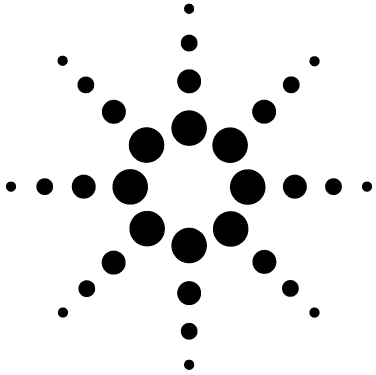


Agilent VEE Pro

VEE Proユーザーズ・ガイド

Agilent Technologies

注意

マニュアル番号

E2120-90014

本版について

Version 7.0
Eighth edition, 2004年2月

Printed in USA

Agilent Technologies, Inc.
815 14 Street SW
Loveland, CO 80537USA

保証

本書に記載されている内容は、「無保証で」提供されており、今後の版では予告なしに変更されることがあります。さらに、準拠法の許す最大限の範囲で、本書並びに本書の中に掲載されている情報に関しては、市場性および特殊目的に対する適合性を含め、それに限定されることなく、明示的にも暗黙的にも一切保証いたしかねます。本書または本書の中に掲載されている情報の誤り、あるいはそれらの提供、利用または内容に関連する付随的または間接的損害に対する責任は負わないものとします。アジレントとユーザがこれらの条件に抵触する別の保証条件契約を書面で結んでいる場合には、別の契約書の保証条件によって規制されるものとします。

技術ライセンス

本書に掲載されているハードウェアやソフトウェアは、ライセンス契約のもとで提供されており、当該ライセンスの諸条件に準拠した使用またはコピーだけが許可されています。

権利の制限について

ソフトウェアを米国政府元請負契約または下請契約を履行するために用いる場合には、ソフトウェアは、DFAR 252.227-7014(1995年6月)に規定されている「市販のコンピュータ・アプリケーション」、FAR 2.101(a)に規定されている「市販品」、FAR 52.227-19(1987年6月)または同等の政府機関の規制や契約条項に規定されている「制限付きコンピュータ・ソフトウェア」として提供およびライセンス供与されます。ソフトウェアの使用、複製または公開は、アジレント・テクノロジーの標準的な商業用使用許諾条件に準拠するものとし、米国の国防総省以外の省や政府機関には、FAR 52.227-19(c)(1-2)(1987年6月)に定義されている限定の権利以上の権利は与えられていません。該当する技術データに関しては、米国政府ユーザには、FAR 52.227-14 (1987年6月) またはDFAR 252.227-7015(b)(2)(1995年11月)に定義されている限定の権利以上の権利は与えられていません。

安全表示

注 意

「**注意**」表示は危険であることを示しています。正しく実行しなかったり守らなかった場合には、本製品に損傷を与えたり、重要なデータが失われるおそれのある操作手順や操作方法などに注意を喚起します。指示されている条件を完全に理解し、それらの条件に対応できるまで、**注意**表示の先には進まないでください。

警 告

「警告」表示は危険であることを示しています。正しく実行しなかったり守らなかった場合には、怪我や死亡事故につながるおそれのある操作手順や操作方法などに注意を喚起します。指示されている条件を完全に理解し、それらの条件に対応できるまで、警告表示の先には進まないでください。

VEE Proユーザーズ・ガイド

このマニュアルで使用する表記規則

このマニュアルでは、次の表記規則を使用します。

表1

File	コンピュータ・フォントの文字は、メニュー名、機能、ボタンなど、図で画面上に表示されるテキストを表します。
dir *filename*	この場合、コンピュータ・フォントの文字は示されたとおり正確に入力する引数を表し、斜体文字は実際の値で置き換えるべき引数を表します。
[File]⇒[Open]	「⇒」は、VEE機能のメニュー内の位置を簡単に表すために使用します。たとえば**[File]** ⇒ **[Open]**は、**[File]**メニューを選択し、次に**[Open]**を選択することを示します。
`Sml | Med | Lrg`	縦棒 (|)で区切られたコンピュータ・フォントの選択肢は、オプションの1つを選択する必要があることを示します。
Enterキーを押します。	この場合、太字はキーボード上のキーを表します。
`Ctrl + O`キーを押します。	キーボード上で同時に押すキーの組合わせを表します。

目次

2 Agilent VEEのプログラミング技術

5　テスト結果の保管方法と読み取り方法

6 ActiveXを使ったレポートの簡単な作成方法

7 VEEでの.NETの使用方法

8 PC用プログラムの統合方法

9 Agilent VEE 関数の使用方法

10 テストのシーケンスを決定する方法

11 オペレータ・インタフェースの使用方法

12 Agilent VEEプログラムの最適化

付録A: 追加の例題

用語集

図目次

はじめに

はじめに

この章では、Agilent VEEとその主要機能を紹介します。また、VEEのインストール方法、習得方法、VEEサポートの利用方法についても解説します。

Agilent VEEの概要

Agilent VEEは、テスト/計測アプリケーション、およびオペレータ・インタフェースを備えたプログラムの構築用に最適化されたグラフィカル・プログラミング言語です。Agilent VEE製品のこのバージョン(バージョン7.0)は、複雑なテスト/計測システムを構築する必要がある技術者グループを対象としています。

テスト開発にAgilent VEEを使用する利点

VEEは、テスト開発ツールとして次のような特長を備えています。

- 生産性を大幅に高めます。顧客からの報告によると、プログラム開発時間を最大80%まで削減できます。
- VEEは、機能テスト、設計の検証、校正、データの取込みと制御など、幅広いアプリケーションで使用できます。
- GPIB、VXI、シリアル、GPIO、PCプラグイン・カード、LAN計測器を制御する計測器I/Oがより柔軟になりました。パネル・ドライバ、VXI*plug & play*ドライバ、標準インタフェース経由の「ダイレクト出入力」、さまざまなベンダからインポートされたライブラリを使用できます。
- ActiveXオートメーションおよびActiveXコントロールを使用して、PC上でMS Word、Excel、Accessなどほかのアプリケーションを制御できます。これらのアプリケーションを利用して、レポートを作成したり、データを表示して解析したり、テスト結果を将来使用するためにデータベース化できます。
- 処理能力が増えたため、より大きなプログラムを簡単に構築できるほか、計測器の管理がより柔軟になりました。VEEが採用するコンパイラは、大規模で複雑なプログラムに適したプロフェッショナル開発環境と高度な計測器管理能力を提供します。
- Visual Studio .NETアセンブルをサポートするので、Visual Studio .NETを真にサポートするテキスト形式言語もすべてサポートされます。
- C/C++、Visual Basic、Pascal、Fortranなどのその他のテキスト形式言語もサポートするので、これらのテキスト形式言語で作成したプログラムが利用できます。

VEEのプログラムは、メニューからオブジェクトを選択し、それらを接続することによって作成されます。VEEで作成されたプログラムはデータ・フロー・ダイアグラムに似ており、従来のコード行に比べて使いやすく、理解しやすくなっています。VEEを使用すると、編集-コンパイル-リンク-実行という面倒な繰返しを行う必要がありません。

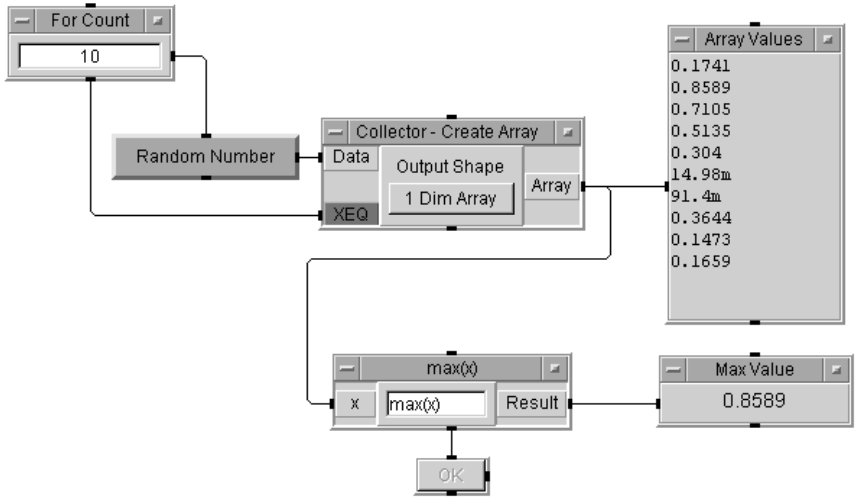
次の2つの図は、簡単な関数をテキスト形式言語(ANSI C)でプログラムした場合と、VEEでプログラムした場合を比較したものです。どちらの場合も、関数は10個のランダムな数値から成る配列を作成し、最大値を見つけ、配列と最大値を表示します。

図I-1に、ANSI C言語で書かれた「Random」というプログラムを示します。

```
/* 配列の最大要素を見つけるプログラム */
#include <math.h>
main( )
{
double num[10],max;
int i;
for (i=0;i<10;i++){
num[i]=(double) rand( )/pow(2.0,15.0);
printf("%f/n",num[i];
}
max=num[0];
for {i=1;i<10;i++){
if (num[i]>max)max=num[i];
}
printf("/nmax; %f/n",max);
}
```

図I-1. ANSI Cによる「Random」プログラム

図I-2に、VEEで作成した同じプログラムを示します。



図I-2. VEEによる同じ「Random」プログラム

VEEでは、プログラムは**オブジェクト**と呼ばれるプログラム要素で構築されます。オブジェクトはプログラムの構成単位であり、I/O操作、解析、表示などさまざまな機能を実行します。図I-2のようにオブジェクトとその接続をすべて表示したものを**詳細ビュー**と呼びます。詳細ビューは、テキスト形式言語におけるソース・コードに似ています。

VEEでは、データが1つのオブジェクトから次のオブジェクトへ整合性を保ちながら移動します。データは、オブジェクトの左側から入力され、右側から出力されます。またオブジェクトの上下には操作シーケンス・ピンがあります。

オブジェクトが結合されて1つのプログラムとなります。プログラムは左から右へ実行されます。図I-2で示した「Random」プログラムでは、ランダムな数値が「**Collector-Create Array**」オブジェクトに10回追加され、配列が作成されます。次にプログラムは、配列中の最大値を見つけて、「**Max Value**」と「**Array Values**」を表示します。

モジュール・プログラミングの手法を用いるVEEを使用すると、計測器を制御するプログラムを作成したり、カスタマイズしたデータを表示したり、オペレータ・インタフェースを開発する時間が短縮されます。このテスト開発方式は、従来の手法に比べて生産性を飛躍的に高めます。

注 記

図I-2には、詳細が表示されたオブジェクトと名前だけが表示されているオブジェクトがあります。詳細が表示されたオブジェクトは、オープン・ビューで表示されます。オープン・ビューでは、オブジェクトの詳細を表示できます。スペースを節約してプログラムの実行速度を上げるには、オブジェクトをアイコン化して、名前だけが表示されるように縮小できます。

たとえば図I-2では、Random Numberのラベルが付いたオブジェクトがアイコンで表示されています。Create Arrayのラベルが付いたオブジェクトはオープン・ビューで表示されています。オープン・ビューでは、オブジェクトがより大きく、詳細に表示されます。

Agilent VEEにおけるオペレータ・インタフェースの作成

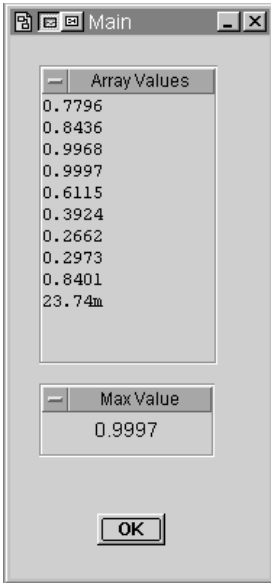
VEEでプログラムを記述するもう1つの利点は、オペレータ・インタフェースを簡単に作成できることです。

図I-2の「Random」プログラムを使用する場合、オペレータに表示する必要があるオブジェクトを選択して**パネル・ビュー**に入れます。パネル・ビューには、オペレータがプログラムを実行するために必要なオブジェクトと、結果として得られたデータだけが表示されます。図I-3 は、図I-2の「Random」プログラムのパネル・ビューを示しています。

注 記

プログラムとそのオペレータ・インタフェースは、同じVEEプログラムを異なる方法で表示したものです。VEEウィンドウのタイトル・バーで詳細ビュー・ボタンとパネル・ビュー・ボタンをクリックすることにより、ビューを切り替えることができます。プログラム(詳細ビュー)を編集したり更新すると、オペレータ・インタフェース(パネル・ビュー)も自動的に編集/更新されます。オペレータ・インタフェースを不必要な変更から保護することもできます。

オペレータ・インタフェースの作成方法についての詳細は、90ページの「例題2-4: パネル・ビュー (オペレータ・インタフェース)の作成方法」を参照してください。



図I-3. VEEプログラムのパネル・ビュー (またはオペレータ・インタフェース)

VEEを使用すると、テキスト形式言語では何日もかかるような作業を数分で実行できる場合があります。

- プログラムに対するカラフルで直観的なフロント・エンドを作成できます。
- キーボードとマウス、またはキーボード入力だけを使用するオペレータ・インタフェースを作成できます。
- 入力方法とデータ表示機能を幅広い組合わせの中から選択できます。
- ポップアップ・パネルを使ってフォーカスを作成し、画面スペースを節約できます。
- プログラムを不必要な変更から保護できます。
- ラベルを使用したり、色とフォントを選択したり、さまざまな形式の警告音、メモ帳、ボタン、スイッチを追加できます。
- 独自または標準の既存ActiveXコントロールを使用して、データ入力やデータの表示を行うことができます。

既存のテスト・プログラムのAgilent VEEでの利用

サポートするすべてのオペレーティング・システム上で、VEEは、市販アプリケーションだけでなく通常のテスト・プログラムも結合できる仕組みを提供します。たとえば、VEEを使用して、C、C++、Visual Basic、Fortran、Pascal (またはそのオペレーティング・システムで使用できる任意のコンパイルまたはインタープリタ言語)で書かれた既存のテストをシーケンス実行できます。Visual Studio .NETを真にサポートする言語もすべて使用できます。

VEEには、データベースやスプレッドシートなどの市販アプリケーションとデータを共有するための数多くのプロセス間通信機能も備わっています。VEEは、ActiveX オートメーション、ActiveX コントロール、DLLとの標準接続もサポートします。

Agilent VEEでの計測器の制御

VEEは、計測器を制御し、計測器と通信するためのオプションを数多く提供します。

- さまざまなベンダから提供されている450以上の計測器用パネル・ドライバ(計測器ドライバ)のほか、Windows 98、Windows 2000、Windows NT 4.0、Windows XPで使用可能なさまざまなベンダのVXI*plug & play*ドライバをすべて使用できます。
- VEEのDirect I/Oを使用すると、計測器のコマンド文字列をGPIB(IEEE - 488)、GPIO、RS 232、VXI、LANベース計測器などの標準インタフェース経由で送信して、リモート・テストを行うことができます。
- 標準ODASドライバ、あるいはボードに付属するDLL(Dynamic Link Library)を供給するメーカのPCプラグイン・ボードを制御できます。
- 組込みPCまたはワークステーションを使ってVXIのバックプレーン制御を直接使用できます。
- 簡単で組織化された計測器管理機能により、多様な計測器を制御できます。

Agilent VEEによるテスト機能の強化

VEE製品には、次の機能と利点があります。

- グラフィカル・プログラミングによる開発/保守時間の削減
- C、C++、Visual Basic、Pascal、Fortranなどの従来の言語との統合
- 使いやすく柔軟なオペレータ・インタフェース機能
- 一般的なテスト・プラットフォームのほとんどをサポート
- ActiveXオートメーションおよびActiveXコントロールの使用
- Agilent Technologiesのさまざまなサポートオプション
- 簡単で強力な文書ツール
- テスト・データをレポート作成のために標準的なスプレッドシートやワープロに簡単に移植可能
- リレーショナル・データベースや統計解析パッケージなどのほかのアプリケーションとリンクするためのプロセス間通信ツール(バージョン6.0以上)
- 大規模で複雑なプログラムの開発と保守を効率よく行うためのデバッグ・ツール(バージョン6.0以上)
- 製品に付属する強力なテスト実行ツール(バージョン6.0以上)
- VEEのWebモニタ機能によるリモート・テスト機能(バージョン6.0以上)
- 自作プログラムを無制限に配布可能((バージョン6.0以上)
- 低価格のサイト・ライセンス(バージョン6.0以上)

Agilent VEEのインストールと習得

このセクションでは、VEEのインストールと習得に関するガイドラインを提供します。VEEのインストール方法、VEEの習得方法、VEEの使用方法、VEEサポートの利用方法などについて説明します。

Agilent VEEおよびI/Oライブラリのインストール

VEEの評価バージョンをダウンロードするか、オーダできます。評価バージョンは、フル機能の製品バージョンです。評価バージョンをインストールすると、製品キーの入力を求められます。“eval”を、引用符なしで入力してください。

製品キーを購入すると、評価バージョンをフルにアクティブにすることができます。メールで製品キーを受信したら、メインメニューの[ヘルプ] ⇒ [Product Key]からキーを入力してください。完全な製品もオーダまたはダウンロードできます。製品キー認証は、製品に付属しているか、WWWからダウンロードした場合には別途送信されます。

VEEは、オペレーティング・システムWindows 98、Windows ME、Windows NT(SP6a)、Windows 2000 SP4、Windows XP SP1で動作します。

CD-ROMからVEEをインストールするには、管理者権限を持つユーザとしてログインします。CD-ROMをCD-ROMドライブに入れます。インストールが自動的に開始します。実行しない場合は、[スタート]メニューに進み、[ファイル名を指定して実行]を選択します。以下を入力します。

[CD-ROMドライブ名]:\cdsetup.exe

WWWダウンロードからVEEをインストールするには VEE自己解凍実行可能ファイルをダウンロードします。ダウンロードした自己解凍ファイルは、選択したダウンロード・ディレクトリに格納されます。自己解凍実行可能ファイルを実行して、VEEインストールファイルを解凍します。ファイルは、選択したディレクトリに解凍されます。このディレクトリから、cdsetup.exeを実行します。

セキュリティニーズを満足するため、PCの全ユーザ用にインストールするか、1人のユーザ用にインストールするかを選択するよう求められます。セキュリティニーズに合ったオプションを選択します。

I/Oライブラリの詳細については、付属のインストールマニュアルを参照してください。I/Oライブラリは、VEEと計測器との通信に使用されます。

オンラインヘルプの「Installing and Distributing VEE Pro RunTime」で、VEE Pro RunTime版のインストール方法と配布方法を説明します。RunTime版は、VEEソフトウェアがインストールされていないPC上でVEEプログラムを実行するときに使用します。RunTime環境については、オンライン・**ヘルプ**を参照してください。**[Help]** ⇒ **[Contents and Index]**をクリックして、**[Installing and Distributing Agilent VEE Pro RunTime]**を選択します。必要に応じて情報を印刷できます。

Agilent VEEの習得

VEEの使用方法についてさらに学習するには、VEEのマルチメディア・チュートリアル、オンライン・ヘルプ、または本書などのマニュアルを参照してください。VEE講習に参加する方法もあります。

- **VEEマルチメディア・チュートリアル**: VEEの**[Help]** ⇒ **[Welcome]**メニューにある、VEEの重要な概念を説明するビデオ・プレゼンテーションです。VEEメニューの使用方法、オブジェクトの編集方法、プログラムの実行方法を説明します。それぞれのプレゼンテーションは終了までに3、4分かかり、何回でも再生できます。チュートリアルを一時停止して、VEEを実行して学んだことを試した後、またチュートリアルを再開できます。
- **VEEオンライン・ヘルプ**: VEEの新機能については、**[Help]** ⇒ **[Contents and Index]** を選択し、**[What's New in Agilent VEE]**を選択してください。**[Help]** ⇒ **[Welcome]** ⇒ **[Introduction]**を選択すると、VEE製品の概要を読むことができます。

  オンライン・ヘルプには、ほかにもさまざまな機能があります。詳細は、97ページの「オンライン・ヘルプの使用方法」を参照してください。
- VEEマニュアル: VEEのマニュアル・セットには、本書のほか、『VEE Pro ユーザーズ・ガイド』と『VEE Pro Advanced Techniques』が含まれます。
- **Agilent VEE講習**: VEE講習についての情報は、Webサイト`http://www.agilent.com/comms/education`を参照してください。

注 記

このマニュアルに記載されている練習問題やプログラミング例で使用したVEEプログラムの多くは、VEEに付属しています。**[Help]** ⇒ **[Open Example...]** ⇒ **[Manual]** ⇒ **[UsersGuide]**を選択してください。

無料評価ソフトウェアの注文

無料の評価ソフトウェアをCDで入手できます。または、VEE Webサイトからダウンロードできます。Agilent Technologies VEE評価キットCDを注文するには、以下のAgilent Technologies Webサイトを参照してください。

http://www.agilent.com/find/products

MATLAB Scriptの概要

MATLAB® Scriptは、The MathWorks社製の高機能標準MATLABのサブセットです。MATLAB Scriptにより、高度な数式処理、データ解析、科学的/工学的グラフィックなどのMATLABの中核機能に直接アクセスすることができます。MATLAB Scriptオブジェクトは、どのAgilent VEEプログラムにも簡単に挿入できます。

MATLAB Scriptには、次のような数多くの機能があります。

- データの解析と視覚化
- 次の数値計算
  - 線形代数および行列演算
  - フーリエ解析および統計解析
  - 微分方程式の計算
  - 三角関数および基本演算
- 次の工学的、科学的グラフィック
  - 三角データ、グリッド・データなどの二次元および三次元表示
  - スカラおよびベクトル・データの量の視覚化
  - 矢印(quiver)、リボン(ribbon)、散布図(scatter)、棒グラフ(bar)、円グラフ(pie)、離散グラフ(stem)のプロット

Signal Processing Toolbox

VEE用のMATLAB Scriptには、フィルタ設計とスペクトル解析技術に基づいて構築されたMATLAB Signal Processing Toolboxのサブセットも含まれます。次の機能があります。

- 信号および線形システム・モデル
- アナログ・フィルタの設計
- FIRおよびIIRディジタル・フィルタの設計、解析、実装
- FFT、DCTなどの変換処理
- スペクトル推定と統計的信号処理
- 時系列パラメータ・モデリング
- 波形の生成

高機能MATLABについて

MATLABは、数値計算、高度なグラフィックおよび視覚化、高水準プログラミング言語を結合した統合型テクニカル・コンピューティング環境です。MATLABには、次のような数多くの機能があります。

- データの解析と視覚化
- 数値および記号の計算
- 工学的、科学的グラフィック
- モデリング、シミュレーションおよび試作
- プログラミング、アプリケーション開発、GUI設計

MATLABは、信号処理、画像処理、制御システム設計、金融解析、医学研究など、さまざまな分野のアプリケーションで使用されます。オープン・アーキテクチャにより、MATLABと関連製品を使用すると、データの調査やカスタム・ツールの作成を簡単に行うことができ、解析時間が短縮されて競争上優位に立つことができます。

VEEユーザは、データ解析、視覚化、モデリングなどを必要とするアプリケーションに、MATLABとSignal Processing Toolboxの全機能を組み込むことができます。これらの製品をフル・バージョンにアップグレードすれば、ユーザ定義関数(Mファイル)の作成、およびMATLABコマンド・ウィンドウ、MATLABエディタ/デバッガ、Signal Processing GUIへのアクセスなどの追加のMATLAB機能をVEEアプリケーションで幅広く使用できます。

注 記

VEEプログラムにおけるMATLAB Scriptオブジェクトの使用方法については、第4章「テスト・データの解析と表示」の188ページの「Agilent VEEにおけるMATLAB Scriptの使用」を参照してください。

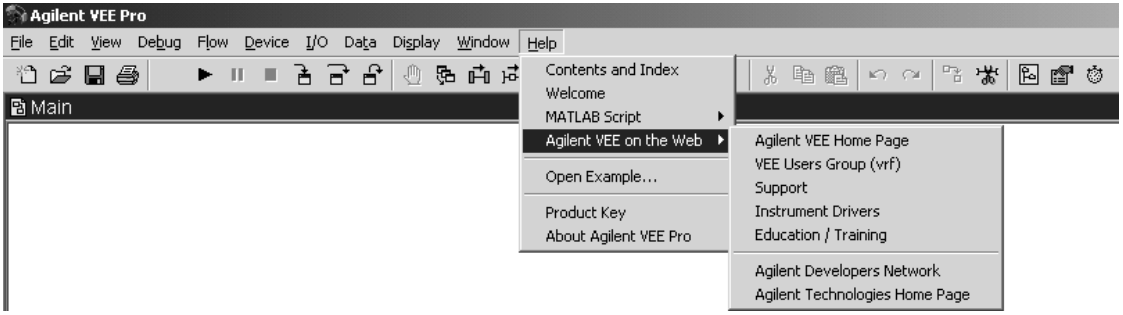
Agilent VEEサポートの利用

VEEを使い始めるときに、Webサイトまたは電話によるサポートを利用できます。

World Wide Webで情報を得る

VEE Webサイトでは、アプリケーション・ノート、ユーザのためのヒント、技術情報、PCプラグイン・ボード・ベンダなどのVEEパートナー情報など、さまざまな情報が提供されています。

- VEEのメインWebサイト:
  `http://www.agilent.com/find/vee:jp`
- **最新のサポート情報**: ネットワークに接続した状態で、VEEで**[Help]** ⇒ **[Agilent VEE on the Web]** をクリックします。図I-4「VEEの[Help]メニューから製品サポートを利用」に、VEEでの[Support]メニューの選択方法を示します。`http://www.agilent.com/find/vee`



図I-4. VEEの[Help]メニューから製品サポートを利用

- **無料のスタートアップ・サポート**については、オンライン・ヘルプの電話サポート情報をご覧ください。

MATLABの追加情報源

MATLAB Scriptオブジェクトの使用法の詳細については、「MATLAB SCRIPT Help Desk」を参照してください。VEEで **[Help]** ⇒ **[MATLAB Script]** ⇒ **[Help Desk]**を選択すると、Webブラウザに**MATLAB Help Desk** が表示されます。

MATLAB、MATLAB Toolboxes、およびThe MathWorks社のその他の製品についての詳細は、`www.mathworks.com`を参照するか、または米国(508) 647-7000に電話してください。

ほかにも次の情報源があります。

- MATLABの全ドキュメント: `www.mathworks.com/access/helpdesk/help/helpdesk.shtml`
- MATLAB製品情報: `www.mathworks.com/products`
- MATLAB技術サポート: `www.mathworks.com/support`
- MathWorksオンラインショップ: `www.mathworks.com/store`
- MathWorksホームページ: `www.mathworks.com`
- Usenetニュースグループ: `comp.soft-sys.matlab`ニュースグループには、MATLABを使用する専門家と学生のためのフォーラムがあります。MATLABと関連製品についての質問やコメントが掲載されています。

1
Agilent VEE開発環境の使用方法

Agilent VEE開発環境の使用方法

この章の内容

- 対応システム
- ヘルプ・システムの使用方法
- VEEの起動方法.
- VEEウィンドウについて
- オブジェクトの扱い方
- ワークスペースの管理方法
- メニュー項目の選択方法
- VEEオブジェクトのピンと端子
- オブジェクトの接続とプログラム作成
- プログラムの作成、実行、印刷、保存、オープン
- VEEプログラムの動作の仕方

平均所要時間: 1.5時間

概要

この章では、VEEの起動方法、メニューの使用方法、オブジェクトの扱い方を説明します。またVEEにおけるピンと端子について解説します。オブジェクトを接続して簡単なVEEプログラムを作成し、VEEプログラムの動作の仕方についても学習します。

Agilent VEEの対話型操作

このセクションでは、VEEのグラフィカル・プログラミング言語の使用方法を説明します。また、対応システム、マウスとメニューの使用方法、ヘルプの利用方法、VEEの起動方法、VEEウィンドウでの操作方法についても説明します。

対応システム

本バージョンのVEEは、次のオペレーティング・システムに対応しています。

- PC上のWindows 98、Windows 2000、Windows NT 4.0、およびWindows XP

マウスとメニュー

プルダウン・メニュー、ツールバー、マウスとキーボードで制御するダイアログ・ボックスなど、マウスとメニューを駆使したコンピュータのインタフェースについては、よくご存知と思います。VEEでは、このコンピュータのインタフェースを使用します。次は、マウスを使ってメニュー、アイコン、ボタン、オブジェクトを操作するときの一般的な手法です。

- アイテムを「クリック」するには、マウス・ポインタを目的のアイテムに合わせて、マウスの**左**ボタンをすばやく押して離します。
- アイテムを「ダブルクリック」するには、マウス・ポインタを目的のアイテムに合わせて、マウスの**左**ボタンを2回、すばやく続けてクリックします。
- アイテムを「ドラッグ」するには、マウス・ポインタを目的のアイテムに合わせて、マウスの**左ボタンを押したまま**、アイテムを適切な位置まで移動します。次にマウスのボタンを離します。

注 記　**マウスの右ボタンはあまり使用しません。マウスの右ボタンをクリックする必要がある場合は、そのことを明記します。また、中央ボタンを備えたマウスもありますが、VEEでは中央ボタンを使用しません。**

Agilent VEEの起動方法

[スタート] ⇒ [プログラム] ⇒ [Agilent VEE Pro] をクリックします。

Agilent VEEウィンドウ

VEEをインストールして起動すると、図1のようなVEEウィンドウが表示されます。

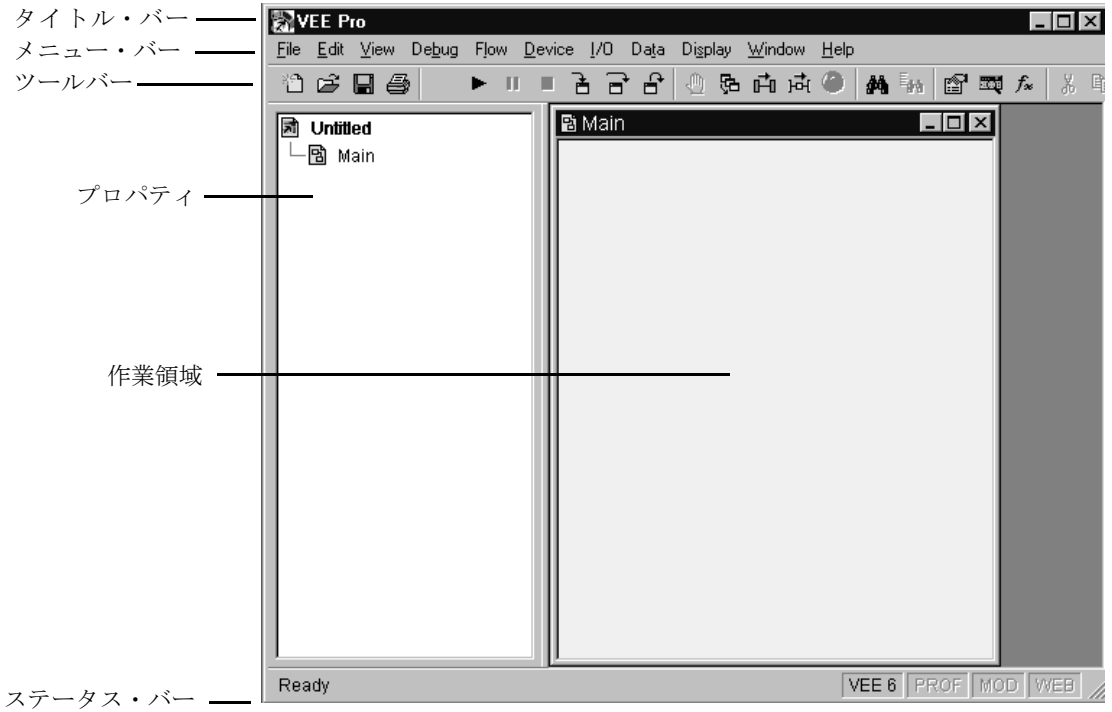


図1　　VEE開発環境

次に、VEEウィンドウの各部について説明します。

表1　次に、VEEウィンドウの各部について説明します。

画面の領域名	説明
タイトル・バー	ウィンドウの最上部の行には、VEEアイコン、ウィンドウ名、[最小化]ボタン、[最大化]ボタン、[閉じる]ボタンが表示されます。タイトル・バーをドラッグすると、ウィンドウが移動します。VEEアイコンをクリックすると、ウィンドウ・メニューが表示されます。
メニュー・バー	2行目にはメニュー項目があり、それぞれVEEのコマンドまたはオブジェクトを提供します。
ツールバー	3行目にはアイコンまたはボタンが並びます。これらのボタンから、使用頻度が高いメニュー・コマンドに直接アクセスできます(ショートカット)。ボタン上にマウス・ポインタを置くと、そのボタンの機能が表示されます。
作業領域	メイン・ウィンドウ、UserObject編集ウィンドウ、UserFunction編集ウィンドウなど、オブジェクトを配置して結線するプログラミング(編集)ウィンドウの領域です。
プログラム・エクスプローラ	VEEウィンドウの左側の領域には、VEEプログラムの構造が表示されます。上隅には、現在のプログラム名が**myprog.vee**または**Untitled**などと表示されます。 プログラム・エクスプローラを使ってプログラミング・ウィンドウ間を移動できます。プログラム・エクスプローラのサイズを変更するには、右境界上でノーマル・ポインタが垂直分割線に変わるまでポインタを動かし、クリックして移動します。
メイン・ウィンドウ	VEEプログラムを開発して編集する作業領域を含むウィンドウです。UserObject編集ウィンドウなどのほかのプログラミング/編集ウィンドウも、ここに表示されます。
ステータス・バー	一番下の行には、VEEの状態に関するメッセージが表示されます。また、右隅には4つの状態インジケータが表示されます。インジケータは左から右に次の事柄を示しています。 実行モード プロファイラのステート プログラムが変更されたときにMODと表示 Webサーバがオン

ヘルプの利用

VEEは、VEE環境全体と、個々のオブジェクトおよびトピックについてのオンライン・ヘルプ・システムを提供します。また、コンピュータとオペレーティング・システムに付属するマニュアルを利用することもできます。PCのオンライン・ヘルプには、次の情報が記載されています。

- メニュー・バー上のコマンドの選択方法
- メニュー項目の選択方法と閉じ方
- ツールバーの使用方法
- タイトル・バーとステータス・バーの理解
- アイコンとボタンのクリック方法
- ダイアログ・ボックスの扱い方
- さまざまなウィンドウの扱い方
- オンライン・ヘルプの使用方法

まず**[Help]** ⇒ **[Welcome]**を選択して、[Welcome]画面から開始することをお勧めします。この画面からVEEマルチメディア・チュートリアルにアクセスできます。[Welcome]画面を図2に示します。

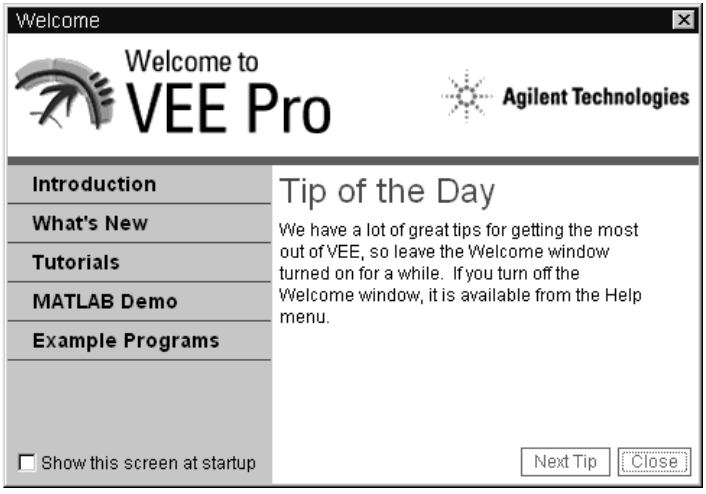


図2　　ヘルプにおけるVEEの[Welcome]画面

VEEのオンライン・ヘルプは、お使いのオペレーティング・システムに合わせて設計されています。**[Help]**をクリックすると、図3のようなメニューが表示されます。[Help]メニューでは、目次と索引、チュートリアルがある[Welcome]メニューのほか、計測器ドライバ、Webサイト情報、サンプル、バージョン情報などが提供されます。

この入門トレーニングは、VEEのマニュアルを使用しなくても完了できます。しかし、特定の機能や概念についてより詳しく知りたい場合は、製品のマニュアルを参照してください。特定のVEEトピックを検索するには、ヘルプ・システムを使用します。ヘルプ・システムでは、関連するトピックに「ジャンプ」することができます。

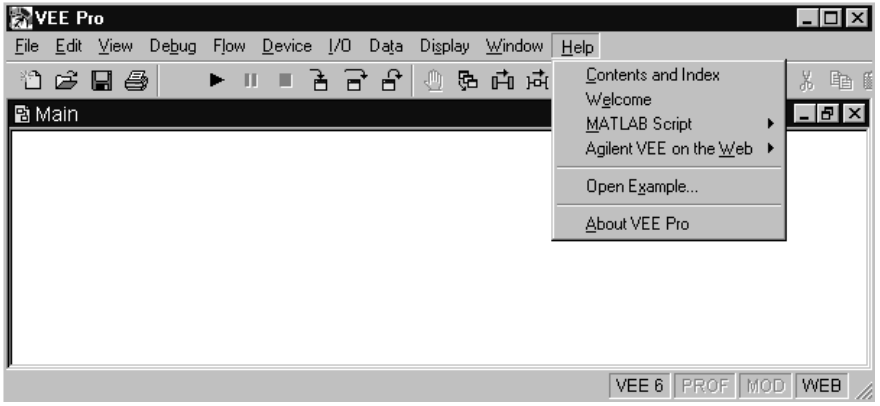


図3　　[Help]メニューの使用

[Contents and Index]を選択すると、図4のようなVEEヘルプが表示されます。

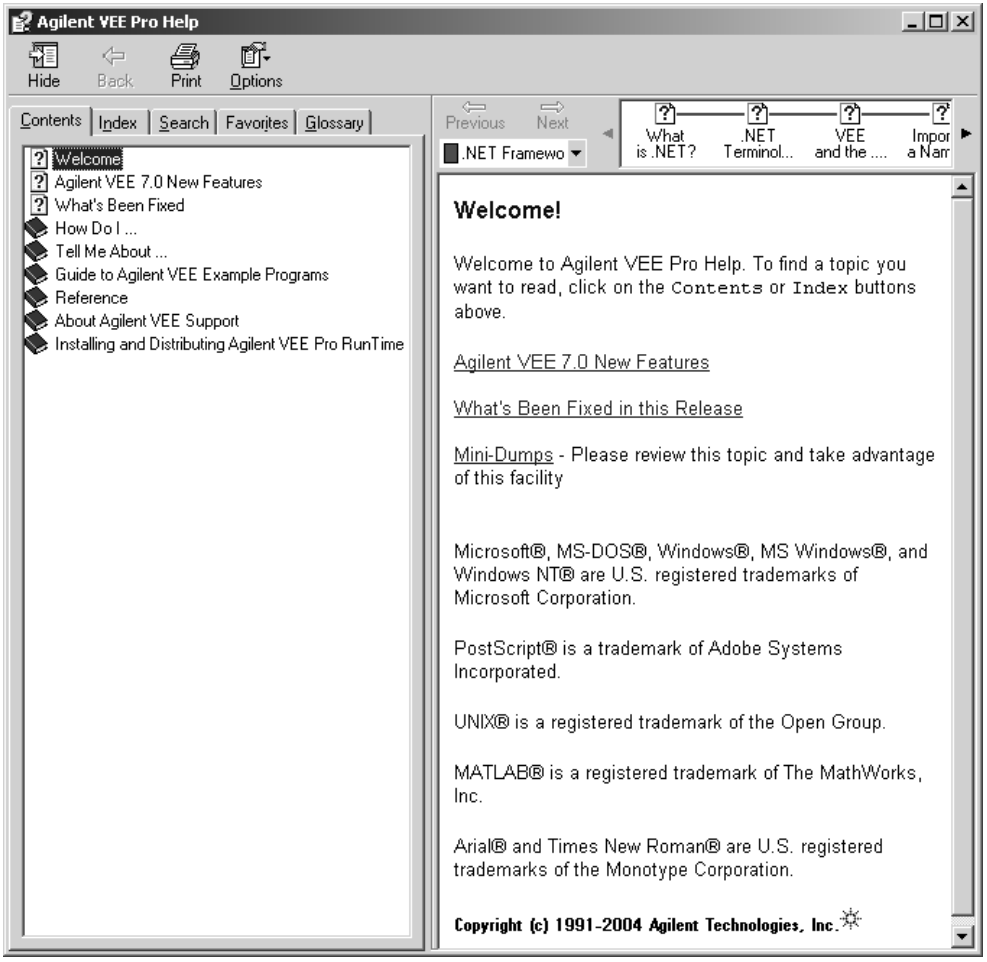


図4　　ヘルプの[目次]タブ

ヘルプの**[目次]**タブには、次のトピックがあります。

表2

VEEの新機能	説明
How Do I...	基本的な操作方法を説明します。

表2

VEEの新機能	説明
Tell Me About...	VEEの概念を説明します。
Guide to Agilent VEE Example Programs	VEEに搭載されているサンプル・プログラムの要約です。
Reference	すべての関数およびオブジェクトに関するリファレンス情報を提供します。
About Agilent VEE Support	VEEサポートの利用方法についての情報を提供します。
Installing and Distributing Agilent VEE Pro Runtime	VEE Pro RunTime環境の配布方法を説明します。

注 記

選択したオブジェクトやダイアログ・ボックスについてのヘルプを簡単に表示するには、キーボードの[F1]キーを押します。また、オブジェクト・メニューの[Help]をクリックして、そのオブジェクトについての特定情報を得ることもできます。

プログラム開発時にオンライン・ヘルプの特定の機能を使用する方法については、97ページの「ヘルプ機能の使用方法」を参照してください。

オブジェクトの扱い方

VEEプログラムは、接続されたオブジェクトで構成されます。プログラムを作成するには、Flow、Data、Display などのVEEメニューからオブジェクトを選択します。オブジェクトのピンに付いているラインを介してオブジェクトを接続します。ピンについての詳細は、46ページの「ピンと端子について」を参照してください。接続されたオブジェクトのグループでプログラムが作成されます。

このセクションでは、プログラム内のオブジェクトの選択方法と使用方法を説明します。

1. VEEを起動します。Windowsで、**[スタート]** ⇒ **[プログラム]** ⇒ **[Agilent VEE Pro]**をクリックします。
2. このセクションの説明に従ってオブジェクトを操作します。

注 記

次の練習では、VEEソフトウェアが実行されていることが前提となります。VEEの起動方法については、上の説明、または23ページの「Agilent VEEの起動方法」を参照してください。

作業領域へのオブジェクトの追加

適切なプルダウン・メニューを開き、目的のオブジェクトをクリックします。次に、オブジェクトを作業領域内の適切な位置までドラッグし、クリックします。

1. 輪郭線が消え、オブジェクトが表示されます。たとえば、Function Generatorオブジェクトを作業領域に追加するには、図5のように、メニュー・バーから**[Device]** ⇒ **[Virtual Source]** ⇒ **[Function Generator]**を選択します。

注 記

[Virtual Source]メニューの右側にある矢印は、サブメニューがあることを示します。メニュー項目名の後に続く3つのピリオドは、そのメニューを選択するとダイアログ・ボックスが表示されることを示します。たとえば、[File] ⇒ **[Save As...]を選択すると、ダイアログ・ボックスが表示されます。**

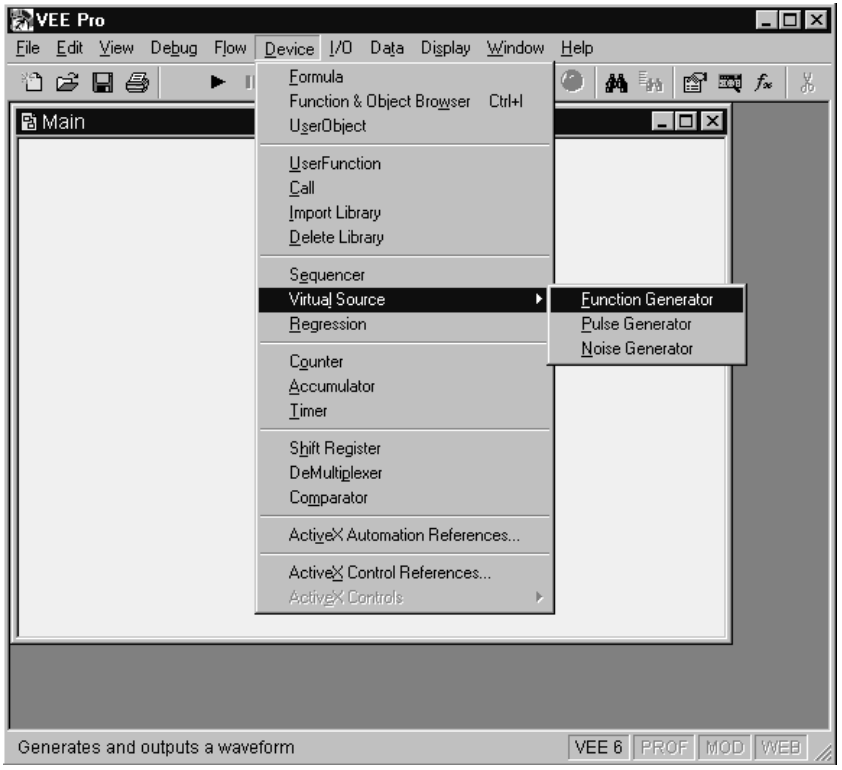


図5　　作業領域へのオブジェクトの追加

オブジェクトの輪郭線が作業領域に表示されます。

2 **Function Generator**を作業領域の中心まで移動し、クリックしてオブジェクトを配置します。図6のように**Function Generator**が表示されます。

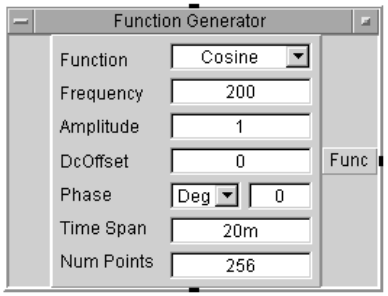


図6　　Function Generatorオブジェクトの追加

作業領域にオブジェクトを配置したら、オブジェクトのタイトル・バーをドラッグしてオブジェクトを移動できます。これは、ウィンドウを移動するときと同じ操作です。

注 記

以降の手順の説明には、簡潔な表現を使用します。たとえば、Function Generatorオブジェクトの選択は、次のように簡略化して表記します。**[Device]** ⇒ **[Virtual Source]** ⇒ **[Function Generator]**を選択します。

注 記

画面上により多くのスペースを空けるには、**[View]** ⇒ **[Program Explorer]**をクリックします。これにより、[Program Explorer]メニューが選択解除され、プログラム・エクスプローラが画面から消えます。メニュー項目の前にチェック・マークが付いている場合、そのメニュー項目は「選択」されています。

オブジェクトの表示の変更

VEEでは、図7のように、オブジェクトが「アイコン・ビュー」または「オープン・ビュー」のいずれかで表示されます。

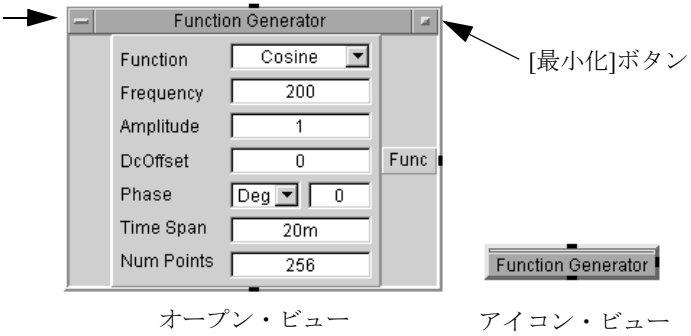


図7　　オープン・ビューおよびアイコン・ビューのオブジェクト

アイコン・ビューは作業領域内のスペースを節約し、プログラムを読みやすくします。オープン・ビューではオブジェクトの詳細を表示し、プロパティと設定を編集できます。

1. オープン・ビューからアイコン・ビューに切り替えるには、**[最小化]**ボタン(オブジェクトのタイトル・バーの右端にある四角形)をクリックします。
2. オープン・ビューに戻るには、アイコン・ビューのオブジェクトの任意の位置をダブルクリックします。

注 記

オブジェクト・メニューでも、[Minimize]と[Restore]を選択することができます。オブジェクト・メニューを表示するには、タイトル・バーの左端にあるオブジェクト・メニュー・ボタンをクリックするか、オブジェクト上でマウスの右ボタンをクリックします。

オブジェクトごとに構造やパーツは異なりますが、オープン・ビューではオブジェクトを編集でき、アイコン・ビューではスペースを節約できます。

オブジェクト・メニューの選択

VEEのオブジェクトにはそれぞれ**オブジェクト・メニュー**があり、オブジェクトに対して [Clone]、[Size]、[Move]、[Minimize] などの操作を実行できます。オブジェクトのほとんどが同様の属性を持っていますが、オブジェクトの機能によって違いがあります。オブジェクト・メニューから、各オブジェクトのオンライン・ヘルプを参照してください。

1. オブジェクト・メニューを選択するには、オブジェクト・メニュー・ボタンを**1度だけ**クリックします。オブジェクト・メニューはすべて同じ方法で開くことができます。オブジェクト・メニューが図8に示すように表示されます。オブジェクト・メニュー・ボタンをダブルクリックしないでください。ダブルクリックすると、オブジェクトが削除されます。
2. 次に、いずれかのオブジェクト・メニュー項目をクリックして、目的の操作を実行します。メニューを閉じるには、メニュー**以外**の何もない領域をクリックします。

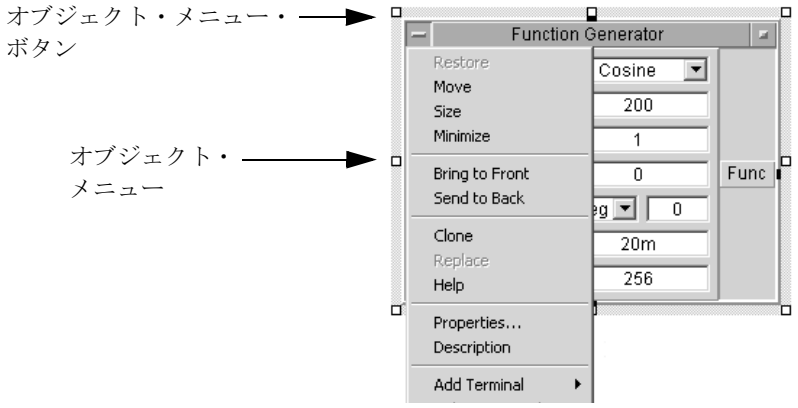


図8　　オブジェクト・メニューの選択

マウス・ポインタをオブジェクト上に置いてマウスの**右**ボタンをクリックしても、オブジェクト・メニューを選択できます。この操作は、オープン・ビューとアイコン・ビューのどちらでも実行できます。

オブジェクトの移動

1. Function Generatorオブジェクトを移動するには、オブジェクト・メニューで**[Move]**を選択します。次にマウスの左ボタンをクリックし、押したままにします。オブジェクトの輪郭線が表示されます。
2. マウスのボタンを押したまま、図9のように輪郭線を新しい位置まで移動します。マウスのボタンを離すと、オブジェクトが新しい位置に移動します。

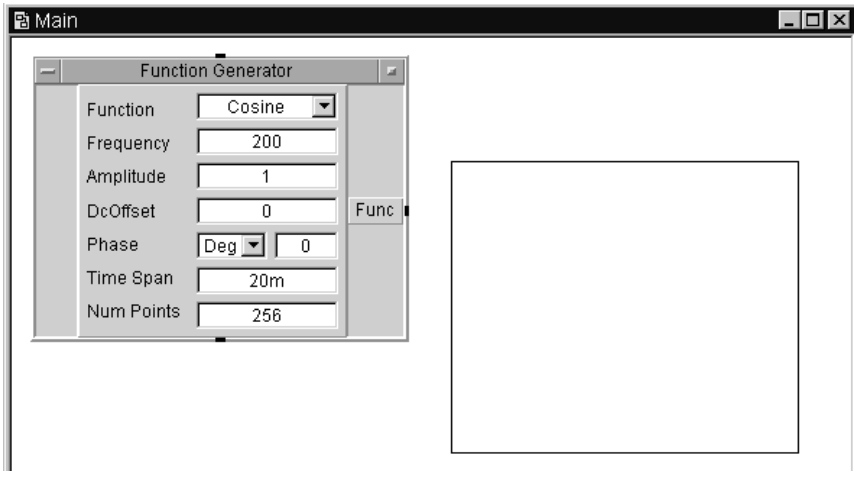


図9　　オブジェクトの移動

オブジェクトの移動は、次の方法でも行うことができます。

- オープン・ビューで表示されているオブジェクトのタイトル部分をクリックし、オブジェクトを新しい位置までドラッグします。
- オープン・ビューで表示されているオブジェクトの、ボタン、入力フィールド、ピン、端子、およびオブジェクトのサイズ変更に使用する四隅**以外**の場所をクリックして、オブジェクトを新しい位置までドラッグします。
- アイコン・ビューで表示されているオブジェクトの四隅から**離れた場所**をクリックして、アイコンを新しい位置までドラッグします。

注 記

VEEウィンドウ最下部のステータス・バーに表示される「オブジェクトの位置情報」は、輪郭線の左上隅の位置をワークスペースの左上隅を基準としたX座 標とY座標(ピクセル単位)で表したものです。オブジェクトの正確な位置を表示するには、オブジェクト上でマウスの左ボタンをクリックしてオブジェクトを選択し、マウスの左ボタンを押したままにします。ステータス・バーに位置が表示されます。オブジェクトを正確に配置する必要がある場合は、この情報を使用します。

オブジェクトの複製作成(クローン)

クローン操作により、サイズや名前の変更などオブジェクトに対して行った変更内容も含めて、オブジェクトの正確な複製を作成できます。クローンは、カット・アンド・ペースト操作のショートカットです。

1 オブジェクト・メニューを開き、[Clone]を選択します。複製されたオブジェクトの輪郭線が表示されます。

2 輪郭線を目的の位置まで移動し、クリックしてオブジェクトを配置します。複製されたオブジェクトが表示されますが、元のオブジェクトも残ります。図10では、Function Generatorを1つ複製した後で、さらにオブジェクト・メニューからクローンを実行するコマンドが選択されています。

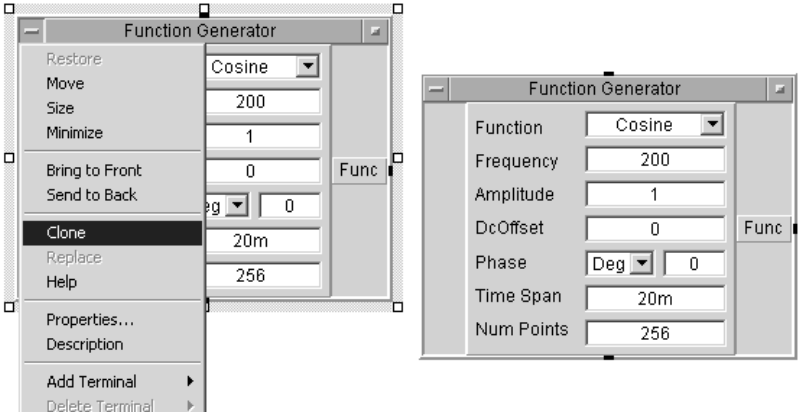


図10　　オブジェクトのクローン作成

オブジェクトのコピー

この操作は、オブジェクトをクリップボードにコピーします。コピーしたオブジェクトは、、またはMS PaintやMS Wordなどのほかのアプリケーションにコピーできます。

オブジェクトをクリックしてオブジェクトを強調表示します。次に**[Edit]** ⇒ **[Copy]**をクリックします。

-または-

オブジェクトをクリックしてオブジェクトを強調表示します。次に`Ctrl+C`キーを押します。

オブジェクトの削除(切取り)

作業領域からオブジェクトを削除する(または切り取る)には、削除するオブジェクトのオブジェクト・メニューを開き、**[Cut]**をクリックします。たとえば、Function Generatorのオブジェクト・メニューを開いて[Cut]をクリックします。オブジェクトは作業領域から消えますが、カット・バッファに保存されます。

オブジェクト・メニューを開き、[Clone]を選択します。

-または-

オブジェクトをクリックしてオブジェクトを選択し、`Ctrl+X`キーを押します。

-または-

マウス・カーソルをオブジェクト・メニュー・ボタンに合わせて、ダブルクリックします。

注 記

オブジェクト・メニュー・ボタンを*ダブルクリック*してオブジェクトを不用意に削除してしまうことがよくありますので、注意してください。誤ってオブジェクトを削除してしまった場合は、ツールバーの[Undo]を使用します。または[Edit] ⇒ [Undo]を選択します。これにより、オブジェクトと接続がすべて復元されます。

オブジェクトの貼付け

次の方法で、コピーまたは削除した(切り取った)オブジェクトを作業領域に貼り付けます。

オブジェクトをコピーまたは削除した後、**[Edit]** ⇒ **[Paste]**をクリックします。オブジェクトの輪郭線が表示されます。オブジェクトを配置した後、クリックしてオブジェクトを離します。

-または-

Ctrl+Vキーを押します。

オブジェクトのサイズの変更

マウス・ポインタをオブジェクトの四隅のいずれかに置き、サイズ変更矢印が表示されたら、クリックして目的の大きさになるまでドラッグします。マウスのボタンを離して、サイズを確定します。図11 は、サイズ変更矢印でオブジェクトのサイズを変更している例です。

-または-

オブジェクト・メニューを開き、**[Size]**をクリックします。マウス・ポインタが角かっこの形になります。この角かっこを、サイズ変更後のオブジェクトの右下隅が位置する場所に移動し、クリックしてサイズを変更します。

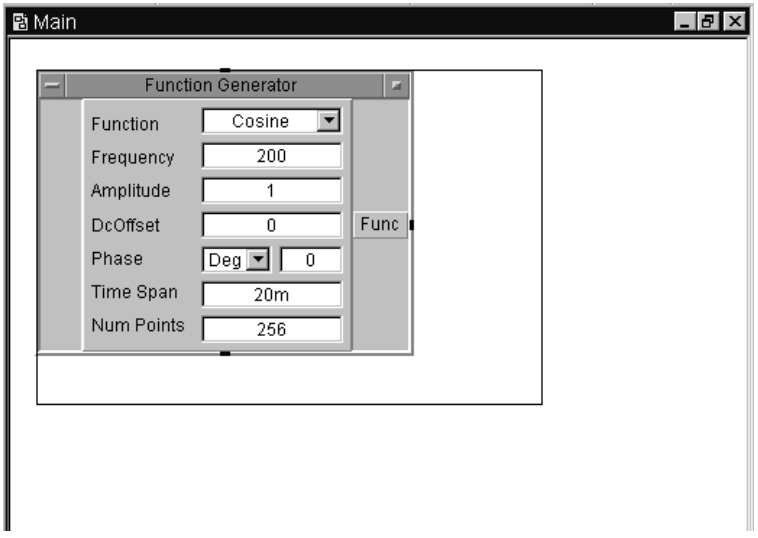


図11　　オブジェクトのサイズの変更

オブジェクトの名前(タイトル)の変更

1 オブジェクト・メニューを開き、**[Properties . . .]**を選択します。[Properties] ダイアログ・ボックスが表示され、図12のように現在のタイトルが強調表示されます。

2 タイトル領域に新しいタイトルが表示されます。オブジェクトを最小化すると、アイコンに新しいタイトルが表示されます。

-または-

オブジェクトのタイトル・バーをダブルクリックして[Properties]ダイアログ・ボックスを直接開きます。

注 記

標準的なキーボードとマウスによる編集テクニックを使用すると、時間を短縮できます。たとえば、[Properties]ダイアログ・ボックスの[Title]フィールドで、編集領域の一番左端でクリックすると、そこにカーソルが表示されます。これにより、既存のタイトルを削除せずに新しいテキストを追加できます。

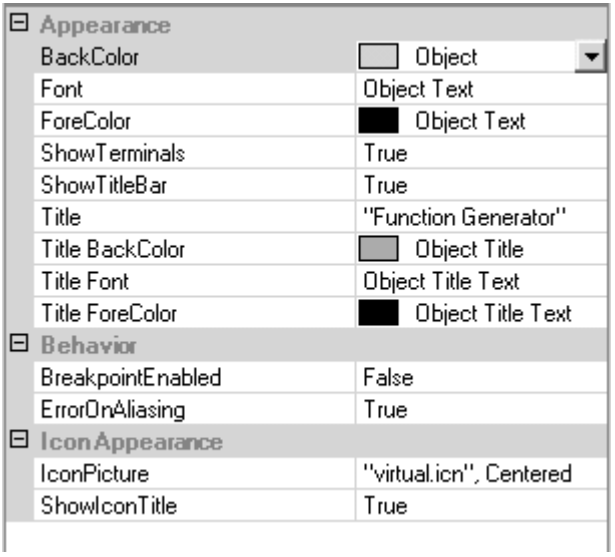


図12　オブジェクトのタイトルの変更

オブジェクトの選択/選択解除

1 オブジェクトを選択するには、そのオブジェクトをクリックします。オブジェクトに影が付きます。たとえば、図13ではFor Countオブジェクトが選択されています。

2 オブジェクトの選択を解除するには、マウス・ポインタを何もない領域に移動してクリックします。影が消えます。たとえば、図13ではFormulaオブジェクトは選択されていません。

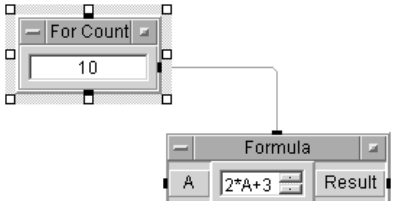


図13　選択されたオブジェクトと選択されていないオブジェクト

注 記　「選択する」という言葉は、メニュー項目の選択を指す場合にも使用されますが、前後の文脈から判断すれば、どちらの意味かを確実に判断できます。

複数オブジェクトの選択

オブジェクトをクリックして選択するとき、オブジェクトは1つしか選択できません。別のオブジェクトをクリックして選択すると、前のオブジェクトは選択が解除されて影が消えます。複数のオブジェクトを選択し、選択したすべてのオブジェクトにまとめて[Cut]などの操作を実行するには、次のようにします。

Ctrlキーを押したまま、選択するオブジェクトのそれぞれをクリックします。選択したいオブジェクトがすべて強調表示されたら、Ctrlキーを離します。

-または-

Ctrlキーを押します。次にクリック・アンド・ドラッグにより、選択するオブジェクトを四角形で囲みます。選択されたオブジェクトに影が付きます。

すべてのオブジェクトの選択/選択解除

1 すべてのオブジェクトを選択するには、**[Edit]** ⇒ **[Select All]**をクリックします。または`Ctrl+A`キーを押します。

2 すべてのオブジェクトの選択を解除するには、ウィンドウ内の何もない領域でクリックします。

複数オブジェクトのコピー

いずれかのオブジェクトの上にカーソルを置き、選択されている複数のオブジェクトをコピーします。`Ctrl`キーを押したままで、マウスの左ボタンを使って複数のオブジェクト(輪郭線)を目的の位置までドラッグします。目的の位置にそれぞれのオブジェクトの新しいインスタンスが表示されます。

-または-

[Edit] ⇒ **[Copy]**を使用して、選択した複数のオブジェクトをカット・バッファにコピーします。[Edit]メニューまたはツールバーにある**[Paste]**をクリックし、オブジェクト(輪郭線)を目的の位置まで移動し、マウスの左ボタンをクリックします。図14 はコピー中のオブジェクトを示しています。

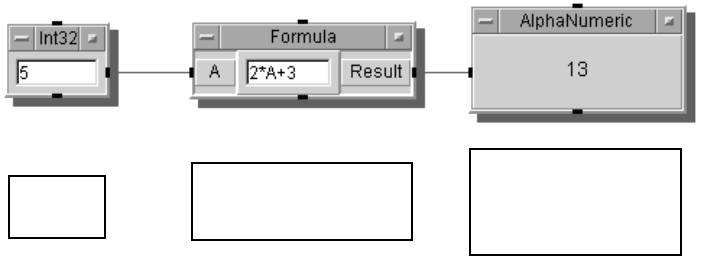


図14　　コピー中の複数オブジェクト

注 記

VEEでは、切り取ったり、コピーしたオブジェクトがクリップボード上にも配置されます。したがって、これらをWindowsのクリップボードをサポートするほかのWindowsアプリケーションに貼り付けることができます。

オブジェクトの編集

VEEでは、オブジェクトをいくつかの方法で編集でき、各編集メニューに表示される選択肢がそれぞれ異なります。編集メニューまたは編集アイコンを選択するには、次のようにします。

VEEのメニュー・バー上で**[Edit]**をクリックして[Edit]メニューを表示し、実行する操作を選択します。[Edit]メニューのコマンドは、すべてのVEEで共通です。

-または-

VEEのツールバー上のアイコンをクリックします。VEEのツールバーには、[Cut]、[Copy]、[Paste]などの使用頻度が高い編集コマンドのアイコンが表示されます。

-または-

オブジェクトのオブジェクト・メニュー・ボタンをクリックしてオブジェクト・メニューを開き、実行する操作を選択します。オブジェクト・メニューには、[Properties]メニューなど、メインの[Edit]メニューには表示されないオブジェクト固有の編集操作が含まれます。オブジェクト・メニューのコマンドは、オブジェクトの種類によって異なります。たとえば、**[Device]** ⇒ **[Formula]**で追加したFormulaオブジェクトと、**[I/O]** ⇒ **[To]** ⇒ **[File]**で追加したFileオブジェクトのオブジェクト・メニューを比較してみてください。この2つのメニューには、それぞれのオブジェクトに固有の、異なる選択肢が表示されます。

-または-

マウス・ポインタを作業領域の**何もない**スペースに置き、マウスの**右**ボタンをクリックします。[Edit]ポップアップ・メニューが表示されます。

注 記

アクティブでないメニュー項目には、アクティブな項目とは異なる影が付きます(淡色表示)。たとえば、[Edit]メニューの[Cut]、[Copy]、[Clone]操作は、オブジェクトが作業領域内で強調表示されないかぎり、アクティブなメニュー項目とは異なる影が付いて表示されます。

オブジェクト間のデータ・ラインの作成

1 一方のオブジェクトのデータ出力ピンの上またはそのすぐ外側でクリックし、次に、もう一方のオブジェクトのデータ入力ピンをクリックし

ます。図15はこの様子を示しています。ポインタを一方のピンからもう一方のピンへ移動すると、ポインタの後にラインが表示されます。

2 カーソルを離すと、は2つのオブジェクト間にラインを描きます。オブジェクトの位置を変更しても、オブジェクト間のラインは維持されます。

注 記　ピンについての詳細は、46ページの「ピンと端子について」を参照してください。

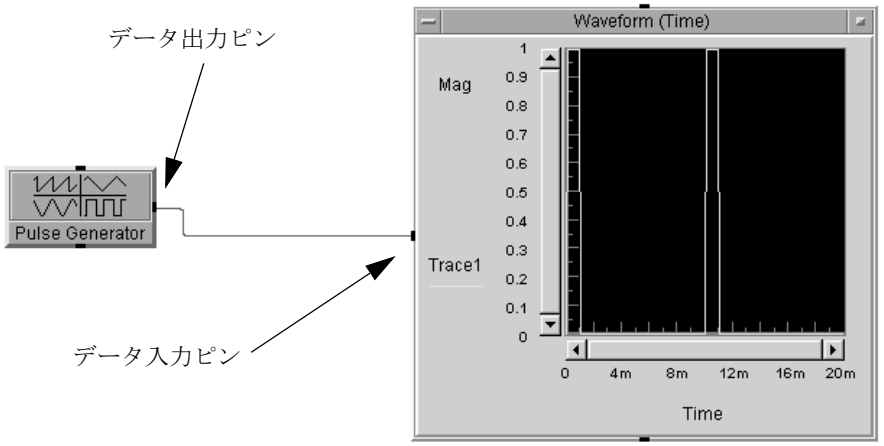


図15　オブジェクト間のデータ・ラインの作成

オブジェクト間のデータ・ラインの削除

`Shift+Ctrl`キーを押したままで、削除するラインをクリックします。

-または-

[Edit] ⇒ **[Delete Line]**を選択します。次に、削除するラインをクリックします。

作業領域全体の移動

作業領域内に少なくとも1つのアイコンがあることを確認します。マウス・ポインタを作業領域の背景に合わせて、マウスの左ボタンを押し§ボタンを押したままで作業領域を目的の方向へ移動します。

注 記　プログラムが作業領域より大きい場合は、図16のようにスクロール・バーが表示されます。

注 記　端子の近くでクリックすると、ライン(ワイヤ)が表示される場合があります。その場合は、ポインタを空き領域に移動してダブルクリックします。

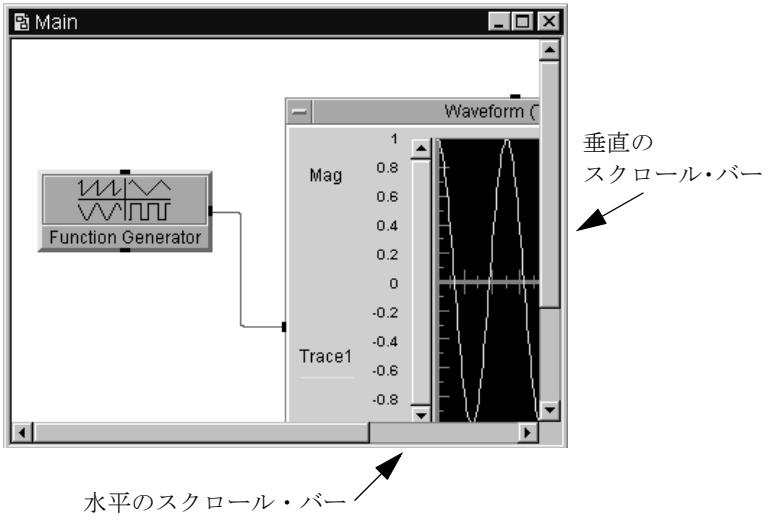


図16　作業領域のスクロール・バー

作業領域のクリア

[Edit] ⇒ **[Select All]**をクリックします。次にツールバー上の[Cut]ボタンをクリックします。これにより、アクティブ・ウィンドウ内のすべてのオブジェクトが削除され、カット・バッファに格納されます。

-または-

[File] ⇒ **[New]**を選択するか、ツールバー上の[New]ボタンをクリックします。変更を保存するかどうかを確認するメッセージが表示されます。

-または-

オブジェクトをクリックしてアクティブにした後、ツールバーの[Cut]ボタンをクリックして、オブジェクトを1つずつクリアします。

デフォルト設定の変更

[Default Preferences]ダイアログ・ボックスで、VEE環境のデフォルト設定を変更します。

ツールバーの[Default Preferences]ボタンをクリックします。

-または-

[File] ⇒ **[Default Preferences]**をクリックします。図17のように、[Default Preferences]ダイアログ・ボックスが表示されます。

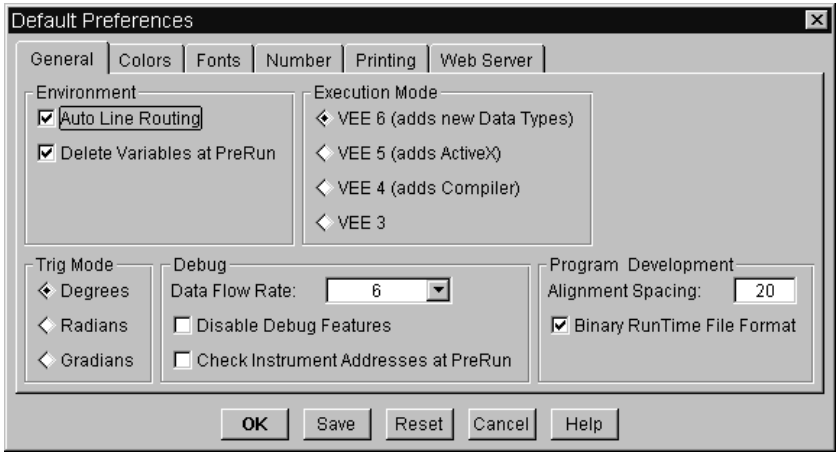


図17　[Default Preferences]ダイアログ・ボックス

このダイアログ・ボックスには**タブ**があり、編集するオプションを選択できます。

表3　[Default Preferences]ダイアログ・ボックスのタブの説明

タブ名	説明
[General]	先ほど示した方法で[Default Preferences]ダイアログ・ボックスを表示したとき、一番手前に表示されるデフォルトのタブです。[Environment]、[Execution Mode]などのパラメータ値を変更します。
[Colors]	VEE環境の色をカスタマイズします。
[Fonts]	VEE環境のフォントをカスタマイズします。
[Number]	数のデフォルト書式を変更します。
[Printing]	プリンタに関するパラメータ値を設定します。
[Web Server]	内臓のWebサーバをオンにしてリモートのWebブラウザからプログラムのモニタとトラブルシューティングを実行できます。

詳細は、VEEメニュー・バーから**[Help]** ⇒ **[Contents and Index]**を選択し、オンライン・ヘルプを参照してください。「**使用法**」、「**以下の項目についての説明**」、「**リファレンス**」のいずれかを選択します。

ピンと端子について

VEEプログラムは、作業領域内のオブジェクトとそのオブジェクトを接続するラインで**構成されます**。オブジェクトを接続するラインは、各オブジェクトの**ピン**間で接続されます。各オブジェクトには図18のように数個のピンがあります。図18では、例としてFormulaオブジェクトを挙げていますが、どのオブジェクトでも同じです。

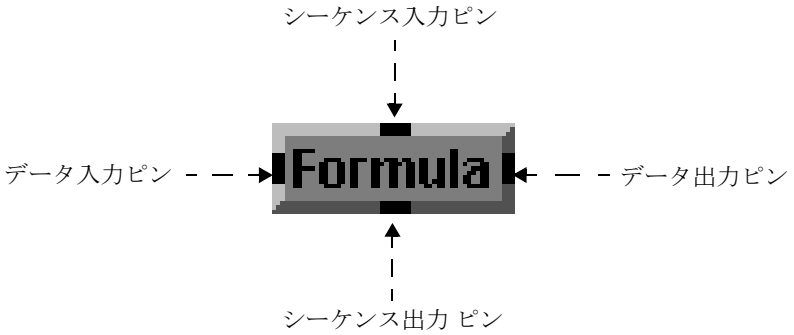


図18　　データ・ピンとシーケンス・ピン

表4

ピンの種類	説明
データ入力ピン	オブジェクトの左側にあるピン
データ出力ピン	オブジェクトの右側にあるピン
シーケンス入力ピン	オブジェクトの上辺にあるピン
シーケンス出力ピン	オブジェクトの下辺にあるピン

オブジェクト間でデータを伝達するには、データ入力ピンとデータ出力ピンを接続します。特に指定しないかぎり、ピンは上から下に実行されます。シーケンス・ピンによる接続はオプションです。シーケンス・ピンを使用すると、実行順序を制御できます。

注 記　詳細は、109ページの「オブジェクト内部のイベントの順番を追跡する」を参照してください。

オブジェクトをオープン・ビューで表示すると、データ入力/出力ピンは入力/出力端子として表示されます。オブジェクトがアイコン・ビューで表示されている場合は、アイコンをダブルクリックしてオープン・ビューに切り替えます。端子は、端子名や伝達されるデータの種類と値などの詳細情報を持ちます。端子のラベルは、オブジェクトがオープン・ビューで表示されており、オブジェクトの[Show Terminals]オプションが選択されている場合にのみ表示されます。オブジェクト・メニューの[Properties...]を参照してください。

たとえば、図19には2つのFormulaオブジェクトがあります。左側のFormulaオブジェクトには「**A**」と「**Result**」の端子ラベルが表示されていますが、右側のFormulaオブジェクトには、[Show Terminals]がオフになっているためにラベルが表示されていません。

[Show terminals]
オプションがオン

[Show terminals]
オプションがオフ

図19　オブジェクトの[Show Terminals]オプション

[Show Terminals]オプションのオン/オフを切り替えるには、オブジェクト・メニューから[Properties]を選択します。画面の左側に[Properties]ウィンドウが表示されます。プロパティ [Show Terminals]を選択し、それを[True]に設定します(図20を参照)。

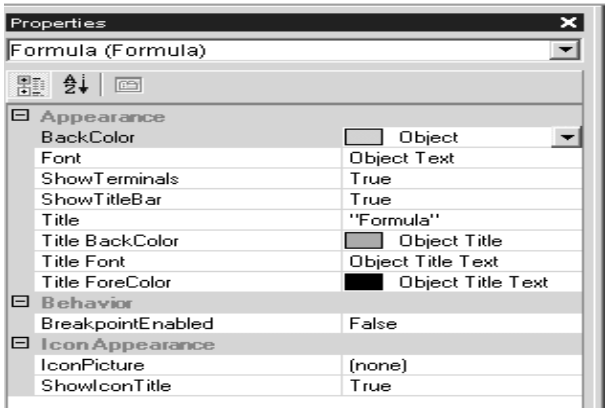


図20　[Show Terminals]プロパティの設定

プロパティを選択し、**[Show Terminals]を[False]**に設定します。再びプロパティを選択し、**[Show Terminals]を[True]**に設定します。

端子の追加

オブジェクトに端子を追加できます。たとえば、Formulaオブジェクトに2番目のデータ入力端子を追加できます。

オブジェクト・メニューを開き、**[Add Terminal]** ⇒ **[Data Input]**を選択します。

-または-

[Show Terminals]がオンの場合は、マウス・ポインタを端子領域(オープン・ビューで表示されているオブジェクトの左側の余白)に置き、**Ctrl+A**キーを押します(`Ctrl`と`A`のキーを同時に押します)。

図21に、データ入力端子を追加するために開かれたFormulaオブジェクト・メニューと、2番目の端子がすでに追加されている別のFormulaオブジェクトを示します。新しい端子にはラベル「**B**」が付けられています。データ入力が計測器ドライバのような特定の機能に結び付けられている場合は、その機能のメニュー名が付けられます。それ以外の場合、端子には「**A**」、「**B**」、「**C**」などの名前が付けられます。

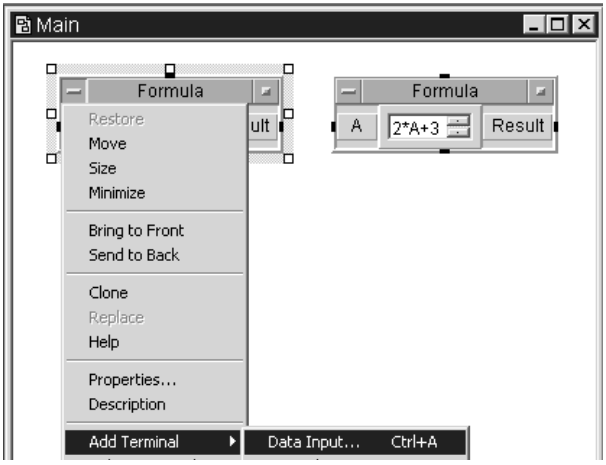


図21　　端子の追加

端子情報の編集

端子についての情報を表示するには、ラベル領域をダブルクリックします。たとえば、「**B**」をダブルクリックすると、図22のダイアログ・ボックスが表示されます。

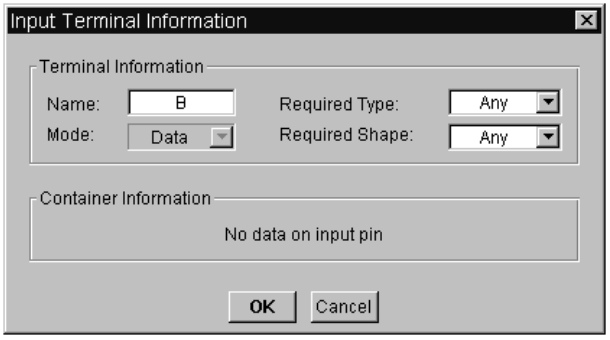


図22　　端子情報の表示

これで、端子を編集できます。ダイアログ・ボックスには3種類のフィールドがあります。

表5

フィールドの種類	説明
入力フィールド	背景が白く矢印の付いていないフィールドです。クリックすると、**入力できる**ようになります。たとえば、[Name]フィールドの**[B]**をクリックすると、端子の名前を変更できます。
ステータス・フィールド	灰色の背景のフィールドで、編集できません。たとえば、[Mode]フィールドは編集できません。
選択フィールド	背景が白で、右側に矢印が付いているフィールドです。フィールドまたは矢印をクリックするとドロップダウン・リストが表示されます。たとえば、[Required Type]フィールドの**[Any]**または矢印をクリックすると、図23のようなリストが表示され、リストをクリックすることによって別のデータ型を選択できます。

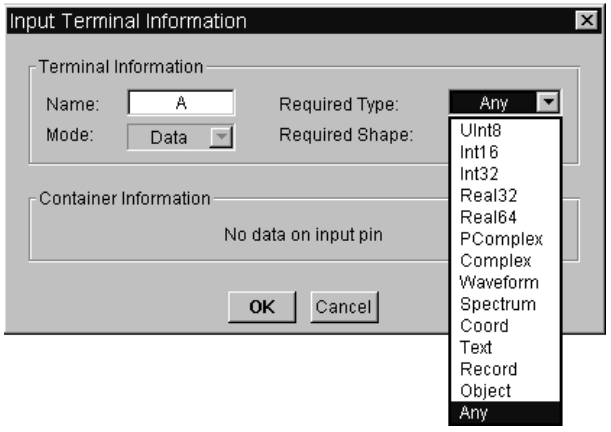


図23　　選択フィールドの使用

データ入力端子に「Any」以外のデータ型を選択すると、端子は、指定したデータ型またはその型に変換できるデータだけを受け入れるようになります。通常、[Required Type]および[Required Shape]フィールドには[Any]を設定しておくことをお勧めします。詳細は、VEEメニュー・バー

から**[Help]** ⇒ **[Contents and Index]**を選択し、オンライン・ヘルプを参照してください。「**使用法**」、「**以下の項目についての説明**」、「**リファレンス**」のいずれかを選択します。

端子の削除

オブジェクト・メニューを開き、**[Delete Terminal]** ⇒ **[Input . . .]**を選択します。または**[Delete Terminal]** ⇒ **[Output]**を選択して、削除する入力/出力端子を選択し、**[OK]**をクリックします。たとえば、図24は、**[Delete Terminal]** ⇒ **[Input . . .]**を選択した場合に表示されるダイアログ・ボックスです。

-または-

マウス・ポインタを端子の上に置き、CTRL+Dキーを押します。

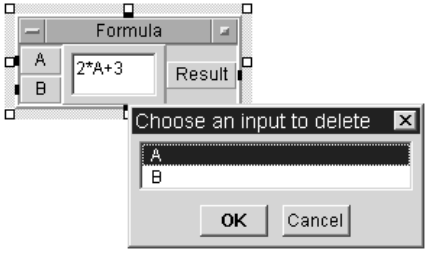


図24　[Delete Terminal]ダイアログ・ボックス

端子を削除しないことを決めた場合は、タスクバーから[Undo]ボタンを選択します。

オブジェクトの接続によるプログラムの作成

このセクションではプログラムについて紹介します。例題1-1では、VEEプログラムの作成、VEEの画面の印刷、プログラムのファイルへの保存を行います。

例題1-1: 波形表示プログラム

VEEプログラムは、実行可能な**オブジェクト・ダイアグラム**に接続されたVEEオブジェクトで構成されています。次のプログラムは波形を表示します。

VEEがすでに実行されている場合は、ツールバーの[New]ボタンをクリックするか、**[File]** ⇒ **[New]**を選択して、ワークスペースをクリアします。まだ起動していない場合は、VEEを起動して、次に進みます。

1 プログラムのドキュメントを作成します。**[Display]** ⇒ **[Note Pad]**を選択して、作業領域の上部中央に配置します。編集領域をクリックしてカーソルを置き、次のように入力します。これまでのテンプレート情報を削除し、以下を入力します。

```
Display Waveform generates a cosine waveform and
sends it to a real time display.
```
(波形表示プログラムは余弦波の波形を生成して、同時に画面に表示します)

画面によっては、Note Padのサイズを変更しなければならない場合があります。オブジェクトのサイズを変更するには、オブジェクト・メニューを開いて`[Size]`を選択し、サイズ変更矢印をオブジェクトの隅に移動してドラッグします。オブジェクトのいずれかの隅をクリックし、ドラッグすることもできます。

2 Function Generatorオブジェクトを追加します。
[Device] ⇒ **[Virtual Source]** ⇒ **[Function Generator]**を選択します。次に、輪郭線を作業領域の左側に置き、クリックしてオブジェクトを配置します。[Frequency]フィールド内でクリックし、「`100`」と入力して周波数に100を指定します。

3 Waveform (Time)オブジェクトを追加します。
[Display] ⇒ **[Waveform (Time)]**を選択して、図25のように作業領域の右側にオブジェクトを配置します。

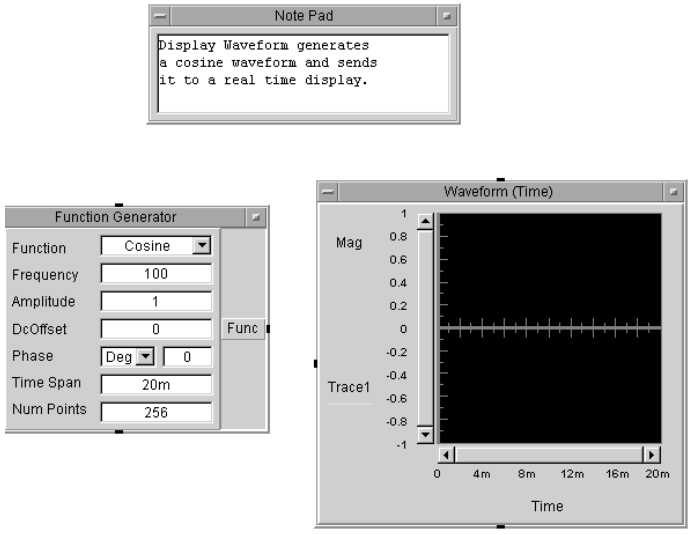


図25　　プログラムの作成

図25のFunction Generatorオブジェクトにある**Func**ラベルは**データ出力ピン**を表し、Waveform (Time)オブジェクトにある**Trace1**ラベルは**データ入力ピン**を表します。プログラムでは、オブジェクト間でデータ・ピンを接続することにより、プログラムの流れを決定します。

4 Function Generatorのデータ出力ピン(Funcの右)をWaveform (Time)のデータ入力ピン(Trace1の左)に接続して、プログラムを完成します。接続するには、カーソルをピンの一方へ移動します。

カーソルを接続可能なピンに近づけると、カーソルの形が変化します。マウスの左ボタンをクリックし、マウス・カーソルをもう一方のピンまで移動して、もう一度クリックします。2つのピンの間にラインが自動的に引かれ、プログラムが完成します。

一方のオブジェクトのタイトル・バーをドラッグして、移動してみてください。ピンや端子をドラッグすると、ラインが引かれるので注意してください。2つのオブジェクト間の論理パスに従って、ラインが自動的に引き直されます。

ラインが煩雑に見える場合は、**[Edit]** ⇒ **[Clean Up Lines]**を選択し、プログラム内のラインを引き直します。

プログラムの実行

5 同じ例題で練習を続けます。ツールバーの [Run] ボタンをクリックするか、**[Debug]** ⇒ **[Run]**を選択して、プログラムを実行します。プログラムは、図26に示すように、Waveform (Time)画面に100Hzの余弦波を表示します。なお、オブジェクトの周波数が異なってもかまいません。周波数は、この例題では重要ではありません。

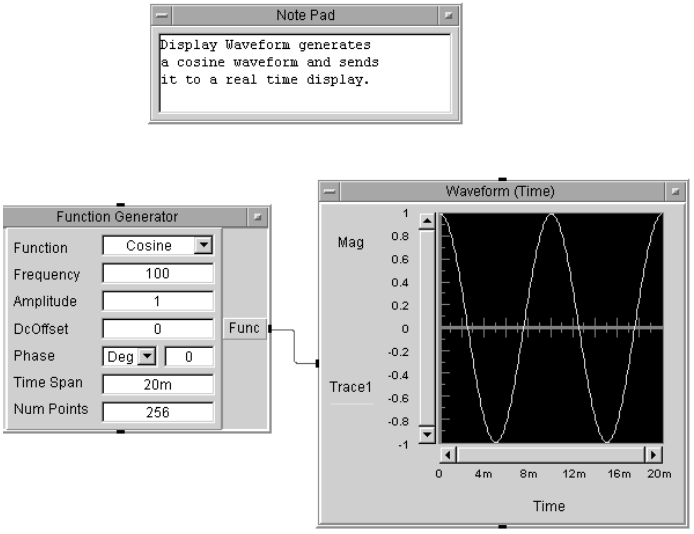


図26　　プログラムの実行

ツールバーでは、[Run]ボタンだけでなく、[Stop]、[Pause]、[Step Into]ボタンもプログラムを制御するために使用できます。プログラムの実行を休止(Pause)した場合は、[Resume]ボタン([Run]ボタンと同じ)を使って再開できます。ツールバーの[Step Into]ボタンを使用して、オブジェクトを1つずつ実行することもできます。

プログラムの実行方法を理解したら、ツールバーの[Run]ボタンをクリックするか、Ctrl+Gキーを押してください。または、次のキーボード・ショートカットを使用します。

表6

コマンド	キーストロークの組合わせ
[Pause]	Ctrl+P
[Resume]	Ctrl+G
[Step Into]	Ctrl+T

オブジェクトのプロパティの変更

オブジェクト・メニューから**[Properties]**を選択してオブジェクトのプロパティを変更する方法については、すでに説明しました。オープン・ビューでは、オブジェクトの一般的なプロパティを直接変更できます。Function Generatorオブジェクトには2種類のフィールドがあります。右側に矢印が付いているフィールドは選択フィールドです。

6 同じ例題で練習を続けます。Functionフィールドの[Cosine](または矢印)をクリックします。選択肢がドロップダウン・リストに表示されます。[Sine]をクリックして、図27のように正弦関数を選択します。[Function]フィールドが[Cosine]から[Sine]に変わります。

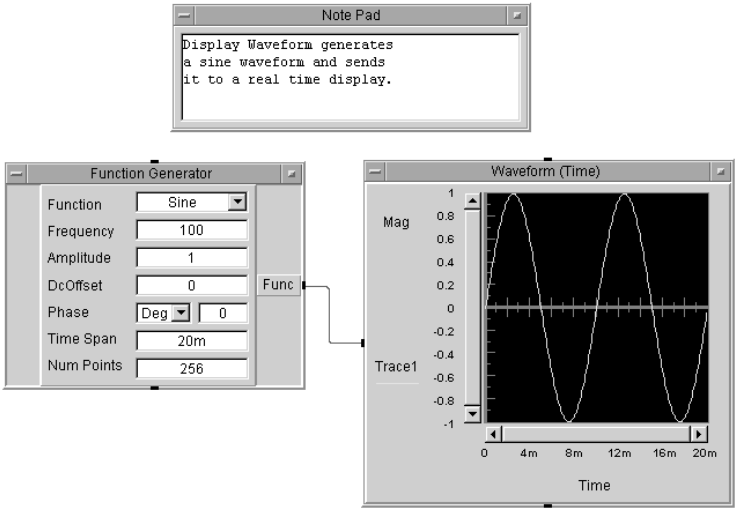


図27　　[Function]フィールドを正弦波に変更

ダイアログ・ボックスには、矢印が付いていないフィールドもあります。これらは**入力**フィールドで、クリックすると入力できます。フィールド内をクリックするとカーソルが表示されます。標準のキーボードとマウスによる編集方法でカーソルを移動し、目的の値を入力します。

7 [Frequency]フィールドの数値**100**の右側をクリックします。次に、マウスのボタンを押したままマウスを左へ動かして、図28に示すように最後の**0**が強調表示されるようにします。

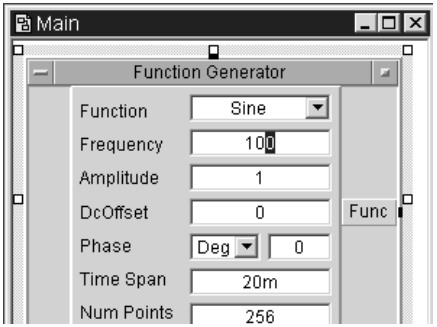


図28　[Frequency]フィールドの数字を強調表示する

8 Deleteキーを押して最後の**0**を削除して、[Frequency]の値を**[10]**に変更します。プログラムを実行します。図29のように表示されます。

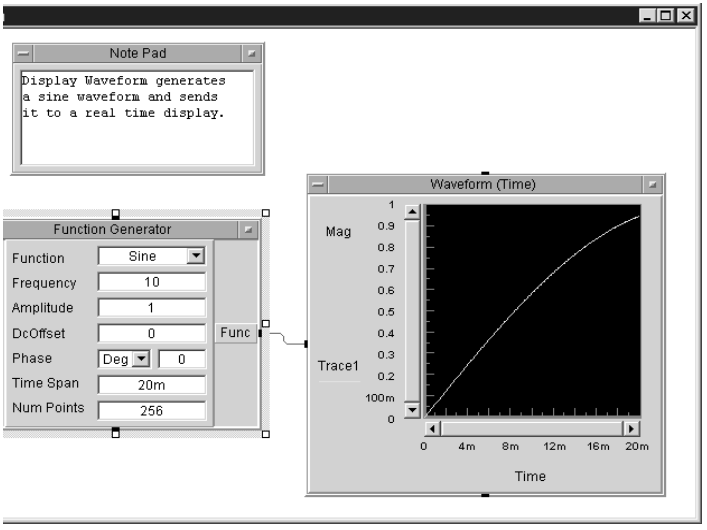


図29　[Frequency]フィールドを[10Hz]に変更した例

今度は、10Hzの正弦波の波形が表示されました。次に、オブジェクトのパラメータを以下のように変更してみます。

- Function Generatorオブジェクトの**[Deg]**(または矢印)をクリックし、位相の単位を[Rad]に変更します。次に、[Phase]値フィールドをクリックして、値「`PI`」を入力します。プログラムを実行して、表示される波形に位相のシフトが起きていることを確認します。次に、[Phase]値を[0]に、単位を[Deg]に戻します。
- Waveform (Time)オブジェクトのY軸の限界値は、あらかじめ-1から1までに設定されています。Y軸の名前[Mag]をクリックして、設定変更のためのダイアログ・ボックスを表示します。[Maximum]および[Minimum]フィールドをクリックして、限界値を[2]と[--2]に変更します。波形が新しい限界値で表示されます。X軸スケールに対して同様のパラメータの変更を行うには、[Time]をクリックします。

画面の印刷

9 同じ例題を使って練習を続けます。画面を印刷するには、**[File]** ⇒ **[Print Screen]**を選択します。Windowsでは、図30のダイアログ・ボックスが表示されます。

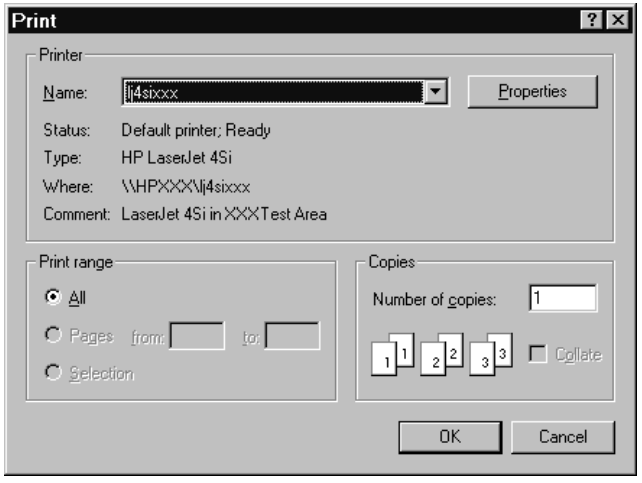


図30　　画面の印刷

[OK]をクリックすると、VEEはダイアログ・ボックスに示されたデフォルトのプリンタで画面を印刷します。プリンタの変更、印刷範囲の変更、部数の変更が可能です。**[プロパティ]**ボタンをクリックすると、ほかの設定も行うことができます。プリンタ・ドライバにより、表示されるダイアログ・ボックスも異なります。Windowsのダイアログ・ボックスの使用方法についての詳細は、『Microsoft Windowsのヘルプ』を参照してください。

ツールバーの**[Print Screen]**ボタンをクリックして、画面を直接印刷することもできます。

プログラムの保存

作業領域内のプログラムは、プログラムが完成しているかどうかにかかわりなく、いつでも保存できます。

10 同じ例題で練習を続けます。**[File]** ⇒ **[Save As . . .]**を選択して、ダイアログ・ボックスで必要事項を設定します。

[Save File]という名前のダイアログ・ボックスが表示されます。図31にPCで表示されるダイアログ・ボックスを示します。

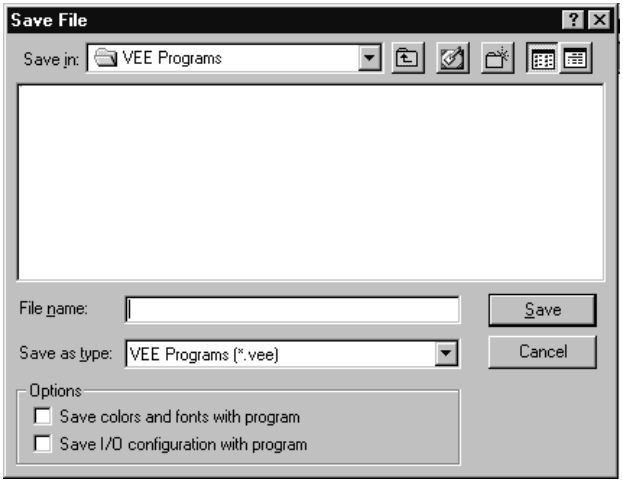


図31　[Save File]ダイアログ・ボックス(PC)

11 特に指定しないかぎり、Windows版VEEは、My Documentsディレクトリのサブディレクトリ**VEE Programs**にファイルを保存します。作成中の例題プログラムを保存するには、[ファイル名] フィールドに「`simple-program`」と名前を入力して**[保存]**をクリックします。拡張子を指定しないと、VEEが自動的に拡張子`.vee`をファイル名に追加します。

注 記

Windows版VEEでは、Windows 98、Windows 2000、Windows NT 4.0、およびWindows XPで認められている長いファイル名を使用できます。

PCの[Save File]ダイアログ・ボックスでは、次の設定を行うことができます。

表7　　ファイルの保存オプション

ファイルの保存オプション	説明
[保存する場所]	ドロップダウン・メニューを開いてディレクトリまたはドライブを変更します。フォルダを開くには、そのフォルダをダブルクリックします。
[ファイル名]	任意のファイル名を入力します。
[ファイルの種類]	VEEプログラムは、通常**.vee**という拡張子が付けられて保存されますが、ファイルの種類は変更できます。拡張子を付けずにファイル名を入力すると、拡張子**.vee**が自動的に付加されます。
[Save colors and fonts with program]	(オプション) [Default Preferences]メニューを使ってプログラムの色とフォントを変更し、次にプログラムをロードしたとき、デフォルトの色およびフォントではなく、変更した色およびフォントが使用されるように設定するには、この項目をチェックします。 チェックすると、デフォルト設定に加えた変更がプログラムの一部として保存されます。

表7　　ファイルの保存オプション

ファイルの保存オプション	説明
[Save I/O configuration with program]	(オプション) [Instrument Manager]で計測器を設定しており、次にプログラムをロードする人にもデフォルトではなくこの設定を使用して欲しい場合には、この項目をチェックします。 チェックすると、I/O設定がプログラムの一部として保存されます。

注 記

同じファイル名でプログラムを再保存するには、任意の時点で[Save]ボタンをクリックするか、Ctrl+Sキーを押します([File] ⇒ [Save])。プログラム作成中はファイルの保存を頻繁に行うことをお勧めします。編集したプログラムを別のファイル名で保存するには、Ctrl+Wキーを押すか、[File] ⇒ [Save As]を選択します。

Agilent VEEの終了

[File] ⇒ **[Exit]**を選択してVEEのアプリケーション・ウィンドウを閉じます。別のショートカットとして、Ctrl+Eキーを押してVEEを終了します。またはタイトル・バーの右端にある**[x]**ボタンをクリックします。

万一、VEEがマウスやキーボードに反応しなくなった場合は、次の操作を行ってください。

表8　　ハングした場合のVEEの終了方法

オペレーティング・システム	方法
Windows 98の場合	Ctrl+Alt+Deleteキーを押すと、さまざまなオプションを含むウィンドウが表示されます。Microsoft Windowsのウィンドウ内の指示に従って操作します。または**[終了]**をクリックします。
Windows NT 4.0、Windows 2000およびWindows XPの場合	Ctrl+Alt+Deleteキーを押し、**[タスクマネージャ]**ボタンをクリックします。アプリケーション・リストでVEEを選択し、**[終了]**をクリックします。

Agilent VEEの再起動とプログラムの実行

1 **[Start]** ⇒ **[Programs]** ⇒ **[Agilent VEE Pro]**を選択します。

2 **[File]** ⇒ **[Open]**を選択し、**[Open File]**ダイアログ・ボックスで必要な設定を行います。

ダイアログ・ボックスの形式は[Save File]ダイアログ・ボックスと同じです。Windows版VEEでは、インストール時に特に設定を行っていなければ、ユーザが作成したプログラムのデフォルトのディレクトリは、「VEE_USER」ディレクトリです。VEEはプログラムをメイン・ウィンドウで開きます。

3 **[Run]**ボタンをクリックします。このボタンは、小さい矢じりに似た形をしており、図32のように、ツールバーの[Debug]メニューの下にあります。

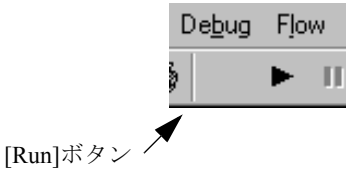


図32　　ツールバーの[Run]ボタン

注 記

Windowsでは、コマンド**vee.exe -r** *filename*を入力すると、VEEが起動して*filename*で指定したプログラムが自動的に実行されます。たとえば、Windowsのデスクトップ上にアイコンを作成し、アイコンの**プロパティ**に、特定のVEEプログラムを実行するための**ショートカット**を設定します。こうしておくと、デスクトップ上のアイコンをダブルクリックするだけで、自動的にを起動してプログラムを実行できます。詳細は、Windowsのコマンドおよびプロンプトのパスについてのヘルプを参照してください。

ワークスペースでの複数ウィンドウの管理

ここまでの説明のほとんどは、メイン・ウィンドウ内の作業領域についてでした。しかし、大きなプログラムでは、メイン・ウィンドウ内に複数のウィンドウが含まれることがあります。たとえば、UserObjectやUserFunctionなど、ユーザが定義したオブジェクトがプログラムに含まれる場合があります。UserObjectおよびUserFunctionは、メイン・プログラ

ムのサブ・ルーチンまたはサブ・プログラムと考えることができます。UserObjectおよびUserFunctionについては、第2章「Agilent VEEのプログラミング技術」の 76ページの「例題2-1: UserObjectの作成方法」セクションで詳述します。ここでは、VEEにおける複数ウィンドウを含むプログラムの管理方法を説明します。

図33は、4つのウィンドウがあるプログラムを示しています。それぞれのウィンドウには、メニュー・コマンドを備えたアイコン、タイトル、[最小化]ボタン、[最大化]ボタン、および[閉じる]ボタンがあります。ウィンドウを最大化すると、ワークスペースで使用できる領域がそのウィンドウで占有されます。ウィンドウを最小化すると、ワークスペースの最下部にそのウィンドウのアイコンが表示されます。ウィンドウを閉じると、ウィンドウはワークスペースから消えます。VEEでは、作業中のウィンドウはタイトル・バーが強調表示されます。

図33　　作業領域における複数ウィンドウ

図33のように、プログラムの階層構造がプログラム・エクスプローラにリストされます。この組込みのモジュール構造により、プログラムのすべての部分に簡単にアクセスできます。

プログラム・エクスプローラが表示されていない場合は、**[View]** ⇒ **[Program Explorer]**をクリックします。デフォルトの設定では、プログラム・エクスプローラが表示されます。[Program Explorer]メニューのチェックを外し、次に**[File]** ⇒ **[Default Preferences]**を選択すると表示されるダイアログ・ボックスで[Save]ボタンをクリックすると、次にVEEを起動したときからプログラム・エクスプローラが表示されなくなります。

メイン・ウィンドウを一番手前に表示するには、メイン・ウィンドウをクリックするか、プログラム・エクスプローラでメイン・ウィンドウのアイコンをダブルクリックします。

注 記

VEEでメイン・ウィンドウを閉じた後、[View] ⇒ [Main]を選択すると、再びメイン・ウィンドウが表示されます。

Agilent VEEプログラムの動作

VEEでは、プログラムの一般的な処理の流れを**伝達**と呼びます。プログラムの伝達は、プログラム内のオブジェクトの物理的な位置によって決定されるのでは**なく**、オブジェクトの接続のされ方によって決まります。伝達は、第一に**データ・フロー**によって決定されます。そしてデータ・フローは、オブジェクトのデータ入力ピンとデータ出力ピンがどのように接続されているかによって決定されます。

注 記

C、BASIC、Pascalなどほかのプログラミング言語では、プログラム文が実行される順序は、シーケンスと選択ルールの組合わせによって決定されます。一般にプログラム文は、特定の文が別の文やコードのスレッドへの分岐を生じないかぎり、プログラムに記述されている順番で実行されます。

プログラムでのデータ・フローの規則は次のとおりです。

- **データはオブジェクト内を左から右へ移動します。**つまり、データ・ピンがあるすべてのオブジェクトにおいて、左側のデータ・ピンは入力ピン、右側のデータ・ピンは出力ピンということになります。
- **オブジェクトのデータ入力ピンは、すべて接続する必要があります。**接続していない場合、プログラムを実行するとエラーになります。
- **オブジェクトは、すべてのデータ入力ピンが新しいデータを受け取るまで、動作しません。**
- **オブジェクトは、接続されている適切なデータ出力ピンがすべてアクティブになったときに動作を終了します。**

VEEでは、シーケンス入力/出力ピンを使って動作の順番を変更できます。しかし、通常はシーケンス・ピンを使用する必要はありません。**一般に、シーケンス・ピンの使用はお勧めできません。できれば、データ・フローによってプログラムの実行を制御してください。**

例題1-2: データ・フローと伝達の表示

データ・フローを表示するには、まず作成したプログラムを開きます。ツールバーの [Open] ボタンをクリックして、「**simple-program.vee**」プログラムを開いてください。**simple-program.vee**」プログラムについては、52ページの「例題1-1: 波形表示プログラム」セクションで説明されています。次に、プログラムを実行します。プログラムが図34のように表示されますが、パラメータを変更している場合は結果が多少異なる場合もあります。

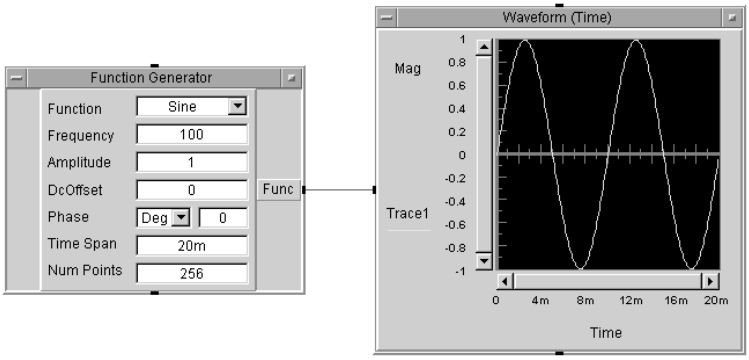


図34　simple-program.veeプログラムの典型的な表示

ここでは、Function Generatorオブジェクトのデータ出力ピンが、Waveform (Time)オブジェクトのデータ入力ピンに接続されています。プログラムを実行しても、Waveform (Time)オブジェクトは、Function Generatorオブジェクトからデータを受け取るまで動作しません。これは、データ・フローの簡単な一例です。

例題1-3: Noise Generatorの追加

Noise Generatorオブジェクトを**simple-program.vee**に追加することにより、図35のように「ノイズの大きな正弦波」を加えます。

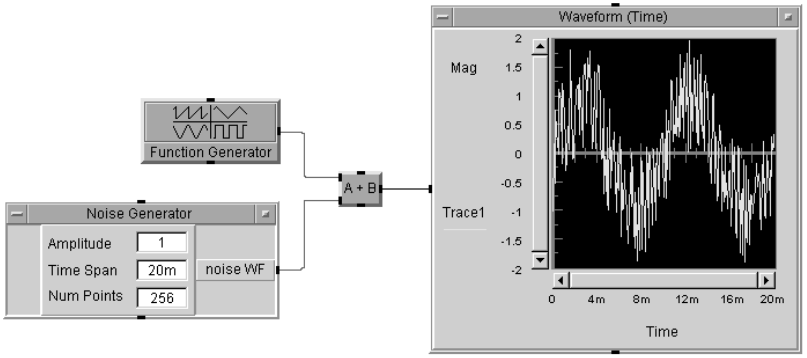


図35　　Noise Generatorオブジェクトを追加した例

注 記

このマニュアルに記載されている練習問題やプログラミング例で使用したVEEプログラムの多くは、VEEに付属しています。**[Help]** ⇒ **[Open Example...]** ⇒ **[Manual]** ⇒ **[UsersGuide]**を選択してください。

1. 元のプログラムで、Function GeneratorオブジェクトとWaveform (Time)オブジェクトを接続しているラインを削除します。ツールバーの[Delete Line]ボタンをクリックし、次にラインをクリックします。または、`Shift+Ctrl`キーを押したままでラインをクリックします。
2. Function Generatorを最小化してアイコンにします。
3. Noise Generatorオブジェクトを追加します。それには、**[Device]** ⇒ **[Virtual Source]** ⇒ **[Noise Generator]**を選択します。
4. **[Device]** ⇒ **[Function & Object Browser]**を選択し、**A+B**オブジェクトを追加します。

   Function & Object Browserが図36のように表示されます。[Type]では、[Operators]を選択します。[Category]では、[Arithmetic]を選択します。[Operators]では、**[+]**を選択します。**[Create Formula]**をクリックし、作業領域のFunction GeneratorオブジェクトとWaveform (Time)オブジェクトの間にオブジェクトを配置します。**A+B**オブジェクトを最小化します。

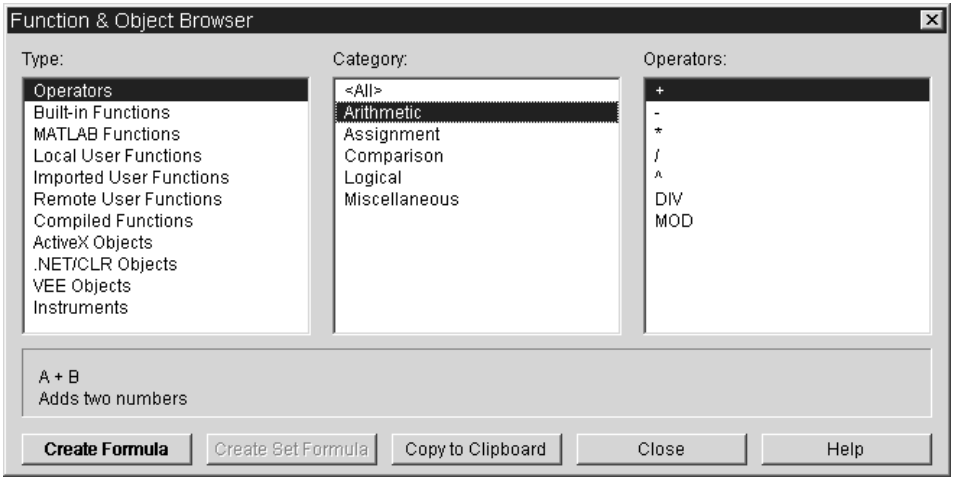


図36　Function and Object Browser

1 入力ピンと出力ピンを図37のように接続します。

2 プログラムを実行します。

A+Bオブジェクトは、Function Generatorオブジェクトと Noise Generatorオブジェクトが動作するまで動作しません。またFunction Generatorと Noise Generatorは、どちらが先に動作してもかまいません。結果は同じになります。

A+Bのデータ入力ピンの両方がデータを受け取ると、**A+B**オブジェクトは動作して2つの信号を合計し、結果をWaveform (Time)オブジェクトに出力します。

注 記　プログラムではデータ・フローが動作を決定します。

動作の順番を表示するには、[Debug]メニューで[Show Execution Flow]と[Show Data Flow]をオンにします。または、ツールバーの該当するボタンをクリックします。プログラムを再度実行します。オブジェクトが動作すると、そのオブジェクトは強調表示され、小さい四角形のマーカがラインに沿って移動してデータの流れを表します。

注 記

[Show Execution Flow]と[Show Data Flow]を同時に、または個別に有効にするには、ツールバーの該当するボタンをクリックするか、[Debug]メニューでそれぞれのコマンドを選択します。プログラムの実行速度が遅くなるため、通常、これらのコマンドはオフにしておくことをお勧めします。

例題1-4: Amplitude入力端子およびReal64 Sliderオブジェクトの追加

simple-program.veeプログラムにAmplitude(振幅)入力端子とReal64 Sliderオブジェクトを追加します。

1 オブジェクト・メニューをクリックするか、またはマウス・ポインタをNoise Generatorの左側の端子領域に置き、`Ctrl+A`キーを押します。入力端子を追加するためのダイアログ・ボックスが図37のように表示されます。

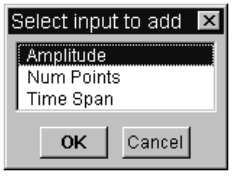


図37　　入力端子を追加する例

2 [Amplitude]を選択し、**[OK]**をクリックします。Amplitude入力端子が表示されます

これで、Noise GeneratorオブジェクトにAmplitude入力ピンが追加され、振幅データを実数として入力できます。VEEには、[Data]メニューにReal64 Sliderというオブジェクトがあり、データ入力を簡単に行うことができます。Real64 ConstantオブジェクトまたはReal64 Knobオブジェクトを使用することもできます。

3 **[Data]** ⇒ **[Continuous]** ⇒ **[Real64 Slider]**を選択して、Real64 Sliderオブジェクトを追加します。次に、図38のように、Real64 Sliderオブジェクトのデータ出力ピンをAmplitude端子に接続します。プログラムを実行します。

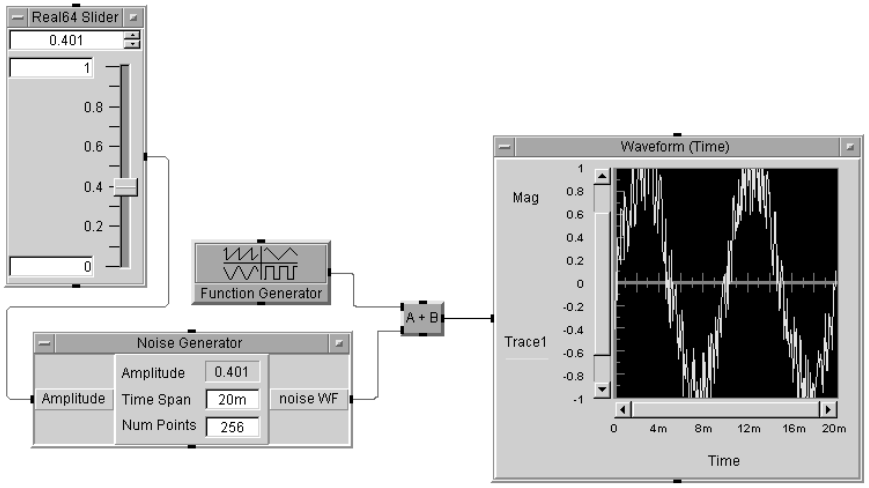


図38　　Real64 Sliderオブジェクトを追加した例

Real64 Sliderオブジェクトのスライダをドラッグして、ノイズの振幅を変更してみてください。プログラムを実行すると、ノイズの振幅が変化します。表示される波形のノイズ成分はReal64 Sliderが出力する値で決まります。

ここでも、実行順序はデータ・フローで決定されます。Noise Generatorは、Real64 Sliderが動作するまで動作しません。**A+B**オブジェクトは、Function GeneratorとNoise Generatorが動作するまで動作しません。ただし、どちらのオブジェクトが先に動作するかは問題になりません。最後に、**A+B**オブジェクトが動作した後でWaveform (Time)オブジェクトが動作します。

