2つのクラスタ
0004 000A ...倍精度の浮動小数点数
0004 0003...倍長整数
```

データの平坦化

2 つの内部関数は、データを保存フォーマットからファイルへの書き込みあるいはファイルからの読み取りが容易なフォーマットに変換します。

文字列や配列はハンドルブロックとして記憶されるため、これらのタイプを含むクラスタは**不連続**データになります。通常、データは**ツリー**フォーマットで記憶されます。たとえば、倍精度の浮動小数点数と文字列からなるクラスタは、8 バイトの浮動小数点数、およびそれに続く 4 バイトの文字列ハンドルとして記憶されます。メモリ内では、文字列データは拡張精度の浮動小数点数に隣接して記憶されません。したがって、クラスタデータをディスクに書き込みたいときは、データを 2 つの場所から入手する必要があります。もちろん、任意の複素数データタイプを使用してデータをメモリの別々の場所に記憶することは可能です。

データを VI ファイルまたはデータログファイルに保存する際には、データは 1 つの文字列に**平坦化された**うえで保存されます。このようにすると、任意の複素数クラスタのデータも分断せずに連続した形で保存することが可能になります。一方、G の開発環境でこのようなファイルをディスクからロードする際には、逆の操作、すなわち 1 つの文字列を読み込み、場合によっては不連続な内部フォーマットに**復元する**操作が必要になります。

平坦化されたデータは、どんなプラットフォームで実行している VI でもデータを変更しないで使用できるように、標準の形式に**正常化されます**。数値データは**ビッグエンディアン**形式（最上位バイトが先）で記憶され、拡張精度の浮動小数点数はこの項で前述した Sun の拡張精度形式を使用して 16 バイト量として記憶されます。

☞ **注**　**G を使用せずに作成したアプリケーションで使用するファイルにデータを書き込む際には、データを平坦化してから転送することが必要になる場合があります。同様に、G を使用せずに作成したアプリケーションで生成したファイルからデータを読み込む際には、データを転送してから平坦化することが必要になる場合があります。G 以外の Windows アプリケーションでは、一般的に数値データはこのグリーンのハイライトは無視（最下位バイトを前の番地に書き込むフォーマット）であるものとみなされます。通常他のWindowsおよびMacintoshのアプリケーションでは拡張精度の浮動小数点を先に述べたようにそれぞれ 80 ビット、96 ビット形式とします**

ブロックダイアグラム関数の Flatten to String および Unflatten from String（詳しくは**ヘルプ→オンラインリファレンス**を選択して**オンラインヘルプ**を参照してください）を使用すると、G の開発環境がデータを保存あるいはロードする際に実行するのと同じ平坦化および復元の操作を実行することができます。

スカラ

ブールタイプだけでなく数値タイプも、平坦化されたフォーマットではビッグエンディアン形式のデータのみを使用します。たとえば、値が -19 の倍長整数は、`FFFF FFED` とコード化されます。値が 1/4 の倍精度の浮動小数点数は、`3FD0 0000 0000 0000` となります。ブールの TRUE は非ゼロ値、FALSE は `00` になります。

拡張精度の数値ファイル形式は、SPARC の 4 倍精度形式に基づきます。これは、16 ビットのバイアス指数で、後ろにビット長が 112 の仮数が続きます。この形式では、仮数の 2 進小数点の左のビットは暗黙ビット（常に 1 とみなされます）で、表記には現れません。

文字列、ハンドル、およびパス

文字列、ハンドル、およびパスのサイズは可変であるため、平坦化形式では実際のデータの前にそのデータのバイト長を示す正規化された倍長整数が挿入されます。パスの場合には、このバイト長の前に `PTH0` という 4 文字が挿入されます。たとえば、値が ABC の文字列タイプは、平坦化形式では `0000 0003 4142 43` となります。

平坦化形式は、メモリ内の文字列の形式と似ています。ただし、ハンドルやパスは、メモリに記憶する際にはバイト長データは挿入されないため、バイト長はメモリ内におけるそれらの実際のバイト長になり、平坦化する際に挿入されます。

配列

平坦化された配列データの前には、配列の各次元サイズ（要素の数）を示す正規化された倍長整数が挿入されます。次元のサイズをメモリに記憶する場合と同様、最初に記憶されるのは最も遅く変化する次元で、その後ろにそれよりも速く変化する各次元が順番に続きます。データは、メモリに記憶するときと同じ順番でこれらのサイズのすぐ後ろに続きます。また、このデータも必要に応じて平坦化されます。次に、6つの8ビット整数からなる2次元配列の例を示します。

`{ {1, 2, 3}, {4, 5, 6} } is stored as 0000 0002 0000 0003 0102 0304 0506.`

次に、平坦化されたブール変数の1次元配列の例を示します。

`{T, F, T, T} is stored as 0000 0004 0100 0101.`（`01`はTRUEの推奨値）

クラスタ

平坦化されたクラスタは、その要素を平坦化したデータを（クラスタ中の要素の順序に基づいて）連結した形になります。たとえば、値が`4`（10進数）のワード整数と値が`12`の倍長整数からなるクラスタを平坦化すると、`0004 0000 000C`となります。

文字列`ABC`と値が`4`のワード整数からなるクラスタを平坦化すると、`0000 0003 4142 4300 04`となります。

値が`7`のワード整数と値が`8`のワード整数のクラスタおよび値が`9`のワード整数からなるクラスタを平坦化すると、`0007 0008 0009`となります。

このデータを復元する場合は、単純に逆の操作を行います。ただし、平坦化された個々のデータ成分のデータのタイプはコード化されないため、各データ成分のタイプデスクリプタが必要になります。Unflatten From String関数を使用するためには、関数が文字列を正しく解読できるようにデータタイプを入力として接続する必要があります。

B

Gに関する一般的な質問

この付録では、Gユーザからよく尋ねられるいくつかの質問にお答えします。

チャートとグラフについて

データをチャートやグラフに配線するには？

ヘルプウィンドウを呼び出し、配線ツールをブロックダイアグラムのグラフ端子の上に移動すると、基本的なタイプのグラフへの配線方法に関する簡単な説明が表示されます。`examples¥general¥graphs`ディレクトリには、わかりやすいサンプルが用意されています。

チャートとグラフの基本的な違いは？

チャートとグラフでは、データの表示や更新の方法が異なります。グラフを使用するVIは、まず最初にデータを記憶してからプロットを生成する表計算プログラムと同様、通常はまず最初にデータを配列として収集したのち、グラフにプロットします。それに対し、チャートは新しいデータポイントをすでに表示されているデータポイントに追加します。この方法では、現在の読み取り値あるいは測定値をすでに収集したデータと対照しながら見ることができます。チャートの履歴バッファのサイズは、チャートのポップアップメニューを使用して設定することができます。もちろんグラフ表示器を備えた履歴バッファを使用することも可能ですが、その場合はブロックダイアグラム上で行う必要があります。チャートには、この機能はすでに組み込まれています。

チャートやグラフのスケールを逆転させるには？

*x*軸あるいは*y*軸のスケールを逆転するには、X Flipped属性またはY Flipped属性を使用します。X Flipped属性をTRUEに設定した場合は、*x*軸スケールの最小値が右に、最大値が左にセットされます。同様に、Y Flipped属性をTRUEに設定した場合は、*y*軸スケールの最小値が上に、最大値が下にセットされます。

チャートやグラフの情報を時間フォーマットで表示するには？

グラフやチャートの数値情報は、相対時間または絶対時間のフォーマットで表示することができます。x 軸あるいは y 軸のスケールのフォーマットを**相対時間**に設定するには、ポップアップメニューの**X 軸→形式...** または**Y 軸→形式...** オプションを使用します。

チャートの x 軸にリアルタイムを表示するには？

`examples¥general¥graphs¥charts.llb` に入っている `Real-Time Chart.vi` というサンプル VI を参照してください。この VI は、日付と時間のクラスタを Seconds to Date/Time 関数から時間、分、秒にアンバンドルし、さらに、これらの数値を午前 0 時からの経過秒数に変換します。最後に、この数値は *x* 軸のスケールが**相対時間**に設定されているチャートの Xo およびデルタ X の属性に対する入力として使用されます。あるいは、**絶対時間**のフォーマットと精度オプションを使用することもできます。

チャートをプログラムで消去するには？

空白の配列を Histry Data 属性に接続します。この空白配列のデータタイプは、チャートに接続されたデータタイプと同じになります。

`examples¥general¥graphs¥charts/llb¥How to Clear Charts & Graphs.vi` に、この方法をわかりやすく示したサンプルが入っています。

グラフの更新時の点滅を避けるには？

