VEE Pro ユーザーズ・ガイド

注意

本書の内容は、予告なく変更されることがあります。

Agilent Technologies は、本製品の商品性または特定の目的への適合性に対する暗黙の保証、およびそれ以外の一切の保証をいたしません。また、本製品のエラーまたは本製品の設置、性能、使用に関して直接的、間接的、偶発的、必然的、あるいは特別に発生した損害に対して賠償の責任を負いません。

商標情報

Microsoft®、MS-DOS®、Windows®、MS Windows®、Windows NT® は、Microsoft 社の米国における登録商標です。

MATLAB® は、MathWorks, Inc. の登録商標です。

Netscape は、Netscape Communications Corporation の合衆国における商標です。

UNIX® は、Open Group の登録商標です。

改版履歴

Edition 1 - March 2000　　ソフトウェアのバージョン 6.0 を反映

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。

Getting Started	斜体文字は、書名または強調語を表します。
Dialog Box	用語集で定義されている語句がマニュアル中に初めて出現したときは、太字で表記します。
`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、入力するテキストなど、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

3. 簡単な計測器の操作

4. テスト・データの分析と表示

5. テスト結果の保管方法と読取り方法

6. ActiveX を使ったレポートの簡単な作成方法

7. 異なる言語で書かれた複数のプログラムの統合方法

8. Agilent VEE 関数の使用方法

9. テストのシーケンスを決定する方法

10. オペレータ・インタフェースの使用方法

11. Agilent VEE プログラムの最適化

図

表

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構築を目的として最適化されたグラフィカルなプログラミング言語です。Agilent VEE 製品グループのこのバージョンには、複雑なテスト / 計測システムを構築する技術者グループ向けの VEE Pro 6.0 と、設計やデータ収集を行う技術者や科学者個人向けの VEE OneLab 6.0 があります。

テスト開発に Agilent VEE を使用する利点

VEE は、テスト開発において、次の多くの利点を提供します。

- 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、評価、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、ODAS ドライバ、標準的なインタフェースのための「direct I/O」、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、データを表示して分析したり、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。
- 処理量が増えたため、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。
- 作成したプログラムを C/C++、Visual Basic、Pascal、Fortran、Rocky Mountain Basic などのテキスト形式言語に統合できます。

Agilent VEE におけるプログラムの作成方法

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、従来のコード列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 個の乱数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 1 は、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 2 は、VEE で作成した同じプログラムを示します。

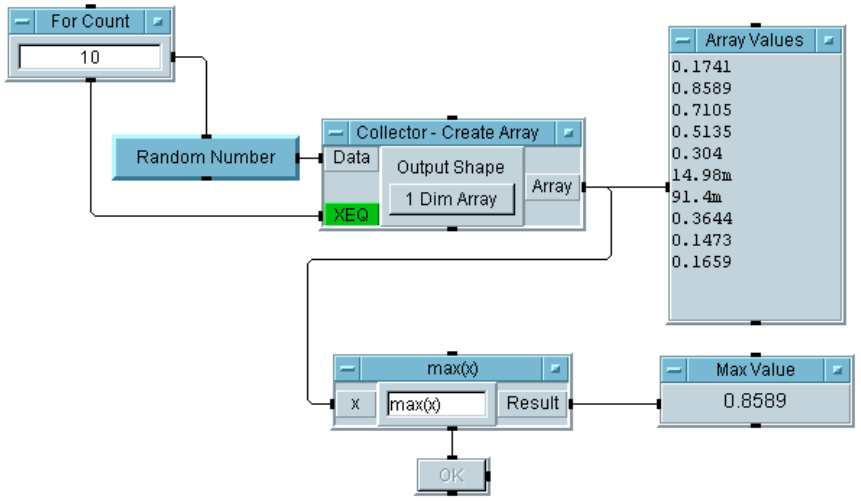


図 I-2. VEE による同じ「Random」プログラム

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 VEE プログラムの構成単位であり、I/O 操作、分析、表示などさまざまな機能を実行します。オブジェクトとその接続をすべて図 2 のように表示したものを**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維持しながら移動します。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動作を操作するシーケンス・ピンがあります。

オブジェクトは、結合して 1 つのプログラムを形成します。プログラムは左から右へ実行されます。図 2 で示した「Random」プログラムでは、乱数が「`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`」と表示します。

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大きく高めます。

メモ

図 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、つまりオブジェクトを縮小して、名前だけが表示されるようにします。

たとえば図 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。オブジェクトの表示については、第 1 章「Agilent VEE 開発環境の使用方法」の 32 ページの「オブジェクトの表示の変更」で詳述します。

Agilent VEE におけるオペレータ・インタフェースの作成

また、VEE でプログラムを記述すれば、オペレータ・インタフェースを簡単に作成できます。

図 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 3 は、図 2 の「Random」プログラムのパネル・ビューを示しています。

メモ

プログラムとオペレータ・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。また、オペレータ・インタフェースが不用意に変更されないように設定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、93 ページの「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。

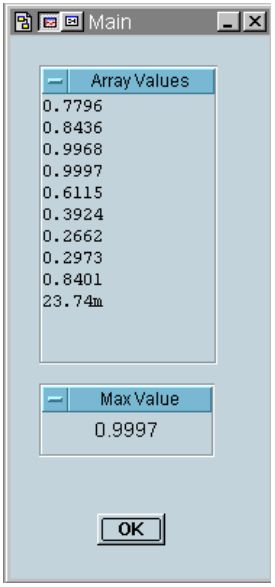


図 I-3. VEE プログラムのパネル・ビュー
（またはオペレータ・インタフェース）

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が数分で完成する場合があります。

- カラフルで直観的なフロント・エンドを作成できる。
- 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る。
- 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る。
- ポップアップ・パネルを使ってフォーカスを作成でき、画面スペースを節約できる。
- プログラムを不用意に変更できないように設定できる。
- 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、notepads、ボタン、スイッチを追加できる。

- 独自または標準の既存 ActiveX コントロールを使用して、データ入力または表示を行うことができる (PC のみ)。

既存のテスト・プログラムと Agilent VEE との統合

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・プログラムも結合する仕組みを提供します。たとえば、VEE を使用して、Rocky Mountain Basic、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ラとインタープリタで書かれた既存のテストを順に実行できます。VEE はまた、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用するために、数多くのプロセス間通信機能を提供します。

PC 上の VEE は、ActiveX オートメーション、ActiveX コントロール、DLL との標準的な接続をサポートします。HP-UX 上の VEE は、名前付きパイプと共有ライブラリをサポートします。.

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) を使用できる。また、Windows 95、Windows 98、Windows 2000、Windows NT 4.0、HP-UX フレームワークと互換性があるさまざまなベンダの VXI *plug & play* ドライバをすべて使用できる。

- 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的なインタフェースに送信して、リモート・テストを行うことができる。

- 標準的な ODAS ドライバ、あるいはボードに付属する DLL(Dynamic Link Library) を供給する製造元の PC プラグイン・ボードを制御できる。

- 埋込み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る。

- 簡単で組織化された計測器管理機能により、多くの種類の計測器を制御できる。

Agilent VEE によるテストの強化

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、RMB 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 の数多くのサポート・オプションを利用可能
- 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分析パッケージ (VEE Pro 6.0 のみ) 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
- 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 (VEE Pro 6.0 のみ)
- 製品に付属する強力なテスト実行ツール (VEE Pro 6.0 のみ)
- VEE の Web モニタ機能によるリモート・テスト機能 (VEE Pro 6.0 のみ)
- RunTime を無制限に配布可能 (VEE Pro 6.0 のみ)
- 低価格のサイト・ライセンス (VEE Pro 6.0 のみ)

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方法

VEE Pro 6.0 および I/O ライブラリのインストール方法についての情報は、VEE に添付されているインストール・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 6.0 RunTime*」では、VEE Pro 6.0 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 環境についての詳細は、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、「Agilent VEEInstalling 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] メニューにあり、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 6.0」を選択して学習できます。[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、25 ページの「ヘルプの利用」と 101 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、このマニュアルのほかに、『*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。

メモ

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、米国 (800) 829-4444 に電話するか、または以下の Agilent Technologies Web サイトにご連絡ください。

`http://www.agilent.com/tmo`

MATLAB Script 概要

MATLAB® Script は、MathWorks 社製の高機能 MATLAB のサブセットです。これにより、ユーザは高度な数式処理、データ分析、科学的 / 工学的グラフィックなど MATLAB の中核機能に直接アクセスできます。MATLAB Script オブジェクトは、どのような 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。

- データの分析と視覚化
- 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分析
  - 微分方程式の計算
  - 三角関数および基本関数の数式演算
- 次の科学的、工学的グラフィック
  - 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分析技術に基づいて構築された MATLAB Signal Processing Toolbox のサブセットが含まれます。これには、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 および 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、高機能プログラミング言語を統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。

- データの分析と視覚化
- 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、財務管理、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、分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、データ分析、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)、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 MATLAB 機能を VEE アプリケーションで幅広く使用できます。

メモ

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の詳細は、第 4 章「テスト・データの分析と表示」の 189 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、アプリケーション情報、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`
- *最新サポート情報*: ネットワークに接続した状態で、VEE で [Help] ⇒ [Agilent VEE on the Web] ⇒ [Support] をクリックします。図 4 は、VEE での [Support] メニューの選択方法を示します。また Web ブラウザでは、`http://www.agilent.com/find/vee` に接続して、[Support] をクリックします。

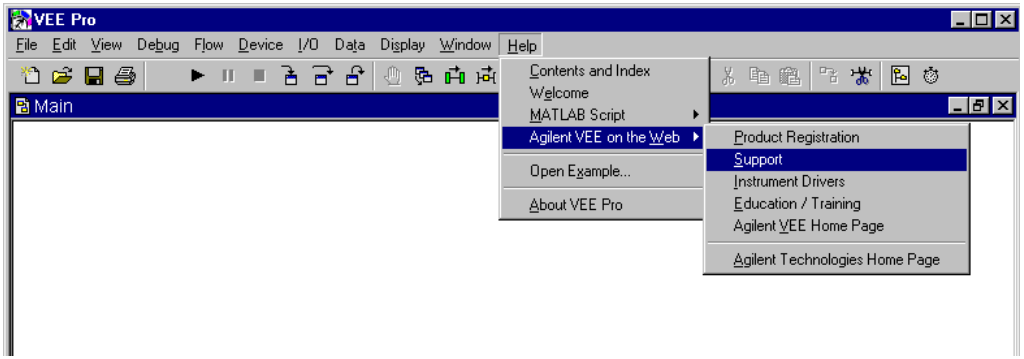


図 I-4. VEE の [Help] メニューから製品サポートを利用

- *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。VEE では、[Help] ⇒ [Contents and Index] をクリックして、「Agilent VEE Support」を選択します。

MATLAB の追加情報源

MATLAB Script オブジェクトの使用についての完全で詳細な情報は、「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 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 ドキュメント全文 : www.mathworks.com/access/helpdesk/help/helpdesk.shtml
- MATLAB アップグレード申込み :VEE Pro 6.0 および VEE OneLab 6.0 のユーザは、特別価格でアップグレードをお申込みになれます。詳細は、www.mathworks.com/veeupgrade を参照してください。
- 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

平均的な必要時間 :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

VEE バージョン 6.0 は、次のシステムに対応しています。

- PC 上の Windows 95、Windows 98、Windows 2000、および Windows NT 4.0
- HP-UX ワークステーション (シリーズ 700 のバージョン 10.20)。VEE の本バージョンは、HP-UX のバージョン 11.x または 10.2 以前のバージョンでは動作しません。

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共通事項です。

- 項目を「クリック」するには、マウス・ポインタを目的の項目に合わせて、マウスの*左*ボタンをすばやく押して離します。
- 項目を「ダブルクリック」するには、マウス・ポインタを目的の項目に合わせて、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。
- 項目を「ドラッグ」するには、マウス・ポインタを目的の項目に合わせて、*マウスの左ボタンを押したまま*、項目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。

メモ

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

Windows	[スタート] ⇒ [プログラム] ⇒ [VEE Pro 6.0] をクリックします。
HP-UX	HP VUE または X11 ウィンドウのシェル・プロンプトで、「`veetest`」と入力し、Return キーを押します。`PATH` 変数に `/usr/bin` が含まれている必要があります。

Agilent VEE ウィンドウ

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

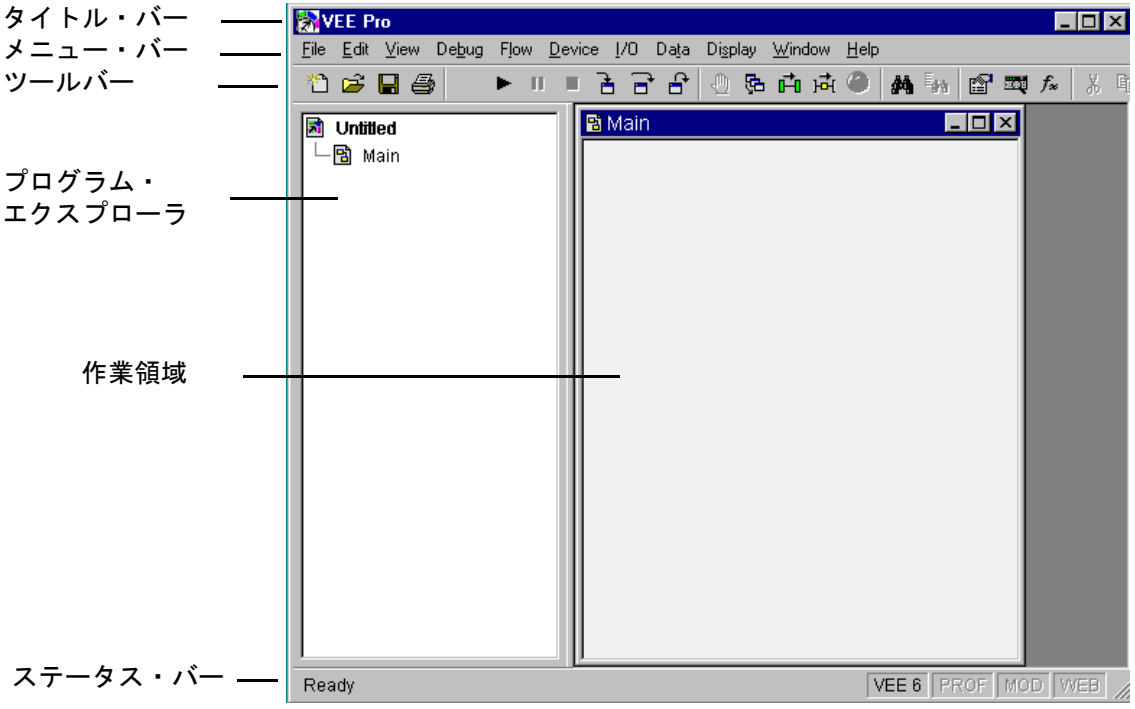


図 1-1. VEE 開発環境

次に、VEE ウィンドウの各部について説明します。

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、右側の作業領域との境界線上に通常のマウス・ポインタを移動し、ポインタが縦の分割ポインタに変化したら、クリックして境界線を移動します。

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。次のインジケータが左から右へ順番に表示されます。
実行モード
Profiler の ON/OFF 状態
プログラムが変更されると MOD と表示されます。
Web サーバが使用可能かどうか

メモ

このマニュアルは、VEE バージョン 6.0 の説明です。VEE の旧バージョンをお持ちの場合は、低価格でアップグレードできます。バージョンを確認するには、[Help] ⇒ [About VEE Pro] をクリックします。ソフトウェアのアップデートについてサポート契約を結んだ方には、自動的に新しいバージョンが配布されます。

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関する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

最初は [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 1-2 に示します。

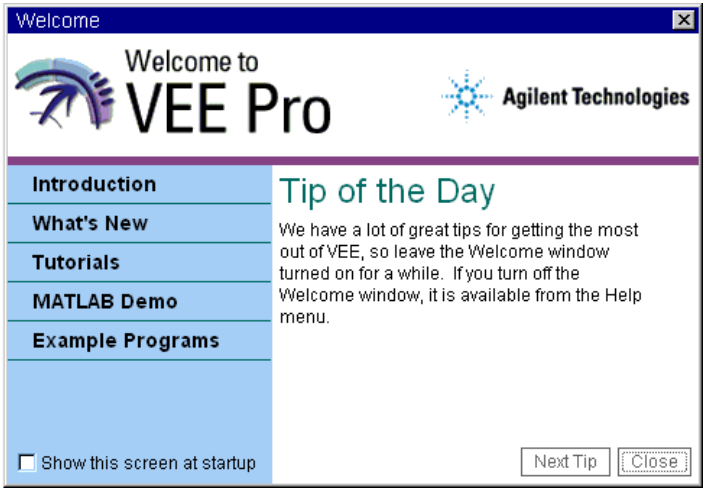


図 1-2. ヘルプにおける VEE の [Welcome] 画面

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 1-3 のようなメニューが表示されます。[Help] メニューでは、[Contents and Index] と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」することもできます。

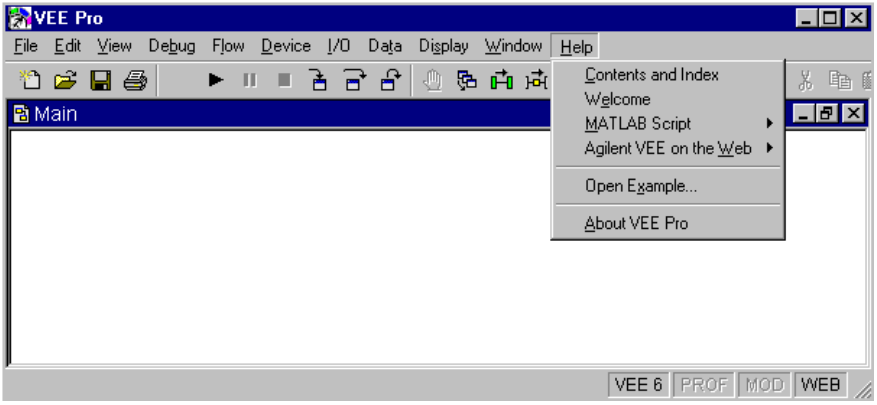


図 1-3. [Help] メニューの使用

`[Contents and Index]` を選択すると、図 1-4 のような VEE ヘルプが表示されます。

ただし、HP-UX の画面は少し異なります。

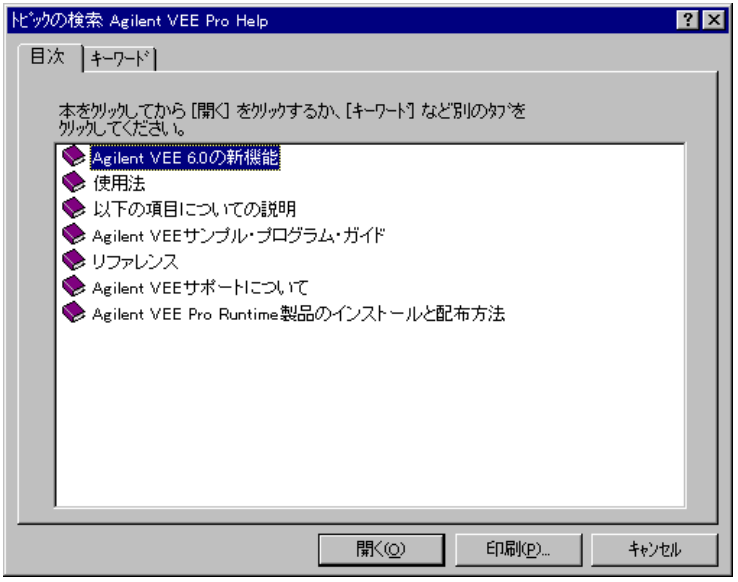


図 1-4. VEE ヘルプの [目次] タブ

ヘルプの [目次] タブには、次のトピックがあります。

「Agilent VEE 6.0 の新機能」	新機能について説明します。
「使用法」	基本的な操作方法を説明します。
「以下の項目についての説明」	VEE の概念を説明します。
「Agilent VEE サンプル・プログラム・ガイド」	VEE に搭載されているサンプル・プログラムの要約です。
「リファレンス」	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。

「Agilent VEE サポートについて」	VEE サポートの利用方法についての情報を提供します。
「Agilent VEE Pro RunTime 製品のインストールと配布方法」	VEE Pro RunTime 環境の配布方法を説明します。

メモ

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の詳細は、101 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 などのオブジェクトを VEE メニューから選択します。そして、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、48 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 を起動する。Windows では、[スタート] ⇒ [プログラム] ⇒ [VEE Pro 6.0] をクリックします。HP-UX では、HP VUE または X11 ウィンドウのシェル・プロンプトで「veetest」と入力し、Return キーを押します。PATH 変数に /usr/bin が含まれている必要があります。

2. このセクションの説明に従ってオブジェクトの扱い方を学びます。

メモ

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」セクション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。輪郭線が消え、オブジェクトが表示されます。

1. 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 1-5 のように、メニュー・バーから [Device] ⇒ [Virtual Source] ⇒ [Function Generator] を選択します。

メモ

[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。

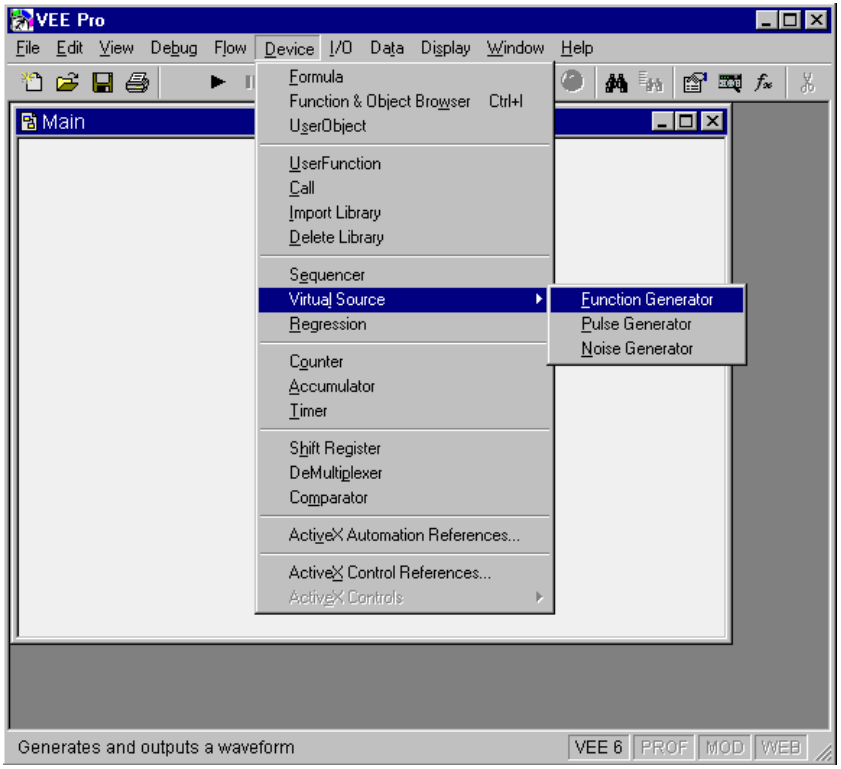


図 1-5. 作業領域へのオブジェクトの追加

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ます。クリックしてオブジェクトを配置すると、図 1-6 のように Function Generator が表示されます。

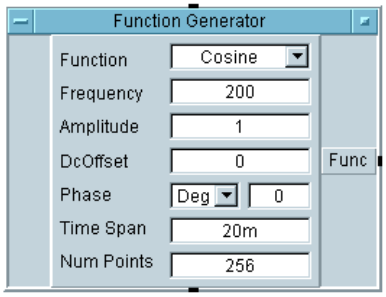


図 1-6. Function Generator オブジェクトの追加

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。

メモ

これ以降は、説明を簡潔な表現で行います。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]`

メモ

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 1-7 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。

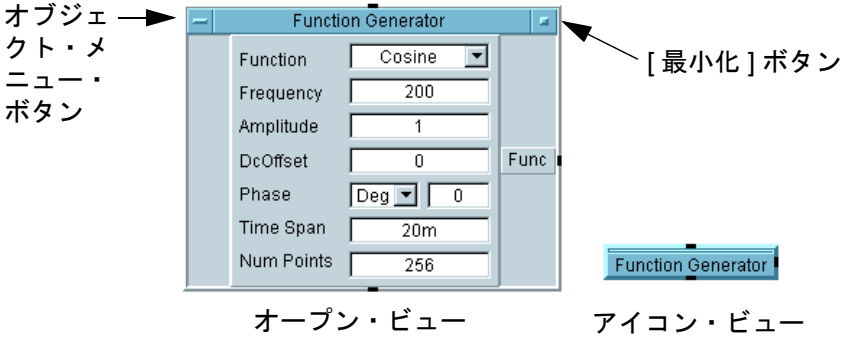


図 1-7. オープン・ビューおよびアイコン・ビューのオブジェクト

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。

メモ

オブジェクト・メニューで、[Minimize] または [Restore] を選択することもできます。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。

すべてのオブジェクトが同じ構造や部品を持っているわけではありませんが、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]、[Cut]、[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メニューを持っていますが、オブジェクトの機能によってメニューに違いがあります。

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 1-8 の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。

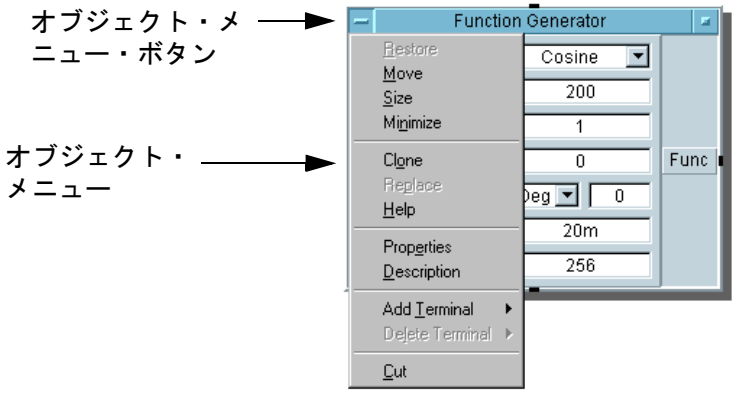


図 1-8. オブジェクト・メニューの選択

ショートカット : 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 1-9 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。

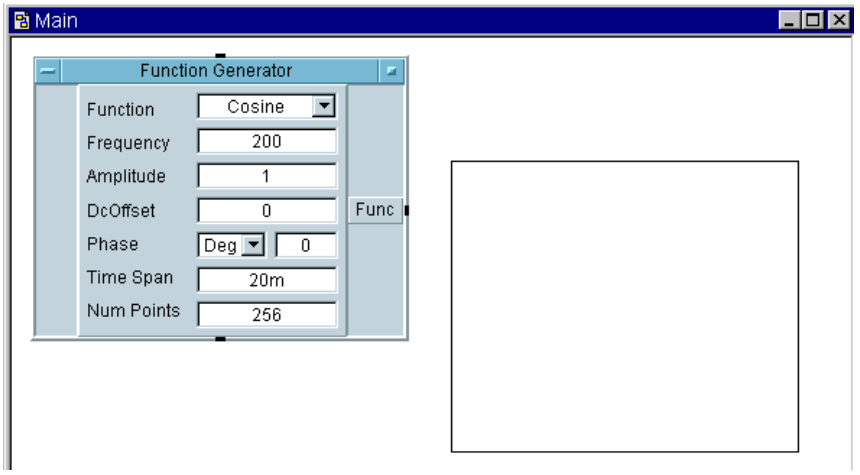


図 1-9. オブジェクトの移動

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。オブジェクトの四隅は、オブジェクトのサイズを変更するときに使用します。

メモ

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 1-10 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。

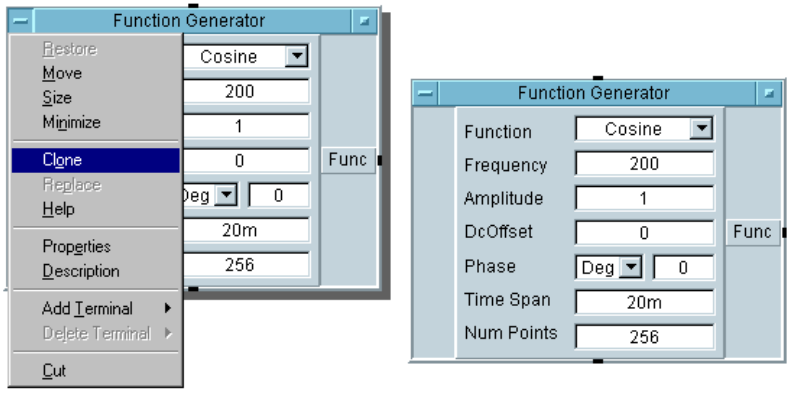


図 1-10. オブジェクトのクローン作成

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、VEE、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。

1. 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。

1. オブジェクト・メニューを開き、[Cut] 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 キーをクリックします。

   - 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。

メモ

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 [Paste] を使用します。または [Edit] ⇒ [Paste] 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 (切取りの復元)

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

1. 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。

メモ

オブジェクトにラインが接続されている場合、接続は維持されます。この操作は、ほかのプログラムの「元に戻す (Undo)」と同じ機能を持っています。この操作を「元に戻す」と呼ばないのは、VEE のすべてのプログラム操作に適用できるとは限らないためです。たとえば、「元に戻す」操作は、削除したオブジェクト群に対しても実行できます。

オブジェクトのサイズの変更

1. 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 1-11 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。

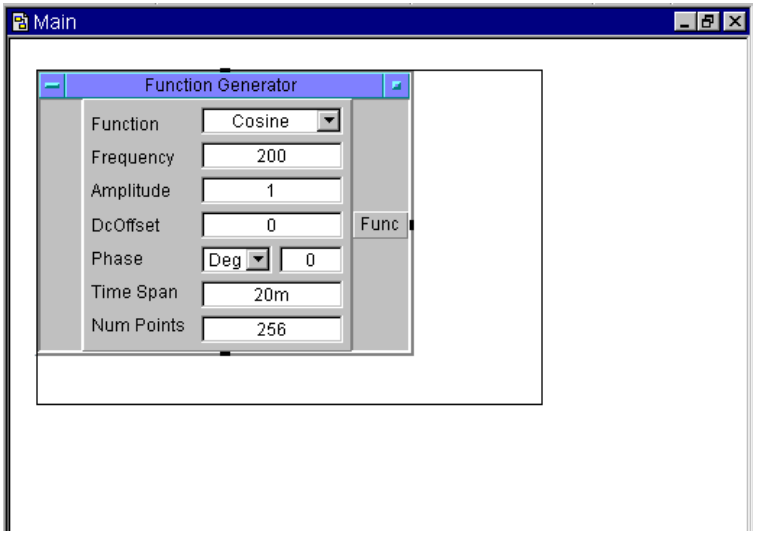


図 1-11. オブジェクトのサイズの変更

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 1-12 のように現在のタイトルが強調表示されます。

2. 新しいタイトルを入力して [OK] をクリックします。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 -