注 記

マウスをライン上に合わせたままにすると、出力値が表示されます。たとえば、Real64 SliderオブジェクトからNoise Generatorオブジェクトまでのライン上にマウスを合わせたままにすると、0.401と表示されます。ライン上の数値(0.401)は、図39のようにReal64 Sliderに表示される値と一致します。オブジェクトはアイコン・ビューで表示されています。

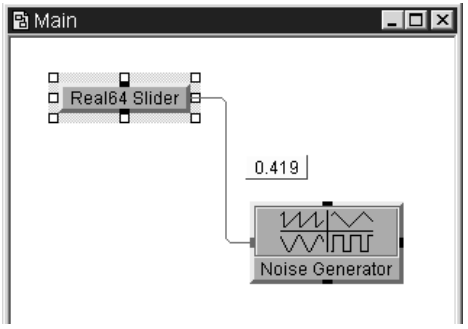


図39　　出力ピン上の値を表示する

4 プログラムを「**simple-program.vee**」にもう一度保存します。次章では、このプログラムにさらに機能を追加します。

この章の復習

本章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。

- メイン・メニュー・バーおよびオブジェクト・メニューからオンライン・ヘルプを参照する。
- VEEを起動します。
- メイン・メニュー・バー、ツールバーのボタン、作業領域、ステータス・バーの位置を確認する。
- プログラム・エクスプローラとその目的について説明する。
- メイン・メニューおよびオブジェクト・メニューからメニュー項目を選択する。
- オブジェクトに対して、移動、名前の変更、アイコン化、オープン・ビューへの切替え、サイズの変更、選択、選択解除、削除、クローン作成などの操作を実行する。
- 作業領域の移動、クリアを行う。また、複数ウィンドウを管理する。
- オブジェクト上のデータ・ピンおよびシーケンス・ピンの位置を確認し、それぞれの目的を説明する。
- 端子を検査し、名前を変更する。
- オブジェクトを接続してプログラムを作成し、波形データをシミュレートする。
- プログラムの作成、実行、印刷、保存を行う。
- VEEを終了し、プログラムを再度開く。
- VEEプログラムのデータ・フローを説明する。

2
Agilent VEEのプログラミング技術

Agilent VEEのプログラミング技術

この章の内容

- UserObjectの作成方法
- 入力用ダイアログ・ボックスの追加方法
- データ・ファイルの使用方法
- パネル・ビュー (オペレータ・インタフェース)の作成方法
- データの数式処理方法
- 計測器との通信方法
- プログラムのドキュメント作成
- デバッグ・ツールの使用方法

平均所要時間: 2時間

概要

この章では、ユーザが独自のプログラムを構築するときに役立つVEEプログラミング技術を学びます。たとえばVEEでは、UserObjectと呼ばれるカスタム・オブジェクトを作成できます。また、オペレータ用にプログラムの必要な部分だけを表示するインタフェースを作成することもできます。これらは、プログラムのパネル・ビューに表示されます。

VEEからファイルにデータを書き出したり、ファイルからVEEにデータを読み込むことができます。データ・ファイルおよび関連するI/Oトランザクションは、計測器、ファイル、文字列、オペレーティング・システム、インタフェース、ほかのプログラム、プリンタとの通信など、さまざまな目的に使用できます。

VEEは数多くのデータ形式をサポートします。また、広範な数式処理能力を提供します。VEEを使用して、さまざまな方法で計測器と通信できます。VEEはまた、プログラム内の問題点をデバッグするための強力なデバッグ・ツールを提供します。

一般的な技術

VEEでは、メイン・プログラム内に「UserObject」と呼ばれるオブジェクトの論理的なグループを作成できます。UserObjectオブジェクト(以下UserObject)は、UserObject編集ウィンドウにオブジェクトの論理的なグループを配置することによって作成します。UserObject編集ウィンドウ内でメイン･プログラムと同じ方法で入力ピンと出力ピンを接続します。また通常のオブジェクトと同じように、UserObject自体をメイン・プログラム内のほかのオブジェクトに入力ピンと出力ピンで接続できます。

UserObjectを開発するということは、メイン・プログラム内で有用な処理を行う独自のコンテキストを作成するということです。メイン・プログラムの作業領域内でスペースを節約できるうえに、構造化によってわかりやすいプログラムを記述することができます。

VEEプログラムでは、メイン・プログラム内に複数のUserObjectを入れ子構造で含めることができます。UserObjectはそれぞれ、メイン・ウィンドウ内ではアイコン・ビューで表示されます。メイン・プログラムにあるUserObjectのアイコン・ビューと、対応するUserObject編集ウィンドウとを関連付けるには、編集ウィンドウでUserObjectに名前を付けます。対応するアイコン・ビューも同じ名前になります。たとえば、UserObjectにAddNoiseという名前を付けた場合、メイン･プログラム内のアイコン･ウィンドウとUserObject編集ウィンドウのタイトル・バーの両方にAddNoiseが表示されます。次の例題では、UserObjectの作成方法を学びます。

例題2-1: UserObjectの作成方法

VEEプログラムでUserObjectを作成する方法は、2通りあります。

- メニュー・バーから**[Device]** ⇒ **[UserObject]**を選択すると、メインウィンドウに空のUserObjectアイコンが現れるので、このアイコンにオブジェクトを追加します。UserObjectアイコンをダブルクリックすると、図40のようにオープン・ビュー表示になります。
- プログラム内のオブジェクトを選択します。次に、**[Edit]** ⇒ **[Create UserObject]**をクリックして、選択したオブジェクトからUserObjectを作成します。

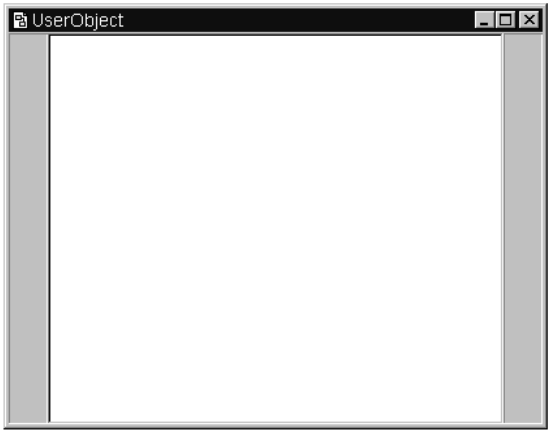


図40　　UserObject編集ウィンドウ

作成したUserObjectは、メイン・プログラムの一部になります。またUserObject編集ウィンドウの表示方法には、次のように、アイコン、オープン・ビュー、画面最下部に最小化の3通りあります。

- [閉じる]ボタンをクリックして編集ウィンドウを閉じます。UserObjectはメイン・ウィンドウ内でアイコンとして表示されます。
- [最大化]ボタンをクリックして編集ウィンドウを最大化します。UserObject編集ウィンドウは、VEEワークスペース内で使用できる全領域を占有します。
- [最小化]ボタンをクリックして編集ウィンドウを最小化します。最小化されたUserObjectはVEEのワークスペースの最下部に表示されます。

注 記　UserObjectのアイコン・ビューはメイン・ウィンドウに常に存在しており、メイン・ウィンドウ内でほかのオブジェクトに接続できます。

注 記

開始前に[View]メニューの[Program Explorer]の選択を解除して、メインでより広い画面スペースが使えるようにします。

では、プログラムからUserObjectを作成してみます。

1. 69ページの「例題1-4: Amplitude入力端子およびReal64 Sliderオブジェクトの追加」で作成したプログラム(**simple-program.vee**)を開きます。プログラムは、主作業領域に表示されます。
2. プログラムからReal64 Sliderを削除します。このオブジェクトは、この例題では使用しません。Real64 Sliderのオブジェクト・メニューを開いて[Delete]を選択するか、**Real64 Slider**のオブジェクト・メニュー・ボタンをダブルクリックします。

注 記

Real64 Sliderオブジェクトを削除し、Noise Generatorに入力ピンを残した状態でプログラムを実行すると、Noise Generator上の入力ピンAmplitudeが接続されていないというエラー・メッセージが表示されます。VEEプログラムを実行するときは、すべての入力ピンを接続する必要があります。

3. Noise Generatorオブジェクトで、オブジェクト・メニュー・ボタンをクリックするか、オブジェクト上でマウスの右ボタンをクリックして、オブジェクト・メニューを開きます。**[Delete Terminal]** ⇒ **[Input]**を選択し、削除する入力ピンを選択するためのダイアログ・ボックスで[Amplitude]を強調表示し、**[OK]**をクリックします。
4. プログラムの名前を変更します。**[File]** ⇒ **[Save As...]** を選択し、新しい名前「`usrobj-program1.vee`」を入力します。
5. 次に、Noise Generatorオブジェクトを最小化して、図41のようにオブジェクトを再配置します。

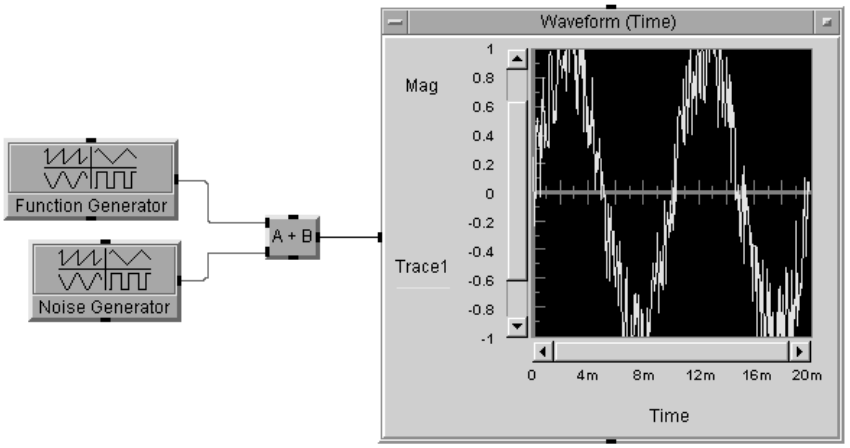


図41　　初期段階のusrobj-program.vee

6 Ctrlキーを押したままマウスの左ボタンを使用して、**Noise Generator**および**A+B**オブジェクトを選択します。**[Edit]** ⇒ **[Create UserObject]**を選択します。「Create UserObject」のラベルが付いたダイアログ・ボックスが表示されます。必要に応じて新しい名前を入力することにより、オブジェクトの名前を変更できます。**[OK]**をクリックしてUserObjectを作成します。

UserObjectの編集ウィンドウにはNoise Generatorオブジェクトと**A+B**オブジェクトが含まれます。またメイン・ウィンドウには、図42のように適切な入力ピンと出力ピンが付けられ、接続済みのUserObjectが自動的に作成されます。

キーボードのHomeボタンを押すだけで、UserObjectの左上にアイコンを配置できます。

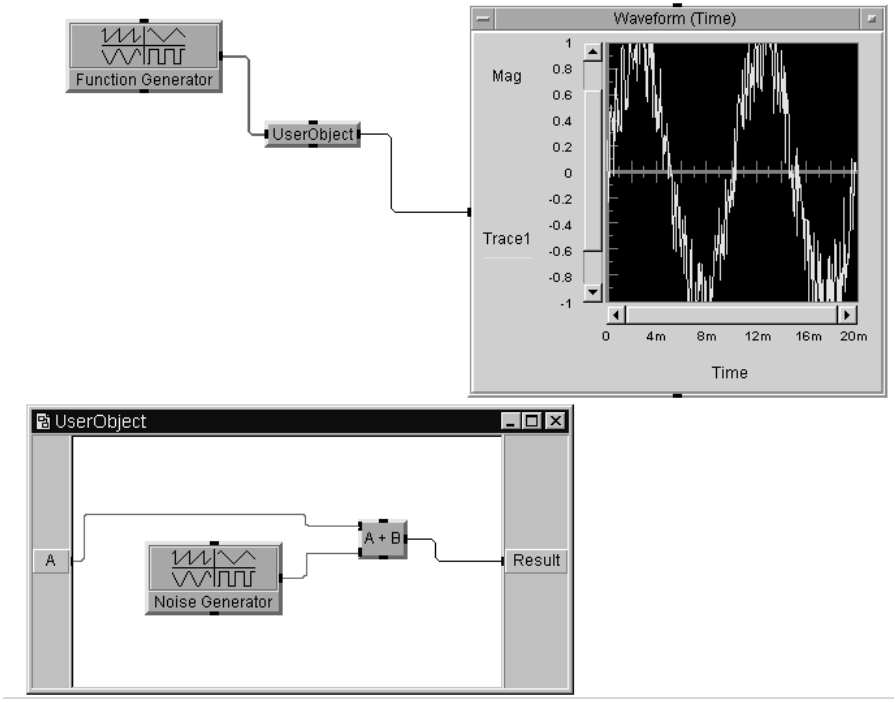


図42　　UserObjectの作成方法

注 記

[Create UserObject]を実行する前にオブジェクトを整理しておくと、作業が楽になります。UserObjectに入れるオブジェクトを1か所に集めておかなかった場合、UserObjectは選択したオブジェクトすべてを囲む大きさになります。その場合は、UserObjectの作業領域を整理して大きさを変更し、主作業領域内の適切な位置にUserObjectを移動します。しかし、あらかじめオブジェクトを論理的に配置した場合は、クリーンアップを行う方が簡単です。

注 記

[Edit] ⇒ [Clean Up Lines]を使用して、プログラム内の経路指定ラインを整理できます。このコマンドはコンテキスト依存です。UserObjectのラインを整理するには、UserObject編集ウィンドウがアクティブでなければなりません。UserObject編集ウィンドウをクリックしてから、**[Edit] ⇒ [Clean Up Lines]**を使用します。

初めにUserObject編集ウィンドウ内でUserObjectを作成してから、UserObjectをアイコン表示にして使用すると、画面のスペースを節約できます。

7 UserObjectの動作を把握しやすいように、UserObjectのタイトルをAddNoiseに変更します。タイトル・バーをダブルクリックして[Properties]ダイアログ・ボックスに新しいタイトルを入力します。図43は、これによりプログラムが追跡しやすくなった様子を示しています。

オブジェクトの[Properties]ダイアログ・ボックスをすばやく表示するには、タイトル・バーをダブルクリックします。

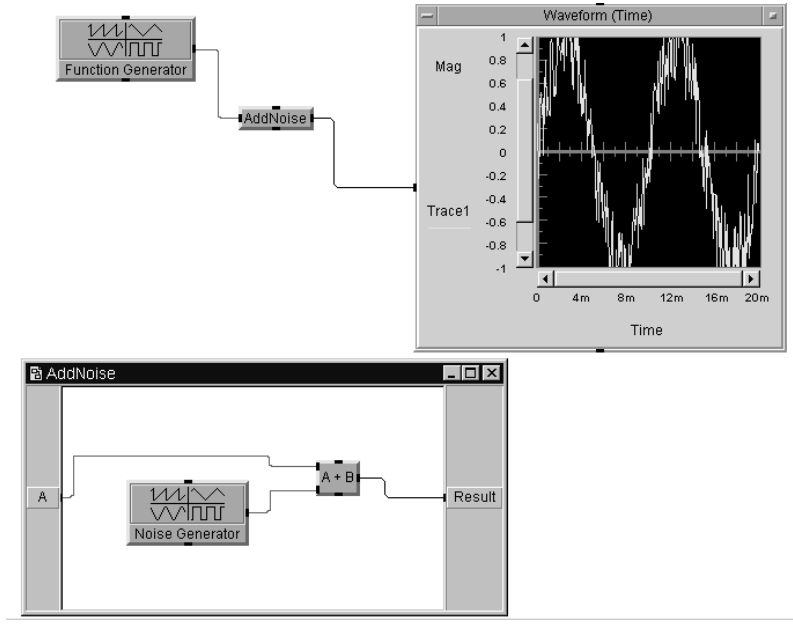


図43　　UserObjectの名前を「AddNoise」に変更

8 **[Run]**ボタンをクリックして、図44のようにノイズの大きな余弦波を表示します。AddNoiseが最小化されてワークスペースの最下部にアイコンで表示されていることを確認してください。AddNoiseを最小化するには、タイトル・バーにあるアンダーバー(_)で示された[最小化]ボタンをクリックします。

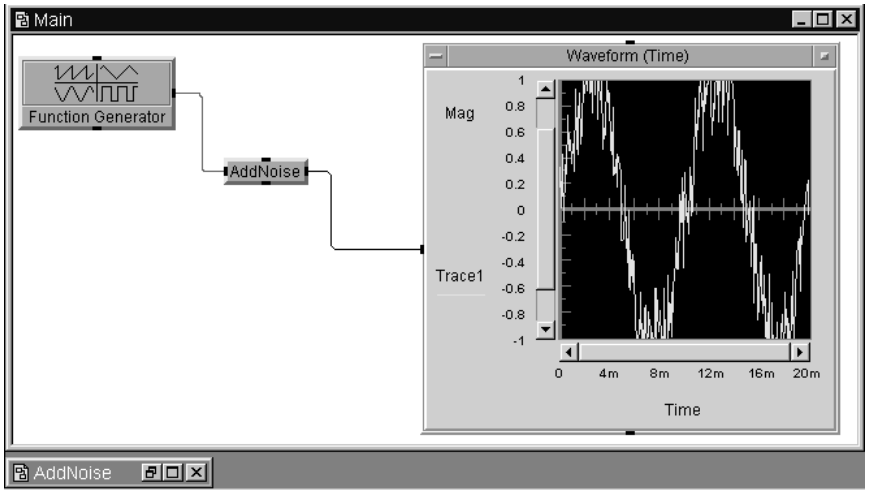


図44　　ノイズの大きな余弦波

有効なUserObjectであるためには、プログラム内で論理的目的を満たす必要があります。この独自のオブジェクトは、単なるスペース節約の手段ではなく、むしろプログラムを構造化するための手段です。UserObjectを使用すると、VEEプログラムに「トップダウン」設計の手法を取り入れることができます。VEEには*UserFunction*と呼ばれるオブジェクトもありますが、これは繰返し使用できるコード・モジュールです。UserObjectとUserFunctionについての詳細は、第9章「Agilent VEE 関数の使用方法」(329ページ)を参照してください。

UserObjectについての詳細は、VEEメニュー・バーから**[Help]** ⇒ **[Contents and Index]**を選択してオンライン・ヘルプを参照してください。「**使用法**」、「**以下の項目についての説明**」、「**リファレンス**」のいずれかを選択します。

この後のセクションでも、この例題を使って学習を続けます。ここでいったん学習を終了する場合は、プログラムに**usrobj-program3.vee**と名前を付けて保存してください。

例題2-2: ユーザ入力用ダイアログ・ボックスの作成方法

プログラム**usrobj-program3.vee**を開いていない場合は、まずプログラムを開きます。

[Data] ⇒ **[Dialog Box]**サブメニューには、ダイアログ・ボックス作成用として[Text Input]、[Int32 Input]、[Real64 Input]、[Message Box]、[List Box]、[File Name Selection]の6つの選択肢があります。それぞれテキスト、整数、実数を入力する場合に、ダイアログ・ボックスで、プロンプトやラベル、デフォルト値、値の制約、エラー・メッセージなどを設定できます。これらのダイアログ・ボックスをプログラムに入れると、プログラム実行時にポップアップ入力ボックスが表示されます。

1 **[Data]** ⇒ **[Dialog Box]** ⇒ **[Int32 Input]**を選択して、**Function Generator**の左にInt32 Inputオブジェクトを配置します。[Prompt/Label]フィールドを[Enter Frequency:]に変更します。変更を行う前に、忘れずにフィールドをクリック・アンド・ドラッグして強調表示してください。[Default Value]を[100]に変更します。

   入力フィールドをダブルクリックして入力内容を強調表示することもできます。

2 **[Value Constraints]**の最低値を**[1]**に、最高値を**[193]**に変更します。図45に示すように、エラー・メッセージの内容も、新しく入力した値を反映するように変更します。最後に**Int32 Input**オブジェクトをアイコン化します。

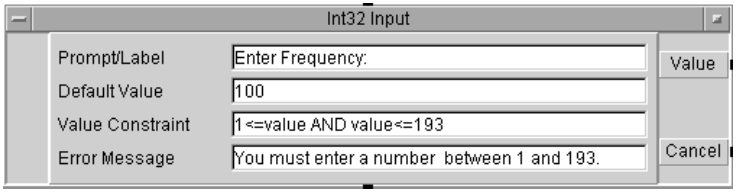


図45　Int32 Input設定ボックス

3 **Function Generator**のオブジェクト・メニューを開き、**[Add Terminal]** ⇒ **[Data Input]**を選択します。**[Select input to add]**のダイアログ・ボックスで**[Frequency]**を選択して**[OK]**をクリックします。

4 Int32 Inputオブジェクトの上の出力ピンをFunction Generatorの入力ピンに接続します。これで、Frequencyは、入力ピンを介してのみ変更可能になり、[Frequency]入力フィールドでは変更できなくなります。プログラムは図46のように表示されます。

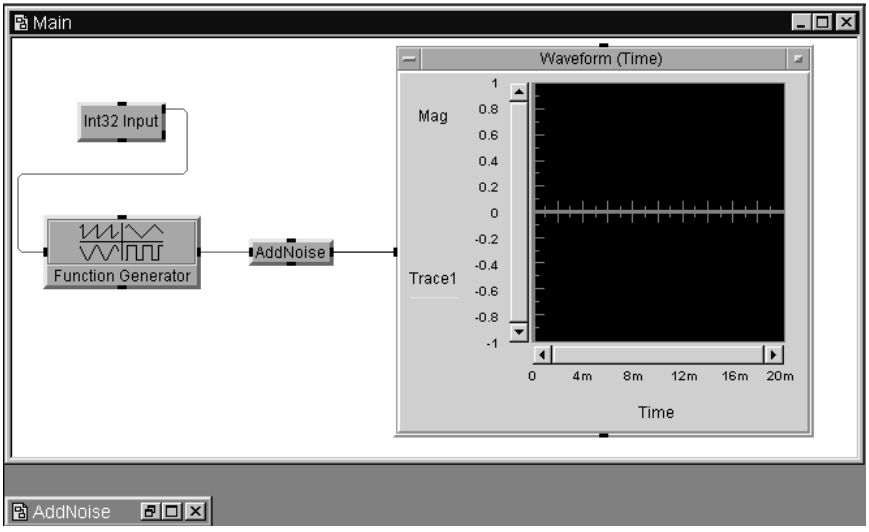


図46　Int32 Inputを追加したusrobj-program.veeプログラム

5 プログラムを実行します。Int32 Inputの入力ボックスが表示され、周波数の入力を求められます。入力ボックスにさまざまな周波数を入力してプログラムを実行してみてください。図47は、プログラム実行時にポップアップ入力ボックスが表示されたところを示したものです。ポップアップ・ボックスをクリック・アンド・ドラッグするだけで表示位置を制御できます。

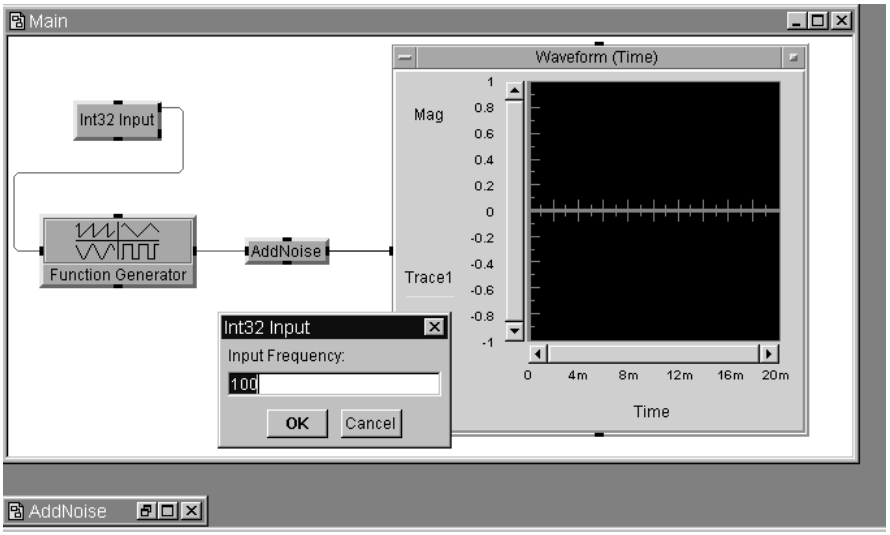


図47　　プログラム実行時のポップアップ入力ボックス

193より高い周波数を指定すると、エラー・メッセージ・ボックスが表示されます。設定したエラー・メッセージが正確に表示されることを確認してください。

この後のセクションでも、この例題を使って学習を続けます。ここでいったん学習を終了する場合は、プログラムに**usrobj1-program4.vee**と名前を付けて保存してください。

注 記

このマニュアルに記載されている練習問題やプログラミング例で使用したVEEプログラムの多くは、VEEに付属しています。[Help] ⇒ [Open Example . . .] ⇒ [Manual] ⇒ [UsersGuide]を選択してください。

例題2-3: データファイルの使用方法

プログラムにTo FileおよびFrom File オブジェクトを入れることにより、VEEからデータをデータ・ファイルに書き出したり、ファイルのデータをVEEに読み込むことができます。例題では、構築したプログラムの詳細ビューにTo Fileオブジェクトを追加します。

プログラム**usrobj-program4.vee**を開いていない場合は、まずプログラムを開きます。

1 **[I/O]** ⇒ **[To]** ⇒ **[File]**を選択して、To Fileオブジェクトを主作業領域に配置します。

2 デフォルトのファイル名**myFile**を`wavedata`に変更します。

[Clear File At PreRun & Open]の左にチェック・マークが付いていない場合は、小さい入力ボックスをクリックします。To Fileのデフォルト設定では、既存ファイルにデータを追加します。しかし、この場合は、プログラムを実行するたびにファイルをクリアします。この時点で、To Fileオブジェクトは図48のように表示されるはずです。

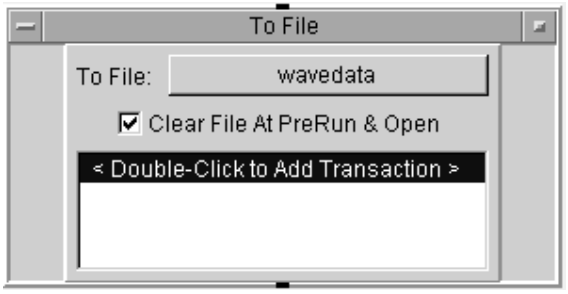


図48　　データ・ファイルの追加

3 データを書き出すには、**Double-Click to Add Transaction**のラベルが付いた領域をダブルクリックします。図49のダイアログ・ボックスが表示されます。**[TEXT]**フィールド(または右側の矢印)をクリックしてデータ型のドロップダウン・リストを表示し、**[CONTAINER]**をクリックします。**[OK]**をクリックします。**[I/O Transaction]**ダイアログ・ボックスで**[OK]**をクリックすると、To Fileオブジェクトに入力ピン「**a**」が自動的に追加されることを確認してください。

[WRITE CONTAINER]以外のトランザクションのオプションを確認するには、To Fileのオブジェクト・メニューでヘルプを参照してください。トランザクションについては、『*VEE Pro Advanced Techniques*』の付録および本書の第5章「テスト結果の保管方法と読み取り方法」を参照してください。

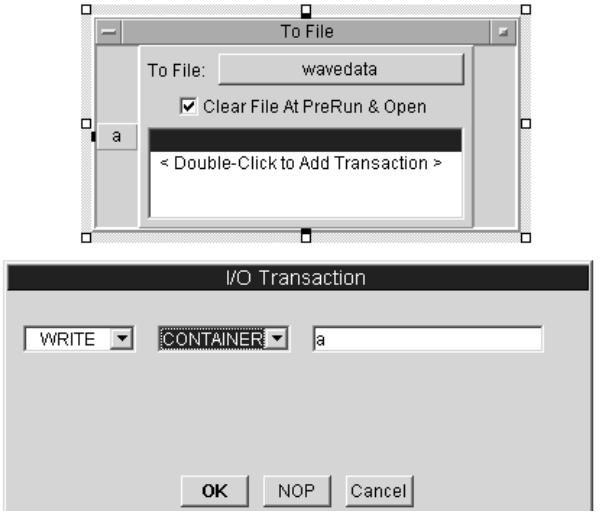


図49　　I/Oトランザクションの選択

4 AddNoise UserObjectのデータ出力ピンをTo Fileのデータ入力ピンに接続します。プログラムは図50のように表示されます。

注 記　1つのデータ出力ピンを複数のデータ入力ピンに接続できます。

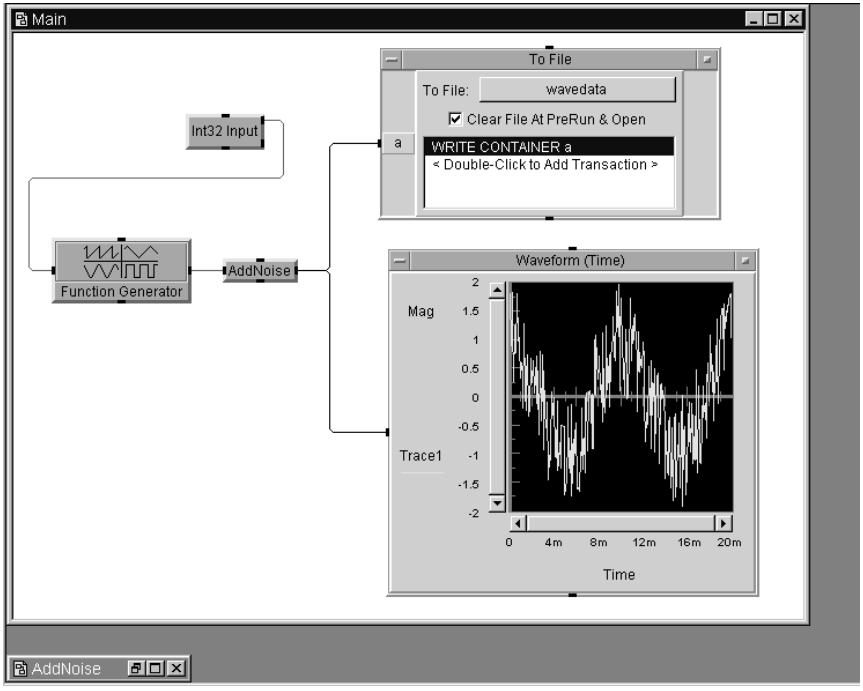


図50　　To Fileオブジェクトの追加

5 ツールバーの **[Run]** ボタンを再度クリックして、プログラムをテストします。プログラムは、AddNoise UserObjectが出力したノイズの大きな余弦波を表示し、波形のデータ・コンテナをwavedataファイルに書き出します。

To Fileオブジェクトをダブルクリックしてオープン・ビューにします。次に入力端子「**a**」をダブルクリックして内容を確認します。256の点を持つ配列が表示されます。

データを読み戻すには、From Fileオブジェクトをプログラムに追加します。

6 **[I/O]** ⇒ **[From]** ⇒ **[File]**を選択し、それを[Main]作業領域内に配置します。**READ CONTAINER x**に読み取りトランザクションを追加して、ファイル名を`wavedata`に変更します(手順は**To File**と同じです)。次に、AddNoiseオブジェクトとWaveform (Time)オブジェクト間のラインを削除して、オブジェクトを図51のように接続します。To FileとFrom

File間のシーケンス・ラインにより、データは読み取られる前に確実にファイルに書き込まれます。

7 プログラムを実行します。図51のように表示されます。プログラムを「**usrobj-program.vee**」のファイル名で保存します。

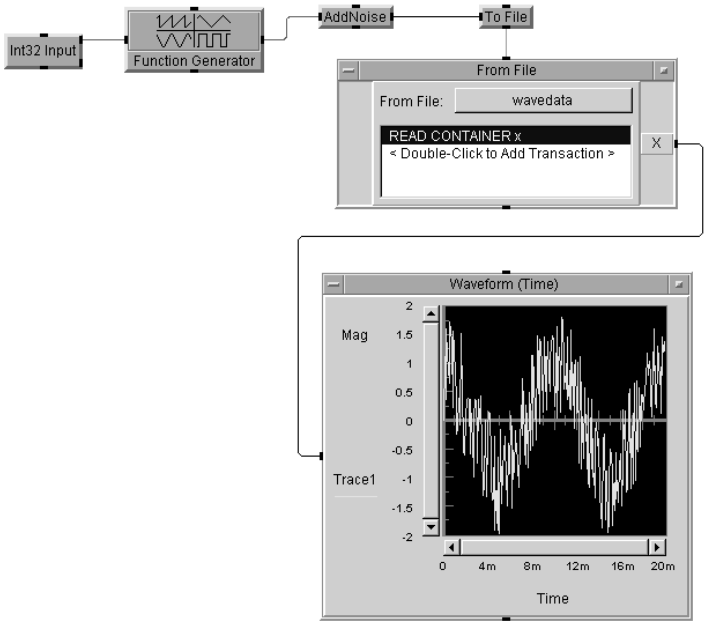


図51　From Fileオブジェクトの追加

例題2-4: パネル・ビュー (オペレータ・インタフェース)の作成方法

プログラムを開発したら、今度は、オペレータ・インタフェースを作成することになります。これを行うには、プログラムのパネル・ビューを作成します。ここでは、66ページの「例題1-2: データ・フローと伝達の表示」で作成したプログラムを使って学習します。

1 プログラム「**simple-program.vee**」を開きます。プログラムは図52のように表示されます。

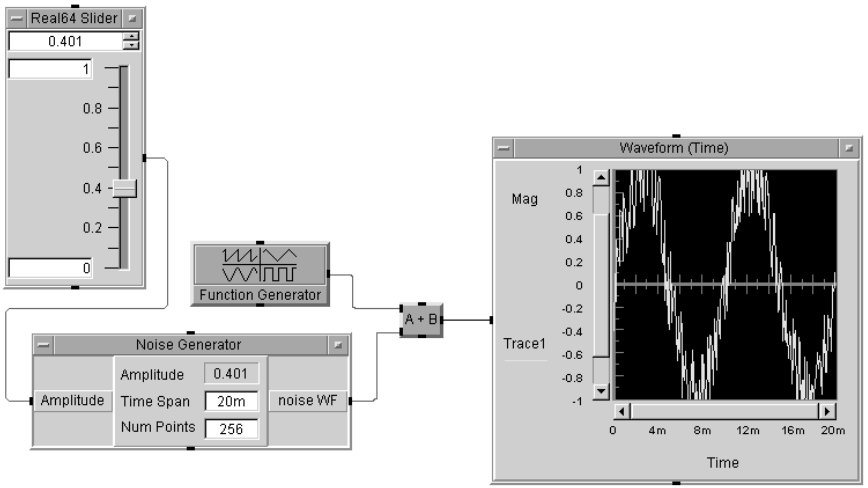


図52　simple-program.vee

2 オペレータ・インタフェースとして動作するパネル・ビューに表示するオブジェクトを選択します。`Ctrl`キーを押しながら、選択するオブジェクトをすべてクリックします(オブジェクトを間違って選択しないよう注意してください)。ここでは、Real64 SliderおよびWaveform (Time)オブジェクトを選択します。選択したオブジェクトには、選択されていることを示す影が表示されます。

3 ツールバーの[Add to Panel]ボタンをクリックして、選択したオブジェクトをパネルに追加します。または**[Edit]** ⇒ **[Add To Panel]**を選択します。パネル・ビューが現れ、パネルに追加した2つのオブジェクトが表示されます。

パネル・ビュー内のオブジェクトのサイズを変更して適切な位置に移動し、図53に示すようなパネルを作成します。

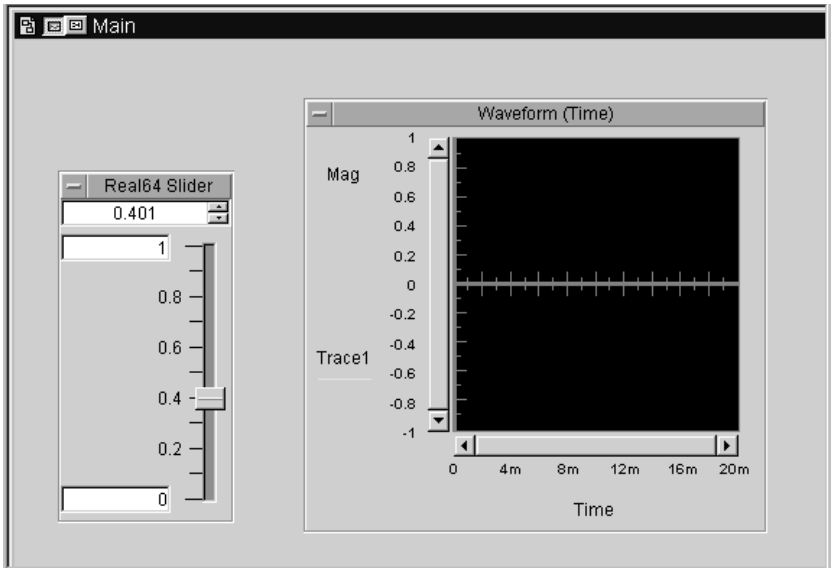


図53　　パネル・ビューの作成例

4 メイン・ウィンドウのタイトル・バーの左上にある[To Detail]ボタンを押して、詳細ビューに切り替えます。[To Panel]ボタンをクリックすると、パネル・ビューに戻ります。

詳細ビューは、プログラムを編集する通常のウィンドウです。パネル・ビューでは、詳細ビューとは独立してオブジェクトの移動、サイズの変更、削除を行うことができます。詳細ビューはプログラムの開発に使用され、パネル・ビューはオペレータ・インタフェースの提供に使用されます。

5 プログラムに**simple-program_with_panel.vee**と名前を付けて保存します。

パネル・ビューでは次のような変更を行うことができます。

- パネルの背景色を変更するには、パネル・ビューのメイン・ウィンドウでオブジェクト・メニューから[Properties]を選択します。**[BackColor]**プロパティを選択し、色を選択します。
- オブジェクトの色またはフォントを変更するには、タイトル・バーをダブルクリックして[Properties]ダイアログ・ボックスを表示します。次に変更する**[Colors]**または**[Fonts]**プロパティをクリックします。
- パネル・ビューのオブジェクトを浮出し表示するには、そのオブジェクトの[Properties]ウィンドウを開き、[Border]プロパティの下にある[Raised]を選択します。
- パネル・ビューの名前を変更するには、メインの[Properties]ウィンドウを開き、[Title]プロパティを変更します。プログラムを実行すると、入力した名前が表示されます。

例題2-5: データの数式処理方法

VEEは、MATLABのデータ/信号処理能力のすべてを含む、広範な組込み数式処理能力を提供します。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

データ型の使用方法

VEEは、テキスト、整数、実数、複素数、座標などを含むさまざまなデータ型をサポートします。すでに、前の例題で**A+B**オブジェクトが2つの波形をどのように合算するかを学びました。加法**(+)**のような算術演算子は、複数のデータ型を処理できるだけでなく、データ型が混在している場合でも処理を行うことができます。

たとえば、次のプログラムを作成するには、メイン・ウィンドウをクリアし、メイン・ウィンドウに以下に示すオブジェクトを配置し、手順に従ってそれらのオブジェクトを接続します。

1. **[File]** ⇒ **[New]**を選択して作業領域をクリアします。
2. **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**を選択して、Real64 Constantオブジェクトを追加します。
3. **[Data]** ⇒ **[Constant]** ⇒ **[Complex]**を選択して、Complex Constantオブジェクトを追加します。

4 **A+B**オブジェクトを追加します。**[Device]** ⇒ **[Function & Object Browser]**を選択して**[Function & Object Browser]**を表示します。次に、[Type]には[Operators]を、[Category]には[Arithmetic]を、[Operators]には**[+]**を選択します。**[Create Formula]**をクリックしてオブジェクトを作成します。

5 **[Display]** ⇒ **[AlphaNumeric]**を選択して、AlphaNumericオブジェクトを追加します。図54に示すようにオブジェクトを接続します。Real64 Constantオブジェクトのデータ入力フィールドに数値「`1.53`」を入力します。また`Complex`オブジェクトに複素数 「`(2.1)`」を入力します。プログラムを実行すると、図54のような結果になります。

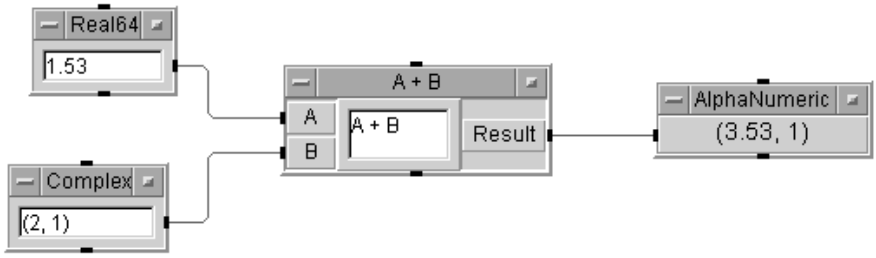


図54　　データ型の使用方法

VEEは、**A+B**オブジェクトで、自動的に必要なデータ変換を行ったうえで加法を実行します。実数**1.53**は複素数**(1.53,0)**に変換され、複素数**(2,1)**に加算されます。結果の複素数**(3.53,1)**は、AlphaNumericオブジェクトに表示されます。

注 記

通常、VEEは、すべてのデータ型変換を自動的に処理します。詳細は、VEEメニュー・バーから[Help] ⇒ [Contents and Index]を選択し、オンライン・ヘルプを参照してください。「使用法」、「以下の項目についての説明」、「リファレンス」のいずれかを選択します。

データの種類の使用方法

VEEは、スカラや配列など、さまざまなデータの種類をサポートします。VEEのオブジェクトは多くのプログラミング言語とは異なり、1つの要素のみに対してではなく、配列全体に対して処理を実行できます。

次のプログラムは、10要素から成る1次元配列を作成し、10個の値の中央値を計算して表示します。

1 **[File]** ⇒ **[New]**を選択して作業領域をクリアします。

2 **[Flow]** ⇒ **[Repeat]** ⇒ **[For Range]**を選択して、For Rangeオブジェクトを追加します。

3 **[Data]** ⇒ **[Sliding Collector]**を選択して、Sliding Collectorオブジェクトを追加します。

4 **median(x)**オブジェクトを追加します。**[Device]** ⇒ **[Function & Object Browser]**を選択します。次に、[Type]には[Built-in Functions]を、[Category]には[Probability & Statistics]を、[Functions]には[median]を選択して、**[Create Formula]**をクリックします。

   [Function & Object Browser]は、ツールバーの**[*fx*]**ボタンをクリックすると表示できます。

5 **[Display]** ⇒ **[AlphaNumeric]**を選択して、AlphaNumericオブジェクトを追加します。図55に示すようにオブジェクトを接続します。プログラムを実行します。オブジェクトの入力値を変更していなければ、結果が図55のように表示されます。

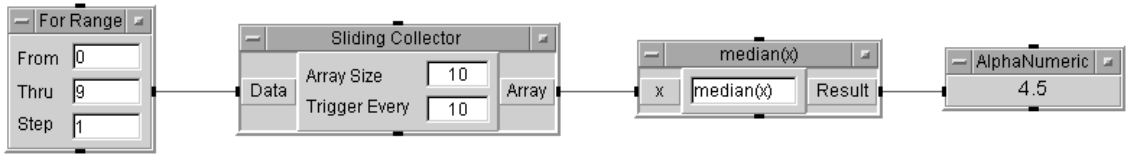


図55　　データ・オブジェクトの接続

Formulaオブジェクトの使用

VEEが提供する算術演算子と関数については、オンライン・ヘルプの「Reference」で解説されています。**[Help]** ⇒ **[Contents and Index]**を選択します。次に、「リファレンス」を選択して目的の項目を表示します。

定義済みの演算子および関数オブジェクトを使用するには、**[Device]** ⇒ **[Function & Object Browser]**(またはツールバーの**[*fx*]**ボタン)を選択します。[Function & Object Browser]の[Type:]、[Category:]、[Functions:]の3種類のリストでエントリをクリックすることによって、演算子/関数を選択します。**[Create Formula]**をクリックしてオブジェクトを作成します。

定義済みの演算子および関数のほかに、[Device]メニューにあるFormulaオブジェクト内に任意の有効なVEE演算式を作成できます。このセクションでは、Formulaオブジェクトを使用してプログラムを作成します。初めにメイン・ウィンドウをクリアしてから、次のステップに進んでください。

1. メイン・ウィンドウにFunction Generatorオブジェクトを追加し、オブジェクトを修正して**100**Hzの正弦波を作成します。**[Device]** ⇒ **[Virtual Source]** ⇒ **[Function Generator]**を選択します。
2. **[Device]** ⇒ **[Formula]**を選択して、Formula オブジェクトをメイン・ウィンドウに追加します。オブジェクトの入力端子領域にマウス・ポインタを置いて`Ctrl+A`キーを押し、2番目の入力端子(**B**)をオブジェクトに追加します。
3. 入力フィールドに演算式「`abs(A)+B`」を入力します。
4. **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**を選択して、Real64 Constantオブジェクトをメイン・ウィンドウに追加します。値「`0.5`」を入力します。
5. **[Display]** ⇒ **[Waveform (Time)]**を選択して、[y-axis]スケールを**[-2]**から**[2]**までに設定します。[Automatic Scaling]を[Off]に設定します。これらのパラメータを設定するダイアログ・ボックスを表示するには、**[Mag]**をクリックします。
6. 図56に示すようにオブジェクトを接続します。プログラムを実行します。

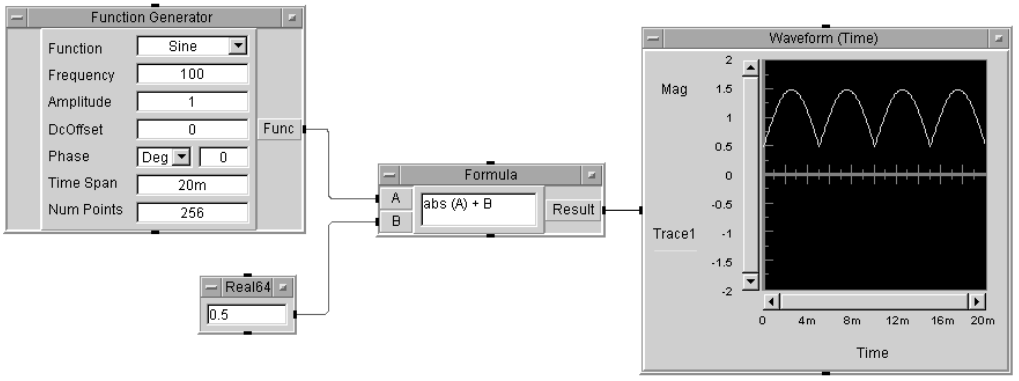


図56　　Formulaオブジェクトのプログラムを作成する

プログラムを実行すると、Formulaオブジェクトは波形の入力値**A**と実数**B**を受け取り、**A**の絶対値に**B**を加算します。その結果、式**abs(A)+B**は正弦波を「整流」して「DCオフセット」を追加します。**A+B**および**abs(x)**オブジェクトを使用しても同じ結果を得ることができますが、Formulaオブジェクトの式を読む方が簡単で、スペースを節約できます。

Formulaオブジェクトの入力/出力端子をダブルクリックしてみてください。入力端子**B**の実数のスカラが、入力端子**A**の波形データ(1次元配列)の各要素に追加されます。そして、結果として得られた波形が**Result**端子に出力されます。

注 記

VEEの演算機能をより高めるためにMATLAB Scriptが統合されており、さらに数多くの算術関数が使用可能になりました。これらの関数は[Function & Object Browser]で表示できます。MATLAB関数の使用については、第4章「テスト・データの解析と表示」の188ページの「Agilent VEEにおけるMATLAB Scriptの使用」を参照してください。

オンライン・ヘルプの使用方法

ここまで簡単なプログラムをいくつか作成しました。VEEについてさらに学ぶには、次の方法があります。

1. まず、**[Help]** ⇒ **[Welcome]**メニューにあるマルチメディア・チュートリアルを実行します。チュートリアルでは、VEEの主要機能に関するデモンストレーションを見て、短時間でVEEについて学習できます。チュートリアルは、VEEプログラムの構築と実行を画面でデモンストレーションし、表示される内容について解説します。チュートリアルでは、VEEを効果的に使用するためのポイントとなる概念についても紹介しています。
2. VEEに慣れてきたら、オブジェクト・メニューのヘルプでさらに詳しい情報を探します。オブジェクトの動作について理解できるまで、オブジェクトを試してみることができます。オブジェクトについてさらに知識が必要になったら、オブジェクト・メニューでオブジェクトに固有の情報を入手することができます。最初にその情報を参照してください。
3. ヘルプの目次、索引、検索機能を使用するには、メイン・ウィンドウのVEEメニュー・バーから[Help]を開きます。

注 記　**メイン・ウィンドウのヘルプ機能を開いてヘルプの目次リストを表示する方法については、25ページの「ヘルプの利用」を参照してください。**

ヘルプ機能の使用方法

オンライン・ヘルプでは次のトピックに関する情報を提供します。

- すべてのメニュー項目、および主なメニューのショートカット
- 計測器ドライバ情報
- 使用頻度の高い処理とプログラム例
- VEE用語の定義
- ヘルプ機能の使用方法
- VEEのバージョン

ヘルプにはブラウズ機能のほか、キーワード索引、関連するトピックへのハイパーリンク、検索などの機能があります。VEEには、プログラム開発時に使用できるヘルプ機能が数多く装備されています。

注 記

VEEには、Line Probeのような、プログラムの開発およびデバッグに役立つ機能もあります。詳細は、100ページの「Agilent VEEにおけるプログラムのデバッグ方法」を参照してください。

オブジェクトについてのヘルプの表示

オブジェクトに関するヘルプを表示するには、オブジェクト・メニュー・ボタンをクリックしてヘルプを選択します。

- **[Flow]** ⇒ **[Repeat]** ⇒ **[For Count]**を選択してFor Countオブジェクトを作成します。オブジェクト・メニュー・ボタンをクリックして[Help]を選択します。For Countオブジェクトについてのヘルプのトピックが表示されます。
- **[Device]** ⇒ **[Formula]**を選択してFormulaオブジェクトを作成します。オブジェクト・メニュー・ボタンをクリックして[Help]を選択します。Formulaオブジェクトに表示される特定の式についてのヘルプのトピックが表示されます。
- **[Device]** ⇒ **[Function & Object Browser]**を選択します。選択項目を任意の組合わせで選択して**[Help]**をクリックします。選択した特定のオブジェクトについてのヘルプのトピックが表示されます。

オブジェクトのメニューの位置を探す

オブジェクトのメニュー内での位置を探して、そのオブジェクトについての情報を表示するには、**[Help]** ⇒ **[Contents and Index]**を選択します。次に、**[Index]**タブをクリックしてオブジェクト名を入力し、**[Display]**をクリックします。

たとえば、**[Help]** ⇒ **[Contents and Index]**を選択して、**[Index]**タブをクリックし、「`Collector`」と入力します。**[Display]**をクリックすると、Collectorオブジェクトについてのヘルプのトピックが表示されます。

ヘルプ機能使用のための追加の練習問題

- オブジェクトを削除するためのショートカットを調べる

  [Help] ⇒ **[Contents and Index]** ⇒ **[HowDol...]** ⇒
  [Use the Keyboard Shortcuts] ⇒ **[Editing Programs]** ⇒
  [To Cut an Object or Text]を選択します。

- 「端子」という言葉を調べる

  [Help] ⇒ **[Contents and Index]** ⇒ **[Reference]** ⇒ **[Glossary]** ⇒ **[Terminal]**を選択します。

- VEEのバージョンを調べる

  [Help] ⇒ **[About VEE Pro]**を選択します。

- このバージョンのVEEの新機能を知る

  [Help] ⇒ **[Contents and Index]** ⇒ **[What's New in Agilent VEE Pro]**を選択します。

Agilent VEEにおけるプログラムのデバッグ方法

ここでは、90ページの「例題2-4: パネル・ビュー (オペレータ・インタフェース)の作成方法」で作成したプログラムを使って学習します。**[File]** ⇒ **[Open]**を選択し、**[simple-program_with_panel.vee]**を強調表示して、**[OK]**をクリックします。

VEEは、プログラムの開発時や実行時にエラー・メッセージを表示します。次のような警告、エラー、情報メッセージが表示されます。

- プログラム実行時に、タイトルが黄色の[Caution]ボックスが表示される場合があります。
- プログラム実行時に、タイトルが赤い[Error]ボックスが表示される場合があります。
- プログラム作成中に、**Int16 Constant**に範囲を超える値**33000**を入力するなどの間違いがあると、タイトル・バーが濃紺の**[Error(エラー)]**メッセージ・ボックスが表示されます。
- また、ステータス・バーにエラーや警告についての情報が表示されます。ステータス・バーはVEEウィンドウの最下部にあります。

データ・フローの表示

1 図57に示すように、ツールバーの中央にある[Show Data Flow]ボタンをクリックします。または、**[Debug]** ⇒ **[Show Data Flow]**を選択します。

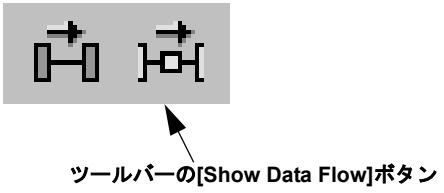


図57　　データ・フローの表示

表示をオフに切り替えるには、ボタンを再度クリックします。プログラムを実行すると、小さな四角形がデータのラインに沿って移動して、データの流れを示します。

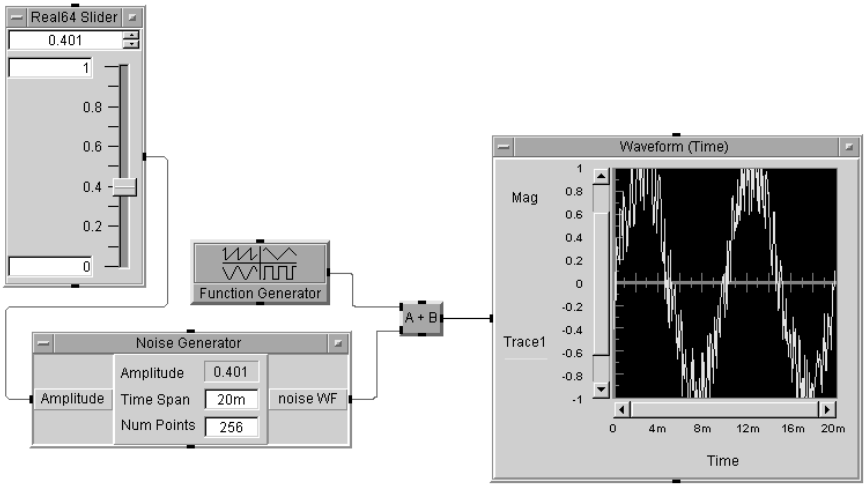
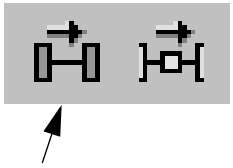


図58　simple-program.veeにおけるデータ・フロー

たとえば図58では、データがReal64 SliderからNoise Generatorへ移動します。Noise GeneratorとFunction Generatorからの出力は**A+B**オブジェクトに入力され、結果がWaveform (Time)の画面に表示されます。

プログラム実行フローの表示

1 図59のように、ツールバーにある[Show Execution Flow]ボタンをクリックします。または、**[Debug]** ⇒ **[Show Execution Flow]**を選択します。



ツールバーの[Show Execution Flow]ボタン

図59　　プログラム実行フローの表示

プログラムを実行すると、動作中のオブジェクトは色付きの輪郭線で囲まれます。

[Data Flow]および[Execution Flow]を使用するとプログラムがどのように動作するかを確認できます。これらをオフにすると、プログラムの実行速度が速くなります。またこれらの機能をブレークポイントのようなデバッグ・ツールと結合すると、VEEプログラムがどのように動作し、エラーの可能性がどこにあるかを確認するのに役立ちます。

ライン上のデータの表示

プログラム内の異なる地点でデータをチェックすると、プログラムのデバッグをすばやく効果的に行うことができます。Line Probeは、指定したライン上のデータを表示します。

詳細ビューで、データ・ライン上にマウス・ポインタを置きます。カーソルが虫めがねの形になります。ラインとその接続が強調表示され、ライン上のデータ値を示すボックスが表示されます。虫めがねのカーソルをクリックするか、**[Debug]** ⇒ **[Line Probe]**を選択しラインをクリックすると、データ・ラインについてのさらに詳しい情報を示すダイアログ・ボックスが表示されます。

たとえば、図60はVEEプログラムの一部を表していますが、アイコン化した**Function Generator**からの出力値が表示されています。出力値は、**Function Generator**が256の点を持つ波形配列を生成していることを示しています。

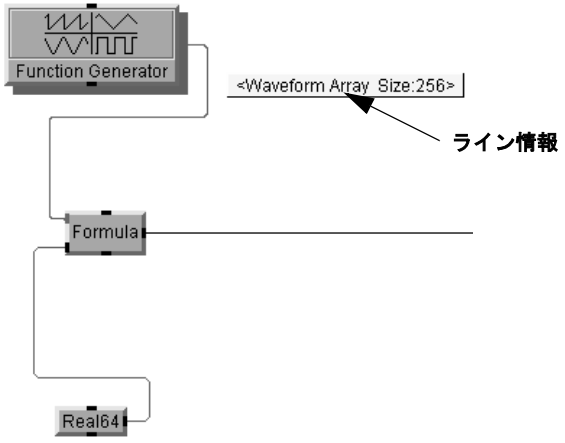


図60　　出力ピン上の値を表示する

データ・ラインをクリックすると、ライン上のデータについてのすべての情報がダイアログ・ボックスに表示されます。たとえば、図61は、Function Generatorの出力ピンをクリックしたときに表示されるダイアログ・ボックスを示しています。

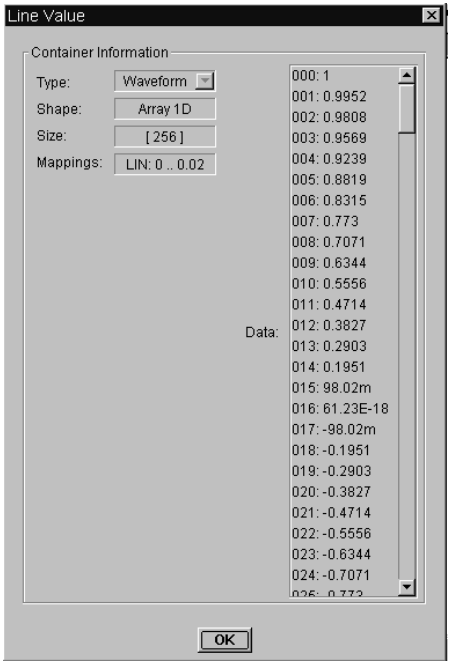


図61　　ライン情報の表示

端子を調べる

端子を調べるには、46ページの「ピンと端子について」で説明したように、オープン・ビューで端子をダブルクリックします。オブジェクトがアイコン化されている場合は、マウス・ポインタを端子上に置くと、端子の名前が自動的にポップアップ表示されます。

デバッグのためのAlphanumeric表示オブジェクトの使用

プログラム内の異なる地点にAlphanumericまたはLogging Alphanumeric表示オブジェクトを追加すると、データの流れを追跡できます。プログラムが正しく実行されたら、これらのオブジェクトは削除します。AlphaNumericは、単一のデータ・コンテナ(スカラ値、**1次元配列**または**2**

次元配列)を表示します。Logging AlphaNumeric(**スカラ**または**1次元配列**)は、連続した入力値を値の履歴として表示します。また、Counterを使用してオブジェクトの実行回数を表示することもできます。

ブレークポイントの使用

ブレークポイントは、特定のオブジェクトを実行する前にプログラムをいったん休止します。プログラム内にブレークポイントを設定してデータを調べることができます。あるオブジェクトにブレークポイントを設定した場合、そのオブジェクトはオレンジ色の輪郭線で強調表示されます。プログラムを実行すると、そのオブジェクトを実行する前に、プログラムが休止します。

1 1つのオブジェクトにブレークポイントを設定します。オブジェクトのタイトル・バーをダブルクリックして[Properties]ダイアログ・ボックスを表示します。[Breakpoint Enabled]を選択して**[OK]**をクリックします。次に、**[Debug]** ⇒ **[Activate Breakpoints]**を選択します。プログラムを実行します。ブレークポイントを設定したオブジェクトの手前でプログラムが休止します。

2 ほかの複数のオブジェクトに追加のブレークポイントを設定します。オブジェクトを選択します。それには、`Ctrl`キーを押しながら各オブジェクトをクリックします。図62に示すように、ツールバーの**[Toggle Breakpoint(s)]**ボタンをクリックします。または`Ctrl+B`キーを押します。再びプログラムを実行すると、ブレークポイントを設定した最初のオブジェクトの手前でプログラムが休止します。

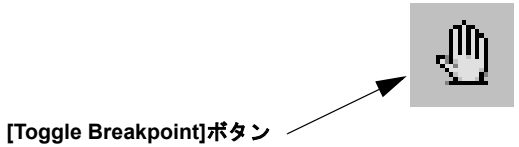


図62　　ブレークポイントの設定

3 プログラムを再開すると、プログラムが続行され、次にブレークポイントが設定されているオブジェクトの手前で休止します。図63に示すように、ツールバーの**[Resume]**ボタンをクリックします。または[Debug]メニューにある[Resume]を選択します。

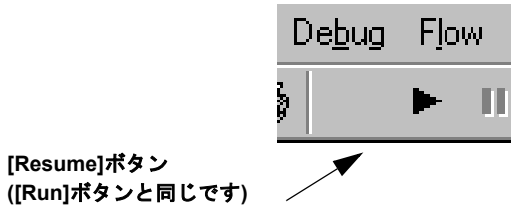


図63　　プログラムを再開する([Run]ボタンと同じ)

4 今度は、ブレークポイントをプログラムからクリアします。ブレークポイントが設定されているオブジェクトを選択します。図64に示すように、ツールバーの**[Toggle Breakpoint(s)]**ボタンをクリックします。または、**[Debug]** ⇒ **[Clear All Breakpoints]**を選択します。

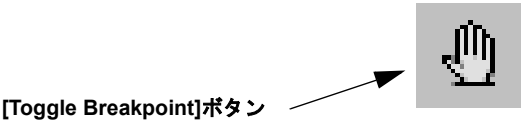


図64　　ブレークポイントのクリア

5 プログラムを休止または停止するには、図65に示すツールバーの**[Pause]**ボタンまたは**[Stop]**ボタンをクリックします。ボタンは[Debug]メニューにもあります。

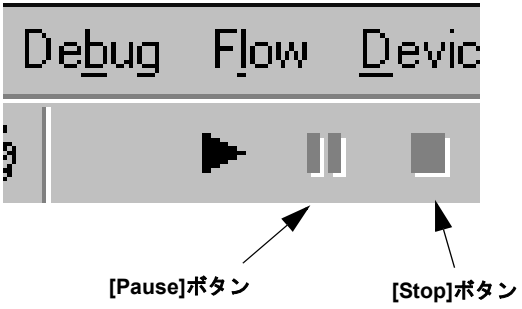


図65　　プログラムの休止または停止

エラーの解決

プログラムの実行時にエラー・メッセージが表示された場合、VEEは、エラーが発見されたオブジェクトを自動的に赤い輪郭線で囲みます。

エラーを修正すると輪郭線は消えます。**[Stop]**ボタンをクリックして赤い輪郭線を先に消去してから、エラーを修正することもできます。**[Stop]**ボタンをクリックした場合は、**[View]** ⇒ **[Last Error]**を選択すると、プログラムを再開する前にエラーを再度表示できます。

[Go To]ボタンを使用してエラーの位置を知る

図66は、実行時エラー・メッセージの例を示しています。このプログラムを実行したところ、VEEは実行時エラーを表示して、UserObjectのAddNoiseの周りに赤い輪郭線を示しました。[Go To]ボタンを押すと、VEEはUserObject AddNoiseを開いて、**A + B**オブジェクトの周りに赤い輪郭線を示します。このオブジェクトでは、入力ピン「**A**」の接続が失われています。大きなプログラムでは、Go To機能を使用すると、エラーの位置をすばやく見つけることができます。

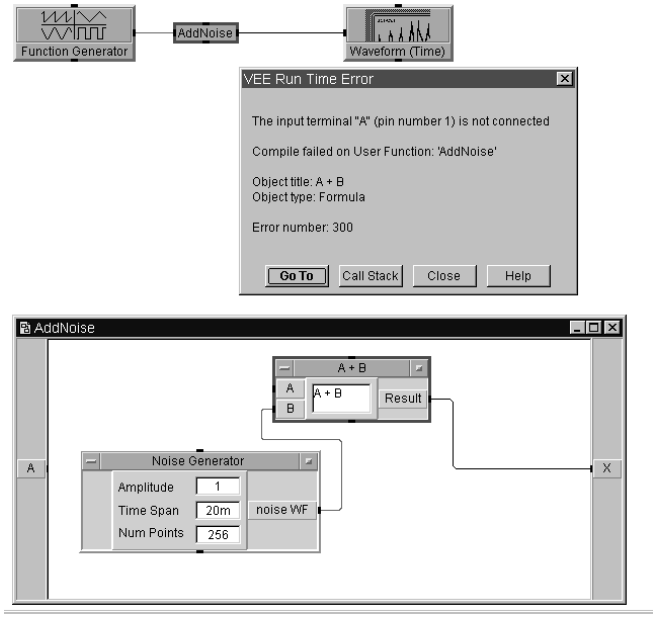


図66　　実行時エラー・メッセージで[Go To]ボタンを使用した例

Call Stackの使用方法

エラーがメイン・プログラム内にある場合は、簡単に見つけることができます。大きなプログラムではCall Stackを使用すると、いくつもの階層の下にあるエラーをよりすばやく探し出すことができます。

1. ツールバーの**[Run]**ボタンの隣にある**[Pause]**ボタンを押します。
2. エラー・ダイアログ・ボックスの**[Call Stack]**ボタンを押します。または**[View]** ⇒ **[Call Stack]**を選択します。Call Stackには、プログラムの実行処理の階層がリストされます。

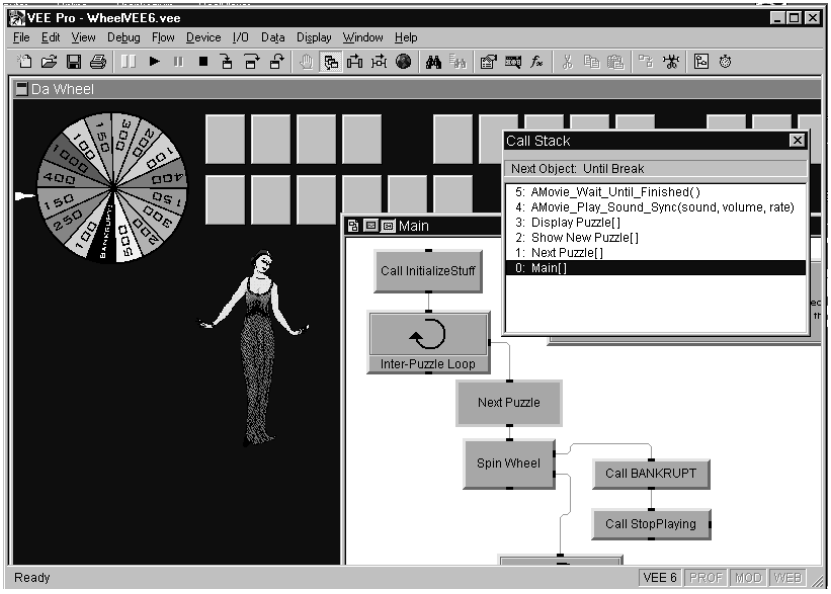


図67　　Wheel.exeでのCall Stackの使用

Call Stackには、プログラムの実行処理の階層が表示されます。図67 は、VEEに付属のサンプル・プログラムである、**Examples/Games**内の**Wheel.exe**プログラムを示しています。図67では、プログラムは現在AMovie_Wait_Until_Finished( )ユーザ関数を実行しています。このユーザ関数は、AMovie_Play_Sound_Syncによって呼び出されており、AMovie_Play_Sound_Syncは元をたどるとメイン・ウィンドウのNext_Puzzleによって呼び出されています。Call Stackのリスト内の項目をダブルクリックすると、VEEがその関数を見つけて表示します。

オブジェクト内部のイベントの順番を追跡する

図68は、オブジェクト内部のイベントの順番を示しています。

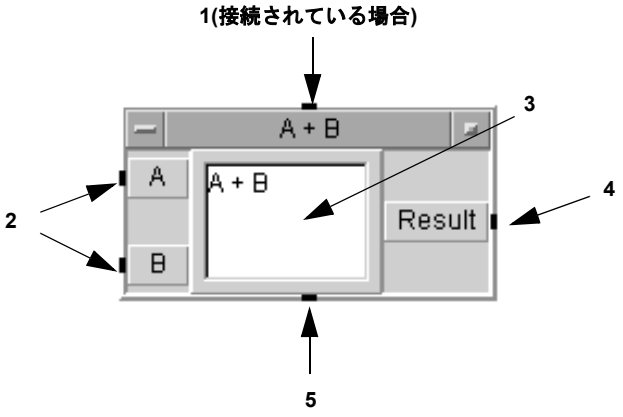


図68　　オブジェクト内部のイベントの順番

図68では、ピンは次のように動作します。

表9　　ピンの動作シーケンス

1	シーケンス入力ピンが接続されている場合、オブジェクトは、実行メッセージを受け取るまで動作しません(VEE用語集の「ping」参照)。ただし、シーケンス入力ピンの接続は、必須では**ありません**。
2	オブジェクトは、すべてのデータ入力ピンがデータを受け取って初めて動作します。ほとんどのオブジェクトにデータ入力/出力ピンを追加できます。オブジェクト・メニューの**[Add/Delete Terminal]**メニューをクリックすると、追加できるピンを確認できます。
3	オブジェクト自身の処理を実行します。この場合は、**A**が**B**に加算され、結果が出力ピンへ渡されます。
4	データ出力ピンがアクティブになります。オブジェクトは、次のオブジェクトからデータを受け取ったという信号を受け取って初めて、処理を完了します。したがって、データ出力ピンに接続されているすべてのオブジェクトがデータを受け取るまでは、このオブジェクトのシーケンス出力ピンはアクティブになりません。
5	シーケンス出力ピンがアクティブになります。

このシーケンス・ピンの動作には例外があります。

- エラー出力ピンを追加してオブジェクト内のエラーを捕捉できます。エラー出力ピンは、標準的なオブジェクトの動作を無視します。オブジェクトの動作時にエラーが発生した場合、エラー・ピンはメッセージを出力し、データ出力ピンはアクティブになりません。
- いくつかのオブジェクトにはコントロール入力ピンを追加できます。コントロール・ピンはオブジェクトに即座に動作を起こさせる場合があります。たとえば、Waveform (Time)にあるTitle表示やAutoscale表示のようなオブジェクトのサブ機能は、コントロール・ピンで動作させることができます。VEEプログラムでは、オブジェクトへのコントロール・ラインの接続は破線で表示されます。

  たとえば、図69は、波形表示にカスタム・タイトルを設定するコントロール・ラインを示しています。オブジェクトでこの操作を行う際、コントロール・ピン上にデータは必要ありません。オブジェクトが実行されるのではなく、タイトル設定のような操作が実行されるだけです。**[Show Data Flow]**をクリックすると、Titleの入力を制御するコントロール・ラインが始めにデータをどのように伝達するかを確認できます。

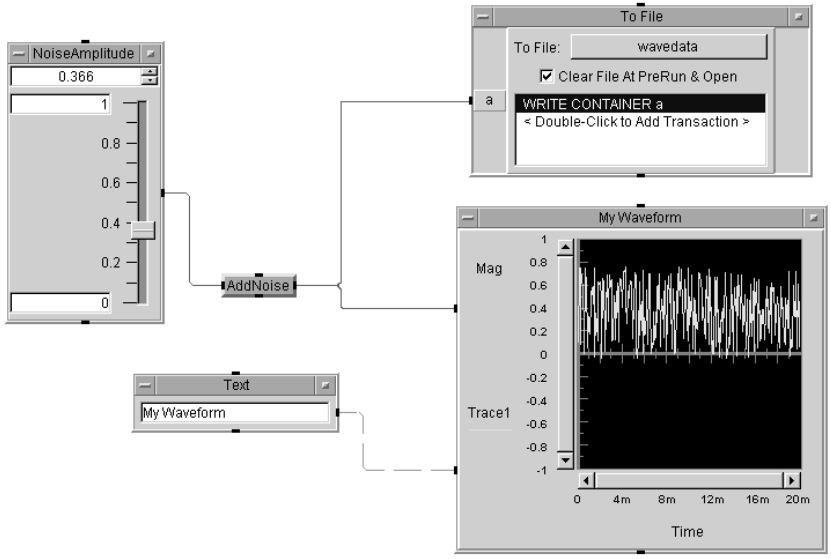


図69　　コントロール・ラインを使用してカスタム・タイトルを実行

プログラム内のオブジェクトの動作の順番を追う

VEEプログラムを実行すると、オブジェクトは次の順番で動作します。

1 Startオブジェクトが最初に動作します。

図70は、2つのスレッドがあるVEEプログラムを示しています。スレッドは、VEEプログラム内の実線で接続された一連のオブジェクトです。Startオブジェクトは、**[Flow]** ⇒ **[Start]**にあり、プログラム内の個々のスレッドを処理するために使われます。プログラムにStartオブジェクトがある場合、Startオブジェクトが最初に動作します。

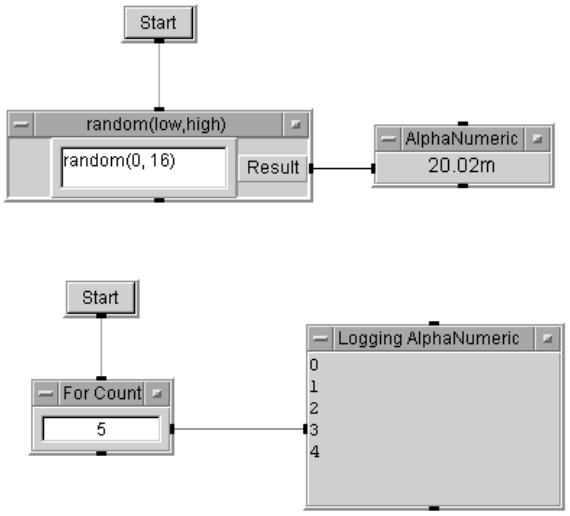


図70　　別々のスレッドを動作させるStartオブジェクト

2 次に、データ入力ピンのないオブジェクトが動作します。**[Data]** ⇒ **[Constant]**にあるオブジェクトはこのカテゴリに入る場合が多いオブジェクトです。

3 入力ピンがあるオブジェクトは、接続されている入力ピンがすべてデータを受け取ったときに初めて動作します。シーケンス入力ピンの接続はオプションであることを思い出してください。

プログラム内のステップ実行

プログラム内のステップ実行は、大変効果的なデバッグ・ツールです。VEEには、オブジェクトの[Step Into]、[Step Over]、[Step Out]の機能があります。

ステップ実行を行うには、図71に示すように、ツールバーの[Step Into]、[Step Over]、[Step Out]のいずれかのボタンをクリックします。

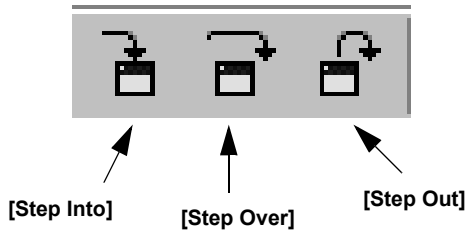


図71　　ツールバー上の[Step Into]、[Step Over]、[Step Out]ボタン

- [Step Into]は、プログラムを実行する際に一度に1つのオブジェクトを実行します。プログラムがUserObjectまたはUserFunctionに出会うと、VEEはUserObjectまたはUserFunctionを詳細ビューで表示し、その中のオブジェクトを1つ1つ実行します。
- [Step Over]および[Step Out]はプログラムを実行する際に一度に１つのオブジェクトを実行しますが、UserObjectまたはUserFunctionを開くことはしません。UserObjectまたはUserFunctionを見つけると、VEEはUserObjectまたはUserFunctionをまるごと実行します。

たとえば、プログラムをステップ実行するには次の手順に従います。

1. **simple-program_with_panel.vee**プログラムを開きます。
2. ツールバーの[Step Into]ボタンをクリックします。
3. [Step Into]をクリックするたびに、次に実行するオブジェクトの周りに色の付いた輪郭線が表示され、プログラム内を次々にたどっていくことができます。

ステップ実行を行うと、VEEは詳細ビューの後ろにパネル・ビューを置いてオブジェクトが実行する順番を示します。メイン・ウィンドウ内では、入力ボックスには入力ピンが接続されていないため、入力ボックスが動作する順番は決まっていません。入力ボックスを特定の順番で動作させたい場合は、シーケンス・ピンを接続することによって順番を制御します。

データは左から右へ流れます。したがって、データ生成プログラム(Generator)が動作する順番は決まっていません。加法(**A+B**)オブジェクトは両方の入力ピンがデータを受け取るまで動作できません。次にWaveform (Time)オブジェクトが動作します。シーケンス・ピンまたは**[Flow]** ⇒ **[Do]**オブジェクトを使用して、プログラム内の任意の場所で動作の順番を指定できます。[Do]オブジェクトについての詳細は、ヘルプを参照してください。

注 記

ステップ機能についての詳細は、オンライン・ヘルプを参照してください。UserFunctionについての詳細は、第9章「Agilent VEE 関数の使用方法」(329ページ)を参照してください。

複雑なプログラム内のオブジェクトの検索

特定のオブジェクトを大きなプログラムの中で検索するには、**[Edit]** ⇒ **[Find]**を選択します。ポップアップ・ダイアログ・ボックスにオブジェクトまたは関数名を入力すると、VEEが、プログラム内にあるそのオブジェクトまたは関数の全インスタンスと場所を表示します。詳細は、356ページの「大型プログラムで関数を検出する方法」を参照してください。

プログラム演習

このセクションのプログラム演習を行うと、VEEの機能についてさらに学ぶことができます。

例題2-6: 乱数の生成

1 プログラムのドキュメントを作成します。

   a **[Display]** ⇒ **[Note Pad]**を選択して、作業領域の上部中央に配置します。編集領域をクリックしてカーソルを置き、そこにあるテンプレートを削除します。次に以下を入力します。

   ```
   This program, Random, generates a real number
   between 0 and 1, then displays the results.
   ```
   (このRandomプログラムは、0から1までの間の実数を生成して結果を表示します)

2 **[Device]** ⇒ **[Function & Object Browser]**を選択します。**[Type]**には**[Built-in]**関数を、**[Category]**には**[All]**を、**[Functions]**には**[random]**を選択します。**[Create Formula]**をクリックします。オブジェクトを作業領域に配置し、クリックして位置を確定します。

3 **[Data]** ⇒ **[Constant]** ⇒ **[Int32]**をクリックして、オブジェクトをrandomの左に配置します。Int32のオブジェクト・メニューを開いて**[Clone]**をクリックし、複製を元のInt32オブジェクトの下に配置します。**0**をダブルクリックしてカーソルを表示し、「1」を入力します。**0**が入力されているConstantオブジェクトをrandomのLow入力ピンに接続します。また**1**が入力されているConstantオブジェクトをHigh入力ピンに接続します。

4 **[Display]** ⇒ **[AlphaNumeric]**を選択して、randomオブジェクトの右に配置します。オブジェクト・メニューを開き、[Help]を選択して、オブジェクトについてさらに理解してください。

5 random オブジェクトの出力ピンを AlphaNumeric の入力ピンに接続します。2つのオブジェクトを接続するデータ・ラインが表示されます。

注 記　**ラインが付いているマウス・ポインタを目的のピンに近づけると、ピンが強調表示されます。再度クリックして接続を確定します。**

注 記　何らかの原因で、接続を確定する前にライン接続の操作を中止する場合は、マウスをダブルクリックしてください。ラインが消えます。

6 ツールバーの[Run]ボタンをクリックします。図72に示すように、乱数が表示されます。

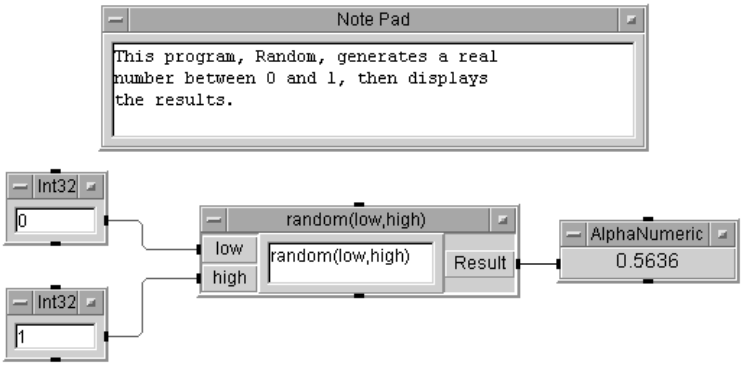


図72　Randomプログラム

7 **[File]** ⇒ **[Save As...]**を選択して、「`Random.VEE`」と入力し、**[OK]**をクリックします。この名前は、次にこのプログラムを開いたときにタイトル・バーのVEEの隣に表示されます。

例題2-7: グローバル変数の設定と表示

このプログラムでは、VEEプログラムを構築する基本的な技術についてさらに練習できます。またグローバル変数について理解できます。Set Variableオブジェクトを使用すると、後でGet Variableオブジェクトを使ってプログラム中に再取込みできる変数を作成できます。VEEのどのよう

なデータ型でも使用できます。この例題では、数値型の**Real64**を使用します。VEEのデータ型についての詳細は、第4章「テスト・データの解析と表示」を参照してください。

1 **[Display]** ⇒ **[Note Pad]**を選択して、オブジェクトを作業領域の上部中央に配置します。編集領域の左上隅をクリックしてカーソルを表示させ、次の情報を入力します。

```
Set and Get a Global Variable prompts the user to
enter a real number. The variable, num, is set to
this real number. Then num is recalled and
displayed.
```

(グローバル変数の設定と表示では、ユーザに実数を入力させます。この実数を変数numに設定します。次にnumを呼び出して表示します)

2 **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**を選択して、オブジェクトを作業領域の左側に配置します。オブジェクト・メニューを開いて、ヘルプの内容を調べます。

3 Real64のオブジェクト・メニューを開いて、[Properties]を選択します。プロンプトのタイトルを`Enter a Real Number:`に変更して**[OK]**をクリックします。

注 記

この練習では、プロンプトのタイトルを変更することによって、Constantオブジェクトの1つを入力ダイアログ・ボックスとして使用しています。これは、ユーザに入力してもらうときの一般的なテクニックです。[Data] ⇒ [Dialog Box] ⇒ [Real64 Input]を使用することもできます。また、タイトル・バーをダブルクリックして[Constant Properties]ダイアログ・ボックスを表示することもできます。

4 **[Data]** ⇒ **[Variable]** ⇒ **[Set Variable]**を選択して、オブジェクトをReal64オブジェクトの右に配置します。[globalA]をダブルクリックして強調表示します。次に「num」を入力します。オブジェクトの名前がSet numに変わります。

このことは、ユーザがReal64オブジェクトに実数を入力することを意味します。ユーザが[Run]ボタンをクリックすると、数値がグローバル変数numに設定されます。

5 Realオブジェクトのデータ出力ピンをSet numオブジェクトのデータ入力ピンに接続します。

6 **[Data]** ⇒ **[Variable]** ⇒ **[Get Variable]**を選択して、オブジェクトをSet numオブジェクトの下に配置します。この変数の名前をnumに変更します。オブジェクトの名前がGet numに変わります。

7 Set numのシーケンス出力ピンをGet numのシーケンス入力ピンに接続します。

注 記

グローバル変数は、使用する前に設定する必要があります。したがって、Get numで再度取り込む前に変数numを確実に設定しておくために、シーケンス・ピンを使用します。

8 **[Display]** ⇒ **[AlphaNumeric]**を選択し、オブジェクトをGet numオブジェクトの右に配置します。

9 Get numのデータ出力ピンをAlphaNumericのデータ入力ピンに接続します。

10 実数を入力してツールバーの[Run]ボタンをクリックします。プログラムは図73のようになります。

11 **[File]** ⇒ **[Save As...]**を選択して、プログラムに**global.vee**と名前を付けて保存します。

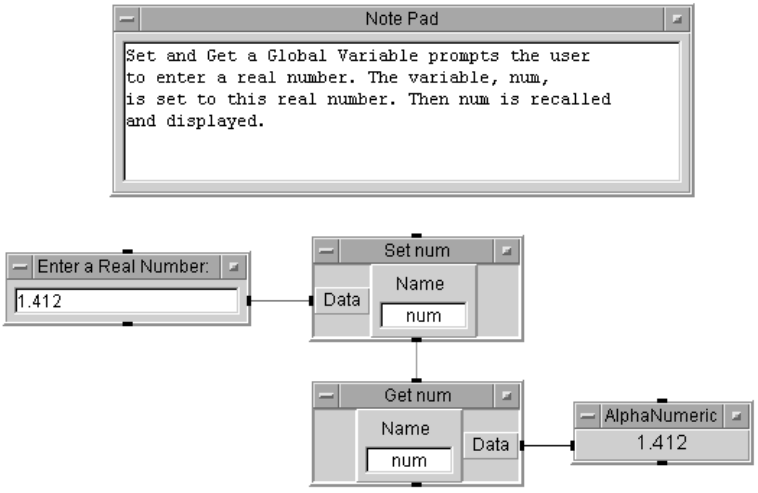


図73　　グローバル変数の設定と表示

Agilent VEEプログラムのドキュメント作成

[File] ⇒ **[Save Documentation...]**コマンドを使用して、プログラムのドキュメントを自動的に生成できます。VEEは、すべてのオブジェクトを基本設定、デフォルト名、ユーザ名、Descriptionで入力したテキスト、「ネスト情報」とともにリストします。たとえば、UserObject内のオブジェクトは、メインのVEE環境から1レベルだけネストされていますが、これらのレベルが数字で示されます。

また、Descriptionを使用して個々のオブジェクトのドキュメントを作成することもできます。この例題では、まず個別オブジェクトのドキュメント作成方法について説明し、次にプログラムのドキュメント生成方法を説明します。

[Description]ダイアログ・ボックスでのオブジェクトのドキュメント作成

すべてのオブジェクトのオブジェクト・メニューにはDescription項目があり、ダイアログ・ボックスにその特定のオブジェクトについてのドキュメントを入力できます。このドキュメント・ファイルで、ドキュメントを画面コピーに関連付けることもできます。このセクションでは、[Description]ダイアログ・ボックスへの入力を行います。

1 **Random.vee**プログラムを開きます。

2 図74に示すように、**[Description]**からテンプレート情報を削除します。

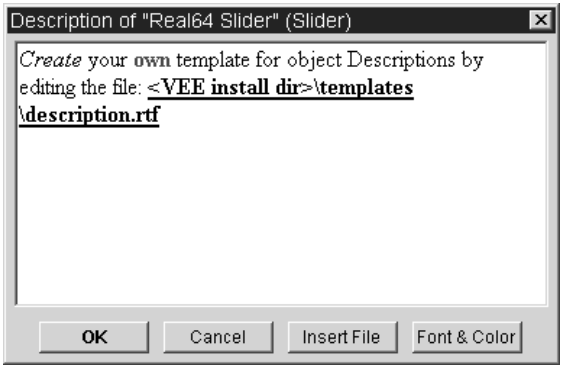


図74　[Description]ダイアログ・ボックス

3 [Description]ボックスは、リッチ・テキスト・フォーマット(RTF)ボックスです。RTFファイルを作成する任意のプログラムからファイルを検索できます。RTFフォーマットによって、テキスト・データの柔軟なプレゼンテーションが可能となります。

4 [Main]オブジェクト・メニューで**[Description]**をクリックします。ダイアログ・ボックスにテキスト「The Random.vee program generates random numbers. (Random.veeプログラムはランダムな数値を生成します。)」を入力します。入力を終えたら**[OK]**をクリックします。

注 記

[Description]ダイアログ・ボックスに入力した内容は、オブジェクト・メニューからアクセスしないと、ユーザには表示されません。このダイアログ・ボックスにはファイルやテンプレートも挿入できます。

ドキュメントの自動生成

プログラム・ドキュメントのファイルを生成するには、次のようにします。

1 **Random.vee**を開きます。**[File]** ⇒ **[Save Documentation...]**をクリックします。拡張子に*.txtを使用してファイル名を入力します(たとえば**Random.txt**)。次に、**[Save]**をクリックします。特に指定しないかぎり、ファイルはPC上のフォルダ「**C:\My Documents\VEE Programs**」に保存されます。

2 ファイルを表示または印刷するには、テキスト・エディタでファイルを開きます。図75、図76、図77は、MS Windows98のメモ帳を使用してドキュメント・ファイルを表示したところを示しています。

図75は、ファイルの開始部分を示しています。ファイルについての情報、更新日時、システムのI/O設定などが記述されています。

Source file: "C:\\My Documents\\VEE Programs\\Random.vee"

File last revised: Mon Jan 03 15:29:02 2003

Date documented: Mon Feb 28 14:43:27 2003

VEE revision: 7.0

Execution mode: VEE 6

Convert Infinity on Binary Read: no

図75　ドキュメント・ファイルの開始部分

M: Main

Device Type : Main

Description :

1. The program, Random, generates a real number between 0 and 1

2. and then displays the results.

Context is secured : off

Trig mode : Degrees

Popup Panel Title Text : Untitled

Show Popup Panel Title : on

Show Popup Panel Border : on

Popup Moveable : on

Popup Panel Title Text Color : Object Title Text

Popup Panel Title Background Color : Object Title

Popup Panel Title Text Font : Object Title Text

Delete Globals at Prerun : on

M.0: Main/Note Pad

Device Type : Note Pad

Note Contents :

1. This program, Random, generates a real

2. number between 0 and 1, then displays

図76　ドキュメント・ファイルの中間部分

図76には、VEEオブジェクトとその設定が記述されています。各オブジェクトの前にある数値は、そのオブジェクトの位置を示します。たとえば、メイン・プログラムにある最初のオブジェクトは「**M1**」としてリストされます。参考までに、図77にこのドキュメント・ファイルの残りの部分を示します。

```
M.2: Main/Int32
 Device Type          : Constant
 Output pin 1         : Int32
 Wait For Event       : off
 Auto execute         : off
 Initialize At Prerun : off
 Initialize at Activate : off
 Constant size fixed  : off
 Password masking     : off
 Indices Enabled      : on
 Int32 Value          : 0
M.4: Main/Int32
 Device Type          : Constant
 Output pin 1         : Int32
 Wait For Event       : off
 Auto execute         : off
 Initialize At Prerun : off
 Initialize at Activate : off
 Constant size fixed  : off
```

図77　　ドキュメント・ファイルの残り部分

注 記

[Save Documentation]コマンドを実行後は、**[File]** ⇒ **[Print Program]**コマンドを実行してオブジェクトに識別番号をつけます。これにより、ドキュメントのテキストとプリンタ出力とを一致させることができます。

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。

- UserObjectを作成する。また、UserObjectのプログラムを構造化して画面上のスペースを節約する。
- ユーザ入力用のポップアップ・ダイアログ・ボックスとスライダ(またはノブ)を作成する。
- データ・ファイルを使ってデータをファイルに保存したり、ファイルからデータをロードする。
- プログラムのパネル・ビューを使ってオペレータ・インタフェースを作成する。
- さまざまなデータ型およびデータの種類を使用する。
- 算術演算子と関数を使用する。
- オンライン・ヘルプを使用する。
- プログラム内のデータ・フローと実行フローを表示する。
- ライン、端子、Alphanumeric表示上のデータを調べてプログラムをデバッグする。
- ブレークポイントを使用する。
- GoToコマンドでエラーを解決する。
- Call Stackを使ってエラーを解決する。
- プログラムを追跡しデバッグするためにStep Into、Step Over、Step Outを使用する。
- 検索機能を使用する。
- [Description] ダイアログ・ボックスでオブジェクトのドキュメントを作成する。
- ドキュメント・ファイルを生成する。

3
簡単な計測器の操作

簡単な計測器の操作

この章の内容

- 計測器の設定方法
- パネル・ドライバの使用方法
- Direct I/Oオブジェクトの使用方法
- PCプラグイン・ボードの制御方法
- VXI*plug&play*ドライバの使用方法

平均所要時間: 1時間

概要

この章では、VEEを使って計測器を制御する方法について学びます。VEEでは、次のような方法で計測器を制御できます。

- **「パネル」ドライバ**は、計測器をコンピュータ画面から制御するための簡単なユーザ・インタフェース(フロント・パネル)を提供します。VEEパネル・ドライバでパラメータを変更すると、計測器の対応するステートが変化します。パネル・ドライバは、Agilent TechnologiesによってVEEと共に提供され、さまざまなメーカから提供されている450以上の計測器をサポートします。
- **Direct I/Oオブジェクト**は、サポートされているさまざまなインタフェース経由でコマンドを伝達したりデータを受け取ることができます。
- **I/Oライブラリ**をインポートしてPCプラグイン・ボードを制御できます。インポートしたライブラリからは、**Call**オブジェクトを使用して関数を呼び出すことができます。これらのライブラリは、通常はDynamic Link Libraries(DLL)として搭載されています。
- **VXI*plug&play*ドライバ**は、C関数を呼び出して計測器を制御するために使用します。Agilent Technologiesおよび計測器をサポートするメーカから提供されています。

この章では、さまざまな状況に応じた計測器の制御方法の基礎知識を解説します。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

パネル・ドライバ

Agilent VEEには、さまざまな計測器メーカ用に450以上のパネル・ドライバが用意されています。パネル・ドライバはVEEプログラム内に表示される画面を使って動作しますが、この画面では対応する物理計測器の設定を制御します。パネル・ドライバは、大変使いやすく、開発時間を大幅に短縮します。図78は、パネル・ドライバの例です。

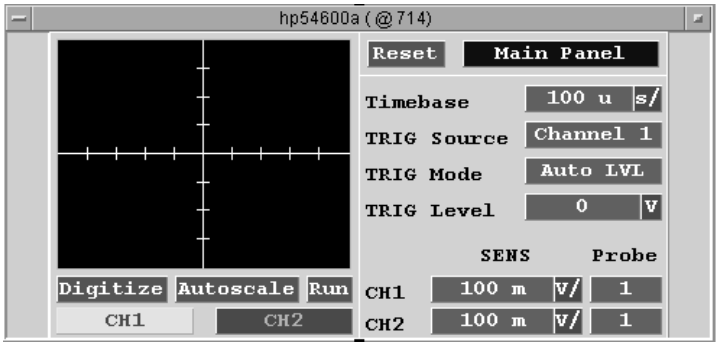


図78　　HP54600Aオシロスコープ・パネル・ドライバ

Direct I/Oオブジェクト

VEEのDirect I/Oオブジェクトを使用すると、標準のインタフェース経由で、すべてのメーカのすべての計測器と通信できます。計測器用のドライバがあるかどうかは関係ありません。Direct I/Oオブジェクトは、計測器にコマンドを伝達したり、計測器からデータを読み取ります。Direct I/Oを使用すると、通常は実行速度が上がります。最良の計測器制御方法としてどの方法を選択すべきかは、ドライバの性能、テスト開発速度の必要性、性能条件によって決まります。図79は、Function Generatorの制御にDirect I/Oを使用している例です。

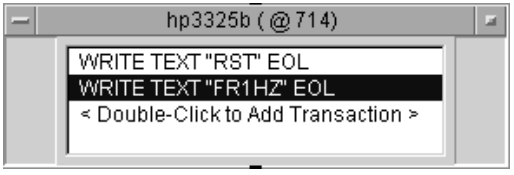


図79　　Function GeneratorオブジェクトのDirect I/O

I/Oライブラリを使用したPCプラグイン・ボード制御

I/Oライブラリは、通常PCプラグイン・ボード用のDynamically Linked Libraries(DLLファイル)として搭載されており、PCプラグイン・ボードのメーカから提供されます。VEEではオブジェクトでライブラリ関数を呼び出すことによってPCプラグイン・ボードを制御できます。図80は、VEEで関数を使用できるようにするImport Libraryオブジェクトの例です。

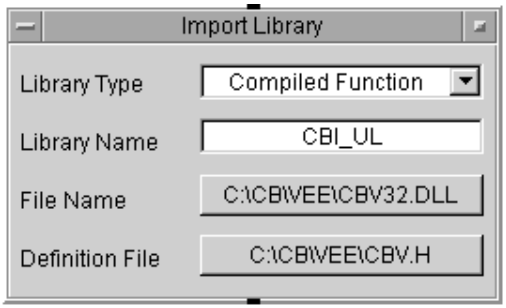


図80　PCプラグイン・ライブラリのインポート

VXI*plug&play*ドライバ

VXI*plug&play*ドライバは、計測器メーカまたはAgilent Technologiesによって提供されます。Agilent Technologiesが提供するVXI*plug&play*ドライバのリストについては、VEE説明書または『*VEE Pro Advanced Techniques*』を参照してください。そのほかのVXI*plug&play*ドライバについては計測器メーカにお問い合せください。VEEでは、VXI*plug&play*ドライバを呼び出すことによって計測器を制御できます。図81は、VEEからVXI*plug&play*ドライバを呼び出している例です

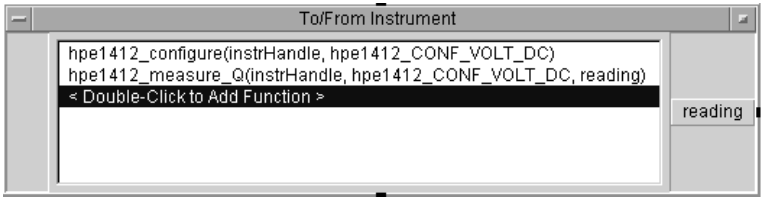


図81　VEEからVXI*plug&play*ドライバの呼び出し

計測器の設定方法

VEEでは、計測器を接続していなくてもプログラムを開発できます。次の例題では、パネル・ドライバを使ってオシロスコープの設定を行います。次に、構成に物理的に計測器を追加します。

例題3-1: 計測器を接続しないで計測器構成を設定する

1 [I/O] ⇒ [Instrument Manager....]を選択します。タイトル・バーをクリック・アンド・ドラッグして、図82に示すようにダイアログ・ボックスを作業領域の左上まで移動します。

図82　[Instrument Manager]ダイアログ・ボックス

注 記

計測器を接続して電源を入れている場合、VEEは自動的に計測器とそのドライバを見つけます。計測器の自動検索および設定についての詳細は、VEEのメイン画面で[Help] ⇒ [Welcome] ⇒ [Tutorials]を選択してオンライン・チュートリアルを参照してください。

特に指定しないかぎり、計測器構成は設定されていません。この例では、[Instrument Manager]のリストに計測器が表示されていないと仮定します。

2 [Instrument Manager]ダイアログ・ボックスで、[My Configuration]が強調表示されていることを確認して、[Instrument]の下にある**[Add...]**をクリックします。図83のような[Instrument Properties]ダイアログ・ボックスが表示されます。

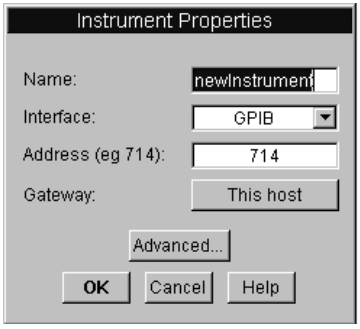


図83　[Instrument Properties]ダイアログ・ボックス

[Instrument Properties]ダイアログ・ボックスには、次の設定項目があります。

表10　[Instrument Properties]の設定項目

設定項目	説明
[Name]	プログラム内での計測器の呼び名を指定します。次の規則に従った名前を選択してください。 • 計測器名の最初の文字はアルファベットでなければなりません。また、計測器名に使用できるのは英数字またはアンダスコアです。 • 計測器名の中に空白を入れることはできません。
[Interface]	インタフェースの種類を指定します。[GPIB]、[Serial]、[GPIO]、[USB]、[LAN]、[VXI]から選択します。
[Address]	インタフェースの論理装置番号(通常GPIBは7)に計測器のローカル・バス・アドレス(0から31までの数値)を加えます。[Address]を[0]のままにした場合は、計測器を接続しないで開発を行っていることを意味します。
[Gateway]	計測器をローカルで制御するかリモート制御するかを指定します。計測器をローカルで制御する場合は、デフォルト値[This host]を使用します。リモート制御を行う場合は`[Gateway]`を指定します。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

3 [Name]を[scope]に変更し、ほかの項目はデフォルト値のままにして、**[Advanced...]**をクリックします。[Advanced Instrument Properties]ダイアログ・ボックスが図84のように表示されます。

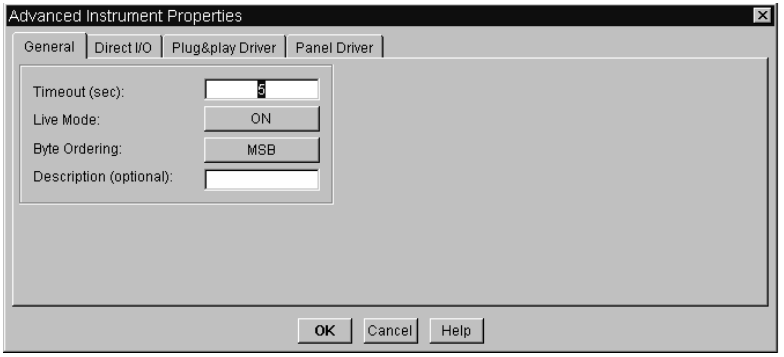


図84　[Advanced Instrument Properties]ダイアログ・ボックス

Generalフォルダでの設定項目は次のとおりです。

表11　Generalフォルダの設定項目

設定項目	説明
[Timeout]	I/Oトランザクションが最大限何秒待機してから、エラー・メッセージを生成するかを指定します。
[Live Mode]	計測器との通信をONにするかどうかを指定します。計測器が設定されていない場合は、**[OFF]**に設定します。VEEのデフォルト設定は**[ON]**です。
[Byte Ordering]	デバイスがバイナリ・データの読み取りや書込みを行う際のバイト順を指定します。このフィールドで、[MSB](最上位バイトから送信)か[LSB](最下位バイトから送信)を指定します。IEEE 488.2準拠デバイスはすべて、[MSB]を指定する必要があります。
[Description]	識別子を入力します。たとえば、タイトル・バーに計測器の番号を入れたい場合は、その番号を入力します。

4 [Live Mode]を[OFF]に切り替えます。次に[Panel Driver]フォルダをクリックします。ダイアログ・ボックスの表示が図85のようになります。

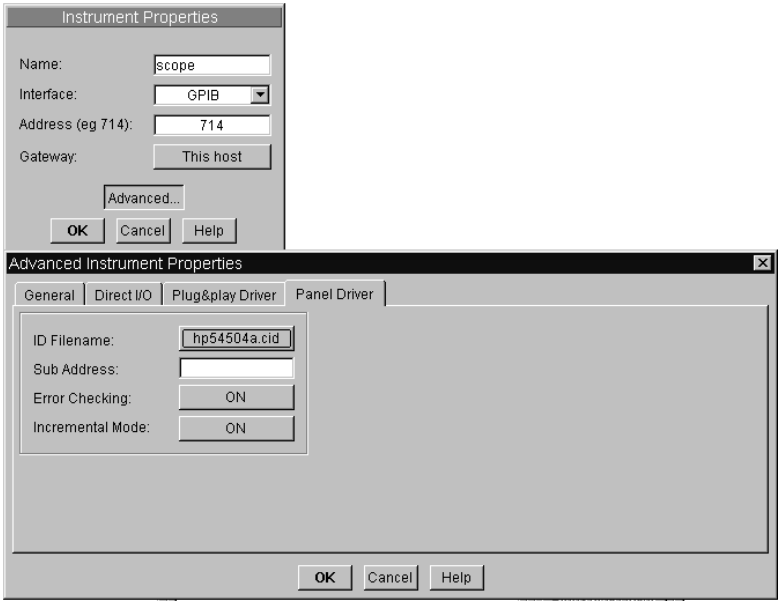


図85　[Panel Driver]フォルダ

5 [ID Filename]の右側にあるフィールドをクリックすると、[Read from what instrument Driver?]という名前のリスト・ボックスが表示されます。このリストには、指定したディレクトリにVEEの改訂版と一緒にロードされた、パネル・ドライバの全ファイルが含まれます。

注 記

この例題を完了するには、VEE CD-ROMからパネル・ドライバをインストールしておく必要があります。*.cidファイルは、コンパイル済みの計測器ドライバ・ファイルを表します。

6 リストを下へスクロールして **[hp54504a.cid]** を強調表示します。次に **[Open]**をクリックします。図85 では、この計測器がすでに選択されています。強調表示したファイルをダブルクリックして選択することもできます。

[Panel Driver]フォルダのそのほかの設定項目は次のとおりです。

表12　[Panel Driver]の構成

[Panel Driver]の設定項目	説明
[Sub Address]	このフィールドは何も入力しないでおきます。**[Sub Address]**は、非VXIカードケージ計測器がプラグイン・モジュールを特定する場合にのみ使用します。
[Error Checking]	デフォルト設定の[ON]のままにします。[Error Checking]は、処理能力を上げたい場合はオフにできますが、その場合はI/Oエラーはチェックされません。
[Incremental Mode]	デフォルト設定の**[ON]**のままにします。設定を変更するたびに、計測器ステート用の計測器コマンド文字列をすべて送信する場合は、**[Incremental Mode]**を[OFF]にします。

7　**[OK]**をクリックすると、[Instrument Properties]ダイアログ・ボックスに戻ります。**[OK]**をクリックします。

図86に示したように、使用できる計測器のリストに、ドライバ・ファイル**hp54504a.cid**を使用してscopeという名前の計測器構成が加わります。実際には接続されていないため、計測器にはバス・アドレスがありません。このモードでプログラムの開発を行い、後で、計測器をコンピュータに接続できるようになった時点でアドレスを追加できます。

フィールド内の入力を終えて次のフィールドに移動するには、`Tab`キーを押します。前のフィールドに移動するには、`Shift+Tab`キーを押します。Enterキーを押すのは、**OK**をクリックするのと同じです。VEEはダイアログ・ボックスを閉じます。

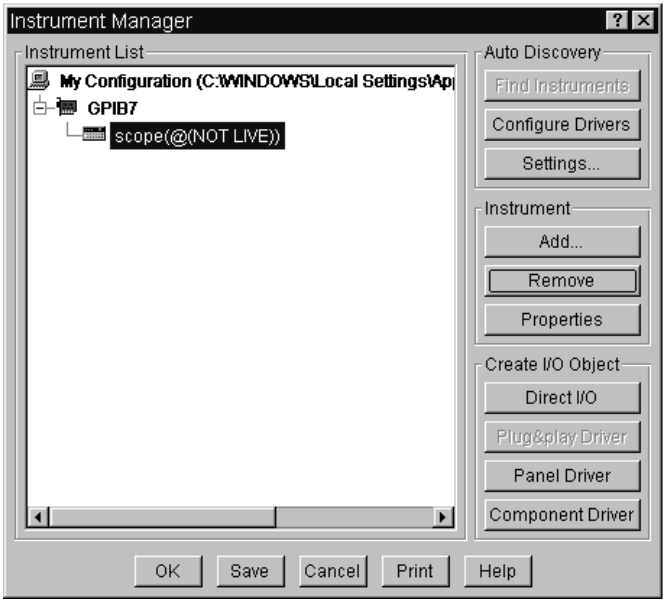


図86　　計測器リストにオシロスコープを追加

8 **[Save]**をクリックして[Instrument Manager]ダイアログ・ボックスを閉じます。[Create I/O Object]にある[Panel Driver]をクリックすると、パネル・ドライバをプログラムにすぐに入れることができます。VEEは構成を自動的に保存します。

これで、HP 54504Aオシロスコープを`scope`という名前で計測器リストに追加できました。実際に計測器を接続していなくても、プログラム作成時にこのドライバを使用できます。

プログラムで使用する計測器の選択

1 **[I/O]** ⇒ **[Instrument Manager....]**を選択します。

2 **[scope(@(NOT LIVE))]**を選択して強調表示にします。次に、[Create I/O Object]の下にある**[Panel Driver]**をクリックします。

注 記

[Instrument Manager]では、設定した計測器の種類に応じて、[Create I/O Object]にあるメニューを使って異なる種類のオブジェクトを作成できます。たとえば、この例題で[Panel Driver]ではなく[Direct I/O]を選択した場合、**scope(@(NOT LIVE))**という名前のDirect I/Oオブジェクトが作成されます。VEEは、コンポーネント・ドライバも提供します。コンポーネント・ドライバは パネル・ドライバが提供する関数のサブセットを使用します。詳細は、「*VEE Pro Advanced Techniques*」を参照してください。

3 [scope]パネルの輪郭線を配置し、クリックして場所を確定します。図87のように表示されます。

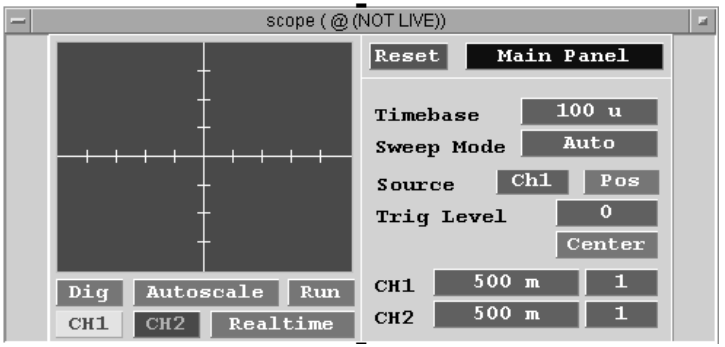


図87　　scope(@(NOT LIVE))の選択

これで、ほかのVEEオブジェクトと同じように、プログラム内でパネル・ドライバを使用できます。

物理計測器の構成への追加

1 **[I/O]** ⇒ **[Instrument Manager...]** を選択して、[scope]を強調表示にします。[Instrument...]の下にある**[Properties]**をクリックします。

2 **[Address]** フィールドをダブルクリックして現在の入力内容を強調表示し、「709」と入力します。709の7は、論理装置番号です。GPIB(HP-IB)の論理装置番号が7でない場合は、7を実際の論理装置番号と置き換えてください。709の9は、オシロスコープのデフォルトのアドレスです。

3 **[Advanced:]**をクリックして[Live Mode]を[ON]に切り替えます。次に**[OK]**をクリックします。**[OK]**をクリックし、[Instrument Properties]ボックスを閉じます。

4 **[Save]**をクリックして変更内容を保存します。

パネル・ドライバの使用

この例題では、例としてHP 3325B Function Generatorを使用します。VEEでは、どのパネル・ドライバを使用する場合でも、基本的な操作は同じです。計測器を直接プログラムする代わりにパネル・ドライバを使用することによって、プログラムの開発/修正時間を節約できます。計測器の設定は、メニューを選択したり、ダイアログ・ボックスのフィールドを編集することによって変更できます。計測器が接続されていて、[Live Mode]が[ON]の場合、変更した設定内容は計測器に登録されます。

プログラムでパネル・ドライバを使用するには、必要な入力/出力端子を追加して、パネル・ドライバをほかのオブジェクトに接続します。計測器に異なるステートを設定するために、プログラム内に同じドライバを複数入れることができます。VEEでは、パネル・ドライバをアイコン化してスペースを節約することも、オープン・ビューにして計測器の設定を表示することもできます。またプログラム実行中に設定を変更できます。

例題3-2: パネル・ドライバの設定

1 **[I/O]** ⇒ **[Instrument Manager...]**を選択します。[My Configuration]を選択して、[Instrument]の下にある**[Add...]**をクリックします。[Instrument Properties]ダイアログ・ボックスが表示されるので、次のように設定を行います。

表13　　パネル・ドライバの設定

設定	説明
[Name]	値を**[fgen]**に変更します。次に、Tabキーを2回押して[Address]フィールドに移動します。
[Address]	値を`[713]`、またはバスに設定するアドレスに変更します。

2 **[Advanced]**をクリックします。[General]フォルダで[Live Mode]を[OFF]に切り替えます。

3 **[Panel Driver]**フォルダをクリックして、[ID Filename:]を**[hp3325b.cid]**に設定します。**[OK]**を2回クリックすると、[Instrument Manager]に戻ります。

4 [Create I/O Object]の下にある**[Panel Driver]**をクリックします。オブジェクトをワークスペースの左側に配置します。計測器が設定されリストに追加されているかぎりは、どのような計測器でも手順は同じです。

注 記　**ここでは、計測器を接続しない前提でプログラムを行っています。計測器を接続している場合は、適切なアドレスを設定してください。**

5 [Function]フィールドの**[Sine]**をクリックしてポップアップ・メニューを表示します。次に、図88のように[Triangle]を選択します。

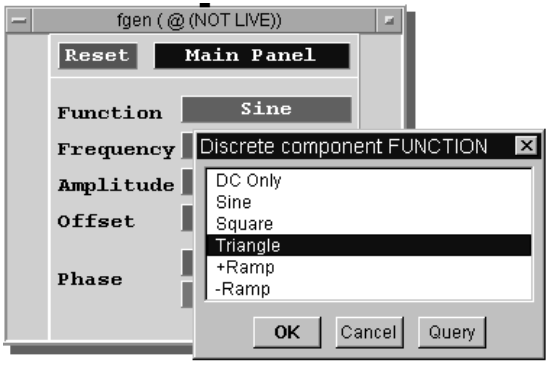


図88　fgenの関数ポップアップ・メニュー

6 **[Frequency]**の右側のフィールドをクリックします。

7 続いて表示される**[Continuous component FREQUENCY]**ダイアログ・ボックスに「100」と入力して**[OK]**をクリックします。[Frequency]の設定が変更されます。

どのようなドライバでも、同じ方法で計測器設定を変更できます。計測器にアドレスを設定し、[Live Mode]を[ON]にした場合、ドライバ・パネルで設定した変更内容はすべて、その計測器に反映されます。

同じドライバの別のパネルへ移動する

ドライバの多くは、ユーザ・インタフェースをわかりやすくするためにパネルを2つ以上持っています。別のパネルへ移動するには、オブジェクトの**[Main Panel]**をクリックしてパネルのメニューを表示します。

1. パネル・ドライバ・オブジェクトで**[Main Panel]**をクリックします。図89のように[Discrete Component MENU]が表示されるので、[Sweep]を選択します。
2. **[OK]**をクリックして[Sweep Panel]を表示します。ほかのパネルも表示して設定内容を確認してください。
3. **[OK]**をクリックして[Main Panel]に戻ります。

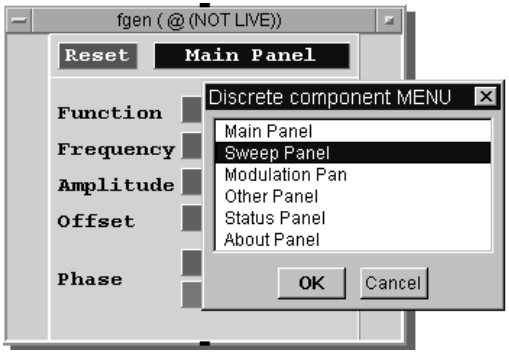


図89　[Discrete Component Menu]の[Sweep Panel]

パネル・ドライバに入力/出力端子を追加する

パネルで直接設定を行うほかに、ドライバにデータ入力/出力端子を追加することによってプログラム内の計測器の設定を制御したり、計測器からデータを読み取ることができます。図90は、入力/出力端子領域を示します。

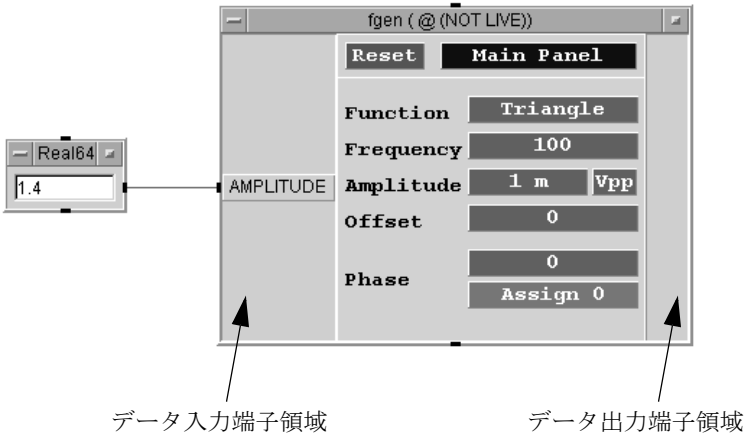


図90　　ドライバ上のデータ入力/出力端子領域

1. マウス・ポインタをFunction Generator計測器パネルのデータ入力端子領域に置き、CTRL+Aキーを押してデータ入力端子を追加します。計測器コンポーネントのリスト・ボックスが表示されます。
2. 表示されたメニューで目的のコンポーネントを選択します。