グラフが点滅しないようにするには、**スムーズアップデート**オプションを使用します。このオプションは、グラフのポップアップメニューの**データ処理**サブメニューに入っています。**スムーズアップデート**を使用すると、グラフはまず最初にスクリーン外バッファで描画しなおされたのち、その画像が画面にコピーされます。このオプションは、表示器の更新をスムーズにしますが、通常はその際に速度が低下し、メモリの使用量も多くなります。

スムーズアップデートは、**編集→環境設定→フロントパネル**ダイアログボックスでグローバルに設定することもできます。また、個々のグラフのポップアップメニュー使用してグラフごとにオン／オフを切り替えることもできます。

グラフを消去せずに更新するには？

あらゆるグラフ（波形、XY、強度）は、新しいデータを書き込む前にかならず消去されます。ただし、毎回の書き込み時にグラフにすでに書き込まれているデータを消去せずに新しいデータを追加するようにすることも簡単にできます。`examples¥general¥arrays.llb¥Separate Array Values.vi` には、Build Array 関数を使用して新しい値を配列に追加する方法を示したサンプルが入っています。LabVIEW をご使用の場合は、

『LabVIEW ユーザーマニュアル』の「第5 章 配列、クラスタ、およびグラフ」を参照してください。

バーグラフを作成するには？

汎用のポップアップメニューにある**一般プロット**項目を使用すると、バープロットを 1 つのスタイルとしてリストします。その他のプロット属性を使用した場合は、異なるベースラインや水平のバープロットの表示が可能になります。`examples¥general¥graphs¥bargraph.llb` も参照してください。この VI ライブラリには、サンプルのバーグラフ、および配列入力からバーグラフを作成するための 2 つの VI が入っています。また、ピクチャ制御器ツールキットを使用してバーグラフを作成することもできます。

極座標プロットやスミスチャートを作成するには？

ピクチャ制御器ツールキットには、極座標プロットやスミスチャートを作成するためのルーチンを使用したサンプルが用意されています。このツールキットは、任意のフロントパネル画像を作成するための多機能パッケージです。このツールキットをセットした画像 VI ライブラリには、グラフィック画像を作成するための 2 次元描画コマンドのセットが含まれています。

90 度回転した y 軸ラベルを作成するには？

BridgeVIEW や LabVIEW では、テキストを回転させることはできません。回転させたテキストを使用するためには、まず最初に Windows のペイントブラシのようにテキスト回転機能を備えたアプリケーションでテキストを作成し、そのテキストを適切なフォーマットで保存します（画像のインポートについては、詳しくは「第 8 章　フロントパネルオブジェクトの概要」の「インポートしたグラフィックスを使用して制御器をカスタマイズする」の項を参照してください）。次に、この画像をインポートし、グラフまたはチャートの y 軸のそばに配置します。

グラフカーソルにテキストラベルを表示するには？

プログラムを使用してこれを行う場合は、まず最初に Cursor Name Visible 属性を使用してカーソル名を表示させます。次に、Cursor Name 属性を使用してカーソルの名前を指定します。対話方式でこれを行う場合は、カーソルスタイルメニューを使用してグラフのカーソルディスプレイの名前を管理します。

グラフに追加したカーソルを削除するには？

それぞれのカーソルの情報を格納したクラスタを含む配列を空にする必要があります。カーソルディスプレイのポップアップメニューから**データ操作→配列を空にする**を選択するか、またはプログラム Cursor List 属性ノードに空の配列を書き込みます。

エラーメッセージとクラッシュについて

アプリケーションのメモリ不足を示すダイアログボックスが表示されたときは？

LabVIEW と BridgeVIEW は必要に応じてメモリを割り当てますが、配列や文字列は連続したメモリブロックに記憶する必要があります。LabVIEW アプリケーションあるいは BridgeVIEW アプリケーションが文字列や配列を記憶できるだけの大きさのメモリブロック（物理メモリまたは仮想メモリ）を見つけられなかった場合は、必要なメモリを割り当てられなかったことを示すダイアログボックスが表示されます。

アプリケーションのメモリ不足の後で VI の変更を保存したいときは、VI を別の場所に保存するか、バックアップコピーを作成しておいて下さい。メモリを増設した後でソフトウェアを再起動したら、これらの VI をメモリにロードして正しく保存されているかどうかをチェックしたのち、VI の開発作業を続行します。

印刷時に LabVIEW あるいは BridgeVIEW がクラッシュしたときは？（Windows）

クラッシュは、ビデオドライバまたはプリンタドライバに関係している可能性があります。作業を続行する方法については、Windows での突発的なクラッシュに関する質問の項を参照してください。

LabVIEW または BridgeVIEW の実行中にソースコードファイルとそのライン番号を含むエラーメッセージが表示されたときは？

これは、エラーが発生したためにアプリケーションの実行を続行できないことを意味します。このメッセージが発生したことをナショナルインスツルメンツに連絡し、その原因となった動作について説明してください。

LabVIEW または BridgeVIEW が突発的にクラッシュする傾向があるときは？（Windows）

クラッシュは、一般保護違反、あるいはソースコードファイルとライン番号を含むエラーメッセージである場合があります。突発的エラーの頻繁な発生には、コンピュータのビデオドライバに関する問題が関与しています。エラーにビデオドライバが関与しているかどうかは、標準の VGA を使用することで確認することができます。ビデオドライバを変更するには、Windows のセットアップ画面でビデオドライバのリストから VGA を選択します。LabVIEW と BridgeVIEW は、画像を呼び出す際に Windows の標準 API を使用しますが、ビデオドライバの多くはこの標準規格に完全には適合していません。標準の VGA を使用したときにクラッシュが発生しな

かった場合は、ビデオドライバの新バージョンを入手する必要があります。ビデオドライバの最新バージョンは、パソコンメーカーまたはビデオカードメーカーのどちらからでも入手できる可能性があります。

ビデオドライバを新しくしても問題が解消されない場合は、ナショナルインスツルメントにご連絡ください。

（Windows）メモリパリティエラーに続いて一般保護違反が発生したときは？

メモリパリティーエラーは、マシンの不良メモリが原因で発生します。このメモリが仮想メモリであるときは、ハードディスクに問題があることを意味し、物理メモリであるときは SIMM に問題があることを意味します。ハードディスクの問題かどうかをチェックするためには、Norton Utilities などの標準のディスクユーティリティパッケージを使用します。物理RAM の問題かどうかをチェックするためには、マシンの SIMM を物理的に 1 つずつ入れ換えて、そのつど PC を再起動します。問題のある SIMM がメモリのいちばん下のバンクにあると、マシンは再起動できなくなります。その場合は、SIMM を交換する必要があります。

プラットフォームと互換性について

プラットフォーム間で VI を転送するために必要なことは？

VI ファイルを別のプラットフォームに転送し、LabVIEW または BridgeVIEW で読み込む場合は、変換は必要ありません。VI を開くと、VI は新しいプラットフォーム用にコンパイルしなおされます。ただし、それには転送した VI にブロックダイアグラムが含まれていなければなりません。CIN を使用している VI については、転送先のプラットフォームで CIN を再度コンパイルし、転送した VI にリンクしなおす必要があります。VI を別のプラットフォームに転送する際には、プラットフォーム固有の関数（Macintosh の Apple Events や Windows の DDE など）を VI からあらかじめ取り除いておくことが必要です。

VI は、ネットワーク、モデム、あるいはディスクを使用して他のプラットフォームに転送することができます。どのプラットフォームでも、VI は同じファイルフォーマットで保存されます。ftp やモデムを使用して VI を転送する場合は、かならずバイナリで転送する必要があります。

転送手段としてディスクを使用する場合は、他のプラットフォームからのディスクを読み込むためのディスク変換ユーティリティが必要になります。ファイルをディスクに保存する際のフォーマットはそれぞれのプラットフォーム（Macintosh、Windows、Sun）で異なるため、ディスクに保存されたファイルのフォーマットを変換ユーティリティを使用して変換しま

す。ほとんどのファイル変換ユーティリティは、他のプラットフォームのファイルを**読み込む**だけでなく、それぞれのプラットフォームのディスクフォーマットでファイルを**書き込む**ことができます。PC 側で Macintosh のディスクフォーマットから PC のフォーマットに（あるいはその逆に）変換するユーティリティとしては、MacDisk や TransferPro などがあります。一方、Macintosh 側で DOS フォーマットのディスクを Macintosh のフォーマットに（あるいはその逆に）変換するユーティリティとしては、DOS Mounter と Apple File Exchanger の 2 種類がありあます。Sun および HP については、SunOS や HP-UX による DOS フォーマットのディスクの読み込みおよび書き込みを補助するための PC File System（PCFS）があります。