1. 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。

2. 新しいタイトルを入力して [OK] をクリックします。

メモ

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。

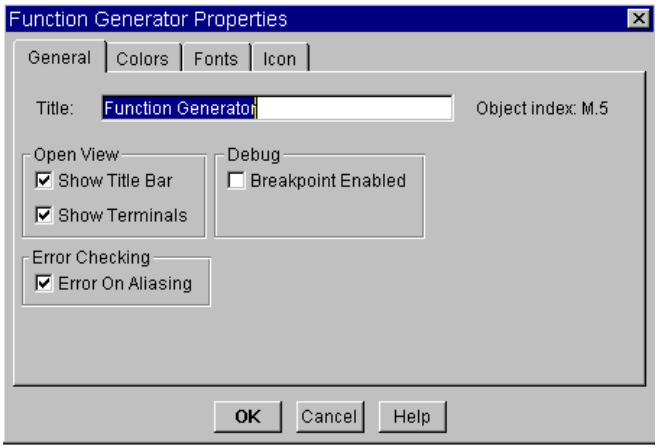


図 1-12. オブジェクトのタイトルの変更

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 1-13 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 1-13 では Formula オブジェクトは選択されていません。

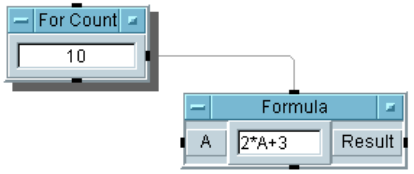


図 1-13. 選択されたオブジェクトと選択されていないオブジェクト

メモ

「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。

1. 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。すべて選択したら、Ctrl キーを離します。

   - または -

   Ctrl キーを押します。次に、選択するオブジェクトを四角形で囲むようにクリック・アンド・ドラッグします。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

1. 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が新しく作成されます。

   - 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 1-14 はコピー中のオブジェクトを示しています。

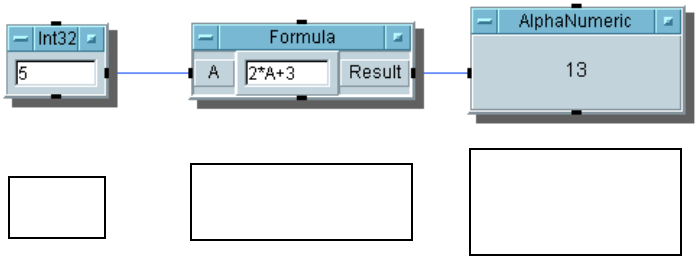


図 1-14. コピー中の複数オブジェクト

メモ

Windows 版 VEE で切取ったオブジェクトとコピーしたオブジェクトは、クリップボード上にも格納されます。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。

1. 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な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。

メモ

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 1-15 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、VEE 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。

メモ

ピンについての詳細は、48 ページの「ピンと端子について」を参照してください。

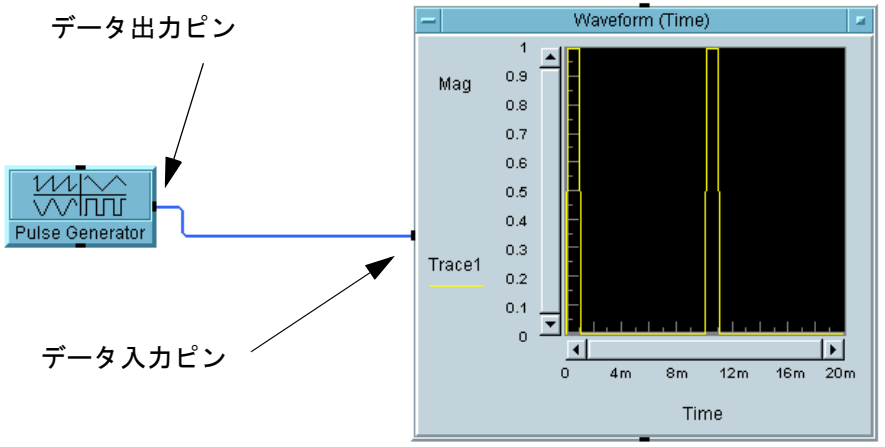


図 1-15. オブジェクト間のデータ・ラインの作成

オブジェクト間のデータ・ラインの削除

1. 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

1. 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ます。次に、作業領域を目的の方向へ移動します。

メモ

プログラムが作業領域より大きい場合は、図 1-16 のようにスクロール・バーが表示されます。

メモ

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、ポインタを作業領域の外側に移動してダブルクリックします。

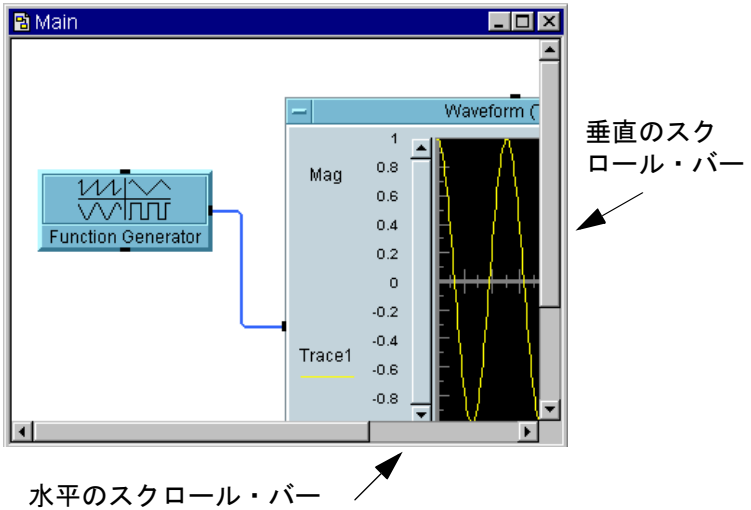


図 1-16. 作業領域のスクロール・バー

作業領域のクリア

1. `[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・ボックスでは、VEE 環境のデフォルト設定を変更します。

1. 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 1-17 のように、[Default Preferences] ダイアログ・ボックスが表示されます。

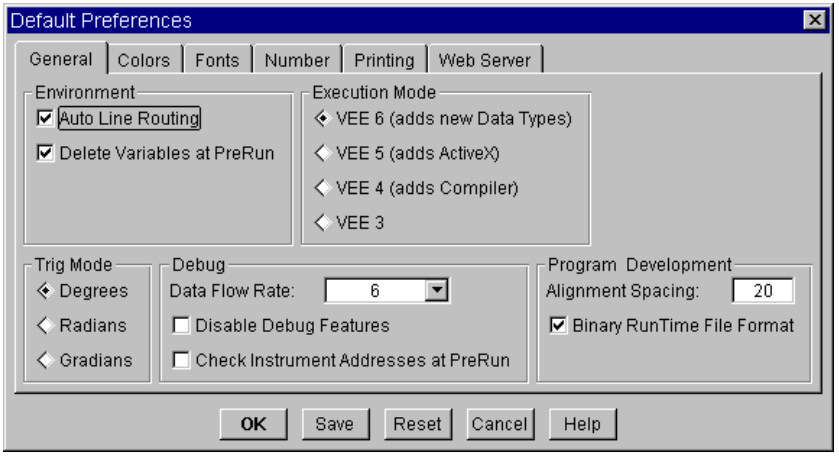


図 1-17. [Default Preferences] ダイアログ・ボックス

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。

[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]	印刷に関するパラメータ値を設定します。
[Web Server]	組込まれている Web サーバ機能を使ってリモートの Web サーバからプログラムをモニタし、トラブルシューティングを行います。

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。目次から「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。VEE 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*で接続されます。各オブジェクトには図 1-18 のように数個のピンがあります。図 1-18 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。

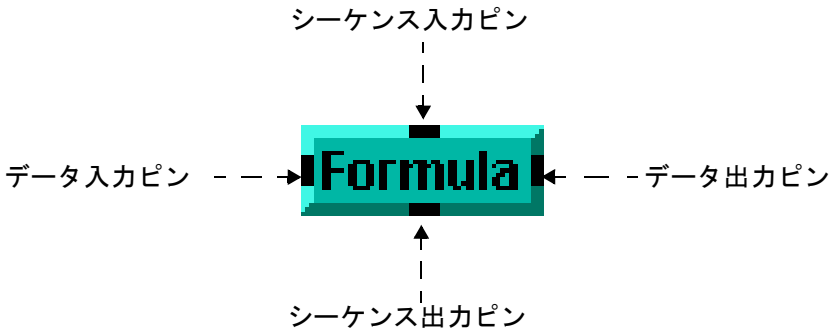


図 1-18. データ・ピンとシーケンス・ピン

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、ピンは上から下の順に実行されます。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。

メモ

詳細は、113 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 1-19 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 1-19. オブジェクトの [Show Terminals] オプション

`[Show Terminals]` オプションのオン / オフを切替えるには、オブジェクト・メニューから `[Properties]` を選択します。[Properties] ダイアログ・ボックスでは、`[Show Terminals]` の前にチェックボックスが表示されます (図 1-20 参照)。

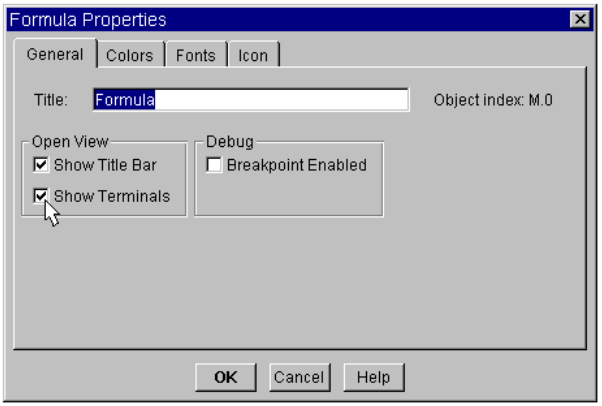


図 1-20. [Show Terminals] チェックボックスの使用

チェックボックスをクリックすると、[Show Terminals] がオフになります。チェックボックスを再度クリックすると、[Show Terminals] がオンに戻ります。選択したら、[OK] をクリックします。

端子の追加

オブジェクトには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。

1. 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 1-21 では、一方の Formula のオブジェクトではデータ入力端子を追加するためにオブジェクト・メニューが開かれており、もう一方の Formula オブジェクトでは 2 番目の端子がすでに追加されています。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。

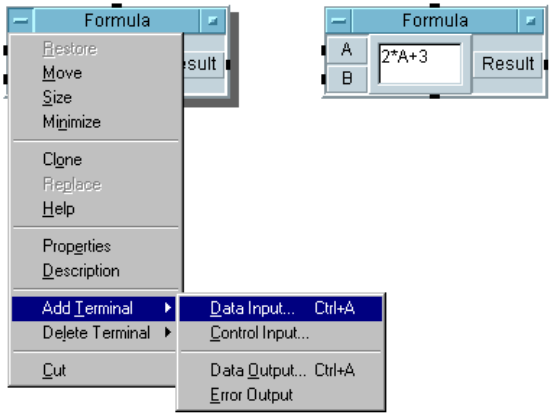


図 1-21. 端子の追加

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 1-22 のダイアログ・ボックスが表示されます。

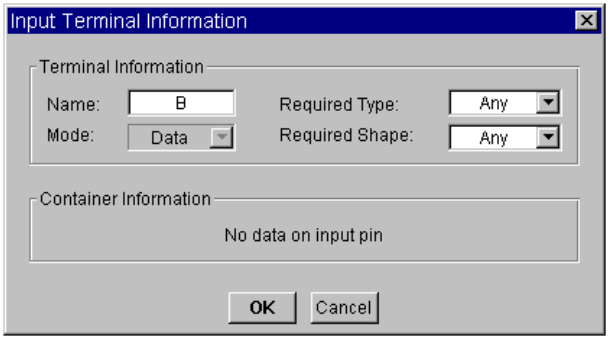


図 1-22. 端子情報の表示

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。

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は、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 1-23 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。

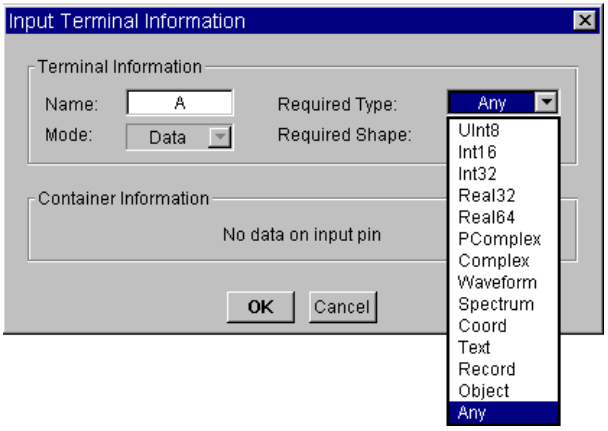


図 1-23. 選択フィールドの使用

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

1. 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 1-24 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。

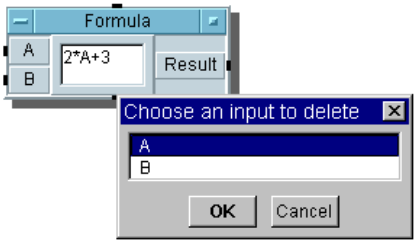


図 1-24. [Delete Terminal] ダイアログ・ボックス

オブジェクトの接続とプログラムの作成

このセクションでは VEE プログラムの作成方法を紹介します。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。