注 記

オブジェクト・メニューを開いて、[Component] ⇒ [Select Input Component]から[Add Terminal]を選択し、ドライバ上で目的のコンポーネント・フィールドを選択する方法もあります。

データ出力端子を追加するには、マウス・ポインタをデータ出力端子領域に置いて、同じ操作を行います。

データ入力/出力端子の削除

マウス・ポインタを端子の上に置き、CTRL+Dキーを押します。

注 記

オブジェクト・メニューを開いて、オブジェクト・メニューから[Delete Terminal] ⇒ [Input...]を選択し、表示されるメニューで削除する入力端子を選択する方法もあります。

独習課題

HP 3325B Function Generator、または使用できるほかのFunction Generator上でステートを設定します。[Function]の設定を[Square]に変更します。[Amplitude]および[Frequency]に、入力コンポーネントを追加します。振幅と周波数を入力するためのダイアログ・ボックスを作成し、タイトルをオペレータに入力を促すものに修正します。振幅と周波数に異なる値を入力してプログラムを実行し、値の入力後に設定が変更されたかどうかを確認します。計測器を接続している場合は、[Live Mode]を[ON]にすると、設定が変更されます。

Direct I/Oの使用方法

特定の計測器用のドライバがない場合や、処理能力を上げたい場合は、Direct I/Oオブジェクトを使用します。

例題3-3: Direct I/Oの使用方法

この例題では、Direct I/Oを使用してHP 3325B Function Generatorの設定を行います。

1. **[I/O]** ⇒ **[Instrument Manager...]**を選択します。
2. **[fgen(@(NOT LIVE))]**の項目を強調表示し、**[Instrument]** ⇒ **[Properties]**を選択します。
3. **[Advanced]**をクリックします。図91のように、[Direct I/O]フォルダを選択します。使用できるオプションをひととおり見たら、**[OK]**をクリックして[Instrument Properties]に戻ります。次に、**[OK]**を再度クリックして[Instrument Manager]に戻ります。

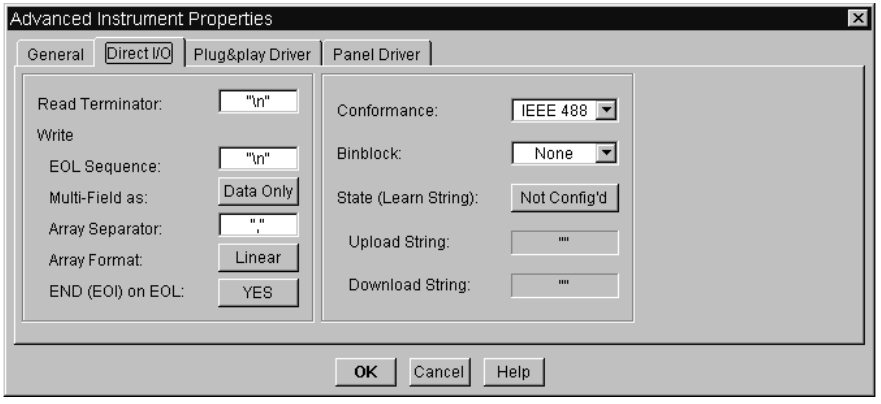


図91　Direct I/O設定フォルダ

注 記

この例では、GPIBインタフェース(IEEE488)を使用します。Serial、GPIO、VXI計測器の設定方法については『*VEE Pro Advanced Techniques*』を参照してください。

4 画面上にオブジェクトを配置するには、**[fgen(@(NOT LIVE))]**が強調表示されていることを確認して**[Create I/O Object]** ⇒ **[Direct I/O]**をクリックします。図92は、作成されたDirect I/Oオブジェクトを示しています。

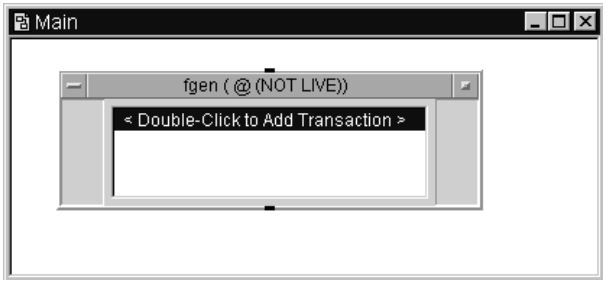


図92　Direct I/Oオブジェクト

プログラム内でDirect I/Oオブジェクトを使用するには、I/Oトランザクションを設定する必要があります。次のセクションでは、テキスト・コマンドの書込み、データの読込み、計測器のステートのアップロード/ダウンロードについて説明します。

計測器に単一のテキスト・コマンドを送信する

計測器に単一のテキスト・コマンドを送信するには、適切な文字列を入力します。GPIB計測器のほとんどは、計測器に送信するコマンドに英数字の文字列を使用します。たとえば、HP3325B Function Generatorに振幅を5ボルトに設定するコマンドを送信するには、コマンド文字列「`AM 5 VO`」を入力します。

この例題では、前のセクションで設定したHP 3325B Function Generatorを使用します。必要に応じて、次へ進む前に147ページの「Direct I/Oの使用方法」に戻り、計測器の設定を行ってください。

1 **fgen(@(NOT LIVE))**オブジェクトでトランザクション・バーをダブルクリックして、図93に示した**[I/O Transaction]**ダイアログ・ボックスを表示します。

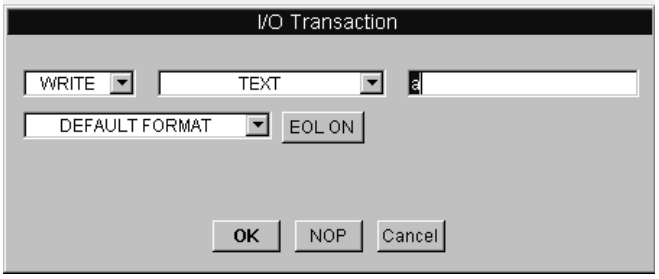


図93　[I/O Transaction]ダイアログ・ボックス

[WRITE]の隣にある下向き矢印をクリックすると、トランザクション・メニューに[READ]、[WRITE]、[EXECUTE]、[WAIT]が表示されます。計測器にデータを書き込むには、デフォルトの設定[WRITE]を使用します。各操作については、オブジェクト・メニューを開いてヘルプを参照してください。

2 デフォルトの設定、[WRITE]、[TEXT]、[DEFAULT FORMAT]、[EOL ON]を使用します。aのラベルが付いた入力フィールドをクリックし、「**"AM 5 VO"**」を(引用符も含めて)入力し、**[OK]**をクリックします。

図94に示したようにトランザクション**[WRITE TEXT "AM 5 VO" EOL]**が表示されます。引用符で囲まれたテキストは、プログラムを実行するとHP3325Bに送信されるコマンドです。

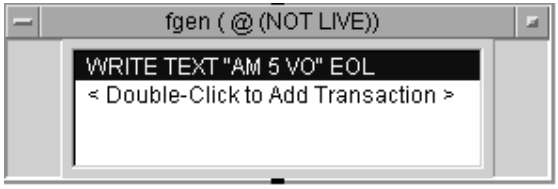


図94　　Direct I/Oトランザクション

ほとんどの場合、計測器にテキスト・コマンドを送信する方法は、同じです。しかし、各コマンドの末尾やコマンド・グループの末尾に特定の文字を送信する必要がある計測器があります。この送信文字については計測器のドキュメントを参照し、[Direct I/O Configuration]ダイアログ・ボックスでその文字を指定してください。

計測器への式リストの送信

計測器に式リストを送信したい場合があります。たとえば、Function Generatorでいくつかの周波数を繰り返させたい場合です。Direct I/Oトランザクションを使用してこれを行うには、式リストで周波数の変数を使用します。Direct I/Oオブジェクトにその変数のためのデータ入力端子を追加します。次の手順で、計測器に式リストを送信します。

1. メイン・ウィンドウにHP3325B用の2番目のDirect I/Oオブジェクトを配置します。トランザクション領域内をダブルクリックして、[I/O Transaction]ダイアログ・ボックスを表示します。

   コマンド文字列以外はすべてデフォルトの設定を使用します。この場合は、**"FR"**, <*frequency*>, **"HZ"**フォーマットを使用します。これは式リストであり、それぞれの式はカンマで区切られています。周波数は変数**A**によって表されますが、これは**Direct I/O**オブジェクトへのデータ入力になります。

2. コマンド文字列の入力フィールドをクリックして「`"FR",A,"HZ"`」と入力します。たとえば、**A**の値が`100`の場合、VEEは文字列**FR100HZ**を送信します。**[OK]**をクリックします。VEEは、**A**のラベルが付いたデータ入力ピンを自動的に追加します。

3 **[Flow]** ⇒ **[Repeat]** ⇒ **[For Range]**を選択し、オブジェクトをDirect I/O オブジェクトの左に配置します。

4 For Rangeのデータ出力ピンをDirect I/Oのデータ入力ピンに接続します。

5 **[For Range]**の各フィールドを編集して、`[From]`は`[10]`、`[Thru]`は`[1.8M]`、`[Step]`は`[50k]`を指定します。

[For Range]は、10から180万までの範囲の数値を5万ステップで送信します。数値がDirect I/Oオブジェクトで受け取られ、コマンド文字列によってFunction Generatorが周波数を出力します。Direct I/Oのセットアップは図95のようになります。

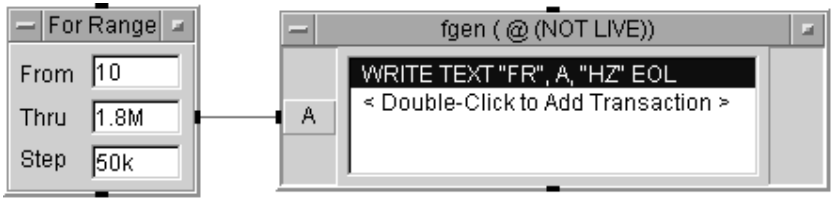


図95　　入力変数を使ったDirect I/Oのセットアップ

6 (オプション)HP3325B(所有している場合)をコンピュータに接続し、このDirect I/Oオブジェクトの設定を編集して計測器のアドレスを付け加えます。プログラムを実行すると、計測器がこの周波数を生成します。

計測器からのデータ読込み

計測器はさまざまなフォーマットでコンピュータにデータを送信します。計測器からデータを読み込むには、読み込むデータの種類と、そのデータが単一の値(スカラ)または配列のどちらで返されるのかを知る必要があります。また計測器がデータをテキスト(ASCII)またはバイナリのどちらで返すのかも知る必要があります。

返されるデータについての情報は計測器のドキュメントで確認できるほか、の[I/O]メニューにあるVEE[Bus I/O Monitor]を使って調べることができます。この情報により、I/Oトランザクションの設定方法が決まります。

この例では、1つ前の例題で説明したとおり、HP3478Aマルチメータが HP3325B Function Generatorに接続されています。Generatorが特定の周波数を送信すると、マルチメータがデータを読み込んで結果をVEEに戻します。次の手順では、マルチメータ用トランザクションの設定方法を説明します。

注 記

この例では、[READ TEXT]トランザクションについて説明しています。[READ]には、ほかに[BINARY]、[BINBLOCK]、[CONTAINER]の選択肢がありますが、これらについては『*VEE Pro Advanced Techniques*』で詳述しています。

1. **[I/O]** ⇒ **[Instrument Manager...]**を選択します。**[Add...]**をクリックします。この名前をdvmに変更します。**[Advanced...]**をクリックし、[Live Mode:]を[OFF]に設定します。HP3478Aを接続していない場合は、**[OK]**をクリックして[Instrument Manager]に戻ります。HP3478Aを接続**している**場合は、アドレスを修正すると、計測器がコマンドを実行します。
2. **[dvm(@(NOT LIVE))]**を強調表示し、[Create I/O Object]の下にある**[Direct I/O]**をクリックします。
3. **[<Double-Click to Add Transaction>]**バーをダブルクリックして[I/O Transaction]ダイアログ・ボックスを表示します。
4. 入力フィールドを強調表示し、「T5」と入力します。次に、**[OK]**をクリックします。これにより、計測器に**T**コマンドが書き込まれます。T5は、マルチメータに対する単一トリガ用のコマンドです。
5. オブジェクト・メニューを開いて**[Add Trans...]**をクリックし、トランザクション・バーをもう一つ追加します。または、**[<Double-Click to Add Transaction>]**をダブルクリックしてトランザクションを追加し、[I/O Transaction]ダイアログ・ボックスを表示します。
6. [WRITE]の隣の下向き矢印をクリックしてドロップダウン・メニューを表示し、[READ]を選択します。[READ]を選択すると、[I/O Transaction]ダイアログ・ボックスに新しいボタンが表示されます。
7. [ExpressionList]入力フィールドをチェックして、「**x**」が入力されていることを確認します。PressTabキーを押して次のフィールドに移動します。計測器から返されるデータは、データ出力ピンへ送られます。この場合は、データは計測器から読み込まれ、「**x**」という名前のデータ出力ピンへ送られます。

注 記　名前は大文字と小文字を区別しません。

8 [REAL64 FORMAT]はデフォルト値のままにします。マルチメータは個々の読み取り値を実数として返します。

9 [DEFAULT NUM CHARS]は変更しません。

デフォルトの文字数は20です。文字数を変更するには、[DEFAULT NUM CHARS]をクリックして[MAX NUM CHARS]に切り替え、**20**を目的の数値に変更します。

10 [SCALAR]は変更しないで**[OK]**をクリックします。

バーに、トランザクションがREAD TEXT X REAL64のとして表示されます。VEEは「**x**」という名前のデータ出力端子を自動的に追加します。

注 記　計測器が数値配列を返す場合は、[I/O Transaction]ダイアログ・ボックスの[SCALAR]メニューを選択して、図96のような次元のメニューを表示します。配列次元を選択した場合は、配列のサイズも指定する必要があります。

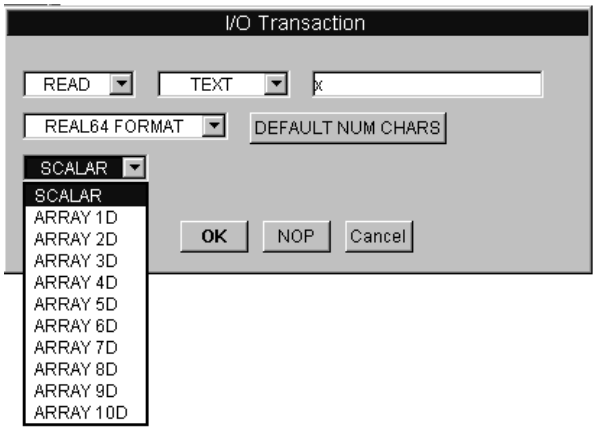


図96　　READトランザクションの設定

11 [Display] ⇒ **[AlphaNumeric]**を選択して**AlphaNumeric**オブジェクトを右側に追加し、入力ピンをDirect I/Oの「**x**」のラベルが付いた出力ピンに接続します。

2つのDirect I/Oトランザクションは図97のようになります。

図97　　測定値を読み込むように設定されたDirect I/O

トランザクションを設定する手順は、READ TEXTトランザクションのデータ形式に関係なく、似ています。使用できるほかの形式を調べることができます。各項目についての詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

完全なテスト・プログラムを作成するには、このマルチメータ・オブジェクトとFunction Generatorオブジェクトを VEEのデータおよび表示オブジェクトで結合します。VEEでは、完全に機能するテスト・プログラムを簡単に作成できます。しかし、使用する可能性があるさまざまな計測器すべての詳細を説明するのは、この入門のための章の範囲を超えています。詳細は、『*VEE Pro Advanced Techniques* manual』を参照してください。

計測器のステートのアップロード・ダウンロード

計測器のなかには、「ラーン・ストリング」機能を提供するものがあります。ラーン・ストリングには、計測器のステートを構成する関数設定のすべてが含まれます。Direct I/Oはこのラーン・ストリングをアップロードして、特定のDirect I/Oオブジェクトとともに保存します。保存したラーン・ストリングは、後でプログラム内の計測器にダウンロードできます。計測器のステートのアップロードは、次の手順で行います。

1. 計測器を手動で目的のステートに設定します。
2. Direct I/Oオブジェクトのオブジェクト・メニューを開いて、**[Upload State]**をクリックします。

   これにより、設定したステートが、Direct I/Oオブジェクトのこの特定のインスタンスに関連付けられます。
3. トランザクション領域でダブルクリックして、[I/O Transaction]ダイアログ・ボックスを開きます。
4. **[TEXT]**をクリックして[STATE (LEARN STRING)]を選択します。次に**[OK]**をクリックして[I/O Transaction]ダイアログ・ボックスを閉じます。このWRITEトランザクションを実行すると、あらかじめ取得したステートが計測器に送信されます。

アップロードおよびダウンロードは、[Direct I/O Configuration]ダイアログ・ボックスの設定によって制御されます。[Conformance]がIEEE 488.2の場合、は、自動的に「488.2 *LRN?」定義を使ってラーン・ストリングを処理します。[Conformance]がIEEE 488の場合は、[Upload String]でステートの照会に使用するコマンドを指定し、[Download String]がダウンロード時にステートの文字列の前に送信されるコマンドを指定します。図98に例を示します。

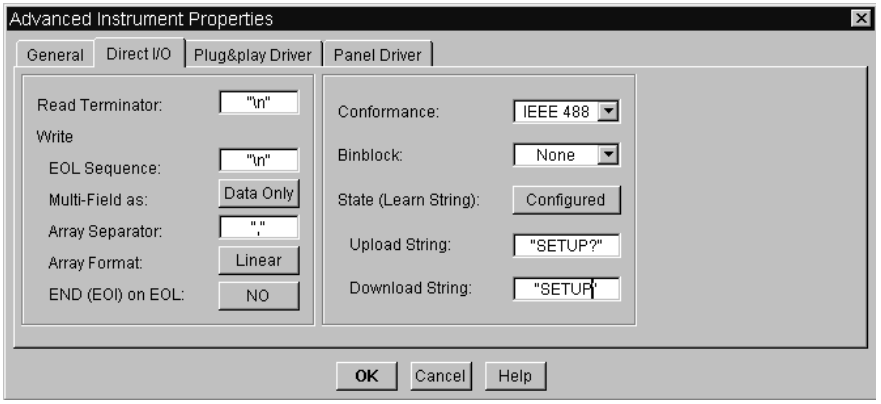


図98　　HP54100A用のラーン・ストリング設定

[Conformance]には、[IEEE 488]または[IEEE 488.2]を指定できます。この例はHP 54100A Digitizing Oscilloscopeを使用していますが、この製品はIEEE 488規格に適合しており、ラーン・ストリングの照会には`SETUP?`コマンドを、ダウンロード時のラーン・ストリングの前には`SETUP`コマンドを必要とします。[State (Learn String)]で[Configured]を選択すると、Upload StringとDownload Stringという2つのフィールドが表示されます。この2つの入力フィールドには、適切な文字列がすでに入力されています。

PCプラグイン・ボードの使用方法

VEEでは、PCプラグイン・ボードまたはカードを制御する方法として次の3つが提供されています。

1. Data Translation社のVisual Programming Interface。VPIアプリケーションはData Translation社に直接注文してください。
2. ComputerBoardsやMeilhausなどのPCボード・メーカから提供されているDLL(Dynamic Link Libraries)。DLLの使用方法についての詳細は、453ページの「ダイナミック・リンク・ライブラリの使用法」を参照してください。

Data Translation社のVisual Programming Interface(VPI)

Data Translation社のVPIはVEEで動作して、PCプラグインのためにシームレスなデータ収集性能を発揮します。Data Translation社のOpen Layers規格の柔軟性を導入することにより、50以上のデータ収集ボードにアクセスできます。

VPIは、低いチャネル数を要求するプラグインISA、PCI、USBベースのデータ収集カードで直接動作します。VPIはメニューの選択項目と特定のPCプラグイン・データ収集アイコンをVEEに追加します。これらはData Translation社製ハードウェアの機能を制御します。

Amplicon

Ampliconは、シリーズ200用の幅広いアナログおよびディジタルのI/O PCプラグイン・ボードでを提供しており、すべてにVEEでサポートされています。

AmpliconのAMPDIOドライバ・パッケージにはソフトウェア・インタフェースが含まれています。このインタフェースは、Windows用マルチスレッドDLLを持ち、割り込み駆動型収集をサポートする32ビットAPIです。APIには100個以上の関数呼び出しが用意されており、VEE固有の定義ファイルを使用して、コンパイル済み関数として効率良く柔軟にプログラミングを行うことができます。また、1つのプログラムで最大8つのボードを簡単に使用することができます。

シリアル通信デバイスを含むAmplicon独自のプラグイン・ボードに加えて、Ampliconでは、データ収集、シリアル通信、GPIBアプリケーション用に、他のさまざまなメーカのボードも提供しています。

図99で示すように、VEE実行ソフトウェア(Ampliconのアナログ出力ボードPCI224とPCI234、およびアナログ入力ボードPCI230とPCI260で無料で提供されています)は、PC上で入力/出力信号を同時に送信できます。

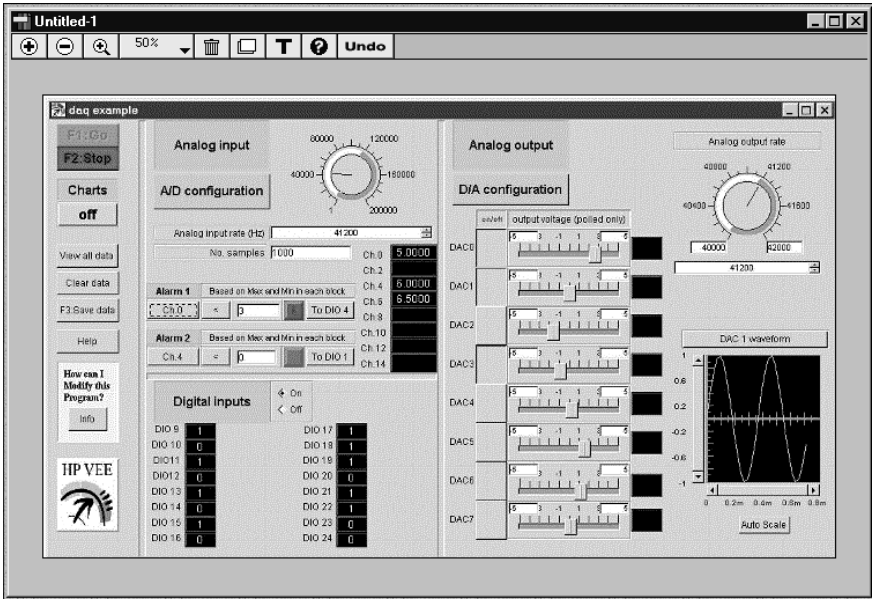


図99　　Ampliconによるデータ収集例

ComputerBoardsが提供するPCプラグイン

ComputerBoardsは、VEEと互換性のある低価格で強力なPCプラグイン・ボードを提供します。サポートされているPCプラグイン・ベンダの完全なリストについては、VEEのドキュメントまたは『*VEE Pro Advanced Techniques*』を参照してください。

ボードとI/Oライブラリをインストールして、メーカが提供するプログラムを使ってボードを設定します。説明書に従ってボードをデバイスに接続します。VEEでは、ライブラリをインポートだけで、ComputerBoards I/Oライブラリの測定関数を呼び出すことができます。メーカが提供するデモ・プログラムから抜粋した次の図を参照してください。

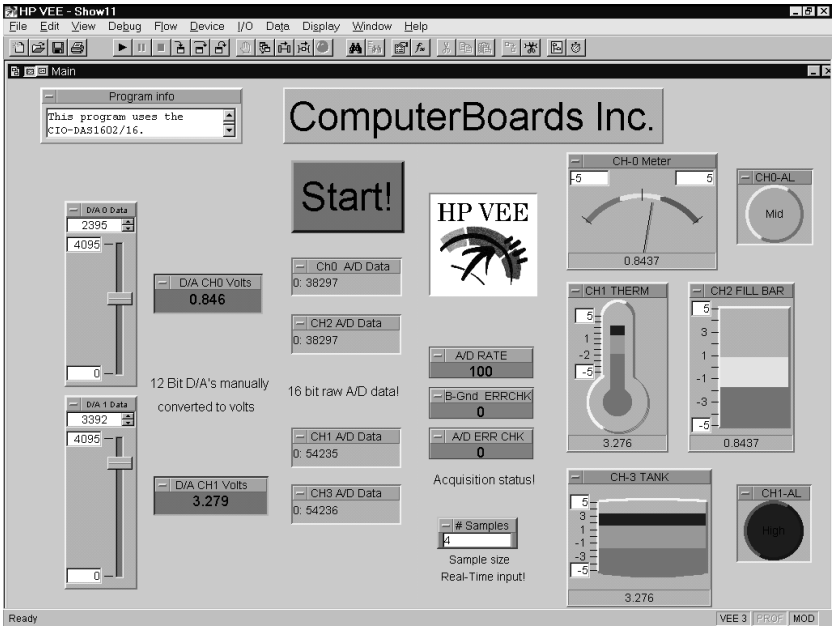


図100　によるComputerBoards 100kHzボードの使用

図100は、この100kHz A/Dボードを使ったデモ・プログラムのパネル・ビューです。また図101は、これらのデータ収集関数の呼び出しを可能にするComputerBoards I/Oライブラリを VEEがインポートしているところを示したものです。

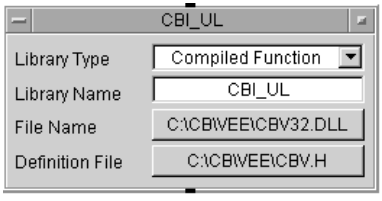


図101　ComputerBoards I/Oライブラリのインポート

Meilhaus ElectronicのME-DriverSystem

Meilhaus Electronicは、ヨーロッパを代表するPCベースのデータ収集/インタフェース技術の設計・生産・販売会社です。CD-ROMで提供しているWindows向けME-DriverSystemには、Meilhaus Electronicによって生産されているすべてのデータ収集ボード(MEシリーズ)が含まれます。ME-DriverSystemはまた、VEEのメニュー構造に統合されます。

VEE用ME-DriverSystemをインストールすると、VEEのプルダウン・メニューにドライバの関数が表示されます。図102はVEEにおけるME Boardメニューを示しています。

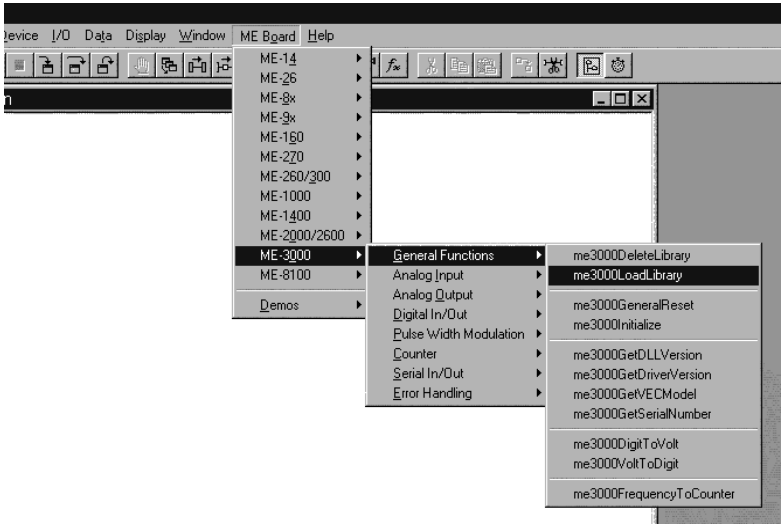


図102　　VEEおけるME Boardメニュー

メニューの第2位のレベルには、Analog InputおよびOutput、Digital I/O、特定のボードの関数といった関数グループがあります。図103は、データ取得ボードME-3000のユーザ・パネルを示します。

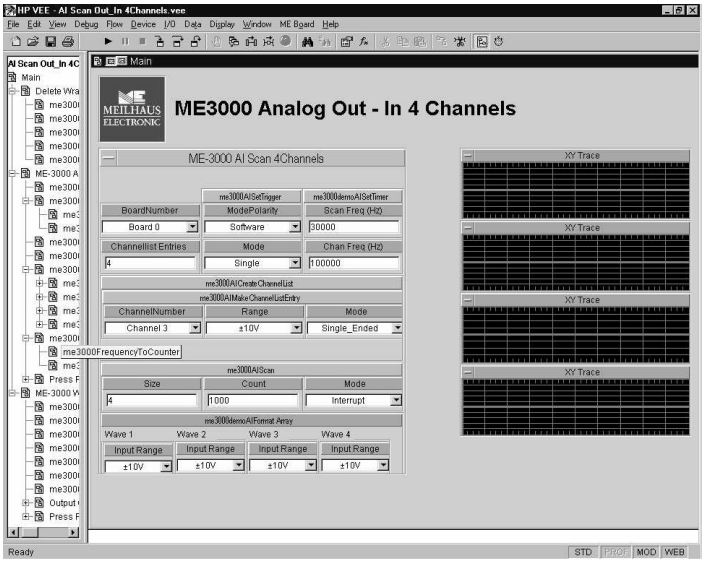


図103　　データ取得ボードME-3000のユーザ・パネル

最後に、第3位のメニューには、me3000AISingleのような実際の関数があります。図104は関数パネルを示します。

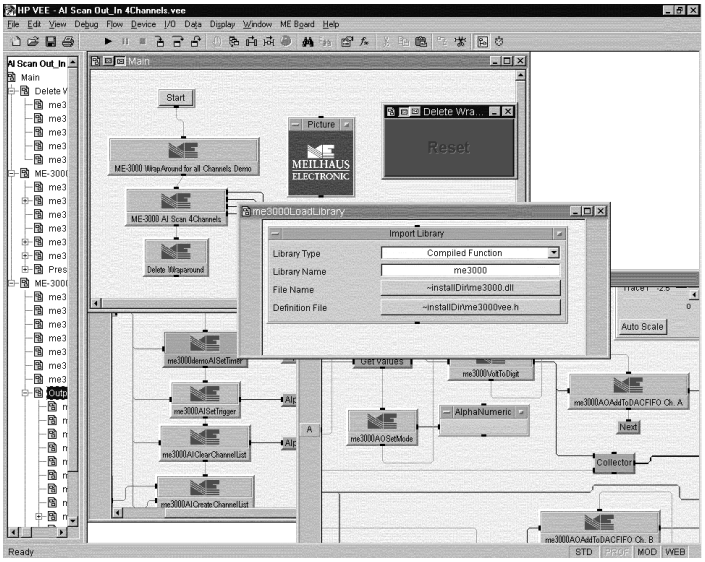


図104　　ME-DriverSystemの関数パネル

VXI*plug&play*ドライバの使用方法

VXI*plug&play*ドライバは、さまざまな計測器ベンダから提供され、サポートされています。これはCベースのドライバであり、最大限のパフォーマンスと使いやすさを実現するように設計されています。

Agilent VEEは、VXI*plug&play*と完全に互換性があります。Agilent VEEは、VXI*plug&play*と完全に互換性があります。Agilent Technologiesから入手できるすべての、VXI*plug&play*ドライバは、別製品として搭載されています。またWebサイト`http://www.agilent.com/find/inst_drivers`とAgilent Developers Network (ADN)、`http://www.agilent.com/find/adn`でも入手できます。また同じドライバが、すべてのAgilent Technologiesパネル・ドライバと共にVEEに入っています。その他の計測器用のVXI*plug&play*ドライバを入手するには、計測器ベンダにお問い合わせください。

例題3-4: VXI*plug&play* Driverの設定

この例では、HPE1412ドライバの設定方法を説明します。

1. **[I/O]** ⇒ **[Instrument Manager...]**を選択します。
2. [My Configuration]を強調表示して、[Instrument]の下にある**[Add...]**をクリックします。[Instrument Properties]ダイアログ・ボックスが表示されます。[instrument]などの名前を入力し、**[Advanced...]**をクリックして、[Advanced instrument Properties]ダイアログ・ボックスを表示します。
3. [Advanced instrument Properties]ダイアログ・ボックスで、[Live Mode:]を[OFF]に切替え、[Plug&play Driver]フォルダを選択します。**[Plug&play Driver Name:]**フィールドをクリックして、コンピュータにインストールされているすべてのドライバをリストしたドロップダウン・メニューを表示します。この例では、図105に示すようにHPE1412ドライバを使用します。

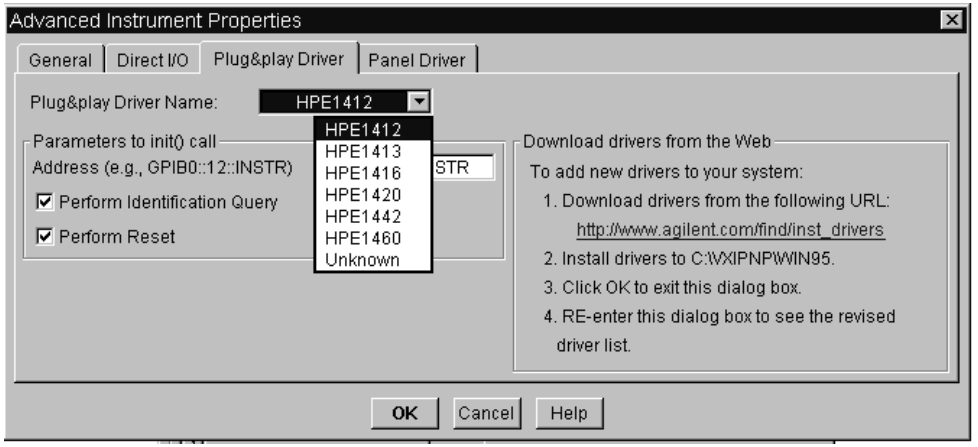


図105　　VXI*plug&play*ドライバの選択

HPE1412ドライバを選択し**[OK]**をクリックして、[Instrument Properties]ダイアログ・ボックスに戻ります。次に、**[OK]**をクリックして[Instrument Manager]に戻ります。これで、リストに[instrument(@(NOT LIVE))]の項目が追加されます。

4 [instrument(@(NOT LIVE))]を強調表示して、[Create I/O Object]の下にある[Plug&play Driver]を選択します。クリックしてオブジェクトを配置します。

注 記　VEEでは、VXI*plug&play*ドライバはDirect I/Oオブジェクトに似ています。

計測器で測定を行うには、VXI*plug&play*ドライバでC関数を使用するI/Oトランザクションを設定する必要があります。ドライバは、使用する適切な関数を選択するためのパネルを提供します。

5 <Double-click to Add Function>のラベルが付いたトランザクション・バーをダブルクリックすると、図106のように[Select a Function Panel]が表示されます。図107は、関数パネル内での関数の階層を示しています。また、選択した項目についてのヘルプがダイアログ・ボックスに表示されます。

注 記

VEEは、自動的に計測器を初期化します。ほかのプログラム言語のようにinit関数を使用する必要はありません。

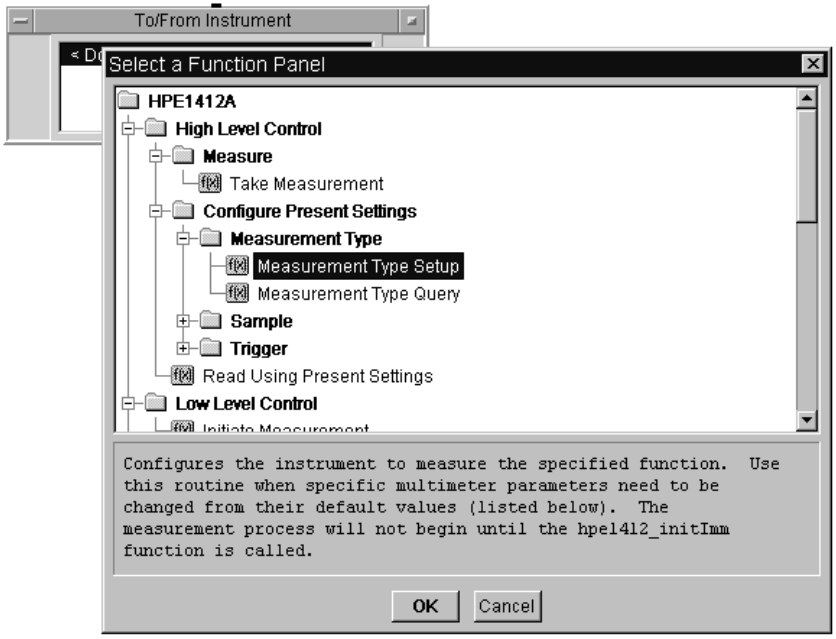


図106　VXI*plug&play*ドライバ用関数の選択

6 **[Configure Present Settings]** ⇒ **[Measurement Type]** ⇒ **[Measurement Type Setup]**をクリックします。**[Edit Function Panel]**が表示されます。**[func]**の下でクリックしてドロップダウン・リストを表示します。図107のに示すように、デフォルトの**[DC Voltage]**を選択します。

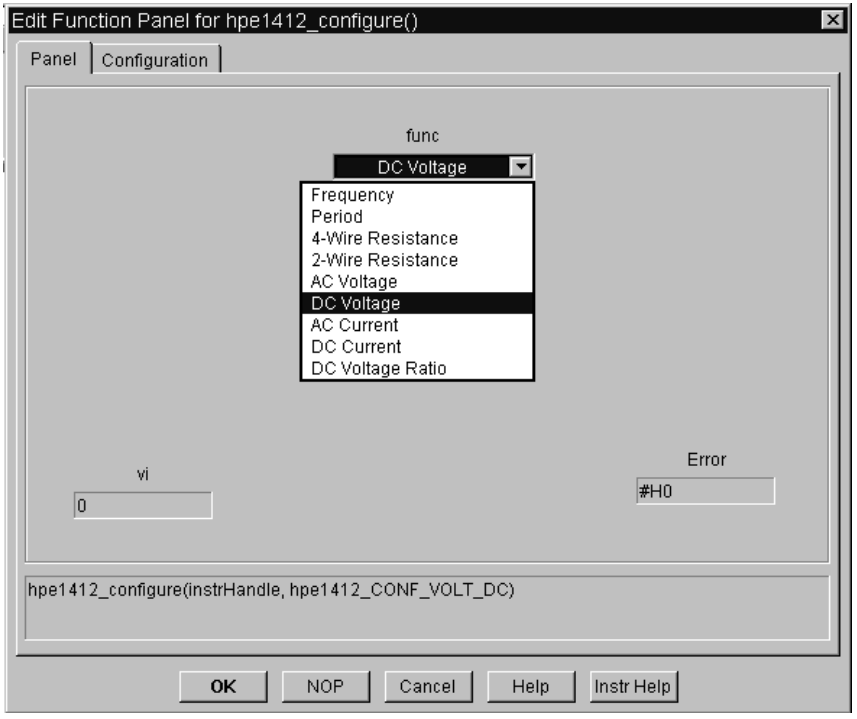


図107　HPE1412の[Edit Function Panel]

7 **[OK]**をクリックします。これで、図108に示すように、[To/From Instrument]オブジェクトに**[hpe1412_configure(instruHandle,hpe1412_CONF_VOLT_DC)]**の項目が追加されます。

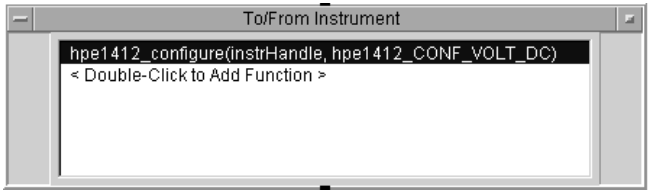


図108　VXI*plug&play*オブジェクトにおけるDC Voltage Function

8 [To/From Instrument]オブジェクトでDouble-click to Add Functionをダブルクリックし、[Measure]の下にある[Take Measurement]を選択します。[Configuration]フォルダをクリックして、ダイアログ・ボックスの表示を図109のようにします。

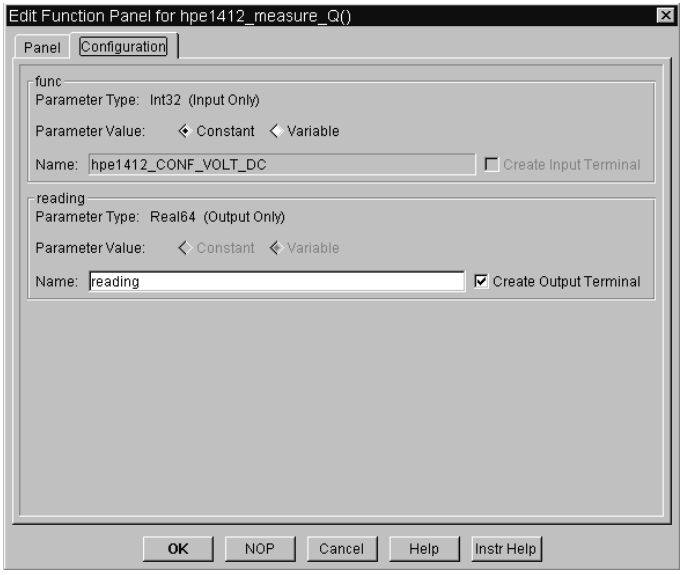


図109　[Edit Function Panel]の[Configuration]フォルダ

9 **[OK]**をクリックします。2番目の関数の呼び出しが[To/From Instrument]オブジェクトに図110に示すようにリストされます。

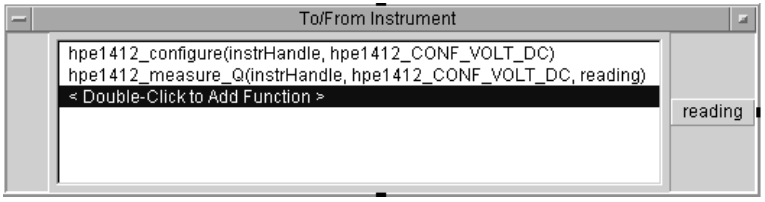


図110　DC読込みのために用意されたHPE1412ドライバ

そのほかのI/O機能

1. VEEのすべてのI/O機能については**[I/O]** ⇒ **[Advanced I/O]**のサブメニュー、[Interface Operations]、[Instrument Event]、[Interface Event]および[Multiinstrument Direct I/O]で調べることができます。
2. I/Oメニューでは、Bus I/O Monitorを使用してデバッグを行うためにバス・アクティビティを表示、印刷、または保存できます。
3. VEEには、プログラムによって計測器を検索できるActiveXオートメーション・サーバが含まれています。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。
4. I/O構成は、実行時にプログラムによって変更することもできます。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じて適切なトピックを復習してください。

- 計測器ドライバおよびDirect I/Oを使用する利点を説明する。
- 計測器を制御する手順を説明する。
- 計測器をステート・ドライバ用にを設定する。
- 計測器をDirect I/O用に設定する。
- 計測器ドライバの設定を変更する。
- コンポーネントの入力/出力を追加/削除する。
- 計測器ドライバ上で異なるパネルに移動する。
- Direct I/Oを使って計測器にコマンドを書き込む。
- Direct I/Oを使って計測器からデータを読み込む。
- ラーン・ストリングを使って計測器のステートをアップロード/ダウンロードする。
- VXI*plug&play*ドライバを使って計測器と通信する。
- PCプラグイン・ボードを制御する3つの方法を説明する。

4
テスト・データの解析と表示

テスト・データの解析と表示

この章の内容

- VEEのデータ型
- VEEの解析機能
- 演算オブジェクトの使用方法
- Formulaオブジェクトの使用
- Scriptオブジェクトの使用方法
- VEEの表示機能
- 表示のカスタマイズ

平均所要時間: 1.5時間

概要

この章では、VEEの解析機能および表示機能について学びます。また、アプリケーションに適切な演算オブジェクトを配置する方法およびテスト結果の表示方法について学びます。これにより、データを有用な情報に簡単にすばやく変換できるようになります。

また、ActiveXオートメーションを使用すると、MS Excelのような一般的なアプリケーションを使ってデータを解析することもできます。詳細は、第6章「ActiveXを使ったレポートの簡単な作成方法」(247ページ)を参照してください。ActiveXコントロールを使用すると、VEEの外部の表示機能を使用できます。詳細は、432ページの「例題11-4: ActiveXコントロールの使用方法」を参照してください。この章では、VEE自身の主要ツールとVEEに含まれるMATLAB Scriptオブジェクトを中心に説明します。

Agilent VEEのデータの種類とデータ型

VEEプログラムでは、データはオブジェクト間のラインを通って伝達され、次のオブジェクトで処理されます。データ集合を明確にするために、VEEではデータを**データの種類(スカラ、配列**など)と**データ型**(Int32、Real64、テキストなど)を持つ「コンテナ」というパッケージにします。

データの種類: スカラは単一の数字で、複素数のような2つ以上の要素で表現される数も含まれます。配列はデータ項目のグループで、1次元(Array 1D)、2次元(Array 2D)、などとして特定できます。

データ型: VEEのデータ型については表14で説明しています。

一般には、データ型やデータの種類について気にする必要はありません。ほとんどのオブジェクトは、VEEのどのデータ型に作用し、そのオブジェクトで必要なデータ型にデータを自動的に変換するからです。たとえば、Magnitude Spectrumの画面がWaveformデータ型を受け取った場合、VEEは自動的に高速フーリエ変換を実行してデータをタイム・ドメインから周波数ドメインに変換します。

しかし、特定のデータ型を要求するオブジェクトもあるので、それらについても知っておく必要があります。VEEとMATLABではサポートされているデータ型が異なるため、この点についても注意が必要です。詳細は、192ページの「データ型の処理方法」のセクションを参照してください。

次の表は、VEEのデータ型についての簡単な説明です。これらのデータ型の使用方法については、次章で説明します。

表14　Agilent VEEのデータ型

データ型	説明
UInt8	0から255までの無符号バイト数です。
Int16	16ビットの2の補数整数です(-32768～32767)。

表14　Agilent VEEのデータ型

データ型	説明
Int32	32ビットの2の補数整数です(-2147483648～2147483647)。
Real32	32ビットの浮動小数点数でIEEE 754標準に準拠しています(+/-3.40282347E+/-38)。
Real64	64ビットの浮動小数点数でIEEE 754標準に準拠しています(+/- 1.797693138623157 E308)。
PComplex	(mag, @phase)の形式で表した振幅および位相成分から成ります。位相は、デフォルトでは度数単位に設定されていますが、**[File]** ⇒ **[Default Preferences]** ⇒ **[Trig Mode setting]** を選択してラジアンやグラディアン単位に設定することができます。
Complex	直交座標またはデカルト座標の複素数で、(real, imag)の形式で表した実数および虚数成分から成ります。各成分のデータ型はReal64です。たとえば、複素数「1 + 2i」は(1,2)として表現されます。
Waveform	タイム・ドメインの値から成る複合データ型であり、等間隔で線形にマッピングされた点のReal64値と、波形の全タイム・スパンを保持しています。Waveformのデータの種類は、1次元配列(Array 1D)でなければなりません。
Spectrum	周波数ドメインの値から成る複合データ型であり、点のPComplex値と、周波数の最小値および最大値を保持しています。ドメイン・データは対数または線形にマッピングすることができます。Spectrumのデータの種類は、1次元配列(Array 1D)でなければなりません。
Coord	(x,y,...)の形式で表され、最低2つの成分から成りたつ複合データ型です。各成分のデータ型はReal64です。Coordのデータの種類はスカラまたはArray 1Dでなくてはなりません。
Enum	整数値に対応させたテキスト文字列です。ordinal(x)関数を使って整数値にアクセスできます。
Text	英数字文字列です。
Record	各データ型ごとのフィールドから成る複合データ型です。各フィールドには名前とコンテナがあり、どのような型と種類(Recordを含む)でも含むことができます。

表14　Agilent VEEのデータ型

データ型	説明
Object	ActiveXオートメーションおよびコントロール専用のデータ型で、ActiveXコントロールに対する参照値、またはActiveXオートメーションの呼び出しから返される参照値です。実際には、このデータ型はIDispatchまたはIUnknownインタフェースに対する参照値です。
Variant	ActiveXオートメーションおよびコントロール専用で、ActiveX方式の呼び出しに対してBy Refパラメータ型として要求されるデータ型です。

注 記

混在した環境でのデータの共有については、**[I/O]** ⇒ **[To/From Socket]**を参照してください。

Agilent VEEの解析機能

VEEは、一般的な算術演算子およびそのほかの数多くの関数をサポートします。さらに、VEEにはMATLAB Script機能も備えられています。MATLAB Script機能は、The MathWorks社製の標準の高機能MATLABのサブセットであり、VEEに、信号処理、高度な数式処理、データ解析、科学的/工学的グラフィックを含む算術機能を付加します。MATLAB Script機能は、VEEと完全に統合されているので、どのようなプログラムにもMATLAB Scriptオブジェクトを入れることができます。

必要な演算関数がVEEとMATLABのどちらにもない場合は、さらにいくつかのオプションを使用できます。この章で後述するFormulaオブジェクトを使用すると関数を作成できます。C言語のようなコンパイル済み言語で関数を記述し、記述した関数をにリンクさせることもできます。またVEEから別のソフトウェアと通信することもできます。

組込み演算オブジェクトの使用方法

VEEで、**[Device]** ⇒ **[Function & Object Browser]**を選択すると、VEEおよびMATLABの組込み(プログラム済み)算術式にアクセスできます。

組込み演算子または関数へのアクセス

VEEの算術演算子および関数にアクセスするには、**[Device]** ⇒ **[Function & Object Browser]**を選択します。たとえば、特定の範囲の乱数を返す式を作成するには、図111に示すように、[Type:]には[Built-in Functions]、[Category:]には[Probability & Statistics]、[Functions:]には[random]を選択します。

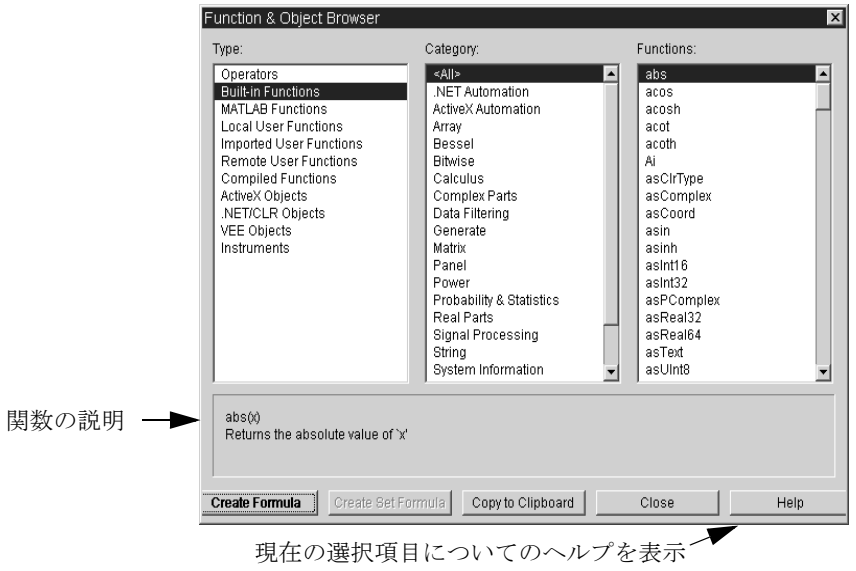


図111　[Function & Object Browser]におけるVEE関数

[Function & Object Browser]には、図111に示したように、現在の選択項目についての簡単な説明が表示されます。また、[Help]ボタンをクリックすると、現在の選択項目についてさらに詳しい説明が表示され、定義、使用方法、構文、使用例といった情報を得ることができます。

MATLABの演算子および関数にアクセスするには、**[Device] ⇒ [Function & Object Browser]**を選択し、[Type:]で[MATLAB Functions]を選択します。たとえば、ルートを多項式に変換するには、図112に示すように、[Type:]には[MATLAB Functions]、[Category:]には[Interpolation & Polynomials]、[Functions:]には[poly]を選択します。

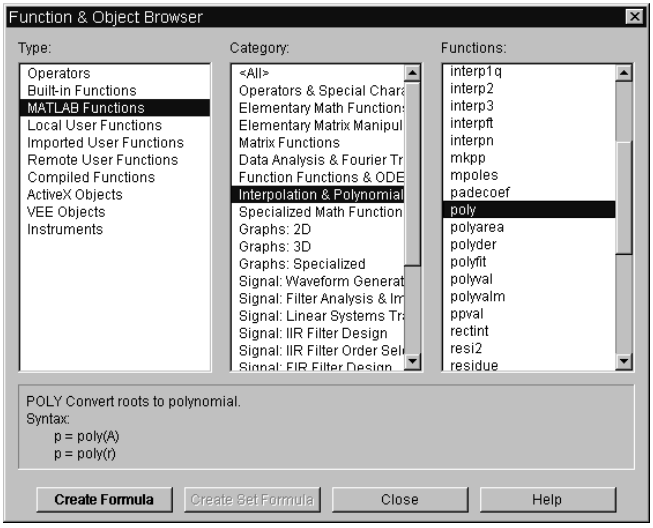


図112　[Function & Object Browser]におけるMATLAB関数

[Function & Object Browser]には、現在の選択項目についての簡単な説明が表示されます。また、[Help]ボタンをクリックすると、現在の選択項目についてさらに詳しい説明が表示されます。MATLABのRuntime EngineおよびScriptについては、188ページの「Agilent VEEにおけるMATLAB Scriptの使用」セクションで詳述しています。

例題4-1: 標準偏差の算出

周波数が1kHz、振幅が1V、タイム・スパンが20msで、256の点で表される余弦波形を生成します。この波形の標準偏差を算出して表示します。

5 **[Device]** ⇒ **[Virtual Source]** ⇒ **[Function Generator]**を選択します。[Frequency]を適切に設定してアイコン化します。

6 **[Device]** ⇒ **[Function & Object Browser]**を選択し、次に、[Built-in Functions]、[Probability & Statistics]、[sdev]を選択します。**[Create Formula]**をクリックします。

注 記

図113のようにツールバー上の[*fx*]アイコンを押すか、またはCtrl+Iキーを押すと、[Function & Object Browser]ダイアログ・ボックスに直接アクセスできます。

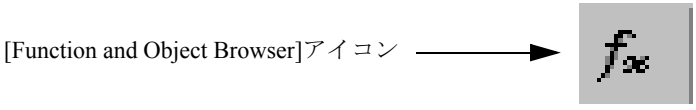


図113　[fx]アイコンを使った[Function and Object Browser]の表示

7 sdev()のオブジェクト・メニューを開いて[Help]を調べます。

注 記

sdev(x)オブジェクトは、**x**の分散の平方根として定義されます。**x**は、UInt8、Int16、Int32、Real32、Real64、Coord、Waveformのいずれのデータ型も可能です。Function Generatorは、Waveformデータ型を出力します。

8 Function Generatorをsdev(x)に接続します。

9 **[Display]** ⇒ **[AlphaNumeric]**を選択し、sdev(x)データ出力ピンに接続します。

10 プログラムを実行します。それは図114のように表示されます。

図114　標準偏差の算出

Formulaオブジェクトでの式の作成

Formulaオブジェクトは、VEEで算術式を記述するために使用できます。式内の変数はデータ入力ピンの名前またはグローバル変数です。式の評価結果はデータ出力ピンに渡されます。

図115はFormulaオブジェクトの例です。式を入力するためのフィールドはオブジェクトの中央にあります。デフォルトの式**(2*A+3)**が式を入力する場所を示しています。フィールドをダブルクリックして、別の式を入力します。

注 記

Formulaの式は、複数行に入力できます。Formula式に改行が含まれる場合、その式は複数行に入力された単一の式として解釈されます。Formulaにセミコロン(;)で分けられた文が含まれる場合、それらの文はFormula内の複数の式として解釈されます。

[Formula]ボックスはリッチ・テキスト・フォーマット・ボックスで、Formula内で式を編集するため標準の編集コマンドを使用できます。たとえば、End、Insert、Delete、Backspaceなどのキーが使用できるほか、マウスをドラッグして文字を強調表示し、`Ctrl+C`キーを使って文字をコピーしたり、`Ctrl+V`キーを使って文字を貼り付けたり、`Ctrl+X`キーを使って文字を削除したりできます。

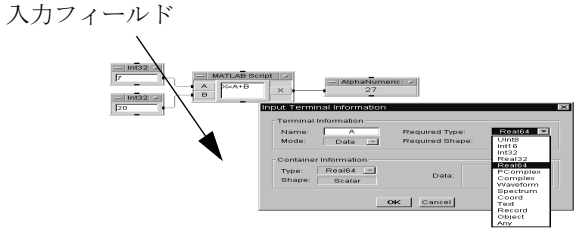


図115　Formulaオブジェクト

注 記

[Devices] ⇒ [Function & Object Browser]で[Type]に[Built-in]を選択して関数を作成すると、適切に設定された式を持つFormulaオブジェクトを簡単に作成できます。これらの関数は、関数を結合したり、入力端子を追加(または削除)するなどして修正できます。また、Formulaオブジェクトに複数行を入力したり、出力端子に値を割り当てることもできます。

Formulaオブジェクトを使った式の評価

この例では、**2*A^6-B**という式を評価します。この式において**A=2**、また**B=1**です。「**^**」の記号は累乗を表します。

注 記

変数の名前は、大文字小文字の区別はされ**ません。**

1 **[Device] ⇒ [Formula]**を選択します。[Formula]の入力フィールドをクリックし、「`2*A^6-B`」と入力します。

2 マウス・ポインタをデータ入力端子領域に置き(入力端子**A**の上に置かないようにします)、`Ctrl+A`キーを押して入力ピンを追加します。

注 記

追加された入力ピンはデフォルトではBのラベルが付けられますが、この名前は変更できます。

3 **[Data] ⇒ [Constant] ⇒ [Int32]**を選択します。次にオブジェクト・メニューから[Clone]を選択してInt32オブジェクトのクローンを作成します。作成した2つのInt32オブジェクトをFormulaの**A**と**B**の入力ピンに接続します。

4 **A Int32**の入力ボックスに「`2`」を入力し、**B Int32**の入力ボックスに「`1`」を入力します。

5 **[Display] ⇒ [AlphaNumeric]**を選択し、作成されたオブジェクトをFormulaの出力ピンに接続します。そしてプログラムを実行します。図116のように結果の127が表示されます。

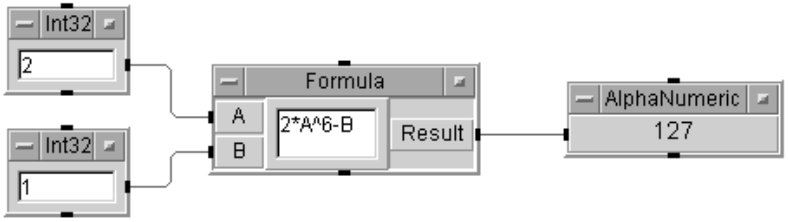


図116　　式の評価

FormulaオブジェクトでのAgilent VEE関数の使用方法

この例では、余弦波を生成し、Formulaオブジェクトを使って標準偏差と根二乗平均を算出します。

1. [Function Generator]、[Formula]、[AlphaNumeric]を選択してオブジェクトを作成し、データ・ピンで互いに接続します。
2. Formulaオブジェクトのオブジェクト・メニューを開いて[Clone]を選択して、クローンを作成します。次に、最初のFormulaオブジェクトの下に作成したクローンを配置します。Function Generatorのデータ出力ピンを2番目のFormulaオブジェクトに接続します。
3. [AlphaNumeric]のクローンを作成し、作成したクローンを2番目の[Formula]オブジェクトに接続します。
4. 最初のFormulaオブジェクトに「`sdev(A)`」を、2番目のFormulaオブジェクトにrms(A)を入力します。

   sdev(A)および**rms(A)**は、**[Device]** ⇒ **[Function & Object Browser]**ダイアログ・ボックスで作成できる演算関数でもあります。このように、これらは「関数」として呼び出すことも「独立したオブジェクト」として呼び出すこともでき、どちらの場合も同じ動作をします。
5. プログラムを実行します。図117に示したように、これらの関数をFormulaオブジェクトに入れると、プログラムは、独立したオブジェクトとして使用した場合と同じ答えを表示します。

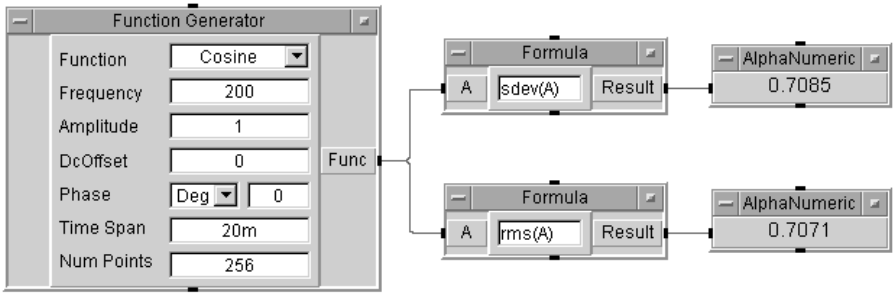


図117　　VEE関数を使ったFormulaの例

今度は、Formulaオブジェクトを1つだけ使って標準偏差および根二乗平均を計算します。Formulaオブジェクトは、それぞれに値を割り当てられた出力端子を複数持つことができます。

6 オブジェクト・メニュー・ボタンをダブルクリックして、[Formula]オブジェクトの1つを削除します。

7 残ったFormulaオブジェクトで式を次のように変更します。

```
B=sdev(A);
C=rms(A)
```

注 記

Formulaオブジェクトに式を複数含める場合は、式の終わりにセミコロンを入れて次の式と区別します。たとえば、**B=sdev(A);**の式で、セミコロンは式の終わりを示します。

注 記

Formulaオブジェクトでは、任意の位置に**改行**を入れることができます。式は、セミコロンがないかぎり、1つの式として解釈されます。たとえば、単一の式を次のように入力することもできます。

```
B=sdev
(A)
```

式にはまた、読みやすいようにスペースを入れることができます。

8 Formulaオブジェクトに出力端子を追加します。出力端子の名前を**B**および**C**に変更します。出力端子**B**をAlphanumericオブジェクトの1つに、また出力端子**C**をもう1つのAlphanumericオブジェクトに接続します。

9 プログラムを実行します。図118のように表示されます。

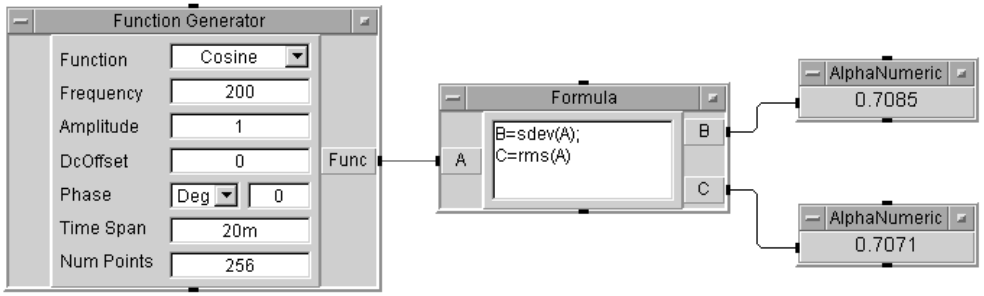


図118　Formulaオブジェクトを1つだけ使ったVEE関数

独習課題

図119に示すように、次の例題を完成して結果をチェックします。

1 [Built-in Functions]のGenerateカテゴリにあるrampオブジェクトを使用して、1から2048までの数の配列を作成します。この配列の標準偏差を算出して、その値を表示します。

2 rampオブジェクトのかわりにFormulaオブジェクトでramp()関数を使用して、手順1で説明した課題を実行します。

3 関数をネストして、手順1で説明した例題を実行します。オブジェクトは2つしか使用しません。

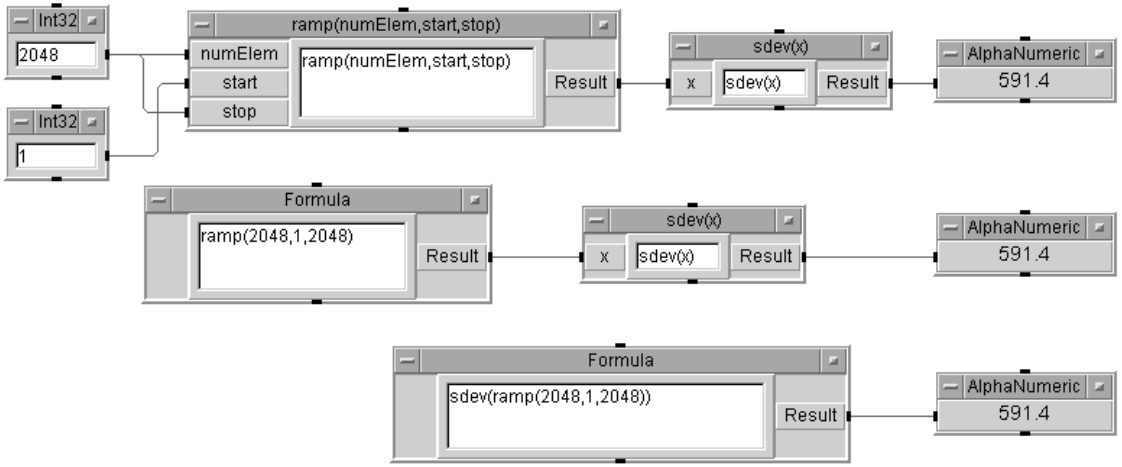


図119　RampおよびSDEVについての独習課題の解答

2番目および3番目の課題では、**Formula**オブジェクトの入力端子**A**を削除してエラー・メッセージが表示されないようにする必要があります。すべてのデータ入力ピンが接続され、データを受け取らないと、オブジェクトが動作できないからです。

Agilent VEEにおけるMATLAB Scriptの使用

VEEにはMATLAB Scriptオブジェクトがあり、MATLABの機能にアクセスできます。VEEはMATLAB Script Engineとデータのやり取りができるので、VEEプログラムにMATLAB算術関数を入れることができます。

注 記

すでにMATLABをインストールしている場合は、VEEはインストール済みのMATLABを使ってMATLAB Scriptを処理します。しかし、Signal Processing Toolboxがない場合は、VEEに同梱されているMATLAB Script Engineを登録しないと、VEEからMATLABの関数を使用することはできません。MATLABを登録するには、ディレクトリを(CDコマンドで)**<VEE_installation_dir>MATLAB\bin**に変更して、次のコマンドを実行します。**MATLAB.exe/regserver**

注 記

VEE MATLABインストールのトラブルシューティングに関する注意点をいくつか示します。まず、VEEは常に、MATLAB Scriptのディレクトリ・パス(~installDir\matlab\bin\win32)をアクティブなVEEセッションのPATH環境変数の最後に追加します。これにより、インストールされたすべてのMATLABが、VEEに付属のMATLAB Scriptに優先されます。また、VEEがlibeng.dllおよびlibmx.dllでLoadLibraryを実行しようとするときに、PATHで検出した最初のMATLABが取得されます。MATLABインストールは通常、matlab.exe、libeng.dll、libmx.dllの存在場所を付け加えるため、パスを修正します。注意事項として、libeng.dllとlibmx.dllのバージョンは、COM経由で登録されているMATLABのバージョンと一致している必要があります。

ロードされたMATLABのバージョンは、最後に登録されたバージョンです。これは通常、最後に実行したMATLABのバージョンとなります。MATLABは動作のたびに再登録を行うからです。このため、フル6.1とフル6.5をシステムにインストールしている場合に、問題が発生する可能性があります。VEEと対立させて実行するMATLABはいずれも、最後に登録されたものであり、さらに重要な点として、その[matlabroot\bin\win32]ディレクトリへのパスが他のMATLABバージョンよりも前にパスに現われる必要があります。

MATLAB Scriptオブジェクトの使用例

- VEEが生成したデータに対してMATLABを動作させる。

- MATLAB Scriptオブジェクトから結果を返して、VEEプログラムの別の部分でその結果を使用する。
- MATLABのSignal Processing Toolbox機能を使用して、MATLAB Scriptオブジェクトで高度なフィルタの設計や実装を行う。
- 2次元または3次元グラフを使ってデータを視覚化する。

図120は、VEEプログラム内でMATLAB Scriptオブジェクトがどのように使われるかを示しています。MATLAB Scriptプログラムが動作すると、データが生成されてAlphanumericオブジェクトに表示されます。

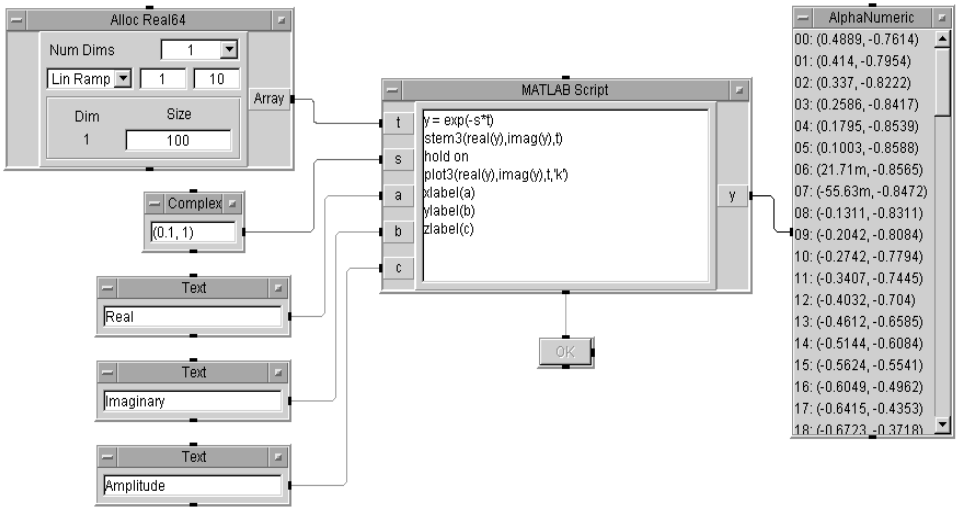


図120　VEEプログラムにおけるMATLAB Scriptオブジェクト

図121は、プログラムが実行されたときに作成されるグラフを示します。

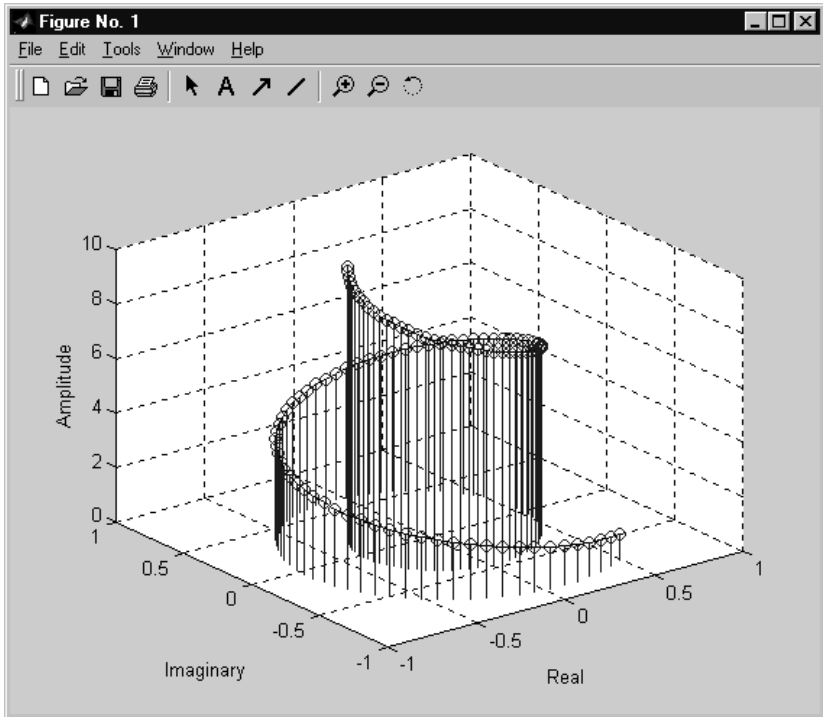


図121　　プログラムによって生成されたグラフ

VEEプログラムにMATLAB Scriptオブジェクトを入れると、VEEはMATLAB Script Engineを呼び出してMATLAB Scriptオブジェクト内の演算を実行します。情報は、VEEからMATLABへ渡され、再度VEEに戻されます。MATLAB Scriptオブジェクト使用にあたってのポイントは次のとおりです。

- プログラム内で最初に動作するMATLAB Scriptオブジェクトは、1つのMATLABセッションを開きます。プログラム内にあるほかのMATLAB Scriptオブジェクトはすべて、そのセッションを共有します。これにより、MATLAB ScriptオブジェクトはMATLABワークスペース内のグローバル変数を共有できます。

- VEEは、MATLAB Script Engineを呼び出さないかぎり、MATLABのコマンドの構文をチェック**しません**。MATLABによって生成されるエラーおよび警告は、VEEのエラーや警告と同じように、通常のVEEのダイアログ・ボックスに表示されます。
- VEEと異なり、MATLABは大文字小文字の区別を行います。MATLAB Scriptオブジェクトの入力/出力端子に大文字の**X**の名前を付けた場合、MATLABでは必ず、小文字の**x**ではなく大文字の**X**を使用してください。
- MATLAB Scriptに入力できるVEEのデータ型には制限があります。この制限については、192ページで詳述します。

Agilent VEEにMATLAB Scriptオブジェクトを入れる

プログラム内で使用するMATLABオブジェクトは、VEEのFormulaオブジェクトに似ています。MATLAB Scriptオブジェクトをプログラムに追加する方法は2通りあります。

1 **[Device]** ⇒ **[MATLAB Script]**を選択して、プログラム内のオブジェクトを配置する位置をクリックします。これにより、目的に応じて編集できるデフォルトのMATLAB Scriptオブジェクトが作成されます。

-または-

[Device] ⇒ **[Function & Object Browser]**を選択し、[Type:]でMATLABの関数を選択します。定義済みのMATLAB関数を選択し、[Create Formula]をクリックします。プログラム内のオブジェクトを配置する位置をクリックします。図122に、VEEプログラムに追加できる定義済みMATLAB関数の例を示します。

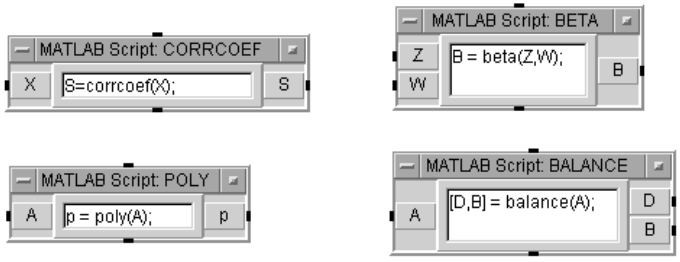


図122　　定義済みMATLABオブジェクトのVEEプログラムへの追加

各MATLABオブジェクトの名前はMATLAB Script<function name>の形式で付けられ、ほかのVEEのFormulaオブジェクトと区別できます。組込みのVEEのFormulaオブジェクトと同じように、各オブジェクトにはそのオブジェクトが実行する関数が入力されており、また必要と思われる入力および出力ピンが付けられています。また、ほかのVEEオブジェクトと同様に編集もできます。

注 記　MATLAB関数についての詳細は、VEEのメイン・ウィンドウで**[Help]** ⇒ **[MATLAB Script]** ⇒ **[Help Desk]**を選択してください。

データ型の処理方法

VEEのデータ型のサブセットのみがMATLABオブジェクトの入力および出力データとしてサポートされています。

VEEは、とMATLABの関数が混在するプログラムで処理しやすいように、1次元配列を自動的に変換します。たとえば、VEEの1次元テキスト配列は、MATLAB Scriptオブジェクトに入力されるとき、自動的に2次元の文字配列に変換されます。またMATLAB Scriptオブジェクトからの1次元文字配列は、MATLAB Scriptオブジェクトから出力されるとき、自動的にText Scalarに変換されます。

注 記　VEEのデータ型とMATLABのデータ型との自動変換の全リストおよび解説については、VEEのオンライン・ヘルプを参照してください。

次の例で示すように、入力端子のデータ型を制限することによっても、別のオブジェクトから入力されるデータをサポートされているデータ型に確実に変換できます。

1. **[Data]** ⇒ **[Constant]** ⇒ **[Int32]**を選択して、オブジェクトを配置する位置をクリックします。値を7に変更します。オブジェクトのクローンを作成して、2番目のInt32を最初のInt32の下に配置します。値を20に変更します。
2. **[Device]** ⇒ **[MATLAB Script]**を選択し、定数オブジェクトの右にオブジェクトを配置します。
3. [Display Alphanumeric]を選択し、オブジェクトをMATLAB Scriptオブジェクトの右に配置します。
4. 上のInt32オブジェクトの出力ピンをMATLAB Scriptオブジェクトの入力ピン**A**に接続します。下のInt32オブジェクトの出力ピンをMATLAB Scriptオブジェクトの入力ピン**B**に接続します。MATLAB Scriptオブジェクトの出力ピンをAlphanumericオブジェクトの入力ピンに接続します。

   プログラムを実行します。期待される入力データ型が、Real64、Complex、Waveform、Textのいずれかであるのに対し、Int32のデータが入力されたことを伝えるVEEの実行時エラーが生成されます。

このようなエラーを避けるには、MATLAB Scriptオブジェクトの入力端子のデータ型を変更します。

5. 端子**A**をダブルクリックして[Input Terminal Information]ダイアログ・ボックスを開きます。**[Required Type:]**をクリックしてドロップダウン・メニューを表示し、[Real64]を選択し**[OK]**をクリックします。端子**B**をダブルクリックし、図123と同様にしてデータ型をReal64に変更します。
6. プログラムを実行します。今度は、Int32データは、自動的に入力ピンでReal64に変換され、MATLABに渡されます。

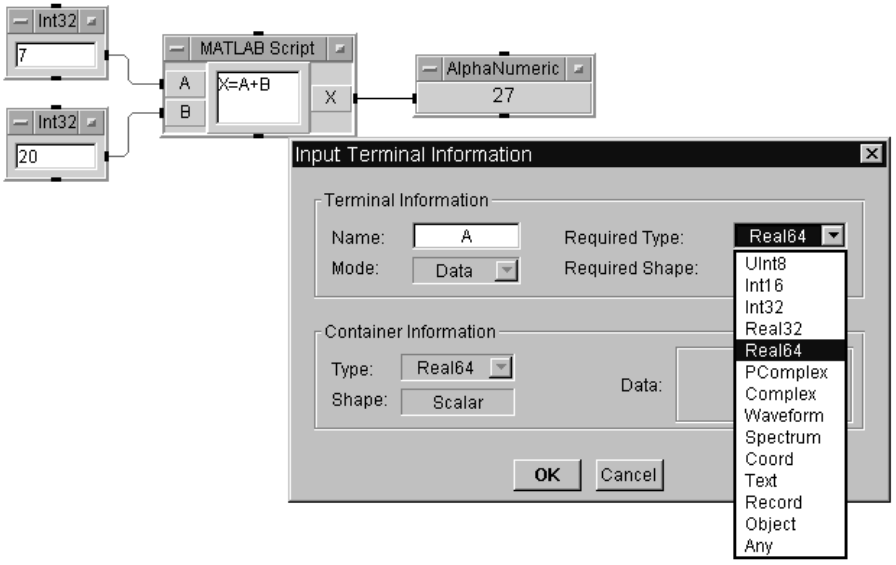


図123　　入力端子のデータ型の変更

テスト・データの表示

表15に、VEEのオブジェクト別の表示機能を示します。

表15　表示機能

表示用オブジェクト	説明
Alphanumeric	値をテキストまたは数字として表示します。表示できるのはSCALAR、ARRAY 1D、ARRAY 2Dのデータ型です。
Beep	音を鳴らして、プログラム内での場所を知らせます。
Complex Plane	Real(実数)対Imaginary(虚数)軸上にComplex、Polar Complex (PComplex)、Coordのいずれかのデータ型の値を表示します。
Indicator ⇒ Meter、Thermometer、Fill Bar、Tank、Color Alarm	これらのインジケータはすべて、数値を名前から予測できるグラフィックで表示します。これらはすべて、通常は3つ、ただしMeterは5つのカラーコード・レンジを持っています。Color Alarmは、テキスト・メッセージ付きの「LED表示」により、各レンジのアラームで色の光を割り当てられて、点灯します。
Label	パネル・ビューにテキスト・ラベルを置くときに使用するオブジェクトです。色とフォントは、パネル・ビューで、オブジェクト・メニューにある[Properties...]を選択すると簡単に調整できます。
Logging Alphanumeric	繰返し記録された値をテキストまたは数字で表示します。表示できるのは、SCALARまたはARRAY 1Dのデータ型です。
Note Pad	プログラムを説明するためのテキストを入力します。
Picture (PC)	パネル・ビューに画像を置くときに使用するオブジェクトです。サポートされている形式は、*.BMP (ビットマップ)、*.GIF(GIF87a および GIF89)、*.JPEG、*.PNG、*.WMF(Windows メタ・ファイル)です。
Polar Plot	半径と角度のデータに関して個別の情報を利用できるときに、データを極座標上にグラフィック表示します。

表15　　表示機能

表示用オブジェクト	説明
Spectrum (Freq)	Magnitude Spectrum、Phase Spectrum、Magnitude vs Phase (Polar)、Magnitude vs Phase (Smith)といった周波数ドメイン表示が含まれるメニューです。入力されるデータは、Waveform、Spectrum、Coordsの配列のいずれかでなければなりません。Waveformの入力値は、高速フーリエ変換(FFT)によって自動的に周波数ドメインに変換されます。
Strip Chart	プログラム実行中に連続して生成されるデータの最新履歴をグラフィック表示します。各yの入力値ごとに、x値が指定されたStepサイズだけ増分されます。新しいデータの表示が画面の右より外側へ出てしまうと、自動的に最新のデータを表示するようにスクロールします。
Waveform (Time)	実数のタイム・ドメインに波形またはスペクトラムをグラフィック表示します。スペクトラムは逆高速フーリエ変換(IFFT)を使って自動的にタイム・ドメインに変換されます。x軸は入力波形のサンプリング単位です。
X vs Y Plot	XおよびYデータに関して個別のデータ情報を利用できるときに、値をグラフィック表示します。
XY Trace	等間隔のx値を使ってyデータを生成したときに、マッピングされた配列または値の集合をグラフィック表示します。自動的に生成されるx値は、トレース・データのデータ型により異なります。たとえば、トレースが実数のときには、xは等間隔のReal値になります。一方、トレースが波形のときには、xは時間値になります。

テスト・データ表示のカスタマイズ

表示は、さまざまな方法でカスタマイズできます。VEEのすべてのオブジェクトと同様に、表示に対してラベルの作成、移動、サイズの変更などを行うことができるだけでなく、x/yスケールの変更、トレースの修正、マーカの設定、グラフィック表示の部分的な拡大表示なども行うことができます。

次の例では、このような機能の一部を説明します。この例では、Noise Generatorを使って波形を生成し、生成した波形をWaveform (Time)の画面に表示します。またこの例では、**X**スケールの変更方法、波形セグメントの拡大表示方法、波形の点の間の距離を測定するためのマーカの設定方法を説明します。操作の基本は、グラフィック表示すべてに共通です。

波形の表示

1. **[Device]** ⇒ **[Virtual Source]** ⇒ **[Noise Generator]**を選択します。
2. **[Display]** ⇒ **[Waveform (Time)]**を選択します。
3. Noise Generatorのデータ出力ピンをWaveform (Time)のデータ入力ピンに接続してプログラムを実行します。図124のように表示されます。

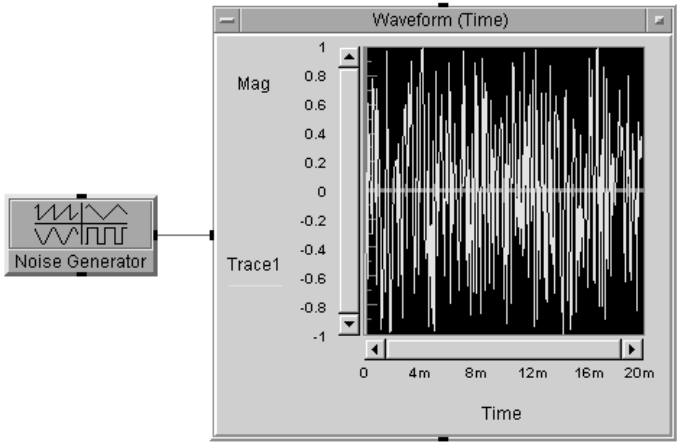


図124　　波形の表示

XおよびYスケールの変更

1 Waveform (Time)のタイトル・バーをダブルクリックして[Y Plot Properties]ダイアログ・ボックスを表示します。[Scales]フォルダを選択し、**[X Maximum]**の**[20m]**を選択し「1m」を入力します。

これにより、表示のタイム・スパンが20ミリ秒から1ミリ秒に変わります。

2 **-1**と表示されている**Y**軸の[Minimum]フィールドをダブルクリックし、「- .5」を入力します。**[OK]**をクリックします。

波形の部分的な拡大表示

1 Waveform (Time)のオブジェクト・メニューを開いて**[Zoom]** ⇒ **[In]**を選択します。

カーソルの形が小さい右向きの角型になります。クリック・アンド・ドラッグすると、拡大表示する領域の輪郭線をグラフ上に四角く描くことができます。

2 波形の山をいくつか含む領域の輪郭線を描いて、マウスのボタンを離します。

画面には、この波形の選択した領域が拡大表示されます。**x**および**y**スケールは自動的に変更されます。

画面にデルタ・マーカを設定する

1 Noise Generatorをオープン・ビュー表示にします。

a [Num Points]の設定を[16]に変更します。プログラムを再度実行します。

b Waveform (Time) のオブジェクト・メニューを開き、[Properties] を選択します。または、タイトル・バーをダブルクリックします。[Markers]の下にある**[Delta]**をクリックします。次に**[OK]**をクリックします。

注 記　**実行時にマーカの値の取得と設定を行うことができます。詳細は、[Contents and Index] ⇒ 「How Do I...」⇒ 「Display Data」にあるオンライン・ヘルプのトピックを参照してください。**

波形のデータ点の1つに上下を向いた2つの白い矢印が表示されます。また、これらのマーカの**x**および**y**座標が画面の一番下に表示されます。2つのピーク間の**x**座標または**y**座標の距離を測定するには、測定するピークまで矢印をクリック・アンド・ドラッグします。マーカの1つが新しいピークへ移動し、図125に示すように、新しい座標が画面の一番下に表示されます。

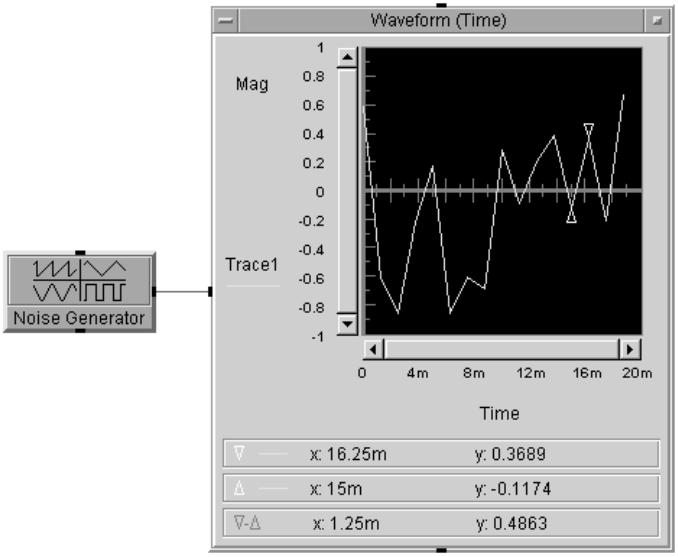


図125　　波形画面上のデルタ・マーカ

VEEは、波形のデータ点の間にマーカを自動的に挿入できます。オブジェクト・メニューを開いて、[Properties]を選択し、[Markers]の下にある**[MakerInterpolate]**をクリックします。

トレース色の変更

1 タイトル・バーをダブルクリックして[Properties]を開き、次に**[Trace]**プロパティをダブルクリックします。

このフォルダで選択したトレースのプロパティとして、Color、LineStyle、PointStyleを選択できます。

注 記

TracesまたはScalesコントロール入力を使用して、プログラム実行時にこれらの値を変更することもできます。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

2 これで、トレースは新しい色で表示されます。Grid Typeといったほかの表示特性は、上記例題の説明と同じような方法でカスタマイズできます。

注 記

VEEの表示のオブジェクト・メニューには[Plot]もあり、プログラムの残りをプリント・アウトすることなくテスト結果を画面にプロットできます。

さらに練習をつむには

このほかのVEEオブジェクトについて理解し、さらに練習を行うには、付録Aの541ページ「テスト・シーケンスの作成方法」の例題を行ってください。

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。

- VEEの主なデータ型について説明する。
- VEEの主な解析機能について説明する。
- [Function & Browser]ダイアログ・ボックスのオブジェクトに関するオンライン・ヘルプの説明を探す。
- VEEの演算オブジェクトにおける入力ピンと変数の関係を説明する。
- Formula オブジェクトを使って算術式を評価する。次に Formula オブジェクトを使って2つの式を評価する。最初の行の後に忘れずにセミコロンを付加する必要がある。
- Formulaオブジェクトの算術式にVEEの関数を使用する。
- MATLAB Scriptオブジェクトを使用する。
- VEEの主な表示機能を説明する。
- スケールの設定、波形の部分表示、マーカの設定、トレース色などに関してグラフィック表示をカスタマイズする。

5
テスト結果の保管方法と読み取り方法

テスト結果の保管方法と読み取り方法

この章の内容

- テスト・データの配列への入力方法
- Collectorオブジェクトの使用方法
- To/From Fileオブジェクトの使用方法
- `Record`を使った混用データ型の作成方法
- `DataSet`を使った検索操作とソート操作の実行方法
- `DataSet`オブジェクトを使った単純なテスト・データベースの作成方法

平均所要時間: 2時間

概要

この章では、テスト・データの保管と読み取りの基本を習得します。テスト結果を保持するため適切なデータ型とサイズの配列を作成し、次にデータまたはその一部にアクセスして、解析または表示を行います。

この章では、To/From Fileオブジェクト、Recordデータ型、DataSetファイルについても説明します。To FileオブジェクトとFrom Fileオブジェクトは、I/Oトランザクションに基づいてファイルにデータを書き込んだり、ファイルからデータを読み取ります。Recordデータ型を使用すると、単一の構造体に異なる型のデータを保管できます。DataSetを使用すると、1つのファイルに1つ以上のレコードを保管して、データセットに対して検索およびソート操作を実行できます。

注 記

To Fileオブジェクトについては、第2章「Agilent VEEのプログラミング技術」の85ページ「例題2-3: データファイルの使用方法」でも説明しています。

配列を使ったテスト結果の保管

データ型は次の2種類の方法で保管できます。

- スカラ値。つまり、「9」、「(32, @10)」などの単一の数値

-または-

- 1次元から10次元までの配列

注 記　VEEデータ型についての概要は、第4章「テスト・データの解析と表示」を参照してください。

VEEの場合、配列のインデックスは0から始まり、角かっこを使って配列要素の位置を示します。たとえば、配列**A**が要素[4, 5, 6]を保持している場合、これらの要素の位置は次のように示されます。

A[0] = 4、A[1] = 5、A[2] = 6

配列の構文は次のとおりです。

表16　配列の構文

構文要素	説明
コロン	要素の範囲を示します。たとえば、上の配列の場合、**A[0:1] = [4, 5]**となります。
アスタリスク(*)	配列の特定の次元にあるすべての要素を指定するためのワイルドカードです。**A[*]**は、配列**A**のすべての要素を返します。
カンマ	サブ配列の構文では、配列の次元を区切るためにカンマを使用します。**B**が、それぞれの次元に3つの要素を持つ2次元の配列であれば、**B[1,0]**は、**B**の2番目の行の最初の要素を返します。

配列の要素にアクセスするための構文は、Formulaオブジェクトまたは To/From Fileオブジェクトなどに含まれる任意の式フィールドで使用できます。

例題5-1: テスト結果用の配列の作成

配列を作成する最も簡単な方法は、Collectorオブジェクトを使用することです。

この例題では、For Countオブジェクトを使用して、計測器からの読み取りを4回シミュレートします。読み取った値を配列に挿入し、結果を出力します。Collectorでは、すべてのデータ型が受け入れられ、送信した要素の数に基づいて配列のサイズが自動的に作成されるため、データ型や配列のサイズにかかわりなく、原理は4回とも同じです。

1 **[Flow]** ⇒ **[Repeat]** ⇒ **[For Count]**、**[Data]** ⇒ **[Collector]**、**[Display]** ⇒ **[AlphaNumeric]**を選択します。

表17

オブジェクト名	説明
For Count オブジェクトについて	For Countは、入力フィールドで指定した反復回数を基にして、0から増加していく整数値を出力します。デフォルトの回数[10]をダブルクリックして強調表示してから、「4」と入力します。For Countは、**0**、**1**、**2**、**3**を出力します。
Collector オブジェクトについて	Collectorは、データ入力端子を介してデータ値を受信します。データの収集が完了したら、XEQ端子にpingを送り、Collectorで配列を作成して出力するよう指示します。For Countシーケンス出力ピンを使用すると、Collector XEQピンにpingを送ることができます。Collectorは、**1 Dim Array**と*n+1* **Dim Array**をトグルするボタンを表示します。 このオブジェクトについての詳細は、[Collector]をダブルクリックしてオープン・ビューを表示し、オブジェクト・メニューのヘルプを参照してください。

2 Collectorの**[n+1 Dim]**をクリックし、**[1 Dim Array]**に変更します。

3 For Countデータ出力ピンをCollectorのデータ入力ピンに接続します。

4 For CountシーケンスCollectorのXEQ入力ピンに接続します。

XEQピンは、複数のオブジェクトに存在する特殊なトリガ・ピンで、オブジェクトをいつ実行するかを決定します。この場合、配列のすべてのデータが収集されてから、オブジェクトを実行する必要があります。

5 CollectorのデータピンをAlphaNumericのデータ入力ピンに接続します。

6 AlphaNumericを拡大して配列を表示するには、このオブジェクトの四隅のいずれかをクリックし、ドラッグします。また、AlphaNumericを初めて選択するときにオブジェクトの輪郭線をクリック・アンド・ドラッグして、オブジェクトを拡大することもできます。

7 プログラムを実行します。図126のように表示されます。

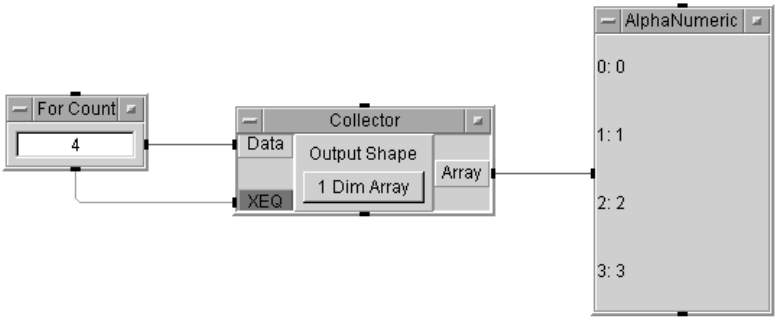


図126　配列を作成するCollector

例題5-2: 配列の値の抽出

配列から値を抽出するには、式の中で角かっこ表記を使用するか、**[Access Array]** ⇒ **[Get Values]**オブジェクトを使用します。次の例では、Formulaオブジェクトの中で式を使用します。この例題では、プログラムに複数のオブジェクトを追加します。

1 CollectorとAlphaNumericをつなぐデータ・ラインにマウス・ポインタを置き、`Shift`キーを押したまま`Ctrl`キーを押し、マウスの左ボタンをクリックして、このラインを削除します。次にCollectorをアイコン化します。

2 **[Device]** ⇒ **[Formula]**を選択し、クローンを作成します。AlphaNumericを右に移動し、Formulaオブジェクトを2つともCollectorの右に置きます。

3 Collectorのデータ出力を2つのFormulaオブジェクトのデータ入力に接続します。上のFormulaの入力フィールドに「`A[2]`」と入力し、下のFormulaの入力フィールドに「`A[1:3]`」と入力します。

4 `A[2]`は、配列の3番目の要素をスカラとして抽出し、`A[1:3]`は、**A**入力端子上の配列を意味する**A**の2、3、4番目の3つの要素から成るサブ配列を返します。

5 AlphaNumericのクローンを作成し、それぞれを各Formulaオブジェクトに接続します。

6 プログラムを実行します。図127のように表示されます。

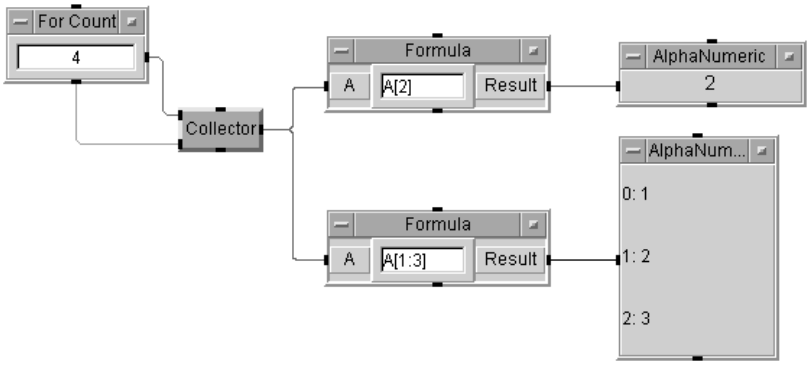


図127　　式を使った配列要素の抽出

To/From Fileオブジェクトの使用

To FileオブジェクトとFrom Fileオブジェクトは、I/Oトランザクションに基づいてファイルにデータを書き込んだり、ファイルからデータを読み取ります。2つのオブジェクトには次の特性があります。

- データ・ファイルは、初めてREADまたはWRITEトランザクションが実行されたときにオープンされます。プログラムが終了すると、VEEは、オープンしているすべてのファイルを自動的にクローズします。
- VEEは、多くのオブジェクトがファイルにアクセスしているオブジェクトの数に関係なく、ファイルごとに1つの読み取りポインタと1つの書込みポインタを保持します。読み取りポインタは次に読み取るデータを識別し、書込みポインタは次にデータを書き込む場所を示します。
- To/From Fileオブジェクトは、既存のファイルにデータを追加したり、既存のファイルを上書きできます。To Fileオブジェクトのオープン・ビューで[Clear File at PreRun & Open]設定にチェック・マークを付けた場合、書込みポインタはファイルの先頭から始まります。チェック・マークを付けない場合、書込みポインタは、既存のファイルの末尾に位置します。各WRITEトランザクションでは、ファイルの書込みポインタの位置に情報を追加します。EXECUTE CLEAR トランザクションが実行された場合、書込みポインタはファイルの先頭に移動し、ファイルの内容が消去されます。
- 読み取りポインタは、ファイルの先頭から始まり、READトランザクションで読み取るデータ量に応じて移動します。From FileオブジェクトのEXECUTE REWINDを実行すると、データに影響を与えることなく、読み取りポインタをファイルの先頭に戻すことができます。

注 記　To Fileオブジェクトについては、第2章「Agilent VEEのプログラミング技術」の85ページ「例題2-3: データファイルの使用方法」でも説明しています。

I/Oトランザクションについて

I/Oトランザクションは、計測器、ファイル、文字列、オペレーティング・システム、インタフェース、そのほかのプログラム、プリンタと通信するために、VEEによって使用されます。たとえば、図128の中のTo Fileオブジェクトについて説明します。

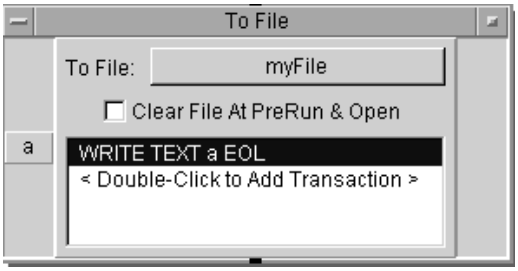


図128　To Fileオブジェクト

図128に示したTo Fileオブジェクトは、データを指定されたファイルmyFileに送信します。このオブジェクトは、プログラムからデータを受け入れるために、トランザクションと呼ばれる入力を保持できます。たとえば、このTo Fileオブジェクトは、WRITE TEXT a EOLトランザクションを保持しています。そのトランザクションをダブルクリックすると、図129に示す[I/O Transaction]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、具体的なトランザクション・ステートメントを設定します。

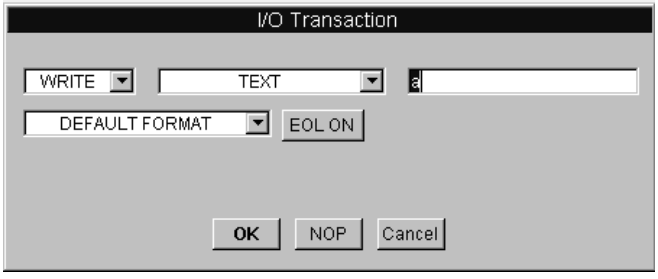


図129　[I/O Transaction]ダイアログ・ボックス

このダイアログ・ボックスは、オブジェクトによって形式が異なりますが、いずれも、action(操作)、encoding(エンコーディング)、expression list (式リスト)、format(フォーマット)、end-of-line (EOL) sequence (行末シーケンス)などの共通要素を持っています。

I/Oトランザクション・フォーマット

データを書き込むI/Oトランザクションは、通常、次のフォーマットを使用します。

<action> <encoding> <expression list> <format> <EOL>

表18は、最もよく行われる操作、READ、WRITE、EXECUTE、WAITについて説明したものです。

表18　　I/Oトランザクションの種類

操作	説明
READ	指定されたエンコードとフォーマットを使用して、指定されたソースからデータを読み取ります。
WRITE	指定されたエンコード・フォーマットを使用して、指定されたターゲットにデータを書き込みます。
EXECUTE	特定のコマンドを実行します。たとえば、EXECUTE REWINDは、ファイルの内容を消去しないで、ファイル読み取りポインタまたはファイル書込みポインタの位置をファイルの先頭に変更します。EXECUTE CLOSEはオープンしているファイルをクローズします。
WAIT	指定された秒数だけ待ってから、次のトランザクションを開始します。

注 記　**[I/O] ⇒ [Advanced I/O Operations]にも多数の操作があります。これらの操作については、メニューでオブジェクトを探して確認できます。**

エンコードとフォーマットは、データをパッケージにして送信する方法を指します。たとえば、TEXTエンコードでは、データをASCII文字として送信します。TEXTエンコードはさまざまな方法でフォーマットできます。たとえば、文字と数字から成る文字列をファイルに送信する場合、WRITE TEXT STRINGトランザクションを使用すると、ASCII文字で表現した文字列全体が送信されます。WRITE TEXT REALトランザクションは、同じ文字列から実数だけを抽出し、個々の数字に対してASCII文字を使用して、それらの実数を送信します。表19は、エンコードについて簡単に説明したものです。

表19　I/Oトランザクション・エンコード

エンコード	説明
TEXT	編集やほかのソフトウェア・アプリケーションへの移植が容易な、人が読むことのできる形式(ASCII)で、すべてのデータ型を読み取ったり、書き込みます。VEE数値データは自動的にテキストに変換されます。
BYTE	数値データを2進整数に変換し、最下位バイトを送信または受信します。
CASE	列挙値や整数を文字列にマッピングし、その文字列を読み取り/書き込みます。たとえば、CASEを使用すると、エラー番号を受け入れて、エラー・メッセージを書き込むことができます。
BINARY	すべてのデータ型をマシン固有のバイナリ・フォーマットで処理します。
BINBLOCK	バイナリ・ファイル内のすべてのVEEデータ型についてIEEE488.2で定義されている長さのブロック・ヘッダを使用します。
CONTAINER	すべてのデータ型についてVEE固有のテキスト・フォーマットを使用します。

書込みトランザクションの場合、式リストは、送信するデータを生成するために評価する必要がある式を単にカンマで区切ったリストです。式は、算術式、データ入力端子名、文字列定数、VEE関数、UserFunction、グローバル変数で構成されます。読み取りトランザクションの場合、式リストは、データを読み取るときにその保管場所を示す出力端子名の、カンマ区切りのリストで構成する必要があります。

計測器からのデータの読み取りとデータ・フォーマットについては、第3章「簡単な計測器の操作」(127ページ)を参照してください。これらのフォーマットのほとんどは、すべてのI/Oトランザクションに適用されます。

ほとんどの**[I/O]** ⇒ **[To]**オブジェクトのオブジェクト・メニューでは、[Properties...]を選択し、次に[Data Format]を選択して、EOLシーケンスを指定できます。また、[Separator Sequence]で変更することもできます。

例題5-3: To/From Fileオブジェクトの使用

この例題では、テスト・データをファイルとやり取りするプロセスについて説明します。ここでは、3つの共通のテスト結果項目であるテスト名、タイム・スタンプ、Real値の1次元配列の保管と読み取りを行います。すべてのVEEデータ型に同じプロセスが適用されます。

ファイルへのテキスト文字列の送信

1 **[I/O]** ⇒ **[To]** ⇒ **[File]**を選択します。エントリを次のように設定します。

表20　　ファイルへのテキスト文字列の送信

操作	説明
filename	デフォルト・ファイルのmyFileを使用します。デフォルト・ファイルを変更するには、[To File]入力フィールドをクリックして、ホーム・ディレクトリ内のファイルを表示するリスト・ボックスを開きます。
Clear File At PreRun & Open	このボックスにチェック・マークを付けます。特に指定しないかぎり、VEEは、新しいデータを既存のファイルの末尾に追加します。このボックスにチェック・マークを付けると、ファイルがクリアされてから、新しいデータが書き込まれます。

2 トランザクション領域内をダブルクリックすると、[I/O Transaction]ダイアログ・ボックスが表示されます。必要に応じて、図128と図129を参照してください。

WRITE TEXT an EOLはデフォルト・トランザクションです。このトランザクションは、TEXTエンコードと指定された行末シーケンスを使用して、ピンにデータを書き込みます。VEEでは大文字と小文字の区別は必要ありません。データ入力端子名とデータ出力端子名には、大文字と小文字のどちらの文字列でも使用できます。

エントリを次のように設定します。

表21　デフォルト・トランザクションの記述

エントリ	説明
a (式フィールド)	式リスト・フィールドは、強調表示されており、デフォルトはaです。「"Test1"」と入力してから、**[OK]**をクリックします。Text文字列であることを示すには、引用符が必要です。引用符を付けないで「Test1」と入力すると、VEEは、これを端子名またはグローバル変数名と解釈します。
WRITE	デフォルトの**WRITE**を使用します。
TEXT	デフォルトの**TEXT**を使用します。**TEXT**エンコードは、ASCII文字を使ってデータを送信します。
DEFAULT FORMAT	**DEFAULT FORMAT**を使用します。**DEFAULT FORMAT**は、**STRING**などの適切なVEEフォーマットを選択します。
EOL ON	デフォルトを使用します。デフォルトのEOLシーケンスは、新しい行のためのエスケープ文字\nです。

3 **[OK]**をクリックして、To Fileオブジェクトに戻ります。トランザクション・バーには、ステートメントWRITE TEXT "Test1" EOLが表示されます。このトランザクションは、文字列Test1を指定されたファイルに送信します。

ファイルへのタイム・スタンプの送信

[Device] ⇒ **[Function & Object Browser]** ⇒ **[Time & Date]**カテゴリの中の関数now()は、Real64 Scalarで表示された現在の時刻を返します。Real値は、西暦0001年1月1日00時00分から経過した秒数です。

したがって、now()は約63Gの値を返します。VEEがこのフォーマットを提供する理由は、数学的に操作しやすく、記憶域を節約できるためです。タイム・スタンプを読みやすいフォーマットで保管するには、To Fileオブジェクト内のTIME STAMP FORMATを使用します。タイム・スタンプをファイルに送信するには、次の手順に従います。

1. 同じTo Fileオブジェクト内で、トランザクション領域の中をダブルクリックし、[I/O Transaction]ボックスを表示します。
2. 式リスト入力フィールドをダブルクリックして[a]を強調表示し、「now()」と入力します。now()関数は、コンピュータ・クロックの現在の時刻をRealフォーマットで送信します。
3. RealフォーマットをTime Stamp Formatに変更します。
   [DEFAULT FORMAT]の横の矢印をクリックしてドロップダウン・メニューを表示し、[TIME STAMP FORMAT]を選択します。[I/O Transaction]ダイアログ・ボックスに追加エントリが表示されます。エントリを次のように設定します。

表22　　タイム・スタンプのフォーマット

タイム・スタンプ	説明
Date & Time	ドロップダウン・メニューで[Time]を選択します。
HH:MM:SS	クリックして、時、分、秒のフォーマットから[HH:MM](時、分のフォーマット)にトグルします。
24 HOUR	クリックして、24時間のフォーマットから**[12 HOUR]**(午前と午後のフォーマット)にトグルします。

[I/O Transaction]ダイアログ・ボックスは、図130のように表示されます。

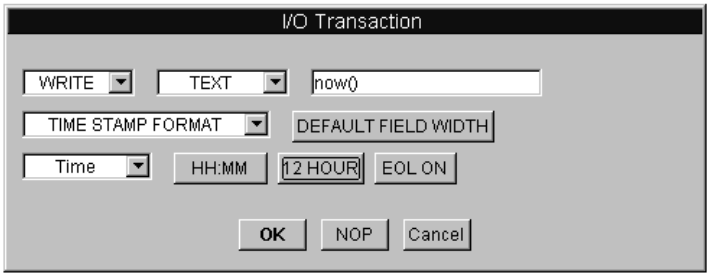


図130　[TIME STAMPのI/O Transaction]ダイアログ・ボックス

4 **[OK]**をクリックして、To Fileボックスに戻ります。2番目のトランザクション・バーには、現在、ステートメントWRITE TEXT now() **TIME: HM:H12 EOL**が表示されているはずです。

ファイルへのReal配列の送信

For Countオブジェクトと Collectorオブジェクトを使用して、4つの要素を持つ1次元の配列を作成し、myFileに追加します。

1 **[Flow]** ⇒ **[Repeat]** ⇒ **[For Count]**を選択します。For Countのデフォルト値を4に変更します。

2 **[Data]** ⇒ **[Collector]**を選択します。[Collector]をダブルクリックしてオープン・ビューに切り替えます。For Countのデータ出力をCollectorのデータ入力(一番上の入力ピン)に接続します。For Countシーケンス出力ピンをCollectorのXEQ ピン(下部にある入力ピン)に接続します。Collectorをアイコン化します。

3 Collectorは配列[0, 1, 2, 3]を作成するので、それをデータ・ファイルに送信します。

4 同じTo Fileオブジェクトを使用して、トランザクション領域をダブルクリックします。[I/O Transaction]ダイアログ・ボックスで[DEFAULT FORMAT]メニューをオープンし、[REAL64 FORMAT]を選択します。

5 [I/O Transaction]ダイアログ・ボックスは、REAL64 FORMATを選択するための追加ボタンを表示します。デフォルトの選択肢のままでよいのですが、参考のために、デフォルト以外の選択肢を調べることもできます。

6 **[OK]**をクリックして[I/O　Transaction]ボックスをクローズします。To Fileオブジェクト内のトランザクション・バーには、現在、ステートメントWRITE TEXT a REAL64 STD EOLが表示されているはずです。VEEが入力端子aも自動的に追加します。

7 Collectorからの出力をTo　Fileの入力**a**に接続します。プログラムは図131のように表示されます。設定した[I/O Transaction]ボックスも表示されます。

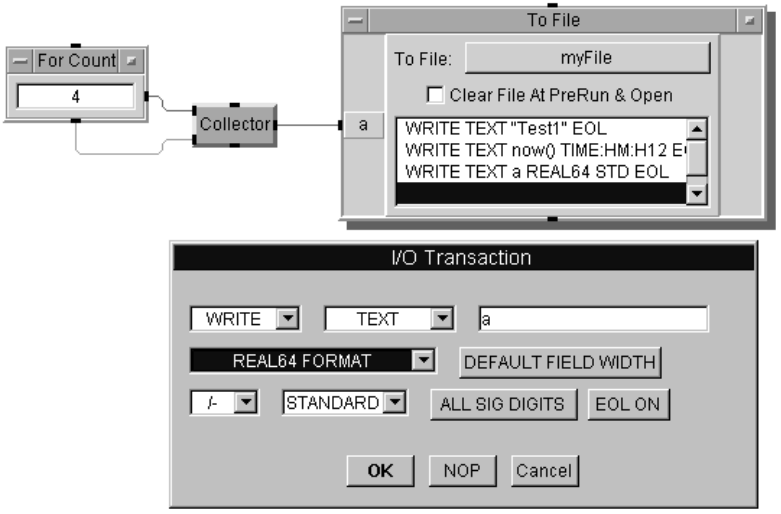


図131　　To Fileオブジェクトを使ったデータの保管

From Fileオブジェクトを使ったデータの読み取り

From Fileオブジェクトを使ってデータを読み取る場合は、そのデータがどのようにして保管されたかを知っておく必要があります。

注 記

To DataSetまたはFrom DataSetを使ってもデータの保管と読み取りができます。この2つのオブジェクトを使用する場合は、ファイル内のデータの型を知っている必要はありません。データセットについては、232ページの「DataSetを使ってRecordを保管および読み取る方法」を参照してください。

この例では、String Formatのテスト名、Time Stamp Format型のタイム・スタンプ、Real64数の配列の順で保管されます。そのデータをが読み取る場合は、From Fileで3つのトランザクションを作成します。

1 **[I/O]** ⇒ **[From]** ⇒ **[File]**を選択し、To Fileオブジェクトの下に配置します。

2 To Fileオブジェクトのシーケンス出力ピンをFrom Fileオブジェクトのシーケンス入力ピンに接続します。

このようにシーケンス・ピンを接続すると、From Fileは、To FileオブジェクトがmyFileへのデータの送信を完了してから、データの抽出を開始します。

3 From Fileオブジェクトでは、デフォルトのデータ・ファイルはmyFileのままにしておきます。トランザクション・バーをダブルクリックして、[I/O Transaction]ダイアログ・ボックスを表示します。[REAL64 FORMAT]をクリックし、図132に示すように、[STRING FORMAT]に変更します。

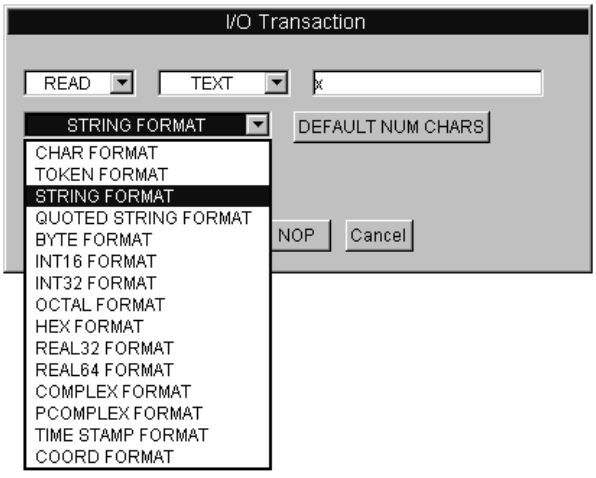


図132　　文字列フォーマットの選択

4 そのほかのデフォルト値はすべて適正であるため、**[OK]**をクリックして[I/O Transaction]ボックスをクローズします。From Fileオブジェクト内のトランザクション・バーには、現在、ステートメントREAD TEXT x STRが表示されているはずです。

さらに2つのトランザクションを追加して、タイム・スタンプとreal配列を読み取ります。

5 同じFrom Fileオブジェクト内で、最初のトランザクション・バーの下をダブルクリックします。[I/O Transaction]ダイアログ・ボックスが表示されます。2番目のトランザクションでピン**y**にデータを読み取る場合は、式リスト入力フィールドをダブルクリックして**[x]**を強調表示し、「y」と入力します。このピンを**x**と、2番目のトランザクションは、最初のトランザクションが**x**に入力したデータに追加しないで、データを上書きしてしまいます。そのデータに追加するのではありません。[REAL64 FORMAT]を[STRING FORMAT]に変更してから、**[OK]**をクリックします。

注 記

タイム・スタンプをテキスト文字列として読み取るには、STRING FORMATを使用します。TIME STAMP FORMATは、タイム・スタンプのデータを変換して実数に戻します。

6 同じFrom Fileオブジェクト内で、2番目のトランザクション・バーの下をダブルクリックし、[I/O Transaction]ダイアログ・ボックスを表示します。エントリを次のように設定します。

表23　　I/Oトランザクションのエントリ

エントリ	説明
(式フィールド)	**x**をzに編集します。これで、Real配列がZ出力端子に読み取られます。
SCALAR	**[SCALAR]**を**[ARRAY 1D]**に変更します。
SIZE:	現在、[I/O Transaction]ボックスには**[SIZE]**ボタンが追加されています。この例題では、配列は4つの要素を持ちます。**10**を[4]に変更し、**[OK]**をクリックします。