VI を VI ライブラリに保存しておくと、プラットフォーム間での転送が容易になります。VI をライブラリに保存するためには、オプション付き保存ダイアログボックスで Development Distribution オプションを選択します。そうすると、`vi.lib` の VI 以外のすべての VI、制御器、および外部のサブルーチンを 1 つのライブラリに保存することが可能になります。

印刷について

フロントパネルの1つの制御器（たとえば1つのグラフ）だけを印刷するには？

フロントパネルの 1 つのグラフだけを印刷するためには、フロントパネル上でグラフを使用したサブ VI を作成します。

1. グラフを表示器から制御器に変更します。
2. サブ VI を開き、**操作→VI 終了後に印刷**を選択します。
3. サブ VI にコネクタを割り当て、メインの VI のグラフからサブ VI のグラフにデータを渡します。

メインの VI がサブ VI を呼び出すたびに、グラフが自動的に印刷されます。

文字列を印刷するには？

プリンタが接続されているポート（PC の場合は `LPT1`、`LPT2`、等。Macintosh の場合はプリンタポート）を `Serial Port Init.vi` を使用して初期化し、初期化したポートに `Serial Port Write.vi` を使用して文字列を書き込みます。プリンタは、ポートのデータを読み取り、印刷します。通常、この操作を実行するためにはプリンタのコマンド言語に関するある程度の知識が必要になりますが、LabVIEW や BridgeVIEW で開発した多くのアプリケーションにはこの方法が有効です。詳細については、LabVIEW をご使用の場合は『LabVIEW ユーザマニュアル』の「付録 B」を参照してください。

（Windows および UNIX）System Exec VI を使用してコマンドライン関数を通じてファイルを印刷します。この VI は、**関数→通信**に入っています。たとえば、Windows NT/95 環境でテキスト文書を印刷する場合は、下記のコマンドを使用して System Exec VI を実行します。

```
notepad /p <document name>
```

ノートパッドの上限は32K です。文書のサイズがこれよりも大きい場合は、別のテキストエディタを使用してください。

（Macintosh）AESend Print Document VI を使用すると、文書を印刷するよう他のアプリケーションに命令することができます。この VI は、**関数→通信→** AppleEvent に入っています。詳しくは、「第 5 章　VI の印刷および文書作成」の「その他の印刷方法を使用する」の項を参照してください。また、前の質問の項で説明したように、プログラムを使用して印刷する方法を使用することもできます。

波形チャートのすべてのデータを印刷するには？

フロントパネルを印刷する際には、表示されているデータだけが波形チャートに印刷されます。チャートに履歴バッファのデータも含めてすべてのデータを印刷するためには、まず最初にチャートをポップアップしてチャートの *x* 軸の自動スケール調整を実行したのち、チャートのパレットを表示して *x* 軸をロックするか、またはチャートの属性ノードを使用してプログラム上で *x* 軸を設定する必要があります。

印刷時に LabVIEW あるいは BridgeVIEW がクラッシュする理由は？

Windows では、このクラッシュにはビデオドライバかプリンタドライバが関与している可能性があります。

対処方法については、この付録の「エラーメッセージとクラッシュについて」の項の各プラットフォームでの突発的なクラッシュに関する項を参照してください。

PostScript 印刷の使用方法は？

プリンタが PostScript 言語対応である場合は、まず最初に PostScript プリンタドライバをインストールし、プリンタ用のドライバとして選択します。LabVIEW または BridgeVIEW で**編集→環境設定→印刷**を選択したのち、PostScript 印刷を選択します。

PostScript プリンタには次のような利点があります。

- PostScript 印刷では、画面の画像をより正確に再現できます。
- PostScript 印刷では、パターンや線のスタイルをより正確に再現できます。

PostScript 印刷を選択したときに印刷ページのいちばん上にテキストがまとめて印刷されるのはなぜ？

印刷時には、VI の印刷データは PostScript（.ps）フォーマットに変換され、PostScript テキストとして Windows のプリンタドライバに渡されます。プリンタドライバが PostScript 対応でない場合は、グラフィックス画像ではなく PostScript ファイルの実際のデータが印刷されます。詳しくは、PostScript に関する前の質問の項を参照してください。

VIを印刷する際にラベルのテキストやフロントパネル制御器の一部が印刷されないのはなぜ？

これは、プリンタ用のフォントサイズがモニタ上でのフォントサイズと一致しない場合発生します。

- プリンタが PostScript 言語対応である場合は、高品質の印刷が得られるようにプリンタに対して PostScript プリンタドライバを使用します。PostScript 印刷については、詳しくは上記の質問の項を参照してください。
- フォントの不一致が原因でテキストが拡大されてもテキストが制御器やラベルの枠内に収まるように、ラベルやフロントパネル制御器を拡大します。
- モニタ上でのサイズとプリンタ上でのサイズが同じフォントを探します。プリンタの多くは、このプロセスをサポートするためのフォント代用アルゴリズムを内蔵しています。
- **（Windows）**モニタで見ているときとまったく同じように印刷できるビットマップ印刷を使用します。

（Windows）ビットマップ印刷の選択方法は？

編集→環境設定を選択したのち、**印刷**を選択します。画面上の**ビットマップ印刷**を選択します。

その他のことがらについて

info-labview-j とは何か、またその加入方法は？

`info-labview-j` は、ユーザが主催しサポートする LabVIEW とユーザのネットワークです。ユーザが所定のメールアドレス（下記参照）に情報を送ると、その情報がリストに登録されているすべてのユーザに転送されます。情報を受け取ったユーザは、ユーザグループを対象として返答したり、情報を発信した本人に直接返答することができます。

何人かの熟練ユーザのほかにナショナルインスツルメンツの従業員もリストに登録されており、さまざまな問題に対応しています。Info-LabVIEW グループは、これまで非常に良好に受け入れられてきました。このグループはナショナルインスツルメンツとは無関係ですが、随時グループに対して公式な対応がなされています。ほとんどの場合、通信は原則としてユーザ間で行われます。

リストに登録を希望される場合は、下記のアドレスに電子メールをお送りください。

`majordomo@lists.twics.com`

ユーザは、標準形式とダイジェスト形式という 2 通りの方法で情報を受け取ることができます。標準形式では、だれかが情報を発信するたびにその情報を受け取ることができます。ダイジェスト形式では、毎日の終わりにその日に発信されたすべての情報をまとめて受け取ることができます。毎日まとめて情報を受け取りたい場合は、`info-labview-request` に送信するメールでダイジェストフォーマットと指定してください。

標準形式で受信したい場合、`majordomo@lists.twics.com` に送信する電子メールに `subscribe info-labview-j` と記載してください。

ダイジェスト形式で受信したい場合、`majordomo@lists.twics.com` に送信する電子メールに `subscribe info-labview-j-digest` と記載してください。

リストへの登録が済むと、他の加入者からの情報を受け取ることができます。グループは活発に情報交換を行っており、毎日 10 ～ 30 件の情報が発信されています。リストに情報を送りたいときは、下記のアドレスに電子メールをお送りください。

`info-labview-j@lists.twics.com`

VI を動的にロードし、実行するには？

サブ VI が別の VI のブロックダイアグラムに存在する場合、発呼者である VI がメモリにロードされると、ただちにそのサブ VI もメモリにロードされます。ときには、メモリスペースの関係上、プログラムの実行中に VI を動的にメモリに VI をロードしたり、メモリからアンロードしたりすることが必要になる場合があります。そのような場合は、VI サーバ機能を使用します。詳しくは、本書の「第 21 章　VI サーバ」を参照してください。

（Macintosh）Macintosh では、プラットフォームに拘束されない VI サーバだけでなく、Apple Events を使用して動的に VI をロードし、実行することができます。VI は、**関数→通信→** AppleEvent パレット、またはそのサブパレットである LabVIEW Specific Apple Events に入っており、次のような名前が付いています。