   ```
   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まで移動してクリックし、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 1-25 のように作業領域の右側にオブジェクトを配置します。

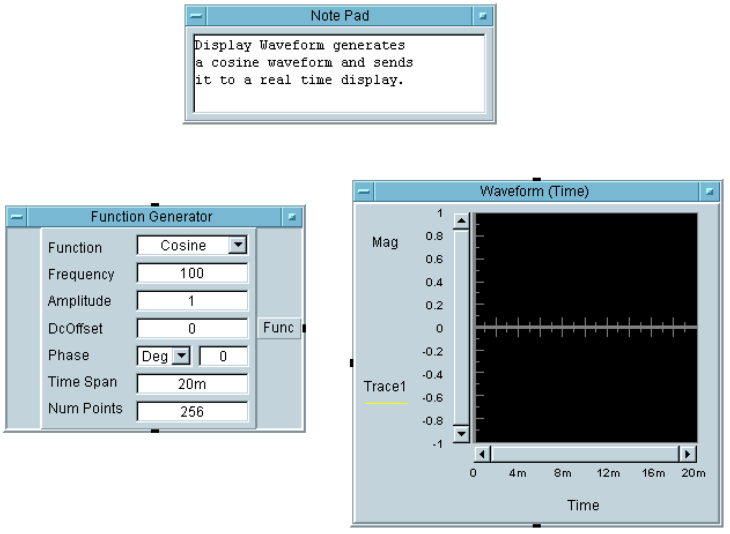


図 1-25. プログラムの作成

図 1-25 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。VEE 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 1-26 のように、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。

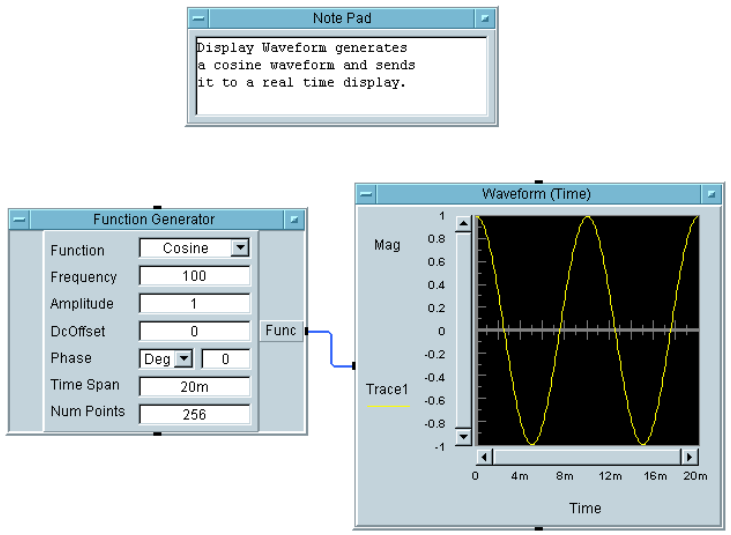


図 1-26. プログラムの実行

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。

[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 1-27 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。

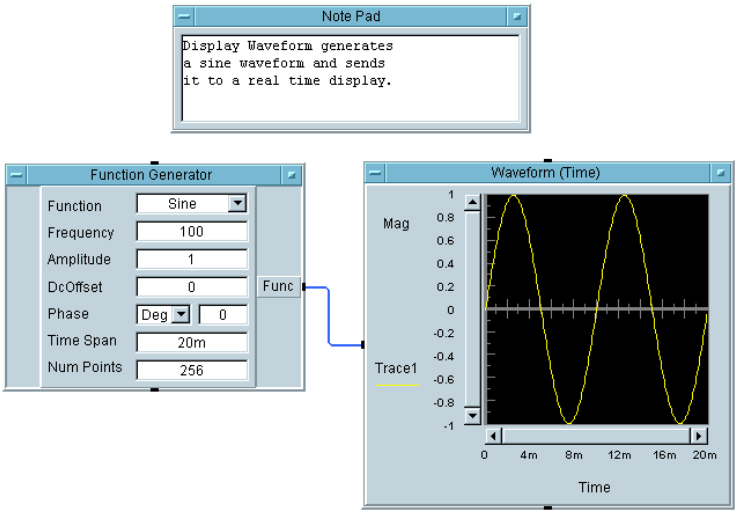


図 1-27. [Function] フィールドを正弦波に変更

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 1-28 のように最後の 0 が強調表示されるようにします。

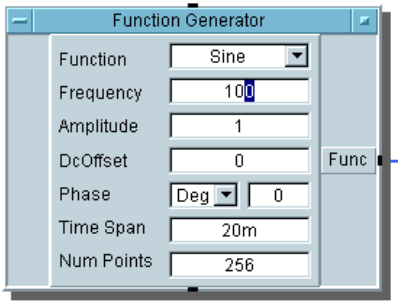


図 1-28. [Frequency] フィールドの数字を強調表示する

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 1-29 のように表示されます。

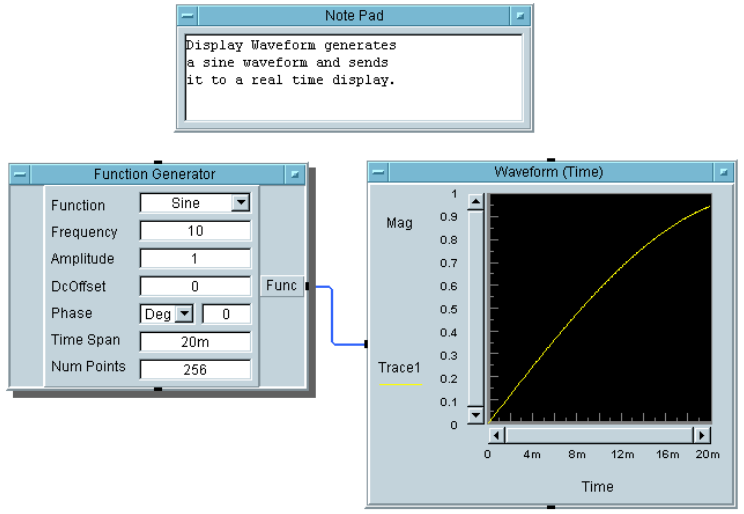


図 1-29. [Frequency] フィールドを [10Hz] に変更した例

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、Phase 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 1-30 のダイアログ・ボックスが表示されます。

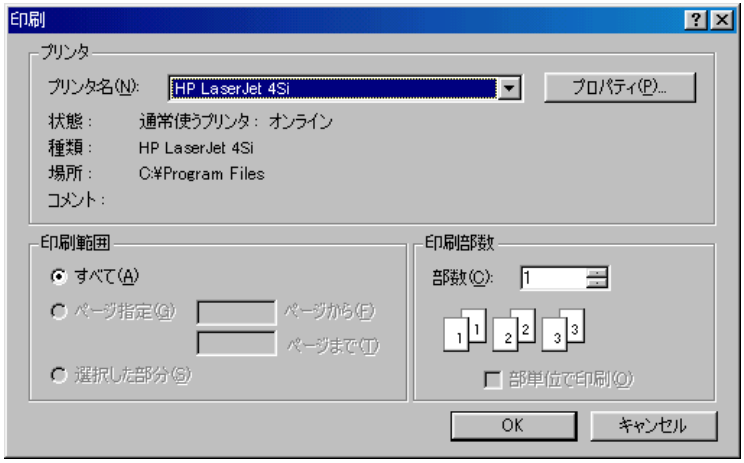


図 1-30. 画面の印刷

[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。

HP-UX では、図 1-31 のダイアログ・ボックスが表示されます。

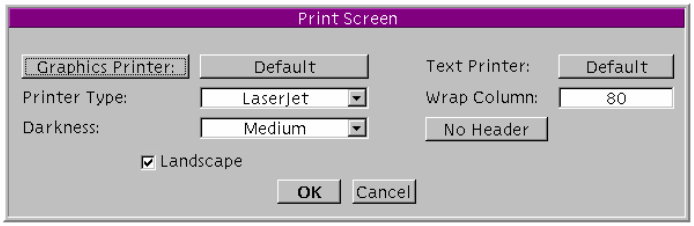


図 1-31. [Print Screen] ダイアログ・ボックス

[OK] をクリックすると、VEE は選択したプリンタで画面を印刷します。このダイアログ・ボックスでは、グラフィック・プリンタまたはテキスト・プリンタを選択します。また、印刷の前にデバイスの設定を変更します。

ショートカット : ツールバーの [Print Screen] ボタンをクリックして、画面を直接印刷できます。

プログラムの保存

プログラムは、作業領域内の内容や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 1-32 に PC で表示されるダイアログ・ボックスの形式を示します。

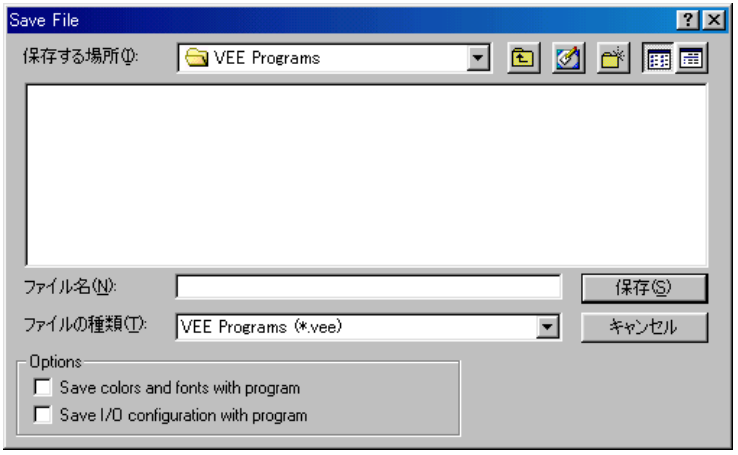


図 1-32. [Save File] ダイアログ・ボックス (PC)

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、`[File name]` フィールドに「`simple-program`」と名前を入力して `[Save]` 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。

メモ

Windows 版 VEE では、Windows 95、Windows 98、Windows 2000、および Windows NT 4.0 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。

[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。

[Save I/O configuration with program] (オプション) [Instrument Manager] で計測器を設定し、次にプログラムをロードしたときに、デフォルトではなく変更した値を使用するには、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。

メモ

評価キット・ソフトウェアを使用している場合、プログラムは EVAL.VEE というファイルにだけ保存できます。ほかの例題で練習する場合は、このファイルを上書きしてください。

HP-UX 版の [Save File] ダイアログ・ボックスについては、図 1-33 を参照してください。

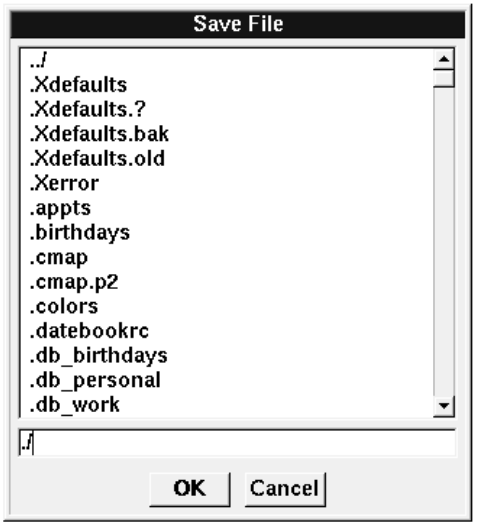


図 1-33. [Save File] ダイアログ・ボックス (UNIX)

特に指定しないかぎり、HP-UX 版 VEE は、VEE を起動したディレクトリ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 (たとえば、`simple-program.vee`) を入力して [OK] をクリックします。

ファイルを別のディレクトリに保存するには、Backspace キーを使用して、「`./`」を削除します。次に、完全なパスを含めてファイル名を入力し、[OK] をクリックします。HP-UX では、拡張子「`.vee`」を追加する必要があります。

ヒント: ダイアログ・ボックスの入力内容を簡単に置換えるには、マウス・ポインタで入力内容をクリック・アンド・ドラッグして強調表示します。または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します。次に訂正内容を入力して [OK] をクリックします。

メモ

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

12. [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。

ショートカット : 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。

Windows 95 および 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 および Windows 2000 の場合	Ctrl+Alt+Delete キーを押し [Task Manager] 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。
HP-UX の場合	プロセスを kill する必要があります。kill は UNIX の用語です。 1. プロセスの ID 番号を確認するために、HP-UX のプロンプトで「ps -ef | grep vee」と入力します。行末に veetest と表示された行が表示されます。ログイン名の後ろの数字がプロセスの ID 番号です。たとえば、johnj *number*... veetest と表示されます。 2.「kill -9 *number*」と入力して VEE アプリケーションを終了します。その後、「veetest」と入力すると再起動できます。

Agilent VEE の再起動とプログラムの実行

1. PC:Windows では、[Start] ⇒ [Programs] ⇒ [VEE Pro 6.0] を選択します。

   HP-UX: ホーム・ディレクトリから、「veetest」と入力します。インストール時に、パスに含めるべき /usr/bin/veetest へのリンクが実

行ファイルに作成されています。作成されていない場合は、そのディレクトリに移動します。VEE を別のディレクトリにインストールした場合は、完全なパスを入力する必要があります。

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 1-34 のように、ツールバーの [Debug] メニューの下にあります。

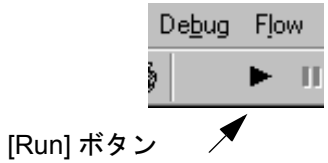


図 1-34. ツールバーの [Run] ボタン

メモ

PC: 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 VEE 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。

HP-UX: `veetest -r` *filename* コマンドを実行すると、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。VEE ディレクトリがパスに含まれていない場合は、完全なパス (通常は `/usr/bin/veetest -r` *filename*) を入力する必要があります。

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 VEE 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、`UserObjects` や `UserFunctions` 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。`UserObjects` および `UserFunctions` 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。`UserObjects` および `UserFunctions` については、第 2 章「Agilent VEE のプログラミング技術」の 80 ページの「UserObject の作成方法」セクションで詳述します。ここでは、VEE の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 1-35 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、VEE 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、VEE 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。

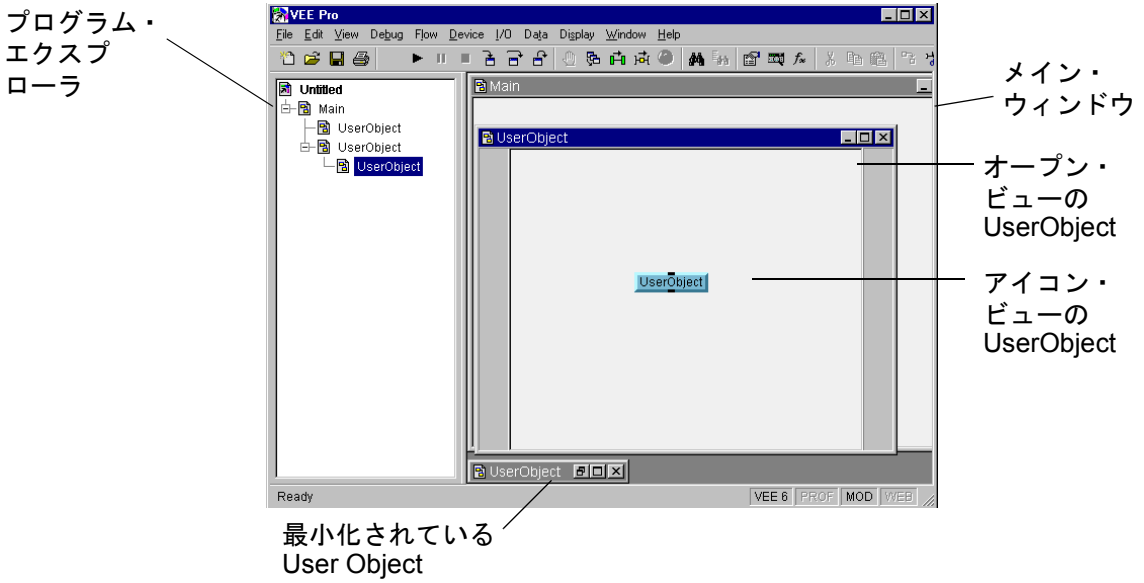


図 1-35. 作業領域における複数ウィンドウ

図 1-35 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。

メモ

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。

メモ

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。

VEE 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、54 ページの「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 1-36 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。

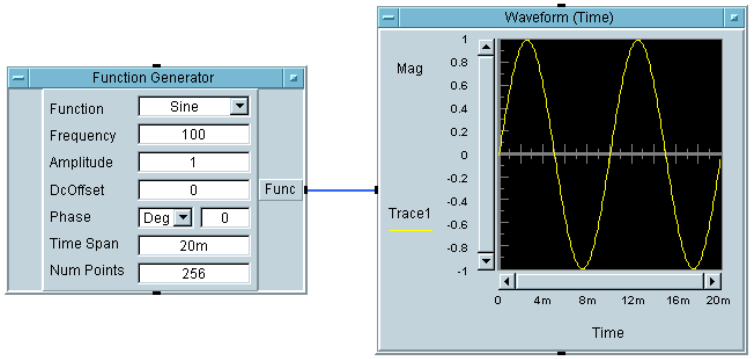


図 1-36. simple-program.vee プログラムの典型的な表示

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 1-37 のようにノイズ付きの正弦波を加えます。

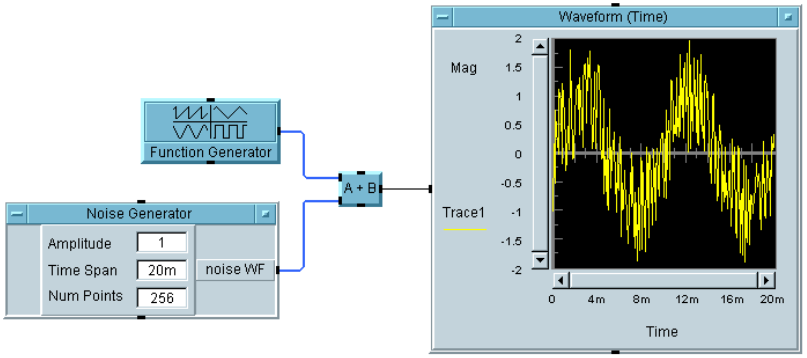


図 1-37. Noise Generator オブジェクトを追加した例

メモ

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 と Waveform (Time) の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 1-38 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 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。

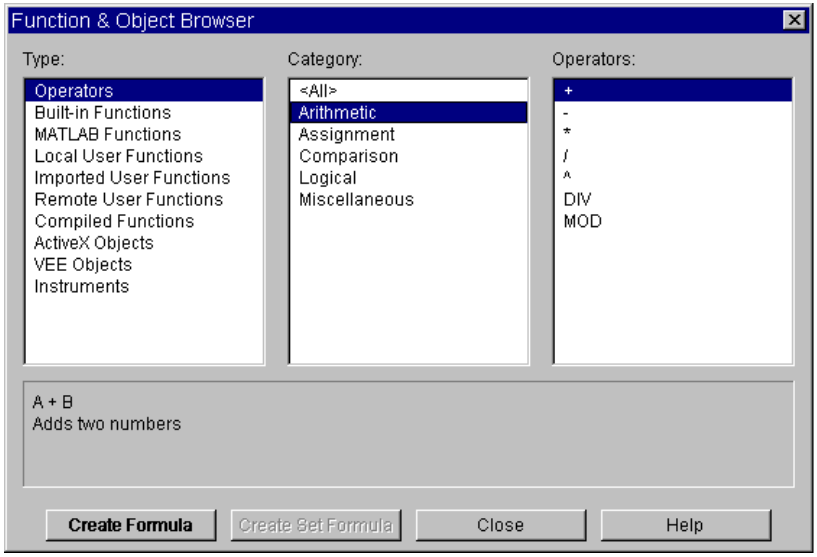


図 1-38. Function and Object Browser

5. 入力ピンと出力ピンを図 1-37 のように接続します。

6. プログラムを実行します。

   A+B オブジェクトは、Function Generator 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取ると、A+B オブジェクトは動作して 2 つの値を合計し、結果を Waveform (Time) オブジェクトに出力します。

メモ

VEE 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。

メモ

[Show Execution Flow] と [Show Data Flow] を有効にするには、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 1-39 のように表示されます。

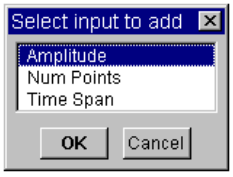


図 1-39. 入力端子を追加する例

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 1-40 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。

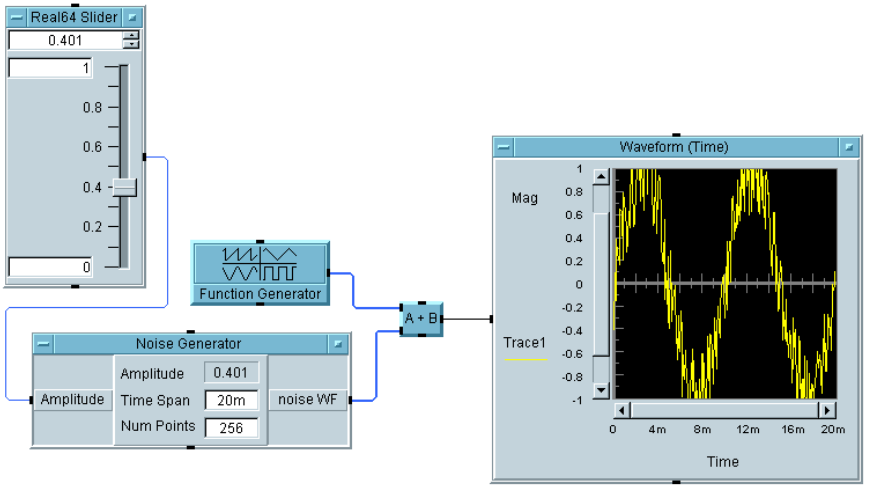


図 1-40. Real64 Slider オブジェクトを追加した例

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要素は 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。

メモ

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 1-41 のように Real64 Slider に表示される値と一致します。これらのオブジェクトはアイコン・ビューで表示されます。

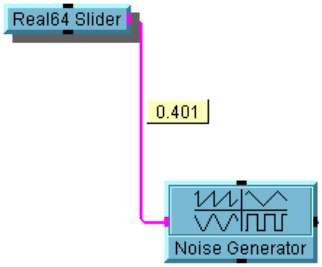


図 1-41. 出力ピン上の値を表示する

4. プログラムを「`simple-program.vee`」に保存します。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、必要な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 を起動する。
- 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・エクスプローラとは何か、またその目的は何かを説明する。
- 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・フローを説明できる。

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

平均的な必要時間 :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、UserObjects と呼ばれるカスタム・オブジェクトを作成できます。また、オペレータが使用するインタフェースをプログラムの必要な部分だけを表示するように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出したり、ファイルから VEE にデータを読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、Rocky Mountain Basic、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「UserObjects」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入れ子構造の UserObjects を入れることができます。UserObjects 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 UserObjects のアイコン・ビューと、対応する UserObjects 編集ウィンドウとを関連付けるには、編集ウィンドウで UserObjects 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、UserObjects 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 UserObjects 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、UserObjects 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 とおりあります。

- 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 2-1 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。

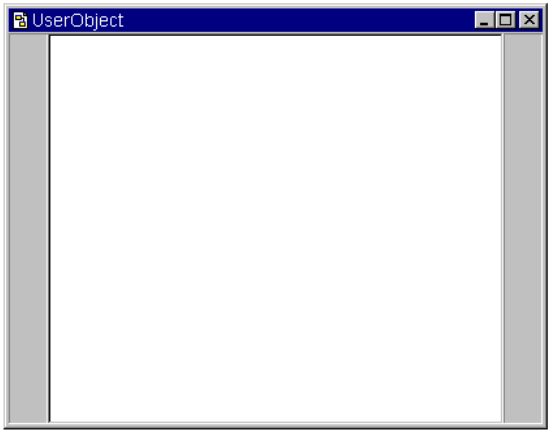


図 2-1. UserObject 編集ウィンドウ

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 とおりあります。

- [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。

メモ

UserObject のアイコン・ビューはメイン・ウィンドウに常にあり、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。

メモ

作業を開始する前に、メイン・ウィンドウ内の画面スペースをできるだけ確保するため、[View] メニューの [Program Explorer] の選択を解除してください。

では、プログラムから UserObject を作成してみます。

1. 73 ページの「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 [Cut] 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。

メモ

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 2-2 のようにオブジェクトを再配置します。

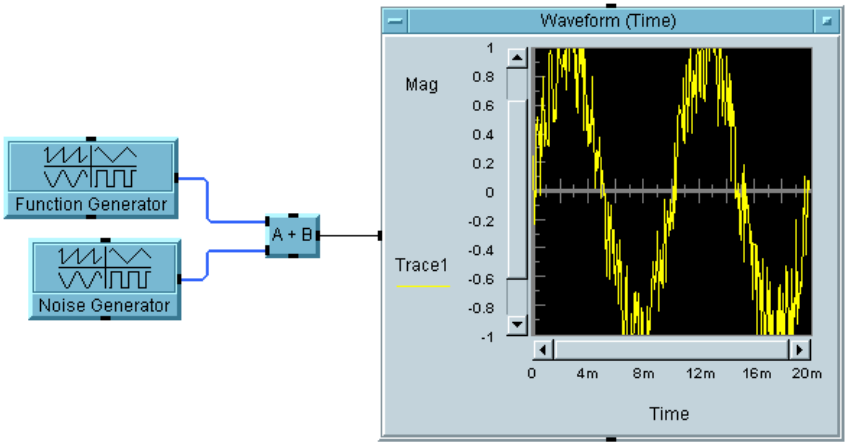


図 2-2. 初期段階の usrobj-program.vee

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。新しい名前を入力して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 および 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 2-3 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。

   ヒント : 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。

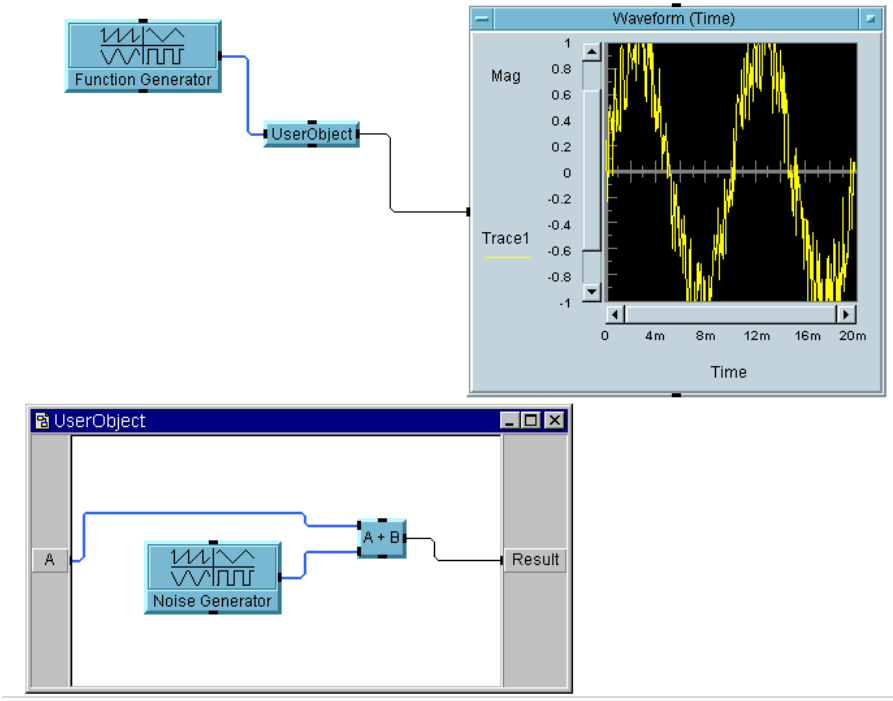


図 2-3. UserObject の作成方法

メモ

UserObject を作成する前にオブジェクトの整理しておくと、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ほどの大きさになります。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。

メモ

[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。

ヒント : 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 2-4 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。

   ヒント : 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。

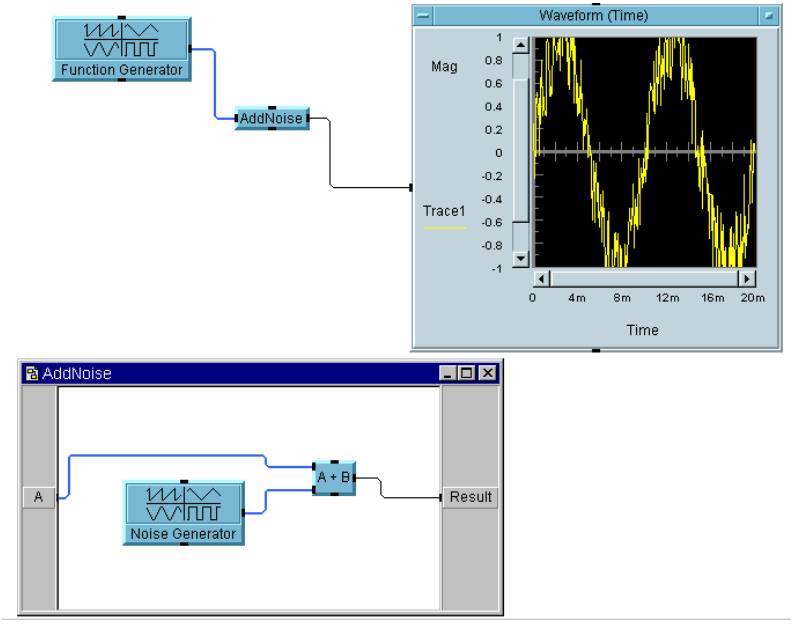


図 2-4. UserObject の名前を「AddNoise」に変更

8. [Run] ボタンをクリックして、図 2-5 のようにノイズ付きの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 (_) の形をした [最小化] ボタンをクリックします。

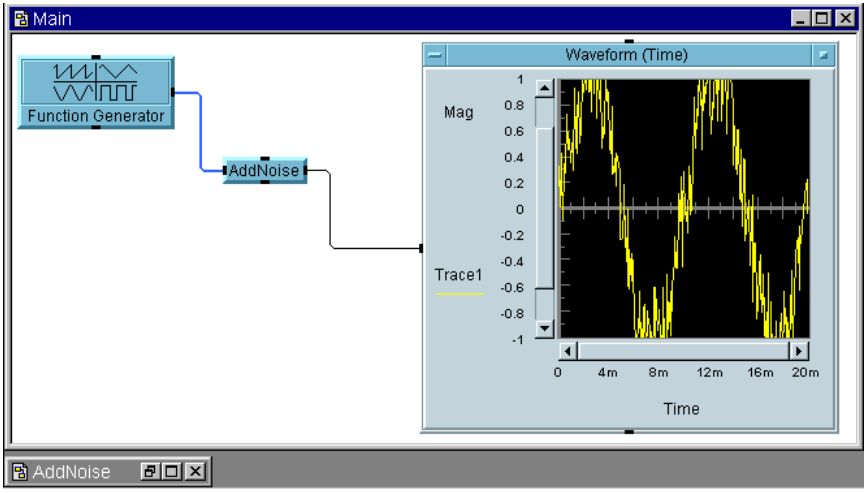


図 2-5. ノイズ付きの余弦波

効果的な UserObjects を作成するコツは、UserObjects にプログラム内で論理的な処理を実行させることです。この独自のオブジェクトは、スペースを節約するためだけの手段ではなく、プログラムを構造化する手段です。UserObjects を使用すると、VEE プログラムに「トップダウン」設計の手法を取入れることができます。VEE にはまた、UserFunction と呼ばれるオブジェクトがありますが、これは繰返し使用できるコード・モジュールです。UserObjects と UserFunctions についての詳細は、299 ページの第 8 章「Agilent VEE 関数の使用方法」を参照してください。

UserObjects 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] のサブメニューには、ダイアログ・ボックス作成のために次の選択肢があります。[Text Input]、[Int32 Input]、[Real64 Input]、[Message Box]、[List Box]、[File Name Selection] の 6 つです。テキスト、整数、実数用のメニューを選択した場合は、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。

   ヒント: 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 2-6 のように、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。

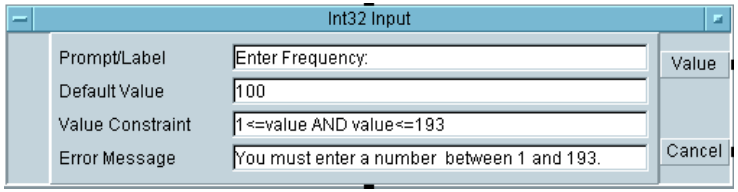


図 2-6. Int32 Input 設定ボックス

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。追加する入力端子選択のためのダイアログ・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 2-7 のように表示されます。

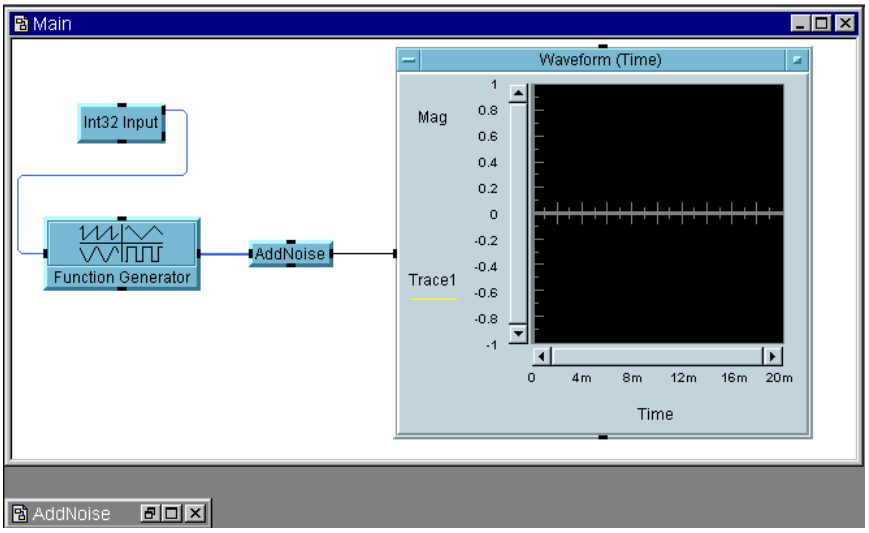


図 2-7. Int32 Input を追加した usrobj-program.vee プログラム

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、Enter Frequency: と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 2-8 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ます。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。

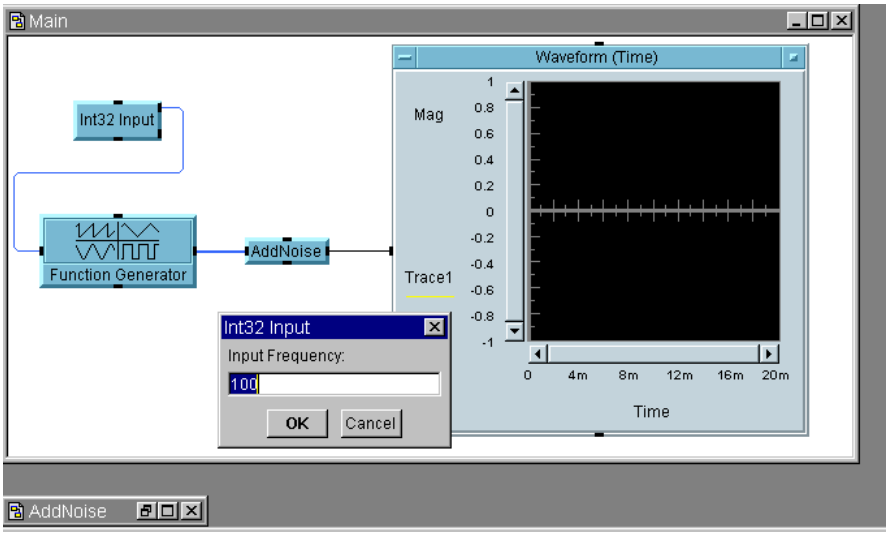


図 2-8. プログラム実行時のポップアップ入力ボックス

周波数に 193 以上を指定すると、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。

メモ

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・ファイルに書出したり、ファイルのデータを VEE に読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 2-9 のように表示されるはずです。

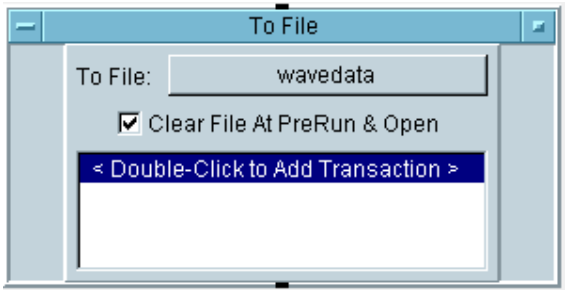


図 2-9. データ・ファイルの追加

3. データを書出すには、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 2-10 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] のほかにトランザクションの別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* マニュアルの付録および第 5 章「テスト結果の保管方法と読取り方法」で詳述しています。

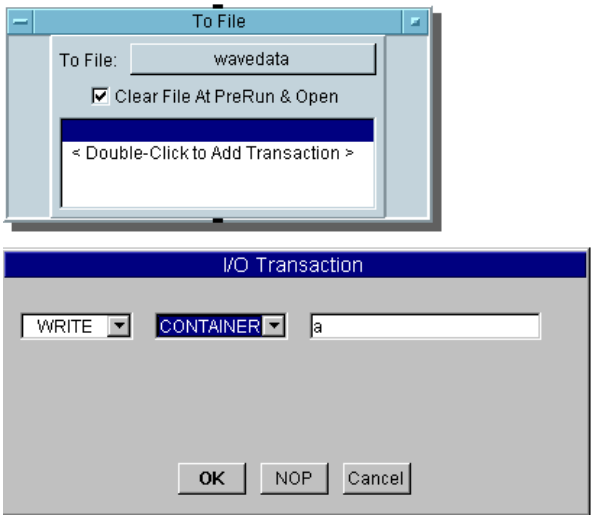


図 2-10. I/O トランザクションの選択

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 2-11 のように表示されます。

メモ

データ出力ピンは複数のデータ入力ピンに接続できます。

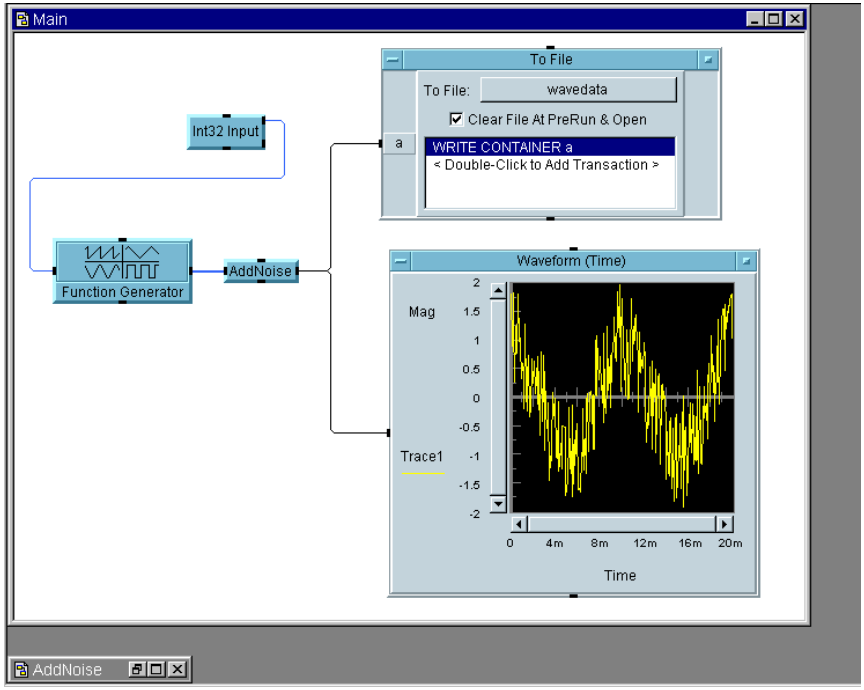


図 2-11. To File オブジェクトの追加

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 が出力したノイズ付きの余弦波を表示し、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・ビュー表示にします。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。

データを読込み直すには、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 に読込み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 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 2-12 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、データは読込まれる前に確実にファイルに書出されます。

7. プログラムを実行します。図 2-12 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。

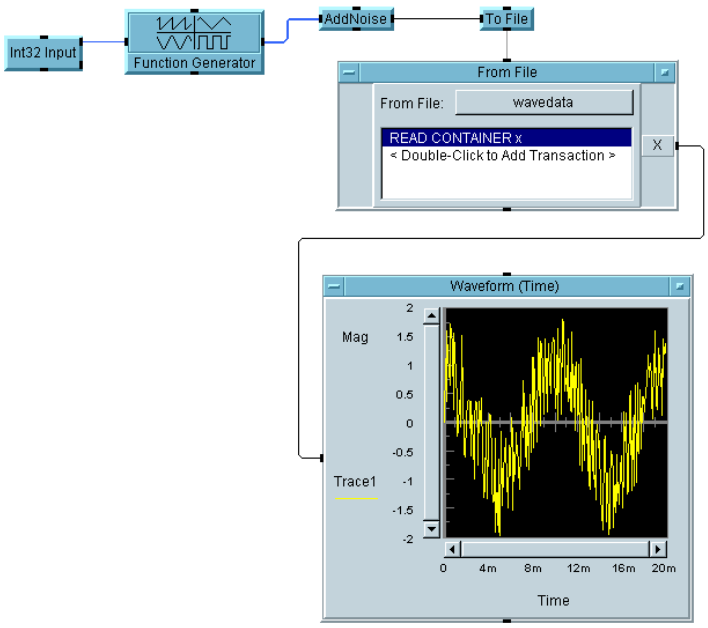


図 2-12. From File オブジェクトの追加

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、69 ページの「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 2-13 のように表示されます。

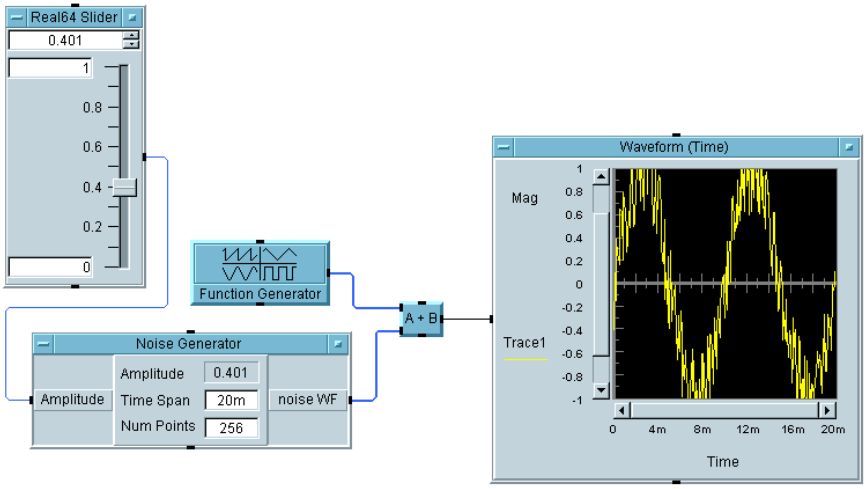


図 2-13. simple-program.vee

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` と `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 2-14 に示したようなパネルを作成します。

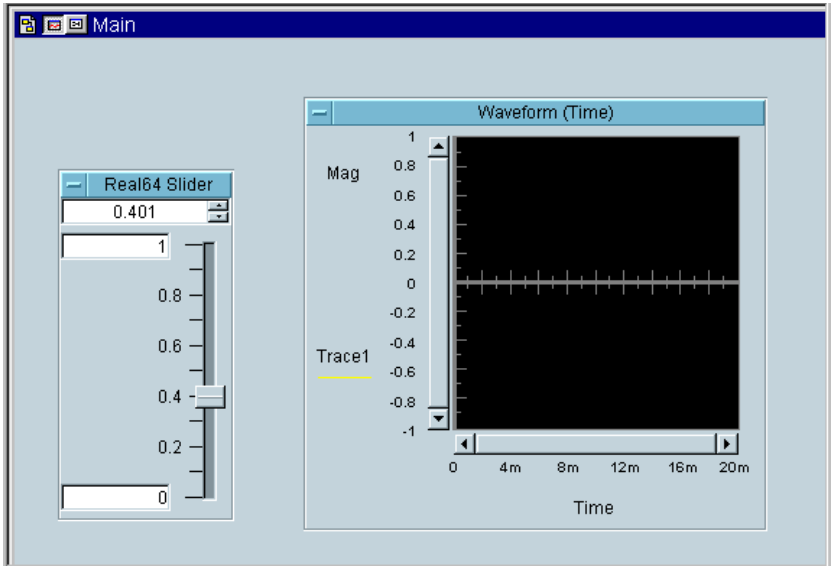


図 2-14. パネル・ビューの作成例

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、詳細ビュー表示に切替えます。[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・ビュー表示のメイン・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。次に `[Colors]` を選択し、`[Panel View]` ⇒ `[Background:]` ボタンをクリックして、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、[Colors] または [Fonts] タブをクリックして、色またはフォントを変更します。
- 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] ダイアログ・ボックスを開き、タブをクリックし [Appearance] フォルダを開きます。次に、[Border] グループの [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] ダイアログ・ボックスを開き、パネル・ビューに名前を付けます。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* マニュアル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 (+) のような数式処理は、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、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 2-15 の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 2-15 のような結果になります。

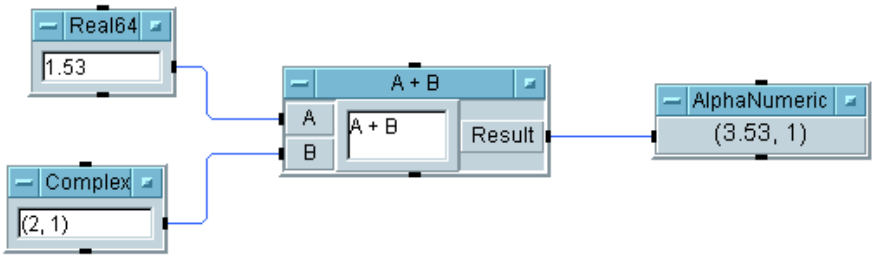


図 2-15. データ型の使用方法

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。

メモ

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。多くのプログラミング言語とは異なり、VEE のオブジェクトは 1 つの要素に対してだけよりも配列全体に対して処理を行います。

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。

ショートカット: [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 2-16 の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 2-16 のように表示されます。

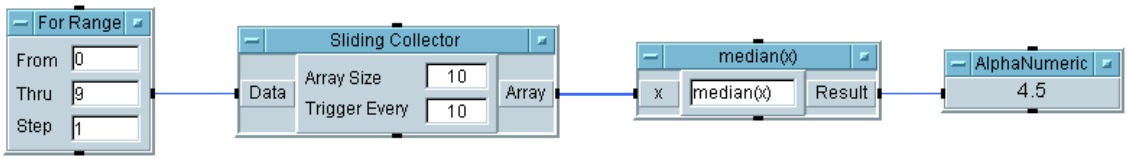


図 2-16. データ・オブジェクトの接続

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、「Reference」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 種類のリストで選択を行うことに

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 オブジェクトを使用して VEE で有効な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 100Hz の余弦波を作成します。[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 2-17 の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。

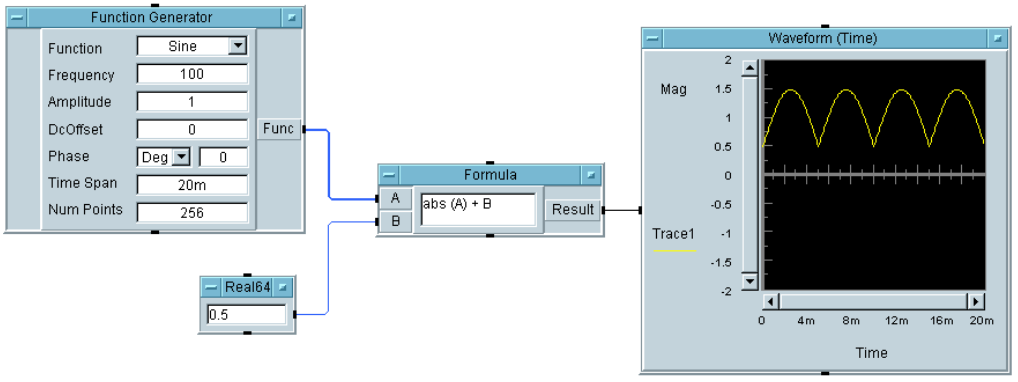


図 2-17. Formula オブジェクトのプログラムを作成する

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 を受取り、A の絶対値に B を加算します。
その結果、式 abs(A)+B は余弦波を「ブレンド」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 オブジェクトに式を読込む方が簡単で、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。

メモ

VEE の広範な数式能力をさらに増大するために、さらに数多くの算術関数が提供されており、MATLAB Script を統合することによって使用できます。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の詳細は、第 4 章「テスト・データの分析と表示」の 189 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 の主要機能の多くをデモンストレーションし、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。チュートリアルはまた、VEE を効果的に使用するためのポイントとなる概念を紹介します。

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な場合は、オブジェクト・メニューでオブジェクトに固有の情報を得ることができます。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。

メモ

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、第 1 章「Agilent VEE 開発環境の使用方法」の 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・ヘルプは次のトピックについて情報を提供します。

- すべてのメニュー項目およびメニューへのショートカット
- 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

ヘルプの表示、キーワード索引の使用、関連するトピックへのハイパーリンクの使用、検索を実行できます。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あります。

メモ

VEE にはまた、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、104 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] ⇒「How Do I...」⇒「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。

■ Agilent VEE の本バージョンの新機能を知る

[Help] ⇒ [Contents and Index] ⇒「What's New in Agilent VEE 6.0」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、2-4 ページの「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。
- プログラムを作成中に、Int16 Constant に 33000 の範囲を超える値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 2-18 のように、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。

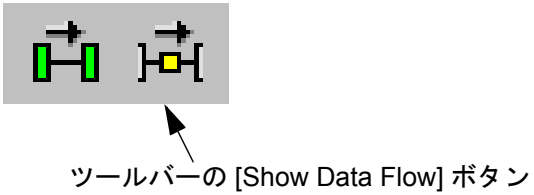


図 2-18. データ・フローの表示

表示をオフに切替えるには、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。

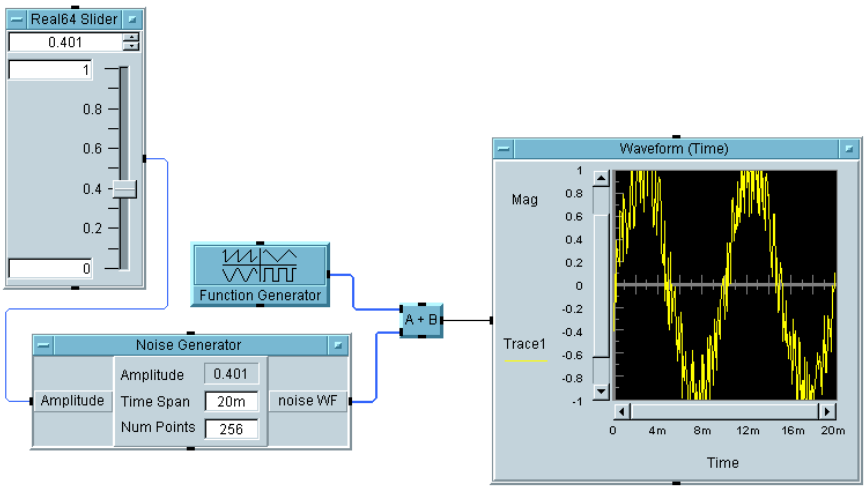
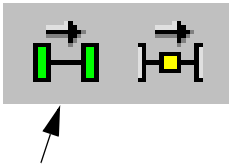


図 2-19. simple-program.vee におけるデータ・フロー

たとえば図 2-19 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 2-20 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 2-20. 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 2-21 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。

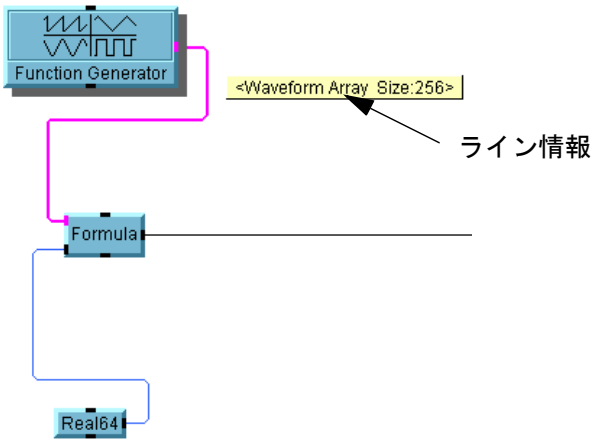


図 2-21. 出力ピン上の値を表示する

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 2-22 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。

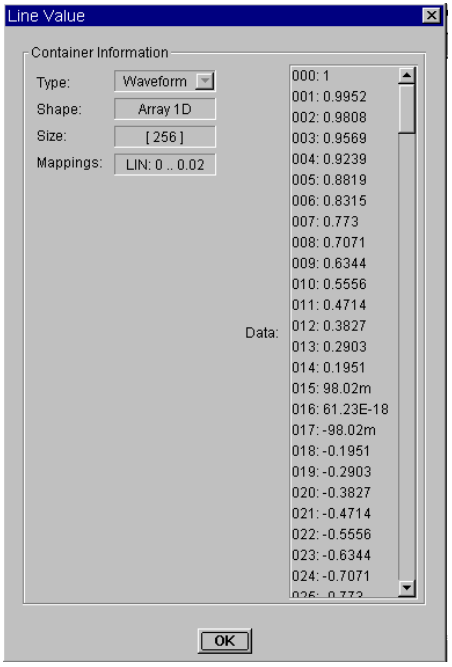


図 2-22. ライン情報の表示

端子を調べる

端子を調べるには、48 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・ビュー表示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強調表示され、オレンジ色の輪郭線で囲まれます。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。次に、[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 2-23 のように、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。

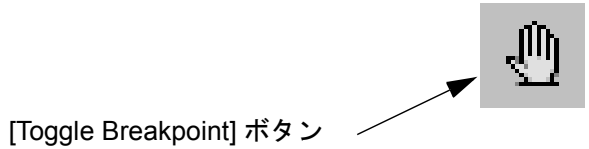


図 2-23. ブレークポイントの設定

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 2-24 に示したように、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。

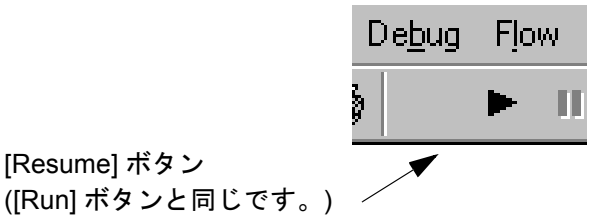


図 2-24. プログラムを再開する ([Run] ボタンと同じ)

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 2-25 に示したように、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。

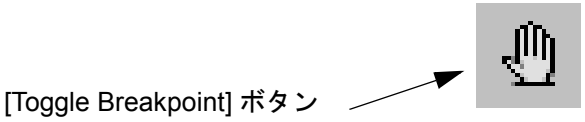


図 2-25. ブレークポイントのクリア

5. プログラムを休止または停止するには、図 2-26 に示したツールバーの [Pause] ボタンまたは [Stop] ボタンをクリックします。または `[Debug]` メニューにある [Toggle Breakpoint(s)] を選択します。

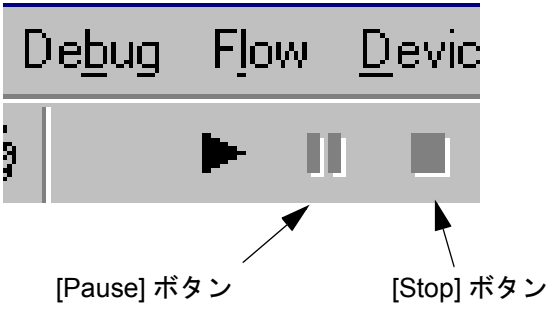


図 2-26. プログラムの休止または停止

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 2-27 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したが、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。

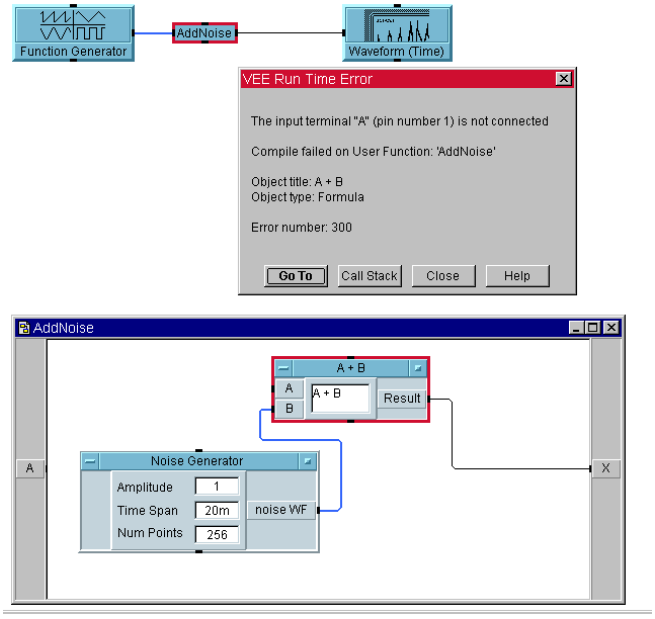


図 2-27. 実行時エラー・メッセージで [Go To] ボタンを使用した例

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。しかし、大きなプログラムでは、`Call Stack` が階層深くにネストされたエラーを探すのを助けてくれます。

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、プログラムが実行した処理の階層がリストされます。

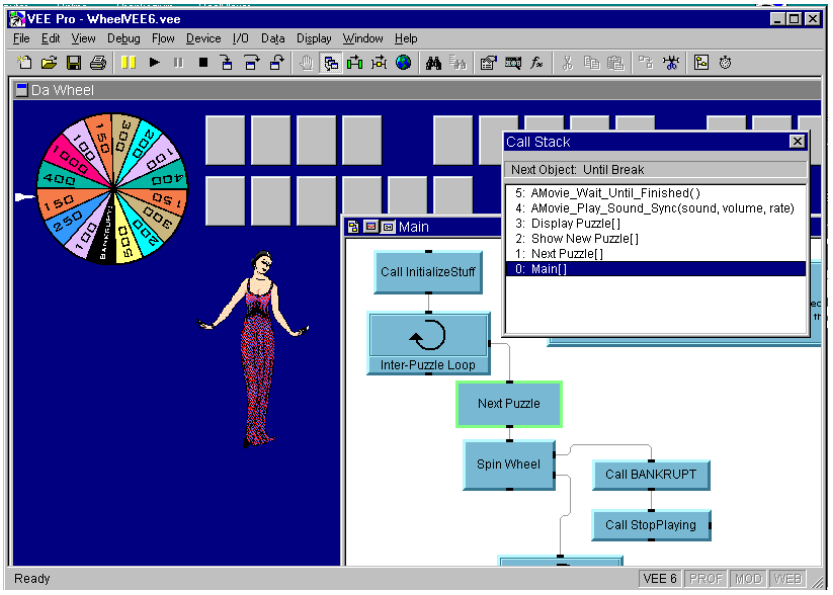


図 2-28. Wheel.exe での Call Stack の使用

`Call Stack` には、プログラムが実行した処理の階層が表示されます。図 2-28 は、VEE に搭載されているサンプル・プログラムの `Examples/Games` に収められている `Wheel.exe` プログラムを示しています。図 2-28 では、プログラムは現在 `AMovie_Wait_Until_Finished()` User Function を実行していますが、この User Function は、`AMovie_Play_Sound_Sync` によって呼出されており、さらに元をたどるとメイン・ウィンドウの `Next_Puzzle` によって呼出されています。Call Stack のリスト内の項目をダブルクリックすると、VEE はその関数の位置と関数を表示します。

オブジェクト内部のイベントの順番を追跡する

図 2-29 は、オブジェクト内部のイベントの順番を示しています。

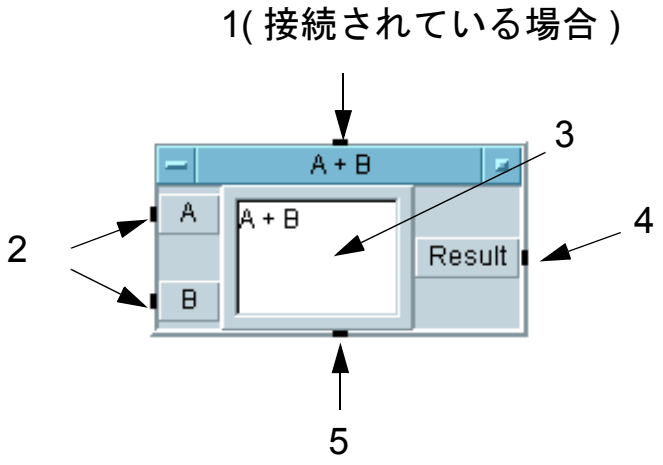


図 2-29. オブジェクト内部のイベントの順番

図 2-29 では、ピンは次のように動作します。

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取るまで動作しません (VEE 用語集の「ピン」参照)。しかし、シーケンス入力ピンは、必ず接続する必要があるわけではありません。

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取ったという信号を受取って初めて、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 2-30 は、Waveform の画面にカスタム・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。オブジェクトは、この操作を行うためデータをコントロール・ピン上に持つ必要はありません。オブジェクト自体が動作するのではなく、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。

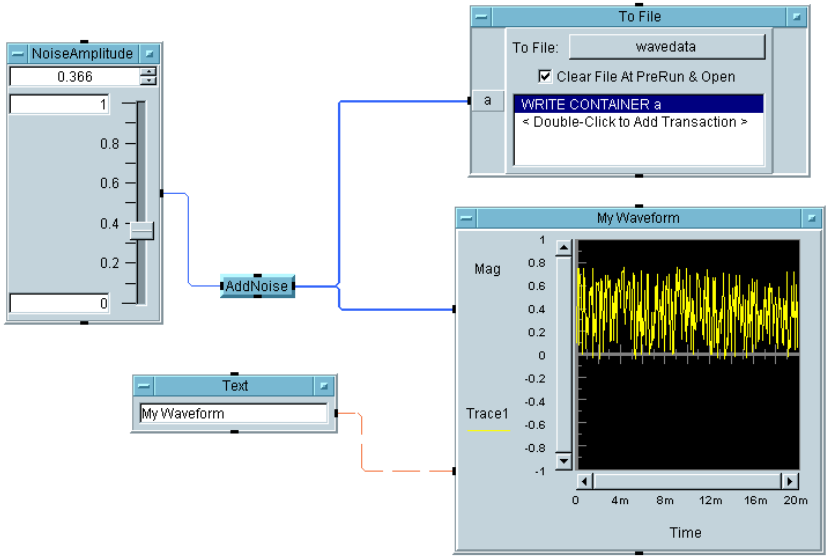


図 2-30. コントロール・ラインを使用してカスタム・タイトルを実行

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 2-31 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]` にありますが、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。

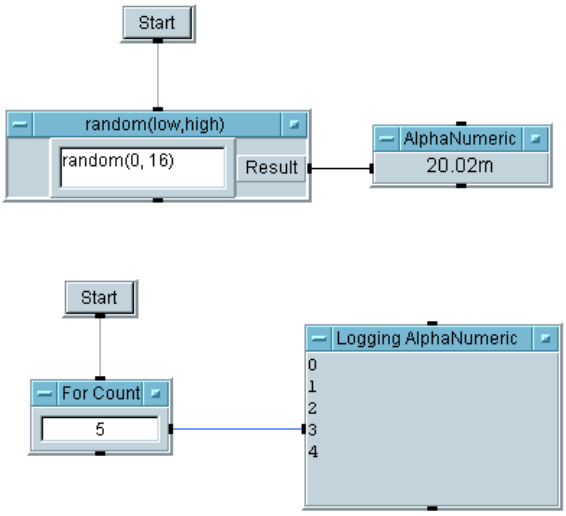


図 2-31. 別々のスレッドを動作させる Start オブジェクト

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。

プログラム中のステップ実行

プログラム中のステップ実行は、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 2-32 に示したように、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。

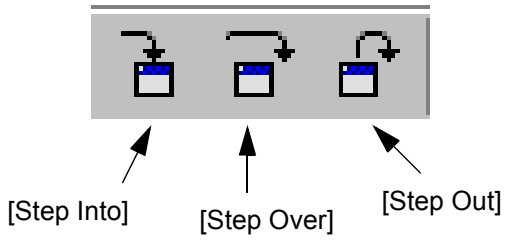


図 2-32. ツールバー上の [Step Into]、[Step Over]、[Step Out] ボタン

- [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、`UserObjects` または `UserFunctions` 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` をすべて実行します。

たとえば、プログラムをステップ実行するには次のようにします。

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、プログラム中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取るまで動作できません。次に

Waveform(Time) オブジェクトを実行します。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。

メモ

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。UserFunctions についての詳細は、299 ページの第 8 章「Agilent VEE 関数の使用方法」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 は、プログラム内にあるそのオブジェクトまたは関数の全リストと場所を表示します。詳細は、327 ページの「大規模プログラム内の関数を検索する方法」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作成

   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、次のように入力します。

   ```
   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] は [Built-in] 関数を、[Category] は [All] を、[Functions] 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。