注 記

配列のサイズがわからない場合は、[SIZE]を[TO END]にトグルします。これにより、が正確なサイズを知らなくても、ファイルの最後までのデータが読み取られます。たとえば、この機能を使用すると、ファイルの全内容を文字列配列として読み取って、ファイルの内容を検査できます。

From Fileオブジェクト内のトランザクション・バーには、現在、READ TEXT y STRとREAD TEXT z REAL64 ARRAY:4のステートメントが表示されているはずです。VEEがx、y、zのデータ出力端子を自動的に追加することに注目してください。入力端子と出力端子は手動で追加または削除することもできます。それには、オブジェクト・メニューの**[Add Terminal]**と**[Delete Terminal]**、またはショートカットの`Ctrl+A`キーと`Ctrl+D`キーを使用します。

7 **[Display]** ⇒ **[AlphaNumeric]**を選択し、2つクローンを作成して3つのオブジェクトを表示します。3つのAlphaNumericオブジェクトをFrom Fileの3つのデータ出力ピンに接続します。オブジェクトのいずれかの隅をクリック・アンド・ドラッグして、配列表示を拡大します。

AlphaNumericオブジェクトの表示サイズは、最初にメニューで選択してオブジェクトを作成したときに、オブジェクトの輪郭線をクリック・アンド・ドラッグして変更することもできます。

8 プログラムを実行します。図133のように表示されます。

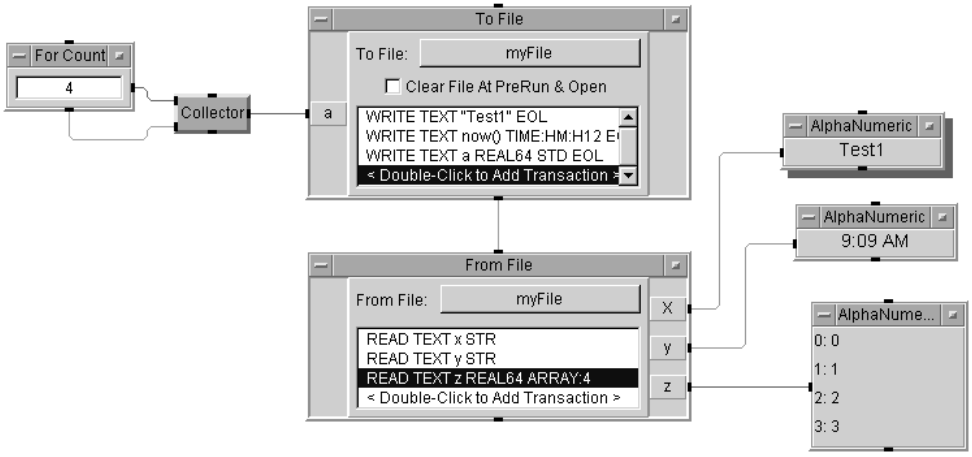


図133　　From Fileオブジェクトを使ったデータの読み取り

最初のAlphanumericがタイトルを表示し、2番目のAlphanumericがテスト時刻を表示し、3番目のAlphanumericが配列内の数字をリストしています。

Recordを使った混用データ型の保管

Recordデータ型を使用すると、単一のデータ・コンテナに複数の異なるデータ型を保管できます。レコードには、のどのようなVEEデータ型でも入れることができます。データは、スカラでも配列でもかまいません。単一のデータ構造内に、テスト名、タイム・スタンプ、実数配列を保管できます。

Record内の個々の要素は、フィールドとして保管され、ドット表記を使ってアクセスできます。たとえば、Rec.Nameを使用すると、Recという名前のRecord内のNameという名前のフィールドにアクセスできます。レコードから成る配列では、Rec[2].Nameは、配列内の3番目のレコードのNameフィールドを意味します。配列のインデックスは0から始まります。

Recordデータ型を使ってテスト・データを構造化することには、次の利点があります。

- 単一のコンテナ内で、混用データ型を論理グループ化できます。このため、プログラムの開発と保守がより容易になります。たとえば、テスト・データを保管するレコードには、テスト名、戻り値、成功/失敗インジケータ、タイム・スタンプ、公称期待値、成功の上限、成功の下限、およびテストの説明を示すフィールドがあります。
- 8つの別々のデータ・コンテナではなく、単一のデータ・コンテナを操作できます。これにより、プログラムが単純になり、読みやすくなります。
- VEEでは、DataSetのRecordの保管と読み取りができます。DataSetは、レコードを保管するために作成する特殊なファイルです。DataSetからレコードを読み取る場合、データ型を知っている必要はありません。VEEは、DataSetに保管している情報を読み取り、ソートし、検索するためのオブジェクトを提供します。

例題5-4: Recordの使用

この例題では、Recordデータ型の使用方法について説明します。ここで習得するのは、レコードを構築する方法、そのレコード内の特定のフィールドを読み取る方法、選択されたフィールドを設定する方法、レコード全体を単一の手順で解体(unbuild)する方法です。また、タイム・スタンプ関数now()の別の使用方法についても説明します。

Recordの構築

3つのフィールドを持つRecordを構築します。3つフィールドとは、Stringとして保管するテスト名、Real Scalarとして保管するタイム・スタンプ、4つの要素から成るArray of Realsとして保管するシミュレートされたテスト結果です。次の例題でこれらのフィールドを読み取る際に、タイム・スタンプをさまざまなフォーマットに変換して表示できることがわかります。

[Data] ⇒ **[Constant]** ⇒ **[Text]**を選択し、入力フィールドで「Test1」と入力してテスト名を作成します。オブジェクトの名前をText Constantに変更します。Text Constantをアイコン化します。

9 **[Device]** ⇒ **[Function & Object Browser]**を選択します。[Type]で[Built-in Functions]をクリックし、[Category]で[Time & Date]をクリックし、[Functions]で[now]を選択し、**[Create Formula]**をクリックします。オブジェクトをText Constantの下に配置します。

10 **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**を選択し、now()の下に配置します。

11 Real64オブジェクトのメニューで**[Properties...]**をクリックし、[1D Array]を選択すると、このScalar Real64をArray 1Dに変えることができます。

12 **[Real64]**のタイトル・バーをダブルクリックして[Constant Properties]ボックスをオープンします。[Configuration]で[1D Array]を選択し、[Size]を[4]に変更し、次に**[OK]**をクリックします。

この配列に4つの値を入力します。それには、要素[0000]の横をダブルクリックして、最初のエントリを強調表示します。次の値に移るときには[Tab]キーを使用しながら、「2.2」、「3.3」、「4.4」、「5.5」を入力します。Real64をアイコン化します。

13 **[Data]** ⇒ **[Build Data]** ⇒ **[Record]**を選択し、ほかの3つのオブジェクトの右側に配置します。3つのフィールドを入力できるように、3番目のデータ入力端子を追加します。端子をダブルクリックしてそれぞれの端子をオープンし、3つの入力端子の名前をtestname、time、dataに変更します。

Build RecordオブジェクトのOutput ShapeはScalarとArrayをトグルします。通常デフォルトのScalarを使用します。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

14 Text Constantオブジェクトを Build Recordオブジェクトのtestname端子に、now()オブジェクトをtime端子に、Real64オブジェクトをdata端子に接続します。

15 プログラムを実行します。Recordデータ出力端子をダブルクリックして、レコードを検査します。図134のように表示されます。

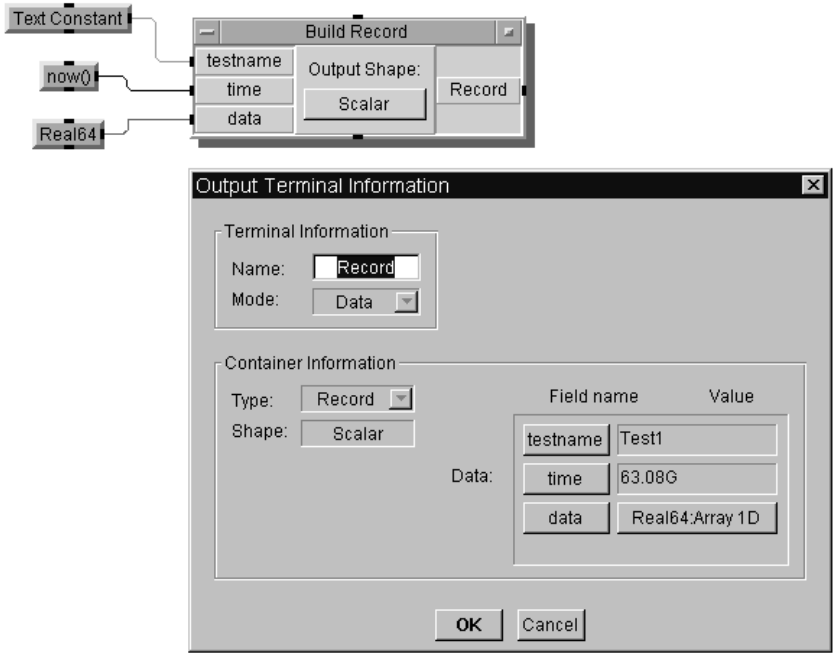


図134　　Recordに関する出力端子情報

3つのフィールドとその値が表示されています。**[Real64: Array 1D]**ボタンをクリックすると、リスト・ボックスに現在の値が表示されます。タイム・スタンプはReal64 Scalarとして保管されています。次の例題では、タイム・スタンプを読みやすい形式に変換します。**[OK]**をクリックして[Output Terminal Information]ダイアログ・ボックスをクローズします。プログラムに`records.vee`と名前を付けて保存します。

Recordからのフィールドの取得

Get Fieldオブジェクトを使用して、レコードから3つのフィールドを抽出し、それぞれの値を表示します。

1. **records.vee**プログラムを開きます。
2. **[Data]** ⇒ **[Access Record]** ⇒ **[Get Field]**を選択します。タイトルがrec.fieldであるオブジェクトが表示されます。

   [rec]というラベルの付いたデータ入力は、フィールドの数と型に関係なく、どのようなレコードでも受け入れます。Rec.fieldは、入力フィールドのデフォルト選択ですが、これを編集すると、どのフィールドでも読み取れます。Recは、Recという名前のデータ入力端子にあるレコードを指します。VEEでは大文字と小文字を区別をする必要は**ありません**。

注 記　**Get Fieldオブジェクトは、Function & Object Browser内の公式と同じように、入力と式を使って設定するFormulaです。**

3. rec.fieldのクローンを2つ作成し、3つともBuild Recordの右に配置します。
4. Build Recordデータ出力を3つのrec.fieldオブジェクトすべてに接続します。
5. 3つのフィールドはtestname、time、dataとして保管されるので、rec.fieldオブジェクトを編集して適切なフィールドを取得する必要があります。
6. 3つのrec.fieldオブジェクト式フィールドをrec.testname、rec.time、rec.dataに編集します。
7. **[Display]** ⇒ **[AlphaNumeric]**を選択し、2つのクローンを作成します。3つのAlphaNumericを3つのrec.fieldオブジェクトに接続します。real配列を保持できるように、3番目の表示サイズをほかのオブジェクトの約3倍の長さに変更します。

8 2 番目の AlphaNumeric 表示のオブジェクト・メニューをオープンし、[Properties]を選択し、次に、[Number]フォルダを選択します。[Global Format]の左をクリックしてチェック・マークを外します。

表示フォーマットを設定します。[Real]セクションで[Standard]メニューをオープンします。[Time Stamp]を選択し、**[OK]**をクリックします。

9 **[HH:MM:SS]**をクリックして**[HH:MM]**にトグルします。**[24 HOUR]**をクリックして**[12 HOUR]**にトグルします。図135を参照してください。

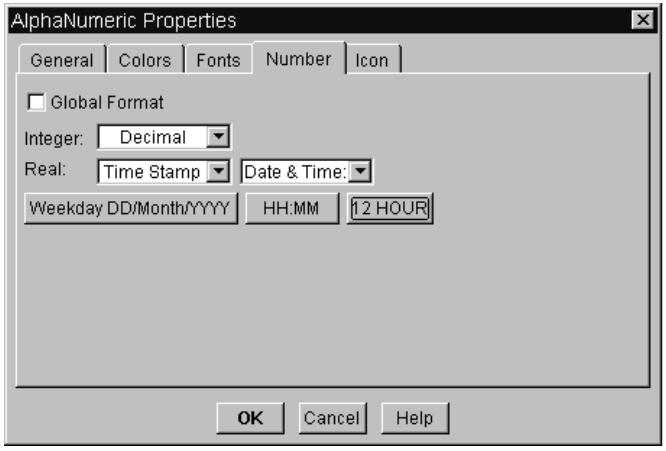


図135　[AlphaNumeric Properties]ボックス

10 プログラムを実行し、**getfield.vee**としてセーブします。プログラムは図136のように表示されます。

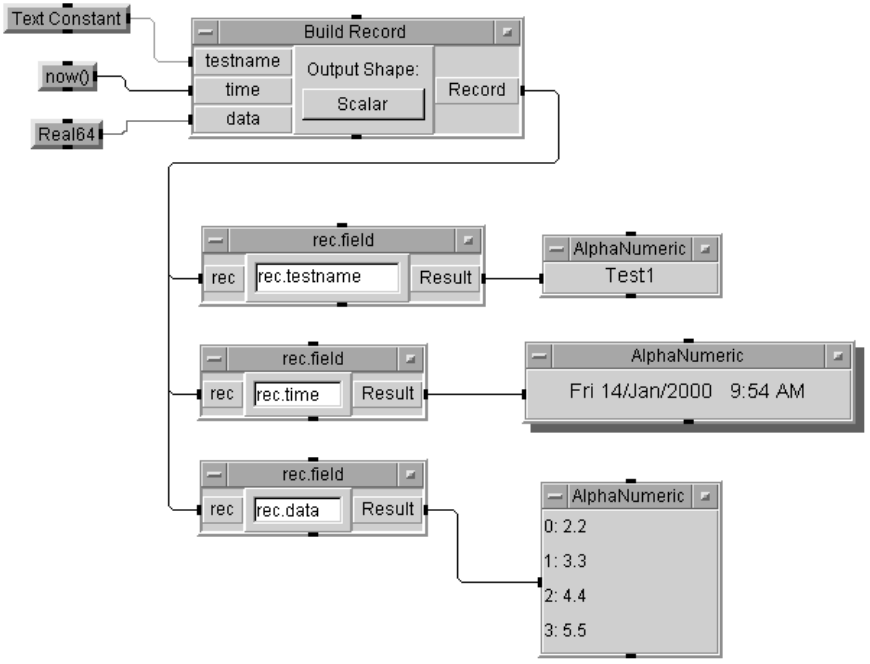


図136　Get Fieldオブジェクトの使用

2番目の表示には、曜日、日付、および時、分、午前または午後の形式で時刻が記載されています。

Record内のフィールドの設定

この例題では、レコードの特定のフィールド内のデータを変更する方法について説明します。

注 記　別のテストで同じRecordを再利用できます。

1 **getfield.vee**プログラムを開きます。

2 Build Recordの後のすべてのオブジェクトを選択し、Ctrl+Xキーを押して削除します。

3 **[Data]** ⇒ **[Access Record]** ⇒ **[Set Field]**を選択し、**Build Record**の右に配置します。Build Recordからの出力をSet Fieldのrec入力に接続します。タイトルは**rec.field = b**となります。

   Set Fieldは、代入記号(=)の右辺の式を左辺に代入します。したがって、recの指定されたフィールドは、右辺の値で変更されます。レコードの残りの部分は変更されません。入力レコードをrecに、新しい入力値を**b**に接続します。変更されたレコードは、recというラベルの付いたデータ出力端子に出力されます。

注 記　**Set Fieldオブジェクトは、Function & Object Browser内の公式と同じように、入力と式を使って設定するFormulaです。**

4 式をrec.data[*]=bに編集して、データ・フィールドの4要素から成る配列の値を変更します。このレコードのフィールドで配列全体を変更するので、配列[*]表記を使用する必要があります。配列の新しい値を入力端子bに入力します。

5 **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**を選択し、Build Recordオブジェクトの下に配置します。オブジェクト・メニューを開き、[Properties]を選択します。[Configuration]で[1D Array]を選択し、**[Size]**を[4]に編集し、**[OK]**をクリックします。

   レコードのフィールドの新しい値を配列に保持する場合、配列のサイズは、現在の配列のサイズと同じでなければなりません。

   最初のエントリを強調表示し、Tabキーで後続のエントリに移動しながら、「1」、「2」、「3」、「4」を[Real64]に入力します。最後の入力の後に、Tabキーを押さないでください。Real64を[b]というラベルの付いた(rec.field=bというタイトルの)Set Field入力に接続します。

   Get Fieldオブジェクトを使用して、そのレコードからフィールドrec.dataを抽出し、結果を表示します。

6 **[Data]** ⇒ **[Access Record]** ⇒ **[Get Field]**を選択し、Set Field (rec.field=b)オブジェクトの下に配置します。Get Fieldオブジェクトの式をrec.fieldからrec.dataに編集します。**rec.field = b**のデータ出力をrec.fieldのデータ入力に接続します。

注 記

式フィールドで、**Formula**オブジェクトと**A.data**を併用することもできます。

7 AlphaNumeric表示を選択し、配列を保持できるようにサイズを変更し、rec.field出力ピンに接続します。

8 プログラムを実行し、**setfield.vee**としてセーブします。プログラムは図137のように表示されます。

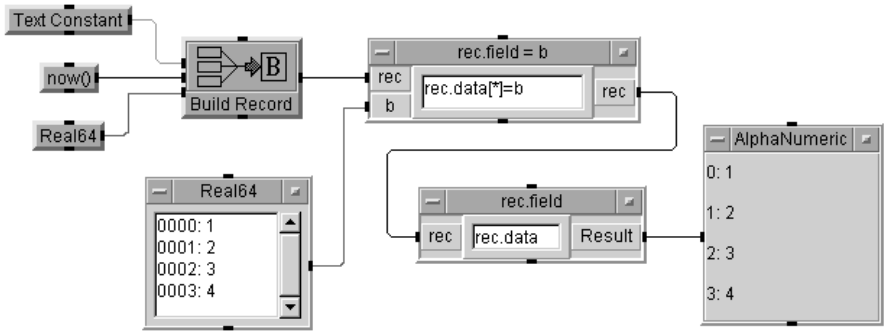


図137　Set Fieldオブジェクトの使用

この例で示しているように、Recordフィールドはすべて変更できます。フィールドの一部を変更することもできます。たとえば、**rec.field = b**内の式をrec.data[1]=20に変更してみてください。次に、**rec.field = b**の入力**b**を削除します。プログラムをもう一度実行すると、配列**[2.2, 20, 4.4, 5.5.]**が表示されます。

単一手順でのRecordのUnbuild方法

レコードのすべてのフィールドを抽出し、そのフィールドの名前と型のリストを取得するには、UnBuild Recordオブジェクトを使用します。

1. **setfield.vee**プログラムを開きます。Build Recordの後のすべてのオブジェクトを削除します。
2. **[Data]** ⇒ **[UnBuild Data]** ⇒ **[Record]**を選択し、Build Recordの下に配置し、オープン・ビューに切り替え、Build Recordの出力をUnBuild Recordの入力に接続します。UnBuild Recordにデータ出力ピンをもう１つ追加し、**A**、**B**、**C**出力のフィールド名をtestname、time、dataに変更します。
3. AlphaNumeric表示を選択し、4つクローンを作成します。5つの表示をUnBuild Recordの5つの出力端子に接続します。配列を保持できるように、Name List、Type List、dataの表示を拡大する必要があります。さらに、日/月/年、12時間制の時、分表記で時刻を表示するように再設定します。
4. プログラムを実行し、**unbuild.vee**としてセーブします。図138のように表示されます。

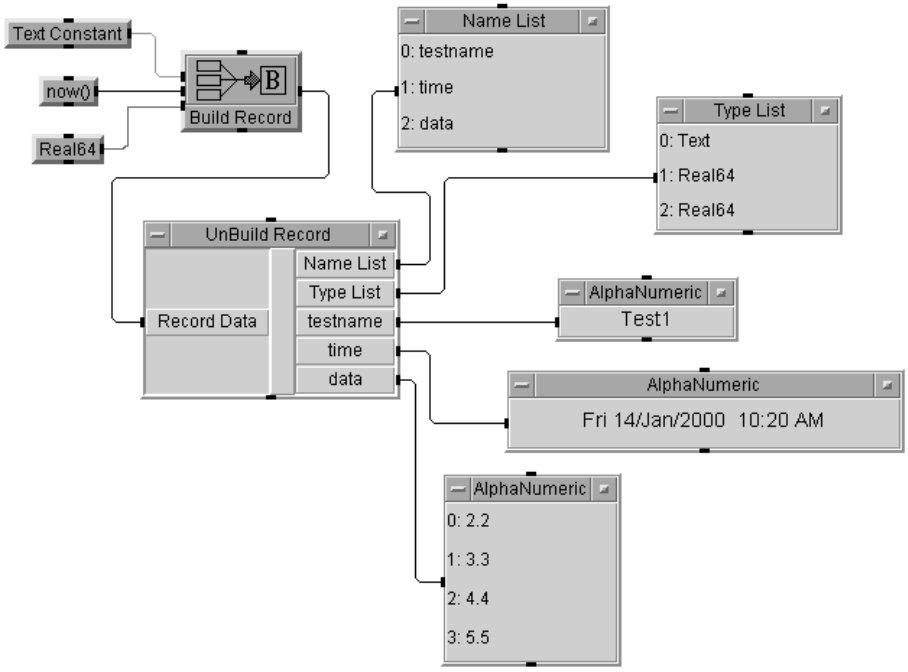


図138　　UnBuild Recordオブジェクトの使用方法

Type Listによって**testname**を**Text**型、**time**と**data**を**Real64**型と識別しているのと同様に、**Name List**ピンによってレコード内の3つのフィールドの名前、**testname**、**time**、**data**を指定しています。

DataSetを使ってRecordを保管および読み取る方法

DataSetは１つ以上のレコードを読み取ることができます。VEEオブジェクトはレコードをアンパックします。したがって、ファイルではなくDataSetにレコードを保管すれば、データ型を覚えておく必要がなくなります。データに対してソートや検索を行って、自分専用のカスタマイズされたテスト・データベースを作成することもできます。

例題5-5: Recordの使用方法

DataSetは、単にファイルに保管されたRecordの配列です。この例題では、DataSetにデータを保管したり読み取る方法について説明します。

DataSetにRecordを保管したり読み取る方法

この例題では、テスト名、Real64 Scalar、Realの配列を3つのフィールドとして保持する、10個のRecordから成る配列を作成します。また、Recordの配列をDataSetに保管し、そのレコードを読み取って表示します。

1. **[Flow]** ⇒ **[Start]**を選択します。**[Flow]** ⇒ **[Repeat]** ⇒ **[For Count]**を選択し、Startの下に配置します。**[Device]** ⇒ **[Formula]**を選択し、そのオブジェクトをFor Countの右に配置します。StartをFor Countのシーケンス入力ピンに接続し、For Countデータ出力ピンをFormulaデータ入力ピンに接続します。
2. Formula 式フィールドをダブルクリックして、デフォルトの式を強調表示し、次に、「`"test" + a`」と入力します。

   [Start]をクリックすると、For Countオブジェクトは、整数0から9までを順にFormulaの**A**ピンに出力します。Formulaオブジェクトでは、これらの整数が`test`に追加され、**Text Scalars**である**test0, test1, test2,..., test9**が出力されます。これらの値は、10個のRecordの最初のフィールドに書き込まれます。
3. **[Data]** ⇒ **[Build Data]** ⇒ **[Record]**を選択し、Formulaの右に配置します。データ入力ピンを1つ追加します。Formulaのデータ出力を**Build Record**の**A**入力に接続します。

4 ツールバーで[Function & Object Browser]アイコンを選択します。

   a [Built-in Functions]、[Probability & Statistics]、[random]を選択し、random (low, high)オブジェクトを作成します。このオブジェクトをFormulaオブジェクトの下に配置します。

   b 入力端子を削除し、入力パラメータを[low]から[0]に、[high]から[1]に変更します。

   c オブジェクトRandom Numberの名前を変更し、そのデータ出力をBuild Recordの**B**端子に接続します。

5 Formulaシーケンス出力ピンをRandom Numberのシーケンス入力ピンに接続します。シーケンス・ピンを互いに接続すると、プログラムを実行するたびに、新しい乱数が特定のレコードの**B**フィールドに必ず入力されるようになります。

6 **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**を選択します。Real64オブジェクトをFormulaオブジェクトの下に配置します。

   a オブジェクト・メニューを開き、**[Properties]**をクリックします。タイトルに「Real Array」と入力し、[Configuration]で**[1D Array]**をクリックし、**[Size]**を[3]に変更します。**[OK]**をクリックします。

   b 配列内の各エントリをダブルクリックして強調表示し、数字の「1」、「2」、「3」を入力します。

   c Real Arrayデータ出力をBuild RecordのC端子に接続します。

7 **[I/O]** ⇒ **[To]** ⇒ **[DataSet]**を選択し、Build Recordの下に配置します。Build Recordのデータ出力をそのデータ入力に接続します。デフォルトのファイル**myfile**はそのままにしておき、[Clear File At PreRun]にチェック・マークを付けます。

8 プログラムを実行します。図139に示すように、10個のレコードから成る配列が**myFile**という名前のDataSetに書き込まれます。

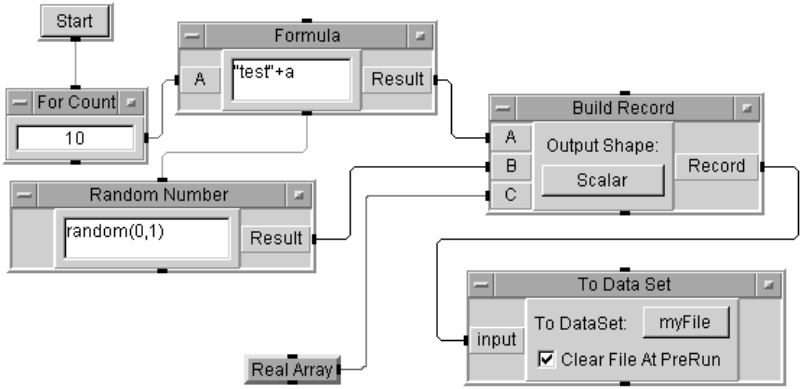


図139　　Recordから成る配列のDataSetへの保管方法

次に、From DataSetとRecord Constantオブジェクトを使用して、レコードから成る配列を読み取って表示します。

9 **[I/O]** ⇒ **[From]** ⇒ **[DataSet]**を選択し、For Countの下に配置します。デフォルトのファイル名**myFile**はそのままにしておきます。**[Get Records]**フィールドをクリックし、[One]から[All]にトグルします。下部の式フィールドのデフォルト値[1]はそのままにしておきます。

VEEは、これらの設定を使用して、**myFile**内のDataSetを調べ、式フィールド内の条件に一致するすべてのレコードを見つけます。Get RecordsをOneに設定した場合、VEEは式フィールド内の条件に一致する最初のレコードを出力します。1は、すべてのレコードが条件に一致していることを意味する**TRUE**状態を表すので、そのファイル内のレコード配列全体が、Recというラベルの付いた出力ピンに出力されます。式フィールドのそのほかの使用方法については、ほかの例題で説明しています。詳細は、オブジェクト・メニューのヘルプを参照してください。

For Countシーケンス出力ピンをFrom Data Setオブジェクトのシーケンス入力に接続します。これにより、プログラムの中の**myFile**にデータを送信する部分は、必ず、ファイルからデータが読み取られる前に、実行されます。[Show Data Flow]をオンにすると、イベントの順序を表示できます。

10 **[Data]** ⇒ **[Constant]** ⇒ **[Record]**を選択し、**To Data Set**の下に配置します。オブジェクト・メニューを開き、**[Add Terminal]** ⇒ **[Control Input]**を選択します。表示されるリスト・ボックスで[Default Value]をクリックし、次に、**[OK]**をクリックします。Recordオブジェクトのサイズを大きくすると、プログラムを実行したときに結果を表示できます。

受信したレコードがデフォルト値となります。この場合、Recordは、From Data Setオブジェクトからレコード配列を受信し、自分でフォーマットしてそのレコード配列を表示します。

11 From Data Setの出力ピンRecをDefault Valueピンに接続します。この端子を表示するには、オブジェクト・メニューをオープンして[Properties]を選択し、次に[Show Terminals]を選択してから**[OK]**を選択します。From Data SetとRecordを結ぶ破線が表示されます。

注 記　**オブジェクトを相互に結ぶ破線は、コントロール・ラインを表します。**

12 プログラムを実行し、**dataset1.vee**としてセーブします。プログラムは図140のように表示されます。

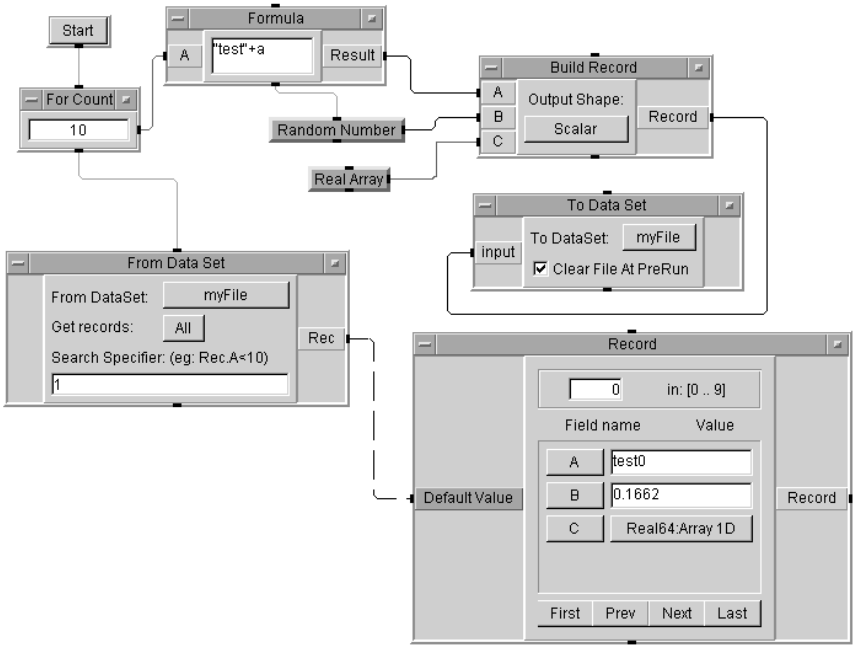


図140　　DataSetを使ってRecordを保管および読み取る方法

注 記

From Data Setオブジェクトは、条件に一致するレコードを少なくとも1つ保持する必要があります。そうでない場合は、エラー・メッセージが表示されます。エラーを避けるには、条件に一致するレコードがなければアクティブ化するオブジェクトに、EOF (end-of-file)出力ピンを追加します。そうすれば、EOFになったときの動作をプログラムに追加できます。

単純なテスト・データベースのカスタマイズ方法

DataSetを検索またはソートすると、テスト名、タイム・スタンプ、テスト・パラメータ、テスト値、成功/失敗インジケータ、テストの説明などの情報を取得できます。したがって、DataSetレコードはテスト・データベースの役割を果たすことができます。情報を検索するには、From Data Setオブジェクトを次のように使用できます。

- From Data Setオブジェクト内の式フィールドは、検索操作に使用されます。
- 関数 sort() を使用すると、指定されたフィールドを使用してレコードをソートできます。

例題5-6: DataSetの場合の検索操作とソート操作の使用方法

この例題では、DataSetを検索して情報を取得する方法、検索操作のためのオペレータ・インタフェースを作成する方法、ソート操作をプログラミングする方法を習得します。

DataSetを検索する方法

1. **dataset1.vee**プログラムを開きます。
2. From Data Setオブジェクトの下部の式フィールドをダブルクリックして、現在の式**[1]**を強調表示します。「**Rec.B>=0.5**」と入力します。From Data Setオブジェクトは、フィールド**B**(たとえば乱数)が、0.5と比較して、大きいか等しいすべてのレコードを出力します。
3. 式フィールド内の条件に一致するレコードが1つもない場合に起動するEOFピンを追加します。From Data Setオブジェクトのデータ出力領域にカーソルを置き、Ctrl+Aキーを押します。図141に示すように、From Data SetオブジェクトにEOF出力ピンが追加されます。

注 記

EOFピンを追加する場合、オブジェクト・メニューをオープンし、[Add Terminal] ⇒ [Data Output...]をクリックすることもできます。

4. プログラムを実行し、**dataset2.vee**としてセーブします。

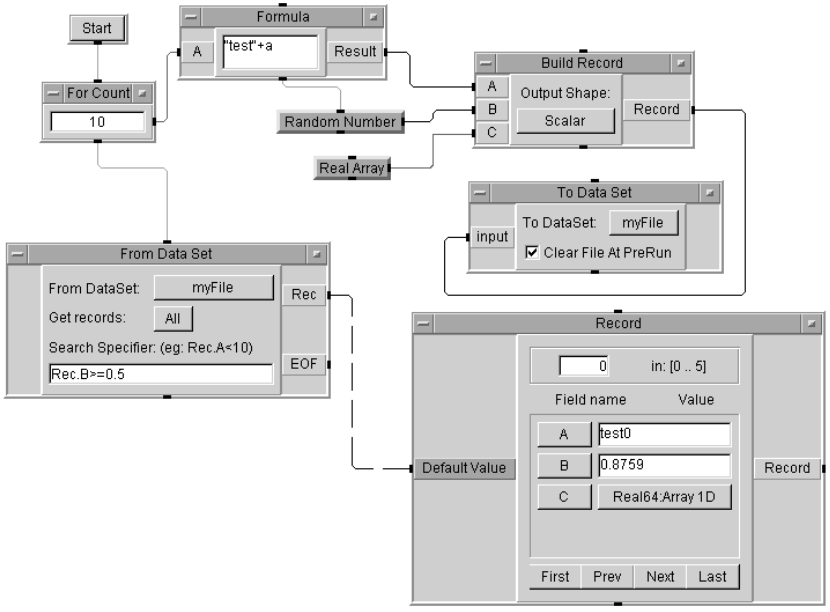


図141　　DataSetの検索

検索操作のためのオペレータ・インタフェースの作成方法

この例題では、オペレータがテスト結果データベースからデータを抽出するためのメニューを追加します。オペレータ・インタフェースにより、プログラムが保護され、不用意に変更されることがなくなります。

プログラムの仕様は次のとおりです。

- オペレータが**test0**から**test9**までの中から特定のテストを選択し、関連するすべてのテスト・データを取得するテスト・メニューを提供する。
- 指定されたテスト結果をフィールドと値にラベルを付けて表示する。オペレータは、対話形式で、より詳細な情報を取得できる。
- 明快な操作手順の説明を含める。

プログラムを作成するには、次の手順に従います。

1 **dataset2.vee**プログラムを開きます。

From Data Setオブジェクトにプログラムで式を入力できるようにコントロール入力を追加します。

2 From Data Setのオブジェクト・メニューをオープンし、**[Add Terminal...]** ⇒ **[Control Input...]**を選択します。表示されるメニューで[Formula]を選択します。Formula入力端子が表示されます。[Get records]フィールドをクリックし、[All]から[One]にトグルし、一度に1つのテスト・レコードにアクセスするようにします。

ユーザに特定のテスト名を選択させる必要があります。テスト名は、すべてのレコードのフィールド**A**にあります。次の式を追加します。

Rec.A==<**引用符で囲んだテスト名**>

Rec.Aは、フィールド**A**が、オペレータの選択したテスト名と一致するレコードを出力します。たとえば、オペレータが**test6**を選択した場合、式は**Rec.A**==**"test6"**となります。このオブジェクトにより、該当するテスト・レコードが抽出され、次に表示できます。

目的の選択肢をオペレータがクリックできるメニューを作成します。

3 **[Data]** ⇒ **[Selection Control]** ⇒ **[Radio Buttons]**を選択し、For Countの左に配置します。

a オブジェクト・メニューをオープンし、[Edit Enum Values...]を選択します。**[0000: Item 1]**を強調表示し、「`test0`」と入力します。Tabキーを押して**[0001: Item2]**に移り、「`test1`」と入力します。3番目のエントリ(`test2`)の後でTabキーを押すと、別のエントリが自動的に表示されます。`test9`になるまで、値の入力を続けます。**[OK]**をクリックすると、`test0`から`test9`までの10個のエントリすべてが表示されます。

b オブジェクト・メニューで[Properties]をクリックし、そのオブジェクト名をRadio ButtonsからTest Menuに変更し、[Execution]で[Auto Execute]を選択し、**[Open View]** ⇒ **[Show Terminals]**を選択し、**[OK]**をクリックします。

4 これで、オペレータがいつメニューを選択してもプログラムを実行できるため、Startオブジェクトを削除します。Startオブジェクト上でマウスの右ボタンを押し、[Cut]を選択します。

5 プログラムは、メニューが選択された場合にのみ実行する必要があるので、Test Menuのデータ出力ピンEnumを**For Count**シーケンス入力ピンに接続します。プログラムは図142のように表示されます。

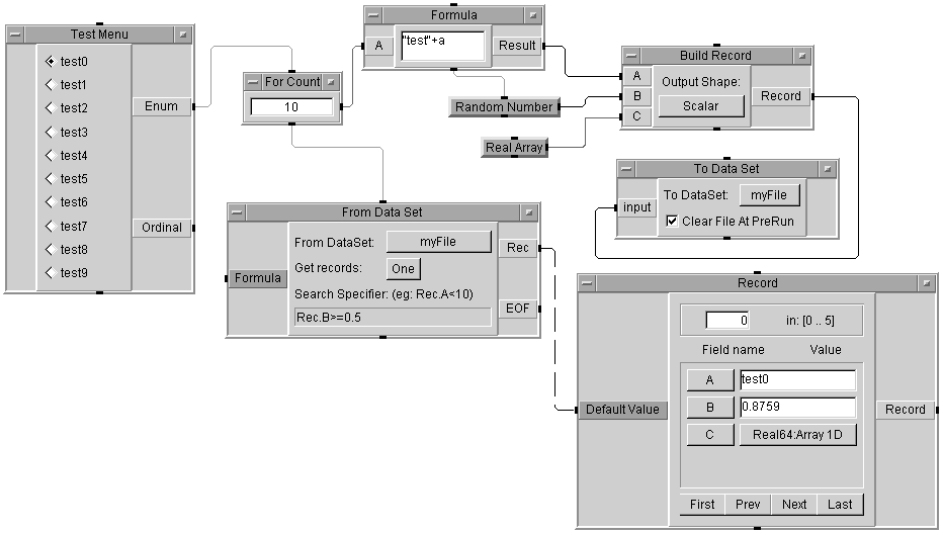


図142　　Test Menuオブジェクトの追加方法

6 Test Menuの出力はFormulaオブジェクトに入り、このオブジェクトは、次に、正しい公式をFrom Data Setオブジェクトに送信します。

[Device] ⇒ **[Formula]**を選択し、Test Menuの下に配置します。練習中は、項目を追加するときに、オブジェクトの位置とサイズを変更できます。新しいFormulaオブジェクトで、次の式を入力します。

```
"Rec.A==" + "\"" + A + "\""
```

表24　　公式の解釈のしかた

要素	説明
“Rec.A==”	`"Rec.A=="`は、Textデータ型をFrom Data Formula式入力に送信します。引用符はテキスト文字列を示します。

表24　公式の解釈のしかた

要素	説明
A	VEEは、DataSetファイル内のすべてのレコードの最初のフィールド**A**を調べ、選択されているテスト名と最初に一致するレコードを選択します。
"\"""	疑問符のエスケープ文字は\"です。エスケープ文字は、テキスト文字列であることを示すために引用符で囲みます。 *test name*は、Enumデータ型としてTest Menuにあったものです。正しい公式をFrom DataSetオブジェクトに入力するには、引用符が必要です。

たとえば、**test6**を選択した場合、最後の公式が**Rec.A=="test6"**を読み取ります。次に、From Data Setオブジェクトが、最初に見つけたレコードを出力します。そのレコードの**A**フィールドは**test6**と一致しています。

7 Test Menu Enumデータ出力ピンをFormulaオブジェクトのデータ入力ピンに接続します。Formulaオブジェクトをアイコン化します。

8 Formulaのデータ出力ピンをFrom Data SetオブジェクトのFormulaというラベルの付いたコントロール入力ピンに接続します。

9 Formulaの古いデータが再使用されないようにするため、For CountとFrom Data Setを結ぶシーケンス・ラインを削除します。For Countシーケンス出力ピンをFormulaシーケンス入力ピンに接続します。

10 Formulaシーケンス出力ピンをFrom Data Setシーケンス入力ピンに接続します。これにより、確実にFormulaの正しいデータが使用されるようになります。

11 オペレータのための手順説明を表示するボックスを作成します。**[Display]** ⇒ **[Note Pad]** を選択します。タイトルを`Test Results Database Instructions`に変更します。テンプレート情報があれば削除します。Note Pad入力領域をクリックし、「`Select the test results you want from the Test Menu.`」と入力します。

12 Record Constantオブジェクトの名前を`Test Results`に変更します。

13 プログラムは図143のように表示されます。プログラムを数回実行し、機能するかどうかを確認します。Test MenuオブジェクトはAutoExecuteをオンにしているため、プログラムを実行する場合は、メニューで選択します。

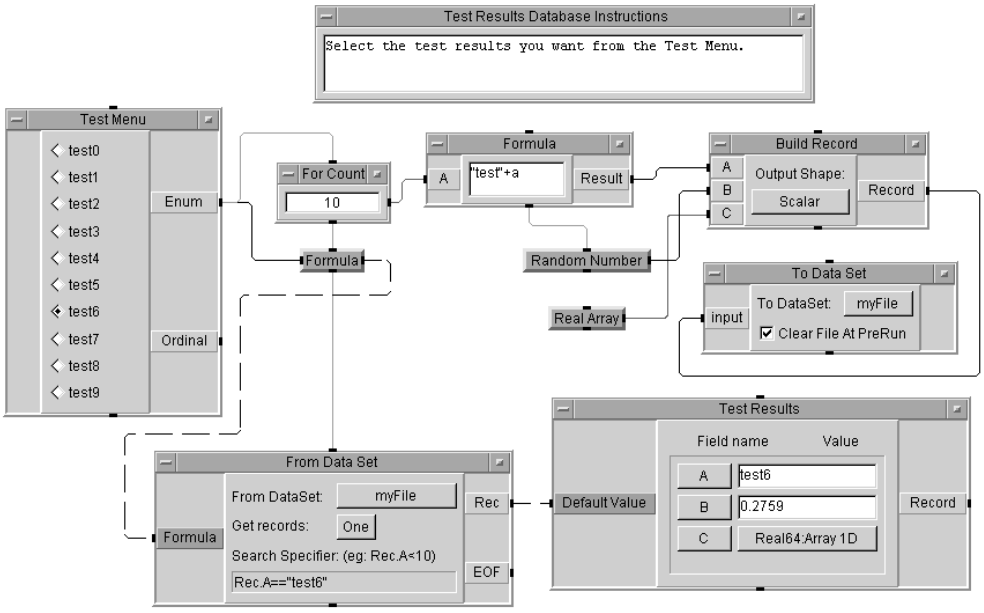


図143　　検索操作へメニューを追加する方法

次に、オペレータ・インタフェースを作成します。

14 Ctrlを押したまま、オブジェクト、**Test Menu**、**Test Results Database Instructions**、**Test Results**をクリックします。

選択したすべてのオブジェクトは、網かけ表示されます。上記以外のオブジェクトが選択されていないことを確認します。

次に、**[Edit]** ⇒ **[Add to Panel]**を選択すると、オペレータ・インタフェースがパネル・ビューとして表示されます。次に、それらのオブジェクトを移動したりサイズを変更できます。レイアウトの例を図144に示します。

注 記

Add to Panelが淡色表示になっている場合は、作業領域内に、選択しているオブジェクトがないことを意味します。

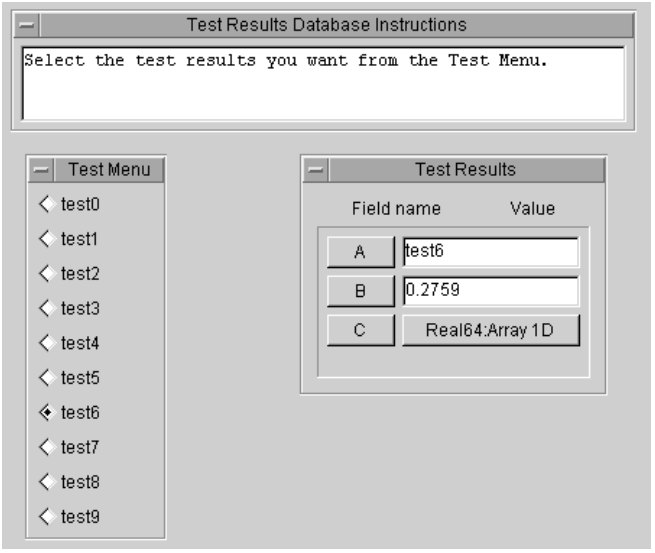


図144　　データベースのオペレータ・インタフェース

15 Test Menuで選択する方法で、プログラムを数回実行します。プログラムに**database.vee**と名前を付けて保存します。

あるレコードについて、Record Constantオブジェクト内で(Test Resultsという名前の付いている)フィールド名や値をクリックすると、そのレコードに関するより詳細な情報を取得できます。

注 記

オペレータ・インタフェースとプログラムが変更されないようにするには、[File] ⇒ [Create RunTime Version...]を選択します。プログラムの名前を入力すると、VEEは、自動的に*.vxe拡張子を追加して、保護されていないバージョンから分離します。

Recordのフィールドに基づくソート操作

この例題では、前の例題の**dataset2.vee**プログラムを使用します。**dataset2.vee**プログラムは、From DataSetオブジェクト内で、**Rec.B>=0.5**などの条件を設定するので、VEEは、それらの条件に一致するすべてのレコードを抽出します。結果レコードから成る配列がRecord Constantオブジェクト内に表示されます。

この例題では、**dataset2.vee**を変更して、最も大きな値で失敗するテストを見つけるために、結果レコードをソートします。各テストを2番目のフィールドで降順にソートします。

1 **dataset2.vee**プログラムを開きます。

2 **[Device]** ⇒ **[Formula]**を選択し、From Data Setのデータ出力ピンRecをFormulaオブジェクトのデータ入力ピンに接続します。Formulaの式フィールドをダブルクリックしてデフォルトの公式を強調表示してから、「`sort(a, 1, "B")`」と入力します。

   Sortオブジェクトは、Function & Object Browser関数とArray Category関数内にあります。オブジェクト・メニューの[Help]エントリで、その機能に関する詳細情報を参照できます。sort()関数は、Formulaオブジェクトから呼び出されます。

   最初のパラメータは、FormulaオブジェクトのAピンにあるデータをソートします。そのデータは、レコード配列です。2番目のパラメータはソートの方向を指定します。つまり、0以外の数は昇順を指定し、0は降順を指定します。デフォルトは昇順です。3番目のパラメータは、Recordデータ型の場合、ソート対象のフィールドの名前を指定します。したがって、このパラメータにより、レコード配列内のBフィールドに対して、昇順ソートが実行されます。

3 **[Display]** ⇒ **[AlphaNumeric]**を選択し、Formulaオブジェクトのデータ出力ピンに接続します。

4 プログラムを数回実行します。プログラムは図145のように表示されるはずです。プログラムは、フィールドBを使用して、DataSetファイルから返されたレコードのすべてを昇順でソートします。

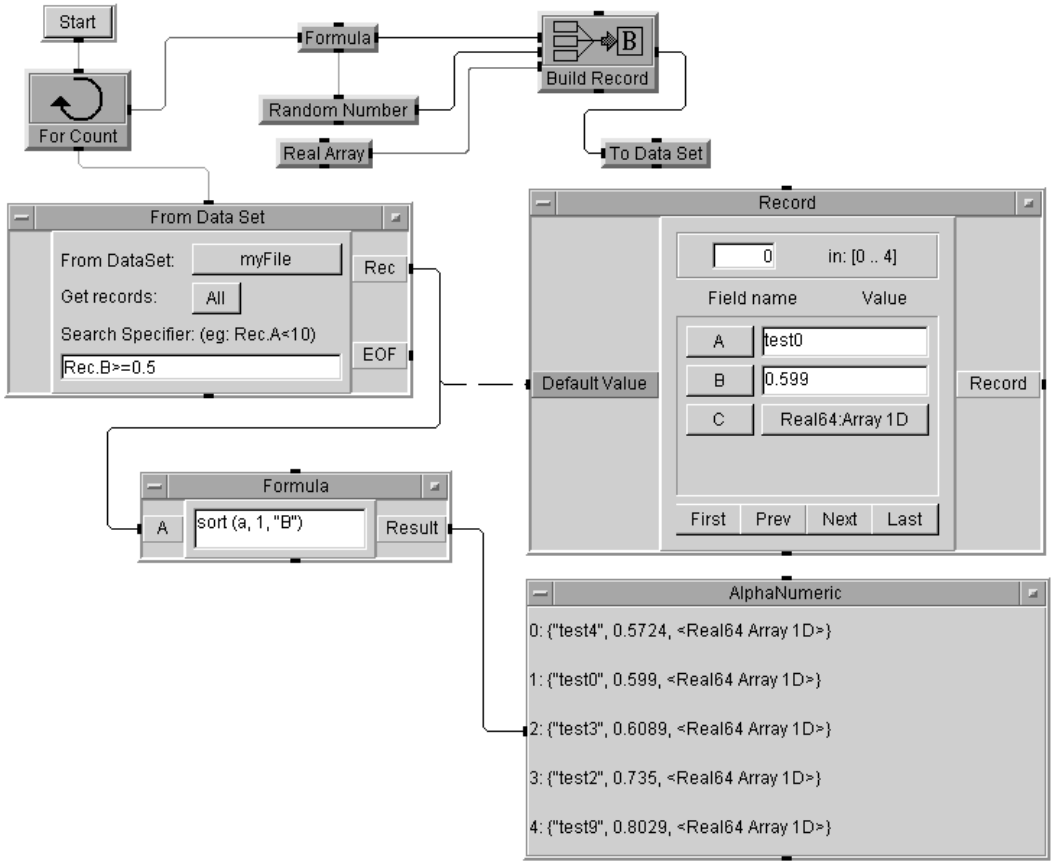


図145　Recordのフィールドに対するソート操作

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。

- 配列を使用する場合の基本的な表記を説明する。
- Collectorオブジェクトを使って配列を作成する。
- Formulaオブジェクトを使用して、配列から要素を抽出する。
- 文字列、タイム・スタンプ、real配列をファイルに送信する。
- 文字列、タイム・スタンプ、real配列をファイルから読み取る。
- 関数now()を使ってタイム・スタンプを取得する。
- タイム・スタンプをさまざまな方法でフォーマットして表示する。
- レコードの構築とunbuildを行う。
- レコード内のフィールドの取得と設定を行う。
- DataSetにレコードを保管する。
- DataSetからレコードを読み取る。
- DataSetに対して検索を実行する。
- Recordフィールドに対してソートを実行する。
- VEEツールを組み合わせて使用し、単純なテスト・データベースを作成する。

6
ActiveXを使ったレポートの簡単な作成方法

ActiveXを使ったレポートの簡単な作成方法

この章の内容

- VEEにおけるActiveXオートメーション
- ActiveXを使ってMS Excelでレポートを表示する方法
- ActiveXを使ってMS Wordでレポートを表示する方法

平均所要時間: 1.5時間

概要

この章では、データをVEEプログラムからMS Excelプログラムに送信して、MS Excelなどのほかのアプリケーションでレポートを生成する方法を習得します。VEEは、ほかのアプリケーションを制御するために、ActiveXオートメーションにより、詳細で効果的なレポートを迅速に作成できます。

最初の例題では、ActiveXオートメーションを使用して、データをMS Excelスプレッドシートに自動的に送信する方法を示します。2番目の例題では、レポートを生成するための一般的なテンプレートと、基本テンプレートの機能を拡張する方法を示します。最後の例題では、VEEでActiveXを使用して、画面ダンプとテスト・データをMS Word文書に送信します。ActiveXオートメーションをサポートしているほかのスプレッドシートやワード・プロセッサ・プログラムでも利用できます。

注 記

VEEでは、DDEがActiveXに置き換わりました。ただし、DDEも引き続きサポートされています。既存のアプリケーションでDDEを使用する場合は、『*Visual Programming with VEE*』第2版を参照してください。

Agilent VEEにおけるActiveXオートメーション

この章では、ActiveXオートメーションは、VEEがMS Excel、MS Word、MS Accessなどのオートメーション・サーバ・アプリケーションのオートメーション・コントローラとして機能する能力を指します。この章の例題では、MicrosoftのActiveX技術を実地に利用して、テストと計測のプログラムに関するレポートを生成します。

注 記

このマニュアルには、ほかにも関連する例題があります。それらについては、432ページの「例題11-4: ActiveXコントロールの使用方法」と、474ページの「Callable VEE ActiveX Automation Server」を参照してください。「オートメーション」の用語と概念についての詳細は、マニュアル『*VEE Pro Advanced Techniques*』を参照してください。

ActiveXオートメーションのタイプ・ライブラリの一覧表示

コンピュータにインストールされているオートメーション・オブジェクトを確認するには、**[Devices]** ⇒ **[ActiveX Automation References]**をクリックします。

注 記

ActiveXコントロールの参照については、第11章「オペレータ・インタフェースの使用方法」を参照してください。第13章の473ページ「概要」も参照してください。

[Devices] ⇒ **[ActiveX Automation References]**を選択すると、PCにインストールされているタイプ・ライブラリの一覧が表示されます。オートメーション・サーバとなることができる各アプリケーションとActiveXコンポーネントは、タイプ・ライブラリを登録します。VEEは、PCで使用可能なタイプ・ライブラリを表示しますこれらのライブラリは、ActiveXクライアントに公開されているアプリケーションやコンポーネントの機能に関する情報を保持しています。

タイプ・ライブラリは、通常、クラスの集合から成ります。一部のクラスはプログラマが作成できます。そのほかのクラスは、常に、アプリケーションまたはコンポーネントによって作成されます。クラスは、プロパティ、メソッド、およびイベントから成りますが、必ずしも3つすべてが存在する必要ははありません。タイプ・ライブラリは、プログラマとVEE環境の両方に、ActiveXインタフェースを使ってアプリケーションまたはコンポーネントを利用する場合に必要な情報を提供します。

[ActiveX Automation References]ボックスで、タイプ・ライブラリの横にチェック・マークを付けると、そのライブラリのオブジェクトがVEEのプログラムで使用可能になります。たとえば、図146では、Microsoft Excel 9.0にチェック・マークが付けられています。

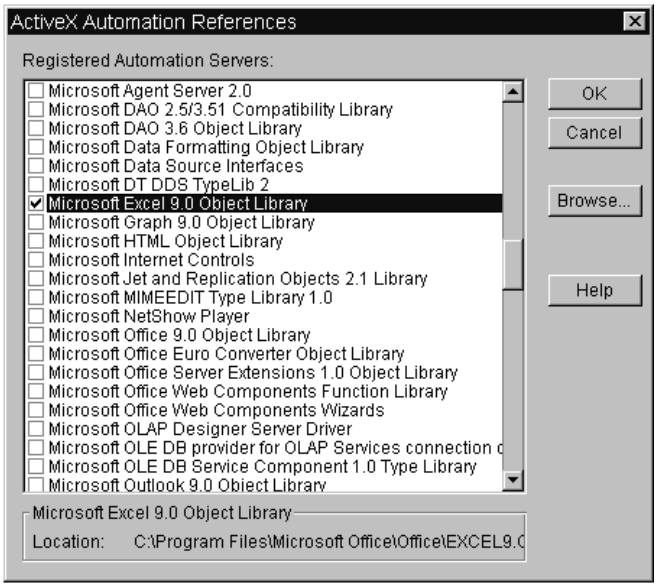


図146　[ActiveX Automation References]ボックス

Agilent VEEでActiveXプログラムを作成および使用する方法

VEEには、Objectと呼ばれるActiveXプログラムのためのデータ型があります。Objectとして指定されるデータ型を持つVEEオブジェクトは、オートメーション・サーバが保持しているデータなどを指すポインタです。たとえば、Objectは、MS Excel内のワークシートを指すポインタやそのワークシート内のセルを指すポインタとなることができます。技術的には、Objectは、MS Excelまたはオートメーション・サーバが返すIDispatchインタフェースを指すポインタです。

たとえば、**[Data]** ⇒ **[Variable]** ⇒ **[Declare Variable]**を選択し、[Name]を[App]に設定し、そのデータ型を[Object]に設定した場合、Excelオートメーション・サーバなどのActiveXオートメーション・オブジェクトを指すために、変数Appを使用できます。図147は、データ型Objectの例です。

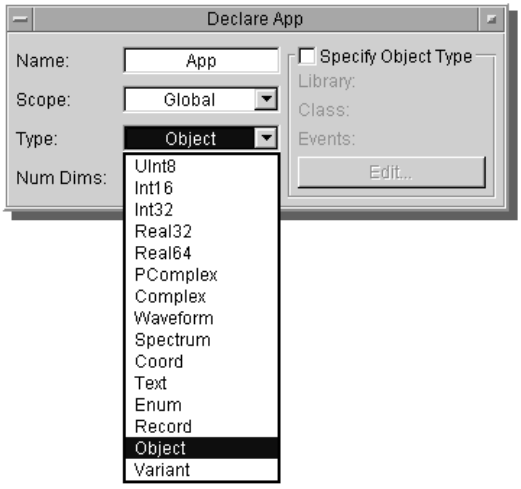


図147　　データ型Objectの例

ActiveXステートメントを使って操作を実行する方法

Excelオートメーション・サーバなどのActiveXオートメーション・サーバと通信する場合は、VEE FormulaオブジェクトにActiveXのコマンドを入力します。たとえば、図148は、**Set Up Excel Worksheet**という名前の付いたVEE Formulaオブジェクトを示しています。このオブジェクトは、Excelワークシートをセットアップしてテスト結果を表示するためのコマンドのリストを保持します。

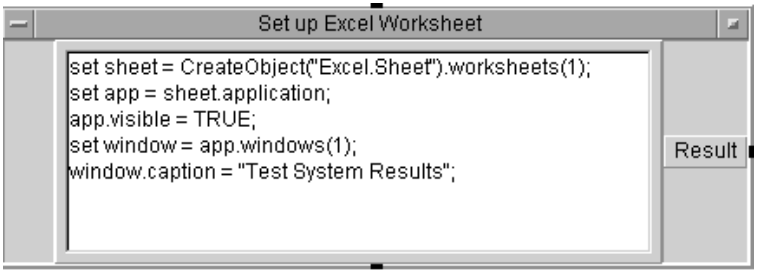


図148　Excelワークシートをセットアップしてテスト結果を表示するためのコマンド

図148に示すコマンドやステートメントを作成するため、VEEでは、Microsoft Visual Basicの標準の構文を使用します。コマンドやステートメントは、3種類の操作を実行します。それらは、*Get Property*、*Set Property*、*Call Methods*です。

- *Get Property*ステートメントは、通常、ある型のデータを取得する場合に使用します。構文は、<**オブジェクト**>.<**プロパティ**>です。たとえば、sheet.applicationは、sheetオブジェクトのapplicationプロパティを取得します。
- *Set Property*ステートメントは、通常、同じ型のデータを設定する場合に使用します。構文は、<**オブジェクト**>.<**プロパティ**>=<**プロパティの種類**>です。たとえば、**object.property = MaxSize**で、1つのプロパティが設定されます。
- *Call Methods*は、メソッドを呼び出します。メソッドは、操作を実行するようにオブジェクトに要求します。メソッドのパラメータを使用して、データの受け渡しを行うことができます。構文は、<**オブジェクト**>.<**メソッド**>(**パラメータ**)です。

注 記　データ型Objectの構文は、Recordのフィールドを取得する構文rec.fieldと、UserFunctionを呼び出すVEE構文myLib.func()に似ているため、変数にはわかりやすい名前を割り当てることが重要です。

CreateObjectとGetObjectの使用方法

図148で、**Set Up Excel Worksheet**内のステートメントの1つが**CreateObject()**関数呼び出しを保持していることに注目してください。CreateObject()とGetObject()はVEEのFunction & Object Browserにある関数であり、VEEにあるActiveXオブジェクトを指すポインタを返すように設計されています。

たとえば、**CreateObject("Excel.Sheet")**は、Excelを起動し、Excelのワークブックに対する参照を返します。Microsoftのステートメントsheetはワークブックを返します。実行中のExcelの既存のデータなどを取得する場合や、実行中のExcelにファイルをロードする場合は、GetObject()を使用します。

CreateObjectとGetObjectは、**[Device]** ⇒ **[Function & Object Browser]**、[Type: Built-in Functions]、[Category: ActiveX Automation]にあります。CreateObjectとGetObjectの例を図149に示します。

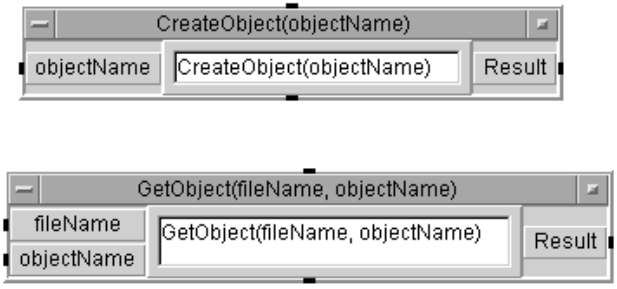


図149　CreateObjectとGetObject

Agilent VEEデータのMS Excelへの送信方法

この節では、レポートを生成するためのVEEオブジェクトとMS Excel関数呼び出しを紹介します。

例題6-1: Agilent VEEデータのMS Excelへの送信方法

この例題では、MS Excel用の仮想テスト・データを生成します。ここではMS Office 2000とMS Excel 9.0オブジェクト・ライブラリを使用しますが、MS Office 97とMS Excel 8.0オブジェクト・ライブラリも使用できます。適切なオートメーション・タイプ・ライブラリを参照した後、Object型のグローバル変数を複数個宣言し、globalsという名前でUserFunctionに保管します。グローバル変数を使用すると、プログラムが単純化して理解しやすくなります。

注 記

このマニュアルに記載されている練習問題やプログラミング例で使用したVEE プログラムの多くは、VEEに付属しています。[Help] ⇒ [Open Example . . .] ⇒ [Manual] ⇒ [UsersGuide]を選択してください。

1. オートメーション・ライブラリを参照します。
   [Device] ⇒ [ActiveX Automation References...]をクリックし、[Microsoft Excel 9.0 Object Library]を選択し、**[OK]**をクリックします。
2. グローバル変数を保管するUserFunctionを作成します。**[Device] ⇒ [UserFunction]**をクリックします。その名前を`globals`に変更します。UserFunctionについての詳細は、第9章「Agilent VEE 関数の使用方法」(329ページ)を参照してください。
3. **[Data] ⇒ [Variable] ⇒ [Declare Variable]**をクリックして、オブジェクトをglobals内の左に配置します。[Name]を`[sheet]`に変更します。[Type]を[Object]に変更します。ダイアログ・ボックスにそのほかの項目が表示されます。この例題の場合、[Object Type]と[Class]を指定する必要はありません。[Type]と[Class]は、この章の別の例で指定します。
4. このオブジェクトのクローンを3つ作成し、3つのオブジェクトの名前を`app`、`range`、`window`に変更します。globals UserFunctionのサイズを変更し、メインの下に移動します。図150のように表示されます。

5 エントリを図150と比較した後で、4つのオブジェクトをアイコン化します。

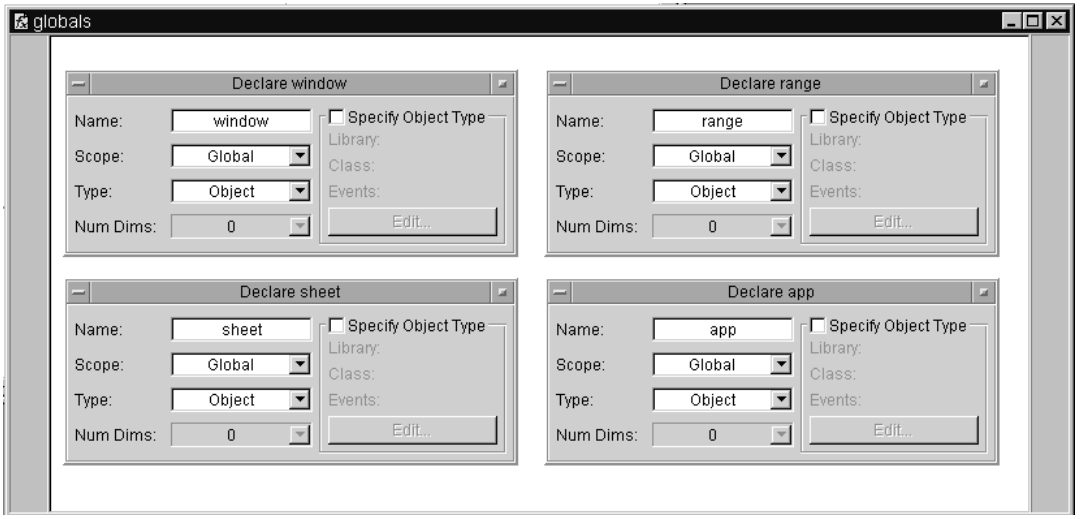


図150　Globals UserFunction

globals UserFunctionでデータ型Objectを使用すると、Object TypeとClassを指定できます。Object TypeとClassを指定する理由は、より具体的に型を検査するためと、イベントをキャッチするためです。

より具体的な型の検査:　たとえば、Object appの型をExcel.Applicationと指定した場合は、型がExcel.Applicationであるobjectしか、appに代入できません。Excel.worksheet型またはWord.bookmark型のObjectを代入するとエラーが発生します。

イベントのキャッチ:　VEEのUserFunctionを使用すると、アプリケーションで発生する可能性のあるMS Excelワークシートでマウスの右ボタンが押されるなどのさまざまなイベントをキャッチすることもできます。どの種類のイベントについても、VEEのUserFunctionを指定すると、そのイベントを処理し、情報をMS Excelに返すことができます。

ActiveXコントロールがとVEEの通信に戻る方法が必要な場合、イベントが役に立ちます。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

6 UserFunction globalsのオブジェクト・メニューをオープンし、**[Generate]** ⇒ **[Call]**をクリックします。これにより、正しく設定されたCall globalsオブジェクトが生成されます。それをメイン・ウィンドウ内の左側に配置し、globals UserFunctionウィンドウをアイコン化します。

7 **[Device]** ⇒ **[Formula]**をクリックし、それをメイン・ウィンドウの上部中央に配置します。名前を`Set up Excel Worksheet`に変更します。globalsのシーケンス出力ピンをFormulaのシーケンス入力ピンに接続します。**Set Up Excel Worksheet**から入力端子**A**を削除します。そのObjectのメニューをオープンし、**[Delete Terminal]** ⇒ **[Input]**を選択します。

8 **Set up Excel Worksheet**内で、図151のように入力します。ANSI Cと同じように、行の区切り文字にはセミコロンを使用します。

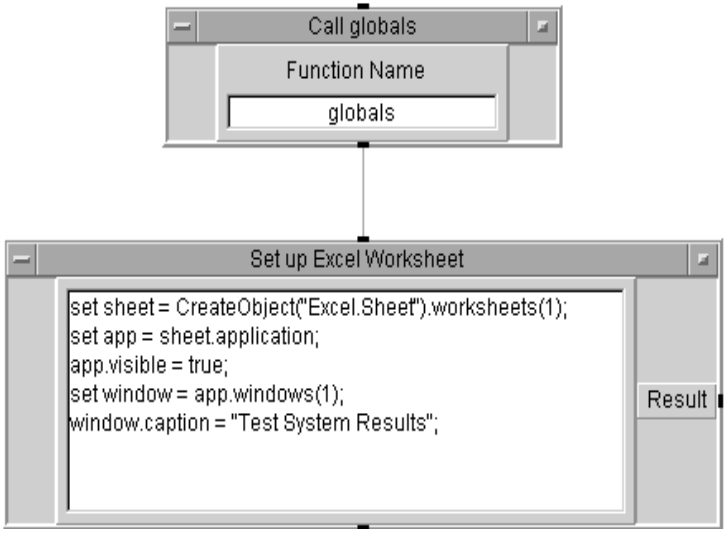


図151　MS Excelワークシートのセットアップ

表25　Formulaオブジェクトにおける Excelワークシートの設定

コマンド	説明
set sheet =	キーワード**set**は、代入演算子(この場合は、等号)の右辺を式の左辺の変数に代入または設定するために使用します。たとえば、set appは、Excelワークシートとして定義されているアプリケーションsheet.applicationに設定します。
CreateObject (“Excel.Sheet”)	オートメーション・サーバ(この場合は、MS Excel)の新しいインスタンスを作成し、空のシート(Excelの用語では新しいワークブック)を作成します。それぞれのオートメーション・サーバには独自の用語がありますが、構文は同じです。たとえば、MS Wordでレポートを作成する場合、Wordを起動して空白の文書を作成するために、`CreateObject ("Word.Document")`と入力します。 setキーワードを使用した場合は、右辺のオブジェクト・ポインタ自体が左辺の変数に代入されます。setを使用しない場合は、右辺のデフォルトのプロパティ(多くの場合、名前)が左辺に代入されます。詳細は、『*VEE Pro Advanced Techniques*』を参照してください。
worksheets(1);	Excelは新しいワークブックを使って実行されるので、**CreateObject ("Excel.Sheet")**を使用し、最初のワークシートを指定する必要があります。ステートメントに`worksheets(1)`を追加します。ステートメント全体は次のようになります。 **setsheet = CreateObject("Excel.Sheet").worksheets(1);** これにより、sheetが、レポートのSheet 1に設定されます。確認するには、MS Excelをオープンし、**[ファイル]** ⇒ **[新規作成]**を選択して、新しいワークブックを作成します。Sheet1、Sheet2などとラベルの付いた複数のシートがあります。必要なのはSheet1です。
set app = sheet.application;	ワークシートにプロパティApplicationを要求し、そのプロパティを変数appに設定すると、Excelに、ワークシートではなく、アプリケーション全体を指すポインタを要求できます。
app.visible = true;	画面にExcelを表示するために、appのvisibleプロパティを[true]に設定します。

表25　Formulaオブジェクトにおける Excelワークシートの設定

コマンド	説明
set window = app.windows(1);	最初のウィンドウを参照します。
window.caption = “Test System Results”;	最初のウィンドウのキャプションを“Test System Results”に設定します。

注 記

アプリケーション・サーバのライブラリについての詳細は、ActiveXオートメーションとMS Visual Basicについて書かれた書籍を参照してください。Office 2000に関する書籍や、『Office 97 Visual Basic Programmer's Guide』の注文方法については、WWW(World Wide Web)を参照してください。VEEの構文はMS Visual Basicと似ているため、これらの書籍はVEEでも役に立ちます。

9　**[Device]** ⇒ **[Formula]**で、Formulaオブジェクトを作成します。Formulaオブジェクトをクローンし、2つ目のFormulaオブジェクトを作成します。**[Flow]** ⇒ **[Repeat]** ⇒ **[For Range]**で、For Rangeオブジェクトを作成します。図152のようにオブジェクトの名前を変更し、相互に接続し、設定します。Formulaオブジェクトの**Fill in Title**にある入力端子は必ず削除してください。

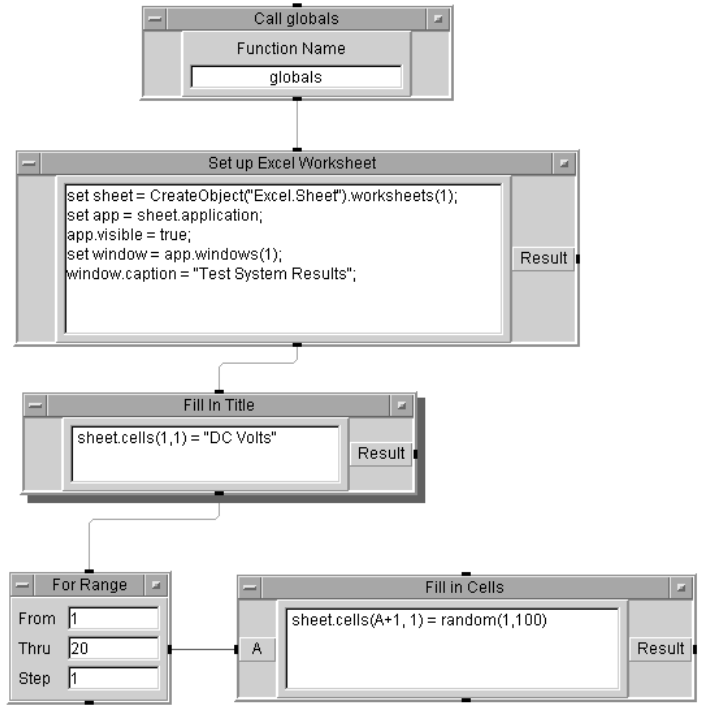


図152　　シートへのタイトルとデータの追加

Formulaオブジェクト内とFor Rangeオブジェクト内の命令について、以下で説明します。

表26　　FormulaオブジェクトとFormula Rangeオブジェクト内の命令

公式	説明
sheet.cells(1,1) = “DC Volts”	Excelワークシートの1行目の1列目を参照します。そこにテキストDC Voltsが配置されます。これにより、セル**(1,1)**のデフォルトのプロパティ (値)が**[DC Volts]**に設定されます。

表26　Formulaオブジェクトと Formula Rangeオブジェクト内の命令

公式	説明
sheet.cells(A+1,1) = random(1,100)	このステートメントは、**sheet.cells(A+1,1).value=random(1,100)**の省略表記です。入力ピン**A**の値に**1**を加算すると、ワークシートの行**A+1**、列**1**のセルは、行番号を取得しますが、列は1のままです。**random**が返す**1**から**100**までの値が、ワークシート内の指定されたセルに代入されます。
1から20まで、ステップは1 (For Range オブジェクト)	For Rangeオブジェクトは**1**から**20**までの整数を出力するので、**Fill in Cells**は、指定されたセルにその乱数を保管します。

10 **Formula**オブジェクトと**AlphaNumeric**オブジェクトを作成し、図153のように、名前を変更し、設定し、相互に接続します。

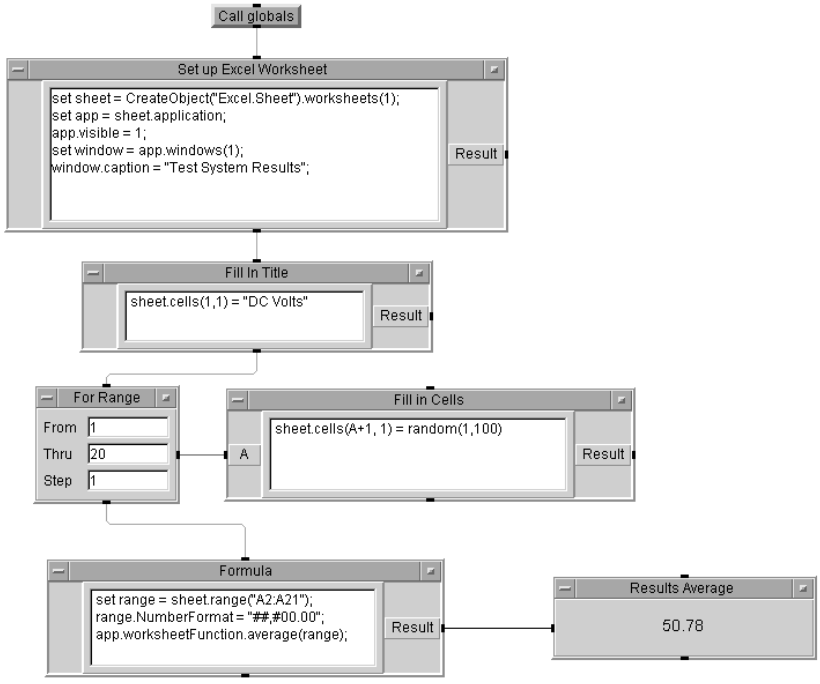


図153　　結果平均プログラム

Formulaオブジェクトでの設定項目は次のとおりです。

表27　　Formulaオブジェクトの設定項目

設定項目	説明
set range = sheet.range(“A2:A21");	VEE変数の範囲をExcelワークシートの**A2**から**A21**までの範囲を参照するように設定します。(**A**はワークシート内の最初の列を参照します。
range.NumberFormat = “##,#00.00";	必要なら、より大きな数字も入力できるように、ポンド記号(**#**)を使ってセルをフォーマットします。

表27　Formulaオブジェクトの設定項目

設定項目	説明
app.worksheetFunction. average(range);	指定されたrangeの値の平均値を返すExcelのメソッドaverage()を呼び出します。平均値はResults Averageに表示されます。

11 プログラムに**results_average.vee**と名前を付けて保存します。プログラムを実行します。MS Excelが起動して、図154のようなワークシートを表示します。

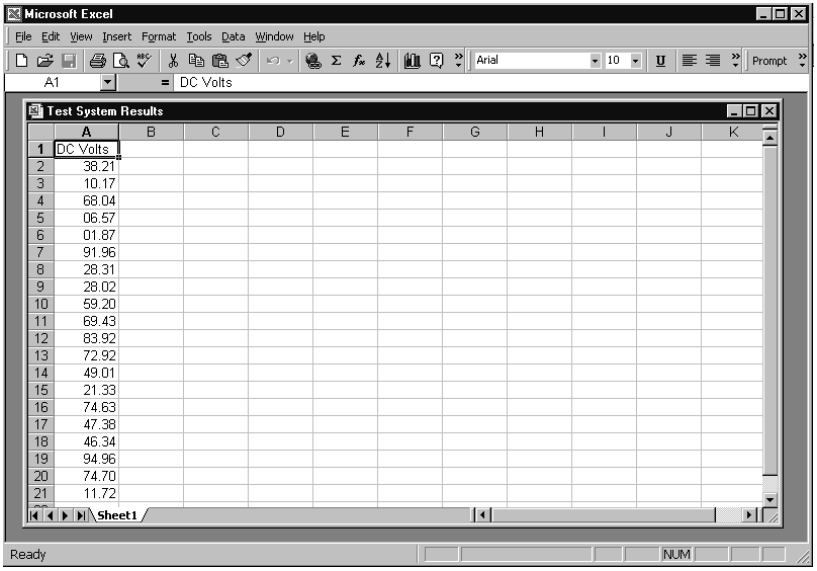


図154　Results AverageプログラムのExcelワークシート

MS Excelテンプレートに合わせたAgilent VEEの作成方法

この例題では、VEEテスト・データから成る配列をMS Excelで表示するためのプログラムを作成します。このプログラムは、ほかのテストの結果をMS Excelスプレッドシートで表示するためのテンプレートとして使用できます。

例題6-2: MS Excelテンプレートに合わせたAgilent VEEの作成方法

1. **results_average.vee**をオープンします。
2. 10回ループするようにFor Rangeオブジェクトを変更します。
3. 入力**B**を**Fill in Cells**に追加し、中のステートメントを`sheet.cells (A+1,1) = B[A-1]`となるように変更します。
4. **[Device]** ⇒ **[Formula]**をクリックし、名前を`Array of Test Data`に変更し、組込み関数「`randomize(ramp(20), 4.5, 5.5)`」を入力し、4.5から5.5までの値を持つ20個の要素から成る乱数配列を作成します。入力ピンを削除し、データ出力ピンを**Fill in Cells**の**B**入力に接続します。
5. 画面下部の**[Formula]**ボックスで、範囲を`[A21]`から`[A11]`に変更します。ステートメントは次のようになります。
   set range = sheet.range("A2:A11");
6. プログラムを**report_template.vee**としてセーブし、実行します。それを図155のExcelワークシートおよび図156の完成プログラムと比較します。

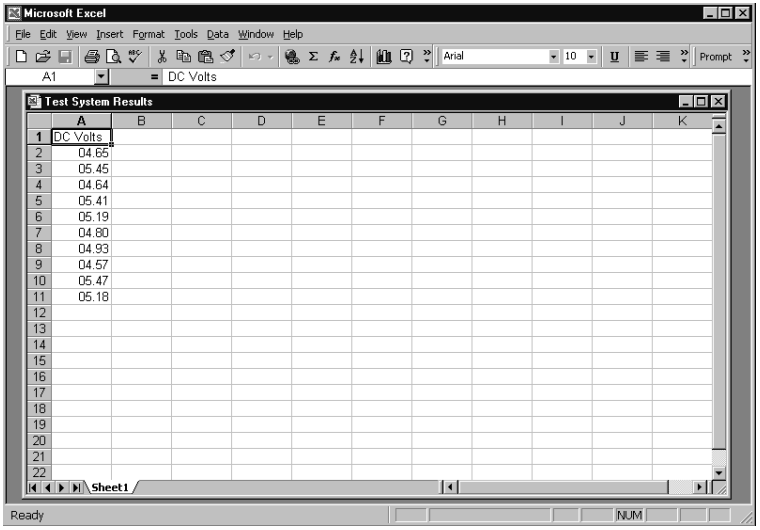


図155　テスト・データから成る配列の場合のExcelワークシート

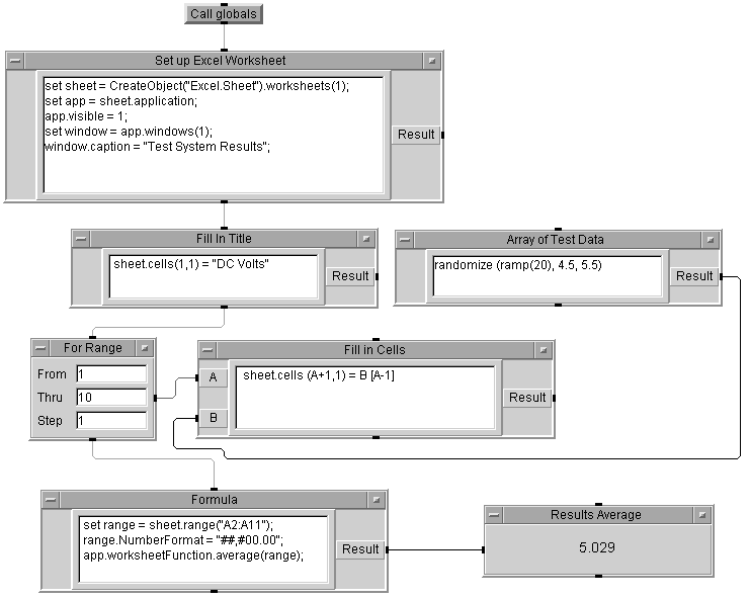


図156　　テスト・データから成る配列の場合のプログラム

このプログラムは、テスト結果をMS Excelで表示するためのテンプレートとして再利用できます。その場合は、テスト・データを配列に保管し、テンプレートの残りを変更し、正しいフォーマットで適切なセルに入力するようにするだけです。

MS Excelのライブラリでさらにメソッドとプロパティを検索するには、[Function & Object Browser]を調べ、[Type]で[ActiveX Objects]を選択し、[Library]で[Excel]を選択します。[Class]または[Member]を選択し、[Help]を押すと、そのオートメーション・サーバの作成者(この場合、Microsoft)が提供しているヘルプを表示できます。これらのライブラリについての詳細は、Microsoftのマニュアルを参照してください。

独習課題

波形を生成し、Time Spanを削除して配列を取得します。ワークシートを持つMS ExcelのためのVEEオブジェクトを作成し、Object変数に設定します。アプリケーションを表示します。次に、256の点を持つ配列をワークシートの範囲A1:A256に入力します。一度に1セルずつではなく、1ステップで全部のセルを入力します。

Unbuild Waveformオブジェクトを使用します。配列構築構文**[a]**を使用して、1次元の配列から2次元の配列を作成します。次に、図157のように、関数transpose()を呼び出し、Excelが1ステップでその配列を受け入れられるように、配列を1行256列の配列ではなく、256行1列の配列にします。

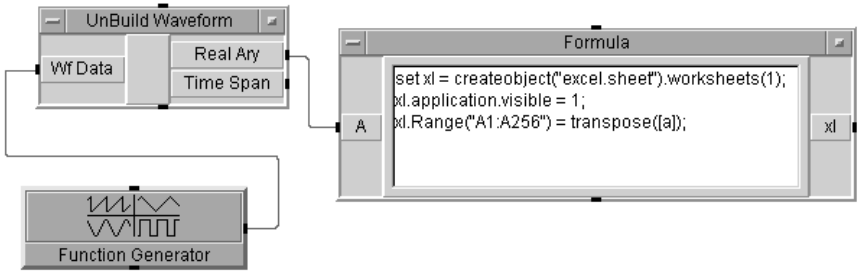


図157　　独習プログラム

MS Excelの機能の拡張

図158は、テスト結果をMS Excelで表示するための複雑なプログラムの例です。MS Excelのライブラリをさらに使いこなすことができれば、VEEデータをMS Excelで表示するためのテンプレートの機能を拡張できます。

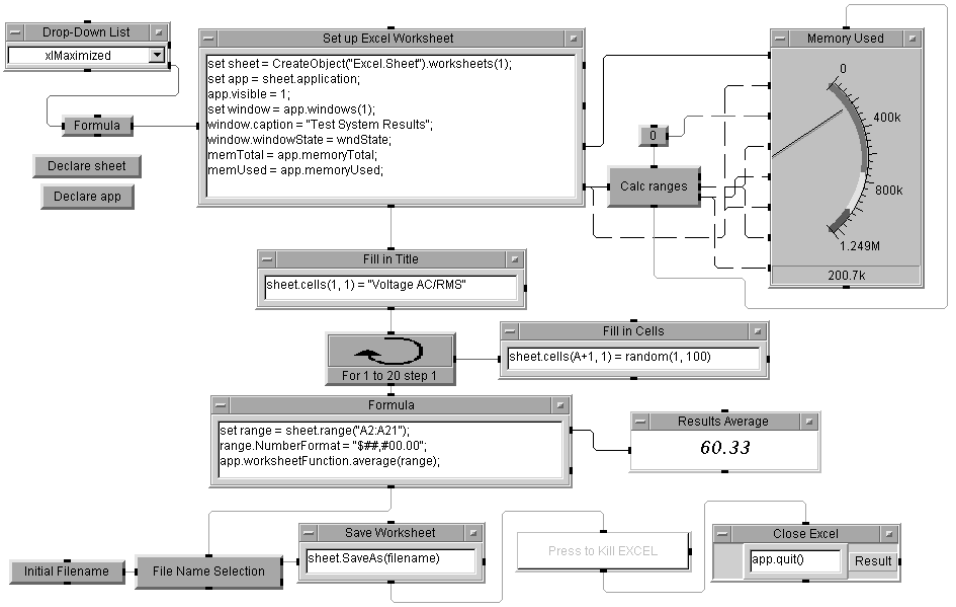


図158　　VEEからMS Excelにデータを渡すプログラムの例

図158内のエントリについて以下で説明します。

表28　　図158のエントリの説明

項目	説明
MS Excel Window Size	作業領域の上部左にあるDrop-Down Listオブジェクトに注目してください。このオブジェクトを使用すると、3つの選択肢、**[xlMaximized]**、**[xlMinimized]**、**[xlNormal]**のいずれかを選択して、Excel内のワークシート・ウィンドウを表示するサイズを選択できます。ウィンドウの各サイズは番号に関連付けられるため、VEEはその番号を算出し**wndState**変数に保管します。この値は、次に、Excelライブラリの**windowState**プロパティに代入されます。

表28　図158のエントリの説明

項目	説明
Memory Tracking	FormulaオブジェクトとMeterオブジェクトの[Properties]ボックスで**[Show Terminals]**をクリックします。Excelライブラリの**memoryTotal**プロパティと**memoryUsed**プロパティが、VEEの変数**memTotal**と**memUsed**に代入されています。2つの値を使い、MS Excelによって使用されているメモリが表示される前に、VEEのメータを設定するための範囲が計算されます。
Number Format	数値フォーマットにドル記号を簡単に追加できます。
sheet.SaveAs (filename)	SaveAs()メソッドは、ワークシートを自動的にセーブするために、Excelのライブラリから呼び出されます。**[Data]** ⇒ **[Dialog Box]**メニューの**[File Name Selection]**ボックスが、ポップアップ[Save As]ボックスをVEEから表示するために使用されています。選択したファイル名は、次に、Excel SaveAs()メソッド呼び出しでパラメータとして使用されます。
Press to Kill Excel	確認([OK])ボタンは、Excelをクローズする必要があるときに、それを知らせるために使用されています。
Close Excel	quit()メソッドは、MS Excelに終了するように指示する場合に呼び出されます。

MS Wordを使ってAgilent VEEレポートを表示する方法。

この例題では、テキスト、タイム・スタンプ、XY表示を使ったVEEポップアップ・パネルの画面ダンプなどのVEEのテスト情報をMS Word文書で表示する方法について説明します。ActiveXオートメーションを使ってほかのアプリケーションからMS Wordを制御する方法については、Microsoftのマニュアルを参照してください。

例題6-3: MS Wordを使ってAgilent VEEレポートを表示する方法。

まず、次の手順に従って、5つの変数の型をObjectとして宣言します。

1 **[Device]** ⇒ **[ActiveX Automation References...]**をクリックし、[Microsoft Word 9.0 Object Library]を選択します。

2 **[Data]** ⇒ **[Variable]** ⇒ **[Declare Variable]**をクリックします。

   a [Type]フィールドを[Object]に変更します。クローンを4つ作成します。

   b 5つのオブジェクト変数に、App、Doc、Wnd、Sel、Bmpという名前を付けます。

   c オブジェクトすべてで、[Specify Object Type]を選択します。Library内で特定のクラスを宣言すること、VEEは型検査を行ってプログラムのエラーを見つけることができ、ユーザはオートメーション・サーバからのイベントをキャッチできるという利点があります。

   d どの場合も、次に[Edit...]ボタンをクリックし、[Word for Library]を選択します。次の[Classes]を選択します。
   [App]は[Application]を使用します。
   [Sel]は[Selection]を使用します。
   [Wnd]は[Window]を使用します。
   [Doc]は[Document]を使用します。
   [Bmp]は[Shape]を使用します。

   e クラスに[Enable Events]がある場合は、選択します。5つをアイコン化します。これらの変数のオープン・ビューについては図159を参照してください。

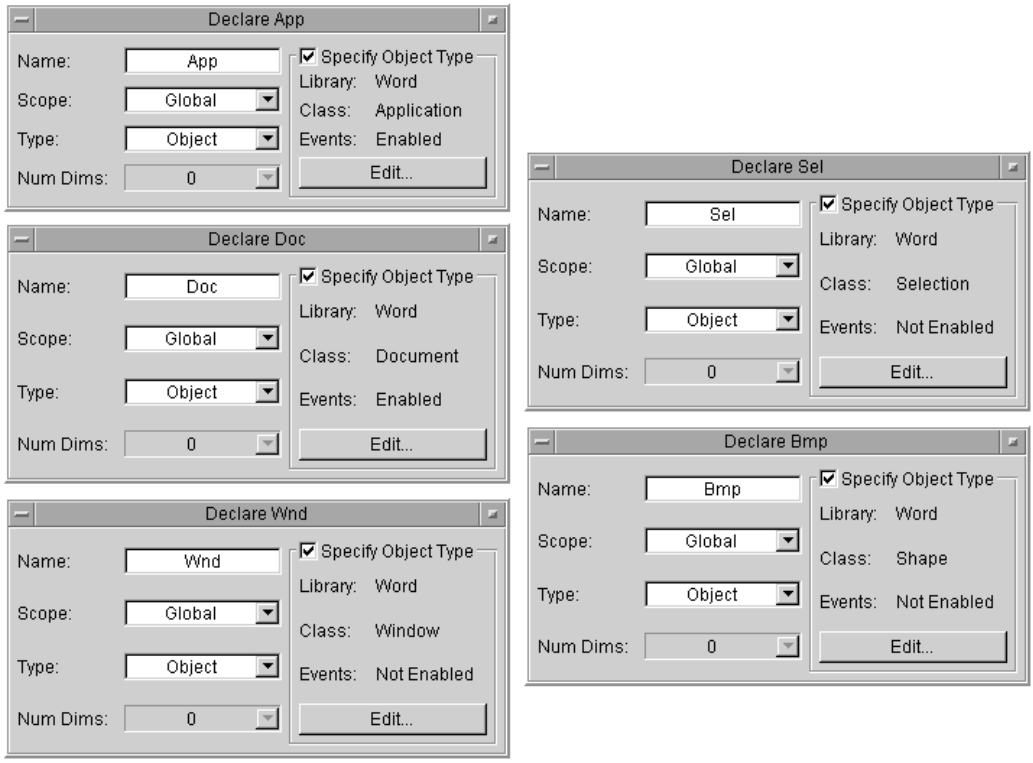


図159　　オブジェクト変数

3 Graphという名前のUserFunctionを作成します。これは、Function Generator仮想ソースを使用して、正弦波をWaveform (Time)表示に送信します。表示のみのパネル・ビューを作成します。次に、メイン・ウィンドウにCall Graphオブジェクトを生成します。UserFunctionのオブジェクト・メニューは、呼び出しを生成する容易な方法を保持しています。

MS Wordでレポートを出力するときに使用するために、波形を表示するパネルのビットマップ・ファイルを作成します。

4 ビットマップ・ファイル名を作成するには、**[Device]** ⇒ **[Formula]**をクリックします。その名前を`Image Filename`に変更します。[Formula]入力フィールドで、「`installDir() + "\\panel.bmp"`」と入

力します。ASCII文字\を指定する場合は、エスケープ・シーケンス\\を使用します。入力端子**A**を削除します。

たとえば**c:\Program Files\Agilent**にインストールした場合は、次のテキスト文字列をResult出力ピンに生成します。
C:\Program Files\Agilent\VEE Pro 6.0\panel.bmp

5 Formulaオブジェクトをもう1つ作成し、「`savePanelImage("Graph", FileName, 256)`」と入力します。その入力端子の名前をFileNameに変更します。

6 これにより、UserFunction Graphの画面ダンプが、インストール・ディレクトリ内のpanel.bmpファイルにピクセル当たりの色の深さ256でセーブされます。

7 Formulaオブジェクトをもう1つ作成し、次のステートメントを入力します。「`Set App = CreateObject("Word.Application")`」このステートメントは、MS Wordを起動し、アプリケーションのこのインスタンスを参照するためにオブジェクト変数appを割り当てます。入力端子**A**を削除します。図160のように、**Call Graph**、**ImageFileName**、**savePanelImage**を接続します。

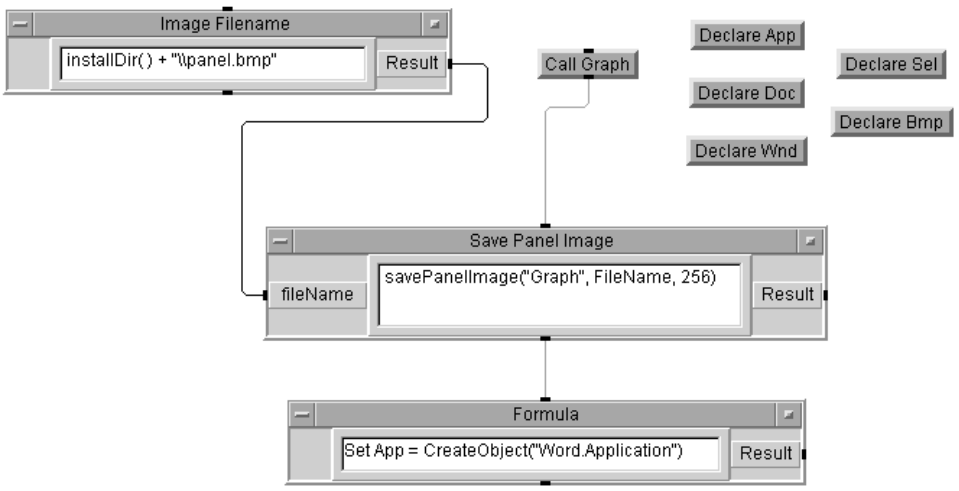


図160　　例題6-3プログラムの初期段階

8 **[Device]** ⇒ **[Formula]**をクリックし、図161のステートメントを入力します。このステートメントについても下で説明します。入力端子**A**の名前を**FileName**に変更します。データ入力ピンとシーケンス入力ピンを図161に示すように接続します。

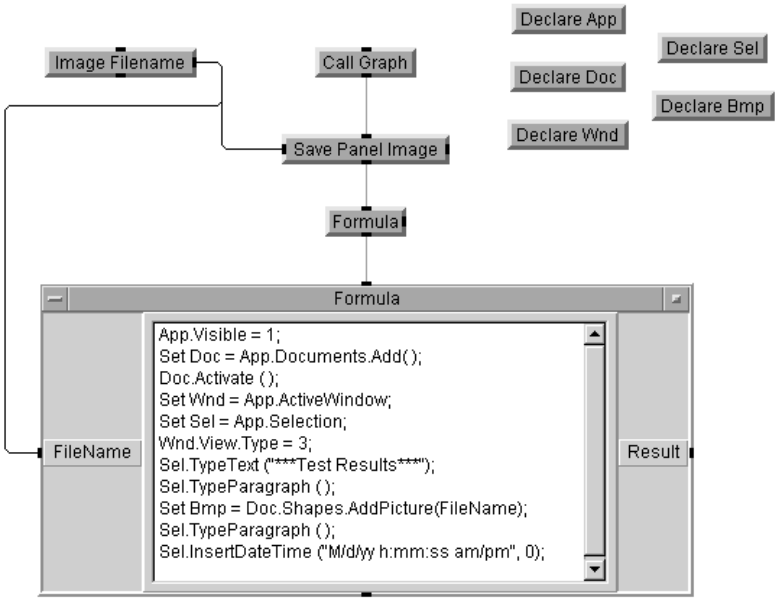


図161　ActiveXステートメントの追加

図161では、Objectのドット表記を使用すると、プロパティとメソッドの呼び出しを一緒にネストできることに注目してください。ターゲット・アプリケーション内の適切なプロパティを検索する場合は、ActiveXのマニュアルを参照してください。この章で説明しているプロパティとメソッドを使用すると、テストと計測のレポートを生成できます。Formulaオブジェクトでの設定項目は次のとおりです。

表29　Formulaオブジェクトでの設定項目

設定項目	説明
App.Visible = 1;	MS Wordを画面に表示します。

表29　Formulaオブジェクトでの設定項目

設定項目	説明
Set Doc = App.Documents. Add();	MS Word内にDocumentを追加し、Object変数**Doc**に代入します。注: Excel の例では、**CreateObject(Excel.Sheet)** を使用して、Excelを空白のワークシートで起動しました。ここでは、Wordを起動し、メソッドAdd()で空のDocumentをWordに追加します。2つのアプリケーションは、どちらの方法でも作成できます。
Doc.Activate();	上のDocumentをアクティブにします。
Set Wnd = App.Active Window;	アクティブなウィンドウ内の文書を選択し、Object変数**Wnd**に代入します。
Set Sel = App.Selection;	文書にフォーカスを合わせ(選択)、Object変数**Sel**に代入します。これにより、テキストを挿入できます。
Wnd.View.Type = 3;	文書を表示するためのウィンドウの種類を指定します。3は通常のサイズのウィンドウを示します。1はウィンドウをアイコン化します。 注: ここでは、定数wdPageViewのかわりに、3を使用します。この定数はOffice 2000のタイプ・ライブラリにないからです。
Sel.TypeText(*** Test Results ***), Sel.TypeParagraph ();	タイトル***** Test Results ***** を文書に挿入し、キャリッジリターン/改行を発行します。
Set Bmp = Doc.Shapes. AddPicture (FileName);	panel.bmpビットマップを文書に挿入し、Shapes Class内でのこの呼び出しをObject変数Bmpに代入します。
Sel.TypeParagraph (); Sel.InsertDateTime (M/d/yy h:mm:ss am/pm, 0);	タイム・スタンプを文書に挿入します。

9 さらに3つのFormulaオブジェクトと1つのIf/Then/Elseオブジェクトを追加し、図162に示すように設定し、相互に接続します。

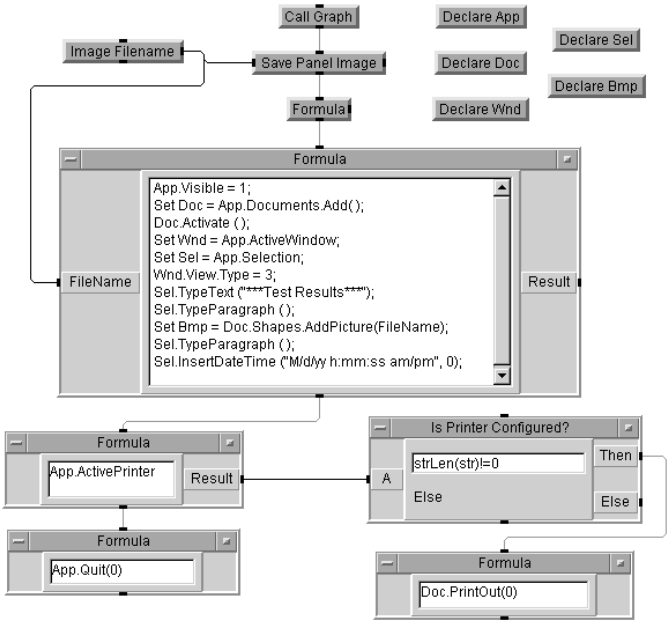


図162　MS Word内のレポート用の完成プログラム

追加したオブジェクト内のエントリについて以下で説明します。

表30　図162の追加オブジェクト

オブジェクト	説明
App.ActivePrinter	ポートを含む文字列を使用して､デフォルトのプリンタを要求します。
strLen(str) != 0	ActivePrinterが、設定されたプリンタを検出したかどうかを確認し(入力時の文字列がヌルでない場合)、次にThenピンに1**(=TRUE)**を出力します。Thenピンは、PrintOut呼び出しを保持するFormulaオブジェクトをpingします。
DocPrintOut(0)	文書を出力します。
App.Quit(0)	MS Wordアプリケーションをクローズします。

10 プログラムを実行します。図163のように表示されます。画面ダンプの色が正常でない場合は、オープンしているアプリケーションをすべてアイコン化すると、PCのカラー・パレットが完全に解放されます。

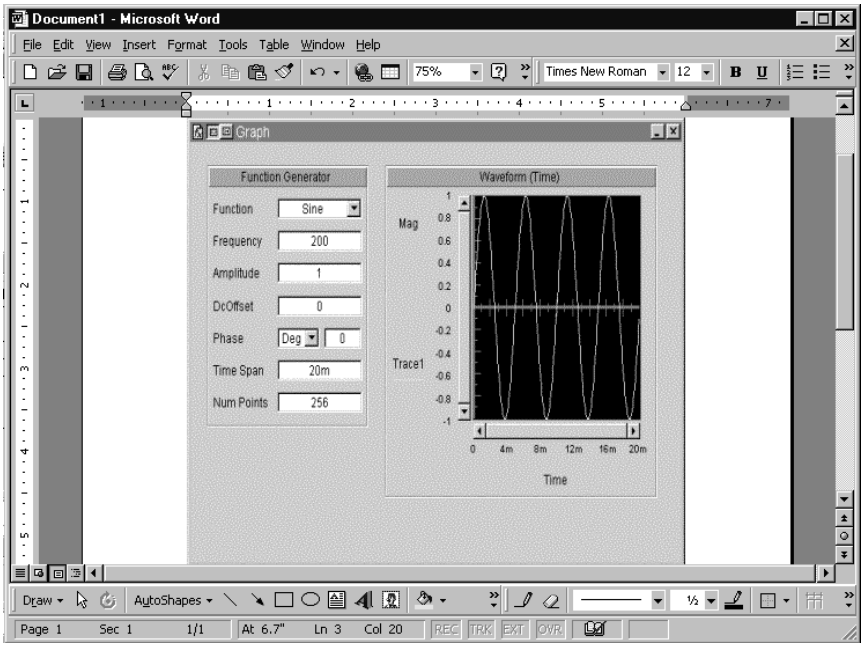


図163　　例題6-3で作成したMS Word文書

ActiveXオートメーションを使ってMS ExcelとMS Wordを制御する方法についての詳細は、Microsoftのマニュアルを参照してください。ActiveXオートメーション(単にオートメーションと呼んだり、OLEオートメーションと呼ばれることもあります)をサポートしているその他のサーバ・アプリケーションも制御できます。

ActiveXコントロールの使用方法についての詳細は、第11章「オペレータ・インタフェースの使用方法」を参照してください。VEEを制御するためにMS Visual BasicプログラムからActiveXを使用する方法についての詳細は、第13章の473ページ「概要」を参照してください。

この章の復習

- この章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。
- VEEでのActiveXオートメーションの基本概念について説明する。
- データをVEEからMS Excelに送信する。
- 一般的なテンプレートを使用して、テスト・データから成る複数の配列をMS Excelワークシートに送信する。配列を一括してスプレッドシートに送信する。
- プログラムが使ったメモリの検索など、MS Excelライブラリの拡張された機能の一部を使用する。
- テキスト、タイム・スタンプ、表示ビットマップをVEEからMS Wordに送信する。

7
VEEでの.NETの使用方法

VEEでの.NETの使用方法

この章の内容

- 基本的な.NETの用語
- 名前空間のVEEへのインポート方法
- PIAの概要とPIAが重要である理由
- VEE Runtimeを配布する際の.NETの特別な注意事項
- VEEおよび.NET Security

平均所要時間: 1.5時間

.NETの概要

Visual Studio .NETは、Microsoftの最新開発プラットフォームです。.NETは、Visual Studio 6などの以前の製品と異なり、デスクトップ・プログラマとインターネットWeb開発者の両方のニーズに対応しています。VEE 7.0では.NET Frameworkが無料配布されているため、VEEで.NETを直ちに使用できます。実際に、VEEのいくつかの新機能にこの無料ランタイムが統合されています。

VEEにとって重要な.NET Frameworkの機能と利点が、Framework Class Library(FCL)およびCOM相互運用技術です。特にFramework Class Libraryは、VEEプログラマにとって非常に価値があります。Function&Object Browserから数多くの新しいプロパティやメソッドが使用できるようになります。この新機能により、ファイルおよびディレクトリ管理、より簡単な文字列管理、オペレーティング・システム環境の簡略表示、Webページの操作、オペレーティング・システム処理へのアクセス、レジストリの読み取りなど、多くの作業がコードで実行可能になります。

.NETは、本来のCOM相互運用技術も提供します。これにより、COMコンポーネントに.NETオブジェクトとしてアクセスでき、.NETオブジェクトにCOMコンポーネントとしてアクセスできます。

この章の最後に、章全体で用いられているMicrosoftの用語の定義を示します。必要に応じて参照してください。

VEEおよび.NET Framework

.NET FrameworkとVEEの相互作用のしかたを見る前に、下記のアドレスにあるMSDNインデックスで「.NET Framework - getting started」を検索し、.NET Frameworkについて調べてください。(http://msdn.microsoft.com/library/en-us/cpguide/html/cpovrintroductiontonetframeworksdk.asp)

[Device]メニューの下に、**[.NET Assembly References]**という新しいメニュー・オプションがあります。**このメニューは、[References for ActiveX]とよく似ています。**定義からすぐに思い出されるように、参照によってアセンブリが使用可能になります。このメニュー選択をオープンします。図164のようなウィンドウが表示されます。

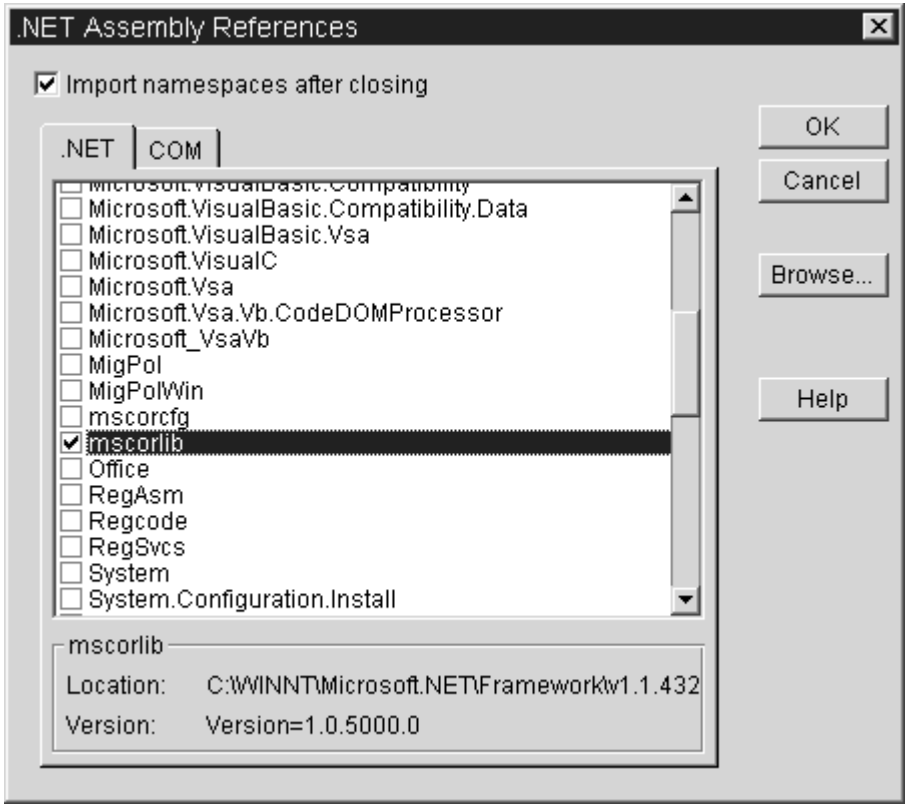


図164　[Import Namespaces]ダイアログ・ボックス

.NET Assembly References

選択可能な多数のアセンブリが表示されます。これらのアセンブリは、VEEプログラムがストアされているディレクトリ、または.NET Frameworkがインストールされているディレクトリに存在します。これら2つのディレクトリ以外にあるアセンブリを使用する場合は、[Browse]ボタンを使ってアセンブリを選択します。**[Import namespaces after closing]**のチェックボックスもあります。名前空間のインポートは、C#での**using**ステートメントまたはVisual Basicでの**Imports**ステートメントに相当します。

前の図では、もっとも一般的に使用されるアセンブリ**mscorlib**が選択されています。先へ進んで、このアセンブリを選択します。**mscorlib**には、ファイル・システム操作、収集管理、データ型変換など、VEEユーザにとって便利な関数が多数はいっています。もう1つの非常によく使用されるアセンブリが**System**です。**System**には、プロセス管理、Web要求および応答、正規表現などの関数が含まれています。

[COM]タブの下には、現在コンピュータに登録されているすべてのCOMタイプ・ライブラリのリストがあります。このリストは、[ActiveX automation references]ダイアログ・ボックスを使用するときに表示されるタイプ・ライブラリのリストとほぼ同じです。

以下に、これら2つの選択肢のどちらを使用するか判断する方法を示します。次のケースにすべてあてはまる場合には、[ActiveX automation references]ダイアログ・ボックスを使用します。

- ライブラリが[ActiveX automation references]ダイアログ・ボックスに表示される場合
- ライブラリが、前バージョンのVEEでフルに機能する場合
- ライブラリにPIA(Primary Interop Assembly)がない場合

ActiveXオートメーション参照を直接使用すると、別のレイヤとして.NETとCOMの相互運用レイヤが追加されません。

次の場合は、ここで説明する.NET/COM相互運用を使用します。

- ライブラリが[ActiveX automation references]ダイアログ・ボックスに表示されない場合
- インポートされていないライブラリ機能がある場合
- COMライブラリにPIAがあることがわかっている場合

例: Function & Object BrowserのActiveX Objectsリストからは見えないIVI-COMドライバ・インタフェースがあります。しかし、.NET相互運用では、すべてのIVIドライバ・インタフェースを見ることができます。

[.NET Assembly References]メニュー・オプションを使用し、リストからCOMタイプ・ライブラリを選択すると、VEEは、COMタイプ・ライブラリを.NETアセンブリとしてインポートしようとします。PIAが存在する場合、VEEはそれを使用します。存在しない場合、VEEは、プログラムがセーブされているディレクトリに1個のInterop Assembly、あるいはCOMタイプ・ライブラリが他のタイプ・ライブラリを参照する場合は複数のInterop Assemblyを作成します。VEEは、Interop Assemblyを取得した後、そこからすべての型情報を入手します。

注 記

生成されたInterop Assemblyをテンポラリ・ディレクトリからVEEプログラムのディレクトリにコピーしなければならないというVEEのオーバーヘッドを避けるため、最初にVEEプログラムをセーブします。この方法により、生成されたInterop AssemblyがVEEプログラムのディレクトリに直接セーブされます。

先へ進んで、[COM]タブを選択します。VEEがレジストリをスキャンしてCOMタイプ・ライブラリを探すので、これにはしばらく時間がかかります。Microsoft ActiveX Pluginを選択します。

名前空間をインポートするよう求めるプロンプトが表示されます。これはオプションのステップですが、先へ進み、Interop.ActiveXPlugin、System、およびSystem.IOを選択します。

注 記

[.NET Assembly References]ダイアログ・ボックスでチェック・ボックスのチェックを削除しない限り、[Import .NET Namespaces]ダイアログ・ボックスが毎回表示されます。

[Device]メニューから、**[Function & Object Browser]**を開きます。メニュー選択の1つとして**[.NET/CLR Objects]**が表示されます。このオプションを選択します。図165の**[Assembly]**ウィンドウに、前に行った2つの選択が示されています。

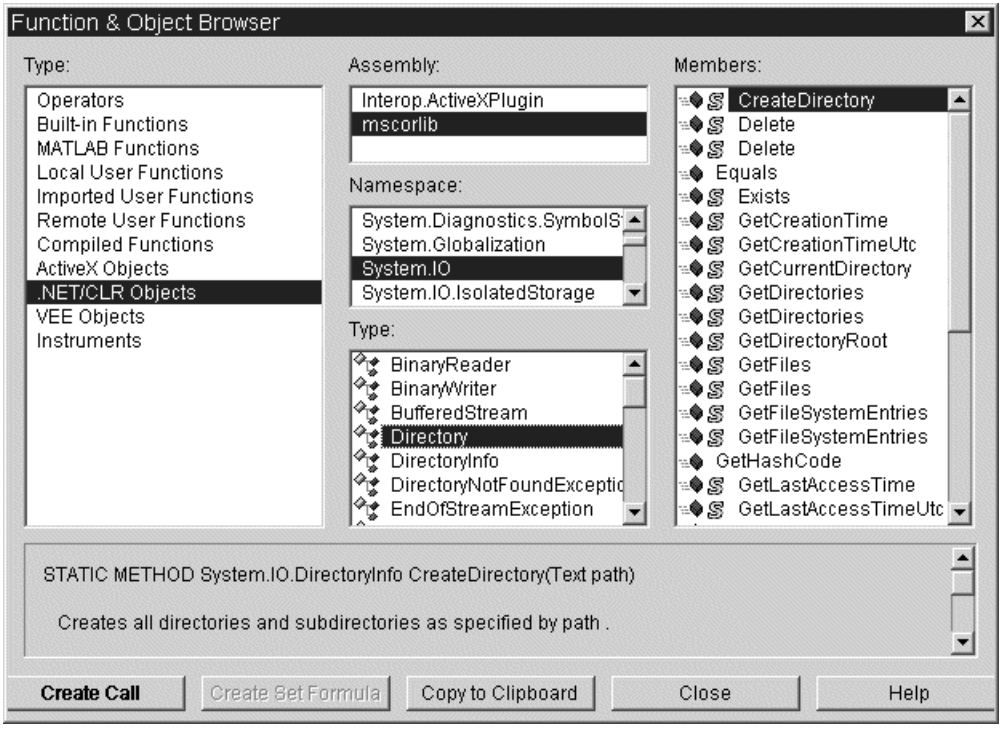


図165　Function & Object Browser - .NET Objects

選択したアセンブリのすべての名前空間が**[Namespace]**リスト・ボックスにリストされます。それぞれのNamespaceに対して、そのすべての型が**[Type]**リスト・ボックスにリストされ、各型のすべてのメンバが**[Members]**リスト・ボックスにリストされます。メンバには、コンストラクタ、フィールド、プロパティ、メソッド、列挙が含まれます。

.NETクラス・メンバの選択プロセスは、ActiveXを使用した場合のプロセスとほぼ同じです。ただし、以下のようなわずかな違いがあります。

1 COMは共有(静的)メンバを持ちません。.NETメンバが静的メンバである場合、ヘルプ領域にキーワードSTATICが表示され、アイコンの横に  が付きます。静的メンバを呼び出す場合は、オブジェクトのインスタンス上ではなく、クラス上で呼び出します。[Function & Object Browser]を使うと、構文に慣れることができます。例について

は、305ページの「例題7-2: .NETを使用したDateTime操作の実行」を参照してください。

2 インスタンス・オブジェクトの変数を生成すると、最初の文字が小文字になります。これは、オブジェクトのインスタンス・メンバでメソッドまたはプロパティを呼び出していることの手掛かりとなります。また、.NETオブジェクトを作成する必要があることも意味します。

3 .NETオブジェクトを作成するには、メンバ・リストでConstructorを選択します。コンストラクタとメソッドはアイコン を共有しています。前の図で示したように、[Create Instance]ボタンが使用可能です。このボタンを選択すると、新しい.NETオブジェクトを作成するための公式テンプレートが生成されます。図166に、VEEによって生成されたdateTime出力ピンを示します。この出力ピンを、この.NETオブジェクトを入力として要求する任意の公式ボックスの入力ピンに接続できます。例については、309ページの「例題7-3: .NETを使用したファイル情報の取得」を参照してください。

.