- AESend Finder Open
- AESend Open, Run, Close VI
- AESend Run VI
- AESend Close VI

サブ VI を同じ名前を持つ別の VI と差し替えるには？

LabVIEW と BridgeVIEW は、あらゆる VI を名前で参照するため、VI のパスがどんなパスであっても同じ名前の 2 つの VI をロードすることはありません。これは、サブ VI にもあてはまります。サブ VI に対して**入れ換え**を選択し、そのサブ VI を同じ名前の VI と差し替える場合、G はすでにその VI のコピーがメモリに存在するため、VI をそれ自身と差し替えます。

サブ VI の新しいコピーをロードするためには、あらかじめ古いコピーをメモリから消去しておく必要があります。VI は、現在ウィンドウメニューにリストされています。サブ VI を最も簡単な方法でメモリから消去するには、まず最初にサブ VI、およびそのサブ VI を呼び出すすべての VI を閉じます。次に、サブ VI の新しいコピーを開いてメモリにロードし、それが必要なバージョンであることを示すために**ファイル→ VI 情報の表示 ...** をチェックしたのち、再度メインの VI を開きます。VI がサブ VI にリンクする際には、メモリでサブ VI の古いバージョンではなく新しいバージョンを探し出し、その新しいバージョンを使用します。

このような問題の多くは、サブ VI に常に固有の名前を割り当てることによって解決できます。

同じ名前を持つ異なる 2 つのサブ VI を呼び出す 2 つの VI をロードすると、問題が発生します。

たとえば、`main1.vi` と `main2.vi` が、`subVI.vi` という同じ名前を持つ異なる 2 つのサブ VI を呼び出すものとします。

1. `main1.vi` がメモリにロードされると、この `main1.vi` が呼び出す `subVI.vi` もロードされます。
2. `main2.vi` がメモリにロードされると、リンカ情報は `subVI.vi` をロードするよう LabVIEW あるいは BridgeVIEW に指示します。アプリケーションは、`subVI.vi` がすでにメモリにロードされているため、`main2.vi` をこの `subVI.vi` にリンクしようとします。このとき、コネクタペーンがまったく同じであると、リンクが確立されます（ただし、

意図した動作は行われません）。コネクタペーンが違っている場合は、リンク不良エラーが返されます。

対策：サブ VI に常に固有の名前を割り当てます。

優先順位が VI の実行に与える影響とは？

G の実行システムは、複数の VI やサブダイアグラムを並列で実行することができます。優先順位には、バックグラウンド（最下位）、低（通常）、中、高、および最高（最上位）という 5 つのレベルがあり、VI にはそのいずれかのレベルを割り当てることができます。優先順位のもう 1 つのレベル、すなわちサブルーチンレベルでは、VI は実行システムのスレッドを他の VI と共有することなく最後まで実行されます。各実行システムには、システム内の VI のコードセクションの待ち行列があります。待ち行列内の項目は優先順位に基づいて順番が決定され、優先順位の高い VI が優先順位の低い VI よりも先に実行されます。待ち行列では、優先順位の高い項目が存在する間は優先順位の低い項目は前に進めないため、実行することはできません。つまり、優先順位の高い VI がタイムアウトや他の非同期の動作の実行によって中断されることなく継続して実行されると、優先順位の低い VI に時間が割り当てられなくなる可能性があります。そのため、VI の優先順位を上げる際には十分注意する必要があります。

G の実行待ち行列には、優先順位の 5 つの各レベルごとにエントリポイントがあります。ただし、サブルーチンレベルのエントリポイントはありません。サブルーチンレベルの VI は、VI から呼び出された時点で実行がスタートし、異なる優先順位レベルの発呼者 VI に戻るまで実行システムのスレッドを独占します。サブルーチンレベルの VI は、別のサブルーチンレベルの VI を呼び出すことはできますが、それ以外の優先順位レベルの VI を呼び出すことはできません。サブルーチンレベルの VI は、表示器を更新せず、Waits やその他の非同期オペレーションを実行することはできないため、待ち行列に入れることはできません。そのため、これらの VI は、いったん実行を開始すると実行システムにおいて最も高い優先順位が与えられます。

マルチスレッドシステムには複数の実行システムがあり、各実行システムごとに 1 つの待ち行列と 1 つまたは複数のスレッドがあります。スレッドは、優先順位の高いスレッドの実行が可能なときでもオペレーティングシステムが優先順位の低いスレッドを優先できるような形で、優先順位に基づいて実行されるように設定されます。VI の優先順位は、VI をどの実行システムの待ち行列に入れるのが最も適しているか選択するために使用されます。マルチスレッドシステムで呼び出されるサブルーチンレベルの VI は、呼び出されたときのスレッドを独占しますが、他のスレッドの VI の実行を妨げることはできません。このことは、サブルーチンレベルの VI を使用しても、優先順位が同等かまたはそれより高い実行システムの他のすべての VI の実行をブロックすることはできないことを意味します。

マルチスレッドシステムでは、ユーザインタフェースは「通常」の優先順位のスレッドです。システムによっては、優先順位の高い VI がタイムアウトや他の非同期の動作の実行のために中断されることなく継続して実行されると、ユーザインタフェースのスレッドに時間が割り当てられなくなる場合があります。優先順位に差がありすぎると、ユーザインターフェースのスレッドの実行が妨げられる場合もあります。そのため、VI の優先順位を上げる際には十分な注意が必要です。

ポーリングを使用しないで割り込みを実行させるには？

オカーレンスは、コードの一部に特定のイベントを待たせるための手段です。たとえば、DLL でイベントが発生したときにオカーレンスを生成するプロセスを発生させる（Windows では DLL を通じて）CIN を作成することができます。ダイアグラムのコードの特定の部分が実行中であると、他の部分または VI はイベントの発生を待ちます。イベントが発生すると、オカーレンスが LabVIEW または BridgeVIEW に記録され、待ち行列で該当するコードに順番が回ってきた時点でそのコードが実行されます。

内部では、LabVIEW および BridgeVIEW はコードの一部の実行が完了すると、待ち行列をチェックします。これは、ほかに実行中のものがある場合には、それが終了するのを待ってオカーレンスすなわち割り込みを処理しなければならないことを意味します。

☞ **注**　**特に、実行が完了するまでに非常に長い時間を要する可能性のある CIN が VI に含まれる場合でも、終了までの時間の長さには明確な制限がないことに注意してください。**

ネスト構造の While ループを使用している場合に、内側のループが停止したあとに外側のループで何も実行せずに両方のループを停止させるには？

最後の回に実行したくないコードは、Case ストラクチャに格納します。内側のループの条件端子が Case ストラクチャの選択端子に接続されている場合は、これが最良の方法です。

パネル順序とは？

編集→パネル順序は、フロントパネル上のオブジェクトの順序を決定します。言い換えると、キーボードの<Tab>を使用してオブジェクト間を移動する際には、パネル順序で決定された順序に従ってオブジェクトからオブジェクトにキーフォーカスが移動します。キーフォーカスを獲得できるのは制御器に限られ、<Tab>キーを押してもキーフォーカスは表示器には移りません。デフォルトでは、フロントパネル上でオブジェクトを作成した順序がパネル順序になります。

フロントパネルのデータロギングを使用している場合は、パネル順序はファイル内のクラスタにさまざまなオブジェクトを格納する際の順序を決定します。

LabVIEW あるいは BridgeVIEW に特定の VI を自動的に起動させるには？

LabVIEW および BridgeVIEW は、デフォルトでは起動時に各称未設定 VI が開かれ、作成されます。一方、起動時に自動的に特定の VI を開くように設定することもできます。また、VI はロード時に実行されるように設定することもできるため、LabVIEW あるいは BridgeVIEW の起動時に特定の VI を開いて実行するように設定することもできます。VI を開いたときにその VI が実行されるようにするためには、**VI 設定** ... で必要な**実行オプション**を設定します。起動時にライブラリ（.llb）が開くように設定した場合は、**最上位**に設定されたすべての VI が開きます。VI を**最上位**に設定するためには、**ファイル→ VI ライブラリの編集** ... を使用します。ライブラリを指定したときに**最上位**に設定された VI がない場合は、起動時にそのライブラリ内の VI を選択するためのダイアログボックスが表示されます。

LabVIEW あるいは BridgeVIEW の起動時に特定の VI が開かれるようにするには、次に示した各プラットフォームでの手順に従ってください。

（Windows）LabVIEW または BridgeVIEW へのショートカットのアイコンを選択したのち、（エクスプローラーで）**ファイル→プロパティ**を選択します。**リンク先**で、パス名を開きたい VI のパス名に変更します。たとえば、起動時に test.vi をロードしたいときは、**リンク先**を次のように設定します。