メモ

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。

メモ

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 2-33 のように、乱数が表示されます。

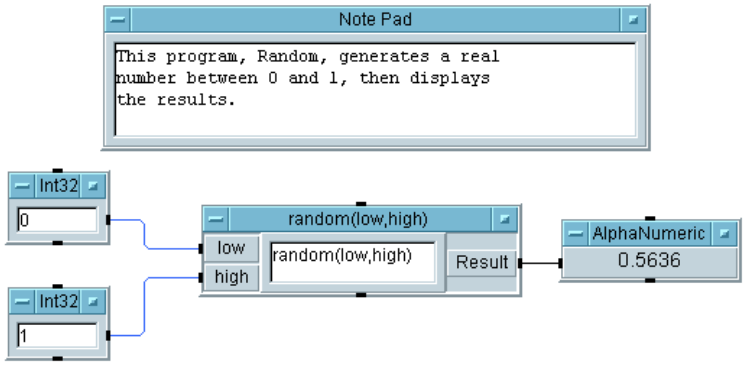


図 2-33. Random プログラム

7. [File] ⇒ [Save As] を選択して、「Random.VEE」と入力し、[OK] をクリックします。評価キット・ソフトウェアをお使いの場合は、EVAL.VEE に保存してください。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・データの分析と表示」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 を再取込みして表示します)

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。

メモ

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。

メモ

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 2-34 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。評価キット・ソフトウェアをお使いの場合は、プログラムを EVAL.VEE に保存してください。

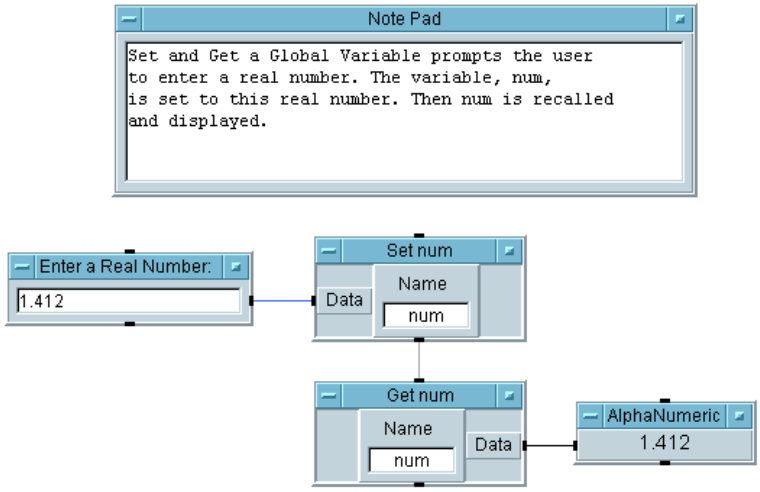


図 2-34. グローバル変数の設定と表示

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、これらのレベルは数字で示されます。

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は、オブジェクト・メニューに Description 項目があり、このメニューのダイアログ・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・ファイルでは、ドキュメントを画面コピーに関連付けることができます。このセクションでは、[Description] ダイアログ・ボックスに入力します。

1. Random.vee プログラムを開きます。

2. メイン・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、図 2-35 のようにテキストを入力します。入力したら、[OK] をクリックします。

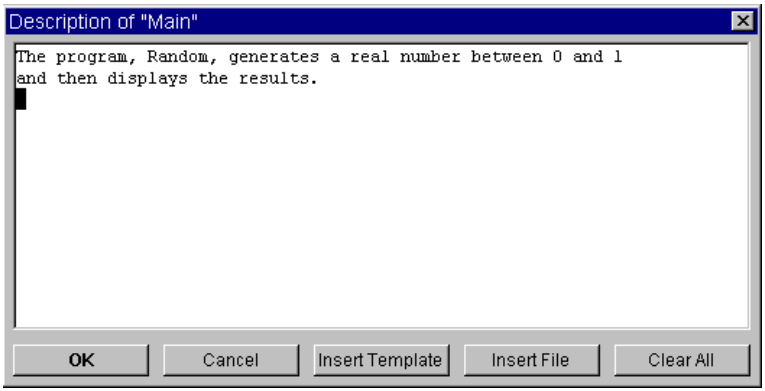


図 2-35. [Description] ダイアログ・ボックス

メモ

[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。また、このダイアログ・ボックスにはファイルやテンプレートを挿入できます。

ドキュメントの自動生成

ファイルまたはプログラムのドキュメントを生成するには、次のようにします。

1. Random.vee をオープンします。[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「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 2-36、図 2-37、図 2-38 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 2-36 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 2000
Date documented:  Mon Feb 28 14:43:27 2000
VEE revision: 6.0
Execution mode: VEE 6
Convert Infinity on Binary Read: no

I/O Configuration
  My Configuration (C:\WINDOWS\Local Settings\Application
Data\Agilent VEE\vee.io)
```

図 2-36. 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
   3. the results.
   4.

M.1: Main/random(low,high)
 Device Type              : Formula
  Input pin  1            : low (Any, Any)
  Input pin  2            : high (Any, Any)
  Output pin 1            : Result
 Formula                  : random(low,high)
```

図 2-37. ドキュメント・ファイルの中間部分

図 2-37 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 2-38 にこのドキュメント・ファイルの末尾部分を示します。

```
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
 Password masking         : off
 Indices Enabled          : on
 Int32 Value              : 1
M.5: Main/AlphaNumeric
 Device Type              : AlphaNumeric
  Input pin  1            : Data (Any, Any)
 Clear At Prerun          : on
 Clear at Activate        : on
 Indices Enabled          : on
```

図 2-38. ドキュメント・ファイルの末尾部分

メモ

[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、必要ならトピックを復習してください。

- UserObject を作成する。また、UserObjects 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

平均的な必要時間 :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上でコマンドを伝達したりデータを受取ることができます。この技術は、Rocky Mountain Basic のようなテキスト形式言語でコマンド文字列を使用するのと同じです。

- Open Data Acquisition Standard **ドライバ** (ODAS **ドライバ**) は、ActiveX オートメーション技術を使って PC プラグイン・カード (PCPI カード) を制御します。ODAS ドライバは標準の形式であるため、PC プラグイン・カードのメーカやほかのサード・パーティから提供されています。

- 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) として搭載されており、ODAS ドライバに似ています。しかし、ODAS ドライバは標準であり簡単に使用できます。

- VXI *plug&play* **ドライバ**は、C 関数を呼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 では、異なる計測器メーカ向けに 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、開発時間を最大限節約します。図 3-1 は、パネル・ドライバの例です。

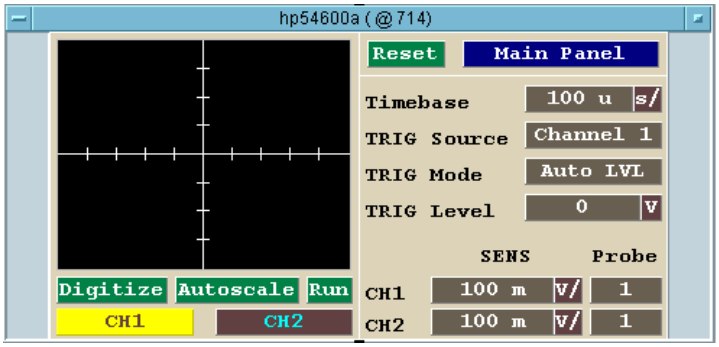


図 3-1. HP54600A スコープ・パネル・ドライバ

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上で、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、必要な性能によって決まります。図 3-2 は、Function Generator の制御に Direct I/O を使用している例です。

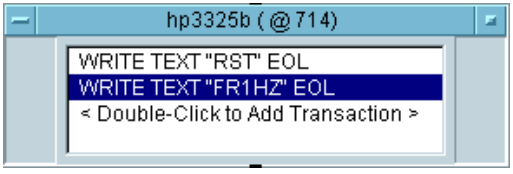


図 3-2. Function Generator オブジェクトの Direct I/O

ODAS ドライバを使った PC プラグイン・ボード制御

ODAS ドライバは、PC プラグイン・カードのメーカから提供されます。また標準のドライバであるため、サード・パーティ製の ODAS ドライバも提供されています。VEE では、`Formula` オブジェクトで PC プラグイン・ボード関数を選択することによって、ODAS ドライバを使って PC プラグイン・ボードを制御できます。

PC プラグイン・ボードの制御には、一般に専用 DLL よりも ODAS ドライバが標準として使用されます。また 1 つの PC から別の PC への移植も ODAS ドライバの方がうまくいきます。図 3-3 は、VEE の `Formula` オブジェクトを使って ODAS ドライバで PC プラグイン・ボードを制御している例です。

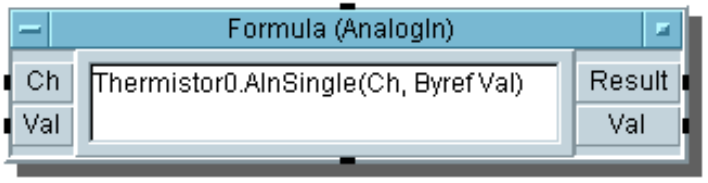


図 3-3. VEE プログラムにおける ODAS ドライバ・オブジェクト

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、`Call` オブジェクトでライブラリ関数を呼出すことによって PC プラグイン・ボードを制御できます。図 3-4 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。

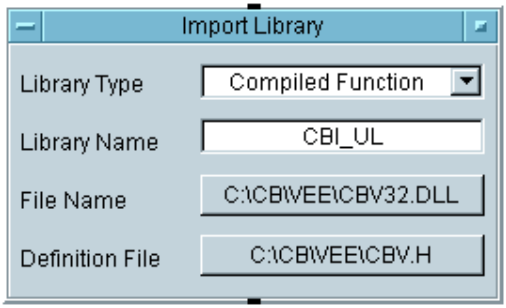


図 3-4. PC プラグイン・ライブラリのインポート

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* マニュアル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合せください。VEE では、VXI *plug&play* ドライバを呼出すことによって計測器を制御できます。図 3-5 は、VEE から VXI *plug&play* ドライバを呼出している例です。

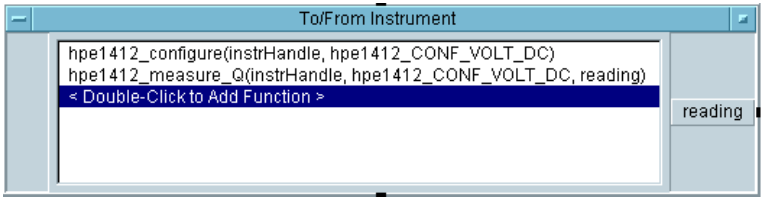


図 3-5. VEE から VXI *plug&play* ドライバの呼出し

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 3-6 に示すダイアログ・ボックスを作業領域の左上まで移動します。

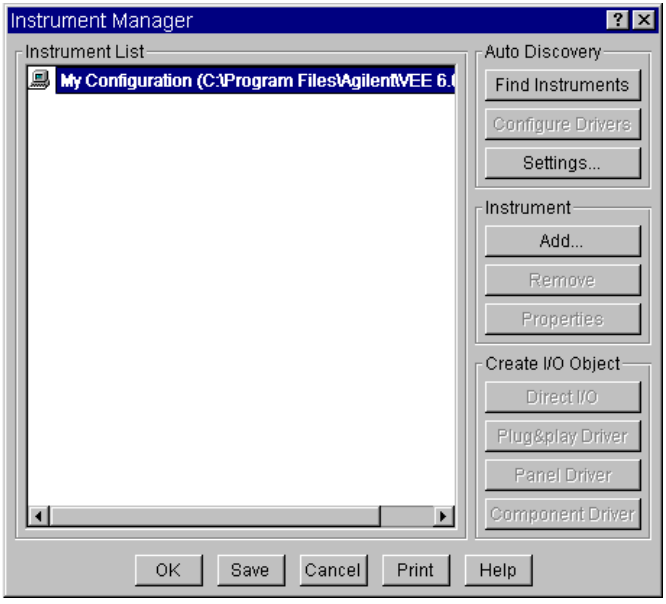


図 3-6. [Instrument Manager] ダイアログ・ボックス

メモ

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構成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 3-7 のような [Instrument Properties] ダイアログ・ボックスが表示されます。

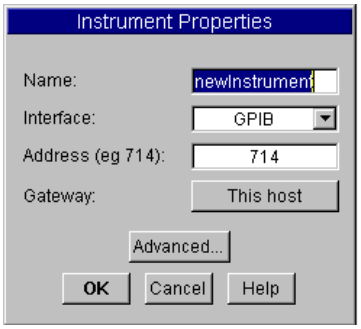


図 3-7. [Instrument Properties] ダイアログ・ボックス

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

[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]、[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・バスのアドレス (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* マニュアル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 3-8 のように表示されます。

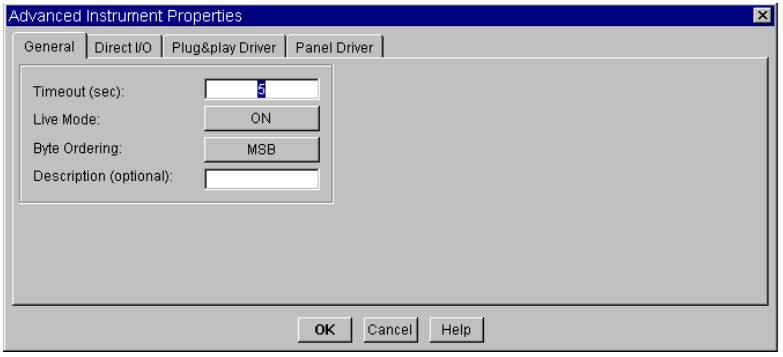


図 3-8. [Advanced Instrument Properties] ダイアログ・ボックス

`General` フォルダでの設定項目は次のとおりです。

[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。計測器が接続されていない場合は、`[OFF]` に設定します。VEE のデフォルトの設定は `[ON]` です。