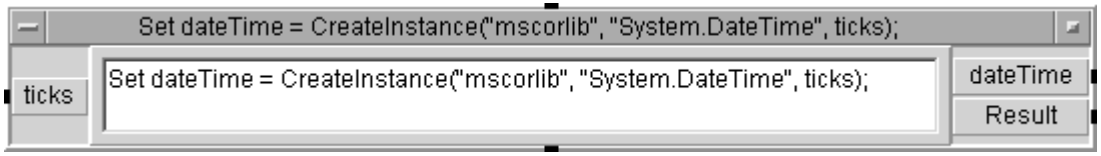


図166　　.NETオブジェクトの作成

4 .NETのEnumeration ( )は厳密に入力されるので、ActiveXオブジェクトで行ったようにEnumの代わりに整数定数を使用することはできません。完全なEnum名を(インポートした場合は名前空間をとって)入力するか、[Create Formula]ボタンを使用して、自分でタイピングをセーブする必要があります。たとえば、examples\dotnetディレクトリのEnum関連サンプルを参照してください。

NamespaceのVEEへのインポート

名前空間の定義を繰り返します。「.NET Framework型は、階層を暗示するドット構文ネーミング・スキームを使用します。この技術では、関連する型が名前空間にグループ化されるので、検索や参照がより簡単に行えます。完全な名前のうち、右端のドットまでの最初のパートは名前空間名です。名前の最後のパートは型名です。たとえば、System.Collections.ArrayListはArrayList型を表し、System.Collections*名前空間に所属します。System.Collections内の型は、オブジェクト集合の操作に使用できます。」

名前空間のインポートは、Visual Basicでの**[Imports]**ステートメントの使用またはC#での**[using]**ステートメントの使用と同じです。どちらの場合も、その名前空間からの型の使用を限定する必要はありません。VEEは、名前空間のインポート方法として、2つの方法をサポートします。[Devices]メニューから[Import .NET Namespaces]を直接呼び出すか、図168に示すように、[.NET Assembly References]のメニュー選択でチェックボックスを使用できます。

使用可能な.NET名前空間の一覧が表示され、隣にあるチェックボックスを選択するだけで名前空間をインポートできます。選択した**アセンブリ**に名前空間がない場合、リストは空です。

*ネストされたクラスは、この一般的なルールにあてはまりません。詳細については、MSDNのマニュアルを参照してください。

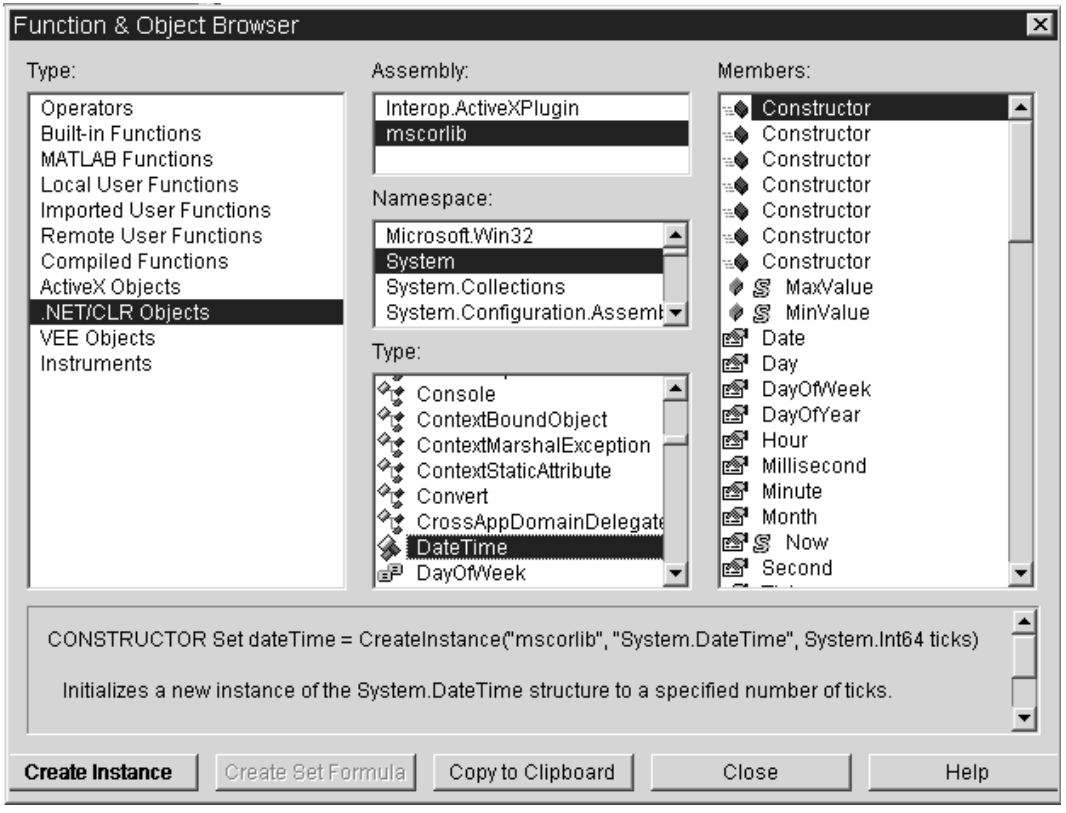


図167　　アセンブリ、名前空間、型、およびメンバ

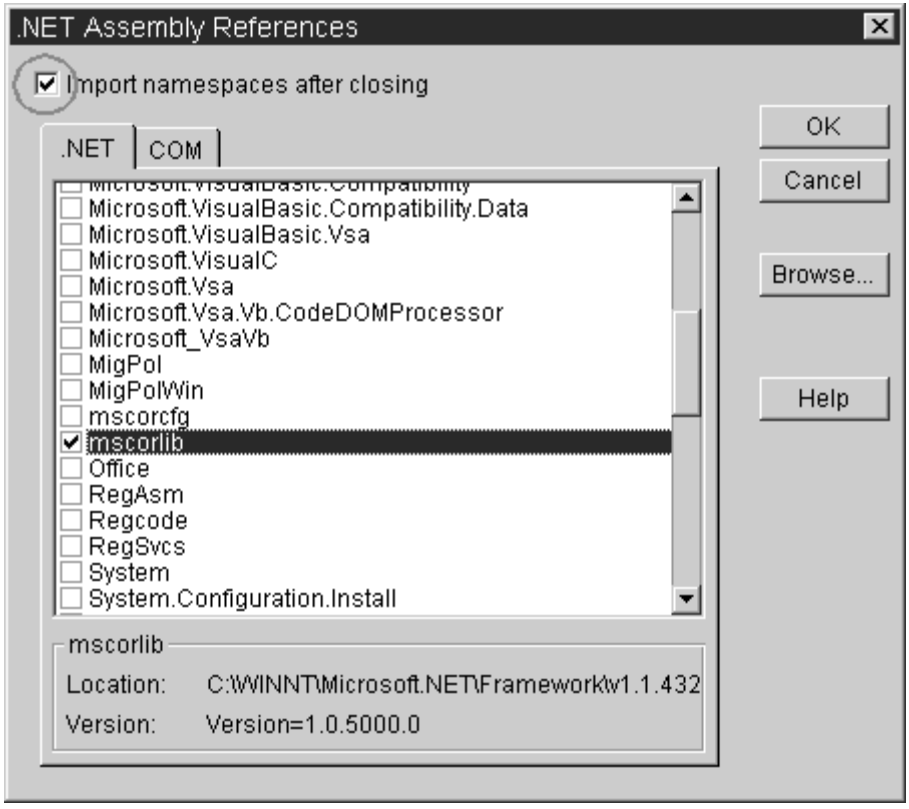


図168　.NET Assembly References - 名前空間のインポート

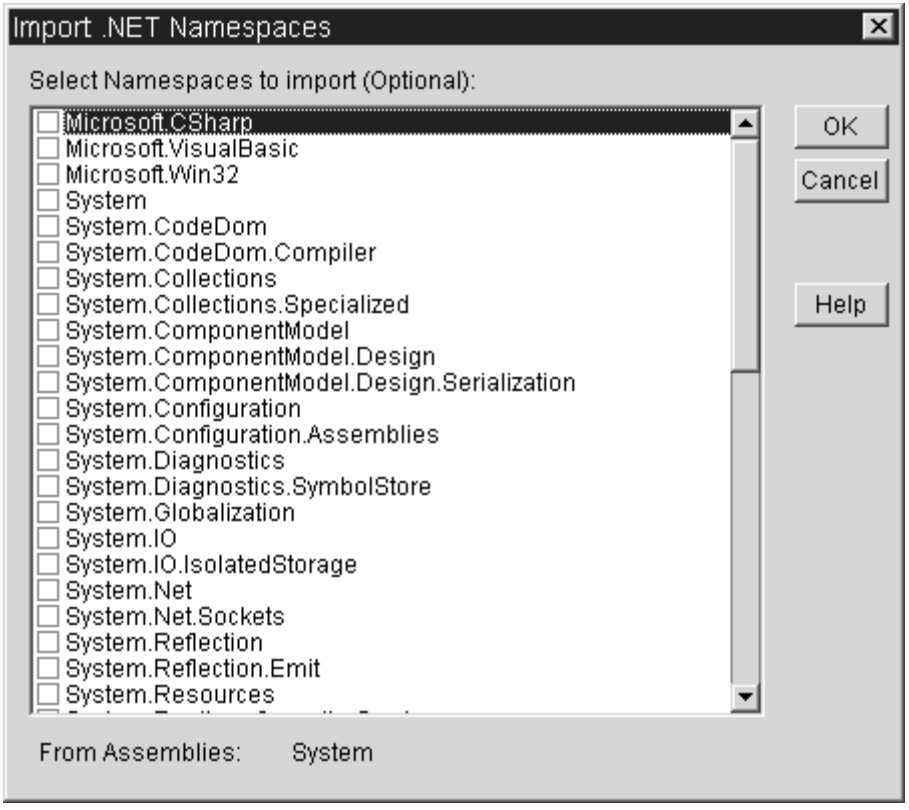


図169　　名前空間選択リスト

あるクラスに対して名前空間をまだインポートしていない場合、そのクラスの静的メンバおよび列挙に対して生成された公式テンプレートには、完全に限定された型名が含まれます。このため、公式ボックスがはるかに大きくなります。

VEEおよびプライマリ相互運用アセンブリ

[Device] ⇒ **[.NET Assembly References]**で[COM]タブを選択すると、VEEはレジストリを参照し、マシン上で登録されたすべてのCOMタイプ・ライブラリを検出します。COMタイプ・ライブラリを選択したときに、ライブラリにPIAがあり、COMタイプ・ライブラリのPrimaryInteropAssembly CodeBaseキーが登録されている場合、PIAの位置が[Function & Object Browser]の説明領域に表示されます。PrimaryInteropAssemblyCodeBaseは、PIAが/codebaseオプション付きで登録されたときにだけ登録されます。すべてのPIAがこの方法で登録されているわけではありません。現在、Agilent IVI-COMドライバはすべて、/codebaseオプション付きで登録されていますが、そうではないアセンブリも多くあります。また、COMタイプ・ライブラリをチェックしたときには、VEEはレジストリの検索しか行いません。VEEは相互運用アセンブリが存在する場合に、相互運用アセンブリをロードしません。**[OK]**を選択したとき、PIAがない場合には、1つまたは複数の相互運用アセンブリを自動的に生成します。相互運用アセンブリは、**[OK]**ボタンをクリックしたあとにもロードされます。

ただし**[Browse]**を選択し、PIAまたはInterop Assemblyを直接選択すると、VEEは、どのCOMタイプ・ライブラリに所属しているか、プライマリ相互運用アセンブリかどうかを見つけるために、実際に相互運用アセンブリをロードします。これは、COMタブ・リストの参照/選択と単なるチェックとの大きな違いです。相互運用アセンブリがロードされると、VEEが、どのCOMタイプ・ライブラリに所属しているかを見つけ、COMタブ・リストの下の対応するCOMタイプ・ライブラリをチェックマークします。

プログラミングの実習

.NETとVEE間のデータ型の変換

VEEは、VEEと.NETの両方が元来サポートするすべてのデータ型に対して、VEEと.NET間でデータ型を自動的に変換します。たとえば、Int16、Int32、Real64などです。この自動変換は「元に戻す」ことができるので、データを真の.NETオブジェクトとして使用できます。これらのケースでは、新しいasClrType()型変換関数を使用して、VEEデータ型を.NET/CLR型に変換します。詳細については、以下の変換テーブルを参照してください。

VEEの.NET操作は通常、パラメータとして精度が上がる(型昇格)データ型を受け入れますが、精度が下がる(型降格)データ型は受け入れません。たとえば、VEEのUInt8を、System.Int16型のパラメータを要求する.NET操作に渡すことができますが、他の道筋はありません。また、.NET操作がByRefパラメータを要求し、その結果を回収したい場合、データ型を正確に一致させる必要があります。たとえば、.NETメソッドがByRef Int32型のパラメータを要求する場合、VEEのByRef Int32データ型だけを渡すことができます。結果を回収する必要がない場合には、ByRefキーワードをスキップできます。これにより通常のパラメータ規則が適用されます。

次の表に、VEEと.NET間のデータ型変換関数を示します。

表31　VEE 7.0での.NETスカラ・データ型からVEEデータ型への変換

.NETデータ型からの変換	VEEデータ型への変換	注記
System.Boolean	Int16	isVariantBool()を使用して、VEE Int16がSystem.Boolean型であるか決定します。
System.Byte	Uint8	

System.Char	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()
System.DateTime	Object	`examples\dotnet\DateTime.vee`を参照してください。
System.Decimal	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32() 値がInt32 に適合しない場合、System.OverflowExceptionを示すエラー 751が発生します。System.Decimal構造自体も多数の変換方法を提供します。
System.Double	Real64	
System.Enum	Object	`examples\dotnet\FileInfo.vee`を参照してください。
System.Int16	Int16	
System.Int32	Int32	
System.Int64	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32() 値がInt32 に適合しない場合、System.OverlfowExceptionを示すエラー 751が発生します。
System.SByte	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt16()
System.Single	Real32	

System.String	Text	VEEは文字列をテキストに自動的に変換します。System.String クラスの機能を使用したい場合、asClrType()を使ってテキストをSystem.Stringに戻すことができます。例 Set dotNetString = asClrType(veeText, System.String); ModifiedString = dotnetString.Replace(" ","_"); 例については、examples\dotnet\stringsplit.veeも参照してください。
System.UInt16	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()
System.UInt32	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32() 値がInt32 に適合しない場合、System.OverflowExceptionを示すエラー 751が発生します。
System.UInt64	Object	System.Convert.To*メソッドを使用して、それをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32() 値がInt32 に適合しない場合、System.OverflowExceptionを示すエラー 751が発生します。
System.Object	Object	

表32　VEE 7.0でのVEEデータ型から.NETスカラ・データ型への変換

VEEデータ型からの変換	.NETデータ型への変換	注記
Int16	System.Int16	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Int32	System.Int32	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。

Real32	System.Single	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Real64	System.Double	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Text	System.String	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
UInt8	System.Byte	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
<型のスカラ *>	System.Boolean	Int16(UInt8、Int16、Int32、Real32、Real64、Text)にキャストできるスカラVEEデータ型でasVariantBool()を使用します。より汎用のasClrType()も使用できます。 例、`asClrType(veescalar, System.Boolean)`
Date/Time	System.DateTime	VEEの日付/時刻をReal64型としてストアします。`asClrType(veeDateTime, System.DateTime)`を使用します。 `examples\dotnet\DateTime.vee`を参照してください。
Object	System.Object	VEEが.NETオブジェクトに対するポインタを保持する場合

表33　VEE 7.0での.NET配列データ型からVEEデータ型への変換

.NETデータ型からの変換	VEEデータ型への変換	注記
System.Boolean Array	Int16 Array	isVariantBool()を使用して、配列がSystem.Boolean型であるか決定します。
System.Byte Array	Uint8 Array	

System.Char Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 公式CreateInstance("mscorlib","System.String", charArray).ToString();を使用して、配列をVEEが自動的に変換できる.NETデータ型に変換できます。
System.DateTime Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。
System.Decimal Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 System.Convert.To*メソッドを使用して、各配列メンバをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()。値がInt32に適合しない場合、System.OverflowExceptionを示すエラー 751が発生します。System.Decimal構造自体も多数の変換方法を提供します。
Double Array	Real64 Array	
System.Enum Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。
System.Int16 Array	Int16 Array	
System.Int32 Array	Int32 Array	
System.Int64 Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 System.Convert.To*メソッドを使用して、各配列メンバをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()。値がInt32に適合しない場合、System.OverflowExceptionを示すエラー 751が発生します。

System.SByte Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 System.Convert.To*メソッドを使用して、各配列メンバをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt16()。
System.Single Array	Real32 Array	
System.String Array	Text Array	VEEは、StringをTextに自動的に変換します。System.Stringクラスで多数の有効な関数を使用したい場合、asClrType()を使用してテキスト配列をSystem.String配列に変換できます。
System.UInt16 Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 System.Convert.To*メソッドを使用して、各配列メンバをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()。
SystemUInt32 Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 System.Convert.To*メソッドを使用して、各配列メンバをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()。値がInt32に適合しない場合、System.OverflowExceptionを示すエラー751が発生します。
System.UInt64 Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。 System.Convert.To*メソッドを使用して、各配列メンバをVEEが自動的に変換できる.NETデータ型に変換できます。例、ToInt32()。値がInt32に適合しない場合、System.OverflowExceptionを示すエラー751が発生します。
System.Object Array	Object	VEEオブジェクトはSystem.Array .NETオブジェクトに対するポインタを保持します。

表34　VEE 7.0でのVEE配列データ型から.NETデータ型への変換

VEEデータ型からの変換	.NETデータ型への変換 *	注記
Int16 Array	System.Int16 Array	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Int32 Array	System.Int32 Array	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Real32 Array	System.Single Array	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Real64 Array	System.Double Array	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
Text Array	System.String Array	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
UInt8 Array	System.Byte Array	asClrType()を使用して他の.NET/CLR型に変換します。asClrType()の説明書を参照してください。
< type *> Array	System.Boolean Array	Int16(UInt8、Int16、Int32、Real32、Real64、Text)にキャストできるVEEデータ型でasVariantBool()を使用します。より汎用のasClrType()も使用できます。 例、`asClrType(veeArray, System.Boolean)`
Date/Time Array	System.DateTime Array	VEEの日付/時刻は実際、Real64型です。asClrType(veeDateTimeArray, System.DateTime)を使用します。`examples\dotnet\DateTime.vee`を参照してください。

表35　　.NETデータ型修飾子

.NETデータ型修飾子	VEE型	注記
ref, out, ByRef	スカラまたは上記のマッピング・テーブルによって示された型の配列	VEEキーワードByRefを使用します。ByRefキーワードを使用しないと、.NET操作の例外は発生しませんが、通過したパラメータが変更されません。これはおそらく.NETクラス・デザイナが意図していなかったことです。

インスタンス・メソッドの呼び出し

インスタンス・メソッドを呼び出すときには、まず.NETオブジェクトを宣言し、作成します。[Function & Object Browser]および[Create Instance]ボタンを使用して、コンストラクタ公式テンプレートを最初に生成することをお勧めします。図173に、例を示します。

図170で、fileInfoは、System.IO.FileInfoオブジェクトとして初期化されます。fileInfo出力ピンを、このオブジェクト・インスタンスを要求する任意の後続公式ボックスの入力ピンに配線できます。CreateInstance公式ボックスとfileInfo.LastAccessTime公式ボックスの両方が、VEEによって自動的に作成されます。必要な操作は、それらをいっしょに配線し、追加パラメータを提供することだけです。

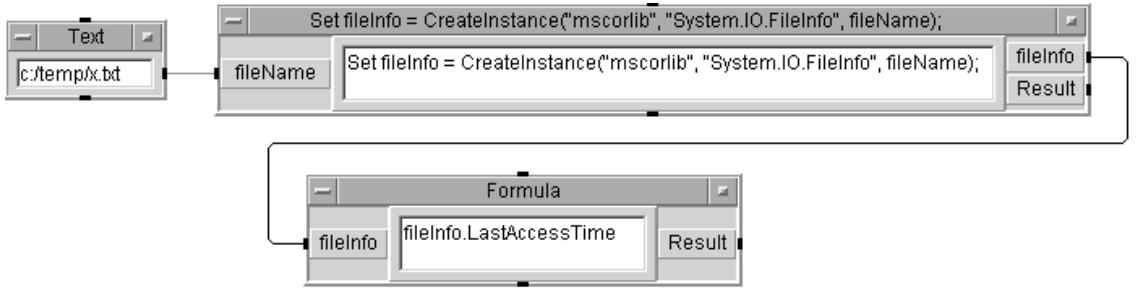


図170　　.NETオブジェクトの作成とインスタンス・メンバのアクセス

共有/静的メソッドの呼び出し

.NETメンバを呼び出すとき、メンバが共有/静的であると、[Function & Object Browser]の説明ボックスに「**STATIC**」と表示されます。静的メソッドは、オブジェクト上ではなく、.NETクラス上で直接呼び出されます。このため、最初に.NETオブジェクトを作成する必要がありません。図171に、例を示します。

図171　　名前空間をインポートしない静的メソッド

名前空間System.IOをインポートした場合、上記の公式テンプレートは図172のように表示されます。

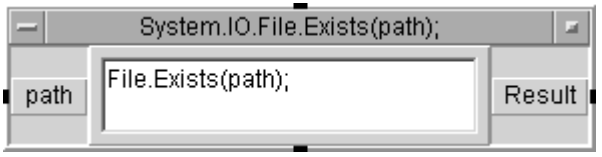


図172　　名前空間をインポートした静的メソッド

.NETプログラミングのヒント

- 特定の数値データ型を要求する .NET メソッドを呼び出しており、パラメータがたとえば入力ピン「A」から来る場合、入力ピンAをダブルクリックし、要求された適合型を設定すると、「Method Not Found」というエラーをしばしば回避できます。これは、組込みasClrType関数または.NET System.Convert.To*関数を呼び出すよりも簡単です。

- インスタンス・メンバ用に生成された公式テンプレートは常に、小文字で始まります。静的メンバは、その名前空間をインポートしていない限り、完全に限定された名前(名前空間＋クラス名)を持ちます。
- .NETメンバを呼び出すとき、メンバがStaticであると、説明ボックスに「**STATIC**」と表示されます。静的メソッドは、オブジェクトのインスタンス上ではなく、.NETクラス上で直接呼び出されます。このため、最初に.NETオブジェクトを作成する必要がありません。
- .NETのenumsは、COMのenumsと異なります。これらは厳密に入力されています。.NETには2種類のenumsがあり、1つは定数の列挙として処理され、もう1つはビット・フィールドとして処理されます。後者のenums型だけが、ビット幅操作用です。VEEでは、どちらのenums型も、[Function & Object Browser]の説明領域に表示された基礎整数値を持ちます。VEEの例、examples\dotnet\FileAttributes.veeで示すように、後者のenums型上でbitOr、bitAndなどを使用できます。
- [Function & Object Browser]に新しい**[Copy to Clipboard]**ボタンがあります。このボタンを使用して、公式テキストをクリップボードにコピーし、後からそれを公式ボックスの任意の場所に貼り付けることができます。これは、1つの公式ボックスの内部に複数のformulaステートメントを入れたいときに特に有効です。

例題7-1: .NETを使用したファイルの選択

型Systems.Windows.FormsOpenFileDialogの.NETオブジェクトを作成します。このオブジェクトを使用して、ファイル拡張子別にファイル・グループをフィルタし、ダイアログ・ボックスにファイル名を表示します。完全な例は、`examples\dotnet\OpenFileDialog.vee`にあります。

1. **[Device]**メニューから、**[.NET Assembly References]**を選択します。
2. **[.NET]**タブから、`System.Windows.Forms`チェックボックスを選択します。**[OK]**を選択します。
3. **[Function & Object Browser]**をオープンし、**[.NET/CLR Objects]**を選択します。
4. `System.Windows.Forms`名前空間を強調表示します。

5 `OpenFileDialog Type`と**Constructor** Memberを強調表示します。**[Create Instance]**を選択します。

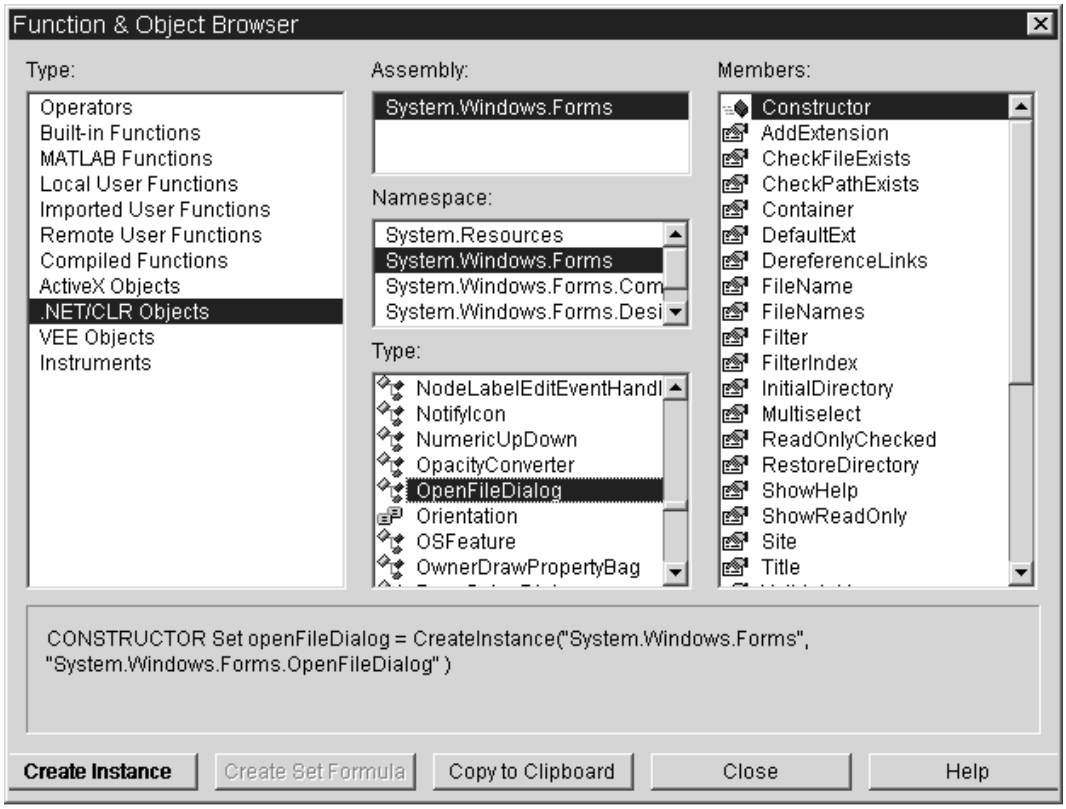


図173　[Function & Object Browser]でのインスタンスの作成

6 グローバル変数`openFileDialog`を宣言し、その型を`Object`に設定します。

7 openFileDialogをグローバル変数として宣言したので、**[CreateInstance]**公式ボックスからopenFileDialog出力ピンを削除します。

8 **CreateInstance("System.Windows.Forms","System.Windows.Forms.OpenFileDialog")** Formulaオブジェクトのタイトルを**Open File Dialog**に変更します。オブジェクト内で、次の手順を実行します。

   a プログラムがダイアログ・ボックスをオープンするディレクトリを設定します。

   b openFileDialogプロパティ**Multiselect**を`True`に設定します(ヒント：System.Booleanを取得するには、asVariantBoolまたはasClrTypeが必要です)。これにより、**[Open File Dialog]**ボックスから複数のファイルを選択できます。

   c ファイル拡張子フィルタ、この場合はDLLファイルを設定します。

   d openFileDialogメソッドShowDialogを呼び出し、ダイアログ・ボックスを表示します。

   e openFileDialogプロパティFileNamesを呼び出し、選択したDDLファイルのリストを検索します。このリストは、Logging Alphanumericオブジェクトに現われます。

プログラムは、図174のように表示されます。

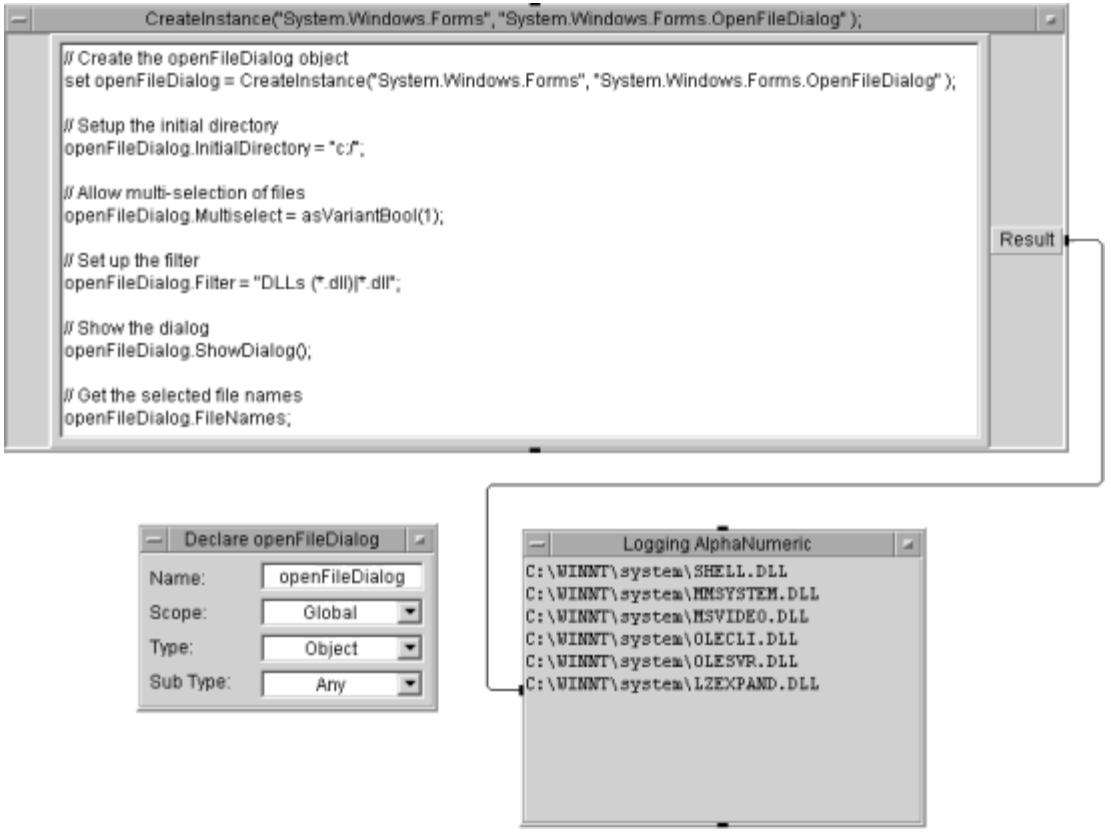


図174　openFileDialogプログラム

例題7-2: .NETを使用したDateTime操作の実行

.NETのDateTime型のインスタンス・メンバと静的メンバの両方を使用します。.NETのDateTime型は、VEEのDateTime関数よりも多くの機能を提供します。この例題では、DateTime型を使用して現在のDateTime、現在の曜日、現在の年を取得し、うるう年かどうかをクエリします。完全な例は、`examples\dotnet\DateTime.vee`にあります。

1 **[Device]** ⇒ **[.NET Assembly References]**を選択します。

2 **[.NET]**タブから、**[mscorlib]**チェックボックスを選択します。**[OK]**を選択します。

3 **[Import .NET Namespaces]**ダイアログ・ボックスで、**[System]**チェックボックスを選択します。**[OK]**を選択します。

4 **[Function & Object Browser]**をオープンし、**[.NET/CLR Objects]**を選択します。

5 **System**名前空間を強調表示します。

6 **DateTime**型と静的プロパティ**Now**を強調表示します。**[Create Get Formula]**ボタンを選択します。静的メンバであるため、最初にDateTimeオブジェクトのインスタンスを作成または取得する必要がありません。

7 ステップ4と5を繰り返して、`DateTime`型を強調表示します。複数のバージョンのToStringメソッドがあります。パラメータを要求しない、一番単純なバージョンを強調表示します。**[Create Call]**ボタンを選択します。

8 **[DateTime.Now]**公式ボックスの結果ピンを**[dateTime.ToString()]**公式ボックスのdateTime入力ピンに配線します。

9 Alphanumeric を追加し、それを **[dateTime.ToString()]** 公式ボックスの出力ピンに配線します。

10 ステップ4と5を繰り返して、`DateTime`型とプロパティ **[DayOfWeek]** を強調表示します。**[Create Get Formula]** ボタンをクリックします。DayOfWeekプロパティが、System.DayOfWeek型の別の.NETオブジェクトを返します。すべての.NETオブジェクトにToString()メソッドがあるので、それを使って曜日をフォーマットし、印刷できます。セミコロンの前に`".ToString()"`を追加することにより、VEEで作成したばかりの公式を編集します。

11 **[DateTime.Now]**公式の結果ピンを**[dateTime.DayOfWeek]**公式ボックスのdateTime入力ピンに配線します。

12 公式の結果ピンを **[dateTime.DayOfWeek]** 公式ボックスの dateTime 入力ピンに配線します。

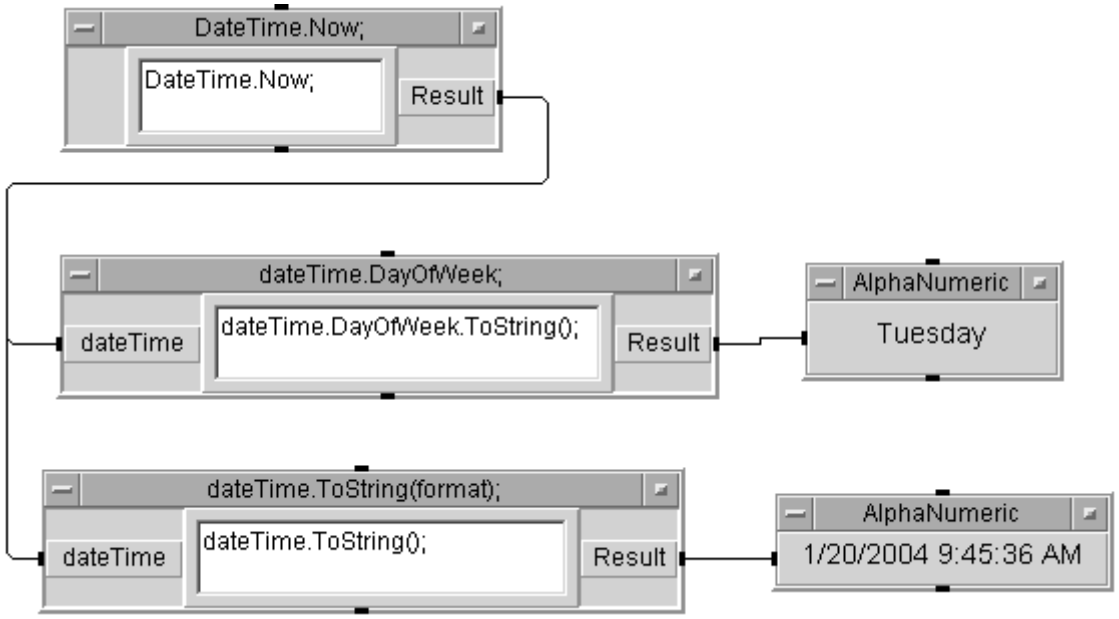


図175　　例題7-2のステップ10

13 ステップ4と5を繰り返して、**[DateTime]**型とプロパティ**[Year]**を強調表示します。**[Create Get Formula]**ボタンを選択します。

14 **[DateTime.Now]**公式ボックスの結果ピンを**[dateTime.Year]**公式ボックスのdateTime入力ピンに配線します。

15 Alphanumericを追加し、その入力ピンを**[dateTime.Year]**公式ボックスの出力ピンに配線します。

16 ステップ4と5を繰り返して、**[DateTime]**型と静的メソッド**[IsLeapYear]**を強調表示します。**[Create Call]**ボタンを選択します。静的メソッドであるため、Datetimeオブジェクトのインスタンスを作成または取得する必要がありません。

17 **[dateTime.Year]**公式ボックスの結果ピンを**[DateTime.IsLeapYear]**公式ボックスのyear入力ピンに配線します。

18 Alphanumericを追加し、その入力ピンを**[DateTime.IsLeapYear]**公式ボックスの出力ピンに配線します。

完了した例題を図176に示します。

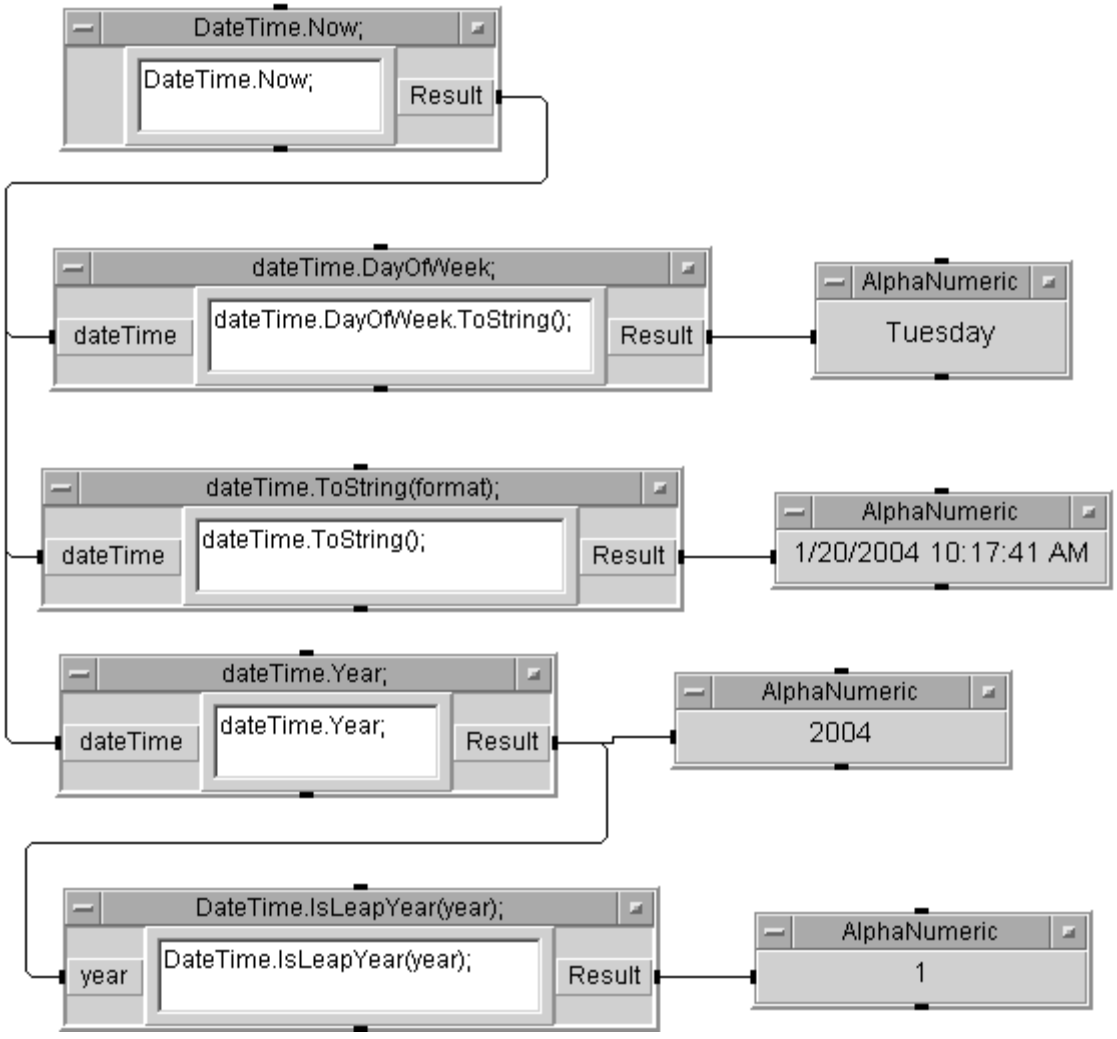


図176　　完了した例題7-2

例題7-3: .NETを使用したファイル情報の取得

.NET System.IO名前空間は、非常に多くのファイル・システム処理機能を提供します。機能へのアクセスも、これまで以上に簡単になっています。この例題では、ファイルに作成時間、最後のアクセス時間、長さについてクエリします。完全な例は、`examples\dotnet\FileInfo.vee`にあります。

1. **[Device]** ⇒ **[.NET Assembly References]**を選択します。
2. **[.NET]**タブから、**[mscorlib]**チェックボックスを選択します。**[OK]**を選択します。
3. **[Import .NET Namespaces]**ダイアログ・ボックスが表示される場合、[**System** namespace]をチェックするか、単に**[Cancel]**を選択します。この例題では、静的メンバやenumメンバを使用しませんが、完全な例では使用します。
4. **[Function & Object Browser]**をオープンし、**[.NET/CLR Objects]**を選択します。
5. **[System.IO]**名前空間を強調表示します。
6. **[FileInfo]** Typeと**[Constructor]**メンバを強調表示します。**[Create Instance]**ボタンを選択します。生成された公式テンプレートを調べます。**[fileInfo]**という出力ピンがVEEによって作成され、新しく作成された**[FileInfo]**オブジェクトに設定されます。
7. **[Data]** ⇒ **[Dialog Box]** ⇒ **[File Name Selection]**を選択します。**[File Name]**出力ピンを、**[Create Instance]**公式ボックスの**[fileName]**入力ピンに配線します。**[File Name Selection]**オブジェクトをアイコン化します。
8. ステップ4と5を繰り返して、**[FileInfo]**型を強調表示します。**[CreationTime]**プロパティを強調表示します。**[Create Get Formula]**ボタンをクリックします。CreateTimeプロパティが.NET DateTiimeオブジェクトを返すので、そのToString()メソッドを使用して作成時間をフォーマットし、印刷します。VEEにより生成された公式の後に`".ToString()"`を追加します。セミコロンの前に追加してください。
9. **[CreateInstance]**公式ボックスからのfileInfo出力ピンを、**[fileInfo.CreationTime]**公式ボックスのfileInfo入力ピンに配線します。
10. Alphanumericを追加し、それを**[fileinfo.CreationTime]**公式ボックスの出力ピンに配線します。

11 ステップ8と9を繰り返して、**[fileInfo.LastAccessTime]**プロパティの公式ボックスを配線します。

12 Alphanumericを追加し、それを**[fileinfo.LastAccess]**公式ボックスの出力ピンに配線します。

13 ステップ8と9を繰り返して、**[fileInfo.Length]**プロパティの公式ボックスを配線します。

14 Alphanumericを追加し、それを**[fileinfo.Length]**公式ボックスの出力ピンに配線します。

完了した例題は、図177のように表示されます。

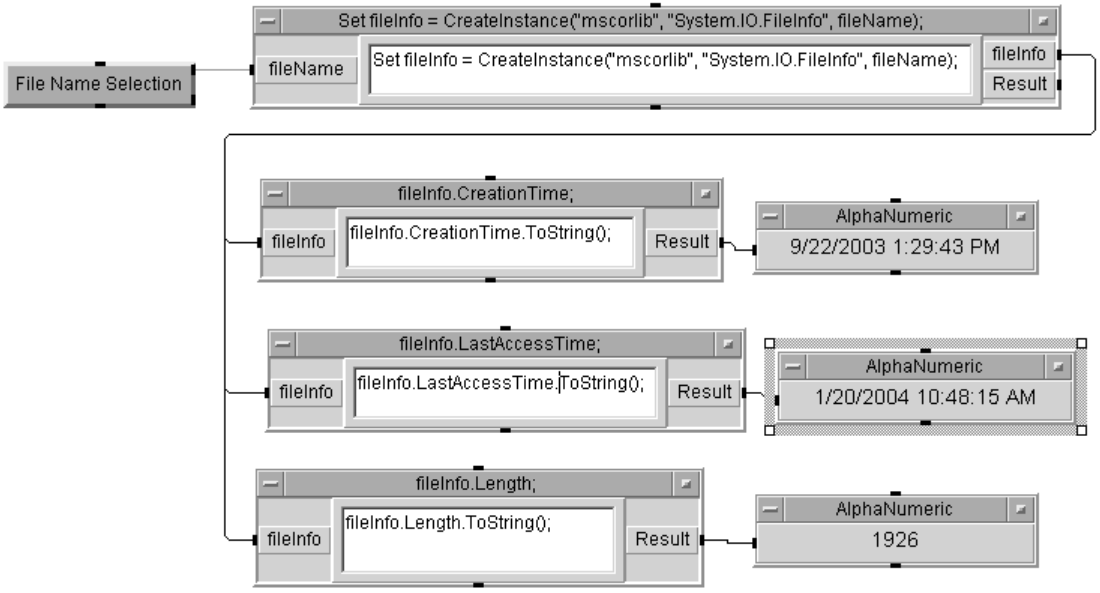


図177　　完了した例題7-3

.NETおよびIVIドライバ

IVIとは何でしょうか。IVIは、IVI Foundationによって開発されている新しい計測器ドライバ標準です。IVI Foundationは、テスト計測器のプログラミングの仕様を広めるために設立されたコンソーシアムです。これらの仕様は、さまざまな利点を備えていますが、重要な点として、計測器の相互交換性を容易にします。IVI Foundationの概要については、http://www.ivifoundation.org/をご覧ください。

IVI-COMドライバを使用する理由は何でしょうか。2つのハードウェア・ベンダがそれぞれのDMMに対して1つのIVI-COMドライバを提供しているとします。IVI標準を満足するには、基本ドライバが相互交換可能でなければなりません。相互変換可能であれば、はじめにあるベンダのDMMと適合ドライバを使用し、プログラムを変更することなく、途中で別のベンダのDMMとドライバに切り替えることができます。コードは、介入なしに再使用可能です。

必要に応じて、IVI-COMドライバをインストールします。計測器用のドライバを見つけるには、http://www.agilent.com/find/adnをご覧ください。ADNメンバである場合、メニュー ⇒ **[Drivers by Driver Type]** ⇒ **[IVI-COM Drivers and Components]**から[Downloads]を選択します。ADNのメンバでない場合、Agilentドライバ、評価ソフトウェア、マニュアル、白書について、オンライン・ソースをご覧ください。この無料サービスに登録するには、新規ユーザ用のオンライン登録フォームに必要事項を記入します。

システムにIVI-COMドライバが登録されると、他のCOMコンポーネントと同様に使用可能になります。ほとんどのCOMコンポーネントと同じく、VEEの**[Device]** ⇒ **[.NET Assembly References]**の[COM]タブに表示されます。

VEEと.NETでIVI-COMドライバを使用可能にするにはどうすればよいでしょうか。.NETのCOM相互運用を介して、IVI-COMドライバがVEEに対して使用可能になります。選択したCOM DLLに使用可能なプライマリ相互運用アセンブリがない場合、VEEは、相互運用アセンブリを1つ、あるいはCOM .dllが他のタイプ・ライブラリを参照する場合は複数、生成しようとします。相互運用アセンブリが参照されると、IVI-COMドライバを、他の.NETオブジェクトと同様に起動できます。

IVI-COMの詳細については、www.agilent.com/find/adnを参照してください。ログインし、ナレッジ・ライブラリに進み、IVI-COM情報を選択します。この場所には多数の白書があります。

アセンブリ

新規アセンブリのインストール

アセンブリは、.NET Frameworkの基本機能ブロックです。アセンブリは、実行可能(.exe)ファイルまたはダイナミック・リンク・ライブラリ(.dll)ファイルの形をとります。

アセンブリは、Global Assembly Cache(GAC)にインストールできます。GACはマシン全体のリソースであるため、適切な特権が必要であり、特別の注意が必要です。GACにインストールされたアセンブリは、コンピュータ上のどのアプリケーションでも表示できます。詳細については、MSDNを参照してください。共有アセンブリが更新されたら、**[Browse]**ボタンを選択し、GACにインストールされる前のアセンブリの元の場所までブラウズすることにより、アセンブリを参照できます。

ほとんどのアセンブリは、GACにインストールされません。こうしたアセンブリは、プライベート・アセンブリと呼ばれます。プライベート・アセンブリは、プライベート・アセンブリを参照するVEEプログラムといっしょに配置する必要があります。最初に[.NET Assembly References]ダイアログ・ボックスを使用してプライベート・アセンブリを参照し、選択すると、まだ配置されていない場合には、VEEがそれを現在のVEEプログラムと同じディレクトリにコピーします。このコピー操作は、VEEプログラムの配布にも役立ち、xcopy展開と呼ばれています。

アセンブリのアップデート

VEEプログラム内で共有アセンブリをすでに参照したことがある場合、新バージョンの共有アセンブリをVEEでロードするには、File/Newを実行するか、VEEをリスタートする必要があります。共有アセンブリではなくプライベート・アセンブリである場合、[File/New]を実行するか、プライベート・アセンブリの新規バージョンを現在のVEEプログラム・ディレクトリに手動でコピーした後、VEEを再起動する必要があります。追加情報については、314ページの「VEE Runtimeの配布」を参照してください。

VEE Runtimeの配布

VEEプログラムがGACにインストールされた共有アセンブリを参照する場合、これらのアセンブリを宛先マシンのGACにインストールする必要があります。共有アセンブリがMicrosoft .NET Frameworkアセンブリである場合、Microsoft .NET Framework redistパッケージをインストールしたときにアセンブリはすでにインストールされているはずです。

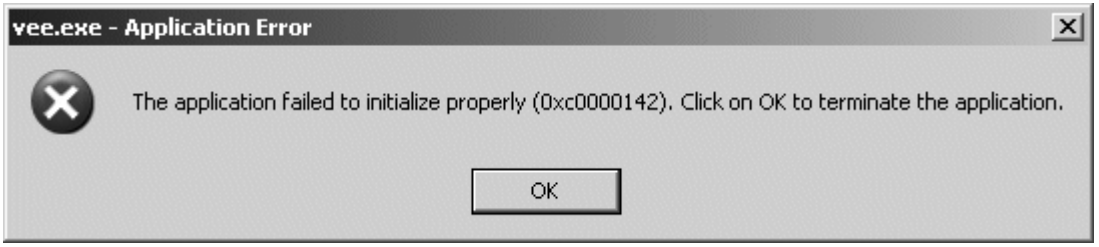
注 記

.NET Framework redistパッケージは、VEE Runtimeのインストール前にインストールする必要があります。

VEEは、参照された任意のプライベート・アセンブリをVEEプログラムと共にセーブします。VEEプログラムがプライベート・アセンブリだけを参照する場合は、VEEプログラムのディレクトリ全体を宛先マシンにコピーすることもできます。配布の場合、すべてのCOMタイプ・ライブラリを宛先マシンに登録する必要があるので、この方がCOMタイプ・ライブラリを参照するVEEプログラムを配布するよりもはるかに簡単です。

VEEおよび.NET Security

VEEで.NET Frameworkを使用する場合、セキュリティ問題が発生します。次のダイアログ・ボックスが表示されたら、.NET Frameworkセキュリティ・プロファイルに対して以下の調整を行ってください。



1. システム・ドライブに移動します。
2. ディレクトリ\[winntまたはwindows]\Microsoft.NET\Framework\v1.1.4322に移動します。
3. コマンド・プロンプトから、以下のコマンドを実行します。

Caspol Dmachine –addgroup “All_Code” –url
File://[VEEInstallDir]/*　FullTrust

VEEInstallDirは、ネットワーク・ドライブ上のVEEインストール・ディレクトリです。Caspolは.NET Framework Redistパッケージに付属しており、VEEをインストールするときにインストールされます。.NET Framework Configuration Wizardを実行しても、同じ結果が得られます。どちらのツールを実行する場合にも、管理者特権が必要となります。

これに加えて、VEEプログラムが.NETアセンブリを参照する場合にはしばしば、VEEは、アクセス可能な場所でアセンブリを生成するか、アクセス可能な場所にアセンブリをセーブする必要があります。このため、たとえば、ローカルにセーブせずに電子メールまたはネットワークからVEEプログラムを.NET参照でオープンすると、警告が表示される場合があります。

.NETの用語

次の用語が章全体で使用されているため、以下のMicrosoft定義をこの章を読む際のリソースとして使用できます。また、復習にも利用できます。

アセンブリ

「コンポーネントはアセンブリにパッケージされます。アセンブリは、.NETアプリケーションの再使用可能、かつバージョン作成可能な、内容が一目でわかる機能ブロックです。一番単純なアセンブリは、配置やバージョン作成に必要な情報をすべて含んだ、1個の実行可能プログラムです。アセンブリは、.NET Frameworkの基本機能ブロックです。アセンブリは、実行可能(.exe)ファイルまたはダイナミック・リンク・ライブラリ(.dll)ファイルの形をとります。」

Primary Interop Assembly (PIA)

「プライマリ相互運用アセンブリは、COMで実装された型の型定義をメタデータとして含む、一意のベンダ供給アセンブリです。プライマリ相互運用アセンブリは1つだけです。プライマリ相互運用アセンブリは、COMタイプ・ライブラリの権威ある発行者による承認を受ける必要があります。」

名前空間

「.NET Framework型は、階層を暗示するドット構文ネーミング・スキームを使用します。この技術では、関連する型が名前空間にグループ化されるので、検索や参照がより簡単に行えます。完全名のうち、右端のドットまでの最初のパートは名前空間名(*)です。名前の最後のパートは型名です。たとえば、System.Collections.ArrayListはArrayList型を表し、System.Collections名前空間に所属します。System.Collectionsにある型は、オブジェクト集合の操作に使用できます。」

*ネストされたクラスは、この一般的なルールにあてはまりません。詳細については、MSDNのマニュアルを参照してください。

参照

アセンブリを使用するには、アセンブリに参照を追加する必要があります。

クラス

「オブジェクト指向プログラミングに慣れている場合、クラスが、オブジェクトが実行できる操作(メソッド、イベント、プロパティ)を定義し、オブジェクト(フィールド)の状態を保持する値を定義することをご存知のはずです。クラスは通常、定義と実装の両方を含みますが、実装を持たないメンバを1つ以上持つこともできます。

クラスのインスタンスは、1つのオブジェクトです。オブジェクトの機能にアクセスするには、そのメソッドを呼び出し、そのプロパティ、イベント、フィールドにアクセスします。」

共有または静的メンバ

「共有メンバは、クラスのすべてのインスタンスによって共有されるプロパティ、プロシージャ、フィールドです。一部のプログラミング言語は、こうしたアイテムを静的メンバとして参照します。

共有フィールドと共有プロパティは、クラスの一部である情報を持っているときには有効ですが、クラスのどのインスタンスにも固有ではありません。通常のフィールドおよびプロパティは、クラスの各インスタンスに対して独立して存在します。あるインスタンスに関連付けられたフィールドまたはプロパティの値を変更しても、クラスの他のインスタンスのフィールドやプロパティの値には影響しません。一方、クラスのインスタンスに関連付けられた共有フィールドおよびプロパティの値を変更すると、クラスの全インスタンスに関連付けられた値を変更することになります。このように、共有フィールドおよびプロパティは、クラスのインスタンスからのみアクセスできるグローバル変数のように動作します。」

インスタンス・メンバ

インスタンス・メンバは、クラスの特定のインスタンスに結合されます。その値に対する変更は、それが関連付けられたオブジェクトにだけ影響し、その他のオブジェクトには影響しません。

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。

- .NETの基本用語を知る。
- 名前空間をVEEにインポートする方法を知る。
- PIAを定義する。
- VEE Runtimeを配布する方法を知る。
- .NETセキュリティをVEEに合わせて調整する方法を知る。

8
PC用プログラムの統合方法

異なる言語で書かれたプログラムの統合方法

この章の内容

- Execute Programオブジェクト
- VEEでオペレーティング・システムのコマンドを使用する方法
- VEEプログラムをプラットフォーム間で移植可能にする方法

平均所要時間: 30分

概要

この章では、VEEでコンパイル済みプログラムとオペレーティング・システムのコマンドを統合する最も簡単な方法を習得します。VEEの大きな利点の1つは、ほかのアプリケーションやプログラムとうまく融合できることです。また、ActiveXにより、ほかのプログラムのコンポーネントを使用することもできます。詳細は、第6章「ActiveXを使ったレポートの簡単な作成方法」を参照してください。

VEEでは、Execute Programオブジェクトでプログラムとパラメータを指定し、オペレーティング・システムのコマンドを使用します。PC用のExecute Programオブジェクトがあります。この章では、PC用のExecute Programオブジェクトを使用する例題を紹介します。

Execute Programオブジェクトについて

VEEからほかの言語のプログラムを実行する方法には、ActiveXオートメーション以外にも、次の3つがあります。

1. Execute Programオブジェクトを使用して、VEEをエスケープし、別のプログラム、アプリケーション、またはオペレーティング・システム・コマンドを実行します。これは最も用途が広く、使いやすい方法です。
2. PC のダイナミック・リンク・ライブラリを介して、ほかの言語で書かれたコンパイル済みの関数をVEEにリンクします。これは少し難しい方法ですが、パフォーマンスが大幅に向上します。ダイナミック・リンク・ライブラリについての詳細は、453ページの「ダイナミック・リンク・ライブラリの使用法」を参照してください。

Execute Programオブジェクトは[I/O]メニューにあります。図178に示すように、PC用に1個のオブジェクトがあります。Execute Programオブジェクトは、プログラムとの通信にトランザクションI/Oを使用しないため、コンパイル済みプログラムにデータを渡すためのデータ入力ピンとデータ出力ピンを追加する必要がありません。

Execute Programオブジェクトの使用方法

図178は、PCの**Execute Program**オブジェクトを示します。

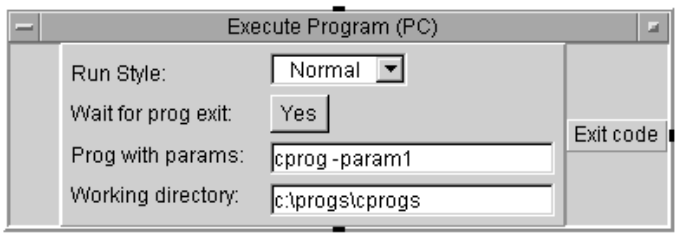


図178　Execute Programオブジェクト(PC)

Execute Programオブジェクトを使用して、VEEから次を実行します。

- ほかの言語で記述されたコンパイル済みプログラム
- *.BATファイルまたは*.COMファイル
- dirなどのMS DOSシステム・コマンド

- 認識されている拡張子を持つ文書またはURL。このファイルの中で「オープン」操作が呼び出されます。「オープン」操作を持たないファイルでは、デフォルトの操作が呼び出されます。
  http://www.agilent.com/find/veeがURLの一例です。

Execute Programオブジェクトのフィールドについて、以下で説明します。

表36　Execute Programオブジェクトのフィールド

フィールド名	説明
Run Style	ウィンドウのサイズを決めます。[Normal]は標準ウィンドウ、[Minimized]はアイコン、[Maximized]は最大ウィンドウ・サイズを指定します。[Working directory]は、プログラムに関連のあるファイルを保持するディレクトリです。
Wait for prog exit	シーケンス・ピンをいつ起動するかを指定します。 • Yesに設定した場合、プログラムが実行を完了するまで、シーケンス・ピンは起動されません。 • Noに設定した場合、指定されたプログラムが実行を完了する前に、シーケンス出力ピンが起動されます。文書またはURLを起動したとき、その文書またはwebサイトが、すでに動作しているアプリケーションにロードされている場合、VEEはアプリケーションが終了するまで待たないことに注意してください。
Prog with params	(パラメータ付きプログラム) このフィールドには、DOSプロンプトに入力するものと同じ文字列が保持されます。たとえば、C言語で書いたプログラムを実行するには、次のように実行可能ファイル名を入力します。 - **myprog.exe** (拡張子 `.exe`は省略可能) • プログラムがパラメータを持っている場合は、次のように、実行可能ファイル名の後に、先頭にハイフンを付けてパラメータを入力します。**myprog -param1 -param2** • DOSのシステム・コマンドを実行するには、最初に、`/c`オプションを使用してDOSコマンド・インタプリタを実行します。たとえば、Windows 98の場合は、コマンド `command.com /c` <**システム・コマンド**>を入力します。 • Windows NT 4.0、Windows 2000、Windows XPの場合は、コマンド`cmd /c` <**システム・コマンド**>を入力します。 このオプションは、コマンド・インタプリタに、`/c`の後に続く文字列をシステム・コマンドとして読み取るように指示します。

システム・コマンドの使用方法

別の言語で書かれたコンパイル済みプログラムを呼び出すには、実行可能ファイルと、パラメータがあればパラメータをExecute Programオブジェクトに入力します。

ただし、MS DOSのシステム・コマンドを実行するには、最初にDOSコマンド・インタプリタを実行する必要があります。この例題では、DOSコマンド・インタプリタを実行し、次に、MS DOSのシステム・コマンドを実行します。

例題8-1: システム・コマンドの使用方法

1 **[I/O]** ⇒ **[Execute Program]**を選択します。**[Prog with params]**フィールドをクリックしてカーソルを表示し、次のように入力します。

```
command.com /c dir >> c:\bob
```

注 記

Windows NT 4.0、Windows 2000、またはWindows XPの場合は、command.comをcmdに置き換えます。ドライブ名がc:と異なる場合は、次の手順に従って、そのドライブ名を置き換えます。NTでは、書込み権限を持っているディレクトリを指定する必要がある場合もあります。

command.com実行可能ファイルの完全パスを含める必要がある場合もあります。このコマンドはDOSコマンド・インタプリタを実行します。DOSコマンド・インタプリタは、システム・コマンドを実行して現在のディレクトリを表示し、出力**(>)**を、コンピュータの画面ではなく、**bob**ファイルにリダイレクトします。

[Wait for prog exit]の選択では、[Yes]のままにしておきます。[Run Style]では[Normal]のままにしておき、[Working directory]に「c:\」と入力します。

2 **[I/O]** ⇒ **[From]** ⇒ **[File]**を選択し、Execute Programの下に配置します。Execute Programのシーケンス出力ピンを**From File**オブジェクトのシーケンス入力ピンに接続します。

myFileというラベルの付いた[From File:]入力フィールドをクリックしてリスト・ボックスを表示し、「**c:\bob**」と入力してから、**[OK]**をクリックします。プログラムがファイル**bob**を作成してくれます。

3 トランザクション・バーをダブルクリックして、[I/O Transaction]ボックスを表示します。

   a **[REAL64 FORMAT]**を**[STRING FORMAT]**に変更します。

   b **[SCALAR]**を**[ARRAY 1D]**に変更します。

   c **[SIZE : (10)]**フィールドをクリックして、**[TO END : (*)]**にトグルしてから、**[OK]**をクリックします。トランザクション・バーは、**READ TEXT x STR ARRAY:***となっているはずです。このトランザクションがbobファイルの内容を読み取ります。

4 **[Display]** ⇒ **[Logging AlphaNumeric]**を選択し、データ入力ピンをFrom Fileのデータ出力に接続します。

5 プログラムを実行します。図179のように表示されます。

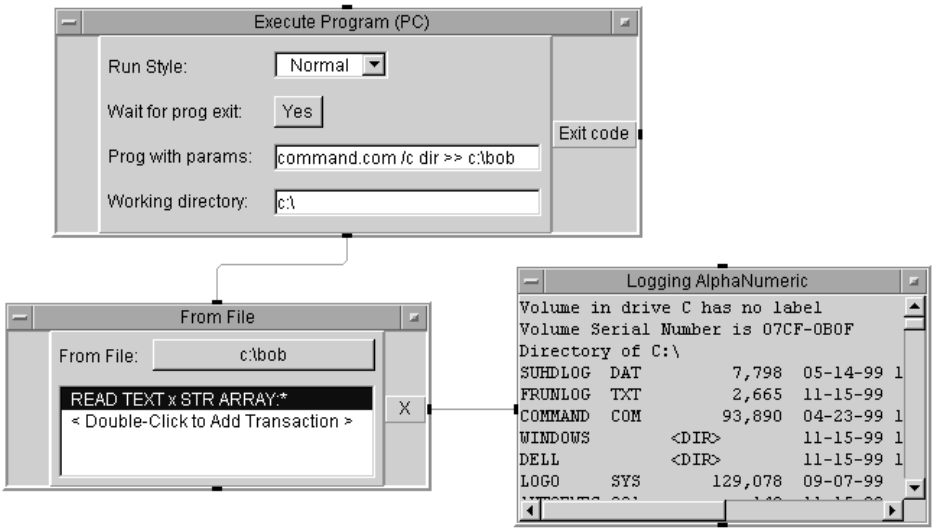


図179　ディレクトリ内のファイルの一覧表示方法

容易に移植できるプログラムの作成方法

異なる言語で書かれた複数のプログラムの統合を計画している場合は、それらのプログラムを他のオペレーティング・システムへ容易に移植できるようにVEEプログラムを作成します。図180に示すように、VEEは、**[Function & Object Browser]** ⇒ **[System Information]**に、システム情報に関するオブジェクトを保持しています。これらのオブジェクトは関数としても使用できます。

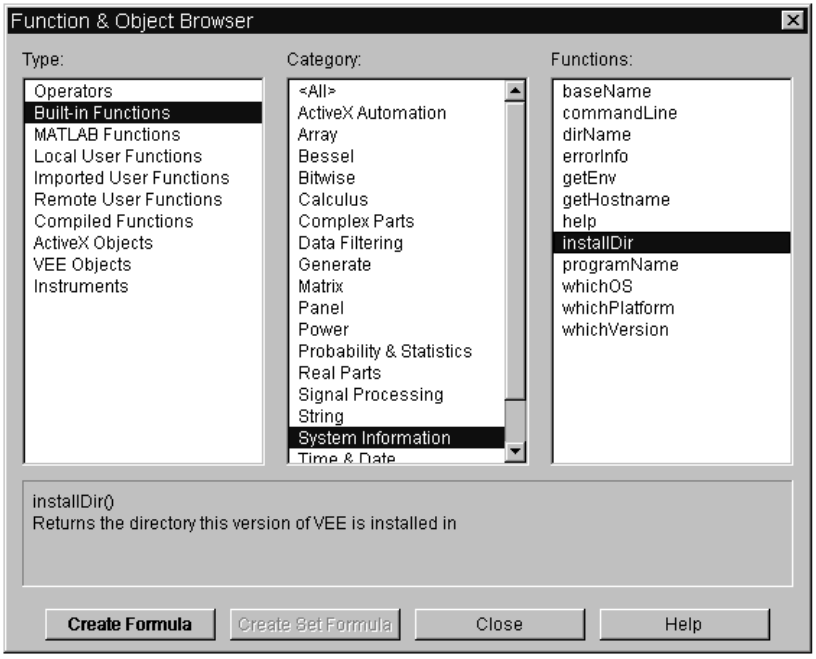


図180　　システム情報関数

Function & Object Browserにある、プログラムの移植性を向上させるためによく使用されるシステム情報関数について、以下で説明します。

表37　システム情報関数

関数名	説明
installDir	VEEをインストールしているディレクトリを指定します。
whichOS	オペレーティング・システムを判別し、次のいずれかの文字列を送信します。**Windows_98**、**Windows_2000**、**Windows_NT**、**Windows XP**。 異なる言語で書かれた複数のプログラムを統合する場合、プログラムは、上記の結果に基づいて分岐できます。たとえば、**examples\manual**ディレクトリの**manual49.vee**でwhichOS()を使用するプログラムを調べ、正しい種類のライブラリをインポートしているかを確認します。whichOS()は、PCでは、ダイナミック・リンク・ライブラリをインポートします。
whiclatform	VEEが実行されているハードウェア・システムを判別し、そのプラットフォームを示す文字列を返します。
whichVersion	VEEのバージョンを指定します。バージョンは、プログラムの保守とデバッグに役立ちます。

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じてトピックを復習してください。

- Execute Programオブジェクトの目的について説明できる。
- Execute Programオブジェクトにある設定値の概要について説明できる。
- Execute Programオブジェクトが、PCプラットフォーム上のプログラムとの間でデータを送受信する一般的なプロセスについて説明できる。
- VEEからオペレーティング・システムのコマンドを実行する。
- whichOS()、whichplatform()、またはwhichVersion()オブジェクトを使用して、異なるオペレーティング・システム上で実行されるプログラムを作成する。

9
Agilent VEE 関数の使用方法

Agilent VEE 関数の使用方法

この章の内容

- ユーザ関数を定義する方法
- 関数の作成、呼び出し、編集を行う方法
- 関数ライブラリを作成、統合、インポート、削除する方法
- 大型プログラムの関数を検索する方法
- 既存のVEEプログラムとテストをマージする方法

平均所要時間: 1時間

概要

この章では、VEEUserFunction、コンパイル済み関数、リモート関数について説明します。関数は、テスト開発時間の大幅な短縮に役立つ再利用可能なモジュール式のコードです。以前のプログラムで作成した関数を使用すると、既存の作業が流用でき、プログラムのコードのサイズも減少するので、テスト・プログラムの保守が容易になります。関数を、ライブラリとしてグループで使用することもできます。作成したライブラリは新しいプログラムにマージできます。複数のプログラムや複数の開発者間で関数を共有することができます。

関数の使用方法

多くのプログラミング言語と同じように、VEEも、特定のタスクを実行するサブプログラムを作成する場合に関数を使用します。この章の例題では、VEEのユーザ定義関数の作成、呼び出し、編集の方法について説明します。関数のライブラリを作成する方法についても説明します。ライブラリは、開発段階でプログラムにマージしたり、実行時にインポートすることができます。

Agilent VEE 関数の定義方法

VEEには、3種類のユーザ定義関数があります。それぞれの関数の概要は次のとおりです。

1　UserFunction

- UserFunctionを**作成する**には、**[Device]** ⇒ **[UserFunction]**を選択するか、または複数のオブジェクトを選択しておいて、**[Edit]** ⇒ **[Create UserFunction]**をクリックします。
- プログラム内のそれぞれ異なる場所からUserFunctionを**呼び出す**には、**Call myFunction**(**[Device]** ⇒ **[Call]**)オブジェクトを使用するか、たとえばFormulaにある式をオブジェクト内で使用します。UserFunctionのオブジェクト・メニューを使用し、**[Generate]** ⇒ **[Call]**などの選択肢を選択すると、メイン・プログラム内で、UserFunctionから、callオブジェクトを生成することもできます。
- UserFunctionを**編集する**には、**[Edit]** ⇒ **[Edit UserFunction . . .]**をクリックし、表示されるリスト・ボックスで、該当するUserFunctionを選択します。
- UserFunctionをあるプログラムから別のプログラムに**転送する**には、プログラム開発中にUserFunctionをマージするか、**[Device]** ⇒ **[Import Library]**でUserFunctionを実行時にインポートします。

2　コンパイル済み関数

- コンパイル済み関数を**作成する**には、VEE 以外のコンパイル済み言語を使って作業します。次に、その関数をDLLなどのライブラリに保管します。
- コンパイル済み関数をプログラムに**リンク**するには、Import Libraryオブジェクトを使用します。このオブジェクトは、実行時に関数ライブラリをVEEにリンクします。詳細は、第12章「Agilent VEEプログラムの最適化」を参照してください。

- コンパイル済み関数を**呼び出す**には、Call myFunctionオブジェクトを使用するか、VEEオブジェクト内で式を作成します。

3 リモート関数

- UserFunctionと類似していますが、ネットワークに接続されたリモート・ホスト・コンピュータで動作する点が異なります。

UserObjectとUserFunctionの違い

UserObjectについては、これまでの章ですでに作成して使用しました。VEEがUserObjectとUserFunctionを提供する理由は、2つは異なる特性を持っており、異なる目的で使用できるためです。UserObjectとUserFunctionの違いは次のとおりです。

[Device] ⇒ **[UserObject]**にあるUserObjectは、自分で定義するオブジェクトであり、VEEにあるほかのオブジェクトとまったく同じように使用できます。UserObjectは、サブプログラムと同じようにプログラミングしますが、画面にグラフィカルに表示されたままとなります。プログラム内のある場所で使用する必要がある場合は、そのクローンを作成してすべてのコピーを保持する必要があります。UserObjectのクローンをいくつも作成すると、プログラムが大きくなり、プログラムのロードに時間がかかるようになるので、注意してください。ある機能を1つのUserObjectに追加した場合、すべてのUserObjectを同じにしておきたければ、ほかのすべてのオブジェクトにも同じ機能を追加する必要があります。

[Device] ⇒ **[UserFunction]**にあるUserFunctionの場合、メモリにはサブルーチンが1コピーだけ存在します。表示する必要がある場合は、ワークスペース内のそれ自身のウィンドウにのみグラフィカルに表示されます。表示する必要がない場合は、保管されて、Callオブジェクトまたは他の式フィールドから呼び出されます。UserFunctionに加えた変更は、そのUserFunctionを呼び出すプログラムの中のすべてのインスタンスによって継承されます。コードをさらに再利用するため、UserFunctionのライブラリを作成することもできます。

例題9-1: UserFunctionによる演算

この例題では、ArrayStatsという名前のUserFunctionの作成方法について説明します。このUserFunctionは、配列を受け付け、その最大値、最小値、平均値、標準偏差を計算し、結果を出力ピンに出力します。

UserFunctionの作成方法

1 **[Device]** ⇒ **[Formula]**を選択し、デフォルトの入力ピンを削除し、デフォルトの式を`ramp(1024,1,1024)`に変更します。

これにより、1から1024までの値を1024個の要素として持つ配列が1つ作成されます。

2 **[Device]** ⇒ **[UserFunction]**を選択します。その名前を`ArrayStats`に変更します。

a 配列用のデータ入力端子を1つ追加します。

b 結果用のデータ出力端子を4つ追加します。

c 出力端子の名前を`Max`、`Min`、`Mean`、`Sdev`に変更します。[Function & Object Browser] ボックスの [Probability & Statistics] カテゴリで、[max]、[min]、[mean]、[sdev] を選択します。

d それらをArrayStats内に配置し、データ入力を**A**に接続し、データ出力を一致する出力端子に接続します。[ArrayStats]ウィンドウのサイズを小さくして、メインとArrayStatsの両方のウィンドウが見えるようにします。図181を参照してください。

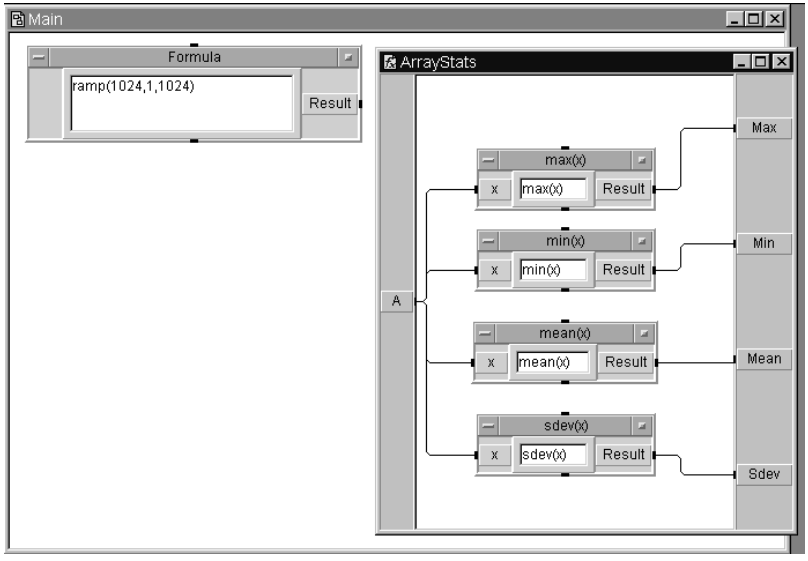


図181　　メイン・ウィンドウとArrayStatsウィンドウ

3 ArrayStatsをアイコン化します。ワークスペースの下部にアイコンとして表示されます。

4 **[Device]** ⇒ **[Call]**をクリックし、オブジェクト・メニューをオープンし、図182に示すように、[Select Function] をクリックします。次に**[OK]**をクリックします。VEEが、オブジェクトの名前を自動的に変更し、正しいピンを追加することに注目してください。

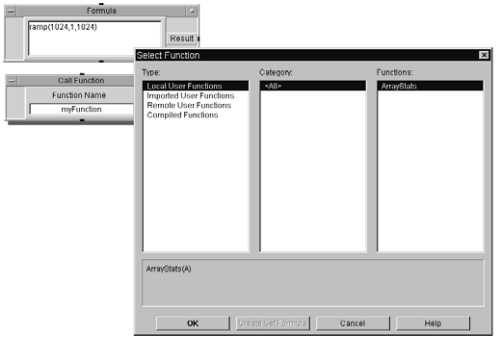


図182　Call myFunctionのためのピンの設定

5 Formulaの出力をCall ArrayStatsの入力に接続します。
[Display] ⇒ **[AlphaNumeric]**を選択し、3つのクローンを作成し、Call ArrayStatsの出力ピンに接続します。オブジェクトの名前を変更します。

6 プログラムを実行します。図183のように表示されます。プログラムを「**array_stats.vee**」のファイル名で保存します。

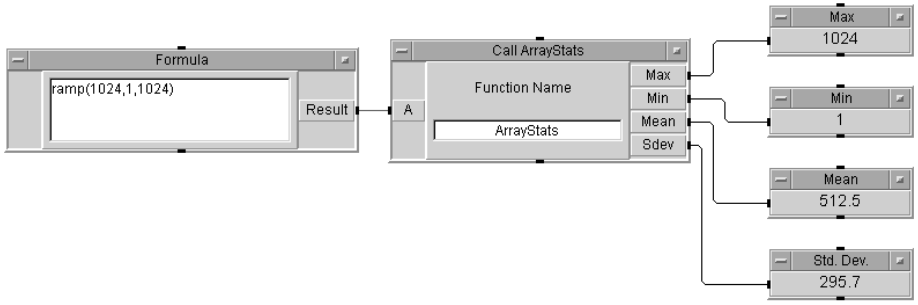


図183　　ユーザ関数ArrayStatsを呼び出す方法

ArrayStatsをプログラム内で使用するには、**[Device]** ⇒ **[Call]**をクリックし、オブジェクト・メニューから[Select Function]ボックスをオープンし、[ArrayStats]を選択します。VEEは、オブジェクトの名前を自動的に`Call ArrayStats`に変更し、必要な入出力端子を追加します。

UserFunctionのオブジェクト・メニューで、**[Generate]** ⇒ **[Call]** を選択し、Call ArrayStatsオブジェクトを表示します。この場合、絶対にUserFunctionをワークスペース全体に拡大しないでください。

UserFunctionの編集方法

この例題では、ArrayStatsを編集し、配列の統計情報を提供する4つのフィールドを持つレコードを送信します。

1. 4つのAlphaNumeric表示を削除します。
2. **[Edit]** ⇒ **[Edit UserFunction . . .]**を選択し、[Edit UserFunction]リスト・ボックスで[ArrayStats] を選択します。プログラム内のすべてのUserFunctionが表示されます。
3. ArrayStatsのオブジェクト・メニューをオープンし、**[size]**をクリックし、編集ウィンドウを拡大します。オブジェクトのサイズを変更する必要がある場合は、オブジェクトの4隅のいずれかをクリック・アンド・ドラッグします。
4. 出力端子に向かう4つの線を削除します。`Ctrl+Shift`キーを押したまま、削除する必要のある線をクリックします。

5 **[Data]** ⇒ **[Build Data]** ⇒ **[Record]**を選択し、[ArrayStats]ウィンドウの右側に配置します。

a データ入力端子を2つ追加します。

b 統計関数のラベルにならって、4つの端子にmax、min、mean、sdevというラベルを付けます。

c 4つのFormulaオブジェクトの出力をBuild Recordのそれぞれの入力に接続します。

d [Max]をダブルクリックし、新しい名前を入力し、**[OK]**をクリックして、Maxの出力端子Xの名前を変更します。

e ArrayStatsのほかのデータ出力端子を削除します。

f User Functionの編集ウィンドウで、Build Recordの出力を**X**出力端子に接続します。プログラムは図184のように表示されます。次に、編集ウィンドウのアイコン化ボタンをクリックします。

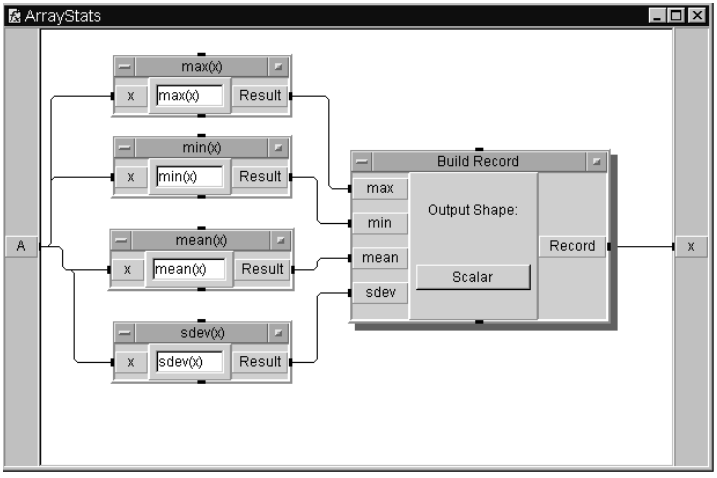


図184　UserFunction ArrayStatsの編集方法

6 Call ArrayStatsのオブジェクト・メニューをオープンし、**[Configure Pinout]**をクリックします。これにより、ピンの数が最新の編集と一致するように調整されます。

注 記

UserFunction内の入力または出力の数を変更したときは必ず、ピンの数を更新するために、オブジェクトをオープンし、**[Configure Pinout]**をクリックする必要があります。Callオブジェクトの入力ピンと出力ピンは手動でも更新できますが、**[Configure Pinout]**を使用するほうがずっと簡単です。[Find]を使用すると、UserFunctionを呼び出すCallオブジェクトおよび式をすべて検索することができます。詳細は、356ページの「大型プログラムで関数を検出する方法」を参照してください。

Record Constantオブジェクトを使ってレコードを1つ表示します。Default Valueのコントロール入力を使用して、ArrayStatsからのレコードを受け入れます。VEEは、Record Constantが受信レコードを保持するように自動的に設定します。

7 **[Data]** ⇒ **[Constant]** ⇒ **[Record]**を選択し、ほかの3つのオブジェクトの右側に配置します。

a Recordのオブジェクト・メニューをオープンし、**[Add Terminal]** ⇒ **[Control Input . . .]**をクリックします。表示されるリスト・ボックスで[Default Value]を選択します。必要であれば、[Properties]メニューをオープンすると、Show Terminalsを表示できます。

b Call Functionのデータ出力をRecordオブジェクトのコントロール入力ピンに接続します。データ・ラインと区別するために、コントロール・ラインが破線で表されています。

8 プログラムを実行します。図185のように表示されます。

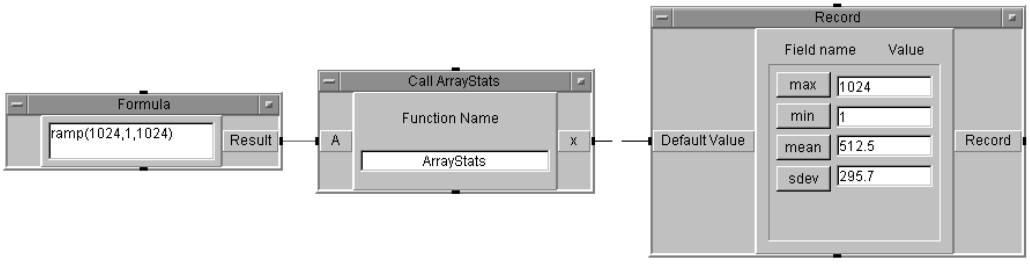


図185　ArrayStatsの出力をRecordに編集した後

式からUserFunctionを呼び出す方法

この例題では、**Formula**オブジェクト内の式からArrayStatsを呼び出す方法を習得します。

1 **[Device]** ⇒ **[Formula]**を選択し、デフォルトの公式を**ArrayStats(A)**に置換します。Call ArrayStatsのオブジェクト・メニューで、**[Replace]**をクリックします。

VEE画面下部のステータス・バーに、置換オブジェクトを選択するように表示されます。ArrayStats関数を呼び出すFormulaオブジェクトをクリックします。VEEは自動的に、Call ArrayStatsオブジェクトを新しいFormulaオブジェクトに置換し、データ・ラインの配線を保持します。

Formulaオブジェクトは、端子**A**の入力を受け取り、UserFunction ArrayStatsに送信します。ArrayStatsは、統計情報のレコードを端子**X**に送信します。**UserFunction (X)**からの最初の出力値は、Formulaオブジェクトに返され、Result出力に送信されます。

2 プログラムを実行します。図186のように表示されます。

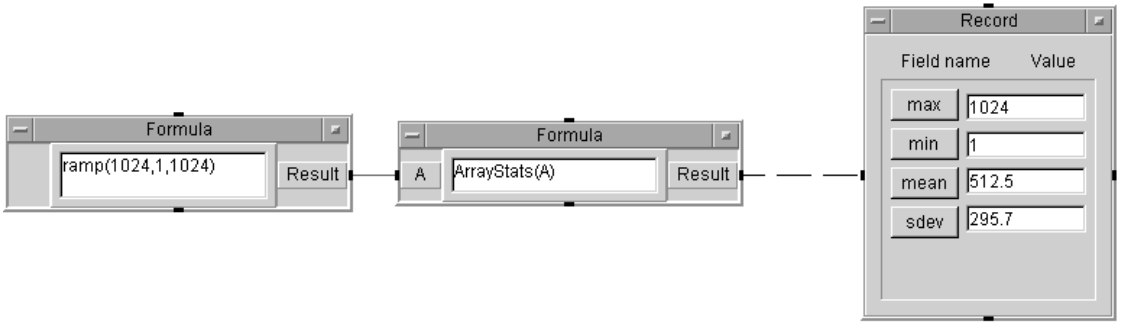


図186　ユーザ関数ArrayStatsを呼び出す方法

FormulaオブジェクトにあるArrayStatsの機能は、それがCall ArrayStatsオブジェクトにあったときの機能とまったく同じです。この例ではFormulaオブジェクトを使用しますが、ArrayStatsは、式を受け付ける入力フィールドであれば、To Fileオブジェクトなど、**どの**入力フィールドからでも呼び出すことができます。

注 記　式からUserFunctionを呼び出した場合、UserFunctionは単一の出力(一番上のデータ出力ピン)のみを送信します。すべての出力を必要とする場合、またはそれらの出力をRecordに保管できない場合は、Call Functionオブジェクトを使用します。

注 記　式からUserFunctionを呼び出した場合、入力端子は、その関数に渡る関数パラメータとして使用されます。データをその関数に渡さない場合でも、関数名の後に空の丸かっこ()を含める必要があります。そうしなければ、VEEは、Global変数または入力端子が参照されていると仮定します。たとえば、MyFunctionという名前のUserFunctionが入力パラメータを持っていなくても、式の中でMyFunction()と書く必要があります。Callオブジェクトの場合、VEEは関数が参照されていることを認識できるので、丸かっこ()を必要としません。

UserFunction呼び出しを生成する方法

UserFunctionから呼び出しオブジェクトを生成し、メイン・プログラムに配置するには、UserFunctionのオブジェクト・メニューのGenerateメニューを使用します。Generateメニューには、UserFunctionを呼び出す一般的なオブジェクトがほとんどそろっています。呼び出す側のオブジェクトを選択した場合は、メイン・プログラムなどの呼び出す側のウィンドウに配置して、正しい名前とピンで適切に設定することができます。

この例題では、ArrayStats UserFunctionから、メイン・プログラム内にArrayStatsオブジェクトを生成する方法を習得します。

1 図186で使用している例で、Formulaオブジェクトの[ArrayStats]をダブルクリックしてこのオブジェクトを削除します。オブジェクト・メニューを選択し、[Cut]を選択することもできます。

2 UserFunction ArrayStatsで、オブジェクト・メニューを選択し、**[Generate]** ⇒ **[Formula Call]**を選択します。図187は、UserFunctionのオブジェクト・メニュー内の[Generate]メニューを示します。

図187　UserFunctionのGenerateメニュー

3 オブジェクトをメイン内に配置します。VEEは、自動的に新しいオブジェクトに**ArrayStats(A)**という名前を付け、UserFunction ArrayStatsを呼び出す式**ArrayStats(A)**を含めることに注目してください。

4 Formulaオブジェクトからの出力を**ArrayStats(A)**に接続し、**ArayStats(A)**からの出力をRecordに接続します。

5 プログラムを実行します。図188のように表示されます。

UserFunctionのオブジェクト・メニューをオープンし、[Generate]メニューを選択して、UserFunctionを呼び出すためにプログラム内に配置できるほかのオブジェクトを検討します。そうしたオブジェクトにはCall、(この例で使用する)Formula Call、If/Then/Else Call、ShowPanel、HidePanelなどがあります。

図188　UserFunctionから呼び出しオブジェクトArrayStats(A)を生成する方法

UserFunctionおよびProgram Explorer

UserFunctionとUserObjectによって、VEEプログラムがよりモジュール化され、理解しやすくなります。Program Explorerは、複雑なプログラム内を移動するための便利なツールです。たとえば、図189のプログラムの階層が、Program Explorerに表示されます。Program Explorerを表示するには、**[View]** ⇒ **[Program Explorer]**をクリックするか、ツールバー上のProgram Explorerアイコンをクリックします。

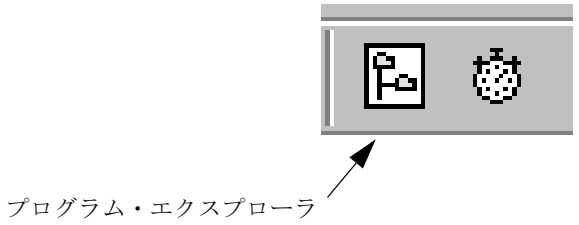


図189　ツールバー上のProgram Explorerアイコン

図190に、使用中のProgram Explorerを示します。

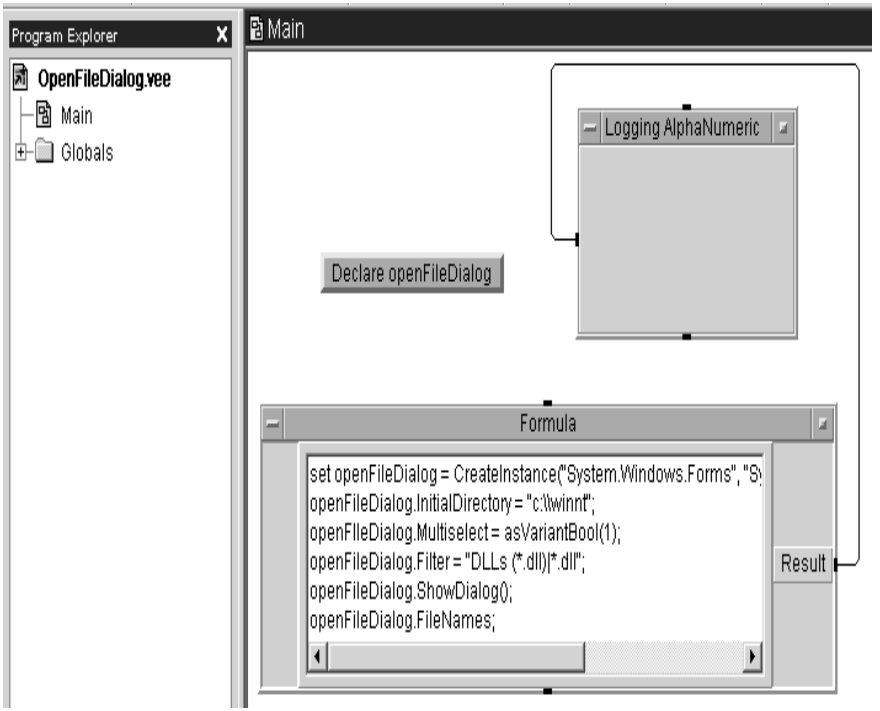


図190　UserFunctionによるProgram Explorerの使用方法

Agilent VEE UserFunctionによるライブラリの使用方法

既存のVEEテスト・プログラムを流用するには、UserFunctionを再使用できます。プログラムをセーブすると、UserFunctionも自動的にセーブされます。UserFunctionは、VEEプログラムまたは論理的に関連付けられたUserFunctionのライブラリを保持できます。

既存のUserFunctionを新しいプログラムに配置する方法は、2通りあります。

[File] ⇒ **[Merge Library . . .]**コマンドを使用して、元のUserFunctionのコピーを現在のプログラムに配置します。プログラムには、各UserFunctionの個別のコピーが保持されます。これらのマージされたUserFunctionは編集可能であり、UserFunctionを修正したいときには**[File]** ⇒ **[Merge Library . . .]**コマンドを使用します。

-または-

[Device] ⇒ **Import Library**オブジェクトを使用して元のUserFunctionにアクセスします。オブジェクトは、コピーを作成せずに、別のファイルにある元の関数にアクセスします。これらのUserFunctionは、実行時にインポートされます。このため、ロード時間が分散され、ディスク容量が一定に保たれ、メモリの節約にもなります。インポートされたUserFunctionをデバッグ目的などのために表示することはできますが、編集することはできません。かわりに、元のファイルを編集できます。**[Device]** ⇒ **Delete Library**オブジェクトを使い、インポートされたUserFunctionをプログラムで削除することもできます。

したがって、修正用に新しい関数コピーを必要とするときやスタンドアローン・プログラムを必要とするときには、UserFunctionを**マージ**します。関数に対して単一のソースを必要とするときやスペースを節約したいときには、UserFunctionを**インポート**します。

例題9-2: UserFunctionのライブラリの作成方法およびマージ方法

この例題では、UserFunctionのVEEライブラリを含むレポート生成プログラムを作成します。次に、UserFunctionのライブラリをマージする新しいプログラムを作成します。

UserFunctionのライブラリの作成方法

1 UserFunction の詳細をプログラミングしないで、トップ・レベルのプログラムを作成します。

a 4個のUserFunctionを作成します。1個の出力ピンを持つBuildRecAry、ReportHeader、1個の入力ピンを持つReportBody、およびReportDisplayです。すべてのUserFunctionをアイコン化します。

b メインで、図191に示すように設定され、接続された4個の**[Device]** ⇒ **Call**オブジェクトを作成します。プログラムを**Report.vee**としてセーブします。

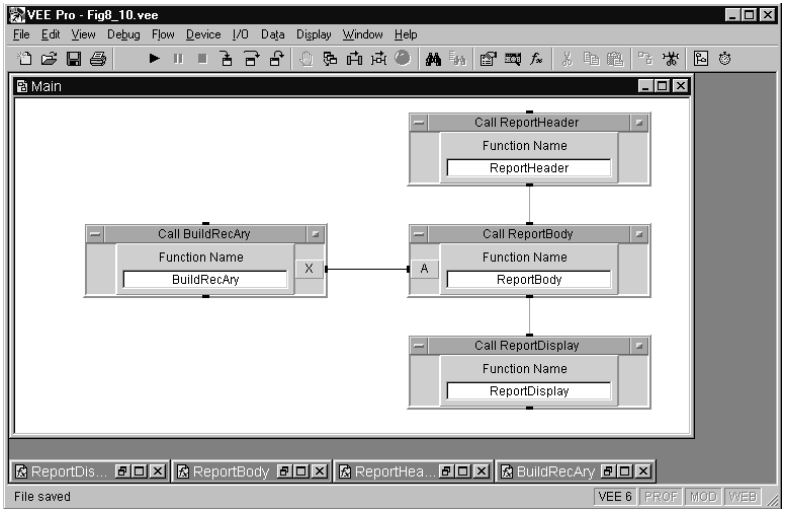


図191　　トップ・レベルからのReport.vee

注 記

Callオブジェクトは、UserFunctionを参照するとき、括弧を要求しません。関数をFormulaオブジェクトから呼び出している場合は、関数がパラメータを使用しているかどうかに関係なく、括弧を含める必要があります。

4個のUserFunctionは1つのライブラリです。**[Edit]** ⇒ **[Edit UserFunction . . .]**をクリックすると、それらをリストとして表示できます。**[Cancel]**をクリックして、リスト・ボックスをクローズします。

2 4個のUserFunctionを次の図に示すようにプログラムします。

図192に、**BuildRecAry** UserFunctionを示します。BuildRecAryには、「**A<=5**」であるかをテストする3項式を持つ**Formula**オブジェクトが含まれます。式が**TRUE**であると評価された場合、オブジェクトが「**"PASS"**」を出力します。それ以外の場合、オブジェクトは「**"FAIL"**」を出力します。式には括弧が必要です。

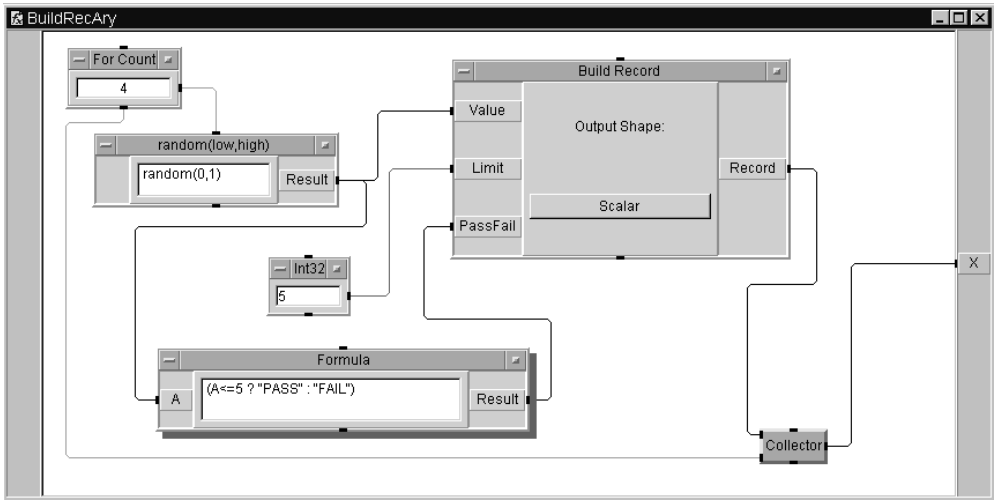


図192　BuildRecAry UserFunction

図193に、**ReportHeader UserFunction**を示します。

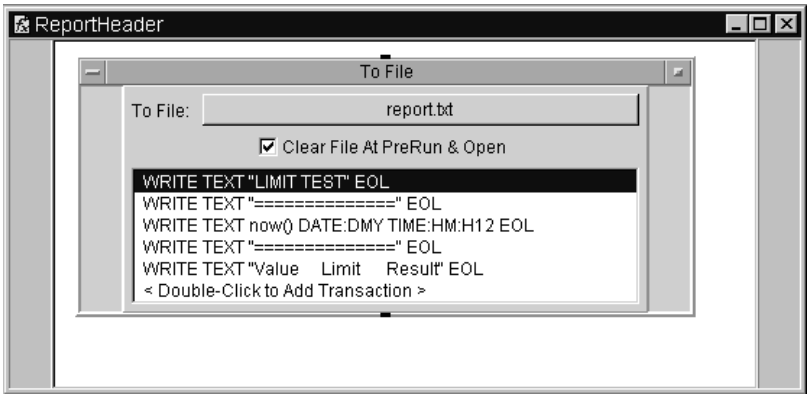


図193　ReportHeader UserFunction

図194に、**ReportBody** UserFunctionを示します。レコードの配列**A[B]**に注意してください。**値B**が0から、1、2へ変化するので、レコードの**Value**、**Limit**、**PassFail**などの特定フィールドに、<**レコード**>.<**フィールド**>表記を使ってアクセスできます。For Count出力は、ゼロから始まります。また、最初のトランザクションのEOLはオフになっています。

引用符で囲まれた垂直バー **"|"**は、垂直バーに対する一定文字列文字を表します。**FW:5**は、文字列フィールドの幅が5であることを表します。**RJ**は、right justified(右揃え)を表します。

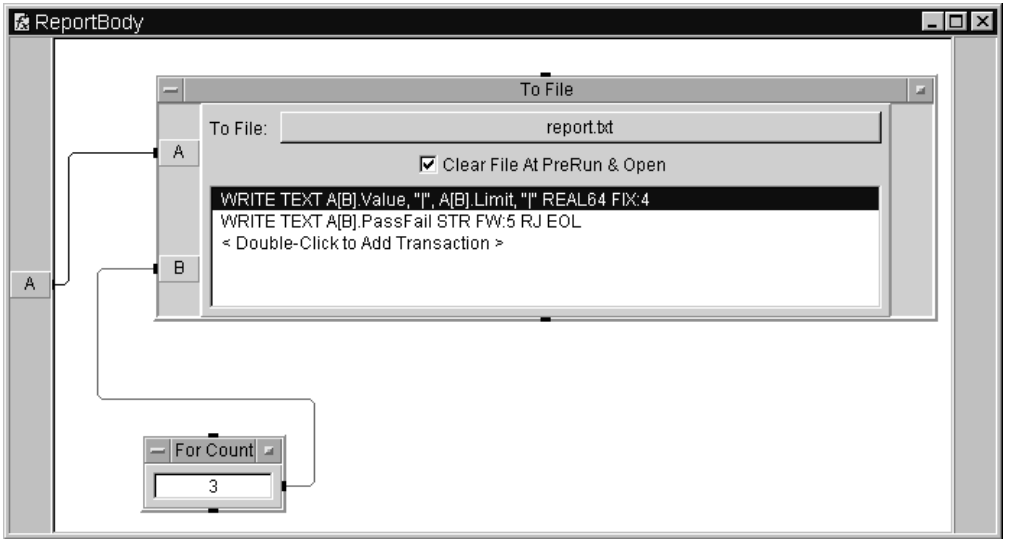


図194　ReportBody UserFunction

図195に、ReportDisplay UserFunctionを詳細ビューで示します。UserFunctionはファイルの最後まで文字列配列を読み取ります。これは、**STR ARRAY**フォーマットの後のアスタリスク記号(*)により指定されています。

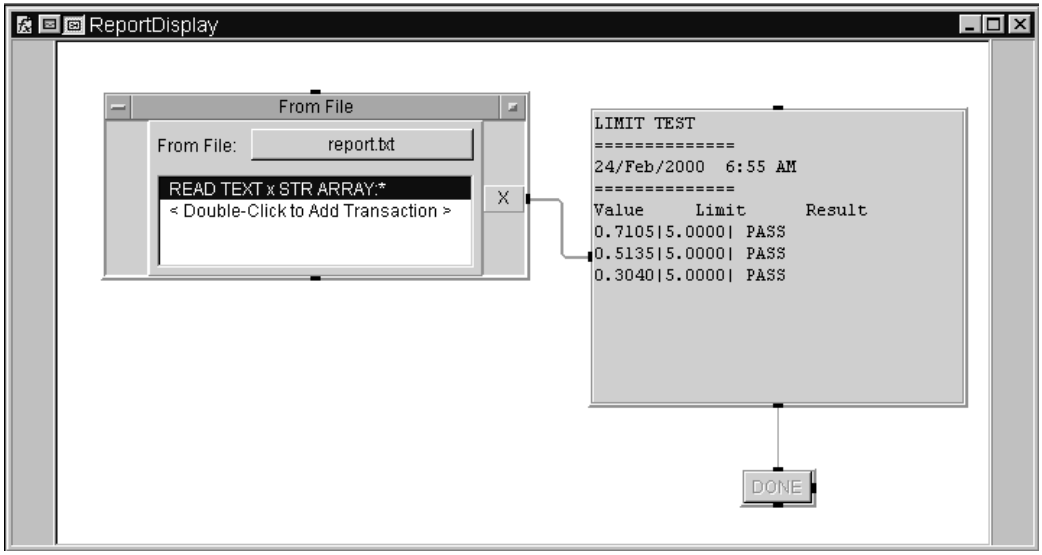


図195　ReportDisplay詳細ビュー

図196は、[Properties]ボックスで[Show Panel on Execute]を選択したときの、ReportDisplay UserFunctionのパネル・ビューを示したものです。[Properties]ボックスのポップアップ・パネル・タイトルもReportDisplayに変更されています。パネルを作成するには、Confirm (OK)およびLogging AlphaNumericオブジェクトを選択し、**[Edit]** ⇒ **[Add to Panel]**をクリックします。Logging AlphaNumeric表示で、[Show Title Bar]の選択が解除されています。また、Confirm (OK)ボタンの名前がDONEに変化しています。Confirm (OK)ボタンを含めたのは、ユーザが見終わるまで画面に表示されているようにするためです。

3 プログラムを実行します。ReportDisplayパネルがポップアップされ、図196に示すように値が表示されます。**[DONE]**をクリックして実行を終了します。次に、プログラムを**Report.vee**としてセーブします。

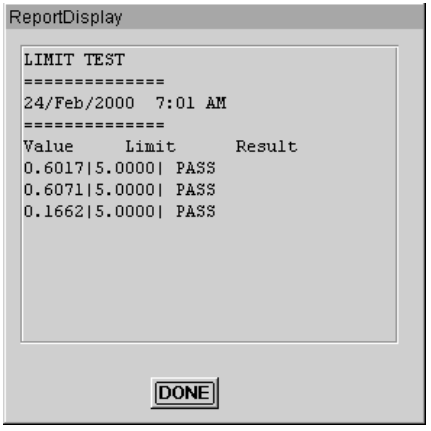


図196　ReportDisplayパネル・ビュー

別のプログラムを作成し、ライブラリでマージする方法

この例題では、新しいプログラムを作成し、ライブラリをマージします。ここでは、レポート生成用の関数のライブラリを構築します。新しいプログラムには、ライブラリの各関数について説明するNote Padオブジェクトを含めます。このプログラムにRepGenという名前を付けます。

RepGenを再使用するには、新しいレポート生成ユーザ関数を作成し、それをプログラムとマージし、最新情報に変えるためNote Padオブジェクトを更新します。[Merge Library...]コマンドを使用して、RepGenからすべての関数を流用することができます。

1　**[File]** ⇒ **[New]**を選択します。

2　**[File]** ⇒ **[Merge Library . . .]**を選択します。[Merge Library]リスト・ボックスから**Report.vee**を選択します。別のディレクトリにいる場合、完全なファイル・パスを入力します。

[Edit] ⇒ **[Edit UserFunction]**を選択するか、またはProgram Explorerを表示して、**Report.vee**からのライブラリが新しいプログラムに転送されたことを確認します。[Merge Library...]コマンドを使用すると、マージされた関数をローカル関数と同様に編集できます。

3 **[Display]** ⇒ **[Note Pad]**を選択し、図197に示す説明と同様のUserFunctionの説明を入力します。次に、プログラムを**RepGen.vee**としてセーブします。

注 記　関数を呼び出す実際のVEEプログラムがない場合でも、ライブラリを作成する目的で、UserFunctionのプログラムをセーブできます。

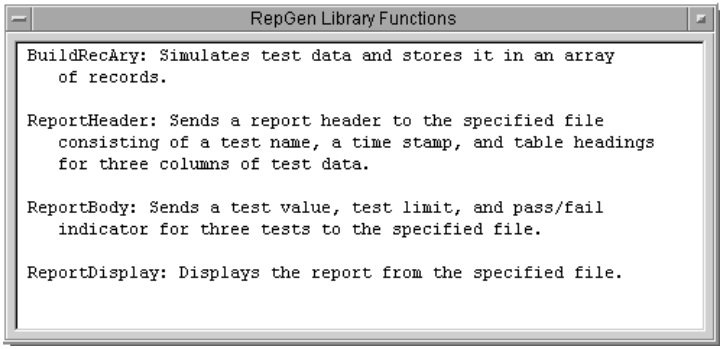


図197　UserFunctionのRepGen.veeライブラリ

例題9-3: ライブラリのインポート方法および削除方法

作成したUserFunctionのライブラリを、どのプログラムにもマージしたいわけではありません。実行時にライブラリを呼んで、いくつかの関数を使用した後、ライブラリを削除した方がメモリの節約になり、好ましい場合があります。Import LibraryオブジェクトとDelete Libraryオブジェクトは、こうした状況に備えて用意されています。

この例題では、RepGenプログラムから関数をインポートします。次に、BuildRecAry関数を呼び出し、テスト・データをシミュレートし、表示してから、最後にライブラリを削除して、メモリとスワップ領域を解放します。

1 **[File]** ⇒ **[New]**を選択します。

2 **[Device]** ⇒ **[Import Library]**を選択し、それをメインに配置します。Import Libraryオブジェクトのフィールドを次のように設定します。

表38　Import Libraryフィールド

Import Library フィールド	説明
[Library Type]	[Library Type]フィールドのメニューから、UserFunction、Compiled Function、またはRemote Functionを選択できます。この場合、UserFunctionライブラリを希望するので、デフォルトのままにします。
[Library Name]	[Library Name]には、**myLib**がデフォルトとして表示されます。この名前は、インポート対象となるさまざまなライブラリを区別するため、VEEプログラムによって「ハンドル」として使用されます。Delete Libraryオブジェクトは、この名前を使用して削除するライブラリを識別します。デフォルト名を使用できます。
[File Name]	[File Name]フィールドは、デフォルトで、ユーザ・プログラム・ディレクトリのダイアログ・ボックスを表示します。関数のライブラリを保持するファイルを指定します。 デフォルト名**myFile**をクリックして、リスト・ボックスを表示します。**RepGen.vee**を選択します(344ページの「例題9-2: UserFunctionのライブラリの作成方法およびマージ方法」で作成)。このファイルは、インストール中に、プログラム用に指定したディレクトリにあります。

3 オブジェクト・メニューをオープンし、[Load Lib]を選択して、実行時まで待つ代わりにライブラリを即座にインポートします。このコマンドは、開発段階で非常に有効です。次のステップで、Callオブジェクトで[Select Function]を選択すると、関数が最初にmyLib.BuildRecAryなどのライブラリ・ハンドルで指名されます。

インポートされた関数のライブラリを表示するには、Program Explorerを使用します。

注 記

ライブラリをインポートしたため、UserFunctionに対しては、表示やブレークポイントの設定しか行えません。UserFunctionを編集することはできません。編集可能な方法でUserFunctionをプログラムに追加するには、[Merge Library...]コマンドを使用します。

4 **[Device]** ⇒ **[Call]**を選択し、Import Libraryオブジェクトの下に配置します。Import Libraryからの出力シーケンス・ピンをCallオブジェクトの入力シーケンス・ピンに接続します。

5 Call Functionのオブジェクト・メニューをオープンし、[Select Function]をクリックして、Load Libコマンドでインポートされた関数のリストを表示します。図198に示すように、**myLib.BuildRecAry**を選択します。

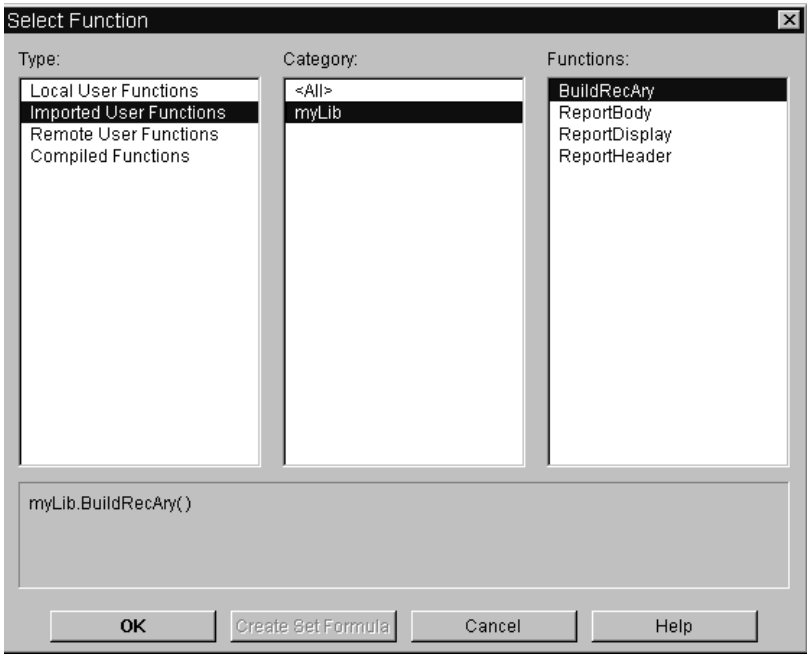


図198　　インポートされたライブラリから関数を選択する方法

VEEは、[Function Name]フィールドに関数を自動的に挿入し、必要な出力端子を追加します。[Function Name]フィールドにmyLib.BuildRecAryを入力しても、同じ結果を実現できます。ライブラリ内の関数の名前をリストする必要があるときには、[Select Function]を使用します。

6 AlphaNumeric表示を選択し、表示を拡大し、それをCallデータ出力に接続します。

7 **[Device]** ⇒ **[Delete Library]**を選択し、**Call**オブジェクトの下に配置します。シーケンス・ピンを接続して、BuildRecAry関数が呼び出された後に、ライブラリが削除されるようにします。Import Libraryオブジェクトで使用した名前と同じであるため、デフォルトのLibrary Nameをそのまま使用することができます。

8 プログラムを実行します。図199のように表示されます。プログラムを**libraries.vee**としてセーブします。

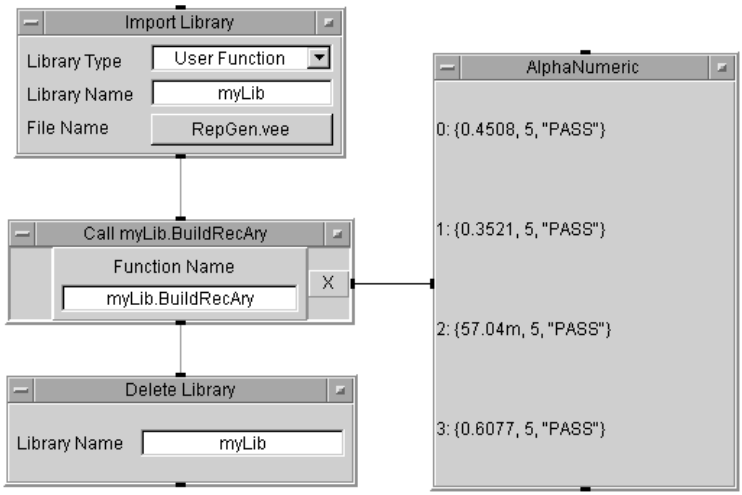


図199　　ライブラリから関数を呼び出す方法

マージされた関数とインポートされた関数の名前に関する注意事項を示します。

- マージされた関数とローカル関数の名前が同じである場合、VEEはエラーを表示します。
- インポートされた関数とローカル関数の名前が同じである場合、同じ名前が許可されますが、関数名だけを使用するとローカル関数が呼び出されます。図199のCallオブジェクト内にあるようなMyLib_func()構文を使って、インポートされた関数を明示的に呼び出すことができます。

- インポートされた2つのライブラリが同じ関数名を持つ場合、結果は不確定です。Callオブジェクトは Library名myLib.BuildRecAryを使用するので、混乱はありません。同じ名前を持つ別のローカル関数または他のインポートされた関数があったとしても、BuildRecAryの名前と場所が指定されます。

大型プログラムで関数を検出する方法

VEEの[Edit]メニューには、大型プログラムでオブジェクトとテキストの位置を見つける際に役立つ[Find]機能があります。たとえば、**Examples\Games**ディレクトリの**Solitaire.vee**プログラムをオープンします。詳細ビューに進み、**[Edit]** ⇒ **[Find . . .]**をクリックして、図200に示すダイアログ・ボックスを表示します。