```
c:¥labview¥labview.exe test.vi
```

test.vi が test.llb というライブラリに入っている場合は、次のように設定します。

```
c:¥labview¥labview.exe "test.llb¥test.vi"
```

VI の完全パスを指定することが必要になる場合もあります。

コマンドライン**スタート→ファイル名を指定して実行** ... から VI を開きたい場合、構文は次のようになります。

```
c:¥labview¥labview.exe <path to VI relative to the
labview directory>
```

たとえば、c:¥labview¥examples ディレクトリの Readme.vi を開く場合のコマンドは、次のようになります。

```
c:¥labview¥labview.exe examples¥readme.vi
```

VI が別のパスに存在する場合は、VI の完全パスを指定する必要があります。

```
c:¥labview¥labview.exe c:¥coolapp¥mycool.vi
```

パスにスペースを使用している場合は、パスをクオーテーションマークで囲む必要があります。

```
c:¥labview¥labview.exe "c:cool application¥mycool.vi"
```

VIがLLBに入っている場合は、下記のいずれかの方法で指定します。

1. フルパスをクォーテーションマークで囲んで指定します。

```
c:¥labview¥labview.exe "c:¥coolapp/eagle.llb¥mycool.vi"
```

または、

2. VIのVI設定を**開かれたら実行**に変更します。次に、**ファイル→VIライブラリの編集**...を選択し、VIを**最上位**に設定します。

この場合は、.llbのパスを指定するだけで済みます。

```
c:¥labview¥labview.exe "c:¥coolapp¥eagle.llb"
```

LabVIEWの起動時に自動的にVIを開くように設定するには、ショートカットプロパティを変更し、上に示すようにリンク先にパスを指定します。

（Macintosh）LabVIEWの起動時に特定のVIが自動的にMacintoshにロードされるよう指定する手段はありません。ただし、FinderでVIのアイコンをダブルクリックすることでVIを開くことができます。マシンにLabVIEWの複数のコピーがインストールされている場合は、VIをダブルクリックしたときにどのコピーを起動するかはFinderが判断します（起動するコピーを指定することはできません）。たとえば、Run-Time System と Full Development Systemの両方がインストールされている場合には、Run-Time Systemが起動されることもあります。System 7以降のオペレーティングシステムを使用している場合は、ドラッグアンドドロップで1つまたは複数のVIあるいはVIライブラリを開くことができます。

（UNIX）LabVIEWは、コマンドラインオプションに応答して起動時に特定のVIを開きます。たとえば、Test VIを開くようにしたいときは、そのパスに基づいて次のようにタイプします。

```
labview /usr/home/test.vi
```

または、

```
labview /usr/home/test.llb/test.vi
```

LabVIEWの起動時に特定のVIを開くコマンドを簡単なスクリプトを使用して作成することができます。

列挙体のエントリをプログラム上で変更するには？

列挙体データタイプのタイプ（文字列）をプログラム上で変更することはできません。これは、整数制御器をダブルに変更したり文字列制御器をパス制御器に変更したりできないのと同じです。Enum内の文字列は、その

データタイプの一部であるため、編集時以外は変更できません。属性ノードを使用して Enum の文字列を読み込むことは可能ですが、属性ノードを使用してこれらの文字列を書き込むことはできません。

Enum のテキストの値をプログラムを使用して変更したいときは、かわりにテキストのリング制御器を使用します。リング制御器を使用すると、プログラム上で Strings[] 属性を通じた文字列の読み込みおよび書き込みが可能になります。

VI のメニューバーを表示しないようにするには？

メニューバーやツールバーを表示するかしないかを含め、VI のすべての属性は**VI 設定...**ダイアログボックスで設定します。VI設定...ダイアログボックスを呼び出すには、フロントパネルの右上にある VI のアイコンをポップアップしたのち、**VI 設定...**を選択します。詳しくは、「第 6 章　VI およびサブ VI をセットアップする」を参照してください。

パフォーマンスを上げるためにマウスの割り込みを不可能にするには？

マウスを動かしたり、マウスで項目をクリックしたりすることによって発生する割り込みは、CPU の時間を消費します。非常にタイムクリティカルな一部のアプリケーションでは、この割り込み動作によってデータが失われたり、アプリケーションに何らかの問題を生じさせる可能性があります。マウスの割り込みを不可能にする方法としては、マウスのプラグを抜く以外に望ましい方法はありません。アプリケーションのタイムクリティカルな部分の実行中にマウスを動かさないようにすれば、パフォーマンスへの影響を最小限に抑えることができます。

C

カスタマーコミュニケーション

この付録には、お客様の技術的問題点を弊社が解決する際に必要な情報をご記入していただく書式と、ドキュメントに関するお客様のご意見やご要望をご記入いただく書式が添付されています。弊社にお問い合わせいただく際には、「テクニカルサポートフォーム」と、システムに関する構成フォーム（お客様のマニュアルに添付されている場合）を事前にご記入のうえご連絡いただきますと、お客様のお問い合わせにさらに迅速に対応できます。

ナショナルインスツルメンツでは、お客様が必要とする情報を速やかに提供するため、電子メール、ファックス、電話によるテクニカルサポートを提供しています。弊社の電子サポートには、FTP サイト、ファックスバックシステムおよび電子メールサポートなどがあります。お持ちのハードウェアまたはソフトウェアに問題が発生した場合は、まず電子サポートシステムをお試しください。そのシステムから入手できる情報では問題が完全に解決しない場合は、ファックスまたは電話サポートをご利用ください。弊社のアプリケーションエンジニアがご質問にお答えします。

電子サポート

Web サポート

弊社の Web サイトにアクセスするには、http://www.natinst.com/nni のアドレスをご利用ください。

FTP サポート

弊社のFTPサイトにアクセスするには、弊社のインターネットホスト `ftp.natinst.com`にanonymousでログオンし、パスワードはお客様のインターネットアドレス（`nihanako@anywhere.co.jp`など）をお使いください。サポートファイルおよび文書は、`/support`ディレクトリにあります。

電子メールサポート

以下のインターネットアドレスで、技術的なご質問を弊社アプリケーションエンジニア宛に電子メールでお送りいただくこともできます。その際には、お客様のご氏名、ご住所、電話番号、および電子メールアドレスを忘れずにご記入ください。折り返し解決方法やご提案を返答します。
`support@nni.co.jp`

なお、一般的なご質問は`info@nni.co.jp`へお送りください。

Fax-on-Demand サポート

Fax-on-Demand は 24 時間体制の情報検索システムで、広範な技術情報に関する文書ライブラリを収録しています。Fax-on-Demand にアクセスするには、プッシュホンで以下の番号にダイヤルしてください。資料はすべて英文になります。
512-418-1111（米国）

電話およびファックスサポート

ナショナルインスツルメンツの支社は世界各地にあります。以下のリストから、国内の技術サポートの番号をお探しください。国内に支社がない場合は、お客様がソフトウェアをご購入された販売店にご連絡いただき、サポートをご依頼ください。

国名	電話番号	Fax 番号
日本	03 5472 2981	03 5472 2977
イスラエル	03 573 4815	03 573 4816
イタリア	02 413091	02 41309215
英国	01635 523545	01635 523154
オーストラリア	03 9879 5166	03 9879 6277
オーストリア	0662 45 79 90 0	0662 45 79 90 19
オランダ	0348 433466	0348 430673
カナダ（オンタリオ）	905 785 0085	905 785 0086
カナダ（ケベック）	514 694 8521	514 694 4399
韓国	02 596 7456	02 596 7455
シンガポール	2265886	2265887
スイス	056 200 51 51	056 200 51 55
スウェーデン	08 730 49 70	08 730 43 70
スペイン	91 640 0085	91 640 0533
台湾	02 377 1200	02 737 4644
デンマーク	45 76 26 00	45 76 26 02
ドイツ	089 741 31 30	089 714 60 35