[Byte Ordering] デバイスがバイナリ・データの読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 3-9 のようになります。

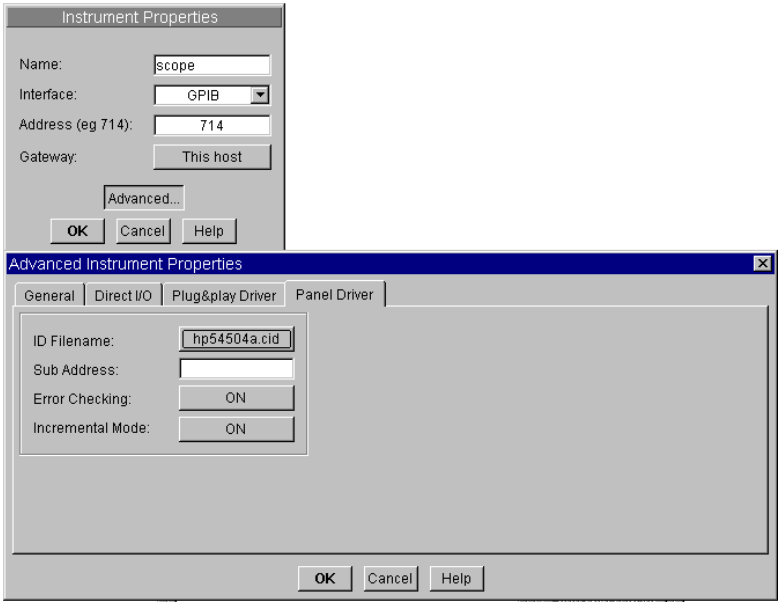


図 3-9. [Panel Driver] フォルダ

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、VEE の改訂版と一緒に指定したディレクトリ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。

メモ

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 3-9 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。

[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] は特に処理量を増大させたい場合にオフにしますが、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、計測器のステートを指定する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 3-10 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。

ヒント : 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。

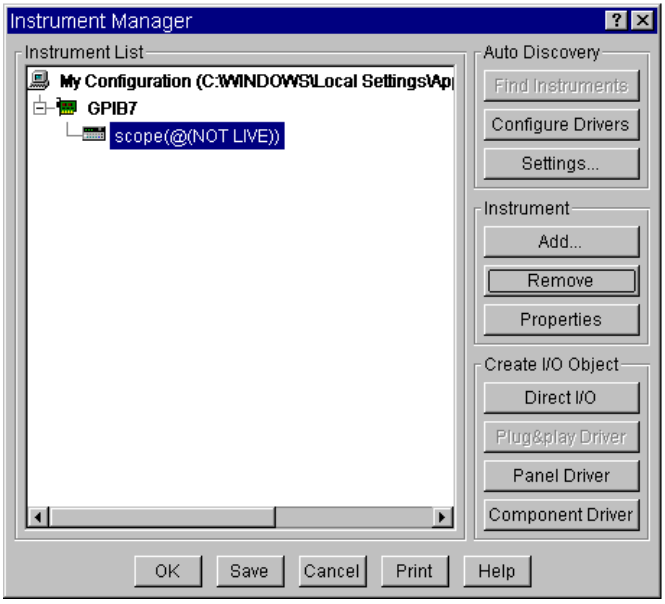


図 3-10. 計測器リストにスコープを追加

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。

メモ

[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 はまた、コンポーネント・ドライバを提供しますが、これは 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* マニュアル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 3-11 のように表示されます。

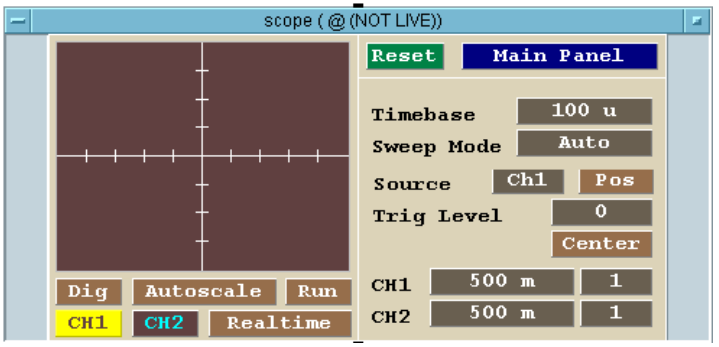


図 3-11. scope(@(NOT LIVE)) の選択

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

号と置換えてください。709 の 9 は、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反映されます。

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・ドライバの設定変更

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。

[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、[InstrumentManager] 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。

メモ

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 3-12 のように [Triangle] を選択します。

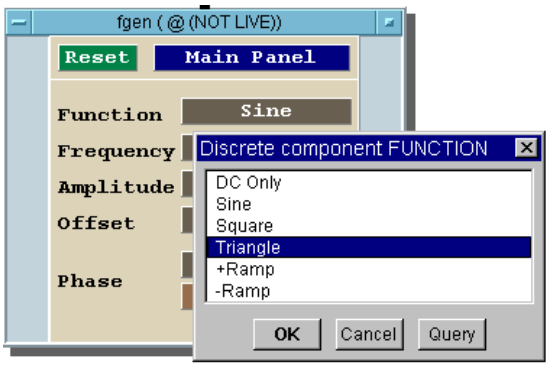


図 3-12. fgen の関数ポップアップ・メニュー

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 [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、パネル・ドライバ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 3-13 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。

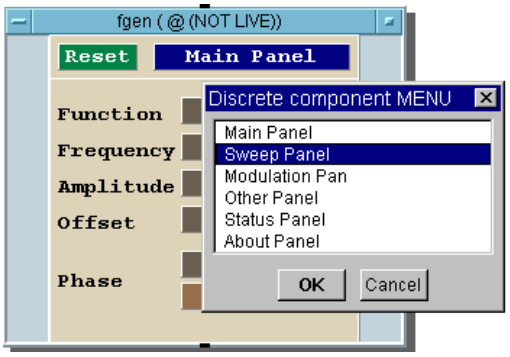


図 3-13. [Discrete Component Menu] の [Sweep Panel]

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取ることができます。図 3-14 は、入力 / 出力端子領域を示します。

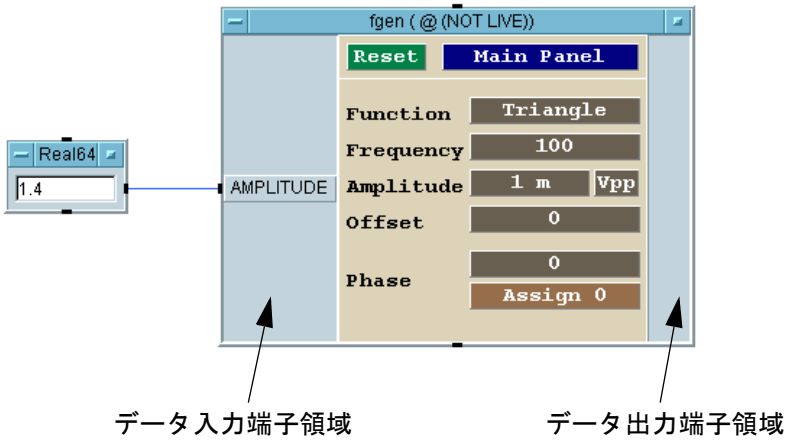


図 3-14. ドライバ上のデータ入力 / 出力端子領域

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。

メモ

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。

メモ

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、処理速度を上げたい場合は、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

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 3-15 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。

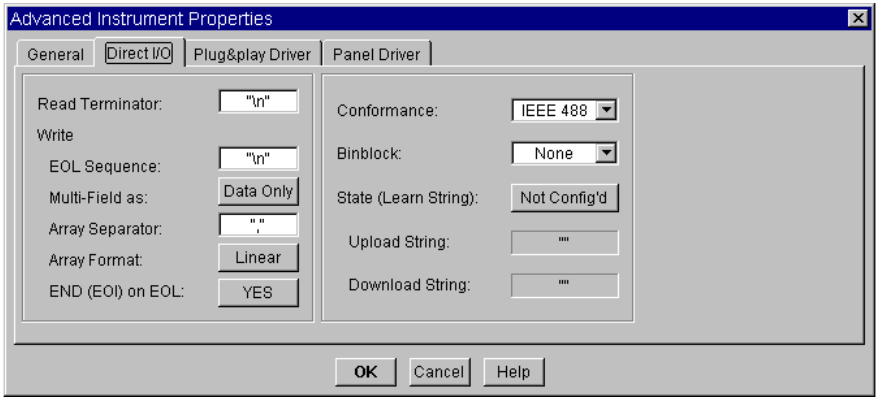


図 3-15. Direct I/O 設定フォルダ

メモ

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* マニュアル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 3-16 は、作成された Direct I/O オブジェクトを示しています。

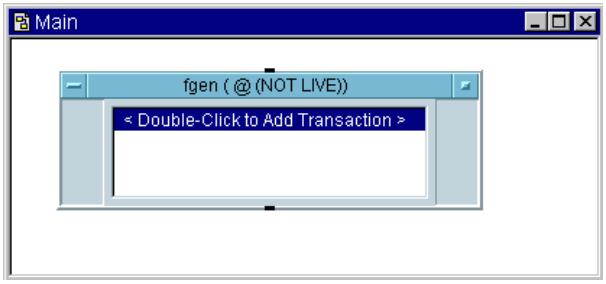


図 3-16. Direct I/O オブジェクト

プログラム中で `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。必要な場合は、次へ進む前に 150 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 3-17 に示した `[I/O Transaction]` ダイアログ・ボックスを表示します。

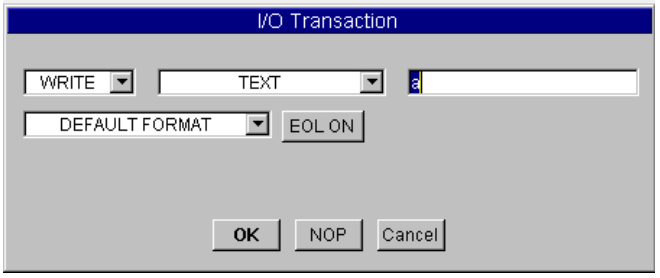


図 3-17. [I/O Transaction] ダイアログ・ボックス

`[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. 各項目はデフォルトの設定、`[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"`」(引用符も含めて) を入力し、`[OK]` をクリックします。

図 3-18 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。

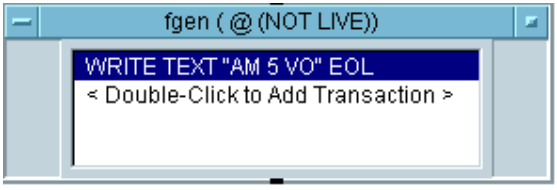


図 3-18. Direct I/O トランザクション

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、それぞれの式はコンマで分けられています。周波数は変数 A によって表されますが、これは Direct I/O オブジェクトのデータ入力端子になります。

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。数値は Direct I/O オブジェクトで受取られ、コマンド文字列

が Function Generator に周波数を出力させます。Direct I/O のセットアップは図 3-19 のようになります。

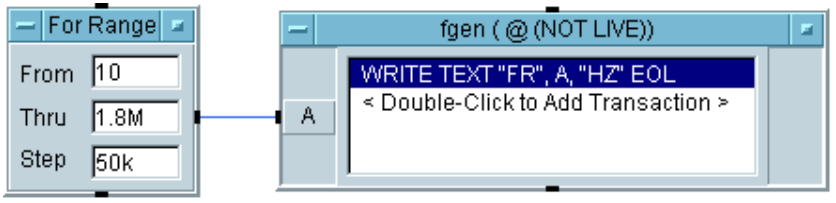


図 3-19. 入力変数を使った Direct I/O のセットアップ

6. (オプション) コンピュータを所有している場合、HP3325B 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、計測器は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込むには、読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、VEE の [I/O] メニューにある [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 が周波数を送信すると、マルチメータはデータを読込んで結果を 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。

メモ

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* マニュアルで詳述しています。

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>] の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 コマンドが書込まれます。T5 は、マルチメータに対する単一のトリガのためのコマンドです。

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。Tab 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込まれ、「x」という名前でデータ出力ピンへ送られます。

メモ　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 のように表示されます。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。

メモ

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 3-20 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。

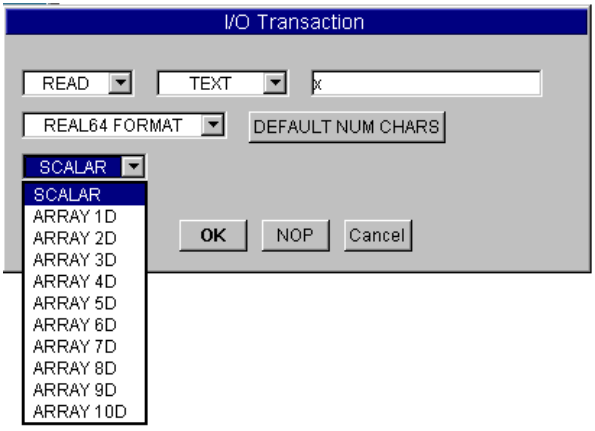


図 3-20. READ トランザクションの設定

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 3-21 のようになります。

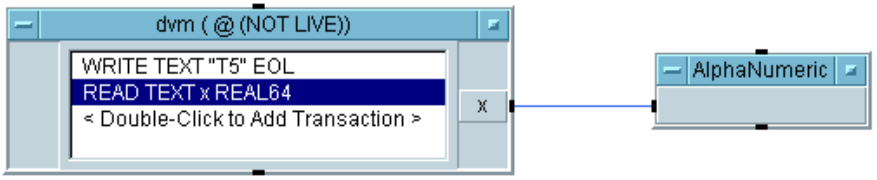


図 3-21. 測定値を読込むように設定された Direct I/O

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* マニュアル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 のデータと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のすべてについて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。より複雑な例については、*VEE Pro Advanced Techniques* マニュアル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」能力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、後でプログラム中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、この特定の Direct I/O オブジェクト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、VEE 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] がステートの照会に使用するコマンドを特定し、[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特定します。図 3-22 に例を示します。

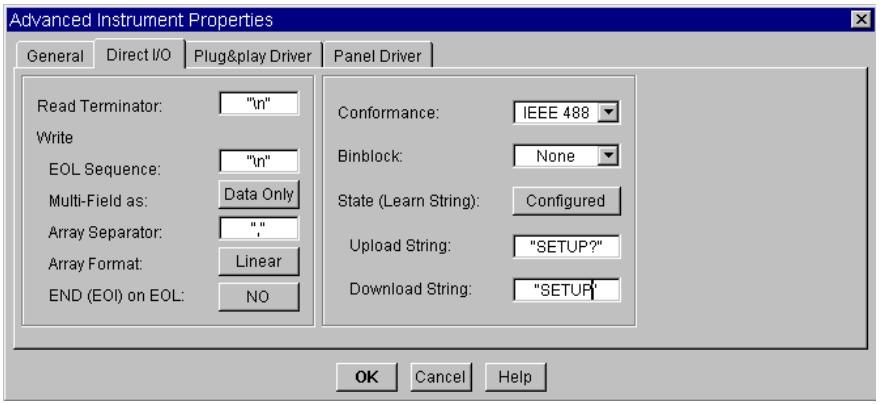


図 3-22. HP54100A 用のラーン・ストリング設定

[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 4881 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 および Download String というラベルが付いた 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. PC プラグイン・カード・メーカから提供される ODAS ドライバ
2. 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。
3. ComputerBoards や Meilhaus のような 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、425 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。

ODAS ドライバの使用方法

製造メーカの説明書に従って、PC プラグイン・ボードのインストール、ODAS ドライバ・ソフトウェアのインストール、ODAS 設定ユーティリティの実行を行います。次に、VEE でドライバの設定を行います。

1. `[I/O]` ⇒ `[Instrument Manager]` を選択します。`[Find Instruments]` を選択します。`[Instrument Manager]` には図 3-23 のようなリストが表示されます。

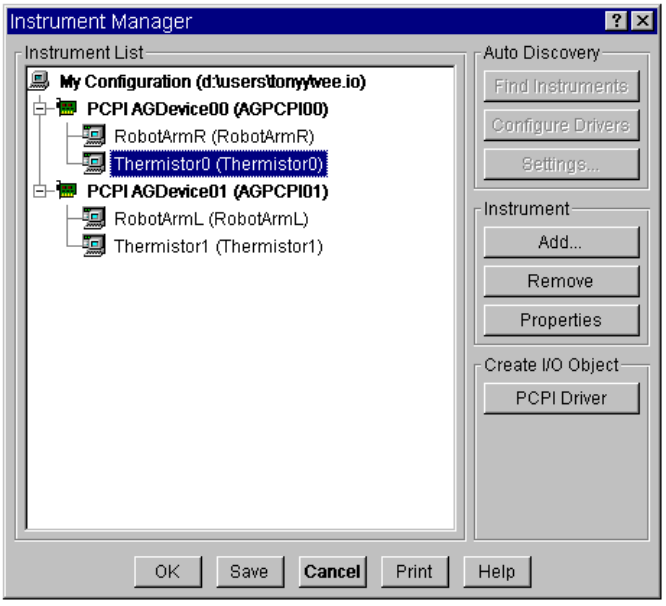


図 3-23. [Instrument Manager] での ODAS ドライバリスト

2. `[Thermistor0]` のようなサブ項目を 1 つ選択し、`[Create I/O Object]` の下にある `[PCPI 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。図 3-24 のように、VEE に `Formula` オブジェクトが表示されます。

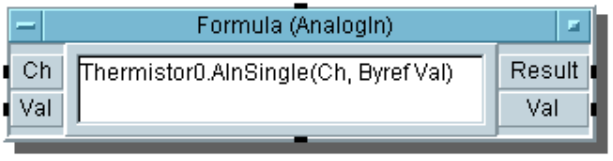


図 3-24. ODAS ドライバを設定した PC プラグイン・カードを表す Formula オブジェクト

ODAS ドライバを設定した PC プラグイン・カードの使用についての詳細は、*VEE Pro Advanced Techniques* マニュアルを参照してください。

Data Translation 社の Visual Programming Interface(VPI)

Data Translation 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取得性能を発揮します。Data Translation 社の Open Layers 規格の柔軟性を導入することにより、50 以上のデータ取得ボードにアクセスできます。

VPI は、低いチャンネル数を要求するプラグイン ISA、PCI、USB ベースのデータ取得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・データ取得アイコンを 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・ボードであり、すべて VEE によってサポートされています。

ソフトウェア・インタフェースは Amplicon の AMPD10 ドライバ・パッケージの一部ですが、このパッケージは、Windows 用マルチスレッド DLL で提供される 32 ビット API であり、割込み操作によるデータ取得をサポートします。効率的で柔軟性のあるプログラム作成のために、VEE 固有の定義ファイルおよび 1 つのプログラムに 8 つまでボードを使用できる機能を利用して、API は、コンパイル済み関数として 100 以上の関数を呼出すことができます。

Amplicon 自身のプラグイン・ボードには、シリアル通信デバイスが含まれますが、Amplicon はさらに、データ取得、シリアル通信、GPIB アプリケーションのために他製造元のボードを幅広く提供します。

図 3-25 で示したように、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。

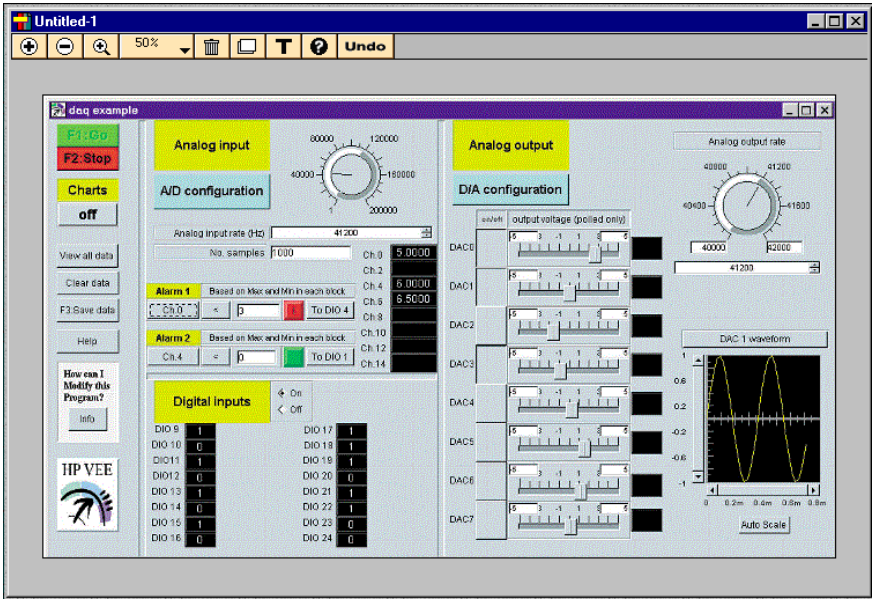


図 3-25. Amplicon によるデータ取得例

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・メーカ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、製造元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すると、ComputerBoards I/O ライブラリに測定関数を呼出すことができます。製造元が提供するデモ・プログラムから抜粋した次の図を参照してください。

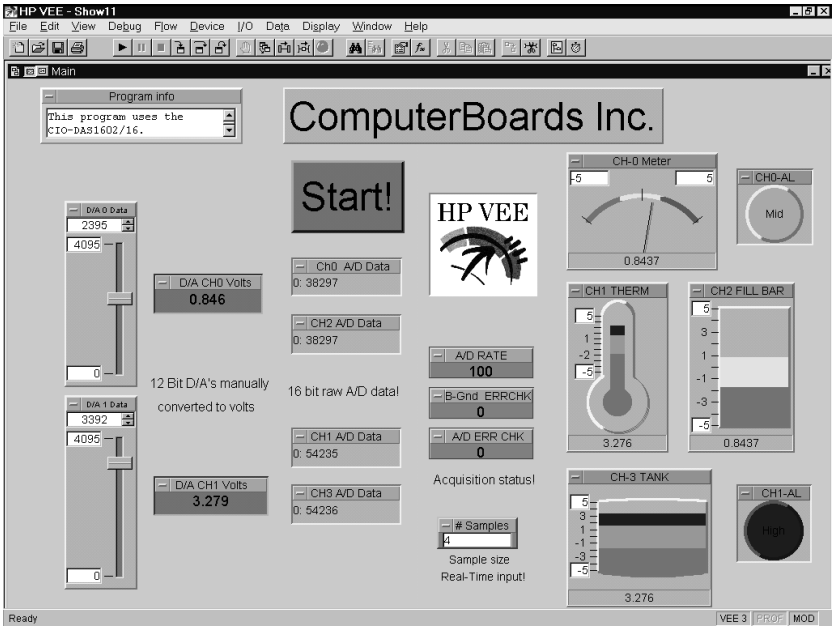


図 3-26. VEE による ComputerBoards 100kHz ボードの使用

図 3-26 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 3-27 は、これらのデータ取得関数の呼出しを可能にする 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ます。

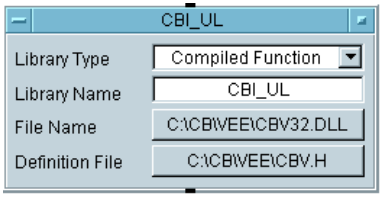


図 3-27. ComputerBoards I/O ライブラリのインポート

Meilhaus Electronic の ME-DriverSystem

Meilhaus Electronic は、ヨーロッパで代表的な PC ベースのデータ取得 / 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取得ボード (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 3-28 は VEE における ME Board メニューを示しています。

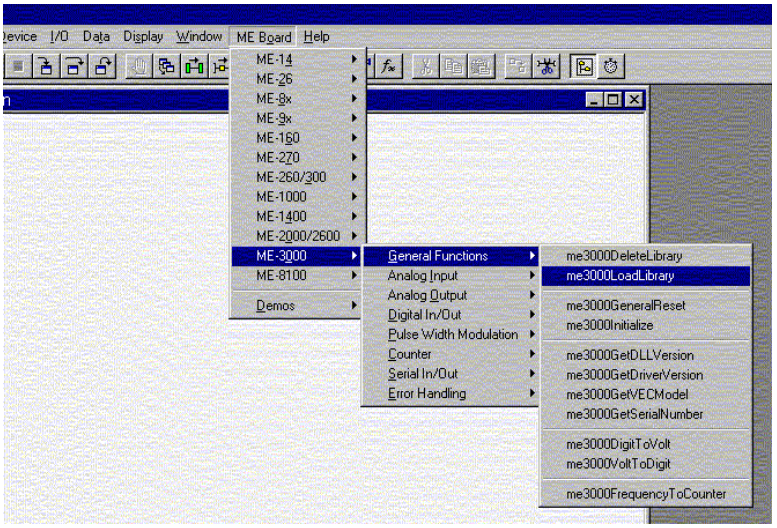


図 3-28. VEE における ME Board メニュー

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 3-29 は、データ取得ボード ME-3000 のユーザ・パネルを示します。

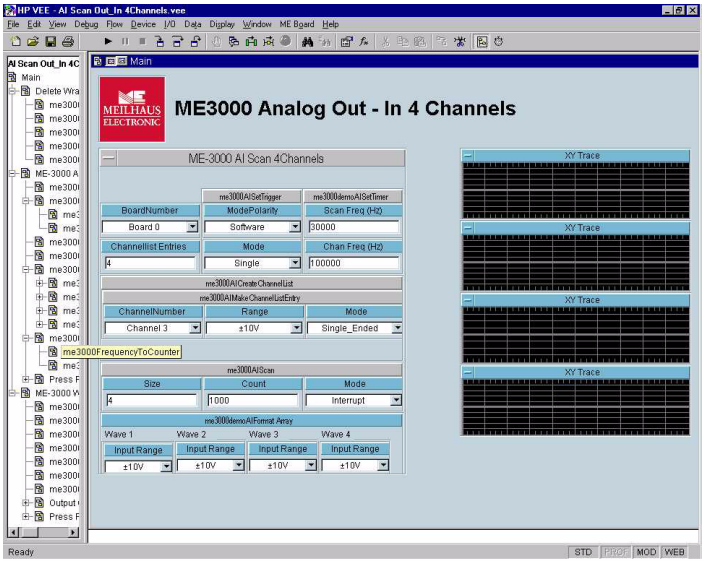


図 3-29. データ取得ボード ME-3000 のユーザ・パネル

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 3-30 は関数パネルを示します。

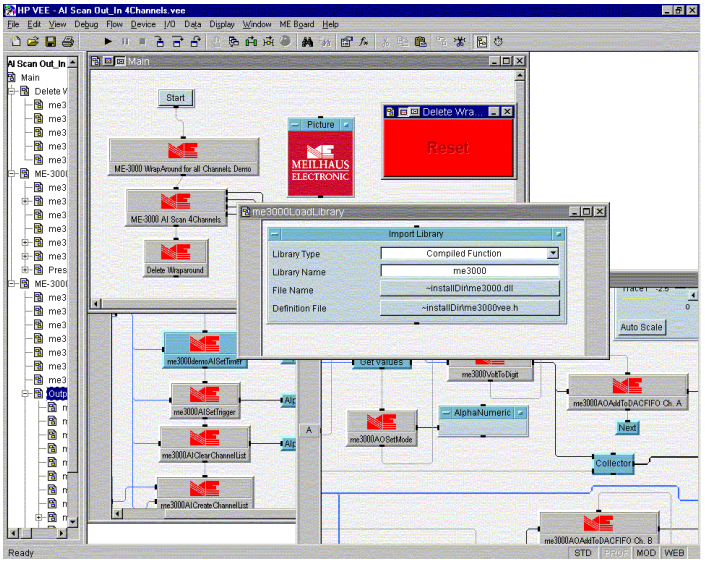


図 3-30. ME-DriverSystem の関数パネル

VXI *plug&play* ドライバの使用方法

VXI *plug&play* ドライバは、さまざまな計測器メーカから提供され、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 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` 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。ほかの計測器用の VXI *plug&play* ドライバを入手するには、計測器メーカにお問合せください。

例題 3-4: VXI *Plug&play* ドライバの設定

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`」のように名前を入力し `[Advanced]` をクリックし、`[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 3-31 に示すように HPE1412 ドライバを使用します。

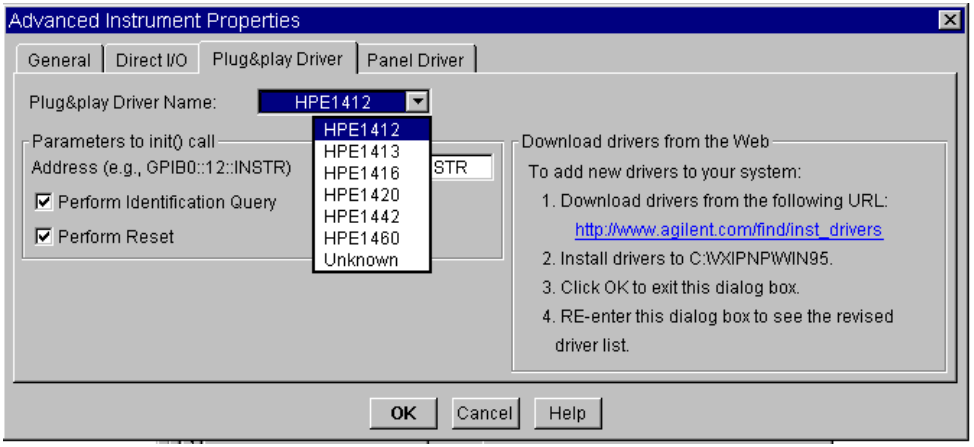


図 3-31. VXI *plug&play* ドライバの選択

4. `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。

5. `[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。

メモ

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、使用すべき適切な関数を選択するためのパネルを提供します。

6. `<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 3-32 のように `[Select a Function Panel]` が表示されます。図 3-33 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。