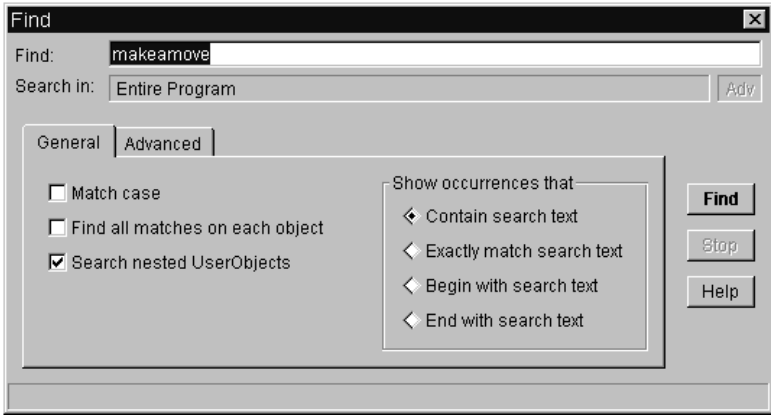


図200　[Find]ダイアログ・ボックス

図201に示すように、`makeamove`(**Solitaire.vee**プログラム内のUserFunction)を入力し、**[Find]**をクリックします。VEEがmakeamoveというUserFunctionの場所を自動的に検出し、図201に示すように、呼び出し元のプログラムの一部を表示します。

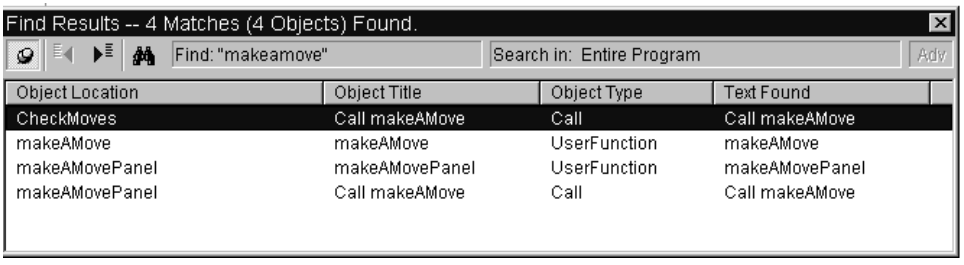


図201　[Find Results]ダイアログ・ボックス

[Find]を使用すると、変数、タイトル、オブジェクトに関する設定など、任意のオブジェクトやテキストの場所を検出できます。[Find Results]ボックスの任意のラインをダブルクリックして、オブジェクトの場所を検出します。

注 記

[Find]は、Program Explorerのオブジェクトにマウスを置き、右ボタンをクリックすることによっても使用できます。この場合、検索範囲が特定のUserFunctionまたはUserObjectに制限されます。

Agilent VEEプログラムのマージ方法

既存のプログラムを流用する一番簡単な方法は、過去のプログラムを現在のテストとマージすることです。プログラムを再使用するには、プログラムをマージし、現在のニーズに合うよう編集します。

[File] ⇒ **[Merge . . .]**コマンドは、作業領域の既存の内容を保持しながら、プログラムまたはセーブされたオブジェクト・セットの内容を作業領域に追加します。デフォルトで、**[File]** ⇒ **[Merge . . .]**は、VEEに付属しているプログラムのディレクトリを表示します。このディレクトリには、棒グラフ表示、(ID番号など)ユーザ入力用のデータ入力キーパッドなど、一般に必要とされるプログラムが格納されています。

例題9-4: 棒グラフ表示プログラムのマージ方法

この例題では、既存のプログラムを新しいプログラムとマージします。例ではVEE Libディレクトリからのプログラムを使用しますが、任意のプログラムを使用できます。ramp()関数を使用し、1～5の5個の値を持つ配列を作成します。内部VEE表示のいずれかで配列を表示する代わりに、BarChartプログラムを作成中のプログラムとマージします。

1. Formulaを選択し、左の作業領域に配置します。
2. データ入力端子を削除します。
3. デフォルトの公式をramp(5,1,5)に変更します。

   最初のパラメータは、ramp配列で必要な要素の数です。2番目のパラメータは開始番号で、3番目のパラメータは最後の番号です。この関数の詳細を知るには、Formulaオブジェクト・メニューのHelpを選択します。オブジェクト・メニューには現在ramp呼び出しがあります。または、**[Help]** ⇒ **[Contents & Index]**を試してから、[Index]フォルダで[Search]機能を使用します。
4. **[File]** ⇒ **[Merge . . .]**をクリックし、[Merge File]リスト・ボックスを取得します。**BarChart.vee**を選択し、それをFormulaオブジェクトの右に配置します。2つのオブジェクトを接続します。
5. プログラムを実行します。図202のように表示されます。

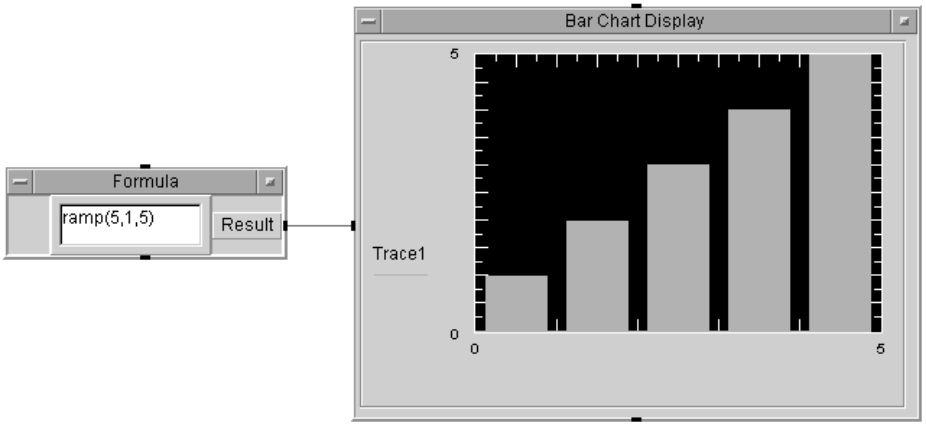


図202　BarChartプログラムのマージ方法

棒グラフ表示は1次元配列をとり、値を縦棒として表示します。配列内の値を表示するために必要な棒の数を使用します。プログラムがどのように作成されているかを表示するには、表示の詳細ビューをオープンします。ライブラリ・ディレクトリの例を見ることにより、プログラムに対する理解が深まります。

注 記

[File] ⇒ **[Merge]**コマンドを使用して、UserObjectとオブジェクトをマージします。**[File]** ⇒ **[Merge Library]**コマンドを使用して、UserFunction同士をマージします。

この章の復習

この章では、次のことをできるようになりました。次の章に進む前に、必要に応じてトピックを復習してください。

- UserFunction を定義し、それをコンパイル済み関数およびリモート関数と比較する。
- UserFunctionを作成し、呼び出し、編集する。
- UserFunctionライブラリを作成、マージ、インポート、削除する。
- ゲーム・プログラムの1つでFind機能を使用する。
- VEEプログラム全体を現在のプログラムとマージする。

10
テストのシーケンスを決定する方法

テストのシーケンスを決定する方法

この章の内容

- Sequencerオブジェクト
- Sequencer用のテストを設定する方法
- 実行時の結果に基づくテスト実行順序を作成する方法
- Sequencerがログとして記録したデータにアクセスする方法
- Sequencerテストとデータの受渡しを行う方法
- Sequencerがログとして記録したデータに対して解析を行う方法
- Sequencerのテスト・データを保管する方法

平均所要時間: 2時間

概要

この章では、Sequencerオブジェクトの使用方法の基本を習得します。Sequencerオブジェクトは一連のシーケンス・トランザクションを実行でき、それぞれのシーケンス・トランザクションは、UserFunction、コンパイル済み関数、またはリモート関数を呼び出すことができます。通常、Sequencerは一連のテストを実行するために使用されます。

Sequencerを使用する利点の一部は次のとおりです。

- テスト計画の開発が容易
- テスト間の分岐機能が多数ある
- カスタマイズされたテスト用実行可能ファイルを構築するための主要コンポーネントであること
- VEEとその他の言語で書かれたテストを呼び出せること
- テスト結果の自動ログ記録が可能なこと

注 記　**Sequencerは、VEEの最も強力な機能の1つです。Sequencerについての詳細は、オンライン・ヘルプを参照してください。**

最初の例題では、Sequencerオブジェクトのテストを設定する方法、テスト実行フローにテストを追加または挿入したりそのフローからテストを削除する方法、およびSequencerによってログに記録されたテスト・データにアクセスする方法を示します。ここでは、random()関数を使ってテスト結果をシミュレートします。

2番目の例題では、テストに渡すデータをグローバル変数を使って作成する方法、SequencerからUserFunctionを呼び出す方法、およびSequencerデータのログをファイルに記録する方法を習得します。最後に、データの各部分を解析します。

注 記　**一連のテストによって更新される例題11-5: ステータス・パネルを使用する場合は、434ページの「例題11-5: ステータス・パネルを作成する方法」を参照してください。**

注 記

この章の例題のほかに、Sequencerの使用方法についての演習が、付録Ａの541ページ「テスト・シーケンスの作成方法」にあります。

Sequencerオブジェクトの使用方法

Sequencerオブジェクトは、複数のテストを実行時の結果に基づいて指定された順序で実行します。テストとして有効なものは、VEE UserFunction、コンパイル済み関数、リモート関数、またはそのほかの単一の結果を返す式です。テストの結果はテスト仕様と比較され、テストが成功(PASS)したかどうかが判定されます。Sequencerは、次に、成功/失敗インジケータを使用して、次のテストを実行する必要があるかを判定します。

次のテストに分岐する場合は、6つの異なる選択肢があります。これらの選択肢は、「次のテストを実行する」、「同じテストを繰返す」、「より以前のテストにジャンプする」などです。例題10-1で、分岐の選択肢について詳細に説明しています。Sequencerは、必要な操作を決定するために、ユーザ入力を要求することさえできます。

指定されたテストの実行が終わると、Sequencerは、自動的にテスト・データのログを出力端子に記録します。これ以降は、データの解析や表示、または将来の調査に備えてのデータのファイルへの保管ができるようになります。

テスト実行順序の作成方法

この例題では、random()関数を使ってテスト結果をシミュレートし、テスト実行順序を確立し、その順序を変更する方法を習得し、ログとして記録された結果から特定のデータを読み取ります。

例題10-1: UserObjectの作成方法

注 記　この例では、テスト結果の一定の期待範囲を使用して、random()関数を実装する方法について説明しますが、原則はどのようなテストでも同じです。

1. **[Device]** ⇒ **[Sequencer]**を選択し、上部左の作業領域内に配置します。
2. **[Display]** ⇒ **[AlphaNumeric]**表示を選択し、Sequencerの下に配置し、幅を広げ、Sequencer Log出力端子をAlphanumericデータ入力に接続します。
3. Sequencerトランザクション・バーをダブルクリックし、[Sequence Transaction]ダイアログ・ボックスを表示します。フィールドを次のように設定します。

注 記　フィールドを編集する場合、フィールドを変更するには新しいフィールドをクリックし、別のフィールドに前進するには`Tab`キーを使用します。カーソルを後退させるには`Shift+Tab`キーを使用します。ダイアログ・ボックスの編集が終わったときにのみ`Enter`キーを押します。

表39　Sequencerトランザクションのフィールド

フィールド名	説明
TEST:	デフォルト名はtest1で、このデフォルト名を使用できます。テスト名はSequencerにおけるテストの単なるラベルです。テスト名はテスト関数そのものではありません。

表39　Sequencerトランザクションのフィールド

フィールド名	説明
SPEC NOMINAL:	期待されるテスト値を表します。デフォルトは**.5**です。この値を`.25`に変更し、次に、右端にある**[RANGE]**フィールドの上限値を**[1]**から`[.5]`に変更します。
FUNCTION:	デフォルトのエントリ**[testFunc(a)]**は、テストを実行する実際の関数を保持しています。この場合、このデフォルトのフィールドを[random()]関数に置き換えます。Random()は、テスト結果をシミュレートする**0**から**1**までのReal64値を返します。この結果がテスト仕様と比較されます。 random(low,high)オブジェクトは、[Function & Object Browser]ボックスの[Probability & Statistics]カテゴリにあります。この算術関数は、実際にFormulaオブジェクトを使用しなくても、式フィールドから呼び出すことができます。この例のように、パラメータlowとhighを指定しなかった場合は、デフォルトのパラメータ**0**と**1**が使用されます。

その他のフィールドはデフォルト値のままでかまいません。この設定で実行すると、約半分の実行結果が**成功(PASS)**になります。ダイアログ・ボックスは図203のように表示されます。

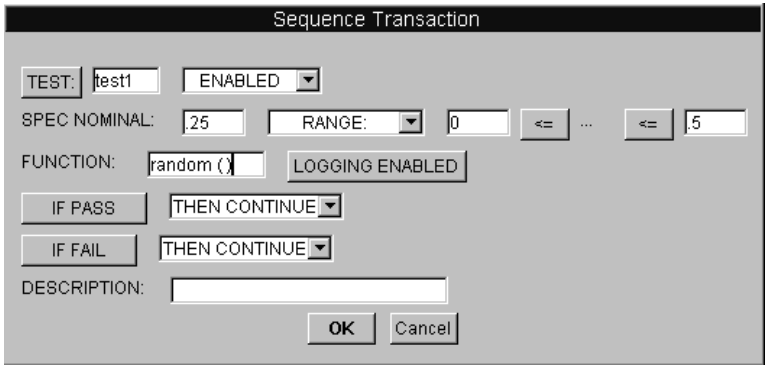


図203　**[Sequence Transaction]**ダイアログ・ボックス

[OK]をクリックして、ダイアログ・ボックスをクローズします。次のトランザクションがトランザクション・バーに表示されます。
test1 0 <= (.25) <= .5これは、戻り値が**0**以上**.5**以下である場合、test1は成功(PASS)することを意味します。期待される結果は約**.25**です。

4 プログラムを実行します。プログラムは、図204に示すように、表示オブジェクトにテスト名、テスト結果、成功/失敗インジケータ(**成功(PASS)**の場合は**1**、**失敗(FAIL)**の場合は**0**)を表示します。

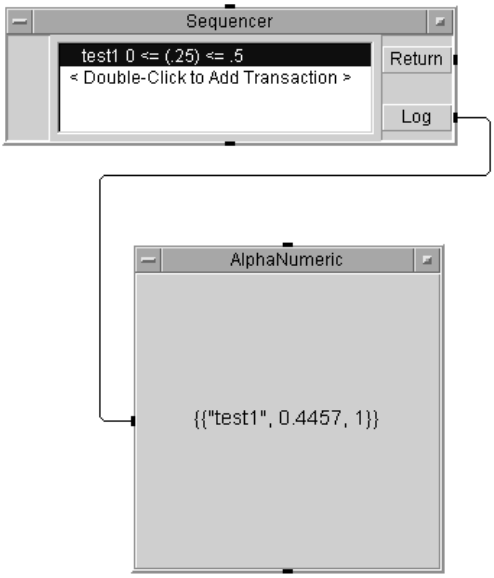


図204　　テストの設定方法

先に進む前に、[Sequence Transaction]ダイアログ・ボックス内のさまざまな選択肢を理解するため、表203を学習してください。トランザクション・バーをダブルクリックして、このダイアログ・ボックスをもう一度オープンします。さまざまなメニューをオープンし、それぞれ異なる選択肢について学んでください。

表40　[Sequence Transaction]ダイアログ・ボックス

Sequence Transaction フィールド	説明
TEST:	Sequencerでテストを参照するために使用する一意な名前。デフォルトの名前は、test1から始まり、テストごとに数字の部分が増分します。**TEST:**を選択すると、テスト結果がテスト仕様と比較され、設定に基づいて次のテストへの分岐が行われます。 **[TEST:]**ボタンは**[EXEC:]**にトグルします。**[TEST:]**を**[EXEC:]**にトグルした場合は、テスト結果とテスト仕様の比較なしで、テストが実行されます。たとえば、UserFunctionでグローバル変数をセットアップしている場合は、**[EXEC:]**を選択できます。**[EXEC:]**を選択すると、テストについてのログ記録もオフになります。
ENABLED	テストをいつ実行するかを決めます。このメニューでは4つの選択肢が表示されます。 • **ENABLED**は、条件にかかわりなくテストを実行します。 • **ENABLED IF:**を選択すると、記述されている式が**TRUE**と評価された場合にテストを実行します。たとえば、入力ピン**A**が値**1**を保持する**(A == 1)**場合にテストを実行できます。監査テストの制御のために**ENABLED IF:**を使用できます。たとえば、10回の実行ごとに、特定のテストを1回実行することもできます。 • **DISABLED**は**ENABLED**の反対です。 • **DISABLED IF:**は**ENABLED IF:**の反対です。
SPEC NOMINAL:	テストの期待値。
RANGE:	テスト値の範囲を指定します。このメニューでは4つの選択肢が表示されます。 • **RANGE**は、**成功(PASS)**条件であるテスト値の範囲を意味します。次の通常の比較演算子からも選択できます。 >、>=、<、<=、==、!=。 • **LIMIT**は、テスト・データの比較のめに1つの値だけを使用します。 • **TOLERANCE**は、**SPEC NOMINAL**値に対して上下に許される誤差の値を指定することにより、許容範囲を指定します。 • **%TOLERANCE**は、**SPEC NOMINAL**値に対して上下に許される公称仕様パーセンテージ誤差を指定することにより、許容範囲を指定します。

表40　[Sequence Transaction]ダイアログ・ボックス

Sequence Transaction フィールド	説明
FUNCTION:	実行するテストを指定します。UserFunction、コンパイル済み関数、リモート関数を呼び出すか、評価対象となる式を入力できます。呼び出した関数(または評価対象の式)の結果は、仕様を基準としてテストされます。 UserFunctionが複数の値を返す場合、VEEは、一番上の出力ピンがテスト対象となる結果を保持していると想定します。 関数は組み合わせたり、ネストすることもできます。 例、**(random(0,myfunc()+3,100)*2)**。
LOGGING ENABLED	テスト・データのログを記録します。ログ記録に関するオプションを指定するには、Sequencerのオブジェクト・メニューをオープンし、[Properties]を選択し、[Logging]フォルダをクリックし、一覧から選択します。By default, Name, Result, and Pass are checked.[Log to Output Pin Only] と[Log Each Transaction To:]のいずれかを選択するフィールドもあります。ログ記録をイネーブルに設定した場合は、各テストでログのレコードが記録されます。 このボタンは**[LOGGING DISABLED]**にトグルします。
IF PASS	分岐命令を決めます。テストが成功した場合、VEEは、ここで分岐命令を確認します。**[IF PASS]**では、VEEはドロップダウン・メニューで選択されている分岐命令を実行します。 このボタンは**[IF PASS CALL:]**にもトグルします。 **[IF PASS CALL:]**では、指定した関数が呼び出されてから、分岐メニューで選択した命令に分岐します。 この表の次の項目である**[THEN CONTINUE]**も参照してください。

表40　[Sequence Transaction]ダイアログ・ボックス

Sequence Transaction フィールド	説明
THEN CONTINUE	テストの分岐を決めます。**[IF PASS]**と**[IF FAIL]**の場合、ドロップダウン・メニュー**[THEN CONTINUE]**には6つの分岐オプションがあります。 • **[THEN CONTINUE]**は、**Sequencer**で設定している次のテストを実行します。 • **[THEN RETURN:]**では、VEEはテストの実行を停止し、指定された式をSequencerのReturn出力ピンに出力します。 • **[THEN GOTO:]**は、フィールドで指定しているテストにジャンプします。 • **[THEN REPEAT]**は、最大で**[MAX TIMES:]**フィールドで指定している回数だけ、現在のテストを繰り返します。最大回数だけ繰り返した後もまだ**PASS/FAIL**条件が存在する場合は、VEEは次のテストを続けます。 • **[THEN ERROR:]**は、所定のエラー番号の付いたエラー条件を生成することにより、実行を停止します。エラーは、SequencerのError出力ピンを使って補足できます。ほかの出力ピンはデータを送信しません。 • **[THEN EVALUATE:]**は、指定されたUserFunctionを呼び出します。このUserFunctionは、分岐メニューのいずれかのオプションを指定する文字列を返す必要があります。UserFunctionが返す有効な文字列は、Continue、Return <式>、Goto <名前>、Repeat <式>、Error <式>です。ここで、<**式**>は有効なVEE式、<**名前**>はシーケンス内のテストの名前です。このオプションを使用すると、次に行うことをユーザとの対話で決めることができます。
IF FAIL	分岐の指示。テストが失敗した場合、VEEは、ここで分岐命令を確認します。**[IF FAIL]**は**[IF FAIL CALL:]**にトグルします。オプションは**[IF PASS]**の場合と同じです。
DESCRIPTION:	テストについてのテキスト・コメント。テキスト・コメントはSequencerトランザクション・バーに表示され、[Properties]ダイアログ・ボックス内のLoggingフォルダを使用すると、テスト・レコードとともに保管できます。

テストを追加、挿入、または削除する方法

このセクションでは、別のテスト・トランザクションをSequencerオブジェクトに追加します。同じrandom()関数を使用するとテスト結果をシミュレートできますが、今回は、結果を値の範囲と比較するのではなく、リミットと比較します。

1 最初のSequencerトランザクション・バーの下をダブルクリックし、[Sequence Transaction]ダイアログ・ボックスを表示します。フィールドに次のように入力します。

表41　[Sequence Transaction]ダイアログ・ボックス

項目	説明
test2	デフォルトを使用します。
SPEC NOMINAL	設定を**.5**から**.25**に変更します。
RANGE	ドロップダウン・メニューで**[LIMIT:]**を選択します。演算子は<を選択します。リミットを**1**から**.5**に変更します。
FUNCTION	testFunc(a)からrandom()に変更します。

ほかの選択肢はデフォルトのままにしておき、**[OK]**をクリックして、Sequencerに戻ります。

注 記　**オブジェクト・メニューで[Add Trans...]を選択すると、強調表示されているトランザクションの後にトランザクションを追加することもできます。**

Sequencerテスト計画には、現在、**test2 (.25) < .5**という2番目のトランザクションが表示されています。次に、2つのテストの間にトランザクションを1つ挿入します。

2 2番目のトランザクション・バーが強調表示されているかどうか確認します。次に、オブジェクト・メニューをオープンし、[Insert Trans...]を選択します。フィールドに次のように入力します。

表42　　トランザクションの挿入

フィールド	説明
TEST	名前フィールドを`Insert`に変更します。
FUNCTION	**random( )**に変更します。

[OK]をクリックします。2番目のトランザクション・バーに**Insert 0 <= (.5) <= 1**と表示されます。プログラムを実行し、3つのテストからの3つのレコードを表示します。すべてのエントリを表示するには、表示を拡大しなければならない場合もあります。

3 Insertを削除します。削除するには[Insert]トランザクション・バーをクリックして、マウス・ポインタを[Insert]トランザクション・バー上に置き、`Ctrl+K`を押します。

注 記

ターゲットのトランザクション・バーをクリックし、オブジェクト・メニューで[Cut Trans]を選択することもできます。オブジェクト・メニューで[Paste Trans]を選択し、切り取っておいたトランザクションを貼り付けることもできます(ショートカットは`Ctrl+Y`)。同様に、[Copy Trans]を選択すると、トランザクションをコピーできます。

4 プログラムを実行し、2つのテストからのデータ・レコードを確認します。プログラムに**seq1.vee**と名前を付けて保存します。プログラムは図205のように表示されます。

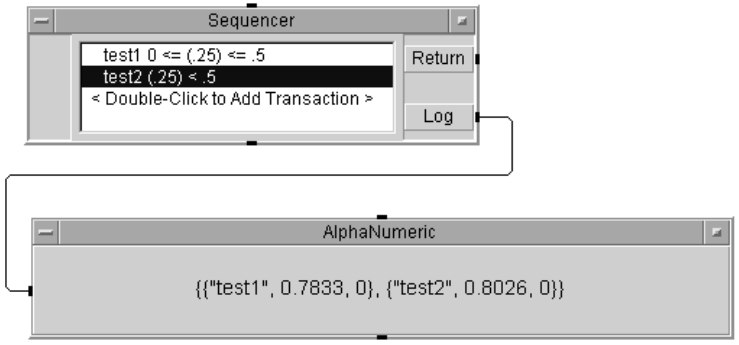


図205　　単純なSequencerの例

中かっこ{}は、Recordデータ型を示します。Sequencerは、AlphaNumeric表示に表示されているように、Record of Records(レコードから成るレコード)を出力します。このことは、ループ内にsequencerを置いて同じテスト・シーケンスを数回実行すると、Records of Records(レコードから成るレコード)を要素とする配列を生成できることを意味します。

ログとして記録されているテスト・データにアクセスする方法

SequencerはRecord of Records(レコードから成るレコード)を出力します。それぞれのテストは、Sequencerレコード内のフィールド名としてテスト名を使用します。それぞれのテスト内のフィールドの名前は、ログ記録に関する設定に従って付けられます。フィールドName、Result、Passについてデフォルトの設定を使用すると、図206に示すように、Log.Test1.Resultという表記を使ってtest1にある結果にアクセスできます。

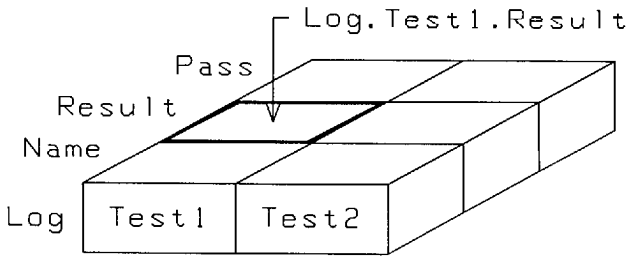


図206　　ログとして記録されているレコードまたは複数のレコード

テスト結果にアクセスするには、次の手順に従います。

1 **seq1.vee**をオープンします。

2 **[Device]** ⇒ **[Formula]**を選択し、表示オブジェクトの下に配置します。式を**Log.Test1.Result**に変更します。VEEでは大文字と小文字を区別する必要がありません。名前の中で大文字を使用しているのは、マニュアルにおいて名前を見やすくするためです。

入力端子名を**A**から**Log**に変更します。デフォルト名**A**をそのままにしておくこともできます。その場合、公式は**A.Test1.Result**となります。Sequencerの出力端子LogをFormulaの入力端子Logに接続します。

3 **[Display]** ⇒ **[AlphaNumeric]**表示を選択し、Formulaの出力に接続します。

4 プログラムを実行します。プログラムは、Test1内のResultフィールドにアクセスします。プログラムに**seq2.vee**と名前を付けて保存します。プログラムは図207のように表示されます。

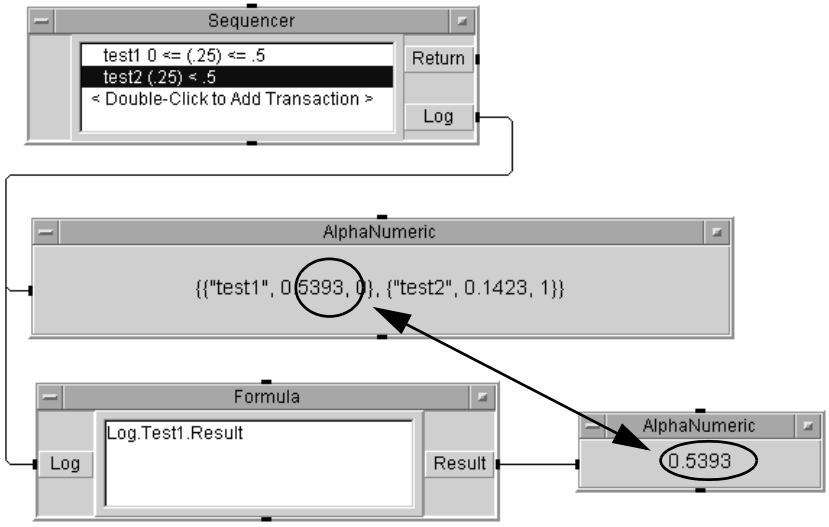


図207　　ログとして記録されているデータにアクセスする方法

注 記

それぞれのテストは、Sequencer内で実行されたときに、テスト名を使って名前を付けたレコードを作成します。このレコードは後続のテストで使用できます。たとえば、test1が成功(PASS)した場合、test2をイネーブルできます(**ENABLED IF: test1.pass == 1**)。テストがまだ実行中であるときに、式フィールド内のテスト・データにアクセスする必要がある場合、テスト・データはテンポラリ・レコードthistestに保管されています。

5 公式をLog.test1に変更し、プログラムをもう一度実行します。プログラムはtest1のレコード全体を読み取ります。test1のレコード全体は、表示内の3つの値を囲んでいる中かっこ{}によって示されています。

6 公式を変更すると、test1とtest2のレコード内のフィールドresult、pass、name、またはその他のフィールドにアクセスできます。[Properties]ボックス内の[Logging]タブを選択し、ログとして記録されているレコードにNominalとTime Stampフィールドを追加します。Formulaオブジェクトを使用して、新しいフィールドにアクセスします。

Sequencerを使ってデータを渡す方法

この例題では、UserFunctionを作成し、3つの別々のテストから呼び出します。1つ目では、Sequencerの入力端子を介してUserFunctionにデータを渡します。2つ目では、入力端子ではなく、グローバル変数を使用するようにプログラムを変更します。これにより、TESTモードではなくEXECモードで関数を呼び出すことができます。3つ目では、波形出力をマスクと比較するテストを行う方法を習得します。

例題10-2: 入力端子を使用してデータを渡す方法

最初に、次の手順に従ってUserFunction Randを作成します。これは計測手順をシミュレートするものです。Rand() は、入力パラメータをrandom(low,high)オブジェクトの出力に追加し、結果を出力ピンに出力します。3つの別々のテストからRand()を呼び出します。

1. **[Device]** ⇒ **[UserFunction]**を選択します。名前UserFunction1をRandに変更します。
2. random(low,high) 関数を表示し、入力端子を削除し、パラメータを削除し、Rand内に配置します。パラメータがなければ、デフォルトは0と1になります。A+Bオブジェクトをrandom(low,high)の右に配置します。random(low,high)の出力を**A+B**オブジェクトの左上の入力に接続します。
3. Randにデータ入力端子を追加します。入力端子**A**を**A+B**オブジェクトの左下の入力端子に接続します。
4. Randにデータ出力端子を追加します。**A+B**オブジェクトの出力をRandの出力端子に接続します。

   UserFunction Randは図208のように表示されます。

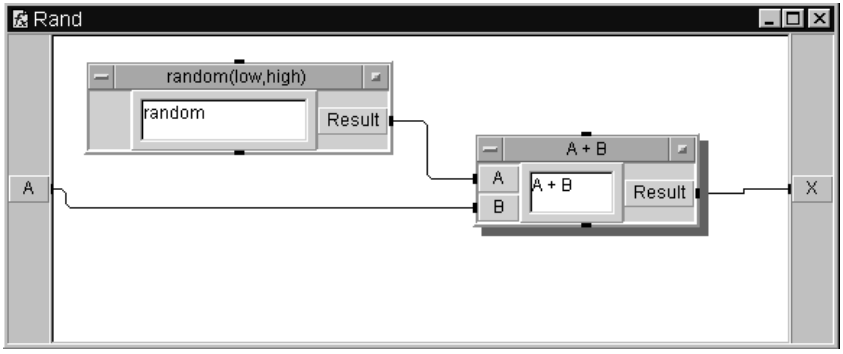


図208　　Rand **UserFunction**

5 プログラムに**seqdat1.vee**と名前を付けて保存します。一番上の右端にある[x]ボタンを使ってRandウィンドウをクローズします。

注 記

ウィンドウをクローズしても、UserFunctionは削除されません。このことを確認する場合は、[Edit] ⇒ [Edit UserFunction]をクリックします。編集対象のUserFunctionを一覧表示したリスト・ボックスにRandが表示されます。またはRand関数をアイコン化することもできます。その場合は、VEE画面の下部にアイコンが表示されます。

Sequencer内の3つのテストをセットアップします。3つのテストでは、Randを呼び出し、Sequencerの入力ピンを使って入力パラメータをRandに供給します。

6 **[Device] ⇒ [Sequencer]**を選択し、メイン内に配置します。Sequencerに入力端子を追加します。トランザクション・バーをクリックして、[Sequence Transaction]ダイアログ・ボックスを表示します。**[FUNCTION]**フィールドを**[testFunc(a)]**から**[rand(a)]**に変更します。これにより、UserFunction Rand()が呼び出され、Sequencerの入力端子**A**の値がRand()に送信されます。**[OK]**をクリックして、Sequencerのオープン・ビューに戻ります。

注 記

AなどのSequencerの入力端子名を使用しても、[Sequence Transaction]ボックス内の式フィールドにデータを渡すことができます。たとえば、Aを使用して、RANGE:とSPEC NOMINAL:にデータを渡すことができます。

トランザクションが強調表示されていることを確認してから、カーソルをトランザクション・バー上に置き、Ctrl+Kを押してテストをカットし、次に、Ctrl+Yを3回押して、テストを元のSequencerに貼り付けます。オブジェクト・メニューを使ってもカットと貼り付けができます。

デフォルトのテスト名は、test1x2、test1x1、test1となります。3つの[Sequence Transaction]ダイアログ・ボックスをオープンし、これらの名前をtest1、test2、test3というわかりやすい名前に変更します。

1 **[Data]** ⇒ **[Continuous]** ⇒ **[Real64 Slider]**を選択し、Sequencerの左に配置します。名前をユーザからの入力を要求する方式のSelect Num:に変更し、オブジェクトのサイズを少し小さくし、Sequencerの入力端子に接続します。

オブジェクトを配置したとき、マウスの左ボタンを使ってオブジェクトの四隅のいずれかをクリック・アンド・ドラッグすると、オブジェクトのサイズを決められます。

2 AlphaNumeric表示を選択し、Sequencerの下に配置し、少し幅を拡大し、SequencerのLog出力端子に接続します。

3 プログラムをseqdat1として再度セーブします。Real64 Sliderオブジェクトにある数字を1つ選択し、seqdat1を実行します。図209のように表示されます。

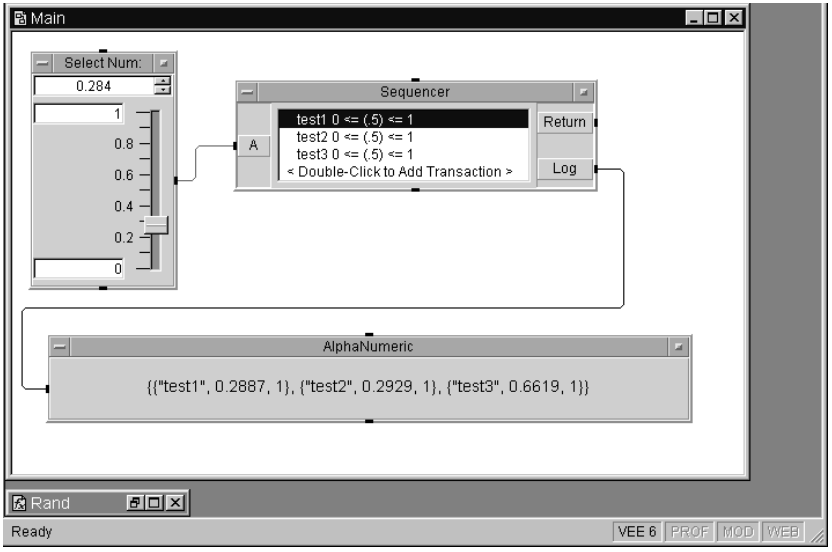


図209　　入力端子を使用してデータを渡す方法

テスト数が増えると、入力端子を使ってデータを渡すのに必要な入力ピンが多くなります。入力ピンを減らすには、入力端子にレコードを渡して、レコード内の別々のフィールドを別々のテストのために使用します。別々のUserFunctionを使用して、グローバル変数をセットアップすることもできます。グローバル変数は、ほかのUserFunctionから、またはプログラム内の式フィールドから呼び出すことができます。次の例題でこのことを例証します。

グローバル変数を使用してデータを渡す方法

この例題では、seqdat1プログラムを変更して、UserFunction Randにパラメータ**a**を渡すためにグローバル変数を追加します。

1. Select Numというラベルの付いている Real64 Sliderオブジェクトを削除します。Sequencerの**A**入力端子を削除します。
2. test1トランザクション・バーを強調表示し、オブジェクト・メニューをオープンし、**[Insert Trans...]**をクリックします。[Sequence Transaction]ボックスが表示されたら、**[TEST]**をクリックして、**[EXEC]**にトグルし、名前を`Setup`に変更します。
3. **EXEC**モードを使用する理由は、User Functionがグローバル変数をセットアップするだけで、仕様に照らしてテストする必要がある結果を生成しないことにあります。
4. **[FUNCTION]** フィールドを `[global()]` に変更し、**[OK]** をクリックして、ダイアログ・ボックスをクローズします。
5. 次に、**UserFunction global()**を作成します。
6. **[Device]** ⇒ **[UserFunction]**を選択します。名前**UserFunction1**を`global`に変更します。

   [Data] ⇒ **[Continuous]** ⇒ **[Real64 Slider]** を選択し、UserFunction内に置き、名前を`Select Num:`に変更し、縦方向のサイズを少し小さくします。

   [Data] ⇒ **[Variable]** ⇒ **[Set Variable]**を選択し、Real64 Sliderの右に配置します。

   グローバル変数名を**globalA**から`a`に変更します。Real64 SliderをSet Variableオブジェクトに接続します。

画面にオペレータが数字を選択するための関数を表示するため、ポップアップ・パネル・ビューを追加します。**[Confirm (OK)]**ボタンを加えます。これにより、オペレータが選択を終えるまで、パネルは画面に表示されたままとなります。この作業は、**global() UserFunction**内の実数入力用ダイアログ・ボックスを使って行うこともできます。

7 **[Flow]** ⇒ **[Confirm(OK)]**を選択し、Real64 Sliderオブジェクトの上に配置します。OKデータ出力ピンをReal64 Sliderのシーケンス入力ピンに接続します。

注 記

OKボタンをSet Variableオブジェクトの下に配置した場合は、論理エラーとなります。これは、VEEがSliderにある古い値をSet Variableオブジェクトに送信し、[OK]ボタンが押されるまで休止するからです。ポップアップ・パネルで入力した新しい値はすべて無視されます。OKをReal64 Sliderの上から接続した場合は、[OK]が押されてからグローバル変数が設定されるため、新しいSlider値を使用できます。[Show Data Flow]をオンにすると、実行の順序を監視できます。

8 **[Display]** ⇒ **[Note Pad]**を選択し、テンプレート情報を削除します。オブジェクトを[OK]ボタンの右に配置します。Note Padに、次のユーザ・プロンプトを入力します。

```
Please select a number for this run of tests
1, 2, and 3.
```

9 Ctrlキーを押したまま、[Note Pad]、[Real64 Slider]、[OK]ボタンをクリックして、3つのオブジェクトを選択します。各オブジェクトは、選択されていることを示す網かけ表示になります。**[Edit]** ⇒ **[Add To Panel]**をクリックします。[Edit]メニューは、VEE画面、あるいはUserObjectまたはUserFunctionの詳細ビューの空いている領域で右ボタンをクリックしても表示できます。パネル・ビューで、パネルのサイズを少し小さくし、Note Padを一番上に、Real64 Sliderを中ほどに、[OK]ボタンを一番下に配置します。

注 記

パネル・ビュー内でオブジェクトを再配置しても、詳細ビュー内でのオブジェクトのレイアウトには影響ありません。

UserFunction の[Properties] ウィンドウをオープンします。[Pop-up Panel]の[General]フォルダで、[ShowPanelonExecute]プロパティをTrue に設定します。図210は詳細ビューのUserFunctionを示し、図211はそのパネル・ビューを示します。

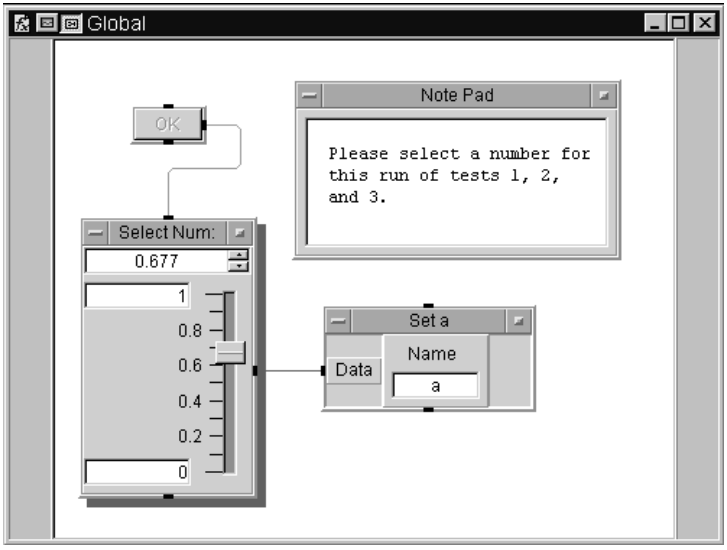


図210　Global UserFunction(詳細)

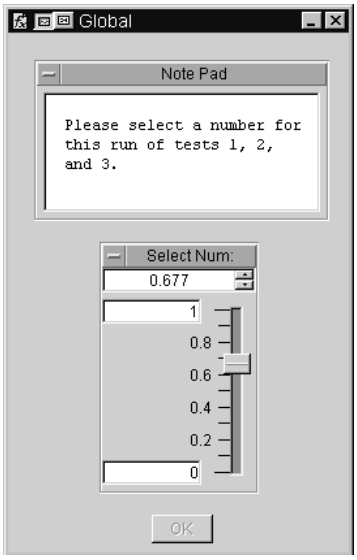


図211　　Global **UserFunction**(パネル)

10 seqdat2という名前を付けてプログラムをセーブし、実行します。ポップアップ・パネルが表示されたら、値を選択し、[OK] を押します。図212のように表示されます。

注 記

ポップアップ・パネルは、デフォルトでは、画面の中央に表示されます。移動するには、タイトル・バーをクリック・アンド・ドラッグします。

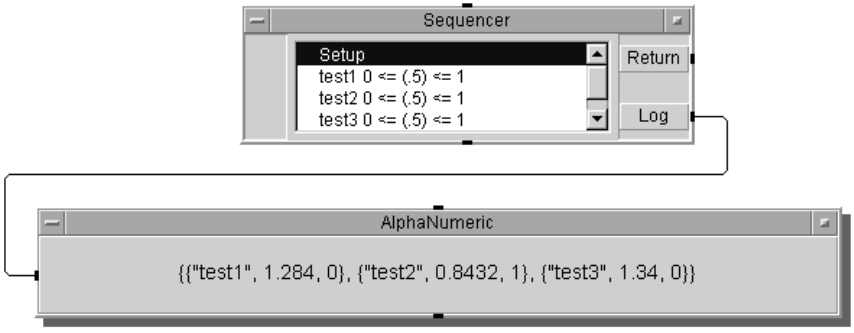


図212　　グローバル変数を使用してデータを渡す方法

波形出力をマスクと比較する方法

この例題では、noisyWvという名前のUserFunctionを作成し、Sequencer内の単一のトランザクション・バーから呼び出します。オペレータは、波形の振幅を0から1の間で変更できます。この関数は、ノイズの大きい波

形を返すテスト結果をシミュレートします。**[Data]** ⇒ **[Constant]**メニューのCoordオブジェクトを使用して、**0.6**で直線マスクを作成します。Sequencerは、これを使ってノイズの大きい波形をテストします。

1 図213の詳細ビューで示すように、noisyWvという名前のUserFunctionを作成します。

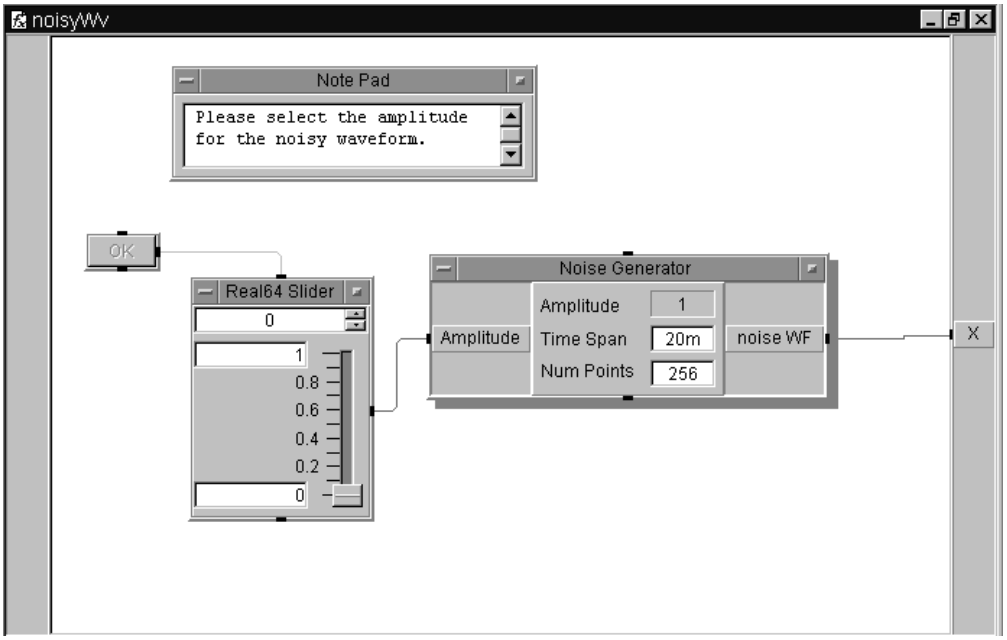


図213　noisyWv UserFunction(詳細)

2 パネル・ビューを作成するため、Ctrlキーを押したまま、[OK]ボタン、[Real64 Slider]、[Note Pad]をクリックして、強調表示します。**[Edit]** ⇒ **[Add To Panel]**を選択します。

パネル・ビューが表示されたら、オブジェクトを任意に並べ替え、ウィンドウのサイズを変更します。

オブジェクト・メニューをオープンし、**[Properties]**をクリックし、[Pop-up Panel]で[Show Panel on Execute]の横をクリックします。

パネル・ビューは図214のように表示されます。

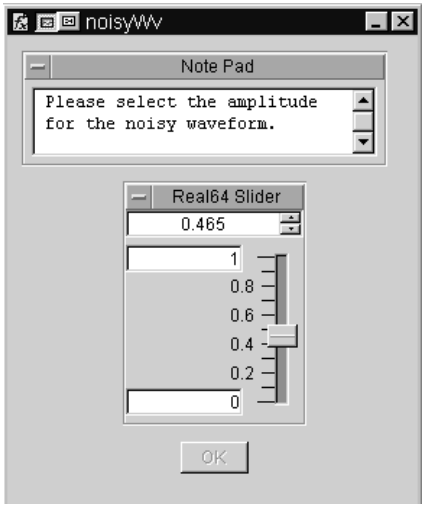


図214　noisyWv UserObject(パネル)

3 **[Device]** ⇒ **[Sequencer]**を選択し、メイン左中央に配置します。データ入力端子を追加し、maskという名前を付けます。

4 トランザクション・バーをクリックして、[Sequence Transaction]ダイアログ・ボックスを表示します。フィールドを次のように変更します。

表43　[Sequence Transaction]ダイアログ・ボックス

フィールド	説明
FUNCTION	「noisyWv()」と入力します。
RANGE	ポップアップ・メニューで[LIMIT]をクリックして選択します。<=と、LIMITの端子名フィールドの種類であるmaskはそのままにしておきます。その他のデフォルト値はすべてそのままでよいので、**[OK]**をクリックします。

test1ではnoisyWv()から結果が得られるので、その結果を**mask**という名前の入力端子でリミット値と比較するテストを行います。すべての点でノイズの大きい波形がマスクと比較して小さいか等しい場合、波形はテストに合格(PASS)です。そうでない場合、波形は不合格(FAIL)となります。

5 **[Data]** ⇒ **[Constant]** ⇒ **[Coord]**を選択し、Sequencerの上に配置します。その出力をSequencerの入力端子maskに接続します。

Coordのオブジェクト・メニューをオープンし、**[Properties]**をクリックし、フィールドを次のように設定します。

表44　Coordオブジェクトのプロパティ

プロパティ	説明
DataShape	1D Arrayに設定します。
ArraySize	「2」と入力します。直線を指定するには、2組の座標があればよいためです。

6 Coordオブジェクトに2組の座標のインデックスが表示されます。最初のインデックスの**[0000:]**をダブルクリックして、カーソルを表示します。カンマで区切って座標を入力すると、VEEが自動的に丸かっこを追加します。「`0, 0.6`」と入力し、`Tab`キーを押し、「`20m, 0.6`」と入力してから、このオブジェクトの外の作業領域をクリックします。次のエントリがあります。

- noisyWv()内のNoise Generatorの`x`軸(時間軸)は**0から20**ミリ秒まで進みます。したがって、`x`軸の2つの値は`0`と`20m`です。
- 直線マスクが必要なので、`y`軸の2つの値は、両方とも**0.6**です。

注 記　**正しい数の座標の組を設定して入力すると、どのようなマスク波形でも作成できます。**

Sequencerの比較のメカニズムは、Coordデータ型を受け付けて波形をテストするComparatorオブジェクトと同じように働きます。もちろん、2つのWaveformデータ型を比較することもできます。座標の組を移動するときはTabキーを押し、入力が終わったら作業領域をクリックします。

7 AlphaNumeric表示を選択し、幅を広げ、SequencerのLog出力に接続します。

8 **seqdat3**という名前を付けてプログラムをセーブし、実行します。図215のように表示されます。

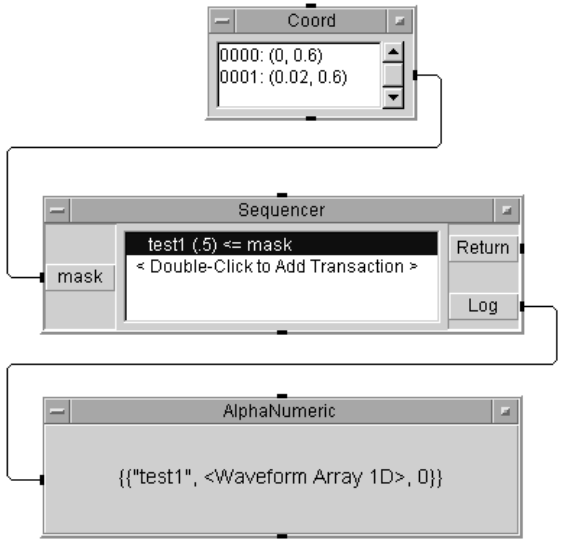


図215　　波形をマスクと比較する方法

Sequencerを使ってデータを渡す例題はこれで完了です。次の例題では、Sequencerを数回繰り返して得たデータにアクセスし、解析する方法を習得します。

Sequencerから得たデータを解析する方法

前に触れたように、Sequencerのデータは、レコードから成るレコードとして出力されます。ただし、多くの場合、Sequencerは一連のテストを数回実行することができます。これにより、レコードを要素とする配列が生成されます。各レコードはSequencerによる1回の実行を表し、1回の実行内の各テストを表すほかのレコードを保持しています。このことを視覚的に理解する最も容易な方法は、図216に示すようなメモリ内のデータから成る立方体を想像してみることです。

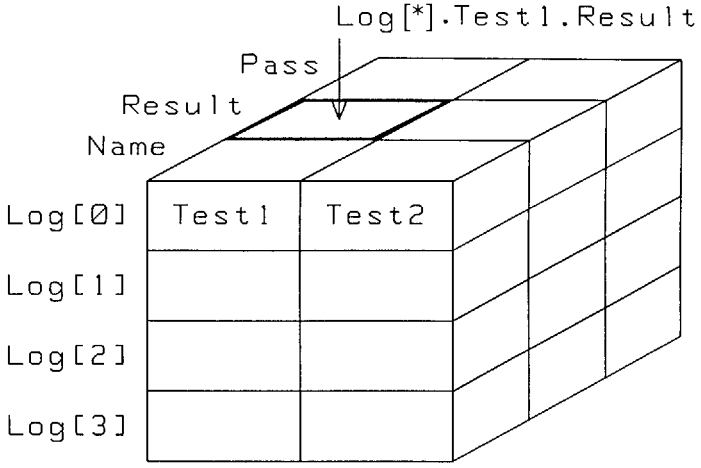


図216　　ログとして記録されたレコードから成るレコードを要素とする配列

レコードを要素とする配列はLogと呼ばれます。これは、LogがSequencerの出力ピンに関連付けられている名前だからです。特定の実行にアクセスするには、角かっこ[]表記を使って配列のインデックスを作成します。

- **Log[0]**は**Sequencer**による最初の実行、**Log[1]**は2番目の実行などとなります。

- 各実行のメイン・レコードには、**Test1**と**Test2**の2つのフィールドがあります。
- レコード**Test1**内には、**Name**、**Result**、**Pass**の3つのフィールドがあります。レコード**Test2**の場合も同じです。
- したがって、**Log.Test1.Result**は、4回の実行のいずれかを表す4つの値を要素とする配列となります。**Log[0].Test1.Result**は、1回目の実行(**Log[0]**)での**Test1**の**Result**であるスカラ値を出力します。

ログとして記録されたレコードを要素とする配列は、データの解析と調査を簡略化してくれます。たとえば、特定の実行で成功したテストの数を確認する場合があります。すべての実行でのTest2の結果を平均したり、4回目の実行でのTest1のすべてのデータを見る場合もあります。このデータを使用すれば、これらの照会はすべて可能です。次の例題では、データに対する解析操作を行います。

例題10-3: Sequencerを数回実行して得られたデータを解析する方法

1 画面をクリアし、**seqdat1.vee**プログラムをオープンします。

Sequencerを3回実行するように**seqdat1.vee**プログラムを変更します。次に、そのデータに対する解析操作を行います。

2 **[Flow]** ⇒ **[Repeat]** ⇒ **[For Count]**を選択し、Real64 Sliderオブジェクトの上に配置します。繰返し回数を 3 に変更し、データ出力ピンをSequencerのシーケンス入力ピンに接続します。

3 SequencerのLogピンと表示オブジェクトを結んでいるデータ・ラインを削除します。**[Data]** ⇒ **[Collector]**を選択し、Sequencerの右に配置します。左上のデータ入力ピンをSequencerのLogピンに接続し、XEQピン(左下)をFor Countオブジェクトのシーケンス出力ピンに接続します。Collectorデータの出力ピンをAlphaNumericの画面に接続します。3つの要素を持つ配列が表示できるように表示を縦方向に少し拡大します。

Sequencerはtest1とtest2を3回実行し、データを3つの要素から成る配列の中に収集します。各要素は各実行のレコードから成るレコードを保持します。これを視覚的に理解するには、図216のデータから成る立方体を参照してください。

この時点でプログラムを実行し、Sequencerデータの表示を調べます。

Formulaオブジェクトを使用して、解析する部分のデータを抽出します。この例題では、3回の実行に関するtest1の結果を例に取り上げ、配列の平均値を計算します。

4 **[Device]** ⇒ **[Formula]**を選択し、表示オブジェクトの下に配置します。Formulaの入力ピンをCollectorの出力に接続します。Formulaの入力フィールドを`a[*].test1.result`に変更します。`mean(x)`オブジェクトをFormulaに、AlphaNumeric表示をmean(x)に接続します。

aは入力端子**A**にある配列を参照します。**Test1.result**は正しいフィールドにアクセスします。すべての実行が1つの配列に表示されます(たとえば、**A[0].test1.result**とすると、1回目の実行だけを参照します)。

5 プログラムを実行します。図217のように表示されます。

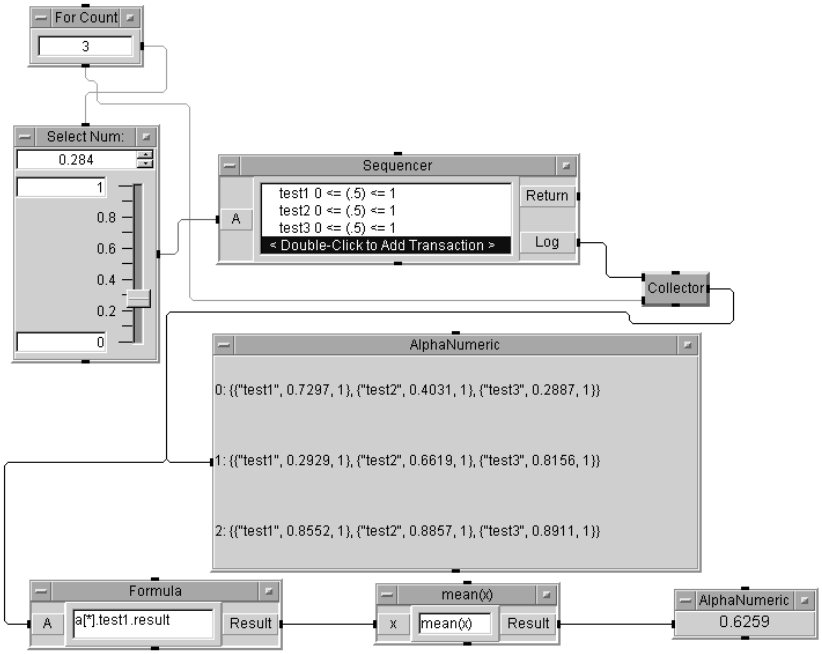


図217　　Sequencerを数回実行して得られたデータを解析する方法

この例題では、単一の配列にアクセスしますが、Sequencerの出力からデータを要素とするほかの複数の配列を抽出する場合も、原則は同じです。[Sequencer Properties]ダイアログ・ボックスの[Logging folder]をオープンすると、それぞれの配列についてどのフィールドをセーブするかを容易に変更できます。

ログとして記録されているデータの保管と読み取りの方法

この例題では、To/From FileオブジェクトとTo/From DataSetオブジェクトの使用方法を示します。

例題10-4: ログとして記録されているデータに関してTo/From File オブジェクトを使用する方法

1. **seqdat2**ファイルをオープンし、表示オブジェクトに接続されているデータ・ラインを削除します。
2. **[Flow]** ⇒ **[Repeat]** ⇒ **[For Count]**を選択し、Sequencerの左に配置します。For Countの数を3に変更し、データ出力ピンをSequencerのシーケンス入力ピンに接続します。
3. 作業領域を縦方向に拡大し、AlphaNumeric表示を下部近くに配置します。**[Data]** ⇒ **[Collector]**を選択し、左の作業領域に配置します。SequencerのLogピンをCollectorのデータ入力ピンに接続します。For Countのシーケンス出力ピンをCollectorのXEQピンに接続します。

   CollectorはSequencerのレコードを使用して、レコードから成るレコードを要素とする配列を作成します。To Fileオブジェクト内のWRITE CONTAINERトランザクションを使用すると、任意のVEEデータ・コンテナを容易にファイルに書き込むことができます。
4. **[I/O]** ⇒ **[To]** ⇒ **[File]**を選択し、Collectorの右に配置します。**[I/O]** ⇒ **[From]** ⇒ **[File]**を選択し、To Fileオブジェクトの下に配置します。入力端子をTo Fileオブジェクトに追加し、Collectorの出力を接続します。To Fileのシーケンス出力をFrom Fileのシーケンス入力ピンに接続します。From Fileのデータ出力を表示オブジェクトに接続します。

   To File内の**[Clear File At PreRun & Open]**にチェック・マークを付け、WRITE CONTAINER aトランザクションを設定します。From Fileオブジェクト内でトランザクションをREAD CONTAINER xと設定します。

   デフォルトのデータ・ファイルを保管のために使用できます。
5. プログラムを実行します。図218のように表示されます。

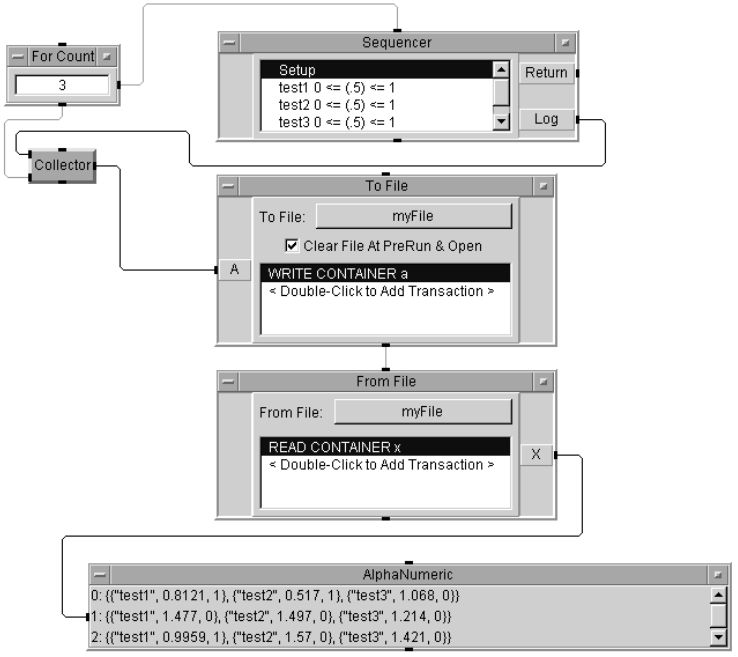


図218　To/From Fileを使用して、ログとして記録されているデータを保管する方法

ログとして記録されているデータに関してTo/From DataSetオブジェクトを使用する方法

テスト・データをレコードとして保管しようとしているので、To/From DataSetオブジェクトが適しています。この場合、Collectorは必要ありません。Sequencerのそれぞれの実行結果をDataSetの末尾に追加できるからです。

最後のプログラムを図219のように変更します。To/From DataSetオブジェクトは[I/O]メニューにあります。From DataSetに向かうシーケンス・ラインに注目してください。このシーケンスが加えられているのは、3回すべての実行結果がDataSetの末尾に追加されるのを待ってから、From DataSetをトリガする必要があるからです。

From DataSetオブジェクト内のSearch Specifier機能を使用すると、データを役立つ情報に変換できるので、To/From Fileオブジェクトではなく、To/From DataSetオブジェクトを使ってデータを収集するほうが有利です。

注 記

From DataSet内の[Get records]フィールドを忘れずに**[One]**から**[All]**に変更してください。

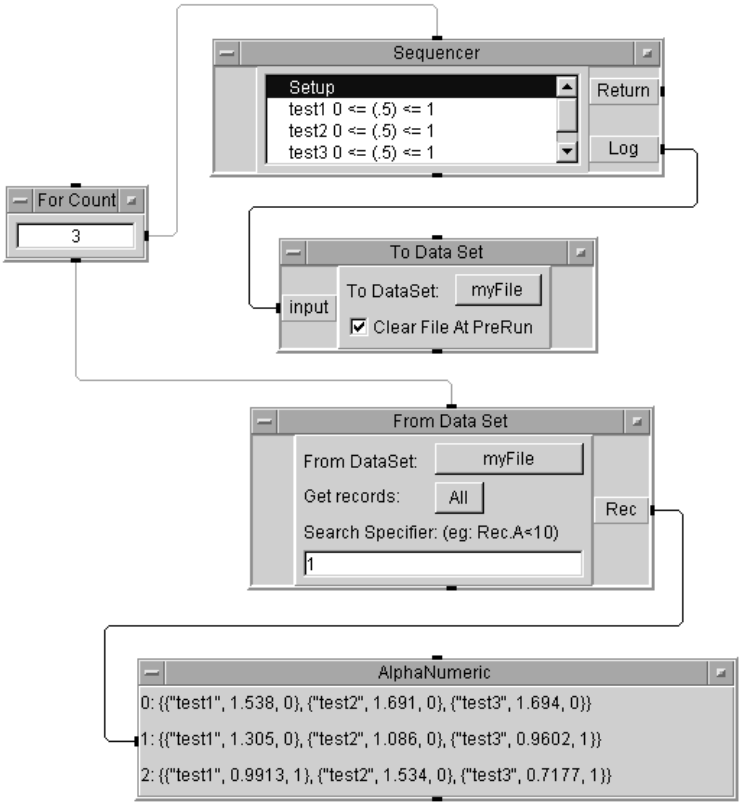


図219　To/From DataSetを使用して、ログとして記録されているデータを保管する方法

この章の復習

次の章に進む前に、次のチェックリストを使用して、必要ならトピックを復習してください。

- Sequencerオブジェクトの概念を説明できる。
- Sequencer用のテストを設定する。
- Sequencerによるテストの操作を追加、挿入、削除する。
- ログとして記録されているデータにSequencerからアクセスする。
- Sequencerの入力端子を使ってデータをテストに渡す。
- グローバル変数を使ってデータをテストに渡す。
- 波形出力をマスクと比較する。
- Sequencerを数回実行して得られたデータを解析する。
- To/From Fileオブジェクトを使ってデータを保管する。
- To/From DataSetオブジェクトを使ってデータを保管する。

11
オペレータ・インタフェースの使用方法

オペレータ・インタフェースの使用方法

この章の内容

- オペレータ・インタフェースを構築する方法
- オペレータ用のメニューを使用する方法
- 明快さを加えるためのビットマップをインポートする方法
- テスト・プログラムを保護する方法
- オペレータ・インタフェースの機能
- ActiveXコントロールを使ってVEEの機能を拡張する方法

平均所要時間: 2時間

概要

この章では、メニューを追加する方法、インタフェースをカスタマイズする方法、警告信号を追加する方法、ビットマップをインポートする方法を始めとするオペレータ・インタフェースについて詳しく学びます。この章では、これまでの章でオペレータ・インタフェースとポップアップ・パネルを作成した例題をさらに詳細に説明します。

VEEのオペレータ・インタフェース機能を使用する利点の一部は次のとおりです。

- オペレータにとって使いやすさが最大限になること
- プログラムのパフォーマンスが向上すること
- 許可のない変更から保護されること
- 視覚的補助による明快さがあること

オペレータ・インタフェースに関する重要点

このセクションでは、VEEでオペレータ・インタフェースを作成する方法の概要について説明します。

オペレータ・インタフェースを作成する方法

VEEには、オペレータ・インタフェースを作成するためのさまざまな選択コントロール、ポップアップ・ダイアログ・ボックス、インジケータ、表示があります。選択コントロールには、ボタン、スイッチ、チェックボックス、ドロップダウン・メニュー、リスト・ボックスなどがあります。インジケータには、タンク、温度計、塗りつぶしバー、音量単位メータ、color alarmなどがあります。

VEE内で提供するオペレータ・インタフェース要素に加え、外部から入手した要素を追加できます。WWWからダウンロードできるオペレータ・インタフェース要素は多数あります。ActiveXコントロールを介して使用できるオペレータ・インタフェースもあります。ダウンロードできるものには、無料のものと有料のものがあります。

VEEで提供されているオペレータ・インタフェース・オブジェクトだけを使用しても、外部から入手した自分専用の要素を追加しても、オペレータ・インタフェースを作成するプロセスは同じです。

VEEプログラムのためのオペレータ・インタフェースを作成するには、プログラムのパネル・ビューを作成します。

1. パネル・ビューに必要なオブジェクトを選択します。複数のオブジェクトを選択する場合は、`Ctrl`キーを押したまま、それぞれのオブジェクトをクリックします。
2. **[Edit]** ⇒ **[Add To Panel]**を選択します。画面が詳細ビューで強調表示したオブジェクトを含むパネル・ビューに切り替わり、デフォルトでは青で表示されます。

これで、オペレータに必要なものだけを表示するようにカスタマイズできるVEEプログラムのビューが作成されました。

パネル・ビューと詳細ビュー間を移動する方法

VEEプログラムのパネル・ビューと詳細ビュー間を移動するには、図220に示すように、ウィンドウの**タイトル・バー**にあるパネル・アイコンまたは詳細アイコンをクリックします。

注 記　**ウィンドウのタイトル・バーにパネル・ビュー・ボタンを表示させるには、プログラムのパネル・ビューを作成する必要があります。**

通常は、詳細ビューでプログラムを開発してから、オペレータ・インタフェースのためのパネル・ビューを作成します。パネル・ビュー・ボタンは、UserObjecウィンドウ、UserFunctionウィンドウ、またはメイン・ウィンドウのタイトル・バーに置くことができます。

.

図220　**タイトル・バーのパネル・ビュー・ボタンと詳細ビュー・ボタン**

オペレータ・インタフェースをカスタマイズする方法

VEEプログラムのパネル・ビュー内では、オブジェクトのサイズを変更したり、オブジェクトの配置を変更したり、オブジェクトを表示する方法を変更できますが、**詳細ビュー内の同じオブジェクトには影響を与えません**。たとえば、Waveform (Time)表示のパネル・ビューからタイトル・バーと目盛り類を削除しても、同じWaveform (Time)表示の詳細ビューには影響を与えません。ただし、詳細ビュー内のオブジェクトを削除した場合は、パネル・ビュー内の該当オブジェクトも削除されます。

パネル・ビュー内では、強調するために異なる色またはフォントを選択したり、明瞭に見せるために拡大可能なビットマップを選択できます。オペレータのためにパネル・ビューの文書化を行うこともできます。その場合は、タイトル・バーを編集し、Note PadとLabelオブジェクトを使用し、オブジェクト・メニューのDescriptionオプションを使用します。

図221は、使用可能なVEEインジケータを示したものです。

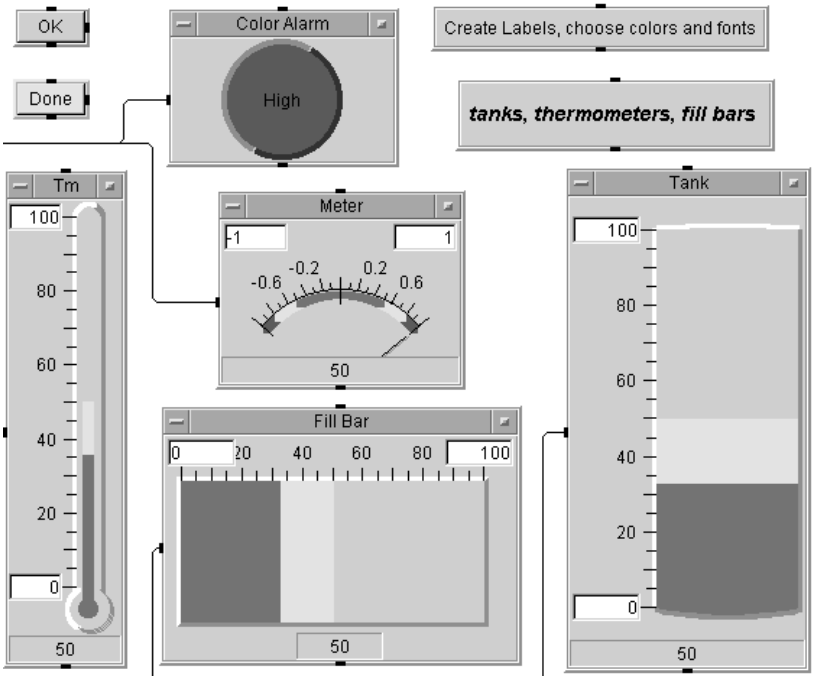


図221　　選択の対象となるVEEインジケータ

オペレータ・インタフェース・オブジェクトの使用方法

このセクションでは、VEEで使用可能なオペレータ・インタフェースのオブジェクトとオプションを紹介します。このセクションをひととおり読むと、プログラムのオペレータ・インタフェースを作成するために選択できる項目と、インタフェースをカスタマイズする方法がわかります。次に例題を実行して、通常のタスク用にオペレータ・インタフェースをセットアップする方法を理解します。

色、フォント、インジケータ

色とフォント

色とフォントについては、**[File]** ⇒ **[Default Preferences]**を選択するか、それぞれのオブジェクト・メニューの[Properties]を選択すると設定できます。色とフォントの選択肢は、インストールしているオペレーティング・システムとフォントによって異なります。

Color Alarms

Color alarmオブジェクトは**[Display]** ⇒ **[Indicator]**メニューにあります。Color alarmオブジェクトには3つの異なる範囲の色とテキスト・メッセージを設定でき、四角または円形を選択できます。通常Alarmは、LEDをシミュレートしたり、対処が必要な状況であることをオペレータに警告するために使用されます。

タンク、温度計、塗りつぶしバー、メータ

これらのオブジェクトも**[Display]** ⇒ **[Indicator]**サブメニューにあります。色とラベルを使用してカスタマイズできます。水平フォーマットまたは垂直フォーマットに設定でき、3つの異なるデフォルト範囲を設定できます。設定はオブジェクト・メニューの[Properties]で行います。

グラフィック・イメージ

Propertiesボックスの[Panel]フォルダで、[Background Picture]を設定すると、パネル・ビュー内にビットマップをインポートできます。VEEがインポートするファイルは、`*.jpeg`、`*.png`、`*.wmf`、`*.GIF`、`*.bmp`で、メイン、UserObject、UserFunctionパネルの背景として使用できます。

ビットマップを背景のピクチャとして設定した場合、ほかのVEEオブジェクトはそのピクチャの上に載った形で表示されます。インポート方法についての詳細は、425ページの「例題11-2: パネルの背景に使用するビットマップをインポートする方法」を参照してください。イメージは拡大または縮小したり、タイルにしたり、切り取ったり、中央寄せできます。

図222は、タイルにした背景ピクチャを示します。

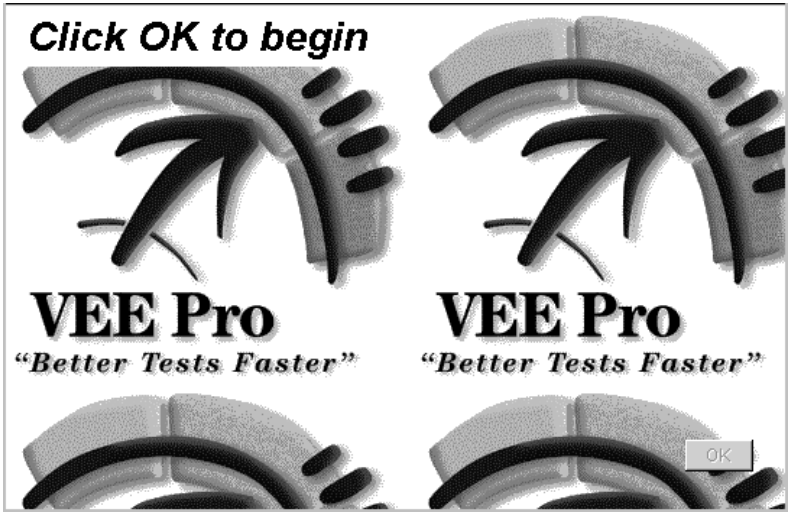


図222　　タイルとして使用している背景ピクチャ

プログラム内にビットマップを配置する必要がある場合は、[Display]メニューにPictureオブジェクトもあります。図223 は、**[Display]** ⇒ **[Picture]**を選択して組み込んだ上で、VEEで切り取ったピクチャを示します。

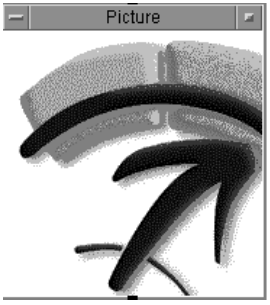


図223　　VEEで切り取られたイメージ

注 記　**[Properties]** ⇒ **[Icon]**タブを使用すると、アイコン用のビットマップも変更できます。

オペレータ入力のためのコントロールの表示方法

オペレータが入力によってプログラムを制御できるようにプログラムをセットアップする方法は多種あります。プログラムがユーザ入力を取得できるのは、ポップアップ・ダイアログ・ボックス、データ定数、スライダ、ノブからです。コントロールを選択するには、**[Data]** ⇒ **[Selection Control]**、**[Data]** ⇒ **[Toggle Control]**、**[Data]** ⇒ **[Continuous]**などのメニューに移動します。図224 は、オペレータにプログラムをわかりやすくするために使用できるオブジェクトのコレクションを示します。

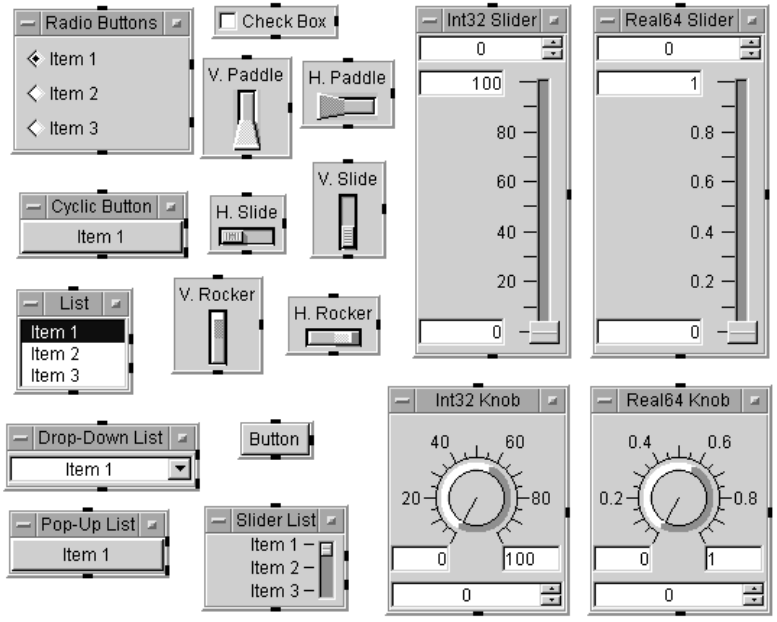


図224　　[Data]のさまざまなサブメニューにあるコントロール

図224に示しているオブジェクトは、外観をカスタマイズできます。たとえば、図225の[Real64 Knob Properties]ダイアログ・ボックスを見てください。このオブジェクトを設定するには、ForeColorsなどのプロパティを選んで、選択を行います。

注 記

ActiveXを使用すると、432ページの「例題11-4: ActiveXコントロールの使用方法」に示しているように、ほかのアプリケーションにあるコントロールと表示を使用することもできます。

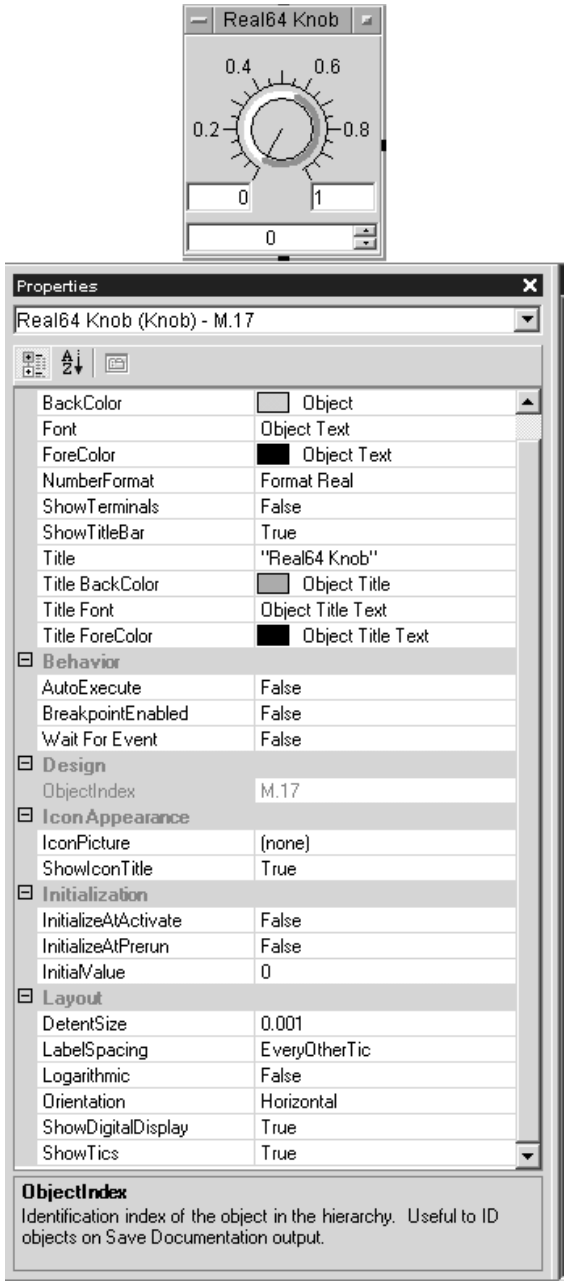


図225　[Properties]ダイアログ・ボックス

オペレータ入力のためのダイアログ・ボックスの表示方法

VEEには、自動エラー検出機能、プロンプト、エラー・メッセージを備えた組み込みポップアップ・ダイアログ・ボックスがあります。これらのダイアログ・ボックスは、**[Data]** ⇒ **[Dialog Box]**にあります。

たとえば、プログラムの実行中にオペレータに実数の入力を要求できます。プログラムの実行中にオペレータのために**[Real64 Input]**ボックスを自動的に表示する**Real64 Input**オブジェクトをプログラムに組み込むことができます。また、**[Real64 Input]**ボックスはオペレータがプロンプトで正しい情報を入力しなかった場合に、エラー・メッセージを自動的に表示します。図226 は、プログラムに組み込むオブジェクトと、プログラムの実行中に表示される**[Real64 Input]**ボックスを示しています。

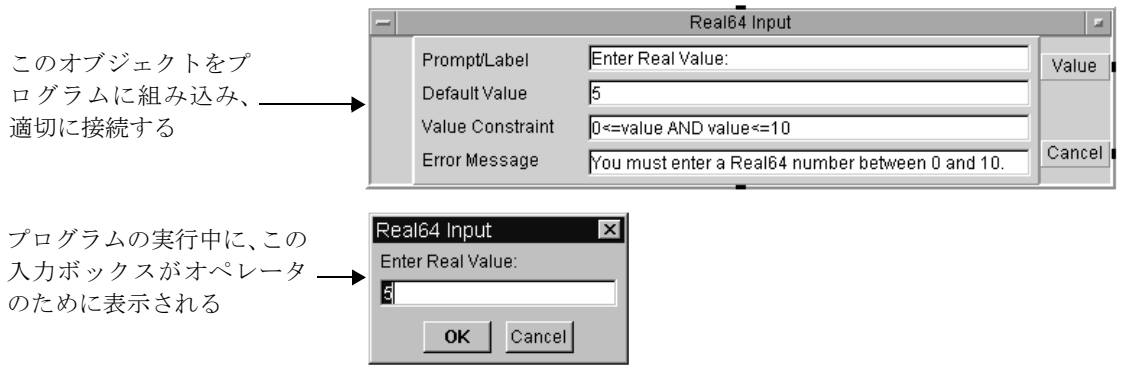


図226　　テキスト入力ボックス

図227は、プログラムの実行中に、オペレータが[Real64 Input]ボックスに正しい情報を入力しないで**[OK]**を押した場合に表示される設定可能エラー・メッセージを示します。

図227　　自動エラー検出の例

Int32とText用の入力ボックスは、**[Data]** ⇒ **[Dialog Box]**にあり、[Real64 Input]に似ています。さらに、**[Data]** ⇒ **[Dialog Box]**メニューには、[Message Box]、[List Box]、[File Name Selection]の選択肢があります。

図228に、ポップアップしてメッセージを表示するダイアログ・ボックスを示します。

図228　　ポップアップ・メッセージ・ボックス

図229に、オペレータがリストを入力するためにポップアップするダイアログ・ボックスを示します。

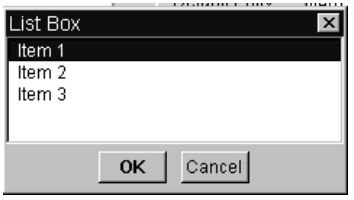


図229　　リスト選択ボックス

図230に、オペレータがファイル名を選択するためにポップアップするダイアログ・ボックスを示します。

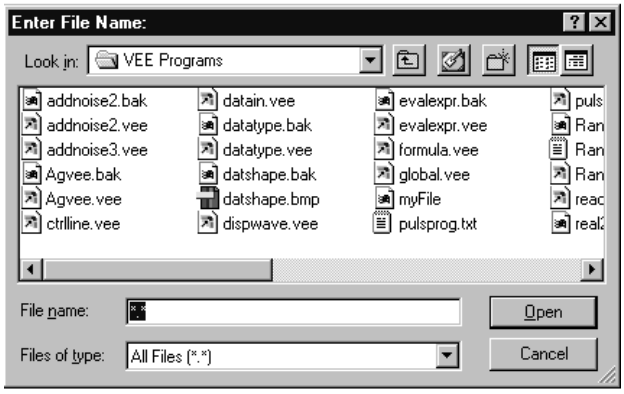


図230　　ポップアップ・ファイル選択ボックス

オペレータのためにトグル・コントロールを表示する方法

VEEには、0または1を送出するために使用できる組み込みトグル・コントロールがあります。トグル・コントロールを使用するには、初期状態を設定しておき、トグルがアクティブ化されたときにサブプログラムを実行します。Toggleにカスタム・ビットマップを置くこともできます。

たとえば、オペレータがスイッチまたはalarmを設定する必要があるプログラムの場合、トグル・コントロールを使用できます。図231に、オペレータがスイッチを設定するためのパネルを示します。

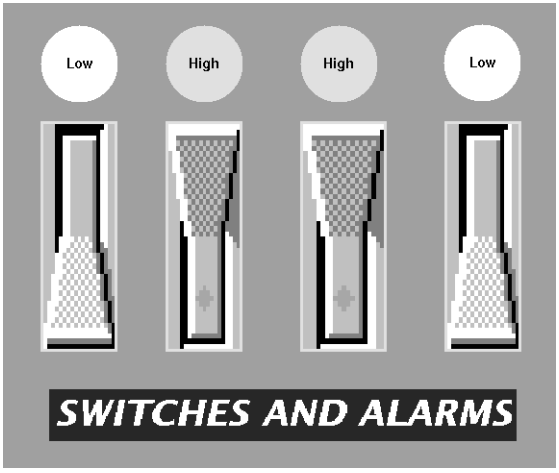


図231　　組合わせたスイッチとAlarm

オペレータ・インタフェースでオブジェクトを位置合わせする方法

パネル・ビューには、デフォルトでグリッドが表示されます。[PanelDrawGrid]プロパティを[False]にトグルすることにより、グリッドをオフにできます。パネル・ビューには、オブジェクトの位置合わせに役立つ自動「snap-to-grid」機能もあります。図232に示すように、グリッドを表示し、PanelGridSizeプロパティを10から1(10がデフォルト)に変更すると、厳密な位置合わせができます。この機能を使用すると、プログラムにきれいな外観を与えることができます。「snap-to-grid」機能は、[UserObject]または[UserFunction]メニューの[Properties]の下のPanelフォルダにあります。Panelフォルダ選択をダイアログ・ボックスで表示するためには、パネル・ビューをあらかじめ作成しておく必要があるので、注意してください。

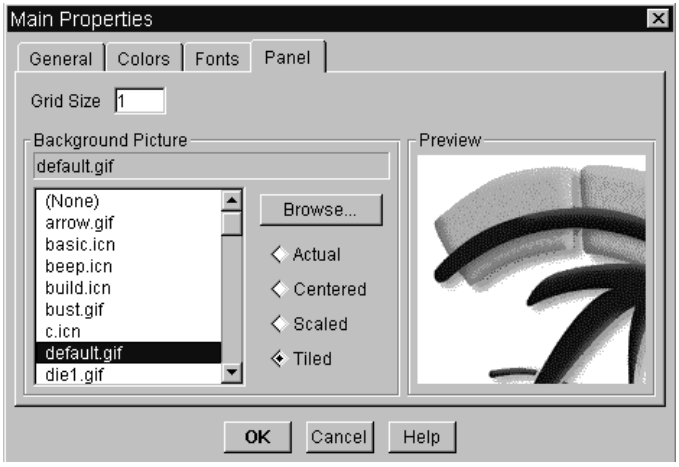


図232　　パネルのプロパティを設定する方法

その他のパネル・フォーマット機能

より機能的なパネル表示を実現するため[Tab Order]と[Front/Back]という追加の機能が提供されています。[Tab Order]を使用すると、パネルのすべてのオブジェクトに、Tabキーでオブジェクト間を移動するときの順番を表す数字が割り当てられます。この順番はユーザが自由に変更できるので、オペレータがパネルのオブジェクトを対話型で処理する順番を簡単に変更できます。[Front/Back]機能は、どのオブジェクトをスタックの後ろに置き、どのオブジェクトをスタックの前に置くかを制御します。オブジェクトのスタックでこの機能を使用すると、オブジェクトの重なりかたを変更できます。

キーボードのみのオペレータ・インタフェースを作成する方法

VEEを使用すると、オペレータがキーボードだけを使用して制御できるインタフェースを作成することもできます。これらのインタフェースはマウスを必要としません。

たとえば、OKオブジェクトをソフトキーとして設定することができます。通常、OKオブジェクトの設定ではいずれかのファンクション・キーに接続します。これにより、図233に示すように、ファンクション・キーを押して、プログラムを制御できます。

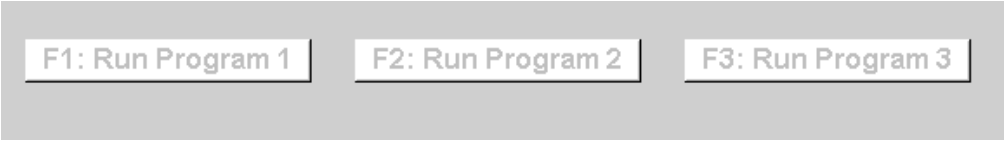


図233　　UserFunctionを実行するソフトキー

図234は、Properties...ダイアログ・ボックスを使用して、OKオブジェクトをファンクション・キー、Enterキー、またはEscキーに接続するように設定する方法を示します。

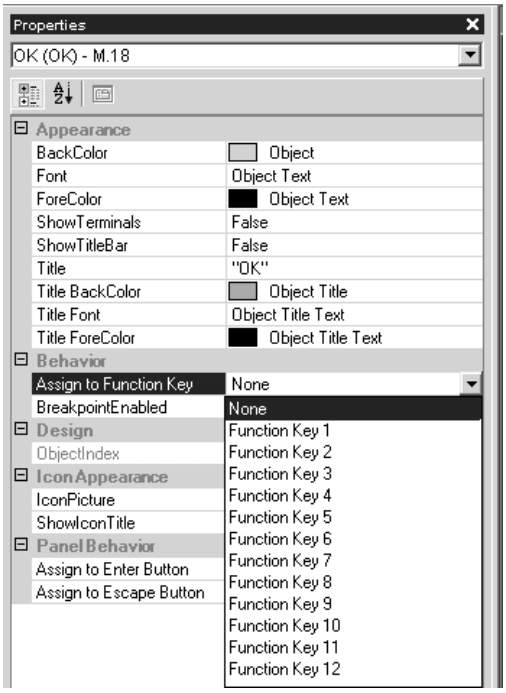


図234　　Confirm(OK)オブジェクトをソフトキーとして設定する方法

さらに、パネル・ビュー内のキーボードを使ってプログラムを制御できます。VEEは、点線を使ってパネル用のボタンを自動的に強調表示します。オペレータがEnterを押すと、実際の該当ボタンが「押されます」。オペレータがテキスト入力領域を編集している場合、Enterキーを押すと編集内容が受け付けられ、Escキーを押すと編集が中止されます。Tabキーを押すと、異なる入力オブジェクトの選択肢間を前進し、アクティブ・オブジェクトを表示します。Shift+Tabキーを押すと後退します。プログラムの実行を制御する場合は、次の組合わせを使用します。

表45

キーの組合わせ	説明
Ctrl+G	実行または継続(再開)
Ctrl+P	休止
Ctrl+T	ステップ実行

画面の色を選択する方法

画面の色を選択するには、**[File]** ⇒ **[Default Preferences]**ダイアログ・ボックスを使用します。VEE環境を任意に設定し、変更をセーブします。図235と図236は、特定の画面要素を目的の色に変更する方法を示します。

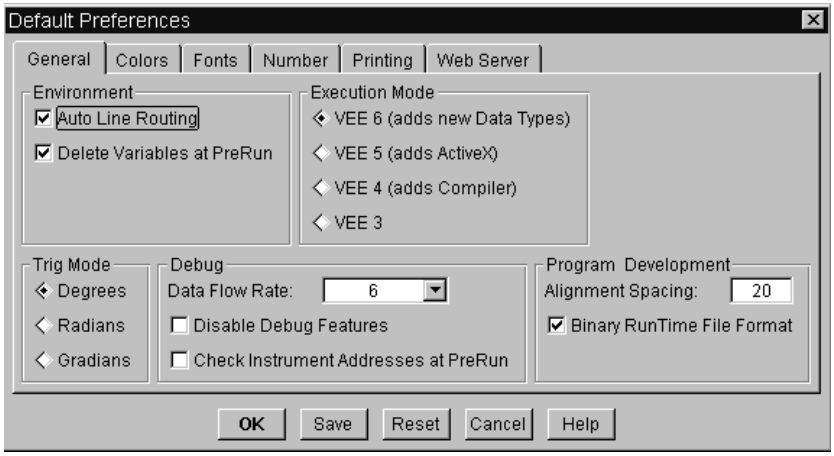


図235　[Default Preferences]ダイアログ・ボックス

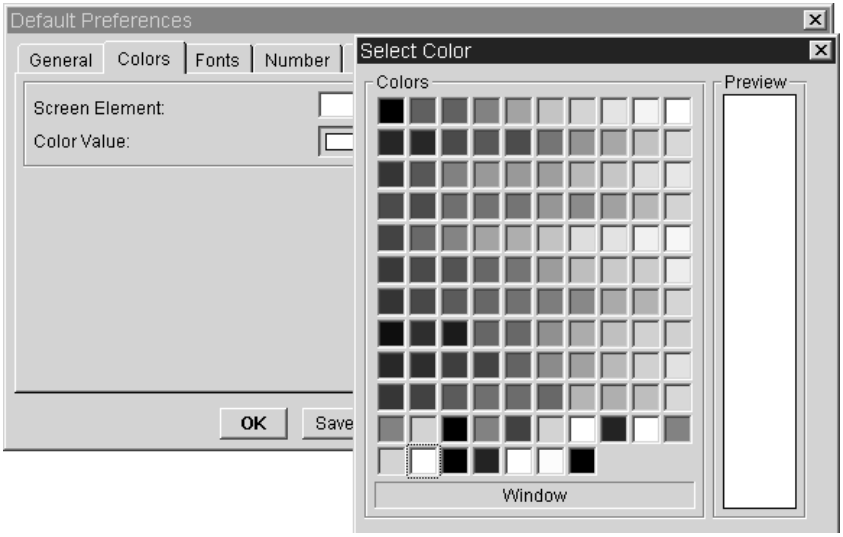


図236　　画面要素の色の選択

プログラムを保護する方法(RunTimeバージョンを作成する方法)

オペレータが誤ってプログラムを変更したり、他の人がプログラムを詳細ビューで表示してプログラムの動作のしくみを見れないようにするため、VEEプログラムの実行時バージョンを作成できます。**元のプログラムと実行時バージョンは、別のファイルにセーブしてください。**

注 記

VEEプログラムの実行時バージョンを作成したとき、実行時バージョンは編集**できません**。詳細ビューを表示することはできません。このため、このプロセスを開始する前に元のプログラムのコピーを複数作成し、指示に従ってください。

VEEプログラムのRunTimeバージョンを作成するには、以下の手順に従います。

1 プログラムの実行時バージョン用のパネル・ビューを作成します。

2 編集可能なコピーを入手するためプログラムをセーブします。

3 **[File]** ⇒ **[Create RunTime Version...]**を選択します。VEEが、実行時バージョンであることを示す拡張子***.vxe**を自動的に付加します。

実行時にポップアップ・パネルを表示する方法

UserObjectまたはUserFunctionがプログラムで実行されたときにパネルをポップアップさせることができます。ポップアップ・パネルを表示するには、オブジェクト・メニューの[Properties]で[ShowPanelonExecute]を選択します。オペレータが先に進む準備ができるまでパネルを画面に表示したままにしておくには、Confirm (OK)オブジェクトを追加します。追加しなければ、UserObjectまたはUserFunctionの実行が終わると、パネルが消えます。

UserFunctionを複数回呼び出す間、ポップアップ・パネルを表示したままにしておくには、ShowPanel()とHidePanel()関数を使用します。たとえば、プログラムの実行中、ポップアップ・パネルをステータス・パネルとして表示したままにしておくことができます。例として、次のセクションを参照してください。

ステータス・パネルの作成方法

VEEでは、複数のテストまたは関数の結果をモニタするためステータス・パネルを作成することができます。この機能を実現するには、図237で示すようにShowPanel()関数とHidePanel()関数を使用します。詳細については、434ページの「例題11-5: ステータス・パネルを作成する方法」を参照してください。

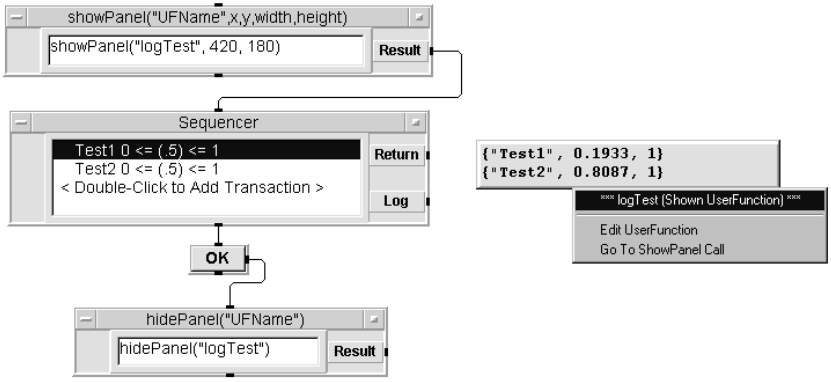


図237　　ステータス・パネルを作成する方法

オペレータ・インタフェースを作成する場合の通常の作業

以下の例題では、オペレータ・インタフェースの多くの機能を実装する方法を習得します。具体的には、メニューを作成する方法、警告を作成する方法、ステータス・パネルを作成する方法、プログラムにより視覚的なインパクトを加えるためのビットマップをインポートする方法を習得します。これらをすべて使って、オペレータ・インタフェースをカスタマイズします。

例題11-1: メニューを使用する方法

この例題では、3つの選択肢**die1**、**die2**、**die3**のあるメニューを持つオペレータ・インタフェースを作成します。オペレータが選択肢を選択すると、同じ名前の関数が呼び出され、表面に1つ、2つ、または3つの目を持つサイコロを表示します。このプログラムでは、オペレータが実行するテストをメニューで選択しなければならない状況をシミュレートします。アイコンの外観を変更するため、ビットマップをインポートする方法も習得します。このプログラムはDice Programと呼びます。

3つのUserFunctionを作成することから始めます。

1 **[Device]** ⇒ **[UserFunction]**を選択します。

インポートされたビットマップを表示するには任意のにアイコンを使用できますが、この例ではPictureオブジェクトを使用します。

2 **[Display]** ⇒ **[Picture]**を選択し、UserFunction内に配置します。

3 **Picture**のオブジェクト・メニューをオープンし、**[Properties]**をクリックしてから、[ShowTitleBar]プロパティを[False]に設定します。[Picture]で**[die1.gif]**を選択し、**[Scaled]**をクリックしてから、**[OK]**をクリックします。

注 記　[Show Title Bar]がオフのときにオブジェクト・メニューにアクセスするには、オブジェクト上で右ボタンをクリックします。

注 記　デフォルトではbitmapsサブディレクトリが使用されますが、どのディレクトリにあるビットマップでも使用できます。

次に、サイコロの表面に目が1つあるピクチャを作成する必要があります。

4 **[Flow]** ⇒ **[Confirm (OK)]**を選択し、サイコロの下に配置します。Pictureシーケンス出力ピンをOKシーケンス入力ピンに接続します。

5 PictureとOKオブジェクトを選択します。`Ctrl`キーを押したままオブジェクトをクリックして、網かけ表示にします。背景にマウス・ポインタを置いてマウスの右ボタンを押し、ポップアップの[Edit]メニューをオープンします。[Add to Panel]を選択します。

6 UserFunction TitleとPanel Titleを`die1`に変更します。必要に応じて、オブジェクトの配置とサイズを変更します。

注 記

パネル・ビュー内でオブジェクトを移動するには、オブジェクトを右クリックし、[Move]を選択します。またはオブジェクトを強調表示し、オブジェクトに灰色の境界線が現われたら、それを新しい場所までドラッグします。

[Properties]ダイアログ・ボックスで**[ShowPanelonExecute]**を選択します。位置合わせをより正確に行うため、[Panel]フォルダをクリックし、**[PanelGridSize]**プロパティをアクティブにして、グリッド・サイズを2に変更します。次に**[OK]**をクリックします。

7 die1のオブジェクト・メニューで[Clone]を選択し、UserFunctionのクローンを2つ作成します。新しいUserFunctionは、自動的に**die2**および**die3**として表示されます。ピクチャ・オブジェクトをそれぞれ`die2.gif`と`die3.gif`に変更します。名前とビットマップを除いて、新しい関数のすべての設定が必ずdie1と一致するように設定します。プログラムは図238のように表示されます。関数ウィンドウをアイコン化します。

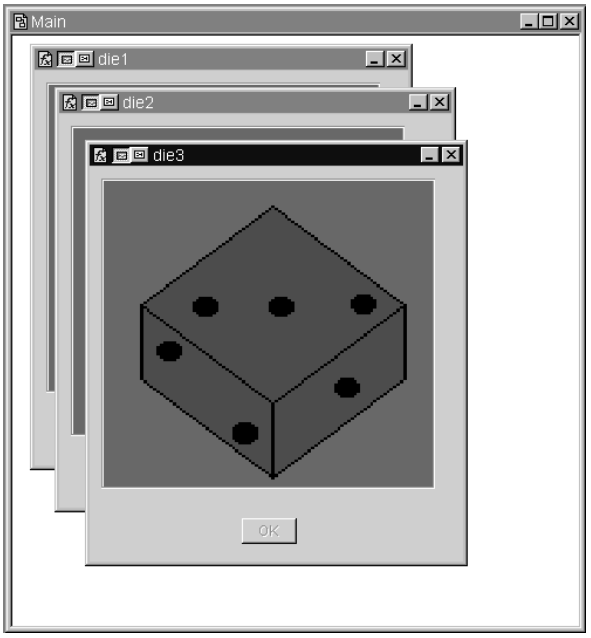


図238　　Dice Programの初期段階

3つの関数から呼び出す関数を選択するメニューを作成します。

8　**[Data]** ⇒ **[Selection Control]** ⇒ **[Radio Buttons]**を選択します。

Radio Buttonsは、ユーザ定義のリストにある列挙型の値(Enumデータ型は、序数が関連付けられているテキスト文字列)を上の出力ピンに出力するオブジェクトです。たとえば、リストをMonday、Tuesday、Wednesday、Thursday、Fridayと定義した場合、オペレータはメニューで曜日を選択でき、選択すると、Radio Buttonsは曜日を出力します。

リスト内の最初の項目には、順序を表す位置0を割り当てます。リスト内のn番目の項目には、順序を表す位置n-1が割り当てられます。たとえば、上述のリスト内のMondayの順序を表す位置は0、Fridayの順序を表す位置は4です。順序を表す位置は、下の出力ピンに表示されます。詳細は、オブジェクト・メニュー内のヘルプのエントリを参照してください。

9 Radio Buttonsのオブジェクト・メニューをオープンし、[Edit Enum Values...]を選択します。

関数の名前を**「die1」**、**「die2」**、**「die3」**と入力します。最後のエントリを**除き**、次のエントリに移るときはTabキーを押します。**[OK]**をクリックします。

注 記

データ選択を制御するためのメニュー・フォーマットは6種類あります。Radio Buttonsは、エントリをボタンとして表示します。オペレータの選択したデータは、Enumデータ型としてテキスト・フォーマットで出力されます。Cyclic Buttonは、オペレータがボタンを1度クリックするたびに1つずつ列挙型値を循環させます。Listは、リスト内のすべての列挙型値について選択されている項目を強調表示して表示します。そのほかの選択肢としてDrop-down list、Pop-up list、Slider listがあります。

10 Radio Buttonsのオブジェクト・メニューをオープンし、**[Properties]**をクリックしてから、**[Auto Execute]**を選択します。タイトルをプロンプト形式の`Make a Selection`に変更します。

Radio Buttonsオブジェクトで選択される値がCall Functionオブジェクトによって呼び出される関数の名前となるように、Callオブジェクトをセットアップします。

11 **[Device]** ⇒ **[Call]**をクリックします。**[Add Terminal]** ⇒ **[Control Input]**を選択してから、[Function Name] を選択し、**[OK]** をクリックします。Function Nameコントロール・ピンはEnum値またはText値を入力として受け入れます。Radio Buttonsのデータ出力ピンをCall Functionオブジェクトの Function Name入力端子に接続します。Radio Buttonsのシーケンス出力ピンをCall Functionのシーケンス入力ピンに接続します。**Make a Selection:** 内の**[die2]** をクリックし、図239に示すように、**Call Function Name**が**die2**に変わります。

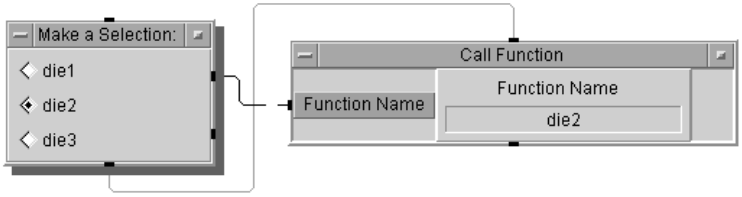


図239　Dice Program(詳細ビュー)

注 記　Callの入力端子は、Text Scalarを要求するため、VEEはEnum ScalarをText Scalarに変換します。

点線はコントロール・ピンを示すので、注意してください。Auto Executeをオンにした場合、Radio Buttonsは、変更が加えられたときはいつでも実行され、選択されたデータをCallに送信します。Call Functionのコントロール・ピンは、データを受信するとすぐ関数名を置き換えます。Callオブジェクトは、シーケンス入力ピンが起動されるまで、指定された関数を呼び出し**ません**。

注 記　**プログラムがAuto Executeとシーケンス・ピンを使用している場合、オペレータは[Run]ボタンをクリックしてプログラムを開始する必要はありません。**

プロンプト、メニュー、および選択対象を表示するポップアップ・パネルだけを表示するオペレータ・インタフェースを追加します。

12 `Ctrl`を押したままターゲット・オブジェクトをクリックして、Radio Buttonsオブジェクトを選択します。次に**[Edit]** ⇒ **[Add To Panel]**を選択します。

13 必要であれば、オブジェクト・メニューをオープンし、[Properties]を選択し、色とフォントを調整します。

プログラムを選択して実行します。プログラムはメニューですでに選択されたデータを使用するため、[Run]ボタンは使用しないでください。

プログラムを実行すると、図240のように表示されます。

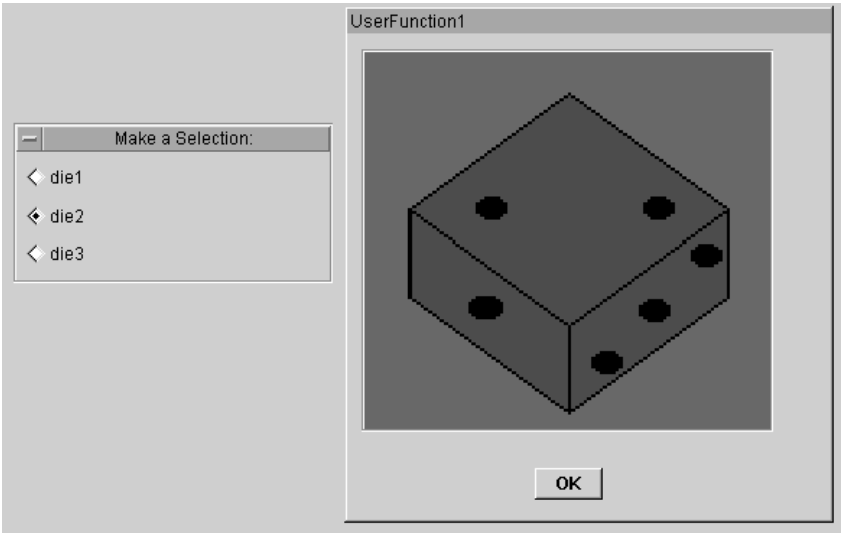


図240　　Dice Program(パネル・ビュー)

次の例題に移る前に留意点がいくつかあります。

- プログラムのためのメニューを作成する場合は、前の例題で使用した手法を使用できます。
- 呼び出し対象のプログラムを示したコントロール・ピン(Command)のあるExecute Programオブジェクトを使用すると、コンパイル済み言語のプログラムを選択する場合にRadio Buttonsも使用できます。コンパイル済み関数のライブラリをすでにインポートしている場合は、Callオブジェクトを使ってライブラリの関数を実行することもできます。
- 単一のUserFunction内でPictureオブジェクトのFile Nameデータ入力ピンを使用し、適切なビットマップ・ファイルをオブジェクトに送信すると、プログラムを最適化できます。これは、異なるビットマップを多数使用しようとする場合より効率的な方法です。

- より複雑なプログラムでは、AutoExecuteの代わりにRunを使用するのが通例です。AutoExecuteの代わりにWait for Inputを使用すると、データ定数または選択制御オブジェクトで、プログラムを休止させることができます。詳細は、ヘルプを参照してください。

例題11-2: パネルの背景に使用するビットマップをインポートする方法

ビットマップはプログラムに必須のものではありませんが、テストに明快さとインパクトを加えることができます。たとえば、テスト対象をよりよく図解するための図をインポートする場合があります。ここでは、標準VEEオブジェクトが表示されているパネルの背景に使用するビットマップをインポートします。

ビットマップは、アイコンやPictureオブジェクト用、またはUserObjectやUserFunction内のパネル・ビューの背景用としてインポートできます。LabelオブジェクトとConfirm (OK)オブジェクトを保持するBitmapという名前のポップアップUserFunctionを作成します。

1 **[Device]** ⇒ **[UserFunction]**を選択します。

2 **[Flow]** ⇒ **[Confirm (OK)]**および**[Display]** ⇒ **[Label]**を選択し、UserFunctionウィンドウ内に配置します。

3 UserFunctionの名前をBitmapに変更します。

4 OKとLabelオブジェクトを選択して、網かけ表示で強調表示します。ポインタをUserFunctionの作業領域に置き、マウスの右ボタンをクリックして、ポップアップの[Edit]メニューをオープンします。[Add to Panel]を選択します。

5 UserFunction のメニューをオープンし、[Properties] を選択してから、[ShowPanelonExecute]を選択します。[Properties]ボックスを表示するにはタイトル・バーをダブルクリックします。[Pop-up Panel]で[Show Title Bar]の選択を解除します。

   Panelフォルダをオープンし、[Grid Size]を[2]に変更し、[Background Picture]で**[default.gif]**と**[Scaled]**を選択してから、**[OK]**をクリックします。

6 **Label**オブジェクトのための**[Properties]**ボックスをオープンし、次のように設定します。

表46　Labelオブジェクトの[Properties]

Labelプロパティ	説明
Title:	`[Bitmap Function]`に変更します。
Justification	`[Center]`に変更します。
BackColor	[Light Gray]を選択し、**[OK]**をクリックします。
Font	太字のより大きいフォントを選択し、**[OK]**をクリックします。[Automatically Resize Object on Font Change]にチェック・マークを付けます。
Border	**[Raised]**をクリックします。**[OK]**をクリックして変更を行い、[Properties]をクローズします。

7 タイトルBitmap Functionと[OK]ボタンの配置を決めます。
Bitmap UserFunctionをアイコン化します。

8 メイン・ウィンドウに行きます。**[Device]** ⇒ **[Call]**をクリックしてから、オブジェクト・メニューで[Select Function]をクリックし、[Bitmap]を選択します。プログラムを実行します。ポップアップ・ボックスは図241のように表示されます。

図241　Bitmap Function

例題11-3: インパクトの強い警告を作成する方法

この例題には、ネストされている数個のUserFunctionがあります。最初のUserFunctionは、alarm自体で、赤い四角とビープを表示します。2番目のUserFunctionは、オペレータがalarmをオフにするまで、光が点滅する効果と短い間隔の断続音を繰り返し生成するalarmを呼び出します。

alarm関数のプログラミングから始めます。

1 **[Device]** ⇒ **[UserFunction]**を選択します。名前を`alarm`に変更します。

2 **[Display]** ⇒ **[Beep]**を選択し、UserFunctionの左上に配置します。高いビープ音が1秒間鳴り続くように設定を調整します。**[Duration (sec)]**フィールドを`[1]`に変更します。**[Volume (0-100)]**フィールドを`[100]`に変更します。

注 記

次の手順では、ビープ音をサポートするハードウェアを使用しているという仮定に立っています。一部のWindows 98、Windows 2000、Windows NT 4.0、Windows XPシステムでは、すでに[Default System Beep]の[Default System configuration]が変更されています。.