ノルウェー	32 84 84 00	32 84 86 00
フィンランド	09 725 725 11	09 725 725 55
フランス	01 48 14 24 24	01 48 14 24 14
米国	512 795 8248	512 794 5678
ベルギー	02 757 00 20	02 757 03 11
香港	2645 3186	2686 8505
メキシコ	5 520 2635	5 520 3282

テクニカルサポートフォーム

この書式はコピーして使用し、ソフトウェアまたはハードウェアの構成を変更するたびにその内容も書き換えてください。ご記入いただいた書式はお客様の現在の構成を参照する資料としてご利用いただけます。技術サポートをご希望の場合は、この書式に必要事項をもれなくご記入のうえご連絡いただきますと､弊社のアプリケーションエンジニアがお客様のご質問にさらに効率よくお答えできます。

ご質問いただいた問題に関連して、弊社の他のハードウェアもしくはソフトウェア製品が使用されている場合には、該当する製品のユーザマニュアルに添付されている構成フォームもご記入ください。

1 ページに納まらない場合は、必要に応じて用紙を追加してください。

ご氏名 _____

貴社名/部課名 _____

ご住所 _____

電話(____) _____ ファックス（____) _____

コンピュータのメーカー名_____ 機種_____CPU タイプ_____

オペレーティングシステム（バージョン番号もご記入ください）

クロック周波数_____MHz　RAM _____MB　ディスプレイアダプタ_____

マウス___有り___無し　インストールされている他のアダプタ _____

ハードディスク容量_____MB　メーカー _____

使用している計測器 _____

ナショナルインスツルメンツのハードウェア製品_____ レビジョン番号 _____

構成 _____

ナショナルインスツルメンツのソフトウェア製品_____バージョン _____

構成 _____

問題点を詳しく説明してください。_____

_____

_____

_____

_____

_____

_____

表示されるエラーメッセージを記入してください。_____

_____

_____

どのような手順で操作すると問題が発生しますか。_____

_____

_____

_____

_____

ハードウェアおよびソフトウェアに関する構成フォーム

各項目の右側の空欄に、ご使用のハードウェアおよびソフトウェアの構成とレビジョン番号をご記入ください。この書式はコピーして使用し、ソフトウェアまたはハードウェアの構成を変更するたびにその内容も書き換えてください。、ご記入いただいた書式はお客様の現在の構成を参照する資料としてご利用いただけます。技術サポートをご希望の場合は、この書式に必要事項をもれなくご記入いただいてからご連絡いただきますと、弊社のアプリケーションエンジニアがお客様のご質問にさらに効率よくお答えすることができます。

ナショナルインスツルメンツの製品

DAQ ハードウェア _____

ハードウェアの割り込みレベル _____

ハードウェアの DMA チャネル _____

ハードウェアのベース I/O アドレス _____

ナショナルインスツルメンツ社のソフトウェア _____

システムで使用している他のボード _____

他のボードのベース I/O アドレス _____

他のボードの DMA チャネル _____

他のボードの割り込みレベル _____

他社の製品

コンピュータの型式とモデル名 _____

CPU _____

クロック周波数 _____

インストールされているビデオボードのタイプ

オペレーティングシステムのバージョン _____

オペレーティングシステムのモード _____

プログラム言語 _____

プログラム言語のバージョン _____

システムに搭載されている他のボード _____

他のボードのベース I/O アドレス _____

他のボードの DMA チャネル _____

他のボードの割り込みレベル _____

マニュアルについてのご意見をお聞かせください

ナショナルインスツルメンツでは、弊社製品のマニュアルについてお客様からのご意見をお待ちしております。このような情報は、お客様のニーズを満たす品質の高い製品を提供する貴重な資料として活用させていただきます。

マニュアル名： LabVIEW™ G プログラミングリファレンスマニュアル

発行： 1998年7月

製品番号： 321993A-01

マニュアルの完成度、わかりやすさ、構成についてのご意見をお聞かせください。

マニュアルに誤りを見つけられた場合は、そのページ番号を明記し、その誤りの内容をご説明ください。

ご協力ありがとうございました。

ご氏名 _____

役職名 _____

貴社名 _____

ご住所 _____

電話 _____

送付先 〒105-0011
東京都港区芝公園2-4-1
秀和芝パークビルB館5F
日本ナショナルインスツルメンツ株式会社
技術部テクニカル出版課
Fax：03-5472-2977

用語集

接頭語	意味	値
n-	ナノ	10^{-9}
m	ミクロ	10^{-6}
m-	ミリ	10^{-3}

記号

∞　無限。

π　パイ。

Δ　デルタ。偏差。Δx は、ある指標から次の指数までx の変化値を示します。

A

ANSI　American National Standards Institute （米国規格協会）。

ASCII　American Standard Code for Information Interchange（情報交換用米国標準コード）。

ATE　Automatic test equipment （自動テスト装置）。

B

BNF　Backus-Naur（バッカスナウル）形式。コンピュータサイエンスにおける言語文法の一般的な表記法です。

C

Case ストラクチャ　入力値によって、実行するサブダイアグラムが異なる構造。制御フロー言語の、IF、THEN、ELSE およびCASE 文に似ています。

CIN　「コードインタフェースノード」の項を参照。

CPU　Central processing unit （中央処理装置）。

D

DUT	Device under test （テスト中のデバイス）。

F

FFT	Fast Fourier transform （高速フーリエ変換）。
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構造。 次のような従来のコードに相当します。`For I=0 to n-1, do･･･`

G

G	LabVIEWおよびBridgeVIEWアプリケーションの開発に使用するグラフィカルプログラミング言語。
GPIB	General Purpose Interface Bus （汎用インタフェースバス）。ANSI/IEEE 規格 488.1-1987 と ANSI/IEEE 規格 488.2-1987 で規定されている通信インタフェースシステムの一般的な名称のことです。このバスを開発したヒューレットパッカード社ではこれを HP-IB と呼んでいます。

H

hex	16 進数。基数を 16 とする数。
Hz	ヘルツ。1 秒あたりのサイクル数。

I

I/O	Input/Output（入出力）。通信チャネル、オペレータ入力装置、またはデータ集録インタフェースおよびデータ制御インタフェースを利用して、コンピュータシステムとの間でデータを転送すること。
IEEE	Institute for Electrical and Electronic Engineers（米国電子電気技術者協会）。
Inf	無限を意味する浮動小数点のデジタル表示値。

L

LED	Light-emitting diode（発光ダイオード）。

M

MB	Megabyte。メモリのメガバイト数。

N

NaN	「数字ではない」を表す浮動小数点のデジタル表示値。通常は未定義の演算の結果で、log（-1）のように表示されます。
not-a-path	パス制御用にあらかじめ定義された値で、パスが無効であることを意味します。
not-a-refnum	パス制御用にあらかじめ定義された値で、refnum（参照番号）が無効であることを意味します。

R

refnum	関連する VI によって参照できる DDE 会話またはオープンファイルの識別子。

U

UUT	Unit under test （テスト中の装置）。

V

VI	virtual instrument。「バーチャルインスツルメンツ（仮想計測器）」の項を参照。
VI アプリケーションクラス	バーチャルインスツルメンツをロードできるアプリケーションについての参照物。アプリケーションの属性とアプリケーションの方法が参照できます。
VI クラス	バーチャルインスツルメンツについての参照物。VI の属性と VI の方法が参照できます。
VI サーバ	VI および G アプリケーションをプログラムで制御するためのメカニズム。VI と G アプリケーションを遠隔操作するのに使用することも可能です。

VI文字列ファイル　タグ付きのテキストファイルで、VIのフロントパネルにあるすべてのローカリゼーション文字列が入っています。

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。

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。従来のプログラミング言語のDoループやRepeat-Untilループに相当します。

あ

アイコン　ブロックダイアグラム上のノードを図式的に表したもの。