メモ

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。

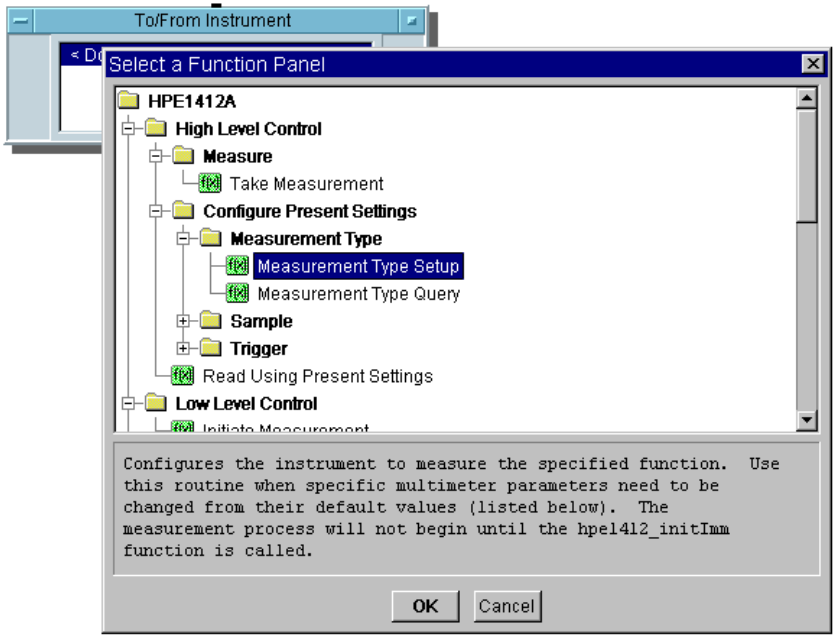


図 3-32. VXI *plug&play* ドライバ用関数の選択

7. `[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 3-33 のように、デフォルトの [DC Voltage] を選択します。

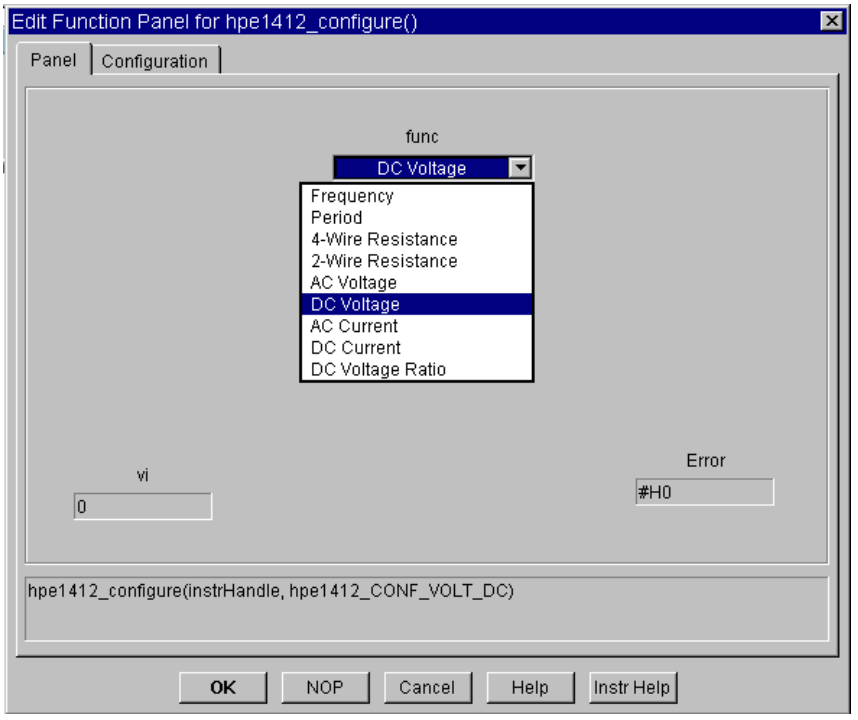


図 3-33. HPE1412 の [Edit Function Panel]

8. [OK] をクリックします。これで、図 3-34 のように、[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。

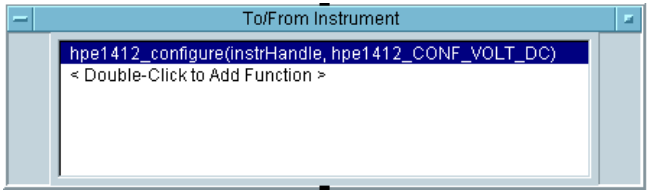


図 3-34. VXI *plug&play* オブジェクトにおける DC Voltage Function

9. [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 3-35 のようにします。

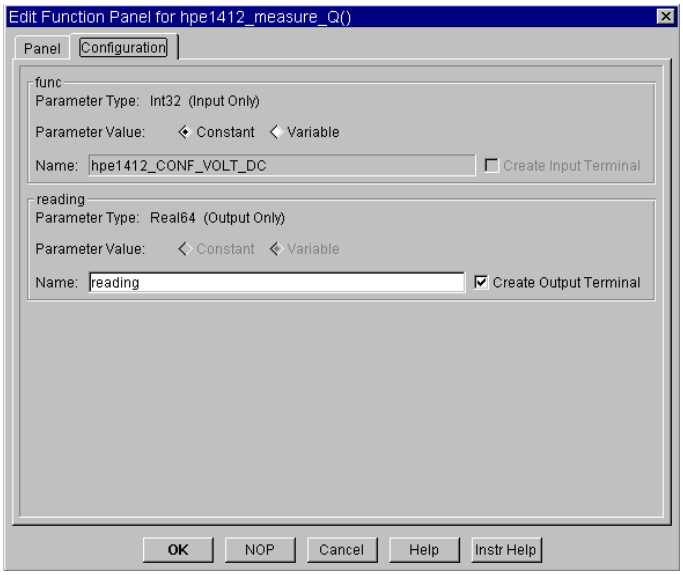


図 3-35. [Edit Function Panel] の [Configuration] フォルダ

10. [OK] をクリックします。2 番目の関数の呼出しが [To/From Instrument] オブジェクトに図 3-36 のようにリストされます。

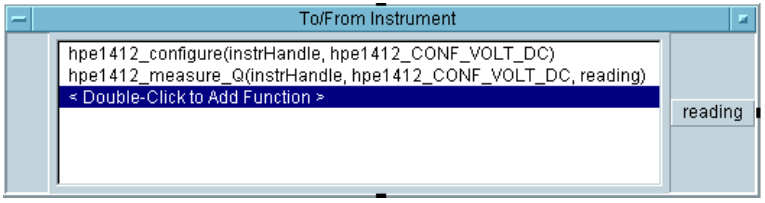


図 3-36. DC 読込みのために用意された HPE1412 ドライバ

そのほかの I/O 機能

- 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。
- I/O メニューでは、Bus I/O Moniter 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。
- 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・サーバが含まれます。詳細は、*VEE Pro Advanced Techniques* マニュアルを参照してください。
- 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* マニュアル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、必要なら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。
- ステート・ドライバで計測器を設定する。
- Direct I/O で計測器を設定する。
- 計測器ドライバの設定を変更する。
- 入力 / 出力コンポーネントを追加 / 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込む。
- Direct I/O を使って計測器からデータを読込む。
- 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・データの分析と表示

テスト・データの分析と表示

この章の内容

- VEE のデータ型
- VEE の分析機能
- 演算オブジェクトの使用方法
- Formula オブジェクトの使用
- MATLAB 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

平均的な必要時間 :1.5 時間

概要

この章では、VEE の分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分析することもできます。詳細は、249 ページの第 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。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、404 ページの「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 4-1 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 のどのようなデータ型に対しても動作しますし、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 のデータ型を受取った場合、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べきです。また、VEE と MATLAB でのサポートされるデータ型の違いについての知識も役立ちます。詳細は、193 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、短時間で読むことができる簡単な説明です。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 4-1. Agilent VEE のデータ型

データ型	[Description]
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)。
Int32	32 ビットの 2 の補数整数です (-2147483648 ～ 2147483647)。

表 4-1. Agilent VEE のデータ型 (続き)

データ型	[Description]
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) の形式で表した大きさ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。

表 4-1. Agilent VEE のデータ型 (続き)

データ型	[Description]
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。
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。

メモ

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 の分析機能

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、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、データ分析、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 VEE 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 VEE 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 4-1 のように、[Type:] は [Built-in Functions]、[Category:] は [Probability & Statistics]、[Functions:] は [random] を選択します。

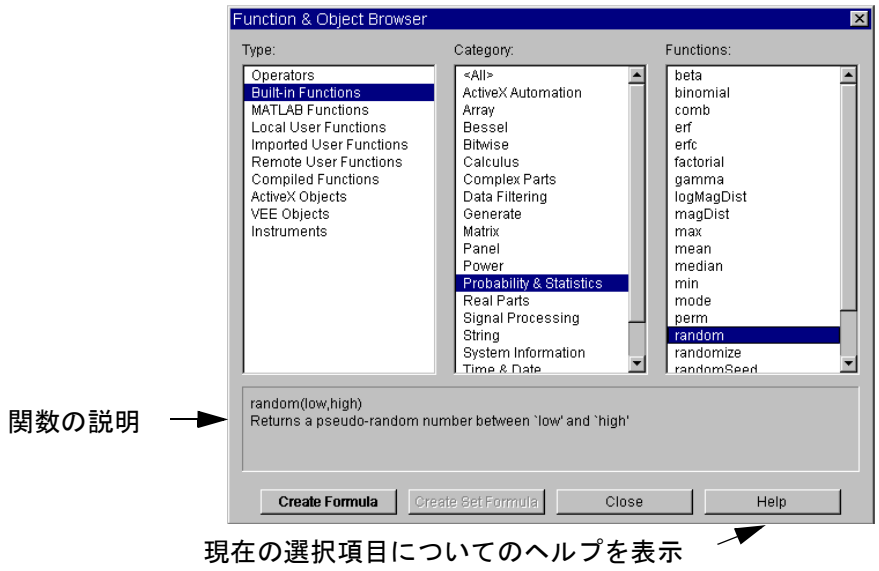


図 4-1. [Function & Object Browser] における VEE 関数

[Function & Object Browser] には、図 4-1 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 4-2 のように、[Type:] は [MATLAB Functions]、[Category:] は [Interpolation & Polynomials]、[Functions:] は [poly] を選択します。

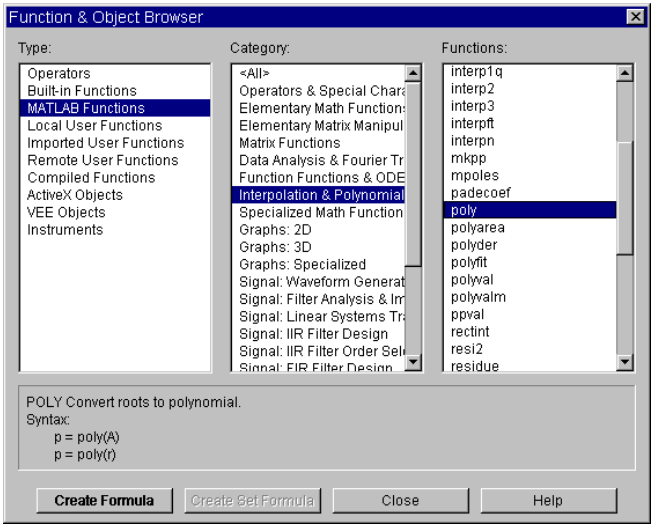


図 4-2. [Function & Object Browser] における MATLAB 関数

[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、189 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。

1. [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。

2. [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。

メモ

図 4-3 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。

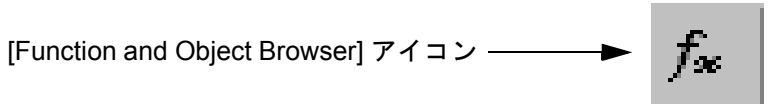


図 4-3. [fx] アイコンを使った [Function and Object Browser] の表示

3. sdev() のオブジェクト・メニューを開いて [Help] を調べます。

メモ

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 のデータ型を出力します。

4. Function Generator を sdev(x) に接続します。

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

6. プログラムを実行します。図 4-4 のように表示されます。

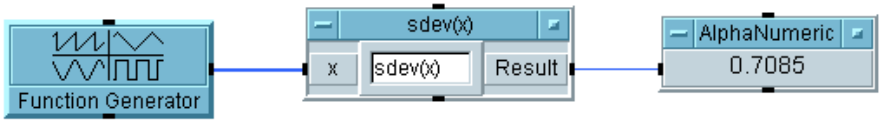


図 4-4. 標準偏差の算出

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 4-5 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。

メモ

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。

また、Formula 内で式を編集するために、標準の編集コマンドを使用できます。たとえば、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。

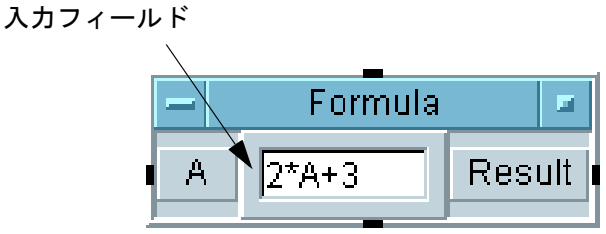


図 4-5. Formula オブジェクト

メモ

[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。

メモ

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。

メモ

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 4-6 のように結果の 127 が表示されます。

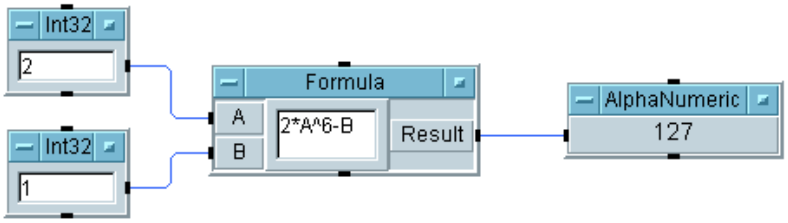


図 4-6. 式の評価

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」として呼出すことも「独立したオブジェクト」として呼出すこともでき、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 4-7 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。

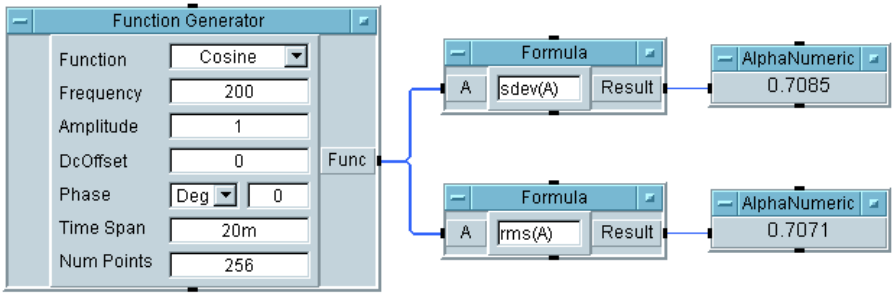


図 4-7. VEE 関数を使った Formula の例

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)
```

メモ

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ができるようにします。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。

メモ

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 4-8 のように表示されます。

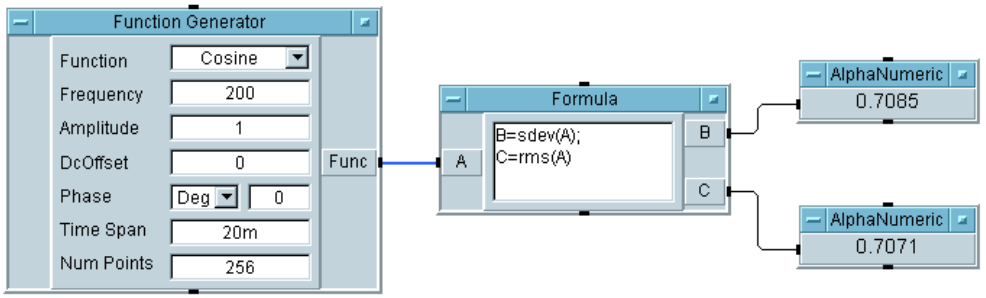


図 4-8. Formula オブジェクトを 1 つだけ使った VEE 関数

独習課題

図 4-9 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、1 つ前の手順で説明したのと同じ課題を実行します。

3. 関数をネストして、1 つ前の手順で説明したの同じ例題を実行します。オブジェクトは 2 つしか使用しません。

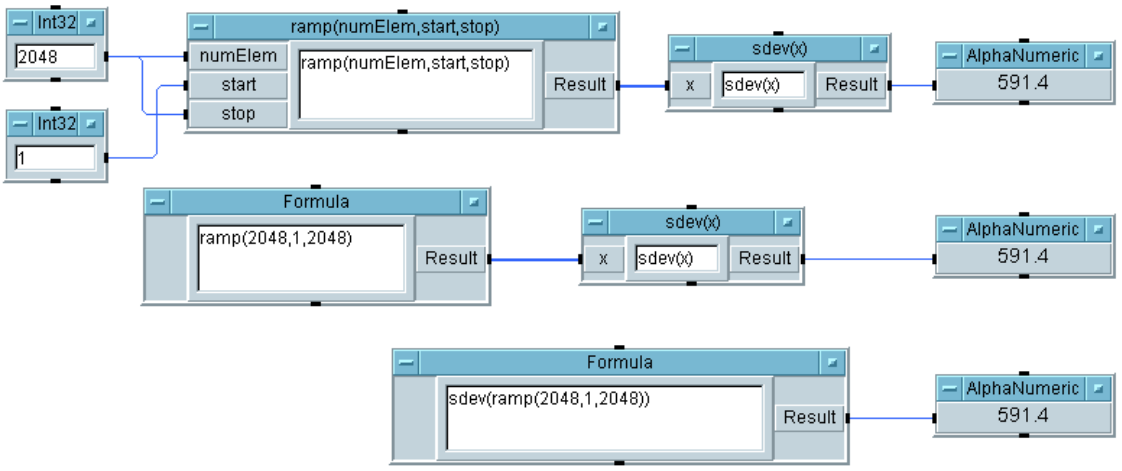


図 4-9. Ramp および SDEV についての独習課題解答

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取ってはじめて、オブジェクトは動作できるからです。

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。

メモ

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 (CD コマンドで) を <*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.
```

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 4-10 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。

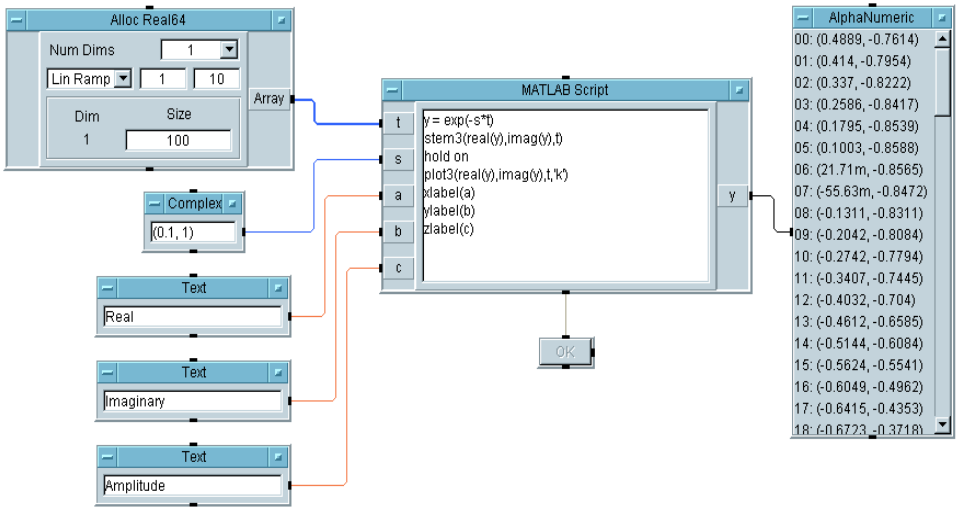


図 4-10. VEE プログラムにおける MATLAB Script オブジェクト

図 4-11 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。

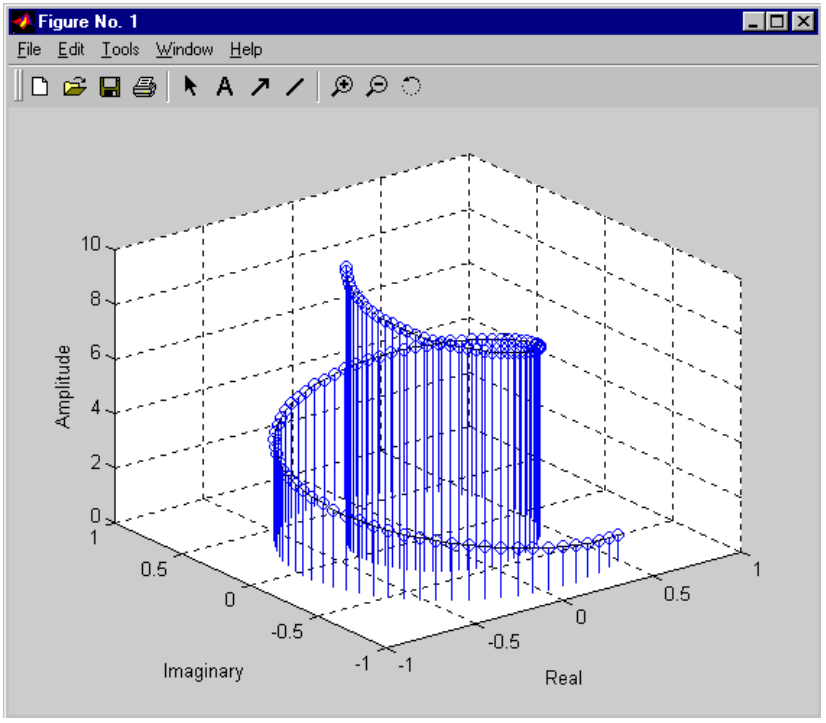


図 4-11. プログラムによって生成されたグラフ

VEE プログラムに MATLAB Script を入れると、VEE は MATLAB Script Engine を呼出して 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、次のセクションで詳述します。

Agilent VEE に MATLAB Script オブジェクトを入れる

VEE 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 とおりあります。

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 4-12 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。

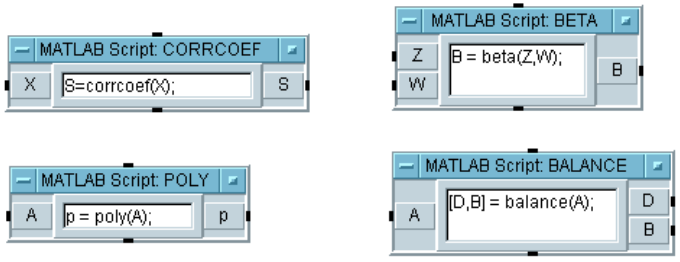


図 4-12. 定義済み MATLAB オブジェクトの VEE プログラムへの追加

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 VEEFormula オブジェクトと区別できます。組込みの VEEFormula 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。

メモ

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、VEE 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。

メモ

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り、別のオブジェクトから入力されるデータがサポートされているデータ型に確実に変換されるようにすることもできます。

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。入力されるデータ型は、Real64、Complex、Waveform、Text のいずれかであるべきところに、代わりに 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 4-13 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。

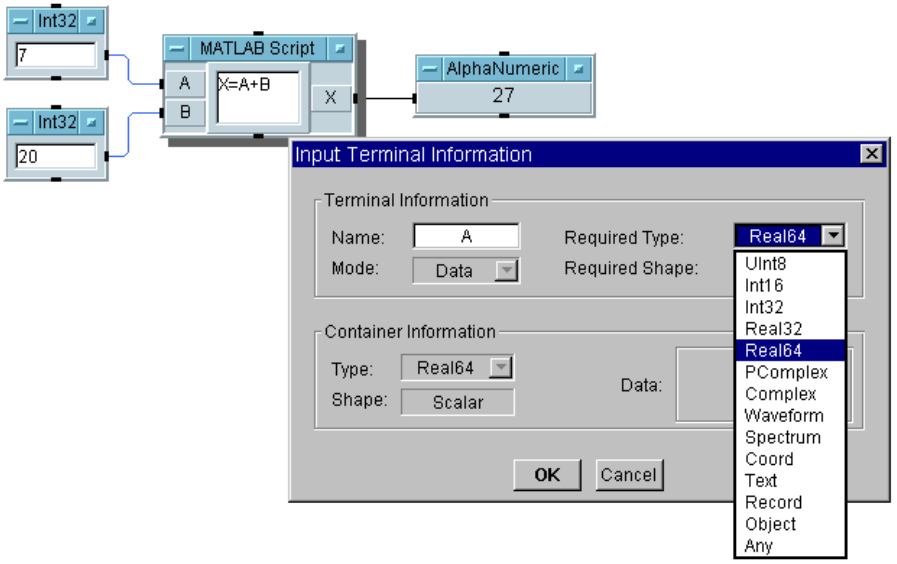


図 4-13. 入力端子のデータ型の変更

テスト・データの表示

表 4-2 では、VEE のオブジェクト別の表示機能を解説します。

表 4-2. 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) および 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、名前から予測できるように数値をグラフィック表示します。これらはすべて、通常は 3 つ、また Meter については 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・メッセージを「LED 表示」して、各レンジでアラームの位置の色を発光させます。
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。

表 4-2. 表示機能 (続き)

表示用オブジェクト	説明
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)、*.ICN (X11 ビットマップ)、*.JPEG、*.PNG、*.WMF(Windows メタ・ファイル) です。
Picture (UNIX)	*.GIF(GIF87a) および *.xwd (X11 Window ダンプ)
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。
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。

表 4-2. 表示機能 (続き)

表示用オブジェクト	説明
XY Trace	スペースの均等な 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 はスペースが均等な 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 4-14 のように表示されます。

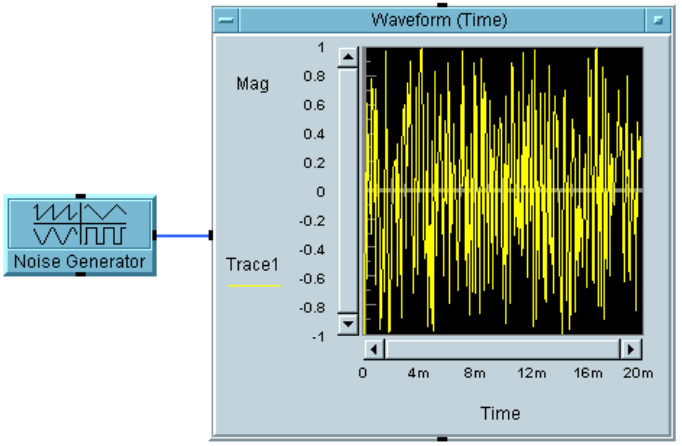


図 4-14. 波形の表示

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。

メモ

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] ⇒「使用法」⇒「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 4-15 のように、新しい座標が画面の一番下に表示されます。

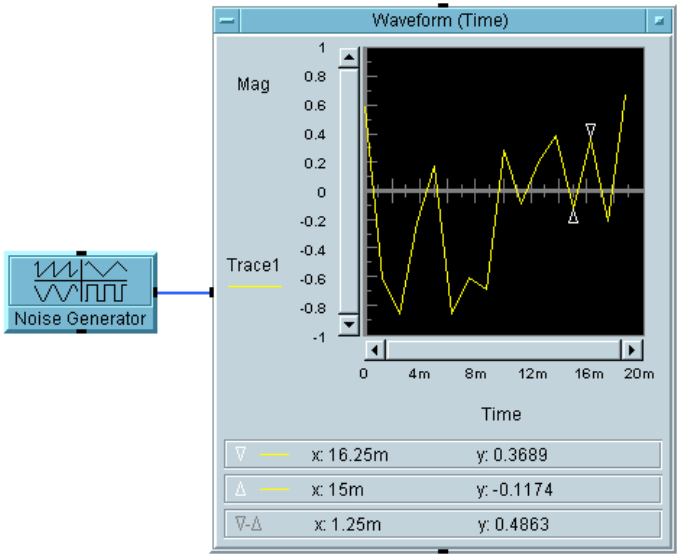


図 4-15. 波形画面上のデルタ・マーカ

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 [Interpolate] 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 [Traces] フォルダのタブをクリックします。

   このフォルダで選択したトレースに関して、色、線種、点のタイプを選択できます。

メモ

また、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、マニュアル『*VEE Pro Advanced Techniques*』を参照してください。

2. 色を選択したら、[OK] をクリックします。[OK] を次にクリックし、[Properties] ボックスを終了します。

これで、トレースは新しい色で表示されます。Panel Layout、Grid Type、Clear Control、Add Right Scale 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。

メモ

また VEE には、表示のオブジェクト・メニューに [Plot] メニューがあり、プログラムの休止をプリント・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、475 ページの付録 A「追加の例題」の例題を行ってください。要点を中心に解説が行われています。

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、必要な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 の主な分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取り方法

テスト結果の保管方法と読取り方法

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

平均的な必要時間 :2 時間

概要

この章では、テスト・データの保管と読取りの基本を習得します。テスト結果を保持するデータ型と適切な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込んだり、ファイルからデータを読取ります。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。

メモ

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 2-3 ページの「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。

メモ

VEE データ型についての概要は、第 4 章「テスト・データの分析と表示」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。

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[0:2] = [4, 5, 6]` 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。
コンマ	サブ配列の構文では、配列の次元を区切るためにコンマを使用します。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、計測器からの読取りを 4 回シミュレートします。読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入れられ、送信した要素の数に基づいて配列のサイズが作成されるため、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。

   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 端子を起動し、Collector で配列を作成して出力します。For Count シーケンス出力ピンを使用すると、Collector XEQ ピンを起動できます。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 シーケンス出力ピンを Collector の XEQ 入力ピンに接続します。

XEQ ピンは、特定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 のデータ出力ピンを 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 5-1 のように表示されます。

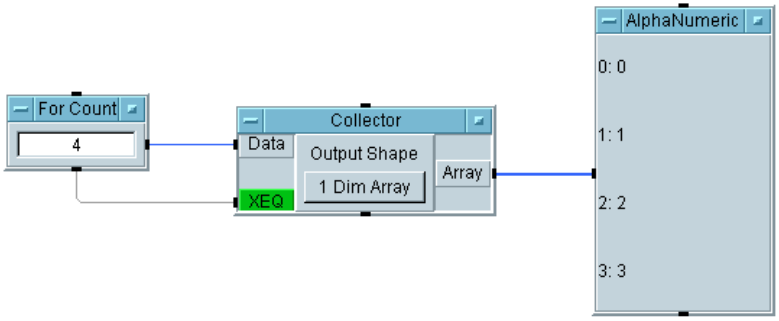


図 5-1. 配列を作成する Collector

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。

   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。

4. 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。

5. プログラムを実行します。図 5-2 のように表示されます。

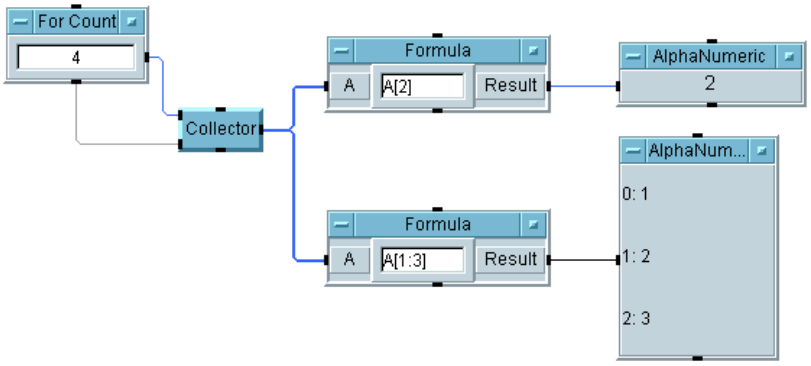


図 5-2. 式を使った配列要素の抽出

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込んだり、ファイルからデータを読取ります。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ても、ファイルごとに 1 つの読取りポインタと 1 つの書込みポインタを保持します。読取りポインタは次に読取るデータを識別し、書込みポインタは次にデータを書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、読取りポインタをファイルの先頭に戻すことができます。