注 記

Beepオブジェクトはどこにも接続する必要はありません。関数が実行されると、このオブジェクトがアクティブ化します。

3 **[Display]** ⇒ **[Indicator]** ⇒ **[Color Alarm]**をクリックし、UserFunction内に配置します。Color Alarmのオブジェクト・メニューをオープンし、[Properties]をクリックし、次のように設定します。[ShowTitleBar property]プロパティをFalseに設定します。[Shape]プロパティを[Rectangular]に設定します。[HighLimit]のプロパティを[.66]に設定し、[HighText]プロパティの横にテキストがあれば削除します。

4 **[Data]** ⇒ **[Constant]** ⇒ **[Real64]**をクリックし、`[1]`に変更し、Color Alarmの入力ピンに接続します。

5 Beepオブジェクトと同期をとるために表示を1秒間画面に表示し続けるには、1秒に設定したDelayオブジェクトを使用します。

6 **[Flow]** ⇒ **[Delay]**を選択し、[1]に設定し、シーケンス入力ピンをColor Alarmのシーケンス出力ピンに接続します。これで、alarmが1秒間鳴り続けます。

7 **[Display]** ⇒ **[Note Pad]**を選択し、テンプレートを削除します。TURN OFF INSTRUMENTS!というメッセージを追加します。必要に応じてNote Padのサイズを調整します。

8 メイン・ウィンドウに行きます。**[Device]** ⇒ **[Call]**を選択してから、Callのオブジェクト・メニューで[Select Function]を選択し、[alarm]を選択します。プログラムを実行してテストします。UserFunction alarmの詳細ビューは図242のように表示されます。

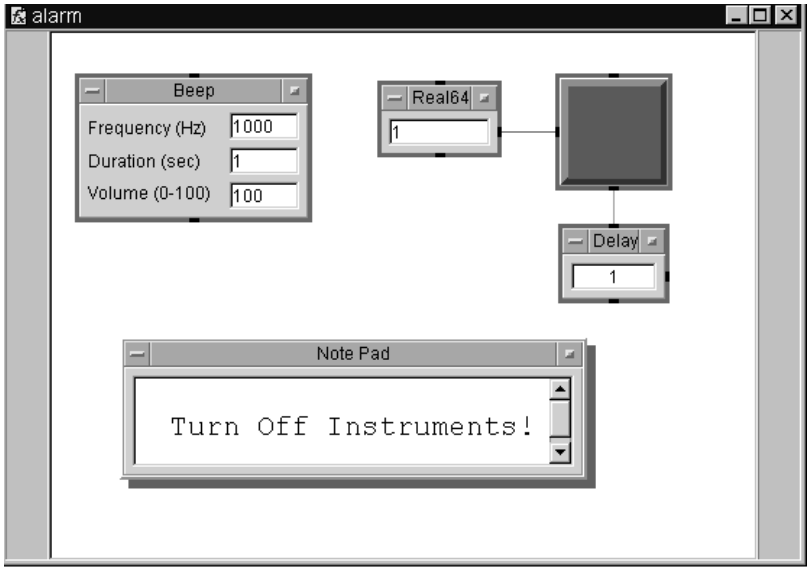


図242　UserFunction alarm(詳細ビュー)

9 alarmウィンドウに戻ります。[Color Alarm]表示と[Note Pad]を選択します。ポップアップの[Edit]メニューをオープンし、[Add To Panel]を選択します。パネル・ビューで、オブジェクト・ハンドルと[Align]関数を使用して、またはオブジェクトを移動することにより、オブジェクトのサイズと配置を調整します。グリッドへのスナップが必要な場合は、[PanelDrawGrid]プロパティをTrueに設定します。Note Padのオブジェクト・メニューをオープンし、[Properties]をクリックします。次のように設定します。

表47　[Notepad]のプロパティ

プロパティ	説明
ShowTitleBar	False
EditEnabled	False
Border	`[Raised border]`に変更します。

10 [Color Alarm]も[Raised Border]に変更します。

11 UserFunction 内の空き領域をクリックして、[Properties] ダイアログ・ボックスを表示し、[ShowPanelonExecute]を選択します。[ShowTitleBar]プロパティをFalseに設定します。[Panel Title]を`[alarm]`に変更します。alarmをアイコン化します。

12 メイン・ウィンドウに移動し、Callオブジェクトを削除します。VEEはまだ関数alarmをメモリに保持しています。この関数をもう一度編集する必要がある場合は、**[Edit]** ⇒ **[Edit UserFunction]**を選択するか、アイコンをダブルクリックします。

13 alarm関数を繰り返し呼び出す関数を作成します。

14 [Device] ⇒ **[UserFunction]**を選択し、UserFunctionの名前を`warning`に変更します。

15 [Flow] ⇒ **[Repeat]** ⇒ **[Until Break]**を選択します。

16 [Device] ⇒ **[Call]**を選択し、Function Nameをalarmに変更し、シーケンス入力ピンをUntil Breakのデータ出力ピンに接続します。

alarmをオフにする必要があるかどうかをオペレータに尋ねるためのCheck Boxオブジェクトを追加します。

17 [Data] ⇒ **[Toggle Control]** ⇒ **[Check Box]**を選択します。[Check Box Properties]ボックスをオープンし、[Title]プロパティを`Turn off alarm?`に変更し、[Scaled]プロパティを選択し、それをTrueに設定します。[InitializeAtPreRun]を選択して、プロパティ値が**[False]**であることを確認し、名前の[TitleFont]プロパティのサイズを大きくします。Callのシーケンス出力ピンをCheck Boxのシーケンス入力ピンに接続します。

これにより、Check Boxを使用する入力オブジェクトが作成されます。オペレータがこのボックスをクリックした場合、**X**が表示され、1が出力されます。そうでない場合は 0 が出力されます。この出力をIf/Then/Elseオブジェクトでテストすると、次に何を行うかをVEEに指示できます。

18 [Flow] ⇒ **[If/Then/Else]**を選択し、Toggleの右に配置します。Toggleのデータ出力を**If/Then/Else**オブジェクトのデータ入力**A**に接続します。**If/Then/Else**オブジェクトの式を編集して、`a == 1`にします。「**に等しい**」を意味する記号は`==`です。`=`ではありません。端子**A**が`1`を保持している場合は、**Then**出力が起動され、そうでない場合は、**Else**出力が起動されます。

Toggleの出力をIf/Then/Elseの入力に接続します。

19 [Flow] ⇒ **[Repeat]** ⇒ **[Break]**を選択し、図243に示すように、**If/Then/Else**オブジェクトの**Then**出力に接続します。

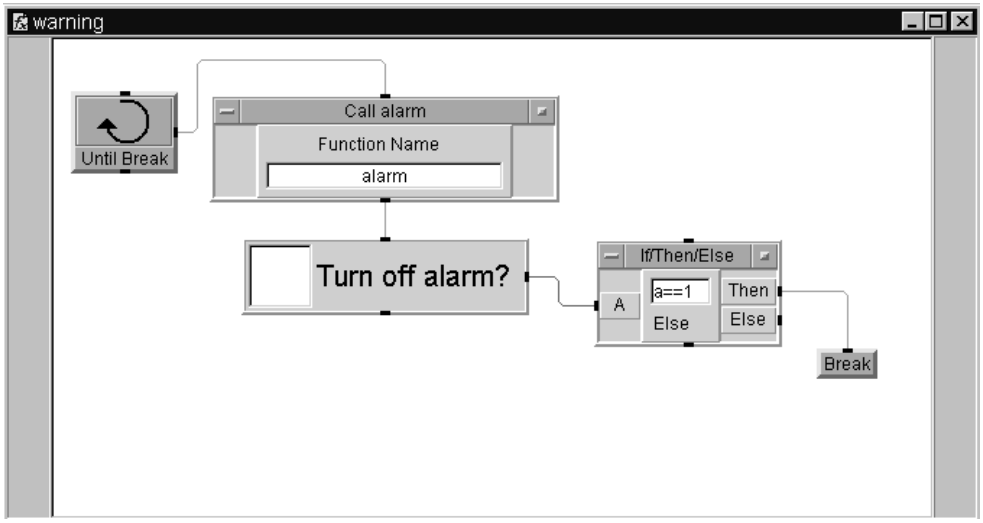


図243　Warning UserFunction(**詳細ビュー**)

20 Check Boxオブジェクト(**Turn off alarm?**)の右側をクリックして選択します。ポップアップの[Edit]メニューをオープンし、[Add To Panel]を選択します。Check Boxを囲む配置になるようにパネル・ビューのサイズを調整します。

21 warning UserFunction Propertiesボックスをオープンし、[ShowPanelonExecute]を選択し、[ShowPopupPanelTitle]をFalseに設定します。タイトルはオペレータにとって意味がないためです。

22 メインに移動し、**[Device]** ⇒ **[Call]**をクリックし、オブジェクト・メニューをオープンし、[Select Function]をクリックしてから、[warning]を選択します。Callオブジェクトを画面の上部中央に移動します。メイン・ウィンドウをアイコン化します。

23 デフォルトでは、alarmとwarningの両方のパネルが画面の中央に表示されるので、alarmは、alarmを停止するためのチェックボックスの上で点滅します。画面上の位置は両方ともロックされないので、ポップアップ・パネルを新しい場所にクリック・アンド・ドラッグすると、画面上の位置を変更できます。ただし、alarmパネルが点滅していると、位置が変更しづらくなります。その代わりに、パネルの端をクリック・アンド・ドラッグします。必要であれば、ツールバーのstopボタンを使用して、プログラムを停止します。プログラムを実行します。

2つのパネルを図244に示すように配置させた場合は、**Turn off alarm?**プロンプトの横のボックスをクリックするとプログラムを停止できます。

図244　　Warningプログラム

例題11-4: ActiveXコントロールの使用方法

この例題では、VEE内でのActiveXコントロールの使用方法を示します。ほかのアプリケーションのActiveXコントロールをVEEプログラムに組み込むことができます。この例題では、ProgressBarコントロールを組み込み、ループを使用して、進行状況表示バーを0%から100%完了まで表示します。この原則はほかのActiveXコントロールにも当てはまります。

1. **[Device]** ⇒ **[ActiveX Control References...]**をクリックし、[Microsoft Windows Common Controls 6.0]を選択します。**[OK]**をクリックします。
2. **[Device]** ⇒ **[ActiveX Controls]** ⇒ **[ProgressBar]**をクリックします。ProgressBarオブジェクトのサイズを少し大きくします。オブジェクト・メニューをオープンし、オブジェクト名がProgressBarであることに注目します。ActiveXコントロール・オブジェクトを参照するように宣言された変数は自動的に作成されます。ほかの変数やデータ入力と同じように、ProgressBarという名前をFormula式の中で使用できます。

3 **[Device]** ⇒ **[Formula & Object Browser]**をクリックし、[ActiveX Objects]、[Library: MSComcltLib]、[Class: ProgressBar]、[Members: Value]を選択し、**[Create Set Formula]**をクリックします。オブジェクトをメイン・ウィンドウの上部中央に配置します。

4 0から100までループさせ、完了済みの割合をパーセントで表示させるために、For Rangeオブジェクトを追加します。**[Flow]** ⇒ **[Repeat]** ⇒ **[For Range]**を選択し、ProgressBarの下に配置し、`[From:]`を`[0]`に、`[Thru:]`を`[100]`に、`[Step:]`を`[10]`に設定します。For Rangeの出力をProgressBarの入力端子Valueに接続します。

5 ProgressBarが更新されていくのを確認できるように、プログラムの実行速度を遅くするには、**[Flow]** ⇒ **[Delay]**を選択し、**For Range**オブジェクトの右に配置します。`.2`に設定します。図245に示すように、ProgressBarのシーケンス出力ピンをDelayオブジェクトのシーケンス入力ピンに接続し、プログラムを実行します。

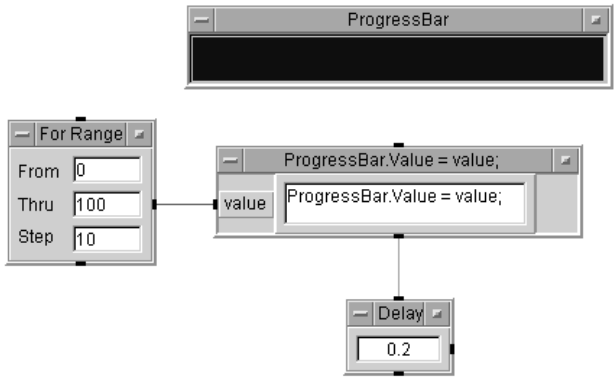


図245　ActiveXコントロールProgressBarの使用方法

注 記

ActiveXコントロールのオブジェクト・メニューには[Properties]と[Control Properties]があります。[Properties]メニューではオブジェクトのVEEプロパティを設定します。[Control Properties]は、ActiveXコントロールが提供するもので、ActiveXコントロールの種類ごとに異なる可能性があります。

VEEとともに出荷されるコントロールの動作をよりよく理解するために、すべてを調べてください。さらに、VEEでのユーザ・インタフェース機能を向上させるために必要な市販のコントロールと表示を探してください。

図246は、MS Chartからコントロールを組み込んだVEEの例を示します。**[Device] ⇒ [ActiveX Controls References]**ダイアログ・ボックスでコントロール・ライブラリを選択したら、[Function & Object Browser]または[Declare Variable Object]を使用して、コントロールのプロパティとメソッドを識別できます。

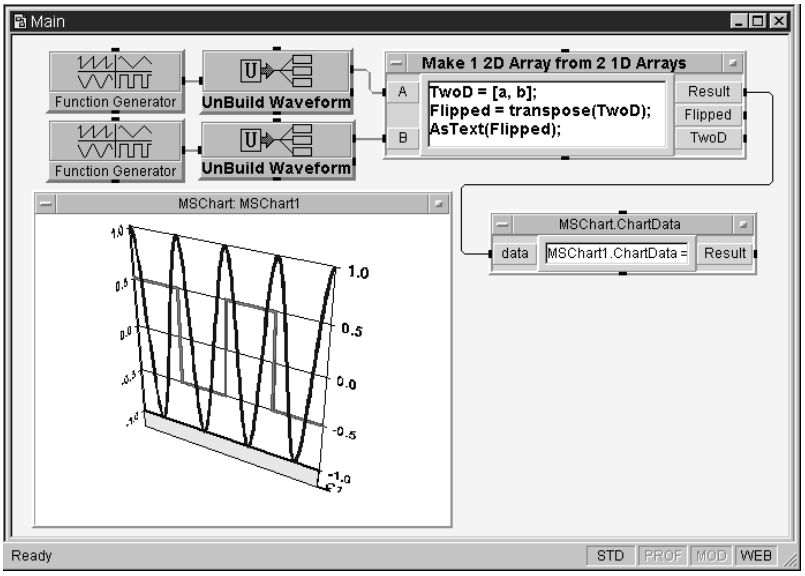


図246　　MSChartを使用するActiveXコントロールの例

例題11-5: ステータス・パネルを作成する方法

この例題では、サイコロのプログラムからの関数を使用して、ステータス・パネルを作成する方法を習得します。サイコロのプログラムの例題は419ページの「例題11-1: メニューを使用する方法」にあります。ステータス・パネルは通常、多数のテストがあり、返ってきた結果を表示した

いときにSequencerオブジェクトといっしょに使用します。関数**random()**を使用します。**random()**は、デフォルト設定を使用する場合、**0**から**1**の間の実数値を返します。

1 **[Device]** ⇒ **[Sequencer]**をクリックします。トランザクション・バーをダブルクリックし、デフォルト名**test1**を使い、**[FUNCTION:]**フィールドを**random()**に置き換えることで、テストを設定します。図247を参照してください。

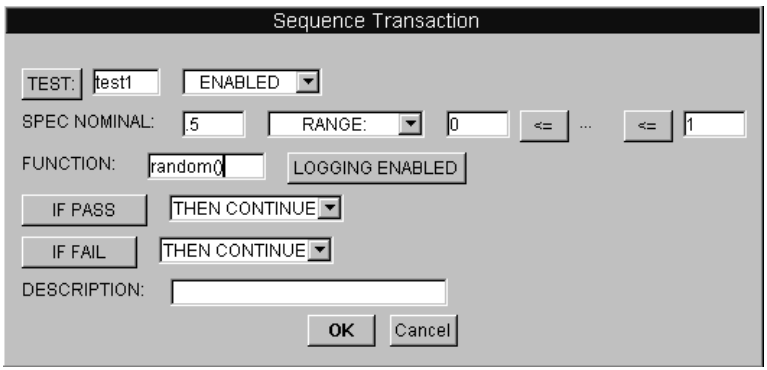


図247　　Test1の設定

2 test2という名前で2番目のテストも同様に設定します。

3 [Sequencer Properties]ボックスをオープンし、[Logging]タブを選択し、[Logging Mode]で[Log Each Transaction To: logTest(thisTest)]を選択してから、**[OK]**をクリックします。

Sequencerが各トランザクションを実行すると、各テストに対してthisTestと呼ばれるレコードが作成されます。レコードのフィールドは、同じタブで設定できます。次にlogTestまたは別の名前を付けたUserFunctionを作成できます。Sequencerは、実行された各トランザクションの最後にレコードthisTestを持つ、LogTest() UserFunctionを呼び出します。この方法で、ステータス・パネルを更新することができます。

4 **[Device]** ⇒ **[Function & Object Browser]** ⇒ **[Built-in Functions]** ⇒ **[Panel]** ⇒ **[showPanel]**を選択し、それをSequencerの上に配置します。入力ピンを削除したら、パラメータを`"logTest",420,180`に編集し、最後の2つのパラメータをそのままにしておきます。showPanelからのResult出力ピンをSequencerのシーケンス入力ピンに接続します。成功すると、showPanelは**1**を出力します。

LogTestは、UserFunctionの名前です。他の2つのパラメータは、画面の左上隅を基準としたXおよびY座標です。これにより、VEEに表示する際のUserFunctionパネルの配置場所を指示しています。パネルの寸法は、この例には含まれていません。

5 下図に示すように、logTestという名前のUserFunctionを作成します。入力ピンを追加します。ログとして記録されたRecordが入力となります。Logging AlphaNumericをパネル上に配置し、それを入力ピンに接続します。Logging AlphaNumericを選択し、**[Edit]** ⇒ **[Add to Panel]**をクリックします。パネル・ビューで、サイズと配置場所を調整します。表示とパネルのタイトル・バーの選択を解除します。

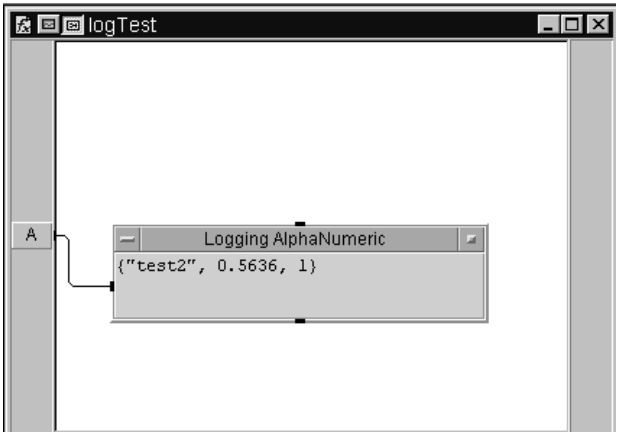


図248　　UserFunction LogTest(詳細)

図249に、プログラム実行後のパネル・ビューを示します。この図から、データがどのように表示されるかがわかります。

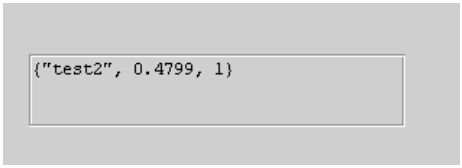


図249　　UserFunction LogTest(パネル)

6 [Function & Object Browser]ボックスからhidePanelを選択し、**[Flow]** ⇒ **[Comfirm(OK)]**を選択します。hidePanel()パラメータをlogTestに変更します。入力ピンを削除します。図250に示すようにオブジェクトを接続します。

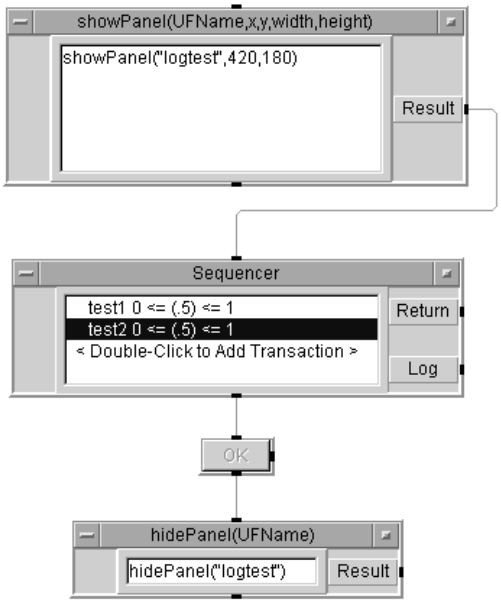


図250　　ステータス・パネル・プログラム(実行前)

7 プログラムを実行します。図251のように表示されます。

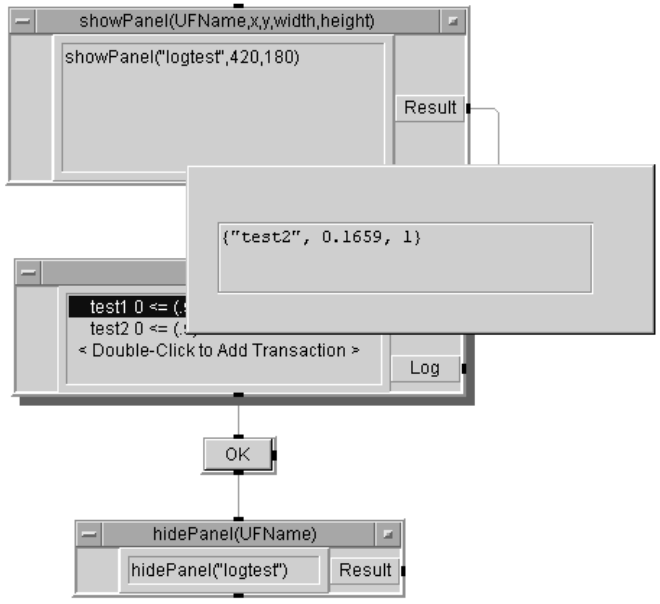


図251　　ステータス・パネル・プログラム(実行中)

要約すると、showPanelオブジェクトはUserFunctionパネルを表示しますが、UserFunctionを呼び出しません。Sequencerは、logging関数を介してUserFunctionを2回呼び出し、各呼び出しによってパネルを更新します。次に、オペレータが[OK]を押すと、パネルが非表示になります。この例では、OKオブジェクトを使用してhidePanelオブジェクトをトリガしますが、実行時間を計るため、プログラムの別の場所に配置することもできます。プログラムをステップ実行して、ステータス・パネルの表示を確認し、Test1とTest2の結果で更新してから非表示にします。

この章の復習

この章では、次の操作について学びました。次の章に進む前に、必要に応じて復習してください。

- オペレータ・インタフェースに関する重要点を要約できる。
- メニューを使用して、オペレータ・インタフェースにあるテストを選択する。
- オペレータ・インタフェースのためのビットマップをインポートする。
- ステータス・パネルを作成する。
- VEEが提供するオペレータ・インタフェース機能の一部を挙げることができる。
- プログラムの安全を保護する。
- インパクトの強い警告を作成する。
- ActiveXコントロールを作成し、Function & Object Browserでプロパティとメソッドを検索する。

12
Agilent VEEプログラムの最適化

Agilent VEEプログラムの最適化

この章の内容

- プログラムを最適化するための基本的な手法
- ダイナミック・リンク・ライブラリ(DLL)の使用法
- コンパイル済みの関数を使った最適化
- VEEコンパイラの使用法
- VEEプロファイラの使用法

平均所要時間: 2時間

概要

この章では、VEEプログラムの実行速度を向上させる方法を習得します。テスト・プログラムのパフォーマンスには、3つの基本要素があります。つまり、測定の速度、データがコンピュータに転送されるレート、プログラムがデータを処理する速度です。VEEプログラムを最適化することにより、処理速度を向上させることができます。

最初のセクションでは、VEEプログラムを最適化するための基本原則を習得します。ダイナミック・リンク・ライブラリ(DLL)についても学びます。次のセクションでは、コンパイル済みの関数を使って最適化する方法について説明します。次に、VEEコンパイラの概要について説明します。最後のセクションで、VEEプロファイラについて説明します。

注 記　**この章で説明されている手法は、コンパイラを使用する場合にも使用しない場合にも適用されます。**

プログラムを最適化するための基本的な手法

VEEプログラムを最適化する場合は、このセクションの情報を参照してください。このセクションに説明されている手法を使用すると、VEEに最適なプログラミング習慣を身に付けることができます。

できるだけ配列に対して演算を実行する

配列に対して演算を行うと、プログラムのパフォーマンスが大幅に向上します。たとえば、取得した測定値の平方根を求めるテストがあるとします。通常のプログラム方法では、ループの内で1つずつ測定値を受け取り、平方根を計算します。VEEでは、そうした方法の代わりに、すべての測定値を配列に格納し、1ステップでその平方根を計算できます。

図252のプログラムは、1024回の反復を行います。各反復で1つの平方根が計算されます。

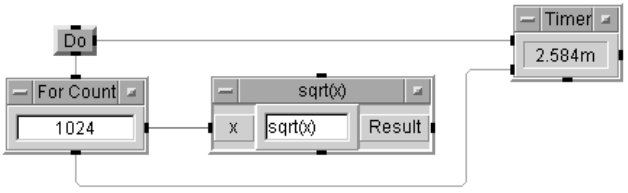


図252　　測定値ごとに平方根を計算する

図253のプログラムは、1024個の要素がある配列を作成し、配列の平方根を計算します。これで、それらの平方根を要素とする配列が生成されます。2つのプログラムの結果は同じですが、図253のプログラムは、図252のプログラムより6倍速く実行されます。この例では、Pavilion PC(300MHz)を使用しています。

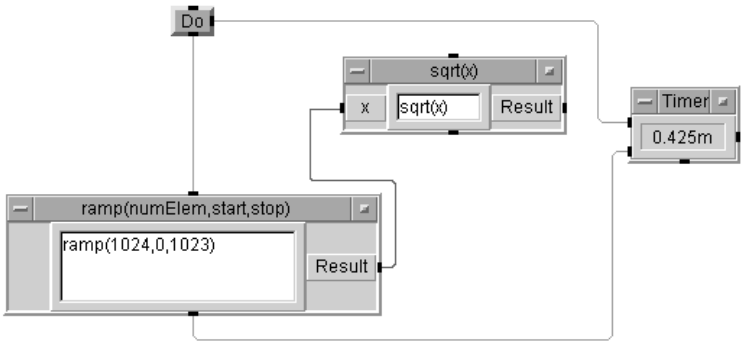


図253　　配列演算を使って平方根を計算する

2つのプログラムの実行速度の違いは、オブジェクトの実行に必要な時間によるものです。オブジェクトが実行されるときのオーバーヘッドの大きさは決まっています。したがって、スカラ値のかわりに配列を使用して、オブジェクトの実行回数を減らすと、プログラムの実行速度が上がります。

どちらのプログラムでも、時間を計測するには、Doオブジェクトを使って最初にタイマを起動するのがよい方法です。ramp関数は、0から始まり1023で終わる1024個の要素を持つ配列を生成します。

注 記

高速に実行するには、VEEの最新の実行モードを使用しているかどうかを必ず確認してください。それには、[File] ⇒ [Default Preferences]をクリックするか、ツールバーにあるそのボタンを使用します。[Execution Mode]の[VEE6]**を選択し、[Save]をクリックします。VEEウィンドウの下部にあるステータス・バーに、"VEE6"と表示されます。**

できるだけオブジェクトをアイコン化する

VEEが画面上で管理する情報が増えるほど、プログラムの実行に要する時間も増えます。プログラムを最適化するには、表示内容が更新されるオブジェクト(Counterなど)にはオープン・ビューを使用せず、アイコン・ビューを使用します。図254の例では、For CountとCounterの各オブジェクトにアイコン・ビューを使用すると、約46倍速く動作します。

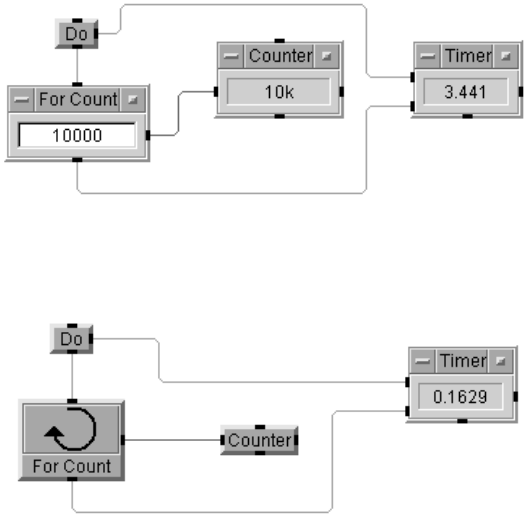


図254　　アイコンを使ったプログラムの最適化

プログラム内のオブジェクトの数を減らす

プログラミングの経験が増すにつれて、VEEプログラム内で使用するオブジェクトの数は少なくなります。オブジェクトの数を減らして、プログラムを最適化するには、さらに2つの方法があります。

1. 別々の数学オブジェクトを使用するかわりに、Formula オブジェクトで単一の計算式を使用します。たとえば、加算、乗算、除算ごとに別々のオブジェクトを使用するのではなく、**Formula**オブジェクトに式「`((a + b) * c)/d`」を入力します。また、定数オブジェクトを入力に接続するかわりに、式の中で定数を使用します。定数は、**Set Variable**を使って設定します。
2. 関数のパラメータ・リスト内に別の関数呼び出しをネストします。たとえば、図255の関数randomizeは、関数rampによって生成された配列を使用しています。図256では、ramp関数呼び出しがrandomize呼び出しの中にネストされているため、プログラムの実行が少し速くなります。

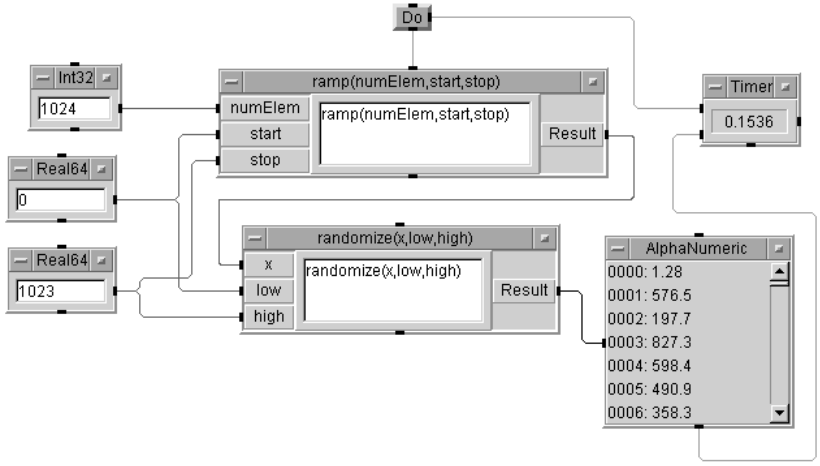


図255　　最適化されていない関数呼び出し

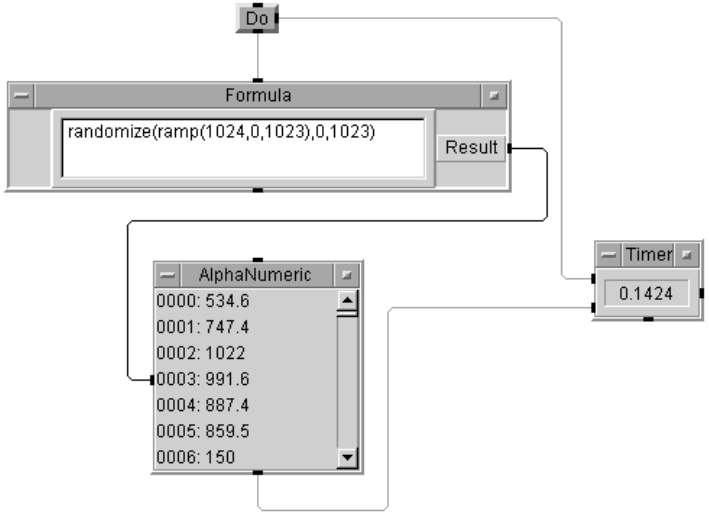


図256　　最適化されている関数呼び出し

VEEプログラムを最適化するその他の方法

ほかに次のような最適化方法があり、適宜プログラム内で使用できます。

- VEE 4以上の実行モードでプログラムを実行することにより、必ずVEEコンパイラを使用します。詳細は、460ページの「Agilent VEEの実行モード」を参照してください。
- 詳細ビューでなく、パネル・ビューからプログラムを実行します。そうすると、VEEが画面上で管理するオブジェクトの数が減ります。
- UserObjectやUserFunctionの間では、値(特に大きな配列やレコード)の受渡しを行うかわりに、グローバル変数を使用します。すべてのグローバル変数を宣言します。これにより、ローカル変数も使用できます。**[Data]** ⇒ **[Variable]** ⇒ **[Declare Variable]**を参照してください。

- グラフィカル表示用のデータを収集したら、個々のスカラ点をプロットするのではなく、配列全体を一度にプロットします。プロット点のX値が一定の間隔を持つ場合は、X vs. Yプロットではなく、XY Trace表示を使用します。
- 複数のIf/Then/Elseオブジェクトを使用するかわりに、複数の条件を持つ単一のIf/Then/Elseオブジェクトを使用します。
- グラフィカル表示の設定は、できるだけ単純にします。更新速度を最大にする設定では、**[Grid Type]**を**[None]**にし、[Properties]ダイアログ・ボックスで何もチェックしません。必要な場所にだけAutoScaleコントロール・ピンを使用し、必要でない場合は、Scalesフォルダの[Automatic AutoScaling]をオフにします。
- ファイルからデータを読み取る場合は、一度に1つの要素に対してREADトランザクションを実行し、EOFピンを使用するのではなく、**ARRAY 1D TO END: (*)**トランザクションを使用します。
- 複数のUserFunctionの実行フローを制御する場合は、個別のCallオブジェクトを使用するかわりにSequencerを使用します。
- Sequencerを使用する場合、ログとしてレコードを記録する必要があるトランザクションのログ機能だけをオンにします。
- Strip ChartsとLogging AlphaNumeric表示を使用する場合は、[Properties]の[Buffer Size]をアプリケーションにとっての最小値に設定します。
- If/Then/ElseオブジェクトをGateやJunctionと組み合わせるかわりに、3項演算子*(condition ? expression1 : expression2)*を使用します。
- ビットマップを使用する場合は、[Scaled] ではなく、[Actual] または[Centered]に設定します。[Scaled]の方が少し時間がかかります。
- Fill BarやThermometerなどのインジケータを使用する場合は、[Show Digital Display]をオフにします。
- Color Alarmを使用している場合、色をすばやく切り替えるときには、[Show 3D Border]をオフにします。

以上のような手法のほかに、ほかの言語で記述されたコンパイル済みの関数を自分のプログラムにリンクすると、実行速度が向上します。PCでコンパイル済み関数を(DLL)としてを使用する方法については、次のセクションで説明します。

コンパイル済み関数の概要

VEEプログラム内で、DLL(ダイナミック・リンク・ライブラリ)などのコンパイル済み関数を使用できます。それには、コンパイル済み関数を入手するか、次の手順にしたがって作成する必要があります。

1. C、C++、Fortran、またはPascalで関数を作成し、コンパイルする。
2. 関数の定義ファイルを作成する。
3. コンパイル済み関数を保持する共有ライブラリを作成する。

コンパイル済み関数を使用する利点

VEEプログラムでコンパイル済み関数を使用することには、次の利点があります。

- 実行速度が上がる。
- 現在のテスト・プログラムをほかの言語で利用する。
- ほかの言語でデータ・フィルタを開発し、それをVEEプログラムに統合する。
- 著作権のあるルーチンを保護する。

注 記

コンパイル済み関数を追加すると、開発プロセスが複雑になります。したがって、VEE UserFunction、オペレーティング システムでのプログラム実行、ActiveXオートメーション(別のプログラムの呼び出し)を使用しても、必要な機能やパフォーマンスを得ることができない場合に*のみ*、コンパイル済み関数を使用してください。

コンパイル済み関数の使用における設計上の留意点

VEEプログラムでコンパイル済み関数を使用する場合は、次の情報を考慮に入れてください。

- 演算ルーチン、測定I/Oなど、オペレーティング・システムで使用できる任意の機能を使用できます。ただし、リンク先のプログラム内からの内部機能にアクセスすることはできません。

- VEEは、外部ルーチン内のエラーを捕捉できないため、コンパイル済み関数内にエラー検出機能を用意する必要があります。
- 外部ルーチン内で割り当てたメモリは、すべて解放する必要があります。
- 外部ルーチンにデータを渡す場合は、Callオブジェクトの入力端子をそのルーチンに必要なデータの型と種類に設定する必要があります。
- システムI/Oのリソースはロックされる可能性があるため、外部ルーチンは、このような状態に対応できる必要があります。
- 外部ルーチンが配列を受け取る場合は、検査するデータ型に対して有効なポインタを持つ必要があります。また、配列のサイズをチェックする必要があります。ルーチンで配列のサイズを変更した場合は、新しいサイズをVEEプログラムに戻す必要があります。
- コンパイル済み関数では、最後のステートメントとして、exit() ではなく、return()ステートメントを使用する必要があります。コンパイル済み関数はVEEにリンクされるため、コンパイル済み関数が終了すると、VEEも終了します。
- 配列の範囲を超えて書き込んだ場合の結果は、使用している言語によって異なります。Pascalでは、範囲検査が行われるため、実行時エラーとなり、VEEは停止します。Cなどの言語では、範囲検査が行われないため、結果は予測できません。断続的にデータが損傷したり、VEEがクラッシュする可能性があります。

コンパイル済み関数を使用する際のガイドライン

VEEプログラムでコンパイル済み関数を使用する場合は、次のガイドラインに従ってください。

- コンパイル済み関数の呼び出しと設定は、UserFunctionを呼び出す場合とまったく同じです。目的の関数を選択する場合は、Callのオブジェクト・メニューで[Select Function]を使用するか、関数の名前を入力します。どちらの場合も、VEEがその関数を認識すると、Call Functionオブジェクトの入力端子と出力端子が自動的に設定されます。必要な情報は定義ファイルから提供されます。ライブラリがインポート済みであれば、VEEは定義ファイルを認識します。

- オブジェクト・メニューの[Configure Pinout]を選択して、Callの入力端子と出力端子を再設定します。どちらの方法の場合も、VEEは、関数に必要な入力端子と、関数の戻り値用のRet Value出力端子を使用して、Callオブジェクトを設定します。さらに、参照によって渡される入力のそれぞれに対応する出力端子が設定されます。
- Formula オブジェクト内の式、または実行時に評価されるほかの式から、コンパイル済み関数を名前で呼び出します。たとえば、To Fileトランザクションの式の中にコンパイル済み関数の名前を入れると、その関数を呼び出すことができます。

注 記

式の中からは、コンパイル済み関数の戻り値(CallオブジェクトのRet Value)だけを取得できます。関数から返されるほかのパラメータを取得する場合は、Callオブジェクトを使用する必要があります。

- コンパイル済み関数のライブラリを削除するには、[Device]メニューのDelete Libraryオブジェクトを使用します。Import Library、Call、Delete Libraryの各オブジェクトを使用して、コンパイル済み関数をインポートし、プログラムがコンパイル済み関数の呼び出しを完了したときにそれらを削除することにより、プログラムのロード時間を短縮し、メモリの浪費を防ぐことができます。

ダイナミック・リンク・ライブラリの使用法

PCでは、ダイナミック・リンク・ライブラリ(DLL)のコンパイル済み関数をプログラムの一部として使用できます。DLLは、独自に記述したコンパイル済み関数、または購入したりWebからダウンロードしたDLLです。DLLの作成方法については、Microsoftにお問い合わせください。

注 記

VEEは、"_cdecl"と"_stdcall"の両方の呼び出し規則をサポートします。カスタマ作成DLLのほとんどは、_cdec1呼び出し規則を使用しています。Win32 API呼び出しのほとんどは、_stdcall呼び出し規則を使用しています。VEE は両方の命名規則をサポートしているため、独自のDLLだけでなく、ほとんどの既製のDLLも使用できます。

DLLをAgilent VEEプログラムに統合する方法

このセクションでは、VEEプログラムにDLLをインポートする方法について説明します。上で説明したようにDLLを作成または入手した後、次の手順に従ってDLLを使用します。

1 **[Device]** ⇒ **[Import Library]**を選択します。

[Library Type] には、[UserFunction]、[Compiled Function]、[Remote Function] の3 つの選択肢があります。[Library Type] フィールドを[Compiled Function]に変更します。デフォルトはUserFunctionです。図257に示すように、[Compiled Function]を選択した場合、Import Libraryオブジェクトには[Definition File]のフィールドがあります。

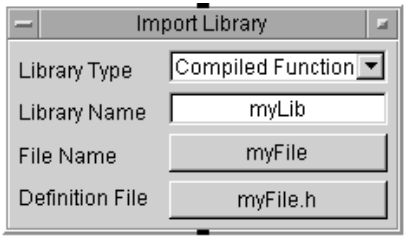


図257　　コンパイル済み関数のライブラリのインポート

次のフィールドがあります。

表48　Import Libraryのフィールド

フィールド	説明
[Library Name]	VEEがライブラリの識別に使用する名前。通常、この名前は、プログラムでライブラリを使用した後、それを削除するために使用されます。
[File Name]	共有ライブラリを保持するファイル。
[Definition File]	関数のプロトタイプが記述されたインクルード・ファイル。通常は、*.hファイルです。

注 記

開発段階では、オブジェクト・メニューの*[Load Lib]*を選択して、ライブラリを手動でロードすることもできます。

2　[Device] ⇒ [Call]を選択します。

[Import Library]を使ってライブラリをインポートしたら、**[Device] ⇒ [Call]**を選択して**Call**オブジェクトを作成します。次に、Callのオブジェクト・メニューで[Select Function]を選択し、表示されるリスト・ボックスで目的の関数を選択すると、コンパイル済み関数を呼び出すことができます。たとえば、図258に示したCallオブジェクトは、arraySizeとarrayをパラメータに使用し、**myLibrary**にあるmyFunctionという名前のコンパイル済み関数を呼び出します。

図258　コンパイル済み関数のCallオブジェクト

VEEは、関数名および適切な数の入力ピンと出力ピンを使用して、自動的にCallオブジェクトを設定します。2番目以降の出力ピンは、その関数に参照によって渡されるパラメータにマッピングされます。関数名を入力した場合は、オブジェクト・メニューの[Configure Pinout]を選択してCallオブジェクトを設定することもできます。

注 記

ライブラリがすでにロードされている場合は、式フィールドからDLL関数を呼び出すこともできます。このように関数を使用する場合は、関数名の後のパラメータを丸かっこで囲む必要があり、関数は戻り値だけを返します。参照によって渡されるパラメータは、Callオブジェクトを使用する場合にのみ取得できます。たとえば、Formulaオブジェクト内で次の式を使用するとします。

```
2 * yourFunc(a,b)
```

aとbはFormulaオブジェクトの2つの入力ピンを参照し、yourFuncの戻り値の2倍の値が出力ピンに出力されます。

3 (省略可)**[Device]** ⇒ **[Delete Library]**をクリックします。

プログラムの開発中は、オブジェクト・メニューの**[Delete Lib]**を選択して、プログラムからライブラリを削除することもできます。プログラムで使用した後にライブラリを削除すると、ロード時間を短縮し、メモリの浪費を防ぐことができます。

DLLの使用例

この練習では、DLLをインポートし、そのDLL内の関数を呼び出します。使用するDLLは、WindowsのVEE製品に付属するものです。すべてのプラットフォームで同じVEEプログラムが動作するように設計されています。

manual49.veeファイルを開きます。このファイルは次の場所にあります。

<**インストール先ディレクトリ**>\EXAMPLES\MANUAL\MANUAL49

この例を詳細に検討してください。図259のように表示されます。

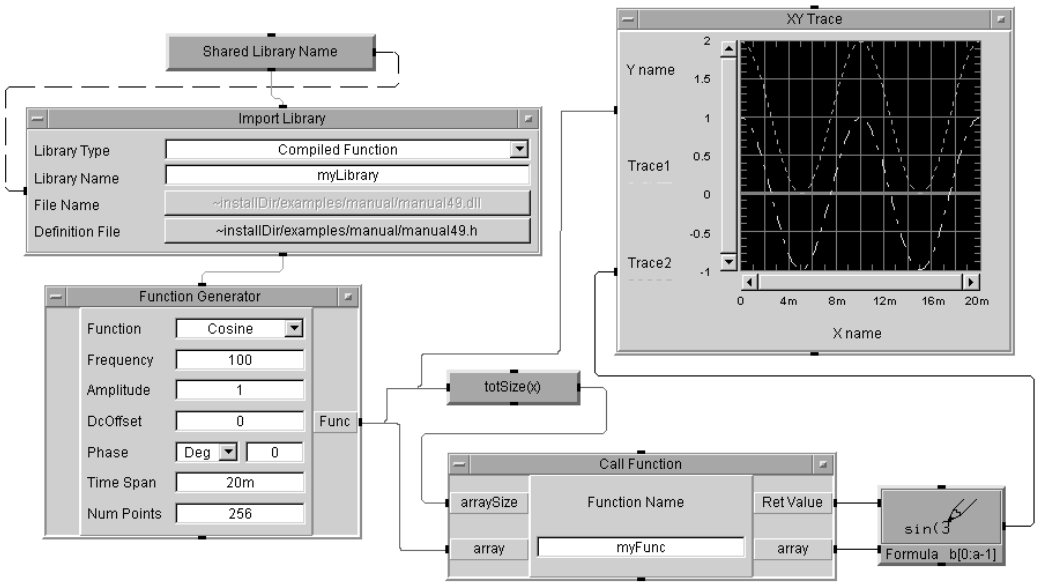


図259　　DLLを使ったプログラム(MANUAL49)

表49　　図259の要素

要素	説明
Import Library	コンパイル済み関数Call Functionを最初に呼び出す前に、Import Libraryオブジェクト([Device]メニュー)を使用して、DLLをロードする必要があります。
Call Function	**MANUAL49**は、myFuncという名前のコンパイル済み関数を呼び出します。MyFuncは、VEEのInt32と同じCのデータ型longを必要とします。この数は配列のサイズを指定するものです。2番目の入力パラメータは、実数の配列へのポインタです。定義ファイルは**MANUAL49.H**、Cコードのソース・ファイルは**MANUAL49.C**にあります。MyFuncは、配列のすべての要素に1を加算します。
Function Generator	Function Generatorは波形の作成に使用され、波形はCall myFuncオブジェクトのarrayピンに出力されます。

表49　　図259の要素

要素	説明
totSize	**totSize**オブジェクト(**[Math & Functions]**ダイアログ・ボックス)は、波形のサイズを決めるために使用され、**Call myFunc**のarraySize入力ピンに出力されます。
XY Trace	**XY Trace**オブジェクトは、元の波形と新しい波形の両方を表示します。
Formula	**"Ret Value"というラベルが付いたCallオブジェクトの出力ピン**は、返された配列のサイズを保持するので、Formulaオブジェクト内の式**B[0:A-1]**は、この配列を正しく表示オブジェクトに指定します。

プログラムを実行し、2番目のトレースが波形上のすべての点で1番目のトレースより大きいことを確認します。

このプログラムで注意しておく点の1つは、プログラムをすべてのプラットフォームで使用できるようにする方法です。Windows 98、Windows 2000、Windows NT 4.0、Windows XPでは、Microsoft 32ビット・コンパイラが使用されます。これらのDLLは、すべて***.dll**拡張子で示されます。

Shared Library Nameという名前のUserObjectは、使用されるオペレーティング・システムを識別し、図260に示すように、正しいライブラリ名をImport Libraryオブジェクトに伝えます。

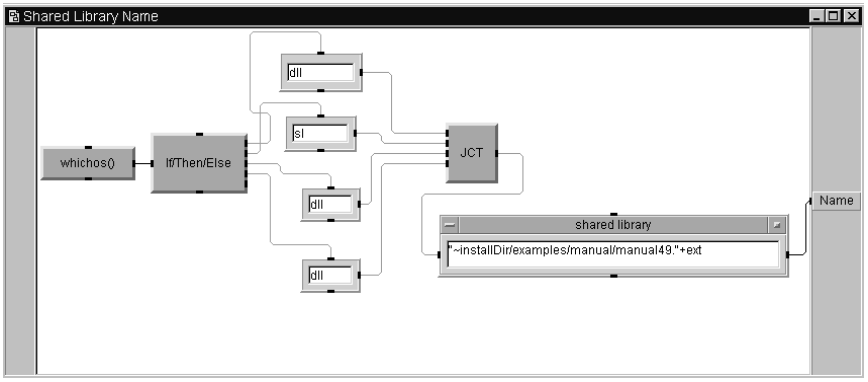


図260　Shared Library Name UserObject

whichos()関数は、オペレーティング・システムを識別するために、名前が変更されたFormulaオブジェクト内で使用されています。展開されたIf/Then/Elseオブジェクトは、whichos()関数の出力を検査し、適切なテキスト定数を出力します。次に、このファイル名拡張子は、名前が変更されたFormulaオブジェクトを使って**MANUAL49**ファイルに追加されます。"shared library"というラベルが付いたFormulaオブジェクトの入力端子も、"ext"に変更されています。

Import LibraryオブジェクトにFile Nameのコントロール・ピンが追加されています。このため、UserObjectとImport Libraryは点線で結ばれています。

注 記

データのデータベースへの送信など、混在する環境でデータを共有するためのTo/From Socketを調査します。サンプル・ゲームのBattleship(戦艦)やEuchre(ユーカー)で、複数のVEEプロセス間の通信におけるソケットの高度な使い方を紹介しています。

Execute Programオブジェクト対コンパイル済み関数

コンパイル済み言語プログラムをVEEと統合するため**Execute Program**オブジェクトとコンパイル済み関数のどちらを使用するか判断するときには、以下の情報を考慮してください。

Execute Programオブジェクト

Execute Programオブジェクトには次の特徴があります。

- 使いやすい
- 起動時間がより長い
- アドレス領域が保護されている
- 同期実行と非同期実行を選択できる
- 非同期イベントのサービス
- より安全(呼び出されたプログラムがクラッシュした場合、エラー・メッセージが表示されます)
- 連続したデータ収集に最適

コンパイル済み関数

Import Libraryオブジェクトと**Call**オブジェクトを使用するコンパイル済み関数には、次の特徴があります。

- 起動時間が短い
- VEEと共有するスタックとメモリ領域を伝えることにより通信を実行

同期実行

- 信号と例外は、ブロックまたは捕捉されません(GPFメッセージなど)。
- テキスト言語のコンパイラが必要です。
- 使い方がより複雑です。使用の際のリスクが高まります。境界のずれによる配列エラーやメモリの上書きによって、VEEがクラッシュします。

Agilent VEEの実行モード

Agilent VEEの実行モードを使用すると、以前のバージョンのVEEを使って作成されたプログラムを実行できます。実行モードを使用すると、古いバージョンのVEEを使って作成されたプログラムを古いバージョンのVEEで実行する場合とまったく同様に、新しいバージョンのVEEで実行できます。これにより、VEEは下位互換となるため、既存のプログラムがサポートされます。

注 記

実行モードは、以前のバージョンのVEEでは、「**互換モード**」と呼ばれていました。

VEEには次の４つの実行モードがあります。

- VEE 6 (新しいデータ型の追加)
- VEE 5 (ActiveXの追加)
- VEE 4 (コンパイル)
- VEE 3.*x*

図261に示すように、実行中のプログラムの実行モードは、VEEのステータス・バーに表示されます。

VEEに開かれている既存のプログラムは、デフォルトでは、そのプログラムの作成に使用されたバージョンのVEEの実行モードで実行されます。たとえば、VEE 6.0に開かれているVEE 5.0プログラムは、デフォルトではVEE 5実行モードで実行されます。

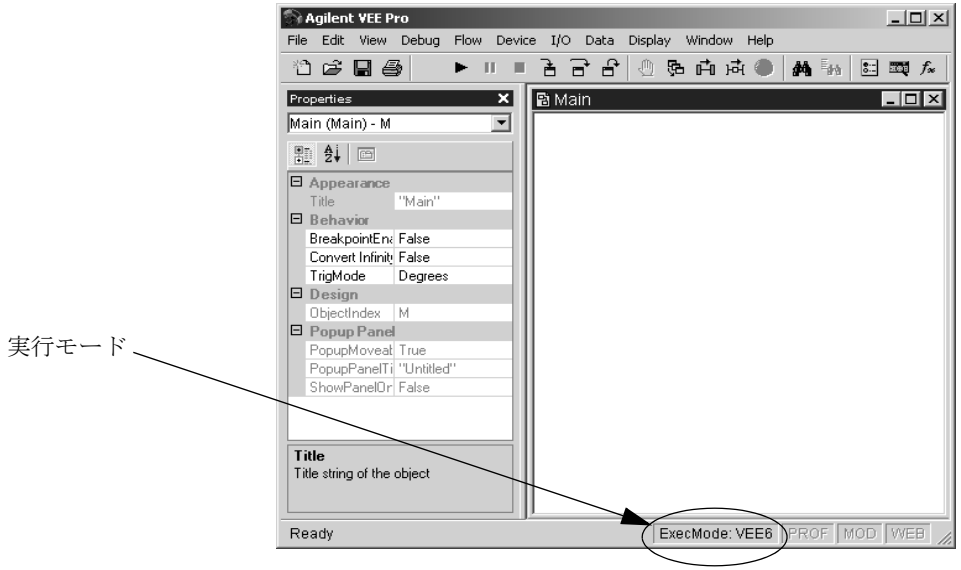


図261　　VEEのステータス・バーに表示される実行モード

Agilent VEEコンパイラ

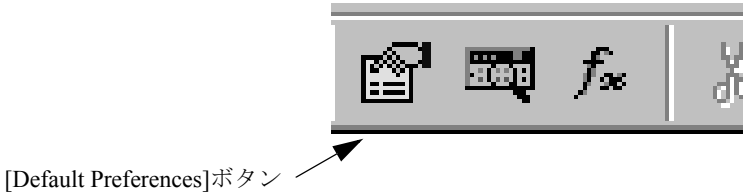
VEEコンパイラは、VEE 4以上の実行モードで自動的に有効になります。コンパイラは、高度なオブジェクト伝達を可能にし、プログラムの実行速度を上げます。コンパイラと実行モード間の相違についての詳細は、『*VEE Pro Advanced Techniques*』を参照してください。

実行モードの変更

新しいプログラムは、すべてVEE 6モードで作成する必要があります。既存のプログラムがあり、そのプログラムに新しい機能を追加する場合は、実行モードを変更します。たとえば、VEE 5.0で記述されたプログラムにの新機能を追加する場合は、実行モードをVEE 6に変更する必要があります。そうしないと、そのVEE 5.0プログラムは正しく実行されません。

実行モードを変更するには、次の手順に従います。

1 VEEのメイン・メニューで、**[File]** ⇒ **[Default Preferences]**をクリックするか、図に示すツールバーの [Default Preferences] ボタンを押します。



ツールバーの[Default Preferences]ボタン

2 図262に示すように、[General]フォルダの[Execution Mode]で[VEE 6.0]を選択します。このフォルダは最初のフォルダなので、すでに表示されています。同じフォルダで、[Disable Debug Features]が選択されて**いない**ことを確認します。**[OK]**をクリックします。

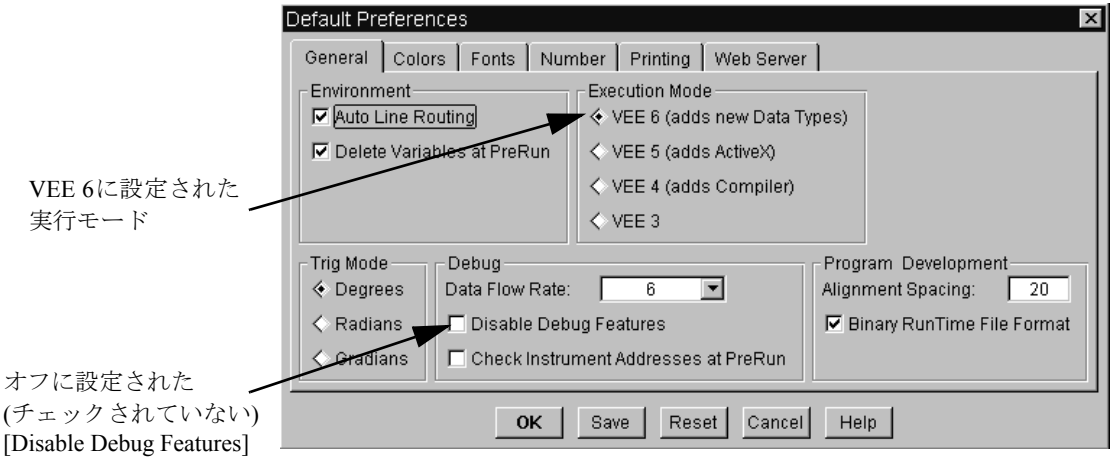


図262　[Default Preferences]ダイアログ・ボックスで実行モードを変更

実行モードの変更の効果

次の例は、プログラムを更新した場合の速度の向上を実証するものです。この例は、計測I/Oがない場合のプログラム速度に注目しています。

1 **examples\Applications** サブディレクトリの **chaos.vee** プログラムを開きます。

このプログラムは、爆発的な人口増加をシミュレートします。次に示すように、結果をチェックするためTimerオブジェクトを使用して、このプログラムを変更します。これらの例では、Pavilion PC (300MHz)を使い、Windows 98上でプログラムを実行しています。ほかに2つの大きなアプリケーションが同時に実行されています。

図263では、VEE 3実行モードで、表示を開き、プログラムの実行時間を測定します。

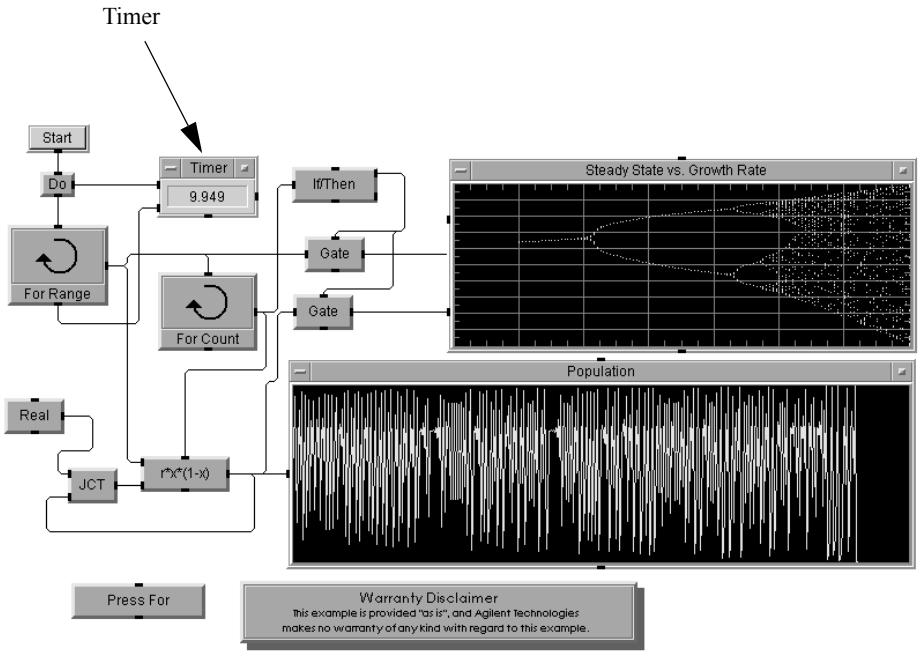


図263　　VEE 3モードで、表示を開いたChaos.vee

図264では、コンパイラをオンにしないまま、表示をアイコン化して速度を上げています。これにより、実行時間が約1/6短縮されます。

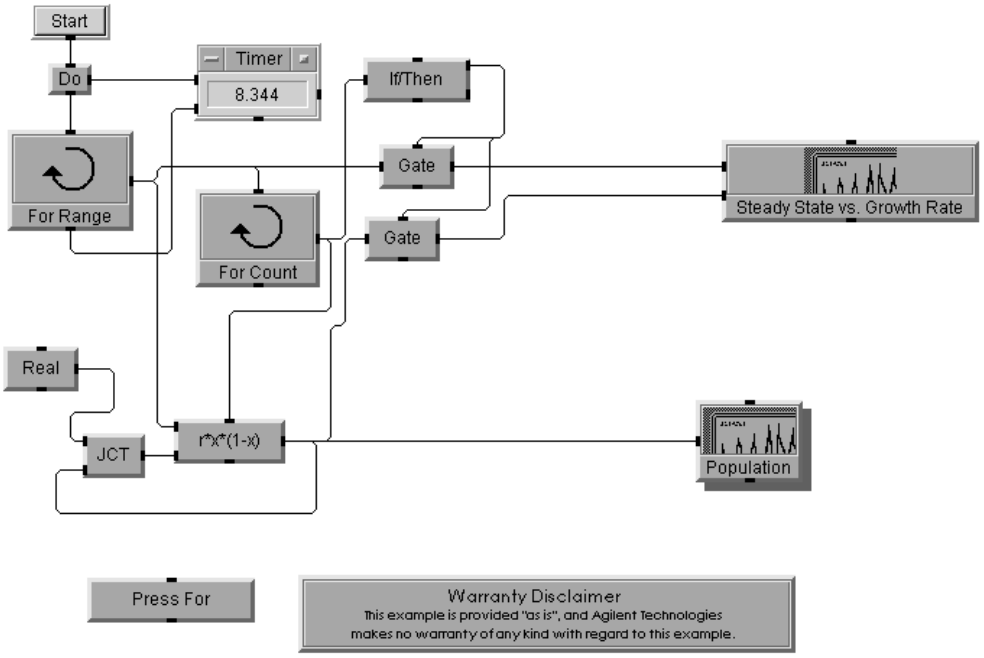


図264　　VEE 3モードで、表示を閉じたChaos.vee

最後に、図265では、デバッグ機能を無効にして、コンパイラをオンにしています。プログラムを完全にデバッグして、使用準備が完了したら、最適なパフォーマンスを得るため、**[File]** ⇒ **[Default Preferences]**の[Disable Debug Features]ボックスをチェックします。

デバッグ機能により、[Show Execution Flow]や[Activate Breakpoints]などのツールが有効になります。[Disable Debug Features]ボックスをチェックすると、メモリ内でのサイズが小さくなり、プログラムの速度が向上します。図のように、プログラムは約12倍速く実行されます。これらの3つの図により、最適化の手法とコンパイラを組み合わせると、最大の処理速度が得られることがわかります。

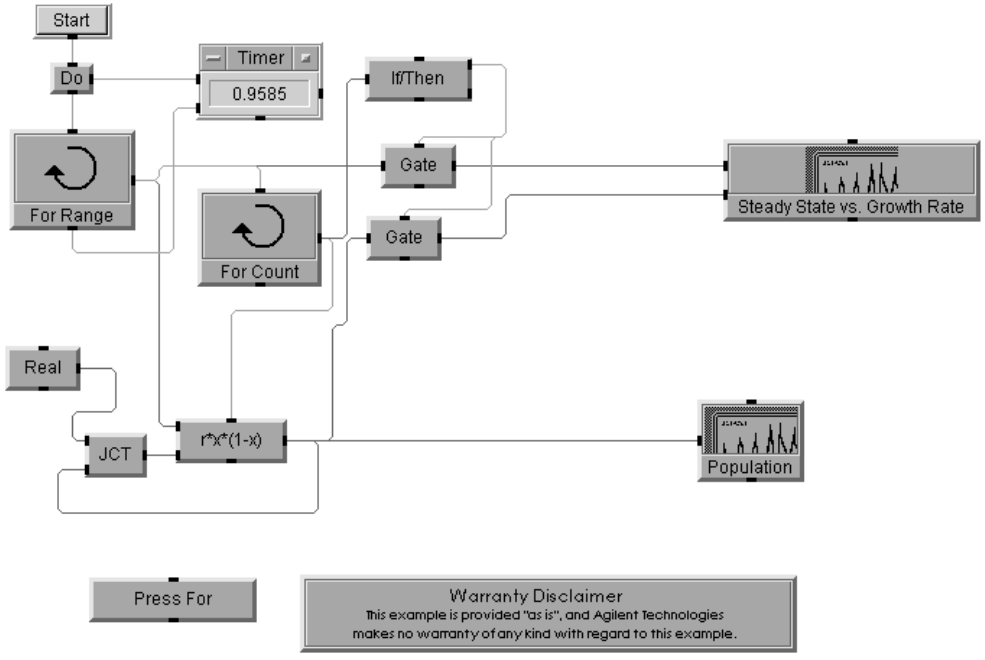


図265　　VEE 4以上のモードで、デバッグを無効にしたChaos.vee

図266と図267では、VEEの速度向上のために、反復スカラ演算ルーチンを含むプログラムでコンパイラを使用します。この例では、スカラ値の平方根を計算しています。結果は保持しません。コンパイラを使用すると、速度は、VEE 3実行モードを使用する場合より約107倍速くなります。

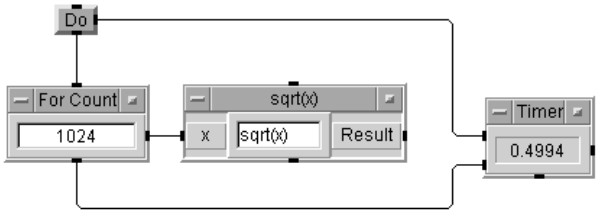


図266　　VEE 3モードでの反復演算の例

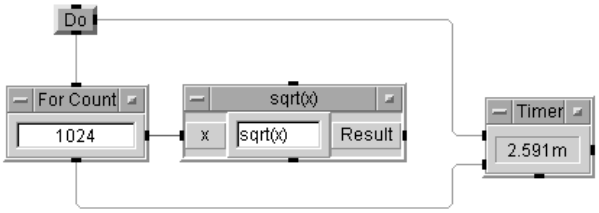


図267　　VEE 4以上のモードでの反復演算の例

VEEに実行モードがあるのは、古いバージョンのVEEでは使用できたが、現在のバージョンでは使用できないプログラミング・オプションが多少あるからです。これらは、現在エラー・メッセージを生成します。さらに、ActiveXオートメーションとコントロールの機能の一部が拡張されたため、VEE 4またはVEE 5モードで実行されていたプログラムをVEE 6モードで実行するには、小さな変更が必要になることがあります。実行モード間の相違についての詳細は、『*VEE Pro Advanced Techniques*』を参照してください。新しいプログラムは、すべてVEE 6モードで作成する必要があります。

Agilent VEEプロファイラ

プロファイラは、VEEのプロ開発環境における機能です。プロファイラを使用すると、プログラム内のUserFunctionまたはUserObjectの実行速度を表示することにより、プログラムを最適化することができます。

プロファイラを使ってプログラム内の遅いポイントを識別し、この章で説明した技法を適用してプログラム速度を増加します。図268に、**examples\Applications\mfgtest.vee**プログラムを示します。

プロファイラをオンにするには、**[View]** ⇒ **[Profiler]**を選択します。次に、プログラムを実行します。画面の下半分にプロファイラが表示されます。プロファイラには、各UserObjectおよびUserFunctionの実行にかかった時間の長さに関する比較情報がリスト表示されます。

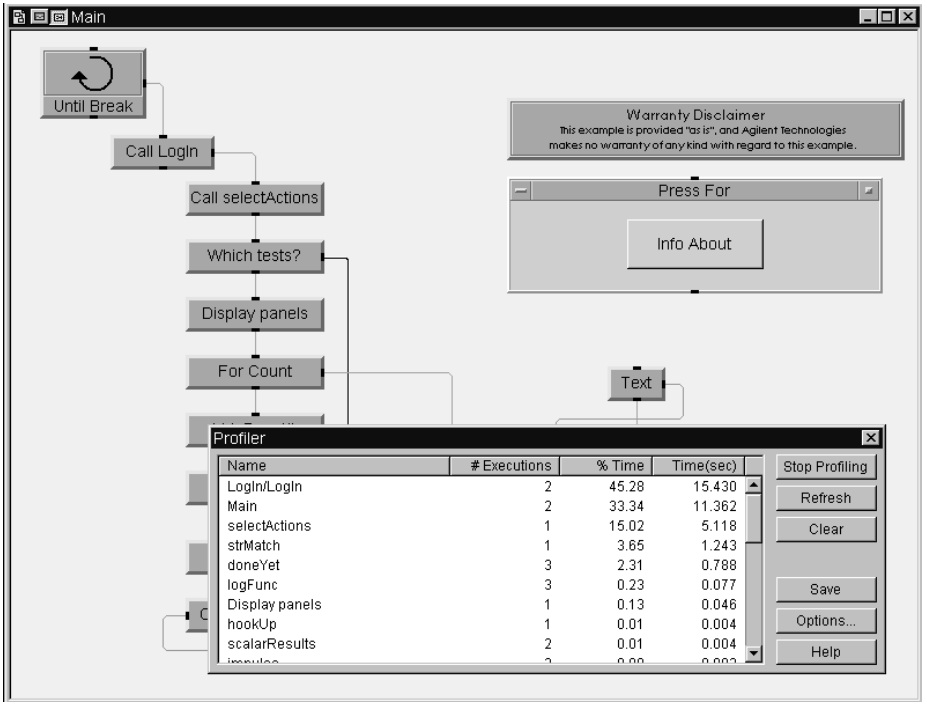


図268　　プロファイラの例

この章の復習

この章では、以下について学びました。必要に応じてトピックを復習してください。

- VEEプログラムを最適化するための3つの基本的な手法について説明し、それぞれの例を示す。
- 上の3つの手法のほか、少なくともあと2つの手法について説明する。
- DLLの基本概念について説明する。
- DLLをインポートし、関数を呼び出し、そのDLLを削除する。
- VEE 6実行モードを使用して、またはVEE 5、VEE 4、VEE 3実行モードを使用してプログラムを実行し、どちらかの実行モードを選択する場合の理由について説明する。
- VEEプロファイラを使用する。

13
プラットフォーム固有の事項とWebモニタ管理

プラットフォーム固有の事項とWebモニタ管理

この章の内容

- ActiveX Automation Serverを使用して、ほかのアプリケーションからVEE関数を呼び出す方法
- Webモニタ管理

平均所要時間: 2時間

概要

この章では、Callable VEE ActiveX Automation Serverを使用して、VEE関数をほかのアプリケーションやプログラムの中に組み込むための重要な手法について学びます。最後に、Webモニタ管理の主要な概念について学びます。

Callable VEE ActiveX Automation Server

Windows 98、Windows 2000、Windows NT 4.0、Windows XP上のVEEでは、MS Visual BasicやCなどの標準言語で書かれた市販の、または著作権を持つほかのテスト・システムの中にVEEオブジェクトを統合することができます。Callable VEE ActiveX Automation Serverは、VEE UserFunctionをカプセル化して、MS ExcelやMS Visual Basicなどのオートメーション・アプリケーションに統合します。このCallable VEE ActiveX Automation Serverは、言語に依存しないオートメーション・インタフェースを実装しています。オートメーション・サーバを呼び出すことのできる任意のアプリケーションまたは言語でこのインタフェースを使用できます。

注 記

VEEでは、ActiveXオートメーションとコントロールを使用して、VEEからほかのアプリケーションを制御できます。MS Visual Basicなどのほかのアプリケーションは、Callable VEE ActiveX Automation Serverを使用して、VEEを制御できます。

Callable VEE ActiveX Automation Serverは、プロパティとメソッドを介してクライアントと通信します。CallableVEE ActiveX Automation Serverの呼び出し元となるVisual BasicやCなどの環境により、呼び出し方法が決まります。Callable VEE ActiveX Automation Serverには、その呼び出し元の環境内で幅広く使用できるオンライン・ヘルプが用意されています。詳細は、VEEのオンライン・ヘルプ、『*VEE Pro Advanced Techniques*』、およびVEEに付属するサンプルを参照してください。

注 記

Callable VEE ActiveX Automation Serverは、バージョン5.0に付属するCallable VEE ActiveX Controlに代わるものです。

Web対応技術

VEEを使用すると、プログラムで収集したデータの配布、テスト・システムの監視、テスト結果の確認をリモートで行うことができます。このセクションでは、テストと測定用アプリケーションのWeb対応技術と、VEEによるサポートについて説明します。

Web技術の概要

この例では、組織内にWebサイト(イントラネットWebサイト)を構築し、これをサーバとして参照データを提供する方法について説明します。このサイトにアクセスするユーザは、ブラウザを持つとします。また、Microsoft環境(Windows 98、Windows NT 4.0、Windows 2000、またはWindows XP)、MS Office、MS Internet Explorerを使用します。図269に示すように、情報は、測定器からサーバへ、サーバからクライアントのブラウザへと順に送信されます。

注 記　この例では、PC画面ダンプを使用します。

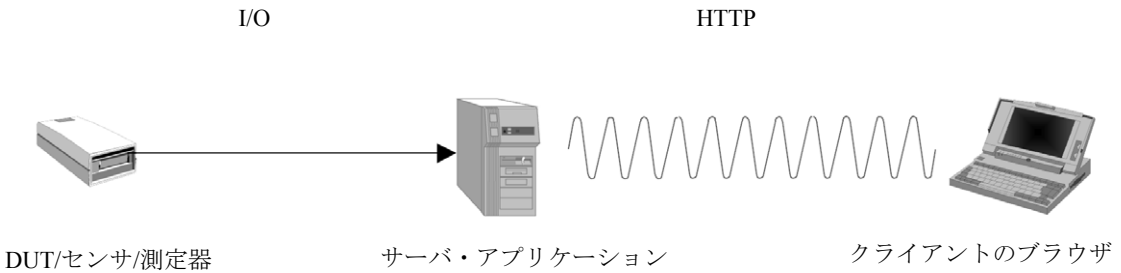


図269　Web測定アプリケーションのモデル

図269に、次の情報の流れを示します。

- **テスト対象の装置(DUT)、センサ、または測定器**は、ネットワークI/Oレイヤを介して、情報をサーバ・アプリケーションに送信します。ネットワークI/Oレイヤには、インタフェースとバックプレーン(GPIB、RS232、VXI、MXI、USB、LAN、PCプラグイン)、およびI/Oプログラム・レイヤ(ドライバ、VISA、SICLなど)があります。
- **サーバ・アプリケーション**は、測定データを生成するVEEプログラムです。
- *HTTP(HyperText Transfer Protocol)*は、情報をクライアントのブラウザに送信します。

HTTPは、Webが使用するTCP/IP (Transmission Control Protocol/Internet Protocol)通信プロトコルの1つです。TCP/IPプロトコルの下位レベルには、トランスポート・レイヤ、ネットワーク・レイヤ、物理レイヤの各通信レイヤがあります。どのTCP/IPアプリケーションもクライアント/サーバ・アプリケーションになります。たとえば、Internet Explorerなどのクライアント・ブラウザは、サーバ・アプリケーションによって生成された情報を要求できます。

次のURL (Universal Resource Locator)をInternet Explorerに入力すると、サーバから情報を要求することになります。

http://www.agilent.com/find/vee

- httpは、情報を転送するためにアクセスされるリソースの種類を表します。
- **www.agilent.com/find/vee**は、リソースのURLです。HTTPによって使用されるハイパーテキスト・フォーマットは、HTML (HyperText Markup Language)と呼ばれるスクリプト記述フォーマットの1つです。HTMLは、文書どうしをリンクする方法の1つで、過去にはWebページの作成に使用できる唯一の言語でした。HTMLは、当初はテキスト専用でしたが、現在では音声、ビデオ、イメージ、対話型画面、ActiveXコントロール、Javaアプレットが導入されています。

ブラウザからサーバに情報を要求しても、ブラウザがそのように設計されていないかぎり、その後で情報が自動的に更新されることはありません。また、ブラウザを使った対話も、それがブラウザ・ページ内に組み込まれていないかぎり、可能ではありません。これらの機能を最も簡単に実現する方法は、VBScript、JavaScript、Jscriptなどのスクリプト記述言語を使用することです。

スクリプト記述言語は、ブラウザによってサポートされるインタープリタ言語の1つです。スクリプト記述言語により、HTMLの使用範囲が広がり、より対話型のWebページを提供できます。スクリプト記述言語はインタープリタ言語なので、Webページに埋込み、ブラウザによってサポートされる必要があります。これは独立したプログラムではありません。これを図270に示します。

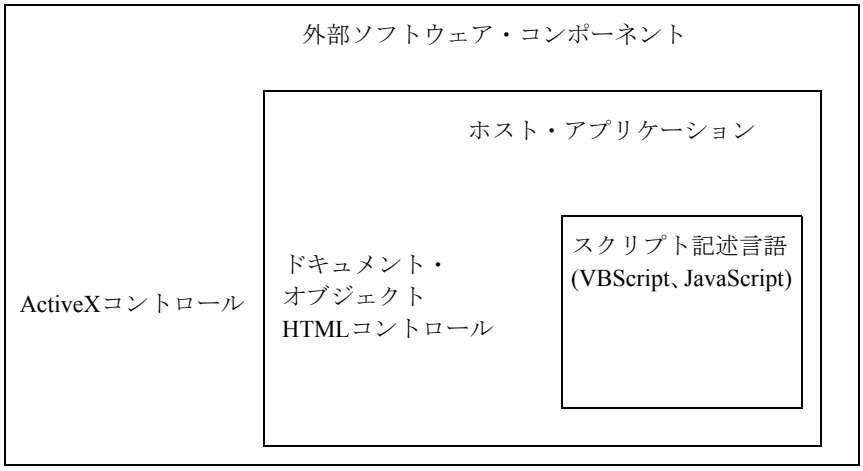


図270　　スクリプト記述言語ホスト・モデル

VBScript、JavaScript、JScriptはスクリプト記述言語です。VBScriptは、MS Visual Basicをベースにします。JavaScriptは、JavaとともにSunによって作成されています。JScriptは、MicrosoftバージョンのJavaScriptをベースにします。

スクリプト記述言語は、ホスト・アプリケーション内に存在している必要があります。ホスト・アプリケーションは、通常、Internet ExplorerやNetscape NavigatorなどのWebブラウザです。ブラウザ内のWebページの全体的な外見は、HTMLによって制御されます。

MicrosoftのCOM(コンポーネント・オブジェクト・モデル)は、ActiveXコントロールと呼ばれるコンパイル済みソフトウェア・コンポーネントを定義し、ActiveXコントロールには、特定の用途を持つ機能がカプセル化されます。通常、ActiveXコントロールは、ユーザ・インタフェースの機能を提供するために使用され、クライアントのコンピュータで実行され

るように設計されます。ActiveXコントロールは、最適化されたソフトウェア・コンポーネントであり、以前はOLEコントロールと呼ばれていました。

Agilent VEEを使ったWebモニタ管理

VEEには、HTTPを介してほかのプログラムと通信するためのWebサーバが組み込まれています。自分のコンピュータにあるVEEプログラムと情報にリモート・ユーザからアクセスできます。このセクションでは、VEEデータを共有し、リモート・ユーザからそのデータにアクセスする方法について説明します。

全般的なガイドラインとヒント

- サーバ側のシステムで実行されるVEEプログラムが、リモート・ユーザがアクセスするVEEプログラムになります。
- サーバ側のシステムでVEEを実行する必要があります。サーバ側のシステムのVEEプログラムにアクセスするために、リモート・ユーザ側にVEEがインストールされている必要はありません。
- リモート・ユーザがネットワーク・ブラウザから VEE に要求を送ると、VEEは、ピクチャを1つ作成してリモート・ユーザのブラウザ・ウィンドウに表示します。このピクチャは、VEEプログラムの「スナップショット」であり、編集することはできません。
- リモート・ユーザは、VEE プログラムのさまざまな部分を表示したり、(プログラムの進捗を監視するため)一定間隔でブラウザ表示を更新するようにVEEプログラムに指示したり、エラー・メッセージ情報を表示することができます。それには、VEE Webサーバ・ホームページでオプションを選択するか、ブラウザでURLにコマンド・ライン・オプションを指定します。

Agilent VEEのデータをリモート・ユーザに提供する方法

リモート・ユーザがサーバ側のシステムにあるデータにアクセスできるように、VEE Webサーバをセットアップします。一般には、次の手順に従います。

1. サーバ側のシステムがネットワークに接続されていることを確認します。
2. サーバ側システムのURLに関する情報をリモート・ユーザに提供します。リモート・ユーザは、このURLをブラウザに入力して、サーバ側のシステムにアクセスします。詳細については、後述します。

3 VEEを起動します。リモート・ユーザからアクセスされるプログラムを開いたり、リモート・ユーザからアクセスされるファイルを作成します。

4 **[File]** ⇒ **[Default Preferences]** ⇒ **[Web Server]**ダイアログ・ボックスの設定を選択して、Webサーバを有効にします。詳細については、後述します。

5 リモート・ユーザは、Internet ExplorerやNetscapeなどのWebブラウザを実行します。

[Web Server]ダイアログ・ボックス

[File] ⇒ **[Default Preferences]** ⇒ **[Web Server]**を選択すると、図271に示すように、[Web Server]ダイアログ・ボックスが表示されます。

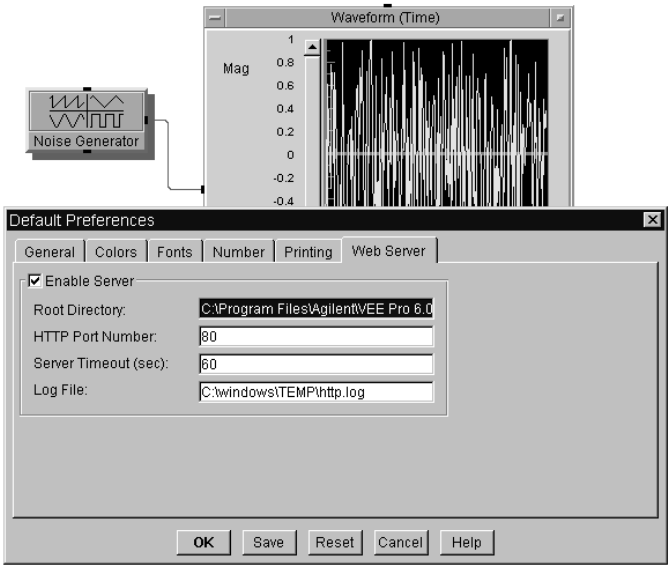


図271　[Default Preferences]の[Web Server]ダイアログ・ボックス

[Web Server]ダイアログ・ボックスには、次のフィールドがあります。

表50　[Web Server]ダイアログ ボックスのフィールド

フィールド名	説明
[Enable Server]	[Enable Server]をチェックした場合は、VEEの内蔵Webサーバがオンになります。Webサーバにより、リモート・ユーザは、自分のWebブラウザにサーバ側のVEEプログラムを表示して、サーバ側システムのVEEプログラムを表示、監視、トラブルシューティングできます。 VEEのWebサーバは、標準のHTTPプロトコルを使用します。デフォルトでは、[Enable Server]はオフ(チェック・マークなし)です。リモート・ユーザが VEE Webサーバにアクセスするには、サーバ側のシステムがネットワークに接続され、そこでVEEが実行されており、リモート・ユーザはWebブラウザを実行している必要があります。
[Root Directory]	リモート・ユーザがアクセスできるファイルの場所を指定します。デフォルトは、*~installDir*/wwwです。Wwindows 98、Windows NT 4.0、Windows 2000、またはWindows XPにおいて、Pro 7.0がデフォルト設定のままインストールされている場合、このフィールドには次のように表示されます。 **C:\Program Files\Agilent\VEE Pro 7.0\www** • このディレクトリやそのサブディレクトリには、個人的なファイルを**置かない**でください。サーバ側のシステムにアクセスできるリモート・ユーザならだれでも、Webブラウザを使ってそれらのファイルを表示できるからです。 • VEE は、インストール時に、Web サーバのデフォルトの**ルート・ディレクトリ**にファイル**index.html**をインストールしています。デフォルトでは、このファイルは **C:\Program Files\Agilent\VEE Pro 7.0\www\index.html**です。これがデフォルトのVEE Webサーバ・ホームページであり、リモート・ユーザがサーバ側システムにアクセスしたときに表示されます。このホームページは、必要に応じて編集できます。 • Webファイルのルート・ディレクトリを変更した場合は、**index.html**ファイルを新しいディレクトリに移動してください。ファイル名を変更してはいけません。

表50　[Web Server]ダイアログ ボックスのフィールド

フィールド名	説明
[HTTP Port Number]	VEE Webサーバのポート番号を指定します。デフォルトのポート番号は80です。デフォルトのポート番号を変更する必要があるのは、同じシステムでもう1つのWebサーバ(VEEの別のインスタンスなど)を実行する場合、またはVEE Webサーバにアクセスするリモート・ユーザを制限する場合だけです。 • 0～65535のHTTPポートを指定することにより、サーバ側のシステムにあるデータにアクセスおよび表示できるリモート・ユーザを制限できます。 • 別のポート番号を指定した場合、VEEプログラムを表示するリモート・ユーザも、同じポート番号を各自のブラウザに入力する必要があります。たとえば、このフィールドを85に設定した場合は、リモート・ユーザもURLに「**http://ホスト名:85**」と入力する必要があります。**ホスト名**については、この章の後の方で詳しく説明します。
[Server Timeout]	VEE WebサーバがVEEのコマンド処理を待機する最大の時間を指定します。デフォルトでは、60秒に設定されています。VEEプログラムがコマンドを処理するために、指定されているより多くの時間を要した場合、VEEはWebサーバにタイムアウトを送信します。
[Log File]	Webサーバによって処理されるすべての着信HTTP要求を記録するログ・ファイルを指定します。PCでは、このファイルのデフォルトの場所は**C:\windows\TEMP\http.log** です。**[Default Preferences]** ⇒ **[Web Server]**ダイアログ・ボックスで変更を行うと、ログ・ファイルがまだ存在しない場合は、ファイルが作成されます。
[Save]	再び変更しないかぎり、以降のVEE Webセッションではこれらの値がデフォルト値になります。現在のVEEセッションで**のみ**、指定されている値を使用する場合は、[OK] をクリックします。

リモート・ユーザがサーバ側システムのAgilent VEEにアクセスする方法

リモート・ユーザが別の場所からサーバ側のシステムにあるVEEファイルにアクセスするには、一般に次の手順に従います。

1 リモート・ユーザのシステムがネットワークに接続されている必要があります。

2 リモート・ユーザがNetscapeやInternet Explorerなどのネットワーク・ブラウザを実行している必要があります。

3 リモート・ユーザがサーバ側システムのURLを入力する必要があります。このアドレスは、サーバ側のユーザからリモート・ユーザに提供する必要があります。詳細については、後述します。

注 記

サーバ側のシステムにあるVEEプログラムにアクセスするために、リモート・ユーザがVEEを実行する必要はなく、VEEがインストールされている必要もありません。

リモート・ユーザはネットワークに接続し、ブラウザを実行し、URLを入力します。以下の説明に基づき、入力するURLをリモート・ユーザに指示してください。リモート・ユーザは、次の形式でURLを入力できます。

http://hostname {:port}

VEE Webサーバ・ホームページを表示します。リモート・ユーザは、このページで、VEEプログラムの表示方法に関するオプションを入力できます。

http://hostname{:port}{/command}{?parameter}

リモート・ユーザがVEEプログラム内に表示するビューまたはUserFunctionを指定します。たとえば、**http://ホスト名/ViewMainDetail**により、メインVEEプログラムの詳細ビューが表示されます。

http://hostname{:port}{/file}

リモート・ユーザが表示する保存済みのファイル(`*.jpeg`、`*.html`など)を指定します。

注 記

中かっこ { } で囲まれたフィールドは、省略可能です。

次に、URLアドレス内のフィールドについて説明します。

リモート・ユーザがサーバ側のシステムにあるVEEプログラムにアクセスする場合、URL内で使用できるコマンドとパラメータは次のとおりです。

表51　URLコマンドおよびパラメータ

フィールド名	説明
[Hostname]	(必須) VEEが実行されているサーバ側のシステムを識別します。フォーマットは、「<**コンピュータ名**>.domain.com」です。 「http://<ホスト名>」のように、リモート・ユーザにURLとして<**ホスト名**>だけを入力してもらうこともできます。このコマンドにより、VEE Webサーバ・ホームページ(**index.html**)が開き、リモート・ユーザに表示されます。リモート・ユーザは、このVEE Webサーバ・ホームページのメニューから、VEEプログラムの表示、監視、トラブルシューティングを選択できます。
[Port]	(省略可) デフォルト値80を使用しない場合に、Webサーバのポート番号を識別します。**[File]** ⇒ **[Default Preferences]** ⇒ **[Web Server]**で入力した値が80以外の場合にだけ、この値を指定します。 たとえば、リモート・ユーザがサーバ側システムのVEEにアクセスしようとしており、**[File]** ⇒ **[Default Preferences]** ⇒ **[Web Server]**のポート番号が85に設定されている場合、リモート・ユーザは「http://<**ホスト名**>:85」と入力する必要があります。
[File]	(省略可) ブラウザで開くディレクトリやファイルをルート・ディレクトリからの相対ディレクトリで識別します。リモート・ユーザが表示するファイルを*.jpegや*.htmlなどのファイルに保存している場合にだけ、ファイルを指定します。 リモート・ユーザがサーバ側のシステムにあるファイルを表示するには、サーバ側のVEEの**[Default Preferences]** ⇒ **[Web Server]**ダイアログ・ボックスで、そのファイルのディレクトリを**ルート・ディレクトリ**として指定する必要があります。
[Commands and Parameters]	(省略可) VEE Webサーバがサポートしているコマンドと、コマンドに必要なパラメータを指定します。コマンドとパラメータを使用すると、リモート・ユーザは、ブラウザを介してVEEプログラムを監視したり、トラブルシューティングできます。コマンドとパラメータの一覧を下に示します。
[Viewing the entire VEE window]	ViewVEE 例: http://**ホスト名**/ViewVEE

表51　URLコマンドおよびパラメータ

フィールド名	説明
[Panel view of main VEE program]	ViewMainPanel 例: `http://ホスト名/ViewMainPanel`
[Detail view of main VEE program]	ViewMainDetail 例: `http://ホスト名/ViewMainDetail`
[VEE execution window during runtime]	ViewExecWindow 例: `http://ホスト名/ViewExecWindow`

[VEE UserFunction in Panel view]	`http://ホスト名/ViewPanel?`
	<*UserFunction*名>
	
	例(**UserFunction**が**AddNoise**の場合):
	`http://ホスト名/ViewPanel? AddNoise`

[Detail view of a UserFunction]	`http://ホスト名/ViewDetail`
	<*UserFunction*名>
	
	例(UserFunctionが**AddNoise**の場合):
	`http://ホスト名/ViewDetail? AddNoise`

[Error window of current program]	`http://ホスト名/ViewError`
[Display list of available command URLs]	ViewHelp `http://ホスト名/ViewHelp`

Agilent VEE Webサーバ・ページの表示方法

VEEは、インストール時に、デフォルトの`index.html`ファイルを`www`ディレクトリに作成します。このファイルは、VEE Webサーバ・ホームページを表示します。リモート・ユーザは、このページにあるオプションをクリックして、サーバ側のVEEプログラムを表示できます。図272に、デフォルトのVEE Webサーバ・ホームページを示します。MS Wordなどを使用し、必要に応じてこのページを編集することもできます。

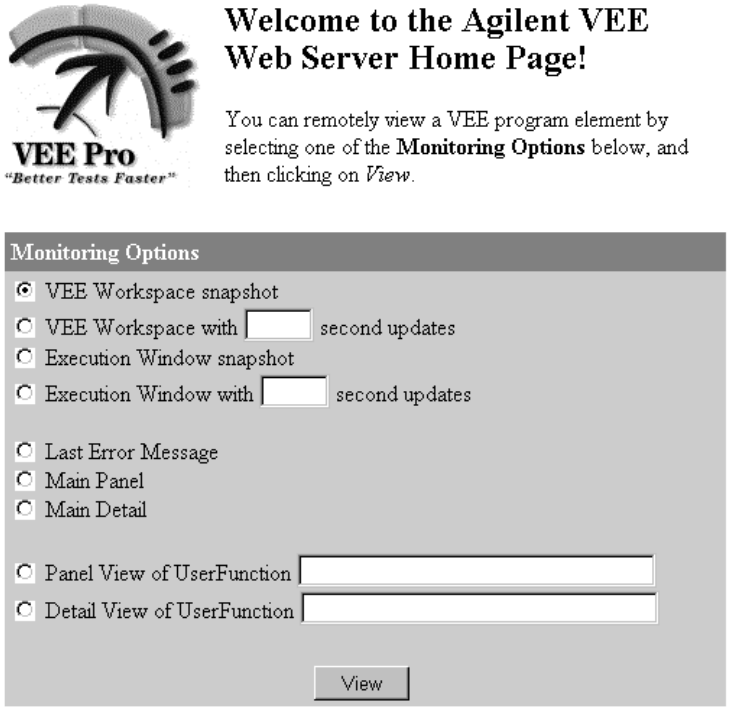


図272　　デフォルトのIndex.htmlページ

注 記

サーバ側のシステムでこのメニューを表示するには、システムをlocalhostとして参照します。たとえば、VEEとネットワーク・ブラウザを実行し、ブラウザに「http://localhost」と入力すると、図272のVEE Webサーバ・ホームページがブラウザに表示されます。このように、リモート・ユーザがさまざまなコマンドを使って表示する内容を簡単に確認できます。

リモート・ユーザは、図272に示すメニューを表示し、メニューからオプションを選択することにより、VEEプログラムのさまざまな部分にアクセスできます。たとえば、[Main Detail]をクリックすると、メインVEEプログラムを詳細ビューで表示できます。メニューでそのように選択すると、ネットワーク・ブラウザにコマンド「`http://`**ホスト名**`/ViewMainDetail`」を入力した場合と同じ情報が表示されます。

例題13-1: Agilent VEE Webブラウザを使った練習セッション

この練習では、リモート・ユーザがサーバ側のシステム上で**Solitaire.vee**プログラムを表示するWebセッションを紹介します。この場合、プログラムにはエラーが1つあるため、リモート・ユーザからその解決方法の問い合わせがあります。

1. VEEを起動します。**[File]** ⇒ **[Default Preferences]** ⇒ **[Web Server]**ダイアログ・ボックスを選択し、**[Enable Server]**をクリックします。デフォルトの設定を使用します。リモート・ユーザが表示する**Solitaire.vee**プログラムを開きます。サーバ側のネットワーク・ブラウザを実行します。
2. リモート・ユーザに連絡し、「`http://Server5`」と入力して Web経由でサーバ側のシステムに接続するように知らせます。「`http://Server5`」は、実際に使用しているコンピュータ名に置き換えてください。
3. リモート・ユーザがURLに「`http://Server5`」と入力すると、リモート・ユーザのブラウザにサーバ側のVEE Webサーバ・ホームページが表示されます。表示内容については、487ページの図272を参照してください。
4. リモート・ユーザは、最初にVEEプログラム全体を表示することにしました。リモート・ユーザは、VEE Webサーバ・ホームページで**[Main Detail]** ⇒ **[View]**をクリックします。ブラウザには、図273に示すビューが表示されます。

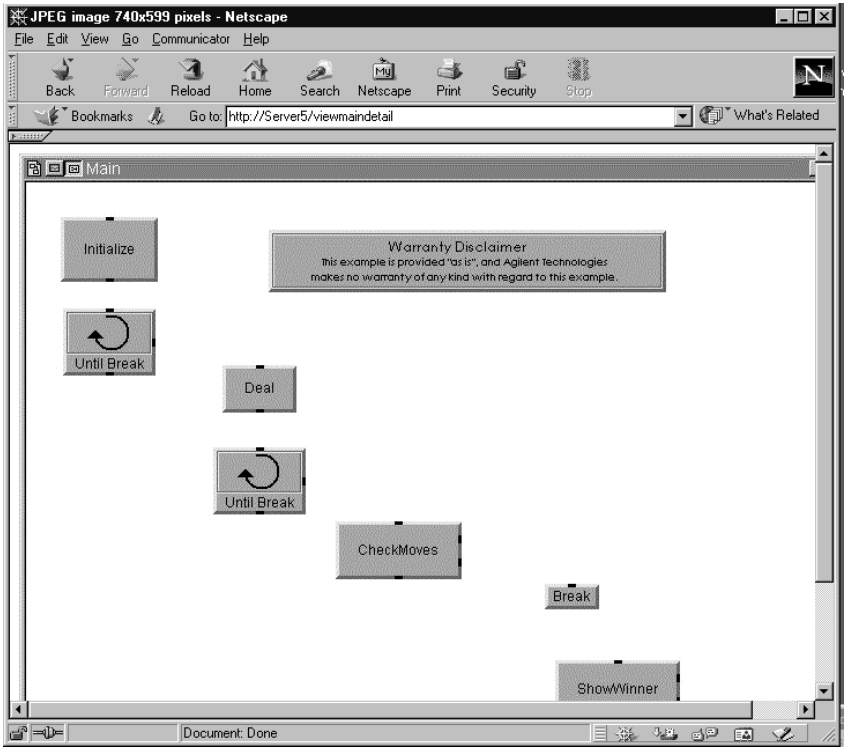


図273　　ブラウザに表示されたSolitaire.veeメイン・プログラム

図273に、VEE内のメイン・プログラムを示します。

注 記

この練習の**Solitaire.vee**プログラムには1つのエラーがありますが、このエラーは、VEE プログラム・サンプルには**ありません**。このプログラムを表示する場合は、**[Help]** ⇒ **[Open Example...]** ⇒ **[Games]** ⇒ **[Solitaire.vee]**を選択してください。

5 リモート・ユーザは、ブラウザで[Back]をクリックしてVEE Webサーバ・ホームページを表示し、[Last Error Message]を選択します。ブラウザには、図274に示すエラー・メッセージが表示されます。

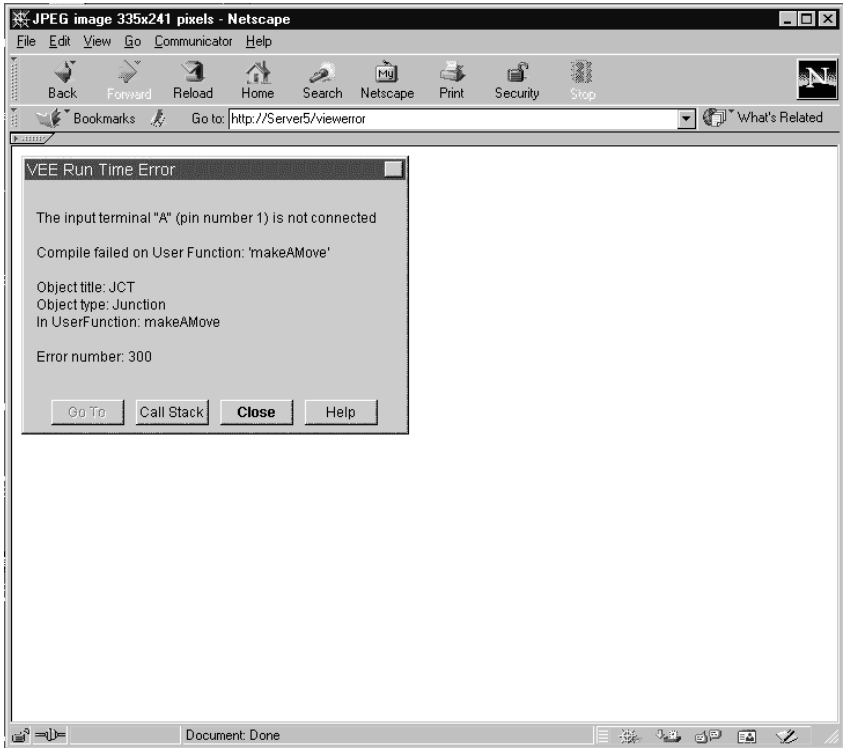


図274　　ブラウザを使ったVEE エラー・メッセージの表示

VEE エラー・メッセージに"UserFunction makeAMove"と明記されていることがわかります。

6 リモート・ユーザは、再びVEE Webサーバ・ホームページに戻り、[Detail View of UserFunction]をクリックして、UserFunction名の「makeAMove」を入力します。図275に示すように、ブラウザには、UserFunction makeAMoveが表示されます。

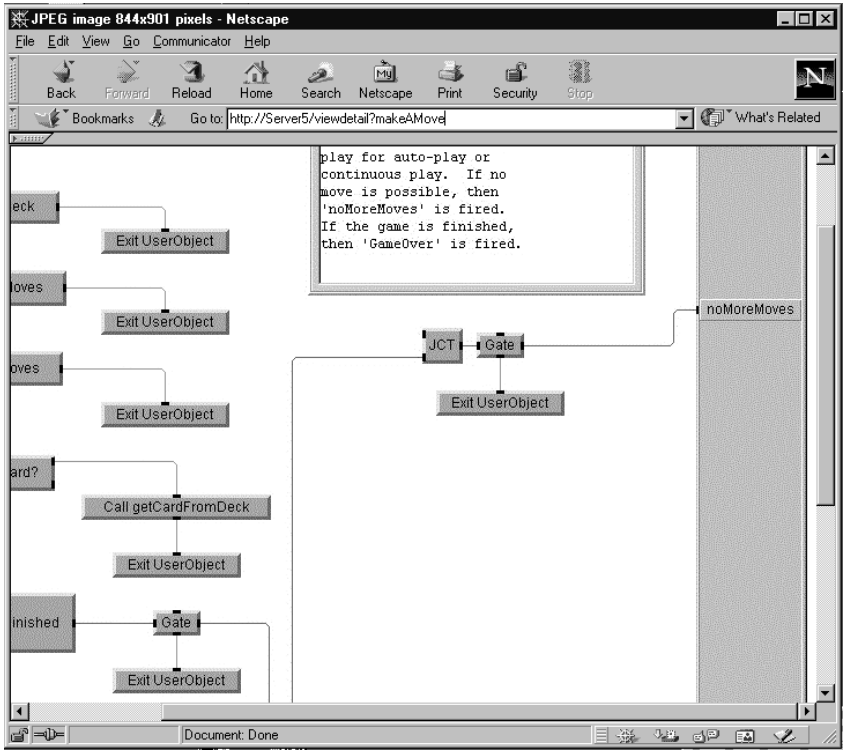


図275　　ブラウザに表示されたUserFunctionの詳細ビュー

リモート・ユーザは、VEEプログラム内のエラーを確認できます。図275に示すように、JCTオブジェクトの入力ピンが接続されていません。ここで、リモート・ユーザは、**Solitaire.vee**のトラブルシューティングに参加して、エラーを解決します。Webを介した同様の共同作業を行うことにより、リモート・ユーザと協力して作業を行ったり、合同でプログラムを開発することができます。

Webを介して表示されるプログラムへのアクセスを制限する方法

VEEプログラムをWeb上で使用可能にする場合でも、リモート・ユーザがプログラム内の一定の部分を表示できないようにする必要が生じることがあります。リモート・ユーザがすでにサーバ側システムのURLを知っている場合は、一定のリモート・ユーザだけが特定のプログラムやWeb

ディレクトリのファイルにアクセスできるようにする必要があります。

リモート・ユーザがWeb上でVEEプログラムの特定の部分を表示できないようにする場合、そのようにプログラムを保護する方法は、3通りあります。

許可されたユーザだけがプログラムを表示できるように、**[Default Preferences]** ⇒ **[Web Server]**フォルダでポート番号を変更します。

-または-

保護されたRunTimeバージョンのVEEプログラムを作成します。これで、プログラム・コードをまったく表示できなくなります。詳細は、416ページの「プログラムを保護する方法(RunTimeバージョンを作成する方法)」を参照してください。

-または-

無効にするコマンドと正確に同じ名前を付けたHTMLファイルを作成し、それをVEEのwwwディレクトリに保存します。ブラウザは、VEEを表示する前に、必ずすべての***.html**ファイルにアクセスします。これで、リモート・ユーザからの要求を途中で止めて、適切な警告やコメントを記載したHTMLページを表示できます。

たとえば、リモート・ユーザがVEEプログラムの詳細ビューを表示できないようにするとします。その場合は、MS Wordなどのプログラムでファイルを作成し、それを**ViewMainDetail.html**という名前でwwwディレクトリに保存します。そのファイルの中に、リモート・ユーザに対して表示するメッセージを記入します。

リモート・ユーザがVEE Webサーバ・ホームページで[Main Detail]を選択したり、ViewMainDetailオプションを使ってURLを入力しても、ブラウザはメインVEE プログラムを詳細ビューで表示しません。そのかわり、ブラウザは、wwwディレクトリ内の**ViewMainDetail.html**ファイルにアクセスし、そのファイルを表示します。図276 に、リモート・ユーザに表示するメッセージの例を示します。

注 記

ファイル名がVEEのWebコマンドと同じ名前であることを確認してください。また、それを[Web Server]で指定したルート・ディレクトリに置く必要があります。

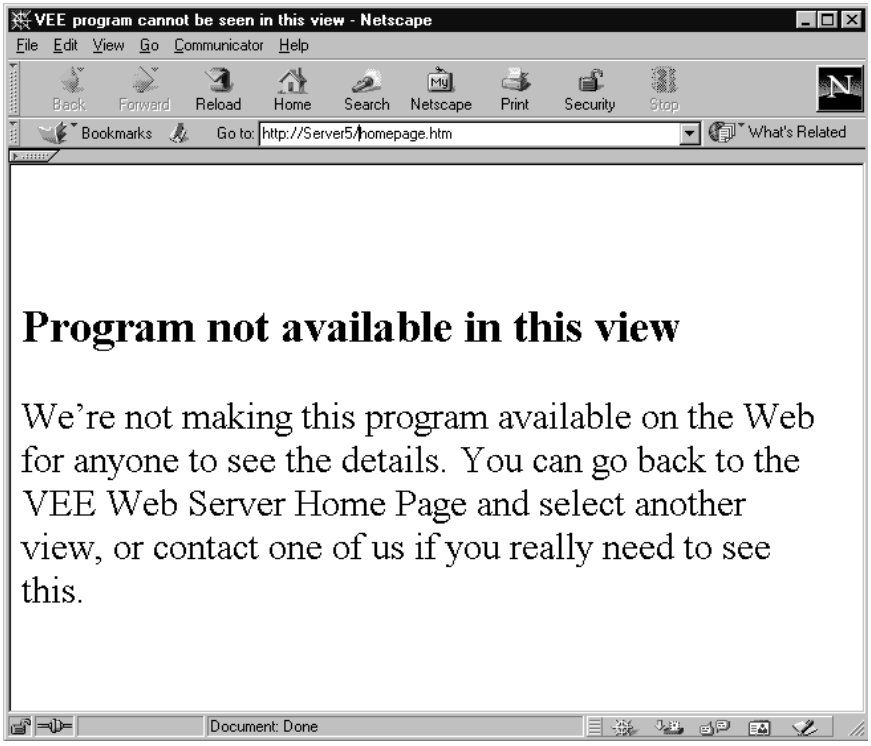


図276　　VEEプログラムのかわりに表示するHTMLメッセージの例

***.html**ファイルは、その他の目的でも使用できます。たとえば、VEEプログラムをパスワードで保護し、パスワードを持つユーザだけがプログラムを表示できるようにすることができます。図277 に、パスワードによる保護の例を示します。

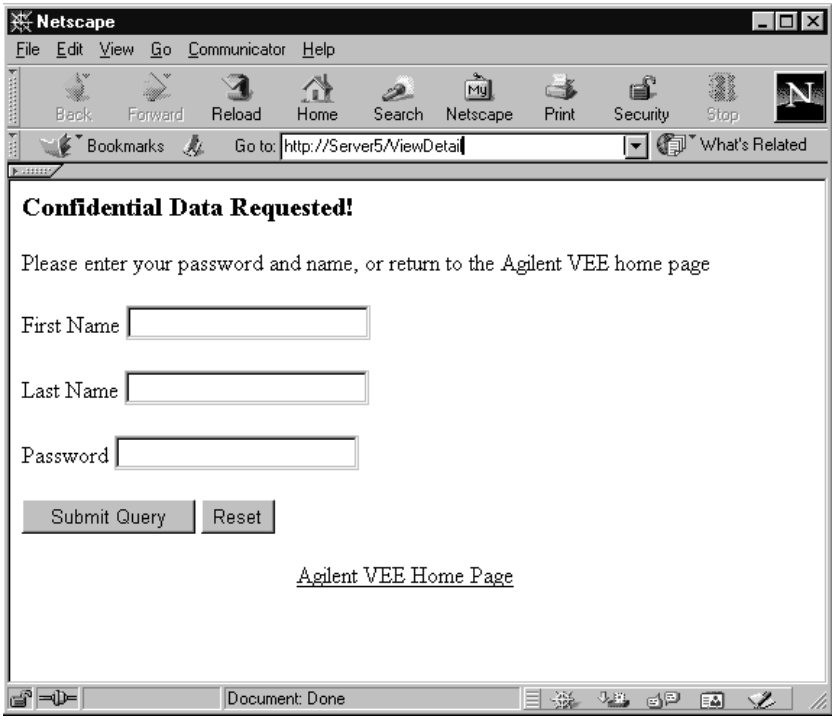


図277　　パスワード・ウィンドウの例

この章の復習

この章では、以下について学びました。

- Callable VEE ActiveX Automation Serverの使用法と、それを使用する場面について説明する。
- VEE の機能をほかのアプリケーションやプログラムに統合する方法について説明する。
- Webを使ってVEEプログラムを監視する際の主な概念について説明する。

付録A: 追加の例題

追加の例題

以下の例題では、このマニュアルで学んだVEEの概念を実践します。練習問題は、いくつかのカテゴリに分かれています。

この付録を使用する場合は、自分の解答を作成してから、記載されている解答と比較します。指定された作業をプログラミングする方法は多数あるため、自分の解答が問題の条件を満たしていれば、有効な解法を開発できたことになります。ただし、実行時間が短く、使いやすいプログラムの方が一般に優れた解法です。解答には、それぞれのキー・ポイントが簡潔にまとめられています。

一般的なプログラミング技術

りんごのかご入れ

かごにりんごを入れて10ポンドにするには何個のりんごが必要かを調べます。何個のりんごでかごがいっぱいになるかを数えるVEEプログラムを作成してください。りんごの重さは、それぞれ0〜1ポンドであるとします。

ヒント

このプログラムは、10個以下のオブジェクトを使って作成できます。次の中から選択してください。

```
Start
Until Break
random()関数
Accumulator
Break
Real64
Conditional (A>=B)
Stop
Counter
If/Then/Else
Alphanumeric
```

注 記

このマニュアルに記載されている練習問題やプログラミング例で使用したVEEプログラムの多くは、VEEに付属しています。[Help] ⇒ [Open Example...] ⇒ [Manual] ⇒ [UsersGuide]を選択してください。

解答1 -「りんごのかご入れ」

図278に、練習「りんごのかご入れ」の解答例を示します。

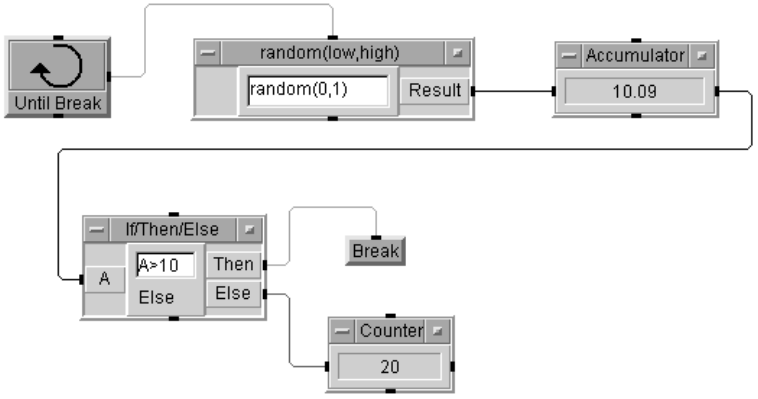


図278　　「りんごのかご入れ」解答1

キー・ポイント

最適な解法:　プログラムのパフォーマンスを上げるには、できるだけ使用するオブジェクトの数を少なくします。この解答では、6個のオブジェクトを使用しています。このプログラムは、図279に示すように、10個のオブジェクトを使って実装することもできます。

Until BreakとBreakオブジェクト:　これらのオブジェクトは、条件の判定を必要とするループに使用します。この例のループは、りんごの総重量が10ポンドより大きくなったときに、停止する必要があります。

Accumulator:　Accumulatorを使って積算合計を保持します。

Counter:　Counterを使って積算カウントを保持します。この例のCounterは、かごの中のりんごの数を追跡するために使用されます。総重量が10を超えた場合は、If/Then/ElseオブジェクトのThenピンだけが起動して、Counterの答が正しくなります。

解答2 - 「りんごのかご入れ」

図279に、もう少しオブジェクトを使った解答例を示します。

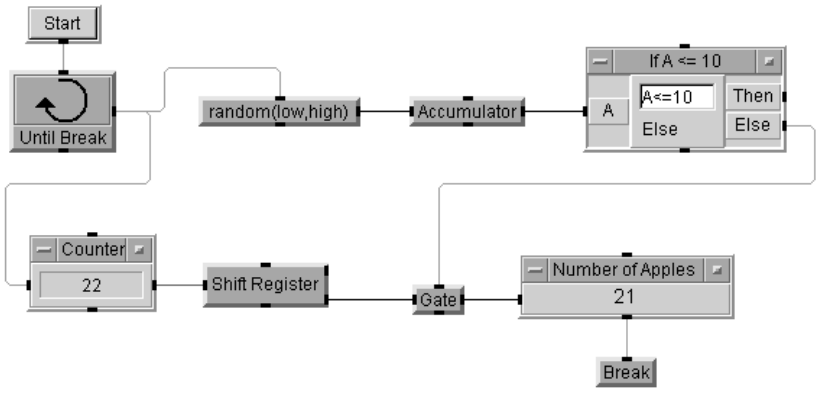


図279　　「りんごのかご入れ」解答2

キー・ポイント

Start:　このプログラムにStartオブジェクトを使用するのは冗長です。なぜなら、メイン・メニュー・バーの[Run]ボタンを使用できるからです。画面に2つのプログラムがあり、それらを互いに無関係に実行する必要がある場合は、Startを使用するのが適切です。また、プログラムにフィードバック・ループがあり、どこで実行を開始するのかを定義する必要がある場合も、Startを使用するのが適切です。

Shift Register:　Shift Registerは、以前の出力値にアクセスするために使用します。解答2では、Counterは、りんごの重さを測定する前にりんごの積算カウントを保持しているため、総重量が10を超えた場合は、そのカウントから1だけ減算する必要があります。

Gate:　Gateは、別の操作が発生し、それがシーケンス・ピンをアクティブ化するまで、出力を保持するために使用します。この図では、条件`A<=10`はすでにTrueではなくなっており、If/Then/Elseオブジェクトの`Else`ピンがゲートをアクティブ化します。

数の判定

「数の判定」ステップ1

ユーザが0～100の数を入力できるプログラムを作成します。その数を50と比較し、大きいか等しい場合はその数を表示します。その数が50より小さい場合は、ポップアップ・ボックスに"Please enter a number between 50 and 100."というメッセージを表示します。

ヒント

このプログラムは、5個以下のオブジェクトを使って作成できます。次の中から選択してください。

```
Start
Int32
Slider
Real64
If/Then/Else
Formula
Gate
Text
Junction
Alphanumeric
Message Box
```

解答 - 「数の判定」ステップ1

図280に、5つのオブジェクトを使った練習「数の判定」の解答例を示します。

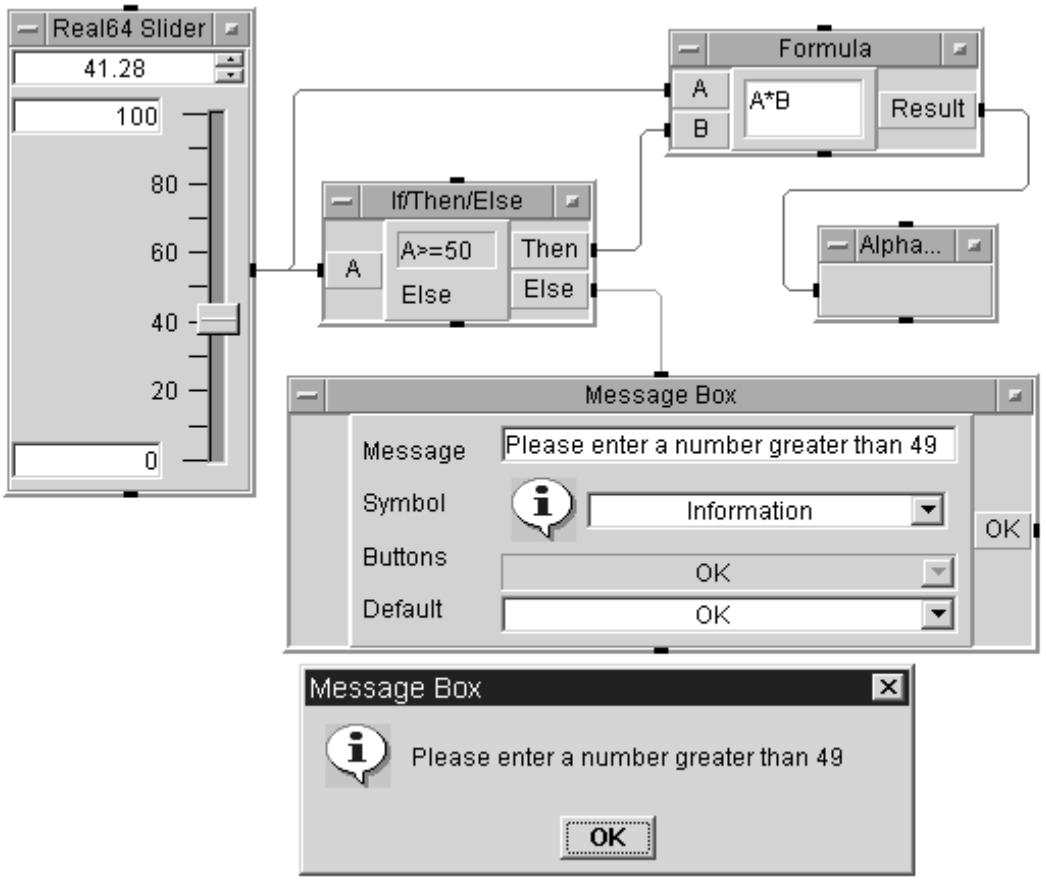


図280　　「数の判定」ステップ1(ポップアップが表示されている)

「数の判定」ステップ2

5つのオブジェクトを使ったモデルが動作し、Message Boxがポップアップを生成したら、Gateオブジェクトを使用しないで、4つのオブジェクトでプログラミングしてみてください。

解答 -「数の判定」ステップ2

図281に、4つのオブジェクトを使った練習「数の判定」の解答例を示します。

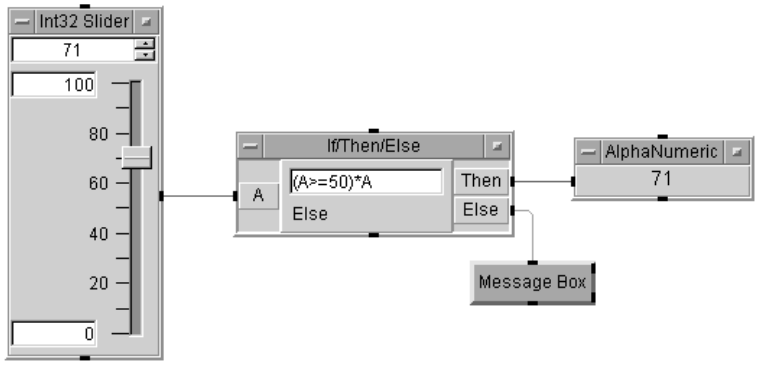


図281　　「数の判定」ステップ2

キー・ポイント

Auto Execute:　Int32 Sliderなどのすべての入力オブジェクトは、[Properties]ダイアログ・ボックスに[Auto Execute]オプションを持っています。これを選択した場合は、[Start]または[Run]ボタンを押さなくても、オブジェクトの値が変わるごとに、オブジェクトが動作します。

式の統合:　If/Then/Elseオブジェクトの式**(A>=50)*A**は、評価されると、**A>=50**がTRUEの場合は**1*A**になり、FALSEの場合は**0**になります。したがって、この式がTRUEの場合は**A**がThenピンに出力され、FALSEの場合は**0**になります。非0と評価される式はすべてTRUEと見なされ、その値がThenピンに伝達されます。

「数の判定」ステップ3

オブジェクトを3つだけ使った解法を作成してください。

ヒント: Formulaオブジェクト内で3項式を使用します。その構文は、(<**式**> ? <**TRUEの場合の出力値**> : <**FALSEの場合の出力値**>)です。たとえば、**A < 10** がTRUEと評価される場合は、**A**の値をResultピンに出力し、そうでない場合は、文字列 "FALSE" をResultピンに出力するとします。この場合は、3項式(A<10 ? A : "FALSE")を使用します。