アイコンエディタ　VIアイコンを作成するためのインタフェースで、ペイントプログラムのインタフェースに似ています。

アイコンペーン　フロントパネルとブロックダイアグラムの右上隅にある領域で、VIアイコンを表示します。

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。通常は一番上に表示されているウィンドウで、アクティブウィンドウのタイトルバーはハイライト表示となります。ウィンドウをアクティブにするには、アクティブにするウィンドウをクリックするか、ウィンドウメニューからウィンドウを選択します。

位置決めツール　オブジェクトの移動、選択、サイズ変更に使用するツール。

インプレース実行　メモリを追加して割り当てないでも、関数またはVIがメモリを再利用できる機能。

オートスケーリング　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定します。

オブジェクト　制御器やノード、ワイヤ、インポートしたピクチャなど、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。

オブジェクトポップアップメニューツール　オブジェクトのポップアップメニューにアクセスするために使用するツール。

か

階層ウィンドウ	VIとサブVIの階層を図で表示するウィンドウ。
階層パレット	パレットおよびサブパレットを含むメニュー。
外部ルーチン	「共有外部ルーチン」の項を参照。
カウント端子	Forループの端子で、カウント端子の値により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。
カスタムPICT制御器とカスタムPICT表示器	標準の制御器および表示器をもとに、ユーザが提供するグラフィックなどを追加または変更して作成した特殊な制御器および表示器。
カラーコピーツール	カラーツールで貼り付ける色をコピーします。
カラーツール	前景色および背景色を設定するのに使用するツール。
空の配列	データタイプは定義されていますが、要素が0の配列。例えば、データディスプレイウィンドウに数値制御器はありますが、要素のどれ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で、従来の言語の演算子、関数または文に相当します。
関数パレット	ブロックダイアグラムストラクチャ、定数、通信機能、およびVIを含むパレット。
疑似コード	簡易言語から独立したプログラミングコードの表記法。
キャスト	データ自体の内容を実質的に変えずに、データ要素のタイプデスクリプタを変更すること。
キャプションラベル	ユーザインタフェースのオブジェクト名をつけるのに使用するフロントパネルオブジェクト。
競合状態	並行して実行する2つ以上のコードが共有するリソースの値（通常は、グローバル変数またはローカル変数）を変更すると発生します。
強制	データ要素の数値表現を変更するため、Gが実行する自動変換。
強制ドット	ノードまたは端子上にあるグリフ（絵文字）。その点でデータ要素の数値表現が変化することを表します。
共有外部ルーチン	複数のCINコードリソースで共有できるサブルーチン。
行列	二次元の配列。

クラスタ	数値、ブール、文字列、配列、またはクラスタなど、順番に並べられていますが、指標付けされていない任意データタイプの要素の集合。クラスタ要素は全部制御器、または全部表示器のいずれかでなければなりません。
クラスタシェル	クラスタ要素が入っているフロントパネルオブジェクト。
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。
クローン化	制御器や G オブジェクトをコピーすること。<Ctrl>（**Windows**）、<option>（**Macintosh**）、<meta>（**Sun**）または<Alt>（**HP-UX**）キーを押したまま、制御器またはその他の G オブジェクトをクリックして新しい位置までコピーをドラッグします。
計測器ドライバ	プログラム可能な計測器を制御する VI。
ケース	Case ストラクチャの 1 つのサブダイアグラム。
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。
コードインタフェースノード	CIN。ブロックダイアグラムの特殊ノードで、従来のテキストベースのコードを VI にリンクすることができます。
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。データはコネクタを経由してノード間で受け渡しが行われます。
コネクタペーン	フロントパネルウィンドウの右上隅にあり、VI 端子のパターンを表示します。通常はアイコンペーンの下に隠れています。
壊れた VI	コンパイルや実行ができない VI。実行ボタンに壊れた矢印が表示されます。
コンパイル	ハイレベルコードをマシンが実行可能なコードに変換するプロセス。VI を作成または変更後初めて実行する場合、実行前に自動的にコンパイルされます。

さ

最上位VI	VI階層の最も上位にあるVI。この語によってこのVIとそのサブVIを区別します。
サイズ変更ポインタ	オブジェクトの隅に表示されるL字部分で、サイズ変更点を示します。
再入可能実行	1つのサブVIの複数インスタンスに対する呼び出しが、別のデータ記憶域を利用して並行して実行できるモード。
サブVI	別の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	ストラクチャ境界内にあるブロックダイアグラム。
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構造。一般的には、データに依存しないノードを指定した順番に強制実行するために使用します。
シーケンスローカル	シーケンスストラクチャのフレーム間で、データを受け渡す端子。
次元	配列のサイズおよび配列構成の属性。
実行ハイライト	VIのデータフローを示すためVIの実行を図式的に表示する機能。
自動サイズ調整	入力されたテキストに合わせて、ラベルのサイズを自動的に変更します。
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。「指標付け使用」を選択した状態で、配列がループに入ると、ループは配列を自動的に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。配列がループを出るとき、その逆の手順に従ってループはデータを配列に組み立てます。
シフトレジスタ	オプションのループストラクチャのメカニズムで、ループの一つの繰り返しから次の繰り返しへ変数の値を渡すために使用します。
条件端子	ブール値を含むWhileループの端子で、このブール値によってVIが次の繰り返しを実行するかどうかが決定されます。
シンク端子	データを受け取る端子。接続先端子とも呼びます。
人工的データ依存	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。
スウィープチャート	スコープチャートに似ていますが、スウィープチャートは、ディスプレイを線で区切り、新しいデータと古いデータを区別します。

数値制御器と数値表示器	数値データの操作および表示、または入出力に使用され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のできる数字。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	ウィンドウのスクロールに使用するツール。
スケール	機械動作、チャート、およびグラフの制御器と表示器の一部で、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表示器。
スタブ VI	サブ VI のプロトタイプで、それ自体では機能せず、入力と出力がありますが不完全です。VI の設計の初期計画段階で、後で開発する VI の配置場所を確保するために使用します。
ストラクチャ	シーケンス、Case、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で、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。
制御器	対話形式で VI にデータを入力したり、プログラムによってサブ VI にデータを入力するため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および表示器を含むパレット。
制御フロー	命令の順序によって実行の順序が決定するプログラミング。C 言語や Pascal、BASIC など、従来からのテキストベース言語の多くは制御フロー言語です。
接続先端子	「シンク端子」の項を参照。
絶対パス	ファイルシステムの最上位に対する相対的な位置を表すファイルまたはディレクトリパス。
説明ボックス	G オブジェクトのためのオンラインドキュメント。
センサ	速度、温度、流量などの物理的特性を、電圧や電流出力として表示するデバイス。
操作ツール	制御器にデータを入力したり、制御器の操作に使用するツール。指をさす手の形をしています。

ソース端子	データを送り出す端子。
属性ノード	特殊なブロックダイアグラムノード。制御器と表示器の外観や機能を変更するのに使用します。

た

ダイアログボックス	プロンプトのある対話式画面。コマンドを完成するのに必要な情報を追加入力します。
タイプデスクリプタ	「データタイプデスクリプタ」の項を参照。
多形性	異なる表現、タイプ、またはストラクチャ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	データの受け渡しが行われるノード上のオブジェクトまたは領域。