メモ

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 2-3 ページの「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、Rocky Mountain

Basic、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 5-3 の中の `To File` オブジェクトについて説明します。

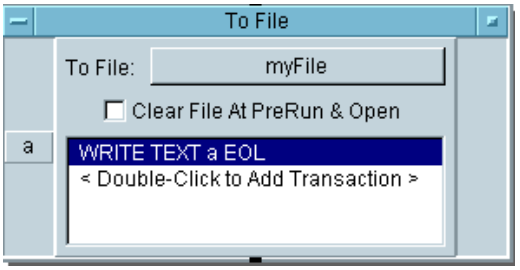


図 5-3. To File オブジェクト

図 5-3 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 5-4 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。

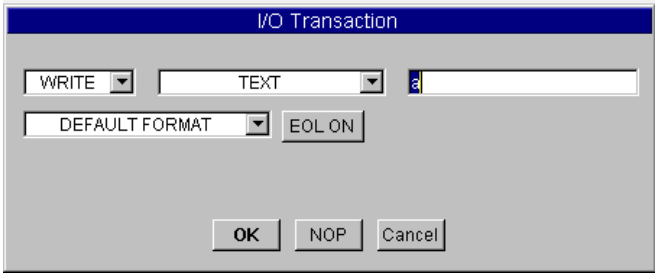


図 5-4. [I/O Transaction] ダイアログ・ボックス

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、actions (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 5-1 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 5-1. 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。

メモ

また、[I/O] ⇒ [Advanced I/O Operations] を選択し、そのメニューでオブジェクトを探して確認できる操作が多数あります。

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、個々の数字について ASCII 文字を使用して、それらの実数を送信します。表 5-2 は、エンコードについて簡単に説明したものです。

表 5-2. 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で、人が読むことのできる形式 (ASCII) ですべてのデータ型を読取ったり、書込みます。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、その文字列を読取り / 書込みます。たとえば、CASE を使用すると、エラー番号を受入れて、エラー・メッセージを書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コ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。読取りトランザクションの場合、式リストは、データを読取るときにその保管場所を示す出力端子名のコ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取りとデータ・フォーマットについては、3-131 ページの第 3 章「簡単な計測器の操作」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。

EOL(文字列の行末シーケンス) はオンまたはオフにすることができます。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取りを行います。すべての VEE データ型で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。

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。必要であれば、図 5-3 と図 5-4 を参照してください。

   WRITE TEXT a 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。

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。

Date & Time	ドロップダウンで・メニュー [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 5-5 のように表示されます。

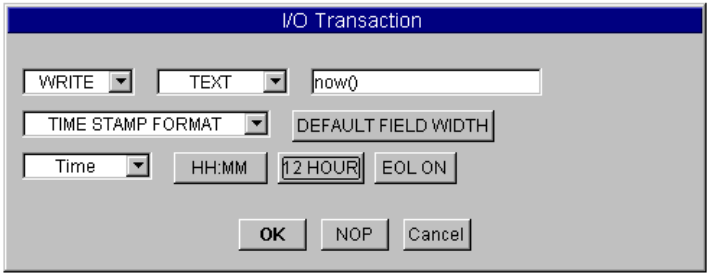


図 5-5. [TIME STAMP I/O] ダイアログ・ボックス

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・ビューに切替えます。For Count のデータ出力を Collector(一番上の入力ピン) のデータ入力に接続します。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。

   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。

3. 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。

   [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。

4. [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。

5. 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 5-6 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。

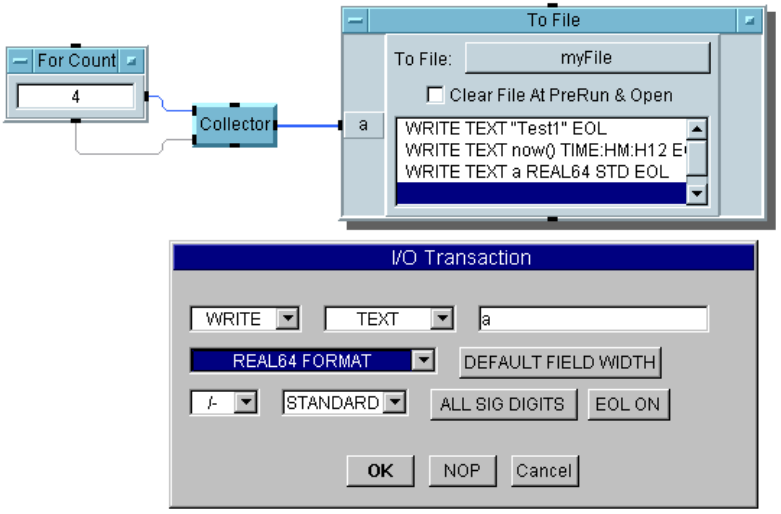


図 5-6. To File オブジェクトを使ったデータの保管

From File オブジェクトを使ったデータの読取り

From File オブジェクトを使ってデータを読取る場合は、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。

メモ

To DataSet または From DataSet を使ってもデータの保管と読取りができます。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、234 ページの「DataSet を使って Record を保管および読取る方法」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 VEE が読取る場合は、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 5-7 のように、[STRING FORMAT] に変更します。

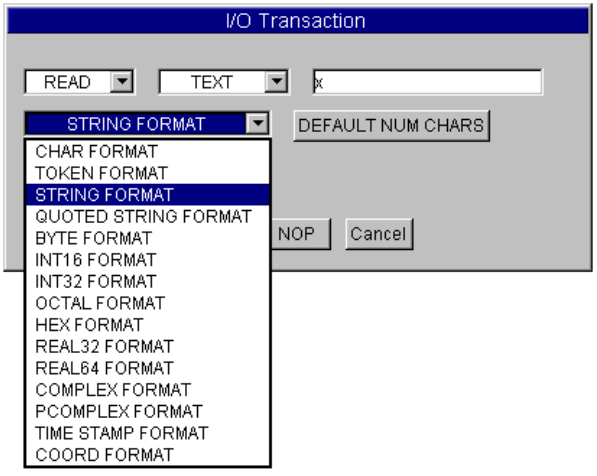


図 5-7. 文字列フォーマットの選択

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 のままにしておいた場合は、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 に入力したデータを上書きします。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。

メモ

タイム・スタンプをテキスト文字列として読取るには、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。

(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。

メモ

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、VEE が正確なサイズを知らなくても、ファイルの末尾までのデータが読取られます。たとえば、この機能を使用すると、文字列配列としてのファイルの全内容を読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 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。

   ヒント : 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 5-8 のように表示されます。

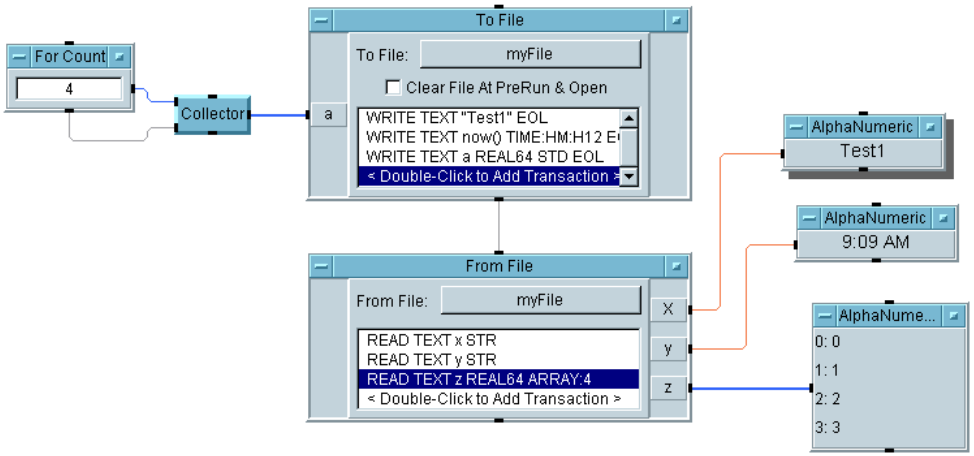


図 5-8. From File オブジェクトを使ったデータの読取り

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、VEE のどのような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。配列の索引は 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。各フィールドは、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取る場合、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。

1. [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。

2. [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。

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

   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。

4. [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、次の値に移るときは [Tab] キーを使用して、「2.2」、「3.3」、「4.4」、「5.5」を入力します。Real64 をアイコン化します。

5. [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

そ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。デフォルトの Scalar を通常使用します。詳細は、マニュアル『*VEE Pro Advanced Techniques*』を参照してください。

6. 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。

7. 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 5-9 のように表示されます。

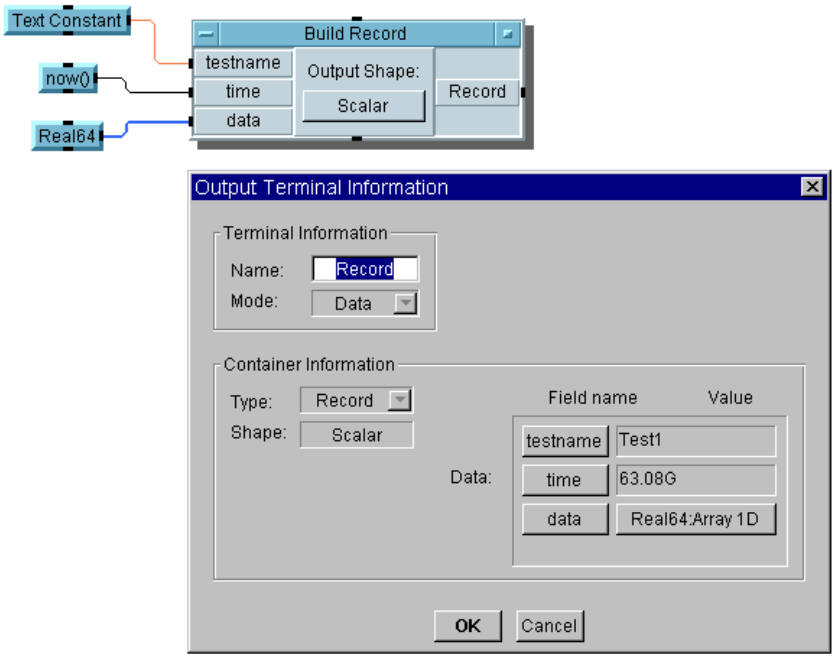


図 5-9. Record に関する出力端子情報

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入れます。Rec.field は、入力フィールドのデフォルトですが、これを編集すると、どのフィールドでも読取れます。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の区別を付ける必要がないので、注意してください。

メモ

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。

   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。

5. 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。
6. [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。

7. 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。

8. [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 5-10 を参照してください。

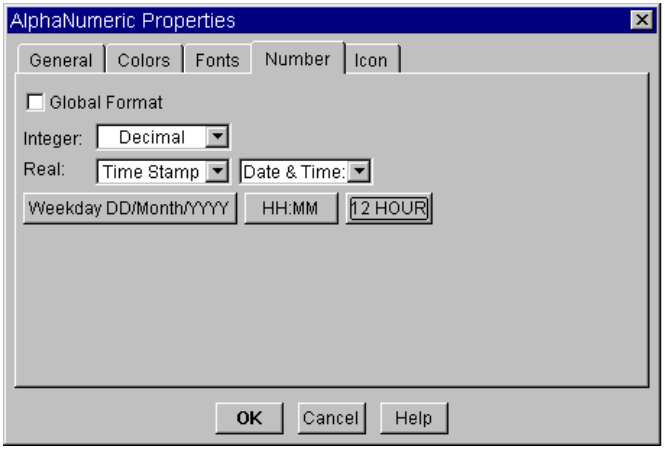


図 5-10. [AlphaNumeric Properties] ボックス

9. 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 5-11 のように表示されます。

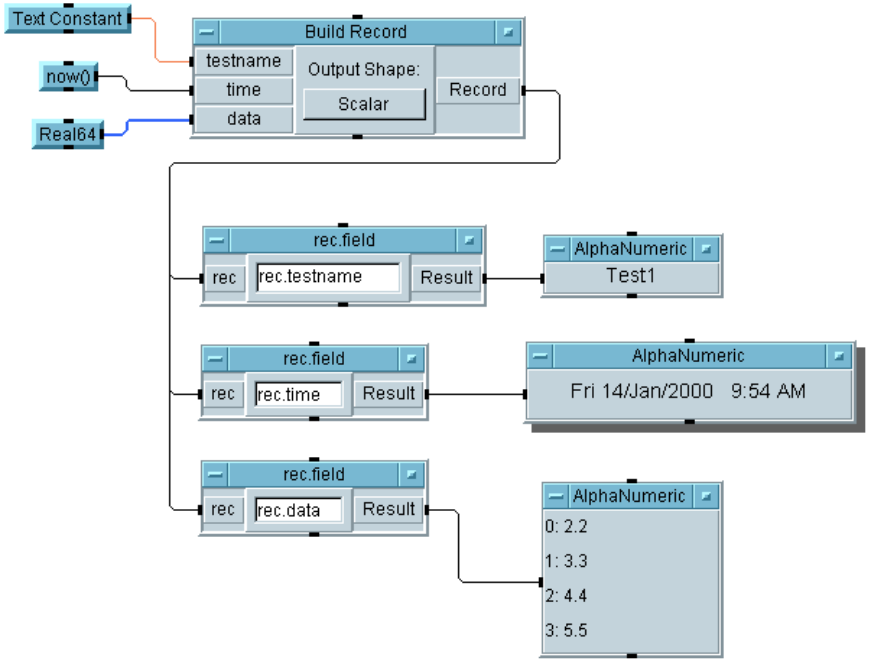


図 5-11. Get Field オブジェクトの使用

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。

メモ

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。

1. [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。

メモ

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。

2. 式 rec.data[*]=b を編集して、データ・フィールドで 4 要素から成る配列の値を変更するようにします。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。

3. [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。最後のエントリに設定した後は、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。

4. [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。

メモ

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。

5. 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。

6. 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 5-12 のように表示されます。

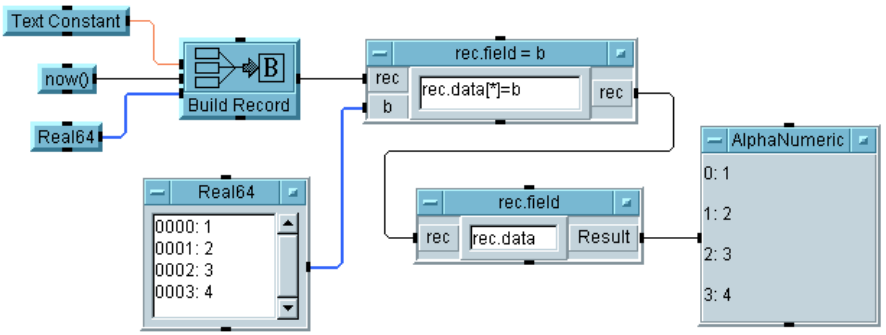


図 5-12. Set Field オブジェクトの使用

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 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。それは図 5-13 のように表示されます。

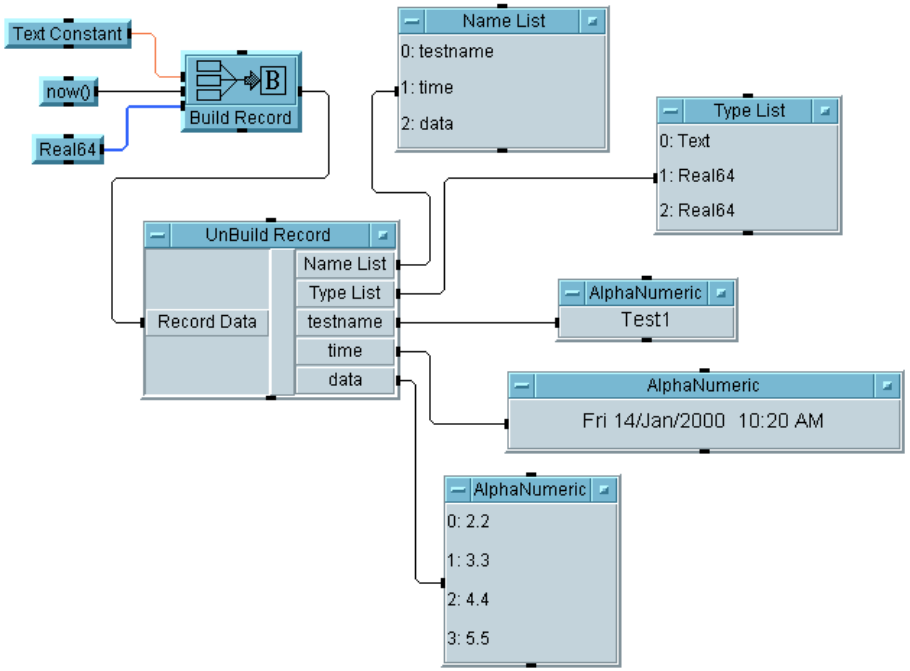


図 5-13. UnBuild Record オブジェクトの使用方法

Type List が testname を Text 型と識別し、time と data を Real64 型と識別するように、Name List ピンがレコード内の 3 つのフィールドの

名前である testname、time、data を与えていることに注目してください。

DataSet を使って Record を保管および読取る方法

DataSet は 1 つ以上のレコードを読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: DataSet 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取る方法について説明します。

DataSet に Record を保管したり読取る方法

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。この値は、10 個の Record の最初のフィールドに書込まれます。

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 5-14 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 に書込まれます。

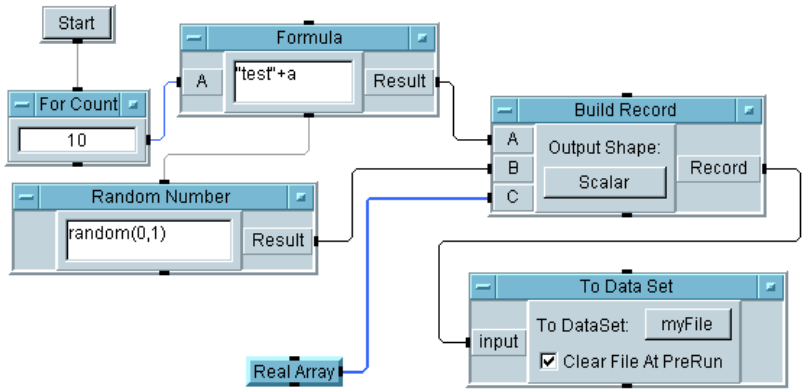


図 5-14. Record から成る配列の DataSet への保管方法

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 Record の 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 を結ぶ波線が表示されます。

メモ

オブジェクトを相互に結ぶ波線は、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 5-15 のように表示されます。

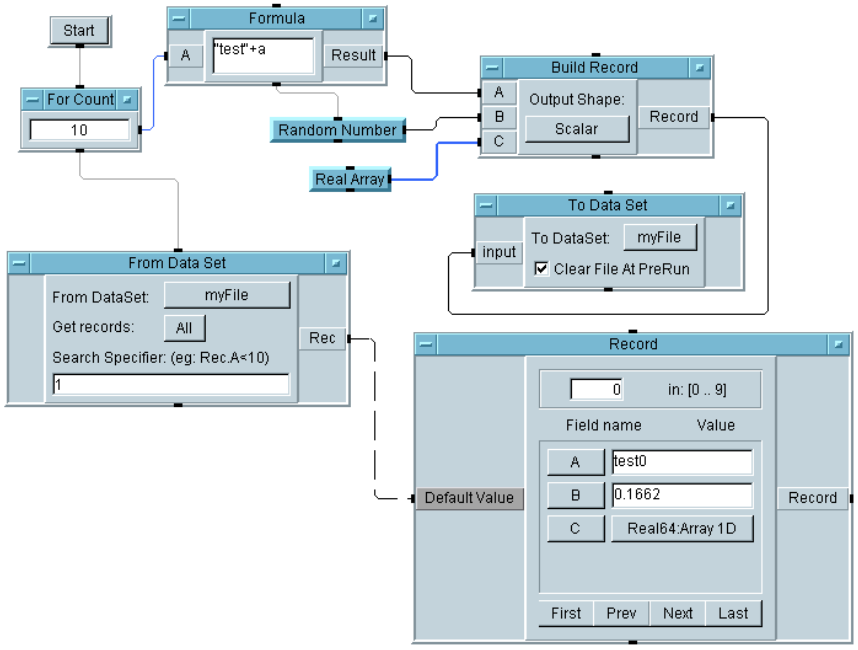


図 5-15. DataSet を使って Record を保管および読取る方法

メモ

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、テスト記述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。情報を検索する場合、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、検索操作を入力するのに使用されます。
- 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 5-16 のように、From Data Set オブジェクトに EOF 出力ピンが追加されます。