解答 - 「数の判定」ステップ3

図282に、3つのオブジェクトを使った練習「数の判定」の解答例を示します。

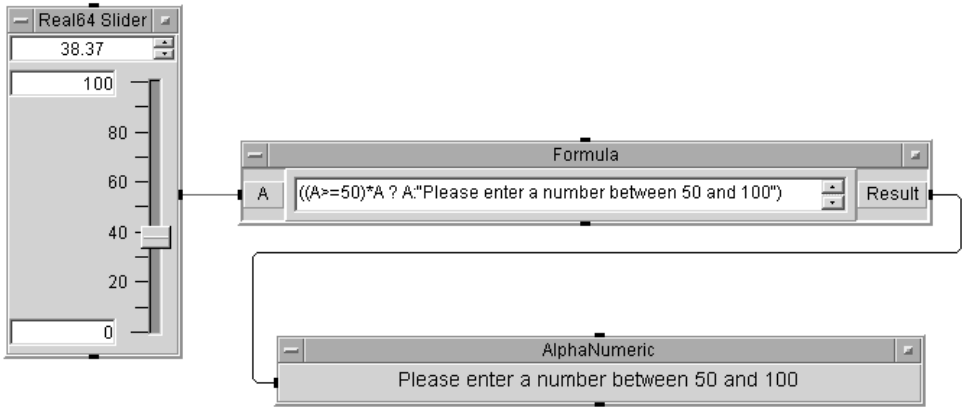


図282　　「数の判定」ステップ3

注 記

これは、自動エラー検出機能を持つ[Real64 Input]ダイアログ・ボックスを使用すると実装できます。ただし、オペレータが有効な数を入力するまで、プログラムは完了できません。

乱数の収集

100個の乱数を生成し、それらを表示するプログラムを作成します。それらの値の生成と表示にかかった合計時間も記録してください。

ヒント

このプログラムは、6個以下のオブジェクトを使って作成できます。次の中から選択してください。

```
Start
For Range
Until Break
randomseed()関数
random()関数
Collector
Formula
Set Values
Alloc Int32
Logging AlphaNumeric
Strip Chart
Meter
Date/Time
Timer
Now()
Break
Do
```

ヒント: パフォーマンスを上げるには、最初にCollectorオブジェクトを使って配列内にデータを収集したうえで、データを一度だけ表示に送信します。パフォーマンスの違いに注意してください。

解答 - 「乱数の収集」

図283に、練習「乱数の収集」の解答例を示します。

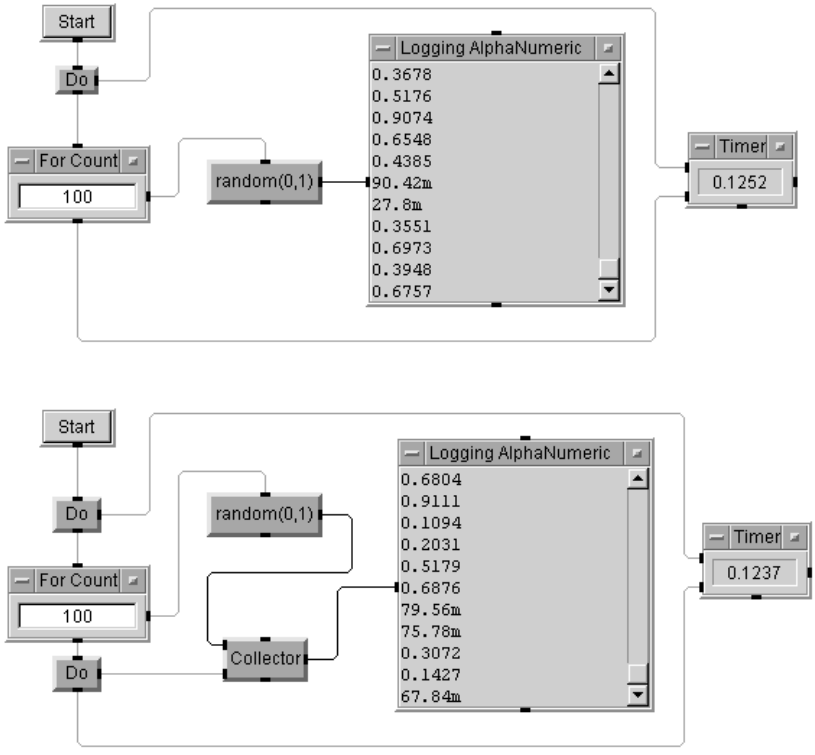


図283　　「乱数の収集」

キー・ポイント

Logging AlphaNumericとAlphaNumeric:　連続する入力(Scalar とArray 1Dのどちらか)を以前の値に続けて表示する場合は、Logging AlphaNumericを使用します。一度(直前)の実行のデータだけを単一の値、Array 1D、またはArray 2Dとして表示する場合は、AlphaNumericを使用します。Logging表示は、インデックス値のない配列であり、AlphaNumeric表示は、省略可能なインデックス番号と値を持つ同じ配列です。

時間測定のピン:　Doオブジェクトは、どのオブジェクトを最初に実行するかを制御します。プログラム終了までの時間は、For Countオブジェクトのシーケンス出力ピンを使って測定されます。このピンは、ループ内のすべてのオブジェクトの実行が終わるまで起動されません。

乱数生成プログラム

「乱数生成プログラム」ステップ1

外部入力を必要とする乱数生成プログラムを作成します。乱数は、ストリップ・チャートに表示してください。次の項目を入力できる必要があります。

乱数の最大値

乱数の最小値

生成する乱数の数

解答 -「乱数生成プログラム」ステップ1

図284に、練習「乱数生成プログラム」ステップ1の解答例を示します。

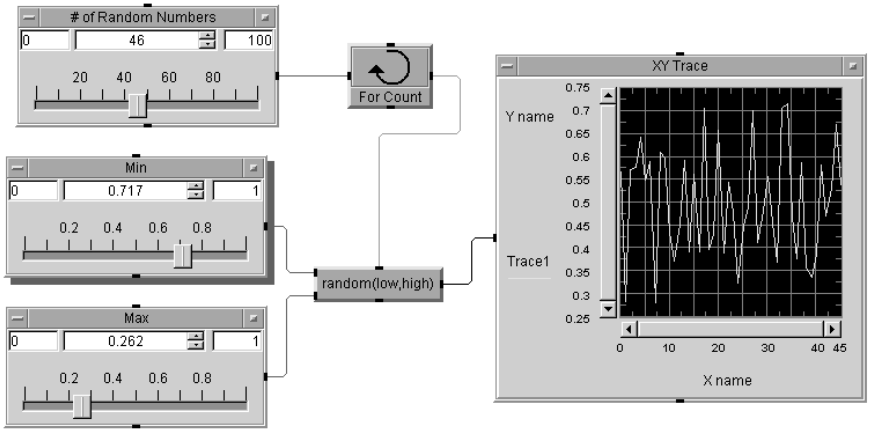


図284　　「乱数生成プログラム」ステップ1

キー・ポイント

Sliderオブジェクトのレイアウト: Sliderオブジェクトの画面イメージについては、[Properties]ボックス内の[Layout]にある[Horizontal}をクリックすると、垂直または水平の形式を選択できます。

XY Trace: 連続して生成されるデータの最近の履歴を表示するには、XY Traceを使用します。

「乱数生成プログラム」ステップ2

乱数を配列に収集します。移動平均を計算し、それを乱数とともに表示してください。

解答 -「乱数生成プログラム」ステップ2

図285に、練習「乱数生成プログラム」ステップ2の解答例を示します。

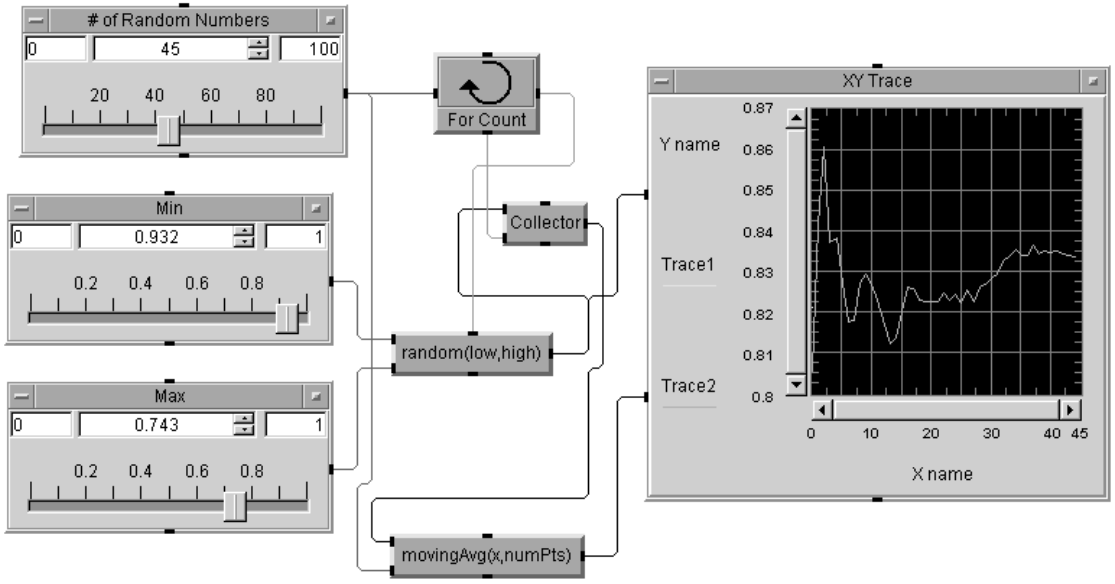


図285 「乱数生成プログラム」ステップ2

MovingAvg(x, numPts): [Function & Object Browser]の[Data Filtering]カテゴリにあるこのオブジェクトを使用すると、平滑化の計算対象となるデータ点の前にある指定した数のデータ点の平均を使用して、入力データを平滑化できます。

マスクの使用方法

「マスク・テスト」ステップ1

調整可能な量のノイズを使用して、50Hzの正弦波を作成します。正弦波のノイズが次の制限内にあるかどうかを判定してください。

(0,0.5)

(2.2m, 1.2)

(7.2m, 1.2)

(10.2m, 0.5)

(20m, 0.5)

正弦波が制限を超えた場合は、その点を赤いひし形でマークします。

ヒント: 線の表示フォーマットは、ドットやひし形に変更できます。[Properties]ダイアログ・ボックスで、各トレース入力の[Traces]タブを選択して、線の種類を実線、破線、点のみなどに変更できます。また、[Point Type]には、単なる点、ひし形、四角などの形状があります。Comparatorオブジェクトが有効であることがわかります。

解答 -「マスクの使用方法」ステップ1

図286に、ステップ1の解答例を示します。

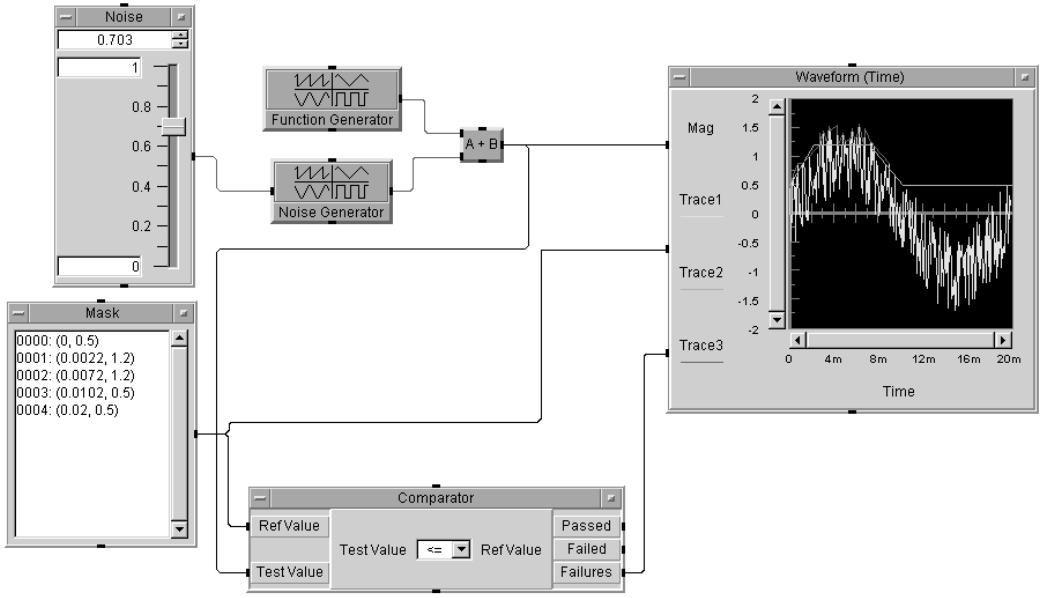


図286　　「マスク・テスト」ステップ1

「マスクの使用方法」ステップ2

失敗の割合を計算して表示するように、プログラムの機能を拡張してください。

解答 -「マスクの使用方法」ステップ2

図287に、ステップ2の解答例を示します。

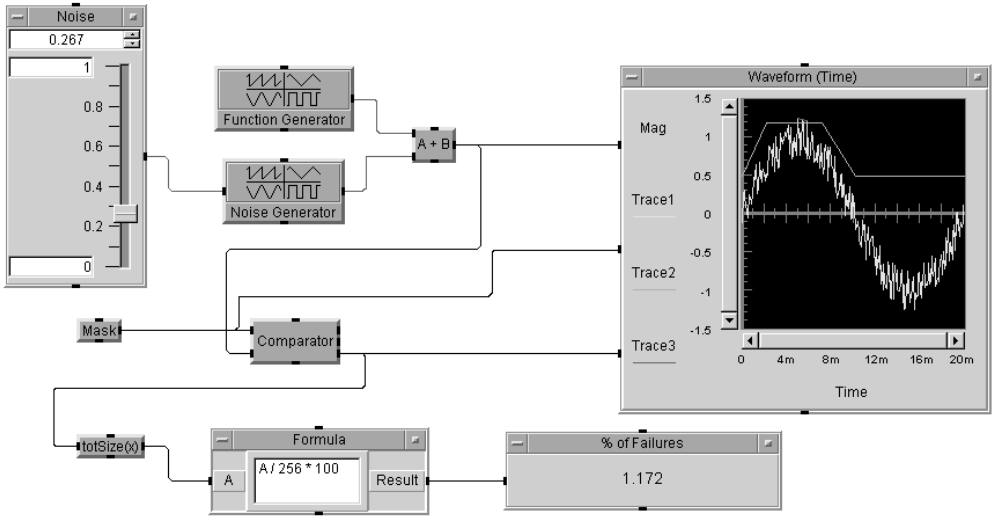


図287　　「マスク・テスト」ステップ2

キー・ポイント

マスク:　マスクを作成するには、**[Data]** ⇒ **[Constant]** ⇒ **[Coord]**オブジェクトを使用します。これを5つの配列要素用に設定します。座標の組をカンマで区切って入力すると、VEEによって丸かっこ()が追加されます。**x**値は、波形のタイム・スパンが20ミリ秒であることによって選択されています。また、Waveform (Time) 表示は、Coordデータ型を入力として受け入れることに注意してください。**[Data]** ⇒ **[Build Data]** ⇒ **[Arb Waveform]**オブジェクトを使用することもできます。このオブジェクトは、Waveform内の点の数を指定することにより、CoordをWaveformデータ型に変換します。

Comparator:　このオブジェクトは、テスト値を参照値と比較します。つまり、波形と座標の配列を比較できます。Failuresピンは、失敗したデータ点の配列を出力します。それを表示に送信し、異なる種類の線や色で強調表示できます。

TotSize:　このオブジェクトは、単に、配列内の要素の数を出力します。この配列には失敗の数が保持されているので、その数を元の波形の要素の総数256で除算し、100を乗算すると、失敗の割合(%)を得ることができます。

Formula:　`A/256*100`は、失敗の割合(%)を計算する式です。Function GeneratorとNoise Generatorは、256の点を出力するように設定されています。

文字列とグローバルの使用方法

文字列とグローバルの操作

文字列のオブジェクトや関数を使用して、「<**空白**><**名**><**空白**><**姓**>」という形式のユーザ名を受け付けるプログラムを作成してください。ユーザが名前を入力した後は、「名」を外して「姓」だけを出力します。文字列をグローバル変数に保管します。Formulaオブジェクトを使用して、文字列を取得します。

解答 - 「文字列とグローバルの操作」

図288に、練習「文字列とグローバルの操作」の解答例を示します。

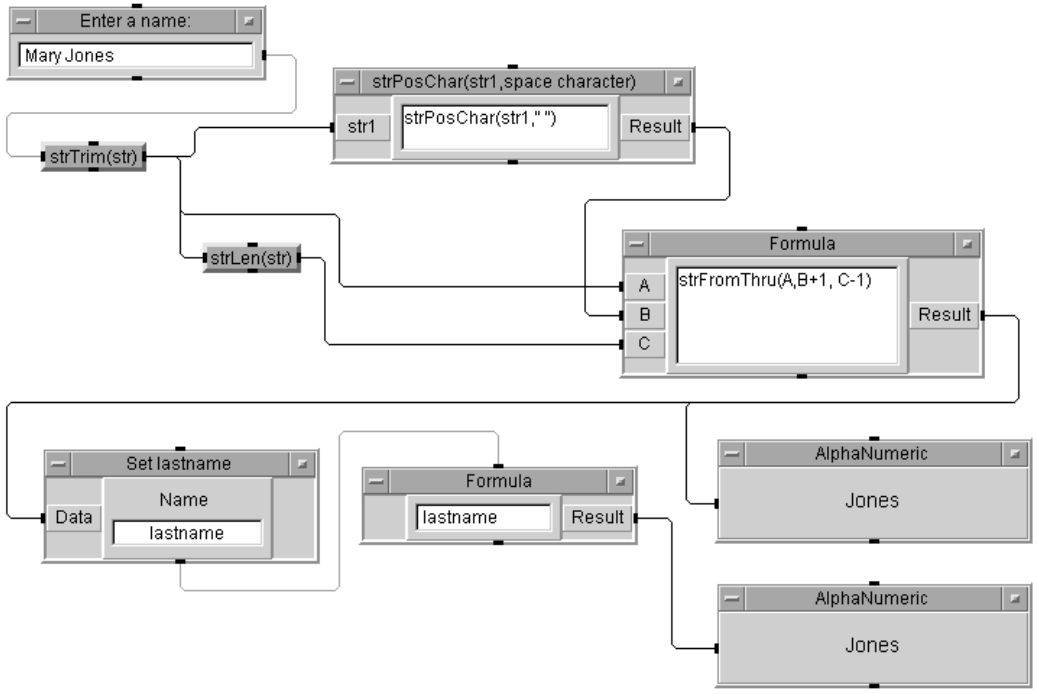


図288　　「文字列とグローバル変数の操作」

キー・ポイント

文字列のオブジェクトと関数: まず、StrTrim(str)は、名前の先頭と末尾からすべてのスペースとタブを削除します。StrPosChar(str1, " ")は、名と姓の間にあるスペースのインデックスを生成します。StrLen(str)は、文字列の長さを出力します。これらの操作は、文字列のオブジェクトを使って実行されましたが、Formulaオブジェクト内で文字列の関数を使って実行することもできます。

Formulaオブジェクト: **StrFromThru(A,B+1,C-1)**では、入力**A**から文字列を受け取り、入力**B**のスペースのインデックスに1を加算し、入力**C**の文字列の長さから1を減算しています。インデックスは0から始まることに注意してください。

Set変数: lastnameという名前のグローバル変数を簡単に設定できることに注目してください。この後、この変数は、任意の式フィールド(この例では、Formulaオブジェクト)で参照できます。

最適化の方法: 3つの式は、組み合わせて1つの式にできます。strTrim()の出力は何回か使用されるため、1つのままにしておく方が適切ですが、ほかの式は組み合わせて1つの式にすると、処理速度が上がります。ただし、多少読みにくくなります。

最適化の手法

この例題では、2つの異なる方法でVEEプログラムを構築し、その実行速度の差に注目します。

「最適化の手法」ステップ1

sin関数とcos関数の両方を介して**0〜710(ステップ10)**の範囲を送信するプログラムを作成してください。関数の結果は、**X vs.Y**表示に出力します。プログラムの実行にかかる時間を測定するには、Timerオブジェクトを使用します。[Trig Mode]のデフォルトを[Radians]に設定します。

解答 -「最適化の手法」ステップ1

図289に、ステップ1の解答例を示します。

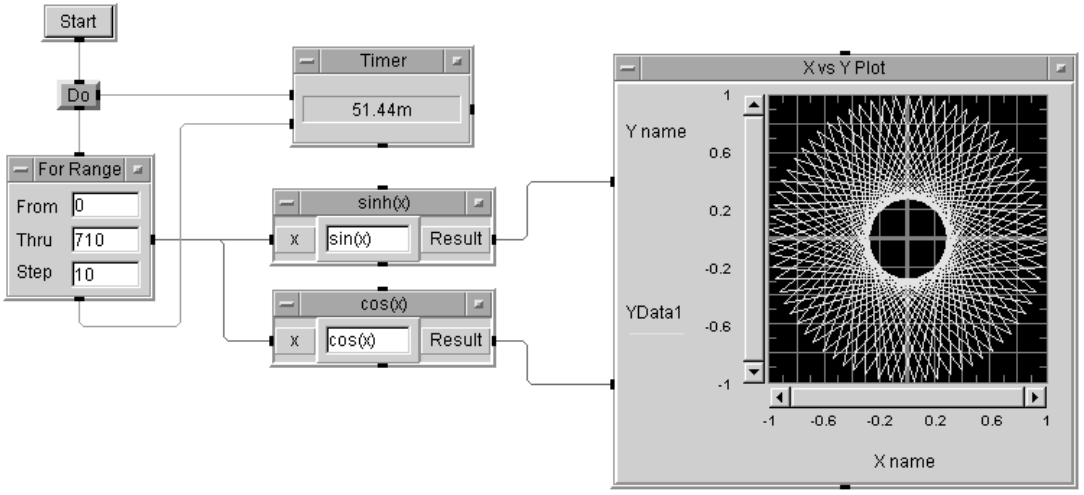


図289　「VEEプログラムの最適化」ステップ1

「最適化の手法」ステップ2

最初のプログラムにあるすべてのオブジェクトをクローンします。範囲を配列内に収集するように新しいセットを変更します。これで、sinとcos関数は点の配列に対して実行され、一度だけプロットされます。どれだけ時間が節約されるかに注目してください。

解答 -「最適化の手法」ステップ2

図290に、ステップ2の解答例を示します。

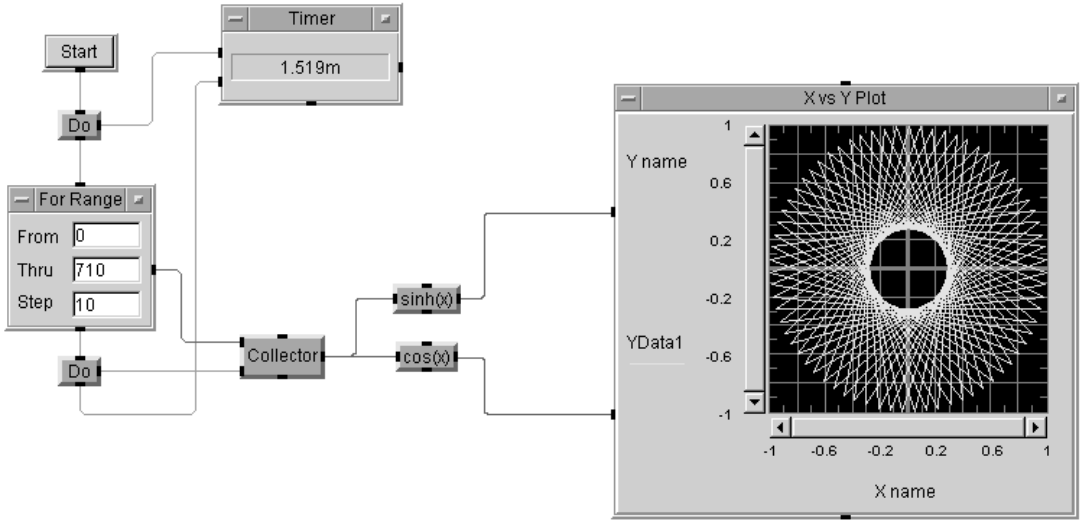


図290　「VEEプログラムの最適化」ステップ2

キー・ポイント

配列を使った最適化:　配列を使用したことで、ステップ1とステップ2の間でパフォーマンスが向上していることに注目してください。できるかぎりスカラ値ではなく配列を使用して、結果の解析や表示を行ってください。

X vs. Y表示:　この例では、WaveformやXY表示のかわりに、この表示を使用しています。それは、XとYにデータが分かれているからです。

UserObject

「不規則ノイズUserObject」

「不規則ノイズUserObject」ステップ1

不規則ノイズの波形を生成するUserObjectを作成してください。ノイズの波形とノイズ・スペクトルをUserObjectの外に表示します。UserObjectの外に、amplitude、number of points、interval(time span)、DC offsetの各コントロールを用意します。

注 記

UserObjectの中で仮想ソースを使用しないでください。Use objectの作成には、Build WaveformやRandomなどのオブジェクトを使用します。

解答 - 「不規則ノイズUserObject」

図291に、「不規則ノイズUserObject」の解答例を示します。

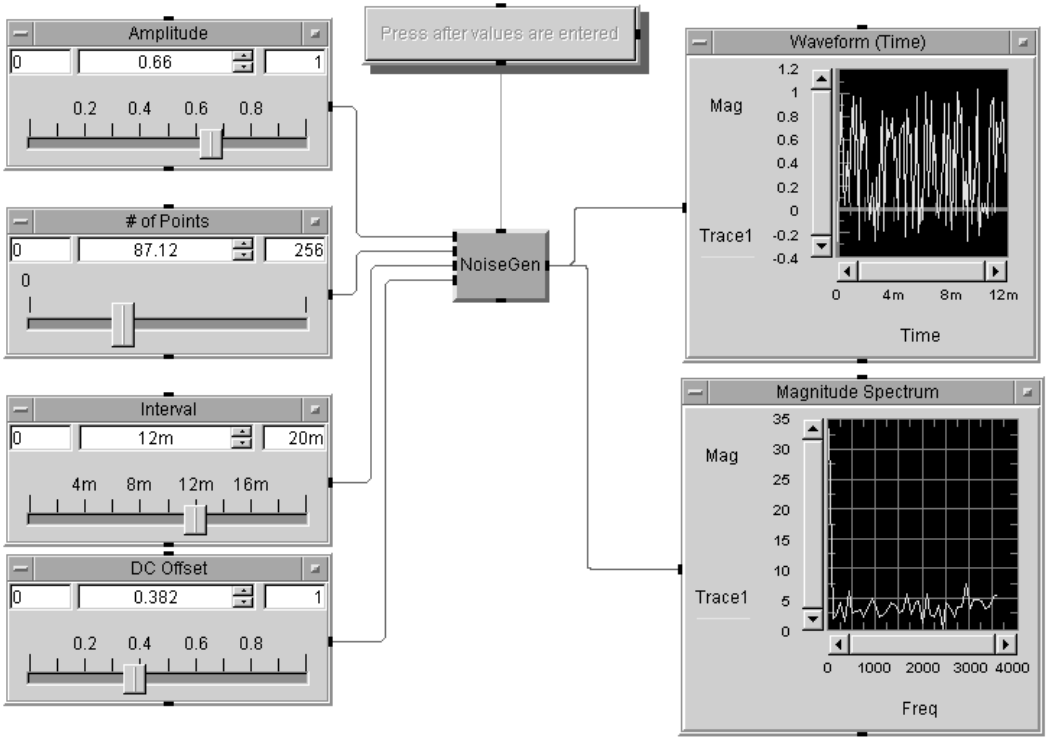


図291　　「不規則ノイズUserObject」

解答 - 「不規則ノイズのNoiseGenオブジェクト」

図292に、「NoiseGen UserObject」の解答例を示します。

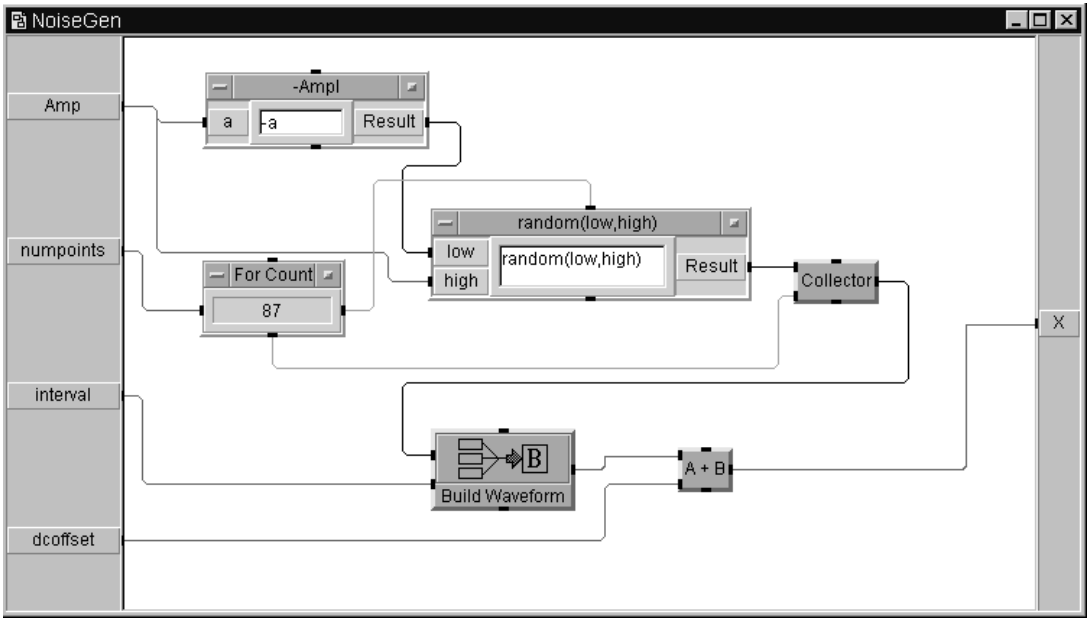


図292　「NoiseGen UserObject」

キー・ポイント

UserObject:　UserObjectは、本質的に、ユーザがカスタマイズしてVEEに追加するオブジェクトです。

Build Waveform:　このオブジェクトは、振幅値のReal配列とtime span (yデータがサンプリングされる秒単位の周期)からWaveformデータ型を作成します。

Agilent VEE UserFunction

UserFunctionの使用方法

「UserFunction」ステップ1

スライダから振幅値(0～1)を受け入れ、ノイズ波形を返す、NoiseGenという名前の関数を作成してください。

次は使用しません。

```
Virtual Source
For Count
For Range
```

次を使用してください。

```
Formula
Ramp
Build Waveform
```

ヒント: randomize(array, -a,a)を使用します。ここで、配列の要素は256であり、**a**は振幅です。この関数を呼び出す簡単なメイン・プログラムを構築して、それが正しく機能していることを確認してください。

解答 - 「UserFunction」ステップ1

図293に、ステップ1の解答例を示します。

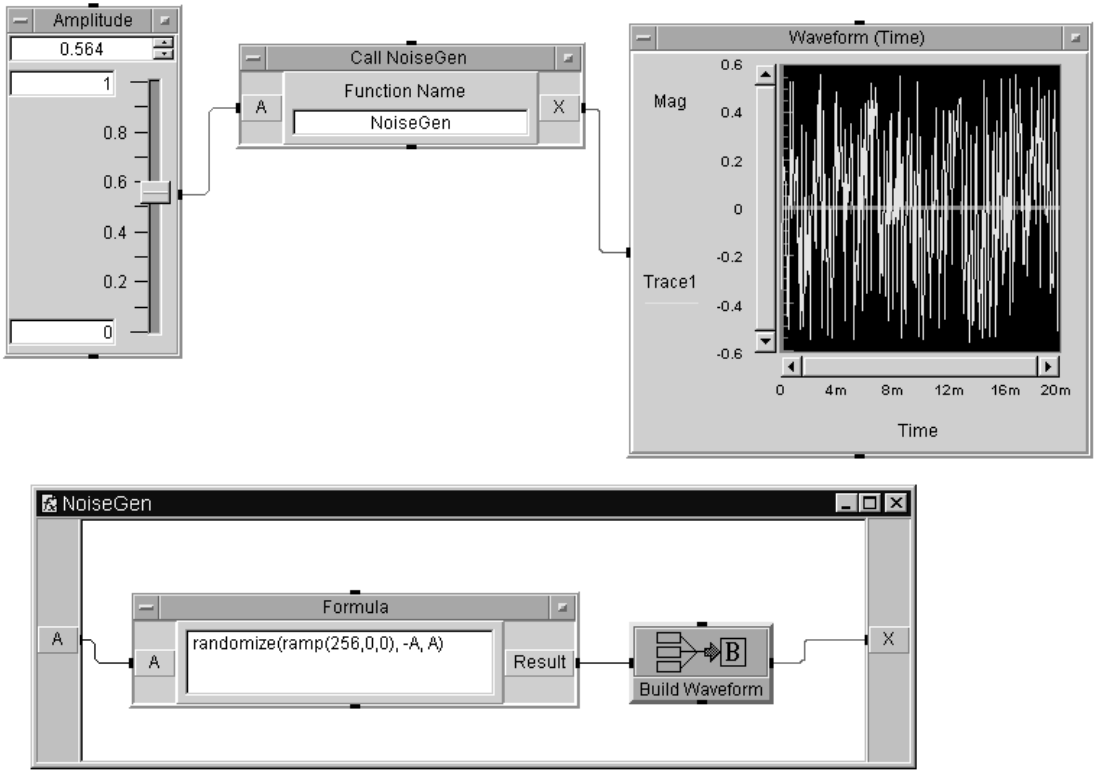


図293　「User Function」ステップ1

キー・ポイント

Ramp():　256の点を持つ配列を生成するために、randomize()のパラメータ・リスト内でramp()関数を使用しています。

Build Waveform:　デフォルトのタイム・スパンは20ミリ秒であり、このオブジェクトに配列を送信するだけで、波形を構築できます。

「UserFunction」ステップ2

同じプログラム内で、最初の関数NoiseGenを呼び出すAddNoiseという名前の別の関数を作成します。AddNoiseは、NoiseGen関数のノイズ波形を正弦波に追加するものです。AddNoiseは、NoiseGenの振幅と正弦波の2つの入力を持つ必要があります。また、結果の出力を1つ持ちます。

ノイズ振幅用のスライダを持つ簡単なメイン・プログラムを構築し、ノイズの追加先の波形として**[Virtual Source]** ⇒ **[Function Generator (sine wave, Freq = 100 Hz)]**を選択します。結果の波形を表示します。

解答 -「UserFunction」ステップ2

図294に、ステップ2の解答例を示します。

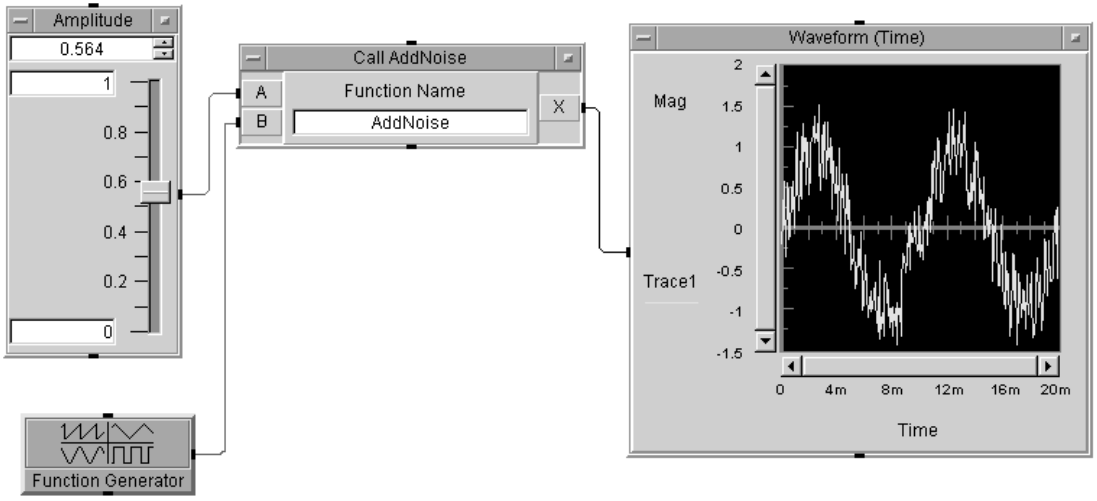


図294　「User Function」ステップ2

「UserFunction」ステップ3

同じプログラム内で、今回はFormulaオブジェクトから再びAddNoise関数を呼び出し、その結果の絶対値を受け取ります。その絶対値波形を同じ表示上に表示します。次に、AddNoise関数を編集する準備をします。**[Debug]** ⇒ **[Show Data Flow]**をオンにします。AddNoiseウィンドウを開いたまま、プログラムを実行します。こうするとデバックにたいへん便利であることに注目してください。

解答 - 「UserFunction」ステップ3

図295に、ステップ3の解答例を示します。

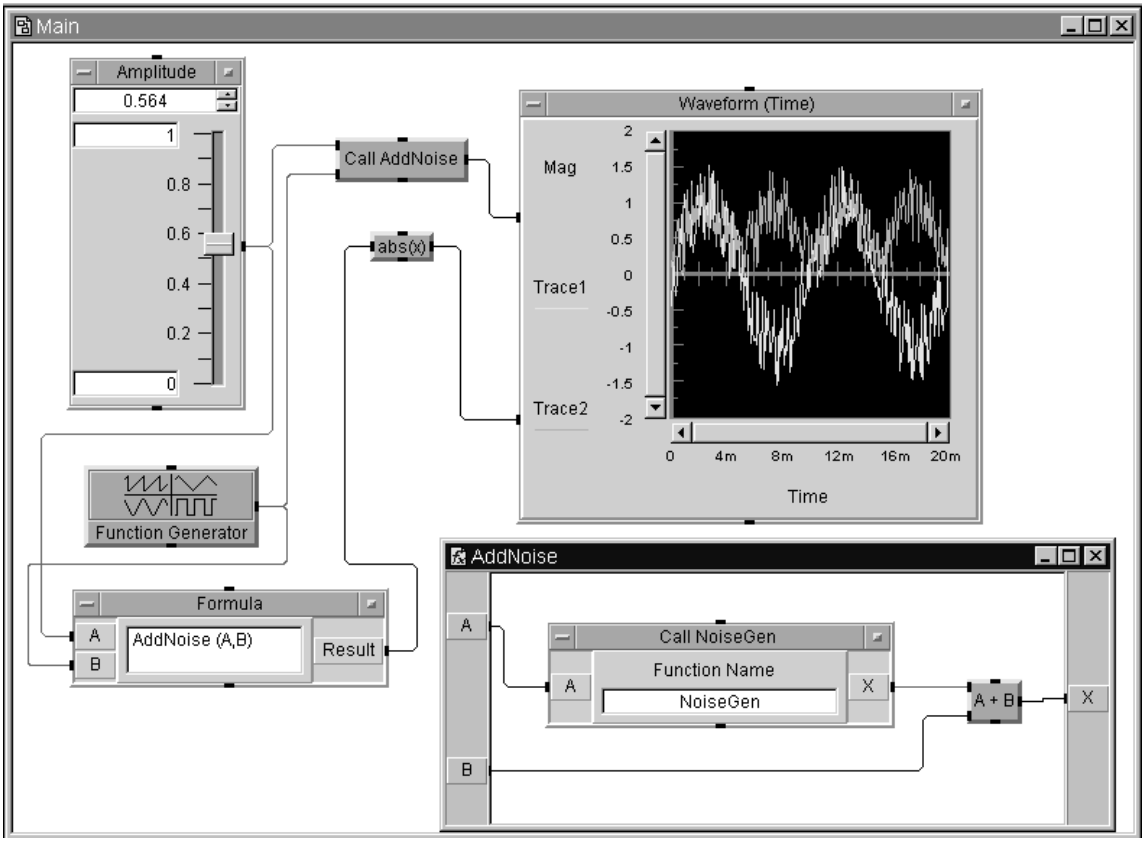


図295　「User Function」ステップ3

「UserFunction」ステップ4

スライダでAmplitudeという名前のグローバル変数を設定するように、プログラムを変更します。NoiseGen関数でそのグローバルを使用します。したがって、NoiseGenに入力ピンは必要なくなります。プログラムを正しく実行します。このファイルを**uflab.vee**という名前で保存します。

解答 -「UserFunctionの使用方法」ステップ4

図296に、ステップ4の解答例を示します。

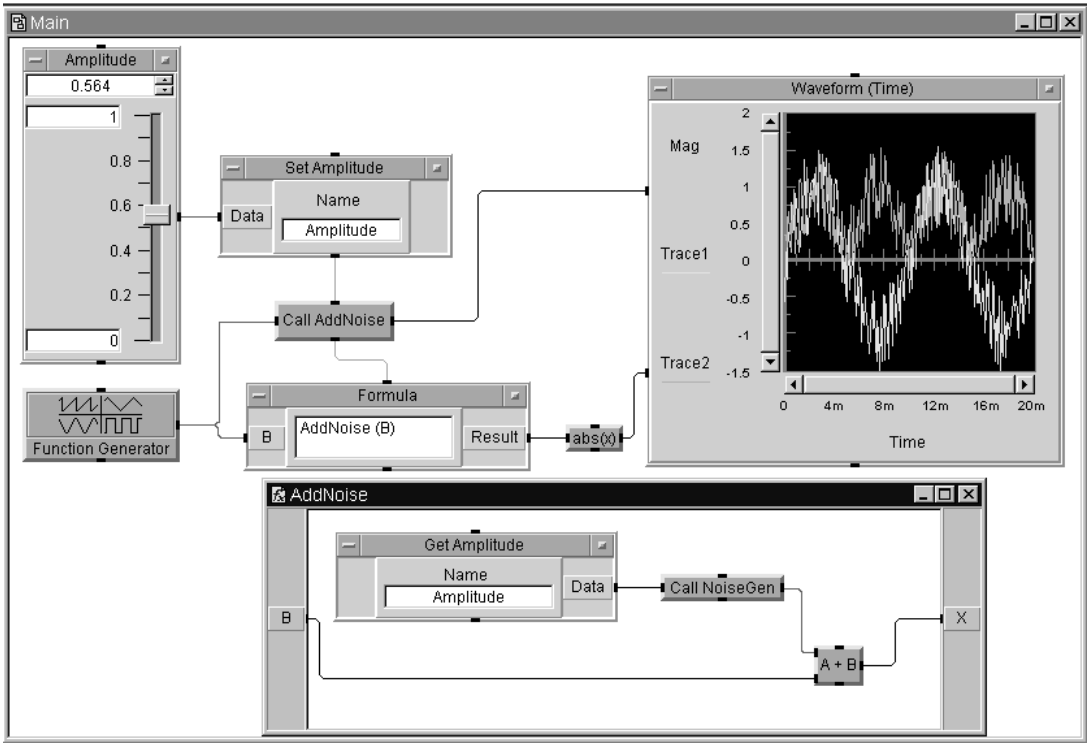


図296　「User Function」ステップ4

ヒント: Call AddNoiseとFormulaオブジェクトはグローバルamplitudeを使用するので、それらのオブジェクトは、どちらもSet Amplitudeオブジェクトの実行後に実行される必要があります。シーケンス・ピンをSet AmplitudeからCall AddNoiseに、またCall AddNoiseからFormulaに接続すると、これらのオブジェクトが必要な順序で実行されます。

UserFunctionのライブラリのインポートと削除

前の例題から**uflab.vee**をインポートする簡単なプログラムを構築します。ノイズを追加する関数を呼び出し、ライブラリをプログラムによって削除します。Callオブジェクトのオブジェクト・メニューで[Select Function]選択肢を使用します。

ヒント: Import Libraryオブジェクト・メニューの[Load Lib]をクリックして、指定したライブラリを手動でロードし、CallでSelect Function機能が使用できるようにします。

解答-プログラムによるライブラリのインポートと削除

図297に、ライブラリをプログラムにより削除するための解答例を示します。

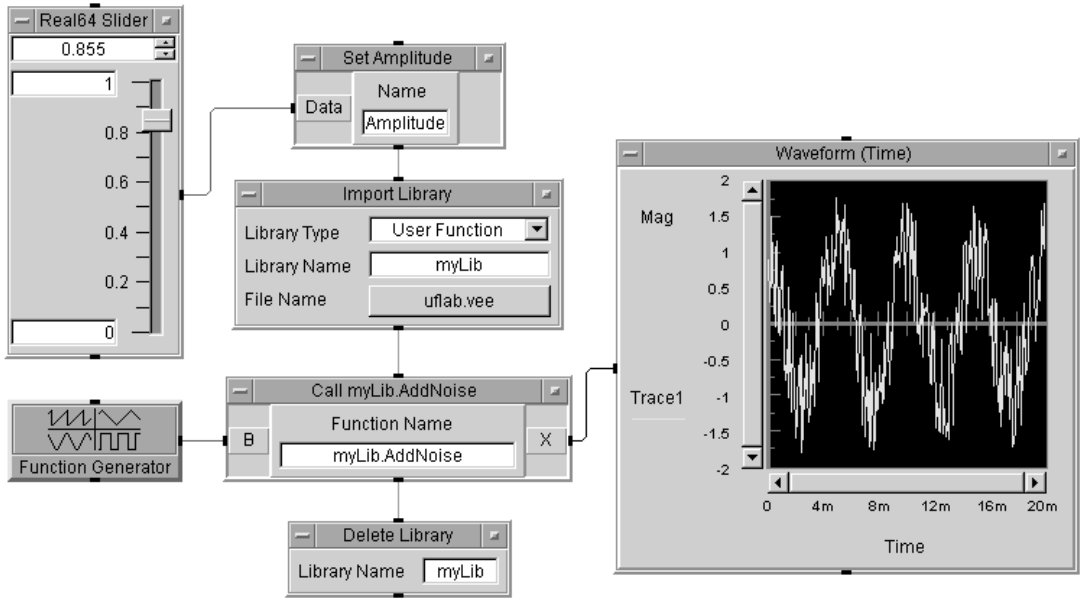


図297　　ライブラリのインポートと削除

キー・ポイント

Select Function: この選択により、選択した関数に適切な入力ピンと出力ピンが設定されます。

UserFunctionを編集する: UserFunctionのライブラリをプログラムによりインポートすると、UserFunctionを編集することができません。表示やデバッグのためのブレークポイントの設定は行えます。インポートしたUserFunctionを編集したい場合は、Merge Libraryコマンドを使用します。

変数設定の注意事項: ある関数でグローバル変数を使用したら、他のプログラムでその関数を使用するときにそのグローバルを忘れずに作成する必要があります。入力と出力の明示的作成には、追跡が容易であるという利点があります。

オペレータ・パネルとポップアップの作成

「オペレータ・パネルとポップアップの作成」ステップ1

オペレータに数字の入力を求めるパネルを作成してください。オペレータと対話するためのUserObjectを作成します。オペレータに、**A**と**B**の2つの入力を要求します。両方の入力を表示に送信します。[Show On Execute]をチェックしてUserObjectを使用すると、パネルを表示できます。

解答 - 「オペレータ・パネルとポップアップの作成」ステップ1

図298に、詳細ビューでの解答例を示します。図299は、プログラムが実行されたときに表示されるパネルを示します。

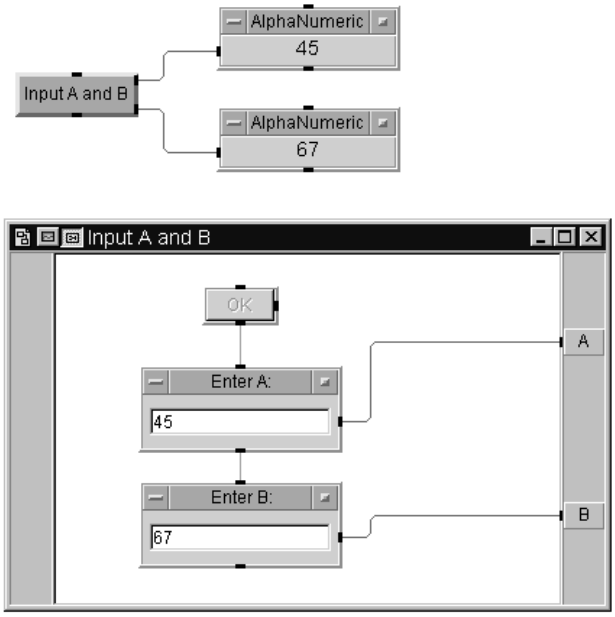


図298　　オペレータにAとBの入力を求めるUserObject

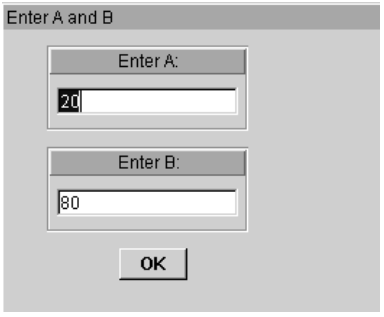


図299　　オペレータがAとBを入力するためのパネル

キー・ポイント

UserObjectのプロパティ： [UserObject Properties]ダイアログ・ボックスで、[Pop-Up Panel]を選択し、[Show Panel On Execute]をクリックしてオンにします。**[Pop-Up Panel]** ⇒ **[Panel Title]**で名前を"Enter A or B"に変更します。

「オペレータ・パネルとポップアップの作成」ステップ2

2つの数が異なる場合は、**A**と**B**の両方を表示するかわりに、オペレータに**A**と**B**のどちらを表示するかをたずねてください。2つの値の入力を求めた後、**A**と**B**の値が等しい場合は、その値を表示します。**A**と**B**の値が異なる場合は、表示する値の選択をオペレータに求めます。オペレータの選択に基づいて、**A**または**B**を表示します。

ヒント: [Show Panel on Execute]を設定したポップアップ・パネルを持つ別のUserObjectを追加し、そこでオペレータに値をたずねます。

解答 - 「オペレータ・パネルとポップアップの作成」ステップ2

図300に、**A**と**B**が異なる数である場合に、オペレータに選択を求めるUserObjectを示します。図301は、**A**と**B**のどちらを表示するかをオペレータにたずねる2つ目のポップアップ・パネルが表示されたところです。

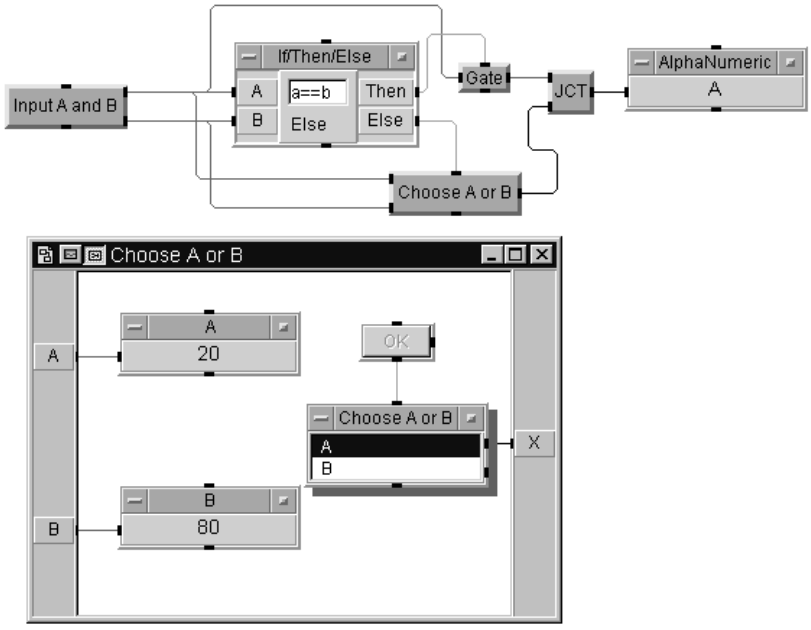


図300　オペレータにAまたはBのどちらを表示するかをたずねるUserObject

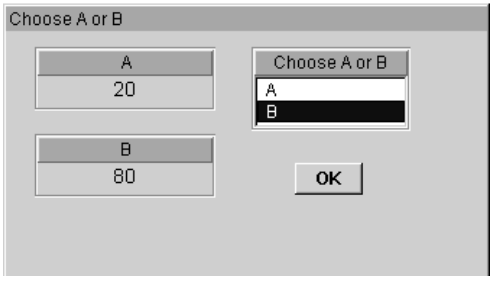


図301　AとBのどちらを表示するかをオペレータが選択するためのパネル

キー・ポイント

Gate: Gateオブジェクトは、2つの数が等しい場合に、単に一方の値を送信します。

結合: JCTオブジェクトは、オブジェクトAlphanumericに対する複数の入力を可能にします。JCTオブジェクトは、配線によるORオブジェクトです。

オブジェクト・リストのメニュー: 2つの選択肢とリストが配置されるように編集された**[Data]** ⇒ **[Selection Controls]** ⇒ **[List]**オブジェクトの使用方法に注意してください。この設定により、テキスト**A**または**B**が出力されます。序数値 (0または1)が必要な場合は、かわりにListオブジェクトの序数データ出力を使用します。

「オペレータ・パネルとポップアップの作成」ステップ3

オペレータが数を入力しなかった場合に、エラー・メッセージを生成します。2つの数が異なる場合に、**A**と**B**のどちらを表示するかをオペレータにたずねる2つ目のUserObjectにエラーを追加します。オペレータが10秒以内に**A**か**B**を選択しなかった場合は、エラーを生成します。

解答 - 「オペレータ・パネルとポップアップの作成」ステップ3

図302に、オペレータが10秒以内に**A**か**B**を選択しなかった場合にエラーを生成するように変更されたUserObjectを示します。

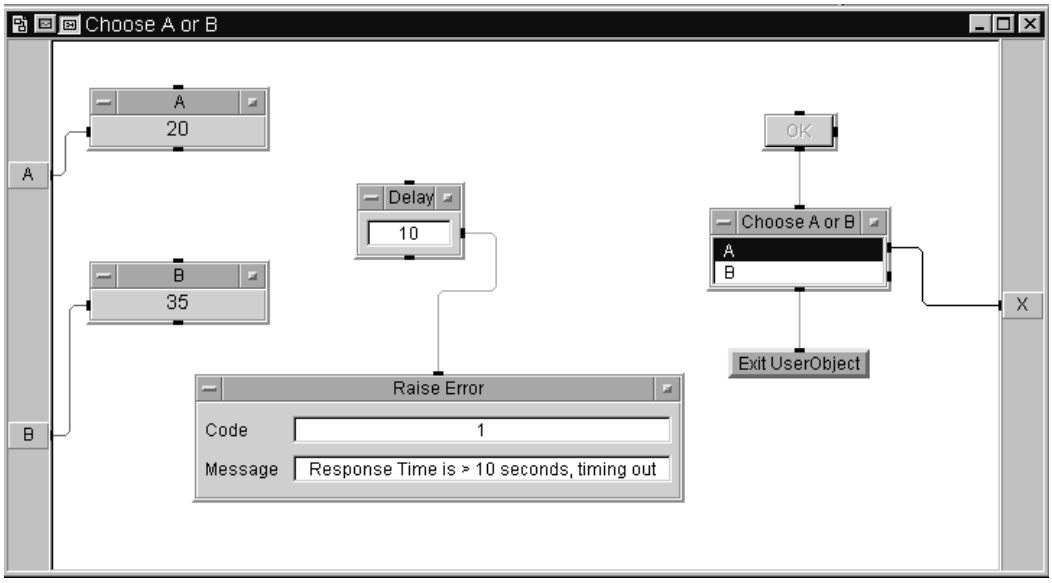


図302　　オペレータが選択を入力しなかった場合にエラーを生成

キー・ポイント

Exit UserObject:　ユーザが10秒以内に応答した場合、Delayオブジェクトがまだ実行を完了していなくても、このオブジェクトはUserObjectを終了させます。

DelayとRaise Error:　DelayオブジェクトがRaise Errorオブジェクトを起動してから10秒後に、Raise Errorオブジェクトはプログラムの実行を休止し、指定されたエラー・メッセージを表示します。エラーを起こしたオブジェクトの周りには赤の輪郭線も表示されますが、メイン・メニュー・バーの**[Stop]**または**[Run]**ボタンをクリックすると消えます。

OKとDelay:　**AorB**内の2つのスレッドは独立しているため、**OK**と**Delay**の両方が同時に実行されることに注意してください。

ファイルの扱い方

ファイルとの間のデータの移動

ファイルに現在の時刻を書き込むVEEプログラムを作成します。100個の不規則な数を生成し、そのファイルに書き込みます。それらの平均と標準偏差を計算し、次の形式でファイルの末尾に追加します。

Mean: xxxxxx

Std Dev: yyyyyy

次に、ファイルから平均と標準偏差だけを読み取ります。図303に、ファイルとの間でデータを移動する方法を示します。

解答 -「ファイルとの間のデータの移動」

図303に、「ファイルとの間のデータの移動」の解答例を示します。

.

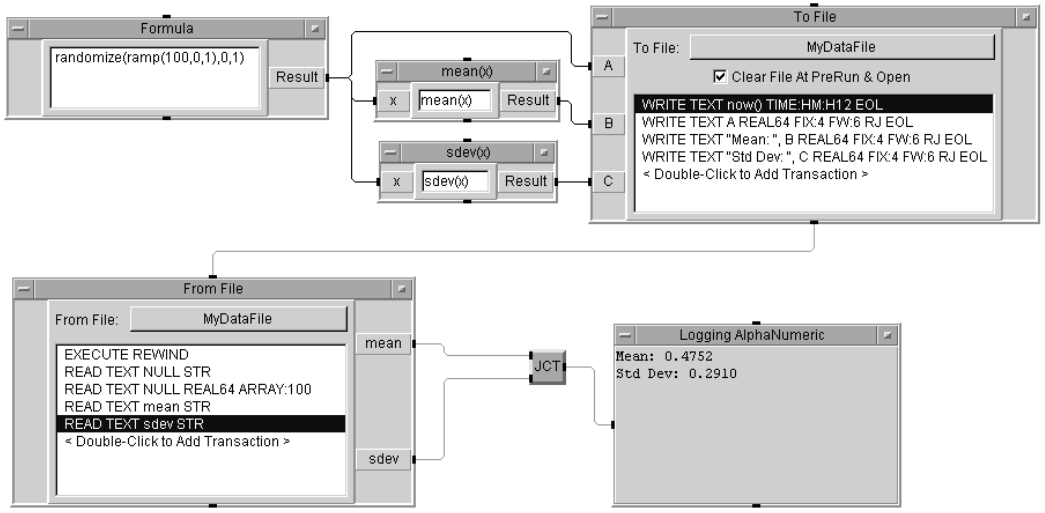


図303　　ファイルとの間のデータの移動

キー・ポイント

配列の生成: Formulaオブジェクト内で**randomize(ramp(100,0,1), 0, 1)** を使用して、100個の乱数の配列を作成します。ramp() 関数は、1次元配列を生成し、それをrandomize() 関数に渡します。次に、randomize() が0～1の乱数値を生成します。

タイム・スタンプ: To Fileオブジェクトのトランザクション1に対する[I/O Transaction]ダイアログ・ボックスの式フィールドで、now() 関数が使用されています。フォーマットを`TIME STAMP FORMAT`に変更した場合、ダイアログ・ボックスには、時刻を保存する方法を指定するための追加ボタンが表示されます。

1行に2つの値を保管する: **To File**オブジェクトの3番目と4番目のトランザクションでは、定数のText文字列の後にReal値を保存します。たとえば、3番目のトランザクションでは、**[I/O Transaction]**ダイアログ・ボックスの式フィールドに「`"Mean: ",B`」と入力します(平均値が**B**入力ピンにある場合)。

ファイルから値を抽出する: 平均と標準偏差を取得するには、まず、読み取りポインタを先頭に置くため、`EXECUTE REWIND`を送信します。次に、正しいフォーマットの`NULL`を使用し、タイム・スタンプとReal配列を読み飛ばします。これで、読み取った値が出力端子に入力されずに捨てられます。最後に、ファイル内の最後の2行を文字列として読み取ります。

結合: 複数の出力を単一の入力に接続する場合は、**[Flow]** ⇒ **[Junction]** オブジェクトを使用します。この例では、**mean**と**sdev**の出力を**Logging AlphaNumeric**表示に接続します。

レコード

レコードの操作

「レコードの操作」ステップ1

3つのフィールドを持つレコードを構築し、そのレコードに整数、現在の時刻の文字列、および4つのReal要素を持つ配列を1つずつ格納します。フィールドには、それぞれint、daytime、rarryという名前を付けます。このレコードと、0～1の乱数と波形を保持する別のレコードをマージします。これらのフィールドには、randとwaveという名前を付けます。

解答 -「レコードの操作」ステップ1

図304に示すように、結果のレコードには5つのフィールドがあります。

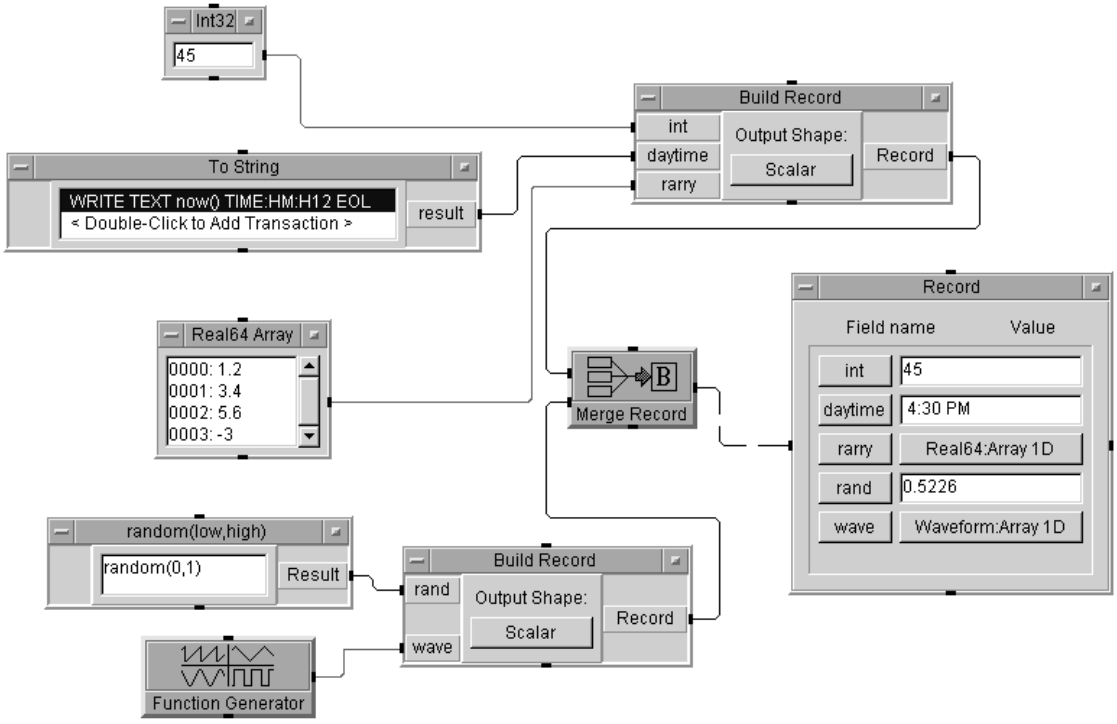


図304　　「レコードの操作」ステップ1

キー・ポイント

タイム・スタンプ:　このプログラムのためのタイム・スタンプを作成するには、To Stringオブジェクト内でnow() 関数を使用し、フォーマットを指定します。

データ定数を配列として設定する:　**[Data]** ⇒ **[Constant]** メニューにある任意のデータ型を配列にできます。それには、[Properties]を選択し、[Configuration]の[1D Array]を選択します。ここでサイズを入力できます。または、値を入力してEnterキーを押すと、続けて値を追加できます。

フィールドに名前を付ける: Build Recordオブジェクトの入力端子の名前を変更すると、レコードの特定のフィールドにint、rand、waveなどの名前を付けることができます。

デフォルト値コントロール入力: Default Value Controlピンを追加すると、Record Constantは、優れた対話型表示オブジェクトとなります。Record Constantは、自分が受信したレコードで自分を自動的に設定します。

「レコードの操作」ステップ2

Formulaオブジェクト内で条件式を使用して、レコード内の乱数を判定し、値またはテキスト文字列のどちらかを表示します。値が0.5より小さい場合はその乱数値を表示し、そうでない場合は、テキスト文字列**"More than 0.5"**を出力します。次に、時刻と波形だけを抽出します。

ヒント: 時刻と波形を抽出する場合に、Formulaオブジェクトを使用しません。AlphaNumericオブジェクトを使用して、このレコードを表示します。

解答 - 「レコードの操作」ステップ2

図305に、「レコードの操作」ステップ2の解答例を示します。

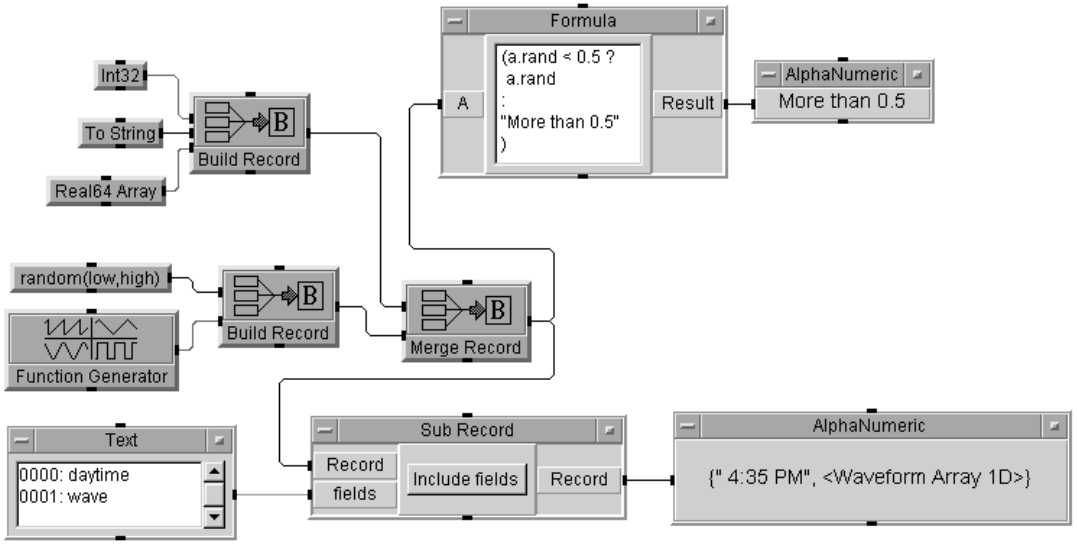


図305　　「レコードの操作」ステップ2

キー・ポイント

条件式の使用方法:　VEEは条件式をサポートしており、if-then-else操作を効率的に実装できます。このFormulaオブジェクト内の条件式(`a.rand < 0.5? a.rand: "More than 0.5"`)は、3項式と呼ばれ、これ全体が1つの式です。図に示されているように、改行を入れながらFormulaオブジェクト内に記述できることに注目してください。Formulaオブジェクト内に複数の式がある場合、それらの式どうしはセミコロン(`;`)で区切られます。

Sub Recordオブジェクト:　[fields]というラベルが付いたSub Recordの入力ピンに接続されているText配列のフィールドに注目してください。指定されたフィールドを保持するようにSub Recordオブジェクト設定した場合、このオブジェクトは、それらのフィールドだけを保持するレコードを出力します。

「レコードの操作」ステップ3

1番目のフィールドの整数入力をFor Countオブジェクトに置換し、10回の反復実行を行います。各反復で、乱数生成と時刻の関数を起動します。完成したレコードをTo DataSetオブジェクトに送信します。別のスレッドで、データセット内のレコードのうち、乱数値が0.5より大きいレコードをすべて読み取ります。読み取ったレコードをレコード定数に入力します。

ヒント: Record Constantオブジェクトには、Default Valueのための制御ピンが必要です。

解答 - 「レコードの操作」ステップ3

図306 に、「レコードの操作」ステップ3の解答例を示します。

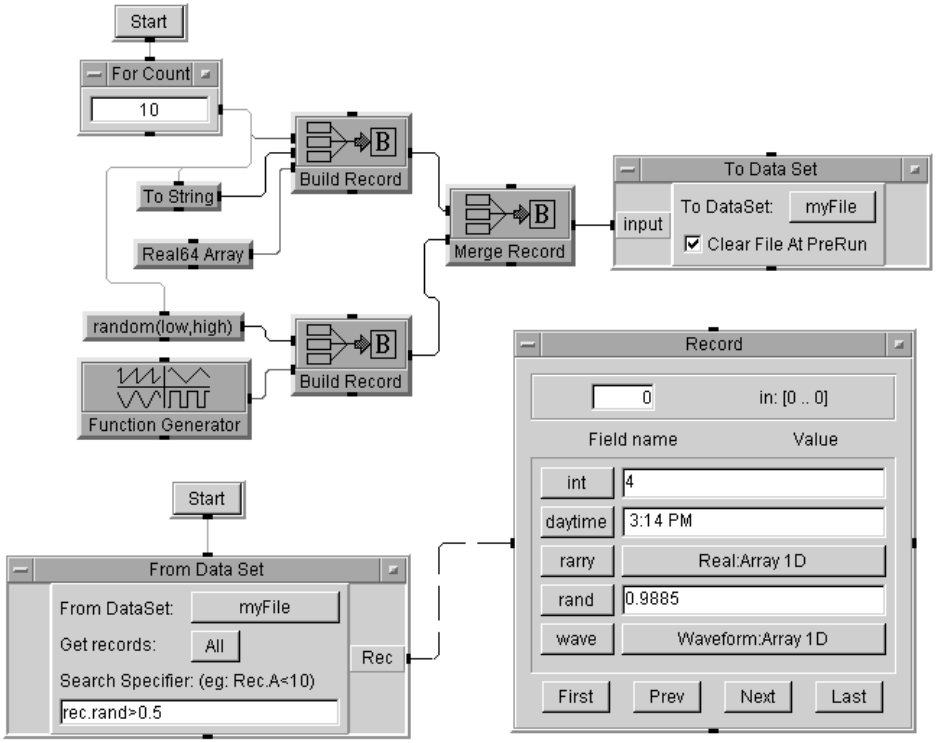


図306　　「レコードの操作」ステップ3

キー・ポイント

To DataSetオブジェクト: [Clear File at PreRun]オプションは、データがファイルに初めて送信される前に、ファイルをクリアします。プログラムは、10個の異なるレコードを同じファイルに続けて送信し、それらのレコードはファイルに追加されていきます。

From DataSetオブジェクト: このオブジェクトは、randフィールドが0.5より大きいレコードをすべて読み取るように設定されます。この例では、10個のレコードのうち5個のレコードが条件を満たし、最初のレコードがインデックス番号0として表示されています。

テスト・シーケンスの作成方法

「テスト・シーケンサの使用」ステップ1

UpperLimitという単純なUserFunctionを作成します。これは、Real64 SliderとConfirm (OK)オブジェクトを持つポップアップ・パネルです。スライダの出力をUpLimitというグローバル変数と、出力端子に送信します。Sequencerオブジェクトを作成し、Sequencerに**test1**を、UpperLimitsを呼び出すEXECトランザクションとして設定します。

呼び出すテストをシミュレートする、AddRandと呼ばれる別の関数を作成します。この関数は、入力値をランダム値(0～1)に追加します。関数は、1個の入力ピンと1個の出力ピンを持ちます。

Sequencerから、AddRandを呼び出すため**test2**を作成し、ゼロを送信します。グローバルUpLimit値未満であるかリミット比較を実行するため、戻り値をテストします。合格の場合は、`"PASS" + test2.result`を返します。不合格の場合は、`"FAILED" + test2.result`を返します。Alphanumeric表示をSequencerのReturnピンに加えます。

Sequencerオブジェクトの後、Get Variableオブジェクト(UpLimit)および別のAlphanumeric表示をpingします。プログラムを数回実行します。

解答 - 「テスト・シーケンサの使用」ステップ1

図307に、Sequencerの使用の最初のステップに対する解答を示します。

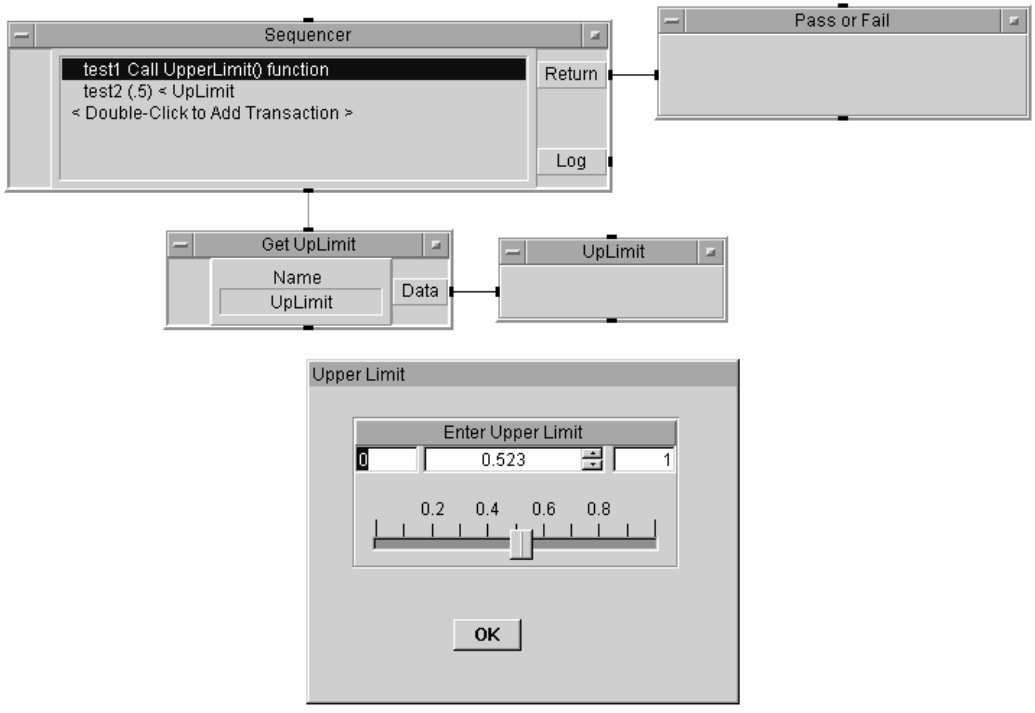


図307　　「テスト・シーケンサの使用」ステップ1

キー・ポイント

UserFunctionを使ってグローバル変数を設定する:　最初のSequencerトランザクションは通常、この実行例のように、グローバル変数を設定するUserFunctionの呼び出しに使用されます。これらの変数は、ここで示すように、後続の任意のテストで利用できるようになります。

SequencerのReturnピン:　この例ではReturnピンが、PASSまたはFAILメッセージと、テストの値を提供します。Returnピンを使って、特定のテストからの任意のメッセージまたは値を提供できます。

「テスト・シーケンサの使用」ステップ2

最初のテストを無効にします。グローバルがもうどこにも必要ないと仮定すると、UpperLimit関数を直接呼び出すことができます。test2を変更し、AddRand(0)の戻り値をUpperLimit関数の結果と比較するようにします。

ヒント: 最初のテストを無効にするため、図308に示すような[Sequencer Transaction]ボックスを使用します。

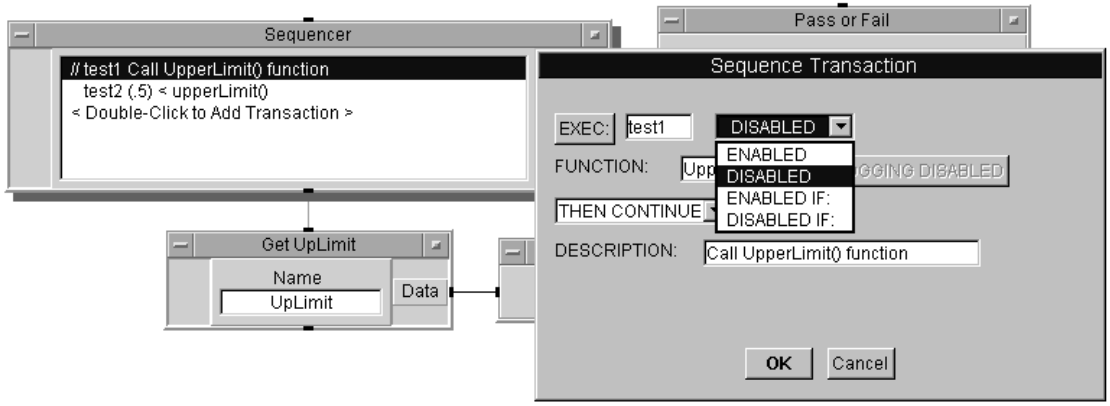


図308　　シーケンスの最初のテストを無効にする

図308で、無効になっていることを示すため、Sequencerの最初のテストが2個のスラッシュでコメントアウトされています。

解答 - 「テスト・シーケンサの使用」ステップ2

図309に、「テスト・シーケンサの使用」ステップ2に対する解答を示します。

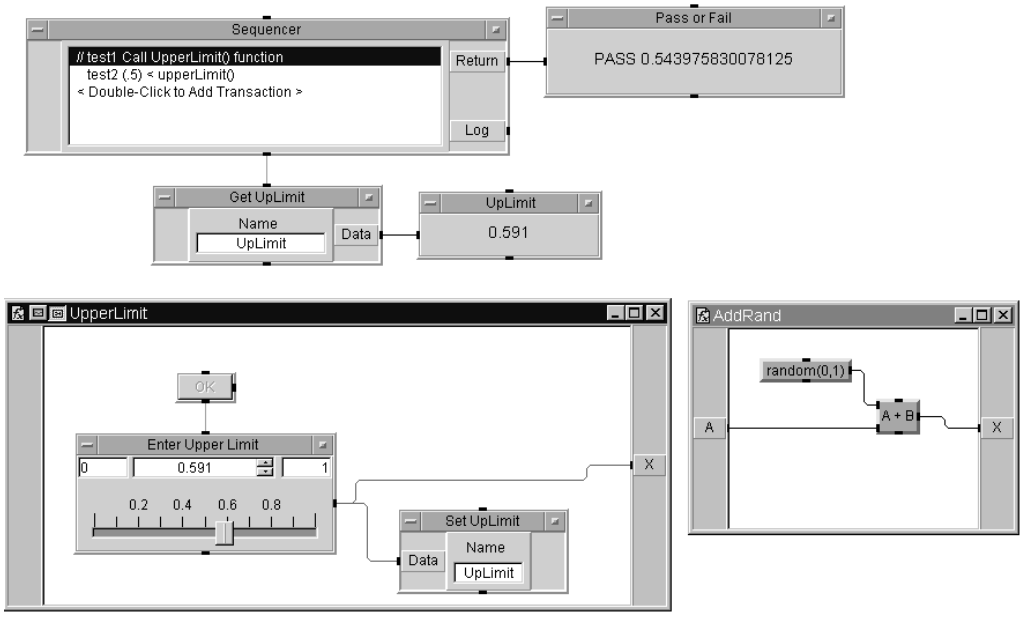


図309　　「テスト・シーケンサの使用」ステップ2

キー・ポイント

[Expression]フィールドのUserFunction:　この例では、テスト結果をUpLimit変数と比較するかわりに、変数が進む式フィールドに関数名UpperLimit()を入力できます。

「テスト・シーケンサの使用」ステップ3

VEE関数**random (0,1)**を呼び出す**test2 Sequencer**トランザクションを編集します。結果を`0.5`未満のリミットと比較します。4つのテストの合計が出るまで、`test1`トランザクションの切り取りと貼り付けを行います。

Sequencerを5回実行するプログラムを構築します。データをレコードのデータ・セットに記録し、配列にデータを収集します。配列を使用し、2番目のテストの結果の最小値、最大値、平均値、標準偏差を見つけます。

解答 - 「テスト・シーケンサの使用」ステップ3

図310に、ステップ3に対する解答を示します。

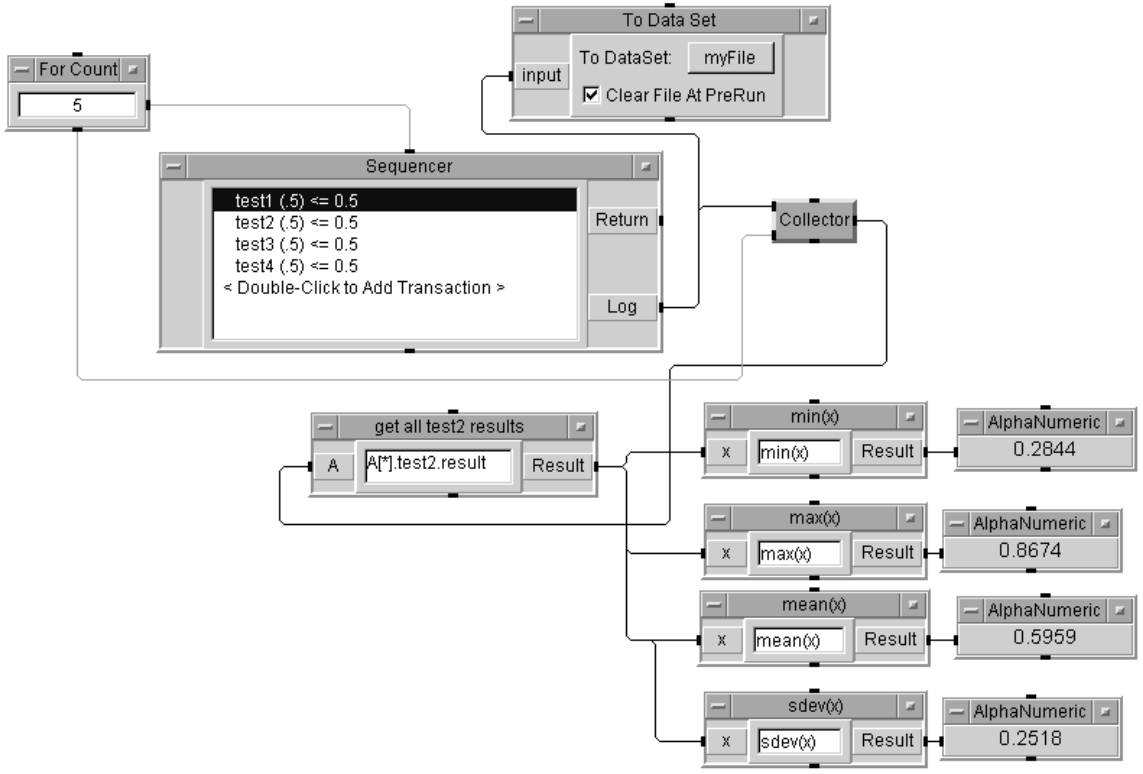


図310　　「テスト・シーケンサの使用」ステップ3

キー・ポイント

Sequencerの複数実行用のデータ・フォーマット(最初のスレッド):

Sequencerを1回実行すると、Record of Records(レコードから成るレコード)が出力されます。最初のレコードにはテスト名と一致したフィールド名が記録され、各フィールドに、特定テストのさまざまなデータ部分を含んだ1個のレコードが保持されます。Sequencerを複数回実行すると、各Record of Records(レコードから成るレコード)を配列に追加して、配列を

調査することができます。Formulaオブジェクトで<**レコード**>/*/.<**レコード**>.<**フィールド**>フォーマットを使用すると、1個のデータ配列を入手できます。この場合、test2の5回の実行のテスト結果を提供する実数値の配列が得られます。次に、この配列から最小値、最大値、平均値、標準偏差を計算できます。レコードから成るレコードの配列にある特定の要素を指定することにより、test2の1回の実行を指定できます。たとえば、`test2`の最初の実行結果を得るには、式`A[0].test2.result`を使用します。

「テスト・シーケンサの使用方法」ステップ4

記録レコードにタイム・スタンプ・フィールドを追加します。各ステップの実行間隔が1秒となるよう、遅延を追加します。別のスレッドでtest2のすべての結果を入手し、それらをレコード定数に送信します。

Delayオブジェクト(最初のスレッド): このオブジェクトは、指定した秒数のあいだ実行フローを保持します。ここでは、Sequencerの各実行間でタイム・スタンプ値を変えるために使用しています。

Time Stampを追加する: タイム・スタンプを追加するには、Sequencerのオブジェクト・メニューを開き、**[Properties]** ⇒ **[Logging]**タブを選択して、**[Record Fields to Log]** ⇒ **[Time Stamp]** をチェックします。図311に、**[Properties]** ⇒ **[Logging]**タブのダイアログを示します。

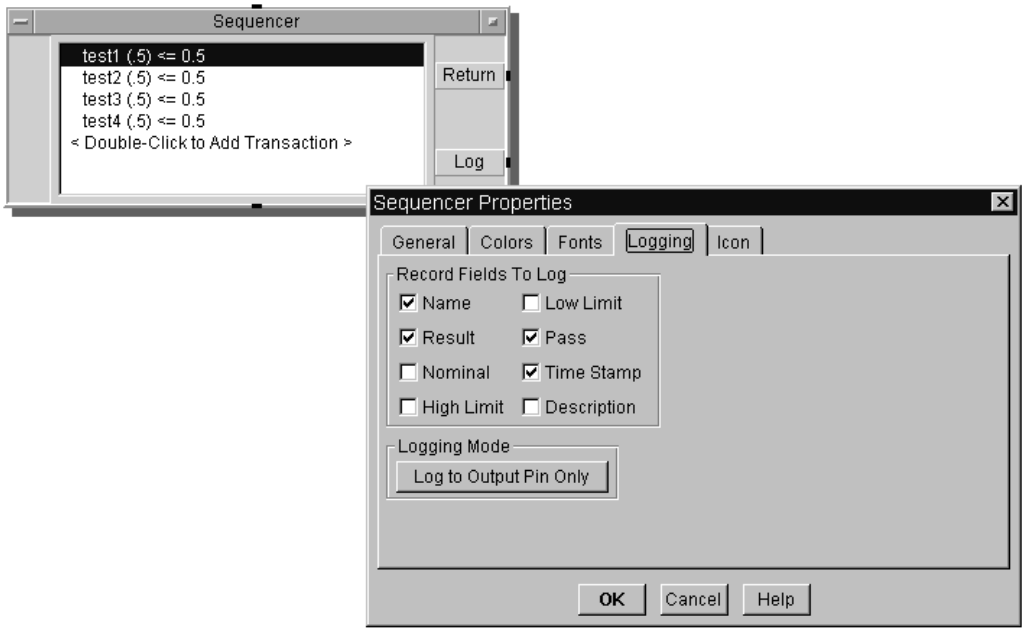


図311　　ログとして記録するレコードへのTime Stampの追加

解答 - 「テスト・シーケンサの使用」ステップ4

図312に、テスト・シーケンサの使用のステップ4に対する解答を示します。

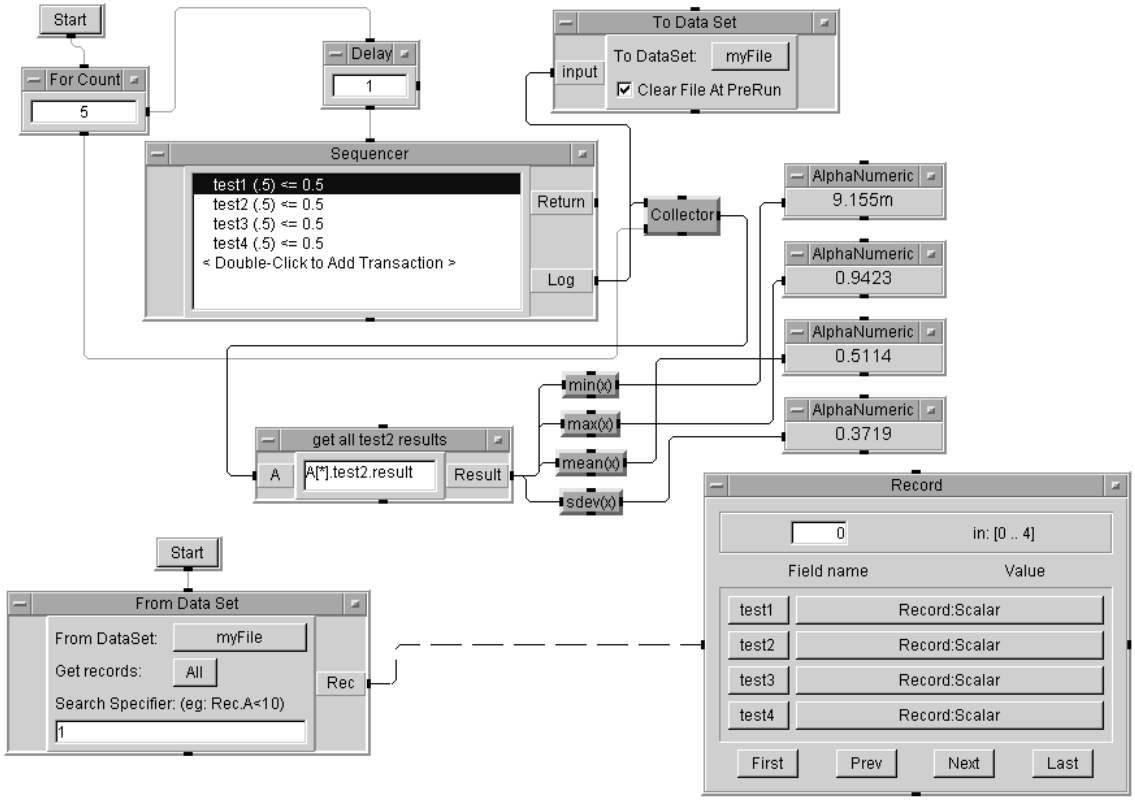


図312　　「テスト・シーケンサの使用」ステップ4

ヒント: レコードを表示するには、いずれかのテストの**[Record]** ⇒ **[Record: Scaler]**フィールドをクリックします。[Record Field Data]ダイアログ・ボックスが表示されます。図313に、[Record Field Data]ダイアログ・ボックスを示します。

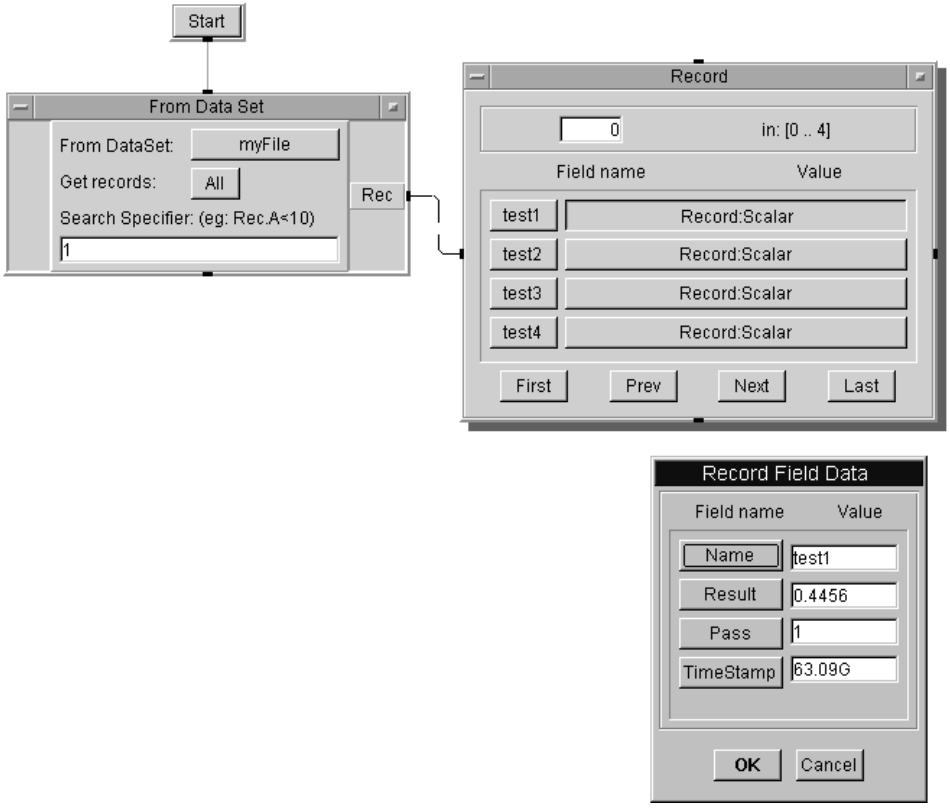


図313　　レコードのチェック

「テスト・シーケンサの使用」ステップ5

Logging Alphanumeric表示のレコードからタイム・スタンプのフィールドを印刷します。

ヒント: 4個のFormulaオブジェクト(各テストに1つ)を使用します。4つのFormula結果すべてを1つのLogging Alphanumeric表示に表示するには、Junctionオブジェクトを追加します。To Stringを使用して、63Gタイム・スタンプ値をより読みやすい文字列にフォーマットします。

解答 - 「テスト・シーケンサの使用」ステップ5

図314に、タイム・スタンプを「テスト・シーケンサの使用」ステップ5の表示に印刷するためのプログラム・スレッドを示します。

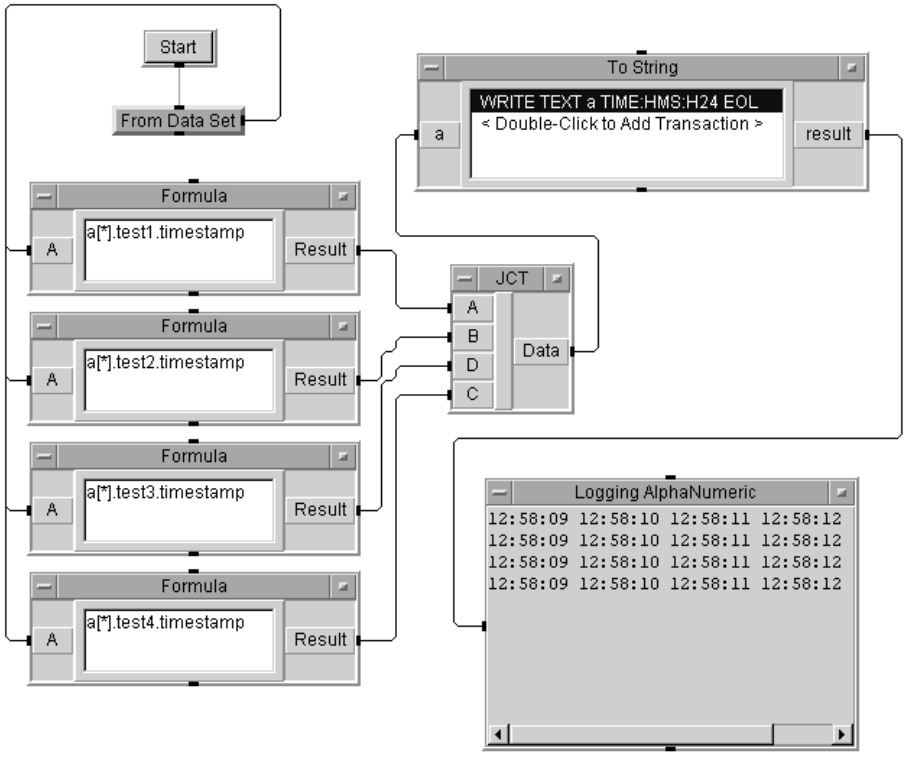


図314　　「テスト・シーケンサの使用」ステップ5

キー・ポイント

タイム・スタンプ・フォーマットを変換する:　Logging Alphanumericの前のTo Stringオブジェクトによって、タイム・スタンプをRealフォーマットからより明快なTime Stampフォーマットに変換します。

「テスト・シーケンサの使用」ステップ6

シーケンサに多くのテストが含まれる場合、多数の個別Formulaオブジェクトを Junction に接続するのは煩雑です。かわりに、1つの式を含むFormulaを使用して、実行時に式を生成し、可能な式をループします。

例では最初に式文字列を生成します。

個別スレッドで、LoopとFormulaを使用し、テストの式文字列を生成します。情報を文字列としてLogging Alphanumericに出力します。Formulaで生成される文字列は、`"a[*].test<x>.timestamp"` となるはずです。`<x>`は、1から4まで変化します。

解答 - 「テスト・シーケンサの使用」ステップ6

図315に、ステップ6の解答を示します。

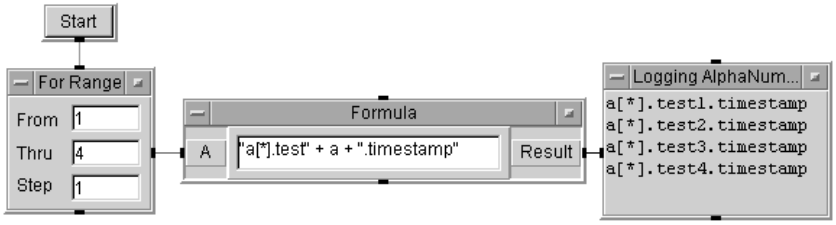


図315　　「テスト・シーケンサの使用」ステップ6

「テスト・シーケンサの使用」ステップ7

ステップ6で構築したLoopとFormulaを取り込み、前のステップの4つのFormulaとJunctionを、そのLoopとFormulaで置き換えます。さらに、構築した文字列の評価を行います。生成された文字列(式`"a[*].test<x>.timestamp"`)を実行時に評価するFormulaに送信します。

Formulaオブジェクト上のFormula Controlピン: 評価したいFormulaは、Loop内のFormulaによって生成されます。そのFormula式に対するコントロール入力を持つ2番目のFormulaボックスを作成できます。2番目のFormulaが評価する式は、実行時に生成されます。

解答 - 「テスト・シーケンサの使用」ステップ7

図316に、ステップ7の解答を示します。

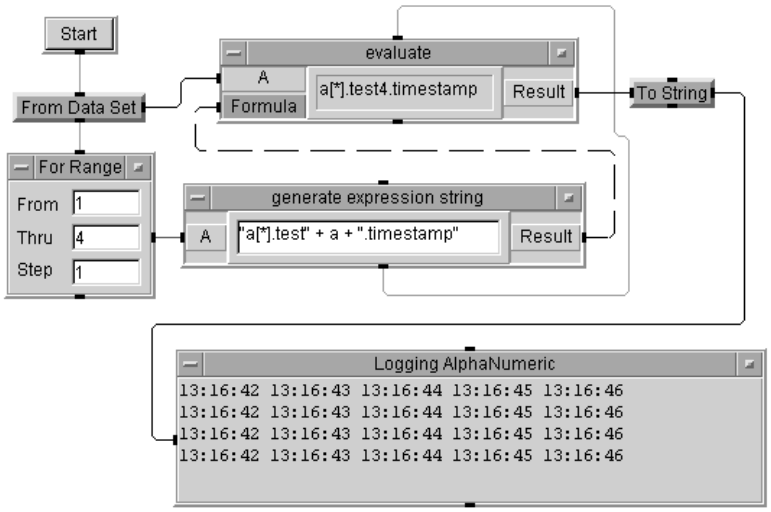


図316　「テスト・シーケンサの使用」ステップ7

キー・ポイント

- To Stringオブジェクトが、Real64をタイム・スタンプ・フォーマットにフォーマットするためそのまま使用されています。
- 最初に「生成する」Formulaと2番目に「評価する」Formulaとの間にシーケンス・ラインがあります。これにより、2番目のFormulaは、評価する新しい文字列を取得するまで動作しません。

「テスト・シーケンサの使用」ステップ8

テスト1に合格し､テスト2に不合格になったレコードだけを表示します。

解答 - 「テスト・シーケンサの使用」ステップ8

図317に、テスト・シーケンサを使用する最後のステップの解答を示します。

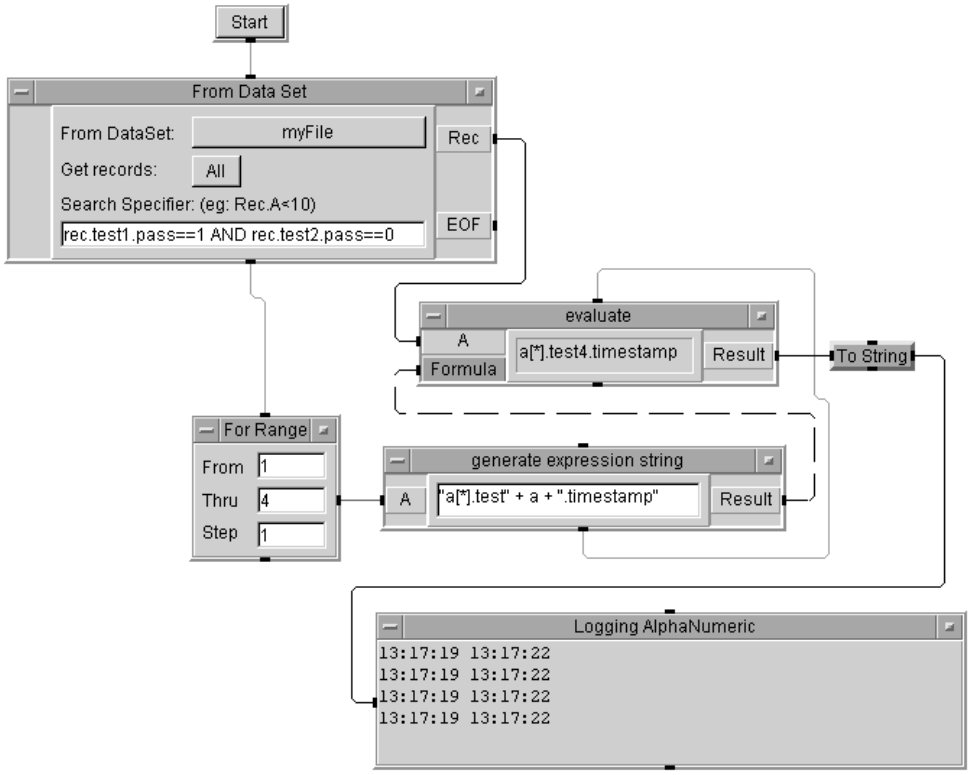


図317　　「テスト・シーケンサの使用」ステップ8

キー・ポイント

From Data Setオブジェクト上のEOFピン(2番目のスレッド):
基準に適合するレコードがないときにEOFピンが追加されます。これが発生すると、VEEがプログラムを停止してエラー・メッセージを表示するかわりに、EOFピンが起動します。

From Data Setオブジェクト内の条件式(2番目のスレッド): 式は、*(Rec.test1.pass==1)* または*(Rec.test2.pass==0)* で、同じ<**レコード**>.<**レコード**>.<**フィールド**>フォーマットを持ちます。Recは、読み取られ、テストされるデータセット内の各レコードの名前です。test1とtest2は、どのテストをVEEが調査するかを指定します。フィールド名passは、VEEで割り当てられた合否インジケータ(1または0)のデフォルト名です。[Sequencer Properties]ボックスの[Logging]タブを選択して、全部のテストの異なるフィールドを有効または無効にします。

用語集

用語集では、このマニュアルで使用している用語を定義します。VEEのさらに詳細な用語集が必要な場合は、**[Help]** ⇒ **[Contents and Index]**をクリックします。さらに、**[Reference]**、**[Glossary]** を順に選択します。[Glossary]で用語をクリックすると、その定義が表示されます。定義を読み終えたら、画面の任意の箇所をクリックするとテキストがクリアされます。

Direct I/Oオブジェクト　VEEが計測ドライバを使用しないで計測器を直接制御するための計測器制御オブジェクト。

Program Explorer　プログラム、特に画面では表示できないプログラムの一部を検索可能にするVEEウィンドウ内の機能。特に、物理的に画面上には表示されない部分の探索を可能にします。

UserObject　プログラム内で特定の目的を果たすためにオブジェクトのグループをカプセル化できるオブジェクト。UserObjectを使用すると、トップダウンの設計手法を使ってプログラムを構築できます。また、ユーザ定義オブジェクトを構築し、ライブラリ内にセーブして再使用できます。

Windows 98、Windows NT 4.0、Windows 2000、Windows XP

Microsoft社の開発によるオペレーティング・システム。VEEは、これらのオペレーティング・システムの下で実行されます。

アイコン:

1. VEEオブジェクトを小さくグラフィカルに表現したもの。計測器、制御、表示などのオブジェクトを表します。
2. Microsoft Windowsオペレーティング・システムで、アプリケーション、ファイル、またはフォルダを小さくグラフィカルに表現したもの。

オープン・ビュー　最小化ビュー (アイコン)より詳細なVEEオブジェクトの表示方法。ほとんどのオブジェクトのオープン・ビューには、そのオブジェクトの操作を変更するためのフィールドがあります。

オブジェクト:

1. 計測器、制御、表示、算術演算子などのプログラム内の要素をグラフィカルに表現したもの。オブジェクトは作業領域に配置され、ほかのオブジェクトに接続されてプログラムを形成します。
2. ActiveXオートメーションとコントロールで使用されるデータ型の1つ。

オブジェクト・メニュー　オブジェクトに関連付けられ、そのオブジェクトを操作するための機能(移動、サイズ変更、コピー、削除など)を保持するメニュー。オブジェクト・メニューを利用するには、オブジェクトの左上隅にあるオブジェクト・メニュー・ボタンをクリックするか、マウス・ポインタをオブジェクト上に置き、マウスを右クリックします。

オブジェクト・メニュー・ボタン　オープン・ビュー・オブジェクトの左上隅にあるボタン。これをクリックすると、オブジェクト・メニューが表示されます。

カーソル　キーボードから情報を入力するときに、英数字データが表示される箇所を示す入力フィールド内のポインタ(カレット)。

カスケード・メニュー　プルダウン・メニューやポップアップ・メニューで追加の選択肢を提供するサブメニュー。

カット・バッファ　切取りまたはコピーされたオブジェクトを保持するバッファ。**[Paste]**ツールバー・ボタン(**[Edit]** ⇒ **[Paste]**)を使用すると、作業領域にオブジェクトを貼り付けることができます。

クリック　マウス・ボタンを押して離すこと。通常は、クリックにより、VEEウィンドウ内でメニュー機能やオブジェクトを選択します。「**ダブルクリック**」と「**ドラッグ**」も参照してください。

グループ・ウィンドウ　Microsoft Windowsで使用され、1つのグループ内のアプリケーション・アイコンを保持するウィンドウ。それぞれのアイコンにより、グループ内の1つのアプリケーションを起動できます。

クローン　VEEのオブジェクト・メニューにあるメニュー項目の1つ。オブジェクトとオブジェクト間の相互接続を複製し、そのコピーをPasteバッファに置きます。クローンは、クローンされるオブジェクトのピン、パラメータ、サイズなどのすべての属性をコピーします。

コンテキスト　ほかのレベルの作業領域(ネストされたUserObjectなど)を保持できるが、それに依存しない作業領域のレベル。

コンテナ　「**データ・コンテナ**」を参照してください。

コンポーネント　VEEの計測器パネルやコンポーネント・ドライバにある単一の計測関数または測定値。たとえば、ボルタメータ・ドライバには、範囲、トリガ・ソース、最新の読み取り値を記録するコンポーネントがあります。

コンポーネント・ドライバ　特別に選択されたコンポーネントについて値の読み取りまたは書込みを行う計測器制御オブジェクト。一度に少数のコンポーネントの値だけを設定することにより、ドライバを使用する計測器を制御するには、コンポーネント・ドライバを使用します。コンポーネント・ドライバはカップリングをサポートしません。

最小化ボタン　オブジェクトまたはVEEウィンドウにあるボタンの1つ。そのオブジェクトまたはVEEウィンドウをアイコン化します。

最大化ボタン　UserObject、UserFunction、またはメインの各ウィンドウにあるボタンの1つ。UserObject、UserFunction、またはメインの各ウィンドウを有効なワークスペース全体に広げます。

作業領域　**メイン**(および`UserObject`と`UserFunction`)ウィンドウ内の領域。この領域内にVEEオブジェクトを配置し、それらのオブジェクトを互いに接続してVEEプログラムを作成します。

シーケンス出力ピン　オブジェクトの**下部**のピン。接続された場合、このオブジェクトとこのオブジェクトからのすべてのデータ伝達が実行を完了したときに、この出力ピンがアクティブ化されます。

シーケンス入力ピン　オブジェクトの**上部**のピン。接続された場合、このピンがコンテナを受信する(起動される)まで、オブジェクトの実行は延期されます。

式　入力フィールド内で入力端子名、グローバル変数名、演算関数、ユーザ定義関数を含むことができる計算式。式は、実行時に評価されます。式は、**Formula**、**If/Then/Else**、**Get Values**、**Get Field**、**Set Field**、**Sequencer**、**Dialog Box**の各オブジェクトとI/Oトランザクション・オブジェクトの中で使用できます。

詳細ビュー　すべてのオブジェクトと、それらを接続する線を示すVEEプログラムのビュー。

スクロール・バー　ドラッグすることにより、ダイアログ・ボックス内でデータ・ファイルなどの選択対象のリストをスクロールしたり、作業領域を移動するための長方形のバー。

スクロール矢印　クリックすることにより、ダイアログ・ボックス内でデータ・ファイルなどの選択対象のリストをスクロールしたり、作業領域を移動するための矢印。

ステータス・バー　ウィンドウの下端にある行。VEEの現在の状態やVEEに関する情報が表示されます。

ステータス・フィールド　編集できない情報が表示されるフィールド。ステータス・フィールドは、入力フィールドに似ていますが、背景が灰色です。

設定　VEE環境の属性。設定は、ツールバーの**[Default Preferences]**ボタンまたは**[File]** ⇒ **[Default Preferences]**メニューを使って変更できます。たとえば、デフォルトの色、フォント、数字のフォーマットなどを変更できます。

選択　オブジェクト、実行する操作、またはメニュー項目を選択すること。通常は、マウスをクリックして選択を行います。

選択フィールド　オブジェクトやダイアログ・ボックス内にあり、ドロップダウン・リストから項目を選択できるフィールド。

タイトル・バー　オブジェクトのオープン・ビューまたはウィンドウの上部にある長方形のバー。そのオブジェクトまたはウィンドウのタイトルが表示されます。オブジェクト・メニューの**[Properties]**を使用すると、オブジェクトのタイトル・バーを非表示にできます。

ダブルクリック　マウス・ボタンを続けて2度すばやく押して離すこと。通常、ダブルクリックは、選択とある操作の実行を一度に行うための方法です。通常、ダブルクリックは、ある操作の選択と実行を一度に行うための方法です。たとえば、**[File]** ⇒ **[Open]**の後にファイル名をダブルクリックすると、そのファイルが選択されて開かれます。

端子　ピンとピンが保持するデータ・コンテナに関する情報を表示するためのピンの内部的な表現。端子をダブルクリックすると、コンテナ情報が表示されます。

淡色表示　メニューが黒ではなく灰色で表示されること。これは、その機能がアクティブでないか、使用できないことを示します。ボタン、チェックボックス、ラジオ・ボタンなどのダイアログ・ボックス内のアイテムも、淡色表示になることがあります。

チェックボックス　VEEのメニューやダイアログ・ボックスに表示されるくぼんだ四角形のボックス。設定を選択するために使用します。設定を選択するには、ボックスをクリックします。ボックスの中にチェック・マークが表示されて、選択されたことが示されます。設定をキャンセルするには、ボックスを再びクリックします。

ツールバー　VEEウィンドウの上部にある長方形のバー。頻繁に使用するコマンドにすばやくアクセスするためのボタンが用意されています。ボタンは、**[File]**、**[Edit]**、**[View]**、**[Device]**、**[Debug]**などのメニューのコマンドを実行します。

データ・コンテナ　回線上を伝送され、オブジェクトによって処理されるデータ・パッケージ。各データ・コンテナは、データのほか、データ型、データの種類、およびマッピング(ある場合)を保持します。

データ・フロー　VEEオブジェクトを介したVEEオブジェクト間のデータの流れ。データは、オブジェクトを介して左から右へと流れますが、オブジェクトは、そのデータ入力ピンのすべてにデータが伝達されるまで実行されません。データは、1つのオブジェクトのデータ出力ピンから、次のオブジェクトのデータ入力ピンに伝達されます。データ・フローは、VEEプログラムの実行を決定する主要な要素です。

データ型　各データ・コンテナには、型と種類の両方があります。VEEは、Text、Real64、Real32、Int32などの多くのデータ型をサポートしています。

データ出力ピン　オブジェクトの右側にある接続点。データ・フローを次のオブジェクトに伝達したり、このオブジェクトの操作結果を次のオブジェクトに渡します。

データ入力ピン　オブジェクトの左側にある接続点。ここからオブジェクトにデータが流れ込みます。

データの種類　各データ・コンテナには、種類と型の両方があります。データの種類は、スカラまたは1次元以上の配列のどちらかになります。VEEでは、1次元の配列を「Array 1D」、2次元の配列を「Array 2D」などと呼びます。

伝達　オブジェクトやプログラムが操作または実行される場合に従う規則。「**データ・フロー**」も参照してください。

ドラッグ　マウス・ボタンを押し、**それを押したまま**マウスを移動すること。ドラッグにより、オブジェクトやスクロール・バーなどを移動します。

トランザクション　の一定のオブジェクトによって使用される入出力(I/O)の仕様。Examples include the `To File`, `From File`, `Sequencer`, and `Direct I/O` objects.トランザクションは、これらのオブジェクトのオープン・ビューの中に句としてリストされます。

ドロップダウン・リスト　選択フィールドの右側にある矢印をクリックすることによって表示される選択肢の一覧。

入力フィールド　データ入力に使用されるフィールド。ダイアログ・ボックスや編集可能なオブジェクトの一部であることが普通です。入力フィールドは、背景色が白の場合に編集できます。

ハイパーテキスト　関連するトピックにジャンプして、さらに情報を得ることができるようにトピックどうしをリンクするシステム。オンライン・ヘルプ・システムでは、ハイパーテキスト・リンクは下線付きのテキストとして示されるのが普通です。そのテキストをクリックすると、関連する情報が表示されます。

パネル・ドライバ　対応する物理計測器のすべての機能設定をオブジェクトのオープン・ビューに表示されている制御パネルの設定と一致させる計測器制御オブジェクト。

パネル・ビュー　ユーザがプログラムを実行し、その結果データを表示するために必要なオブジェクトだけが表示されるVEE プログラム(UserObjectまたはUserFunction)のビュー。パネル・ビューを使用すると、プログラムのオペレータ・インタフェースを作成できます。パネル・ビューを使用すると、プログラムのオペレータ・インタフェースを作成できます。

ビュー　プログラムを表示するための方法。VEE には、2つのビューがあります。パネル・ビューは、VEEプログラムのユーザ・インタフェースです。詳細ビューは、VEEプログラムを開発するためのウィンドウです。

ピン　線を接続できるオブジェクトの外部接続点。

フォント　VEEでは、さまざまなVEEオブジェクトやタイトルなどのテキストの表示に使用されるフォントのサイズとスタイルを変更できます。

プルダウン・メニュー　メニュー・タイトルの上にポインタを置き、マウスの左ボタンをクリックしたときに、メニュー・バーの下に表示されるメニュー。

プログラム　VEEでは、線で接続されたオブジェクトの集合で構成されるグラフィカルなプログラム。一般に、プログラムは、エンジニアリング上の問題に対する解決策を表現します。

プロパティ　色、フォント、タイトルなどのVEEオブジェクトの属性。オブジェクトのプロパティは、オブジェクト・メニューの**[Properties]**を使って変更できます。

ポインタ　マウスの移動位置をグラフィカルに示すもの。ユーザは、ポインタを使って選択したり、ポインタから特定の操作の状態を知ることができます。VEEでは、操作の状態に対応して、矢印、十字、砂時計などのさまざまな形状のポインタが表示されます。

ボタン　押しボタンや選択ボタンに似せたグラフィカル・オブジェクト。VEEの画面に飛び出たように表示されます。VEE内のボタンをマウスでクリックして「押す」と、その操作が行われます。マウスの左または右のボタンを指す場合もあります。

ポップアップ・メニュー　マウスの右ボタンをクリックして画面に表示されるメニュー。たとえば、作業領域内の何もない領域でマウスの右ボタンをクリックすると、**[Edit]**メニューが表示されます。また、オブジェクトのアクティブでない領域でマウスの右ボタンをクリックすると、オブジェクト・メニューが表示されます。

メイン・ウィンドウ　VEEプログラムを開発する基本の作業領域を含むウィンドウです。このウィンドウの作業領域は、VEEウィンドウのワークスペース内にあります。

メニュー・バー　VEEウィンドウの上部にあるバー。プルダウン・メニューのタイトルが表示され、そこからコマンドやオブジェクトを選択できます。

ワークスペース　VEEウィンドウ内にあり、**メイン**、UserObject、UserFunctionなどのプログラミングまたは編集ウィンドウを保持する領域。これらのウィンドウには、VEEオブジェクトを配置し、それらを互いに接続するための作業領域があります。

記号

数字

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

あ

か

さ

た

な

は

ま

や

ら

わ