チャート	「スコープチャート」、「ストリップチャート」、および「スウィープチャート」の項を参照。
ツール	特定の操作を実行するために使用する特殊カーソル。
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の入ったバー。
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のオブジェクトの編集およびデバッグに使用するツールの入ったパレット。
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。「人工的データ依存」の項も参照。
データ記憶形式	メモリ上に格納されるデータの配置と表記の方法。
データ集録	DAQ。データを集録するプロセス。一般的には A/D またはデジタル入力プラグインボードから集録します。
データタイプデスクリプタ	データタイプの識別コード。データの格納や表記に用いられます。
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取ったときだけ実行され、実行されると自動的に出力を生成します。G 言語はデータフローシステムです。
データロギング	一般的には、データを集録すると同時にそれをディスクファイルに保存すること。G のファイル I/O 関数はデータのロギングができます。

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続きのレコードとしてデータを保存するファイル。データログファイルのすべてのレコードは同一タイプでなければなりませんが、そのタイプが複合的であってもかまいません。例えば、各レコードを文字列や数値、配列を含むクラスタとして指定することができます。

テーブル駆動の実行　実行方法の一つで、各タスクは、While ループにおける Case ストラクチャ内の個別のケースの処理を行います。順序はケース番号の配列順に指定されます。

ドラッグ　オブジェクトを選択したり、移動、コピーまたは削除するために、スクリーン上のマウスカーソルを動かすこと。

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。

な

ニーモニック　整数値に対応する文字列。

ノード　関数、ストラクチャ、サブ VI などのブロックダイアグラムの実行要素。「データフロー」および「ワイヤ」の項を参照。

バーチャルインスツルメンツ（VI：仮想計測器）　グラフィカルプログラミング言語 G で書かれたプログラムで、実際の計測器の外観や機能を模倣しているためこう呼ばれます。

は

配線ツール　ソース端子とシンク端子間のデータパスを定義するために使用されるツール。

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保存するファイル。

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の集合。

配列シェル　配列が入っているフロントパネルオブジェクト。指標ディスプレイ、データオブジェクトウィンドウ、およびラベル（オプション）で構成され、様々なデータタイプを受け付けることができます。

ハウジング　スライダとスケールのあるフロントパネルの制御器および表示器の固定部分。

パレット　使用可能なオプションを表すアイコン表示。

ハンドル	ブロックメモリのポインタに対するポインタで、配列と文字列の参照を行います。文字列の配列は、文字列に対するハンドルが入っているメモリブロックを参照するハンドルです。
バンドルノード	様々なタイプの構成要素からクラスタを作成する関数。
凡例	チャートまたはグラフが所有するオブジェクトで、そのチャートまたはグラフにプロット名とプロット方式を表示します。
ピクセルマップ	画像を記憶するための標準フォーマットで、各ピクセルを色の値で表します。ビットマップは、ピクセルマップを白黒で表示したものです。
非同期実行	複数の処理がプロセッサ時間を共有するモード。例えば、他の処理がデバイス入出力中割り込み待ちの間、またはクロックのタイムアウト待ちの間に、1つの処理を実行します。
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、ワード、倍長整数のほか、単精度、倍精度および拡張精度の浮動小数点数があります。
表示器	出力を表示するフロントパネルオブジェクト。
表示不能文字	改行やタブなど、表示できないASCII文字。
ヒントラベル	オブジェクト名、制御器名、あるいは端子名を表示するテキストラベル。
ファイルの終わり	EOF。ファイルの初めに対するファイルの終わりの文字オフセット（すなわち、EOFはファイルのサイズです）。
ブール制御器とブール表示器	ブール（TRUEまたはFALSE）データの操作および表示または入出力に使用するフロントパネルオブジェクト。スイッチ、ボタン、LEDなど様々な種類があります。
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。ブロックダイアグラムの形で作成すると手間がかかる長い式には特に便利です。
プラットフォーム	コンピュータの基盤となるハードウェアもしくはソフトウェア。
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。
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。プルダウンメニューのオプションは一般的なものです。
ブレークポイント	デバッグを行うための一時的な実行の中断点。ブレークポイントを設定するには、ツールパレットのブレークポイントツールを使って、VIやノード、あるいはワイヤをクリックします。

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に使用するツール。

フレーム　シーケンスストラクチャのサブダイアグラム。

プローブ　VI の中間的な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のに使用するツール。

プログラム印刷　実行後、VI フロントパネルを自動的に印刷する機能。

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。G 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、ノード間でデータの受け渡しを行うワイヤで構成されています。ブロックダイアグラムは、VI のソースコードで、VI のダイアグラムウィンドウに常駐します。

プロット　データ配列を、グラフまたはチャートのいずれかで図式的に表すこと。

フロントパネル　VI の対話式ユーザインタフェース。実際の計測器のフロントパネルを模倣したもので、スイッチ、スライド、メータ、グラフ、チャート、ゲージ、LED またはその他の制御器および表示器で構成されています。

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、通常ファイルに書き込むためのものです。

ヘルプウィンドウ　関数またはサブ VI の端子名および位置、制御器や表示器の説明、ユニバーサル定数の値、制御器属性の説明およびデータタイプを表示する特殊なウィンドウ。

変換　データ構成要素のタイプを変更すること。

ポップアップ　オブジェクト上でマウスの右ボタンをクリックする（Windows および UNIX プラットフォームの場合）か、またはコマンドキーを押したままクリック（Macintosh）して、そのオブジェクトの専用メニューを呼び出すこと。

ポップアップメニュー　ポップアップしてアクセスするメニュー。通常はオブジェクトで、そのオブジェクトに特有のメニューオプションが付属しています。

ま

マーキー　選択したオブジェクトのまわりを囲む破線で、移動することができます。

メニューバー　メインメニューの名前を一覧表示する水平のバー。

モジュール式プログラミング　相互に交換性のあるコンピュータルーチンを使用するプログラミングのタイプ。

文字列制御器と文字列表示器　テキスト処理や表示、または入出力に使用するフロントパネルオブジェクト。

や

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 pi など、標準の数値定数を送り出す編集不可能なブロックダイアグラムオブジェクト。

ら

ラベリングツール　ラベルを作成して、テキストウィンドウにテキストを入力するために使用するツール。

ラベル　フロントパネルやブロックダイアグラム上の他のオブジェクトや領域に名前を付けたり、説明を加えたりするのに使用するテキストオブジェクト。

リング制御器　32 ビット整数を一連のテキストラベルまたはグラフィックスに結びつける特殊な数値制御器。0 から始まり順次増分します。

連続実行　オペレータが停止しない限り VI の実行を繰り返す実行モード。連続実行ボタンをクリックすると連続実行モードが実行可能になります。

わ

ワイヤ　ノード間のデータパス。「データフロー」の項も参照。

名前ラベル　オブジェクトに名前を付けたり、他のオブジェクトと区別するのに使うフロントパネルオブジェクトのラベル。オブジェクトの一部を構成する、端子、ローカル変数、および属性ノードに使用されます。

索引

記号

数値

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ぬ

ね

の

は

ひ

ふ

へ

ほ

ま

む

め

も

や

ゆ

よ

ら

り

る

れ

ろ

わ