メモ

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。

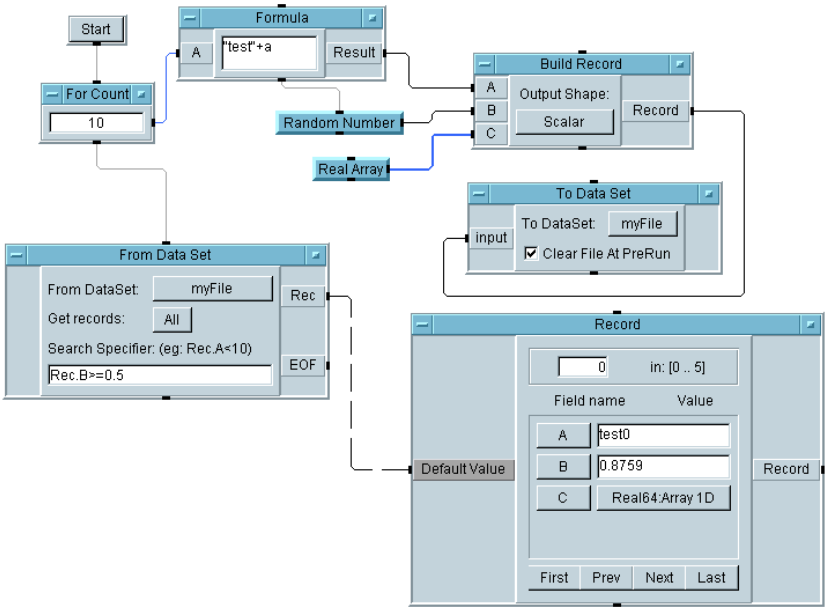


図 5-16. DataSet の検索

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 5-17 のように表示されます。

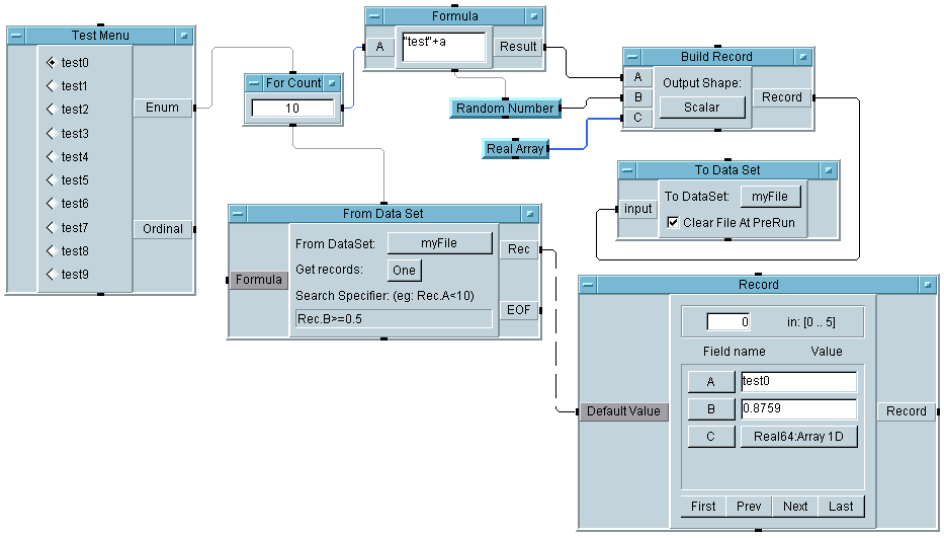


図 5-17. Test Menu オブジェクトの追加方法

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 + "\""
```

“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。
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まれます。 *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。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 5-18 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。

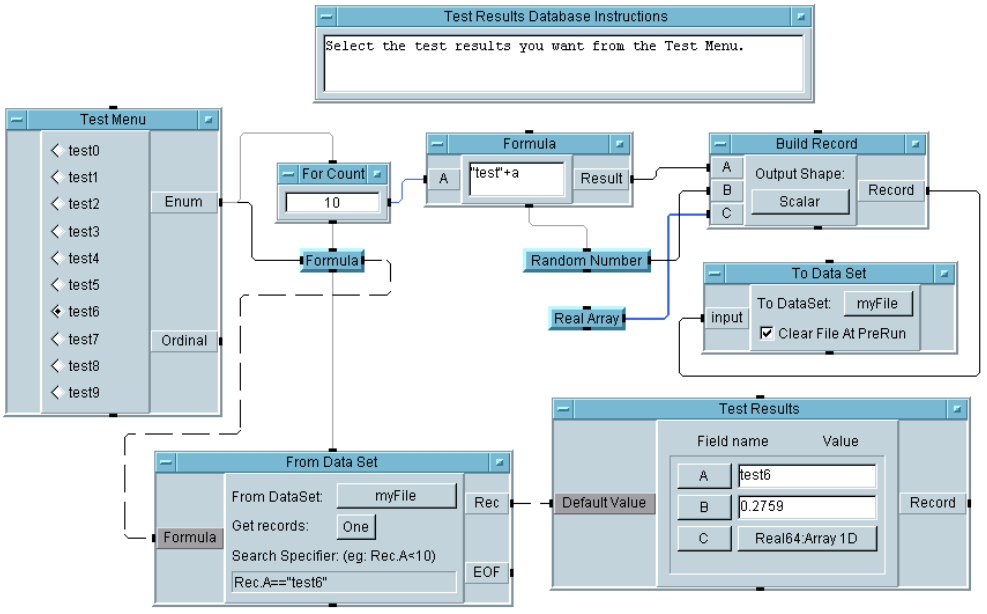


図 5-18. 検索操作へメニューを追加する方法

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、次のオブジェクトをクリックします。それらは、Test Menu、Test Results Database Instructions、Test Results です。

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 5-19 に示します。

メモ

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。

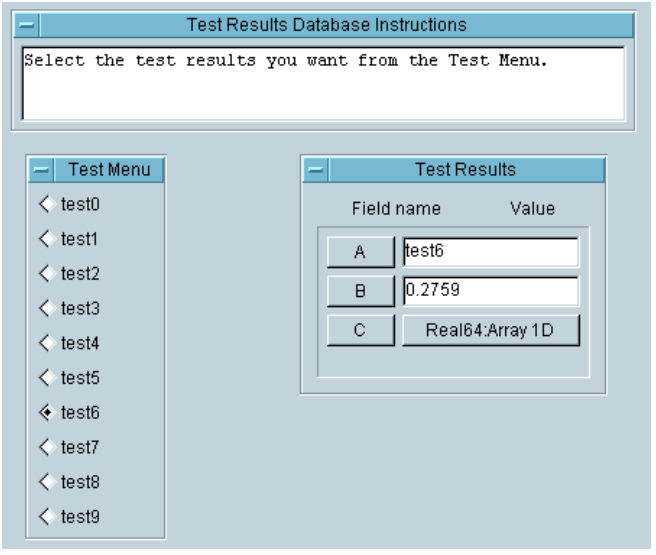


図 5-19. データベースのオペレータ・インタフェース

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。

メモ

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、それらの要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るけるために、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 5-20 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。

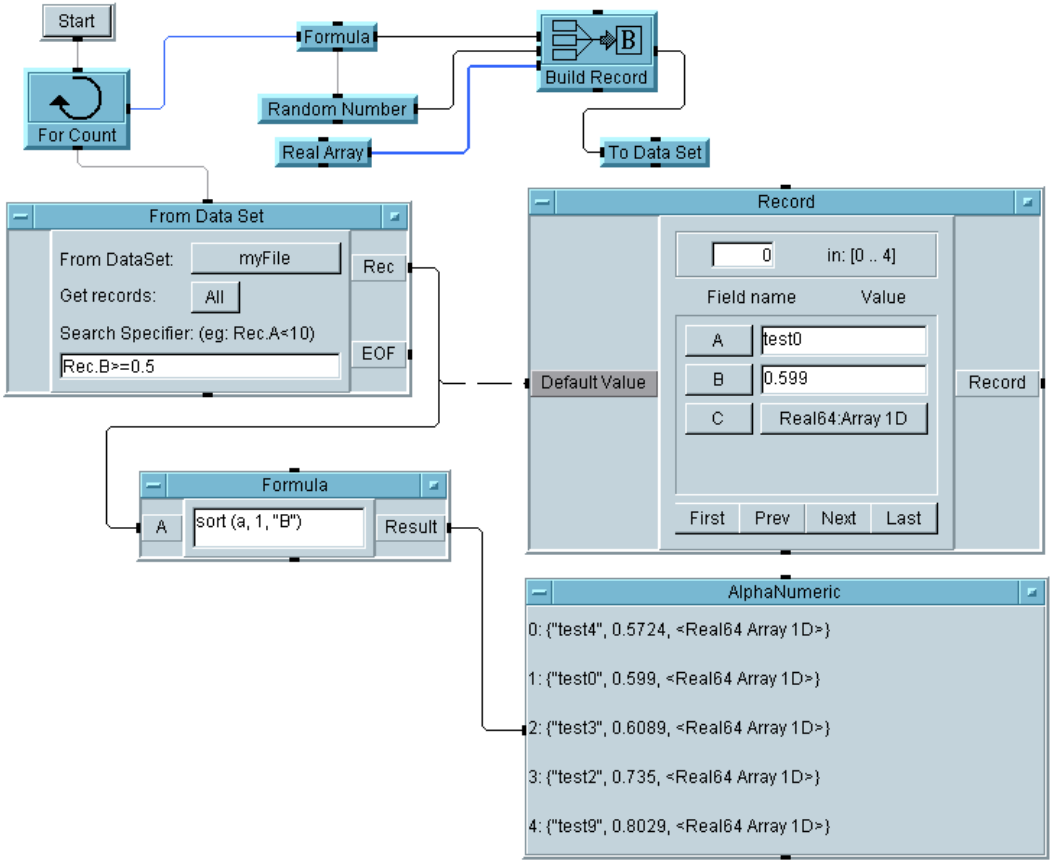


図 5-20. Record のフィールドに対するソート操作

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、必要なら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 での 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

平均的な必要時間 :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 オートメーションを使用します。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が示されます。2 番目の例題では、レポートを生成するための共通テンプレートが示され、基本テンプレートの機能を拡張する方法が解説されます。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。

メモ

VEE では、DDE が ActiveX に置き換わりました。ただし、DDE も引続きサポートされています。既存のアプリケーションから DDE を使用する場合は、『*Visual Programming with HP 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 オートメーションという用語で、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すことにします。この章の例題では、Microsoft の ActiveX 技術の実用的な応用面に焦点を合わせ、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。

メモ

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、404 ページの「ActiveX コントロールの使用方法」と、454 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。

メモ

ActiveX コントロールの参照については、第 10 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 12 章「プラットフォーム固有の事項と Web モニタ管理」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 つすべてが存在しなければならないわけではありません。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 6-1 では、Microsoft Excel 9.0 にチェック・マークが付けられています。

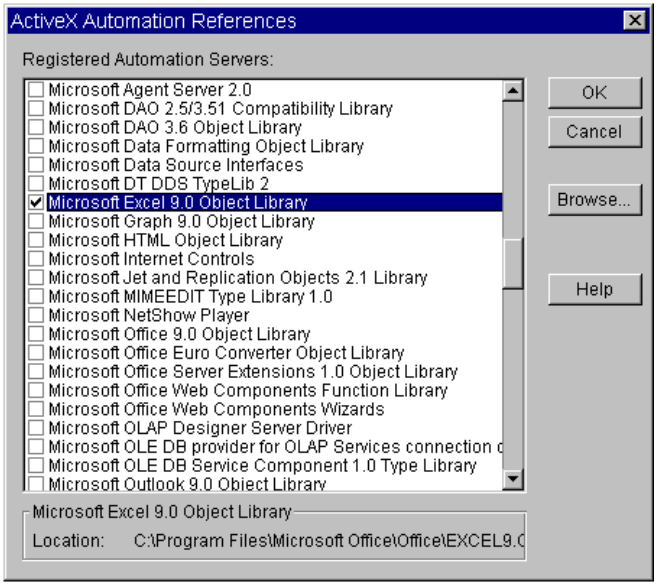


図 6-1. [ActiveX Automation References] ボックス

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 6-2 は、データ型 Object の例です。

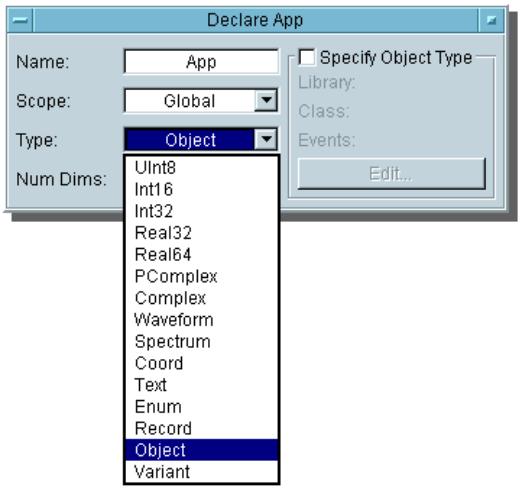


図 6-2. データ型 Object の例

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、VEEFormula オブジェクトに ActiveX のコマンドを入力します。たとえば、図 6-3 は、Set Up Excel Worksheet という名前の付いた VEEFormula 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。

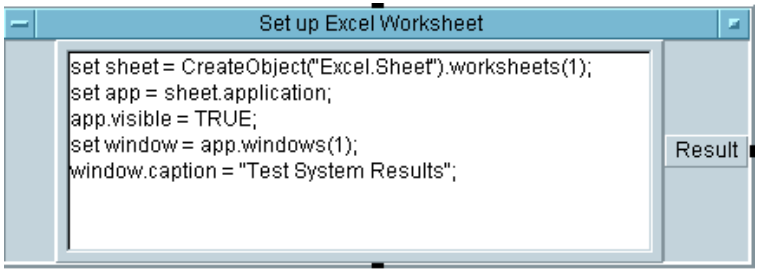


図 6-3. Excel ワークシートをセットアップしてテスト結果を表示するためのコマンド

図 6-3 のようなコマンドやステートメントを作成する場合、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*、または *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、データを受け渡すことができます。構文は、< *オブジェクト* >.< *メソッド* >(*パラメータ*) です。

メモ

データ型 `Object` の構文は、Record のフィールドを取得する VEE 構文 `rec.field` と、`UserFunction` を呼出す 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当てることが重要です。

CreateObject と GetObject の使用方法

図 6-3 で、`Set Up Excel Worksheet` 内のステートメントの 1 つが `CreateObject()` 関数呼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ことを特定の目的として設計されています。

たとえば、`CreateObject("Excel.Sheet")` は、Excel を起動し、Excel のワークブックのリファレンスを返します。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 6-4 に示します。

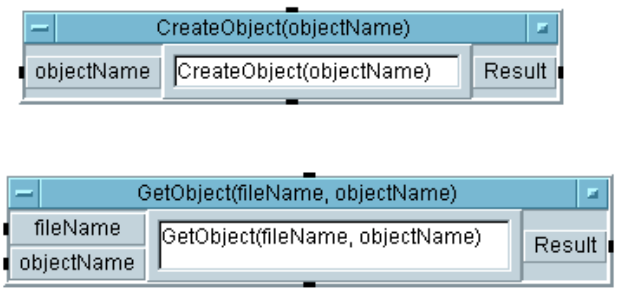


図 6-4. CreateObject と GetObject

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。

メモ

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、299 ページの第 8 章「Agilent VEE 関数の使用方法」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 6-5 のように表示されます。

5. エントリを図 6-5 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。

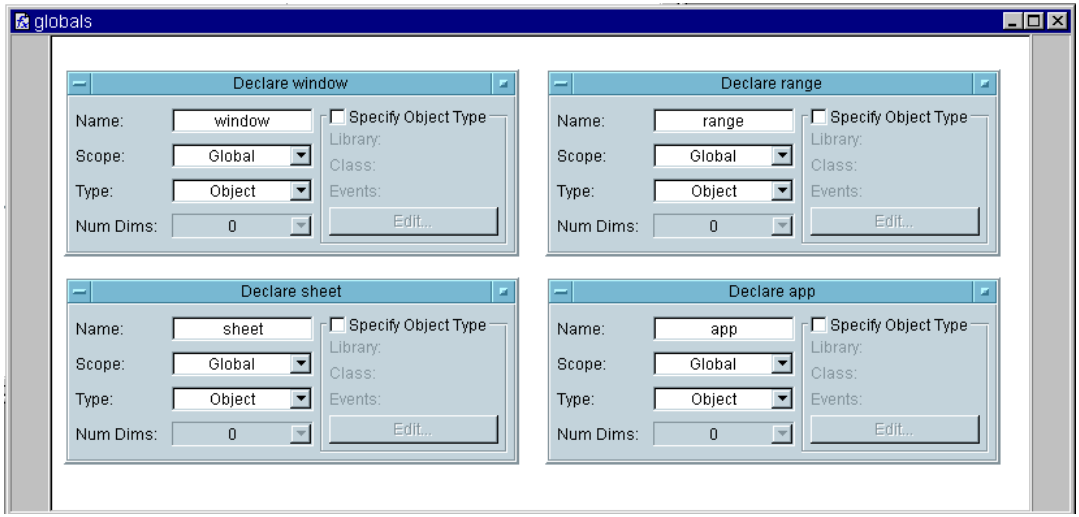


図 6-5. Globals UserFunction

globals UserFunction でデータ型 Object を使用すると、Object Type と Class を指定できることに注目してください。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 の 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 VEE と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、マニュアル『*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 6-6 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。

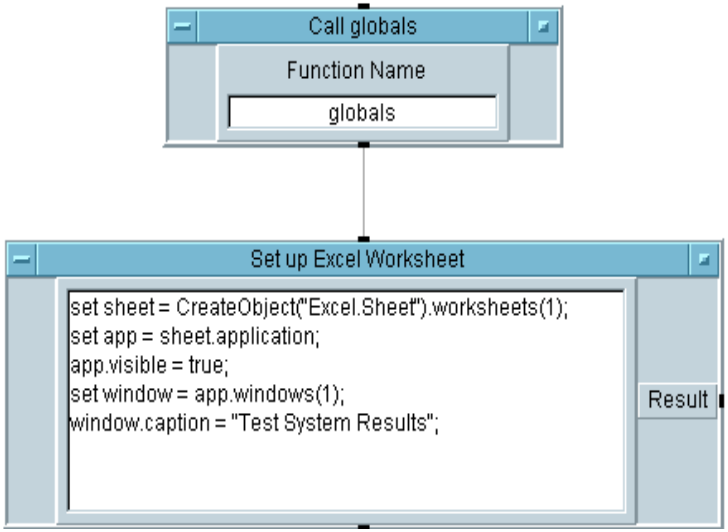


図 6-6. MS Excel ワークシートのセットアップ

`Formula` オブジェクト `Set Up Excel Worksheet` 内のコマンドは次のとおりです。

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* マニュアル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。

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。

メモ

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が多数市販されているので、それら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 6-7 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。

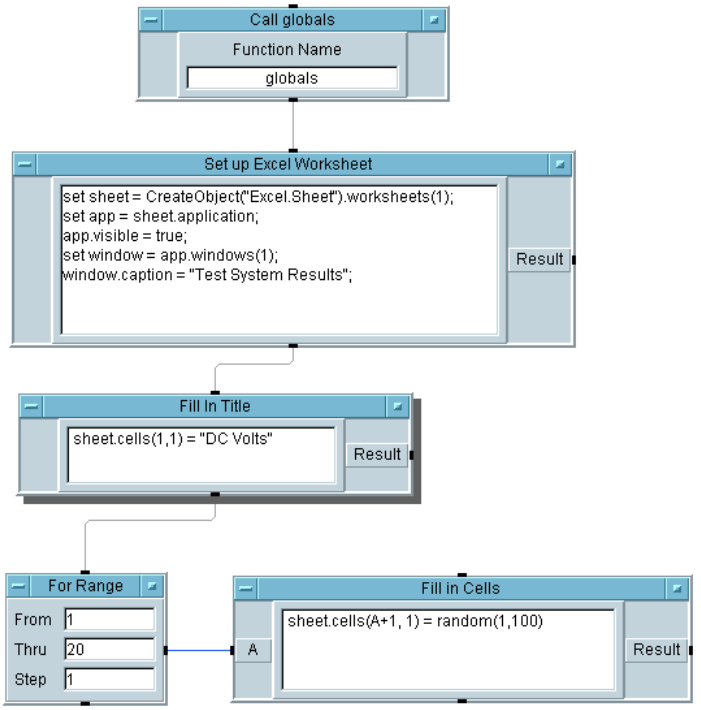


図 6-7. シートへのタイトルとデータの追加

Formula オブジェクト内と For Range オブジェクト内の命庸 1 について、以下で説明します。

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。

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 6-8 のように、名前を変更し、設定し、相互に接続します。

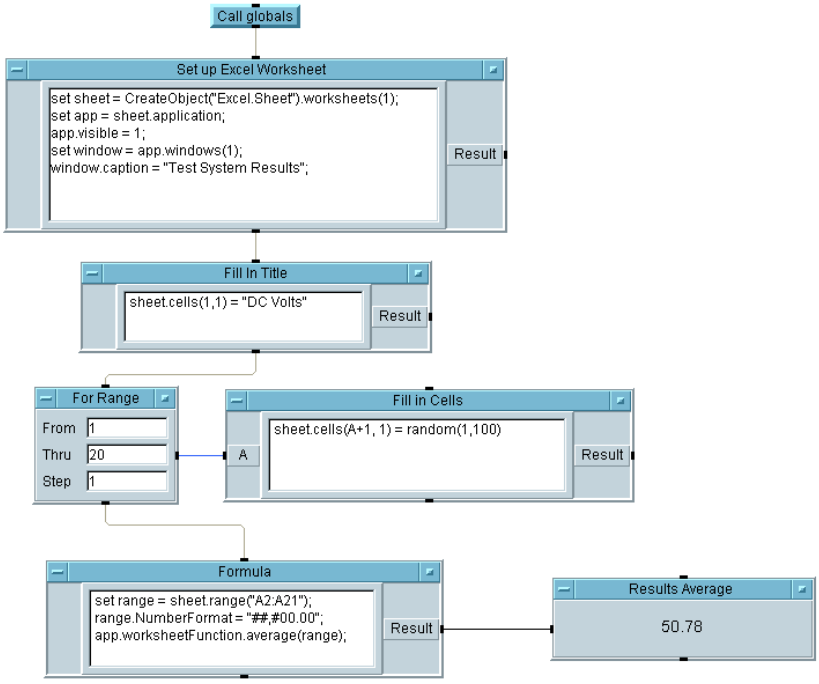


図 6-8. 結果平均プログラム

Formula オブジェクトでの設定項目は次のとおりです。

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。
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 6-9 のようなワークシートを表示します。

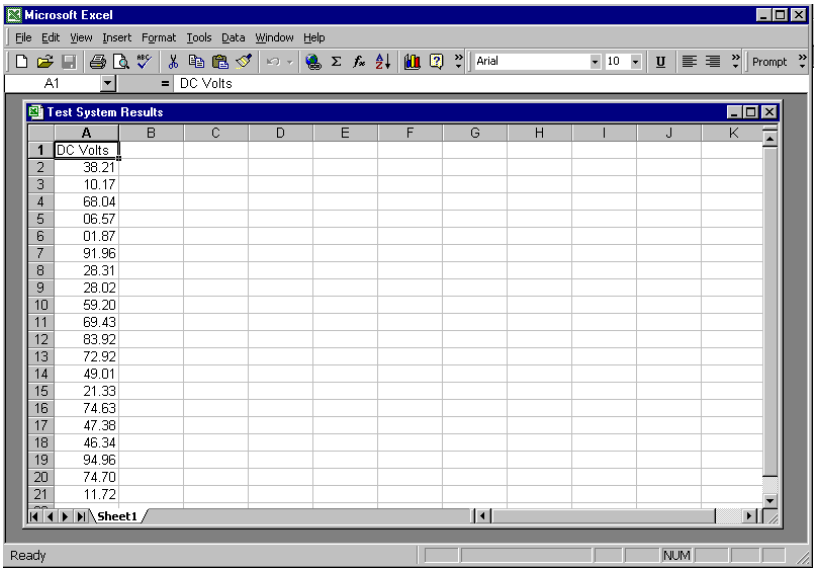


図 6-9. Results Average プログラムの Excel ワークシート

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。

   `[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。
4. 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");
   ```
5. 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 6-10 の Excel ワークシートおよび図 6-11 の完成プログラムと比較します。

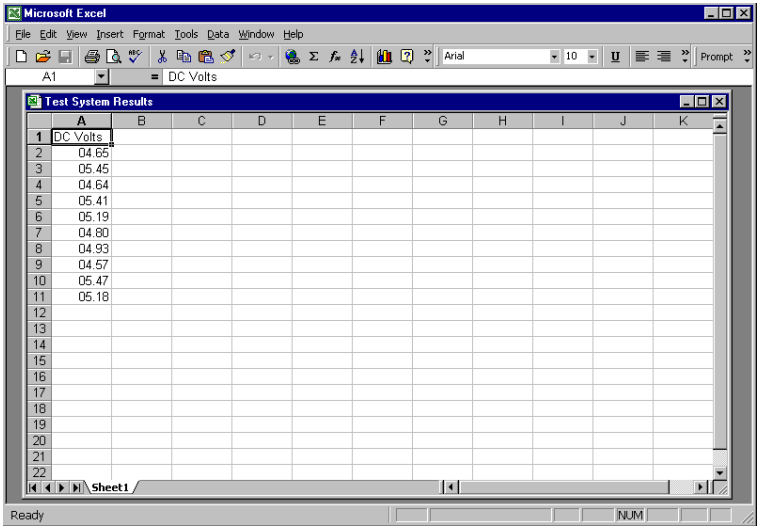


図 6-10. テスト・データから成る配列の場合の Excel ワークシート

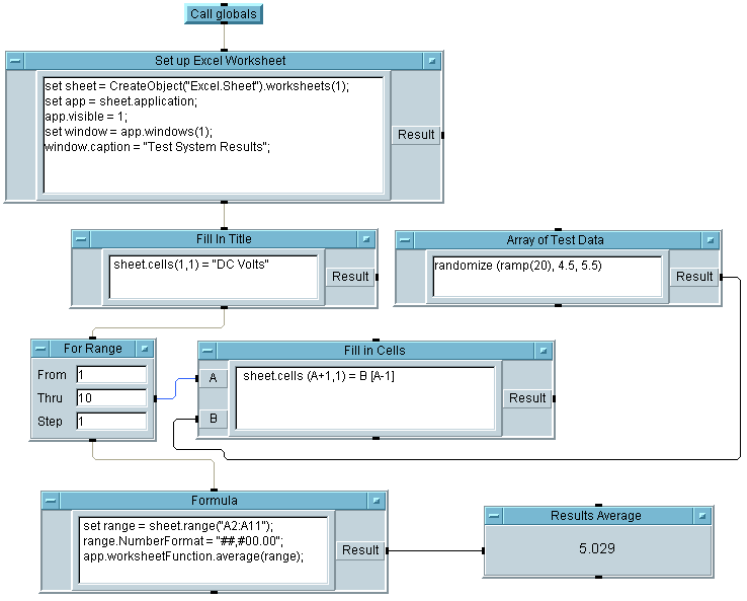


図 6-11. テスト・データから成る配列の場合のプログラム

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。

ヒント : `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 6-12 のように、関数 `transpose()` を呼出し、Excel が 1 ステップでその配列を受入れられるように、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。

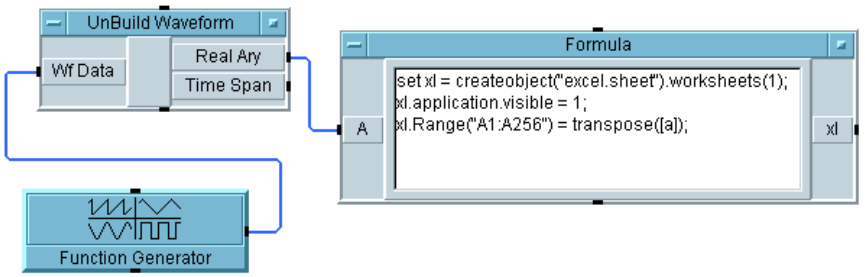


図 6-12. 独習プログラム

MS Excel の機能の拡張

図 6-13 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。

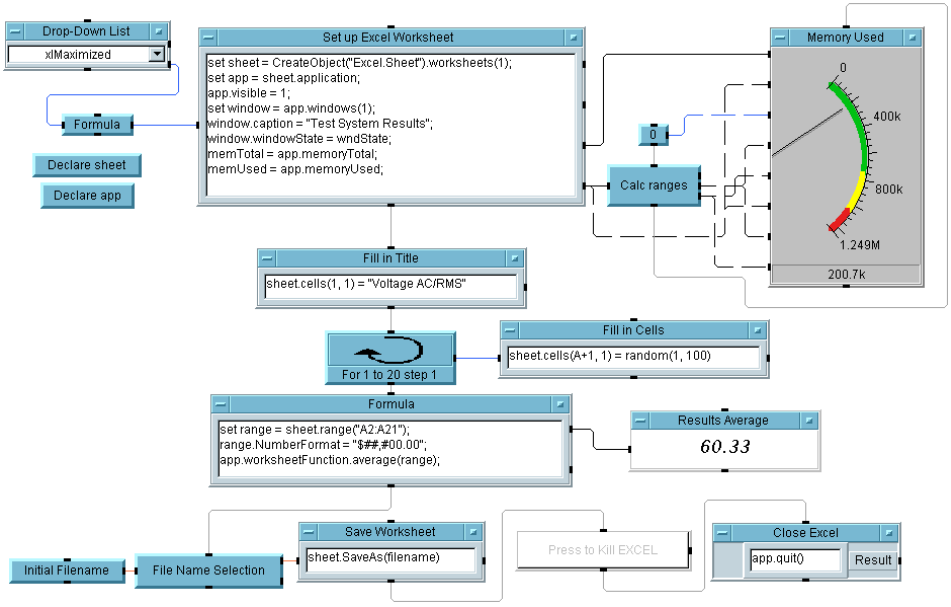


図 6-13. VEE から MS Excel にデータを渡すプログラムの例

図 6-13 内のエントリについて以下で説明します。

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。

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。VEE の変数 memTotal と memUsed に割当てられている Excel ライブラリの memoryTotal プロパティと memoryUsed プロパティに注目してください。次に、2 つの値は、MS Excel によって使用された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を計算するのに使用されます。
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ることに注目してください。選択したファイル名は、次に、Excel SaveAs() メソッド呼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たほかのアプリケーションから MS Word を制御するさらに複雑な方法については、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の利点は次のとおりです。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 6-14 を参照してください。

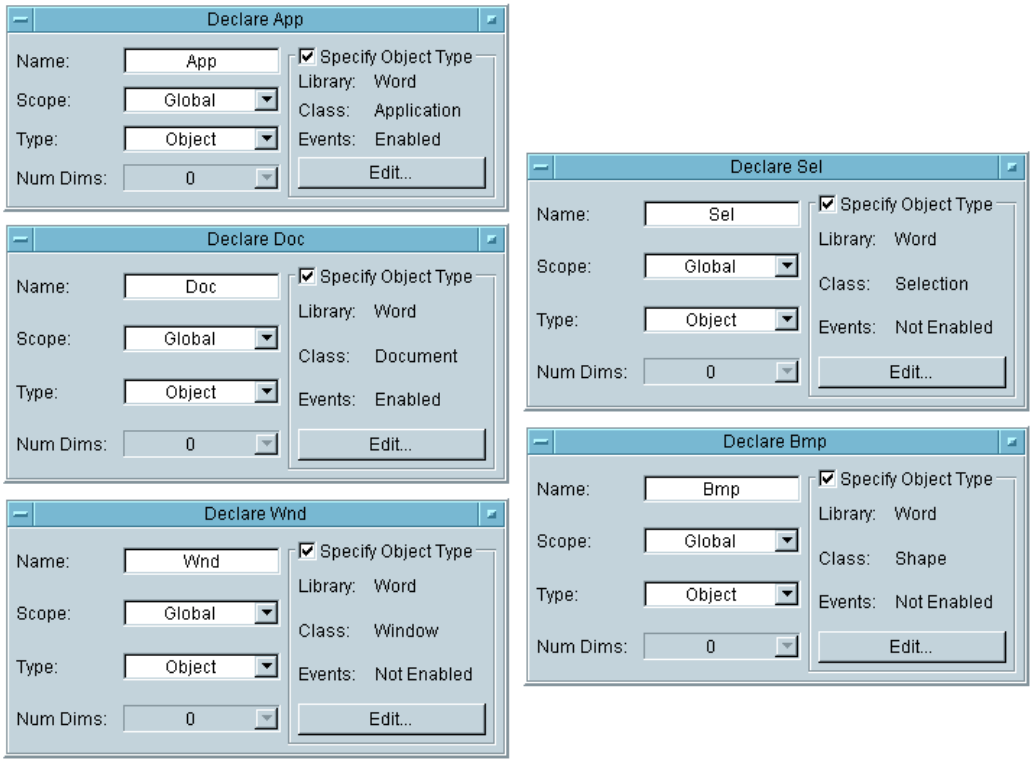


図 6-14. オブジェクト変数

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。

   これにより、ピクセル当たりの色の深さが 256 で 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 ファイルにセーブされます。

6. 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 を割当てます。入力端子 A を削除します。図 6-15 のように、Call Graph、ImageFileName、savePanelImage を接続します。

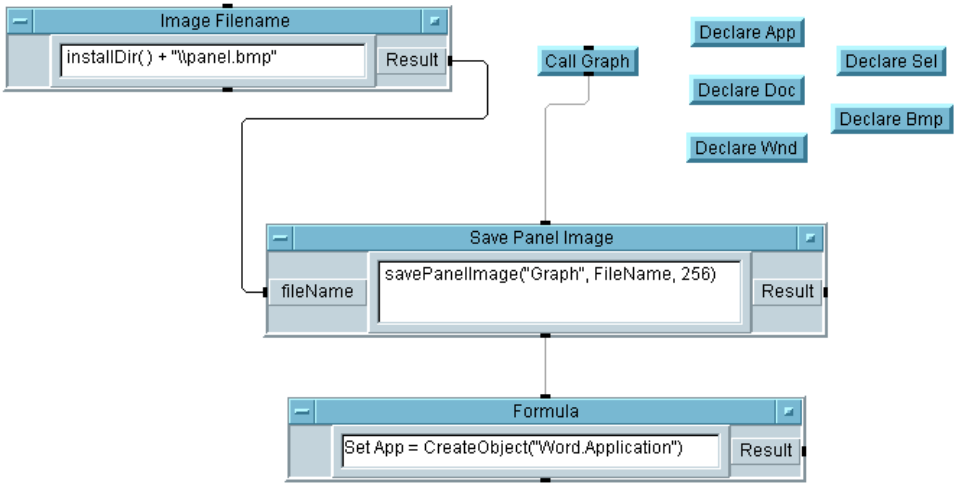


図 6-15. 置換 6-3 プログラムの初期段階

7. `[Device]` ⇒ `[Formula]` をクリックし、図 6-16 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 6-16 のように接続します。

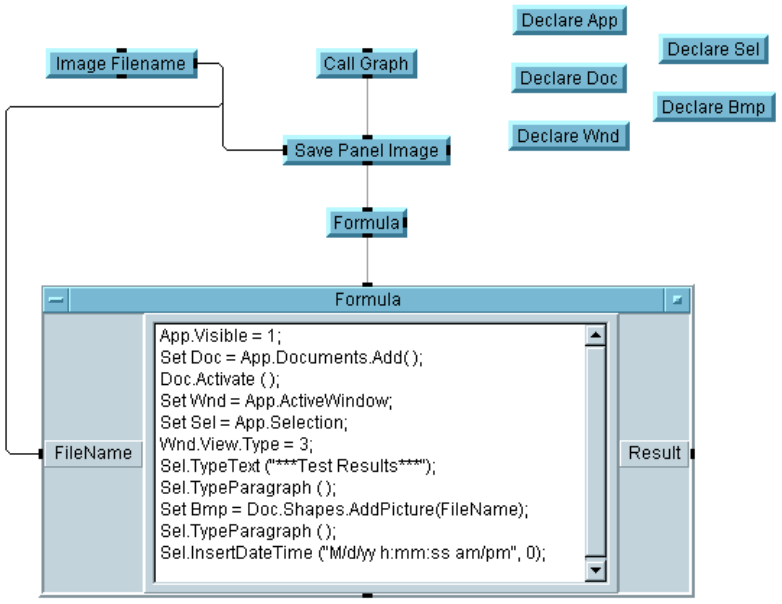


図 6-16. ActiveX ステートメントの追加

図 6-16 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。

App.Visible = 1;	MS Word を画面に表示します。
Set Doc = App.Documents. Add();	MS Word 内に Document を追加し、`Object` 変数 `Doc` に代入します。 メモ : 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。 メモ : 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。

8. 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 6-17 のように設定し、相互に接続します。

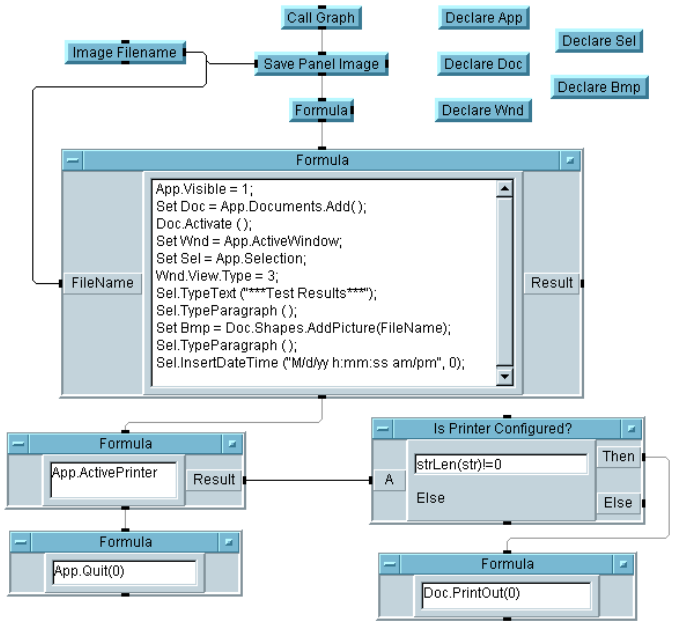


図 6-17. MS Word 内のレポート用の完成プログラム

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。

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` 呼出しを保持する `Formula` オブジェクトを起動します。
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。

9. プログラムを実行します。図 6-18 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。

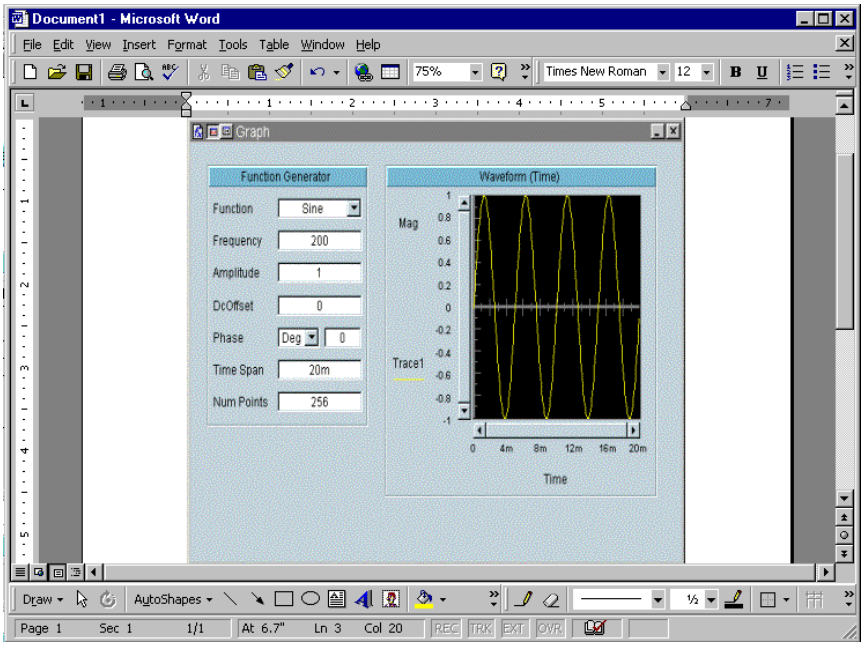


図 6-18. 置換 6-3 で作成した MS Word 文書

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。これらのサーバは、単にオートメーション、OLE オートメーションなどと呼ばれることもあります。

ActiveX コントロールの使用方法についての詳細は、第 10 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 12 章「プラットフォーム固有の事項と Web モニタ管理」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、必要なら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。
- 共通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。

この章の復習

7

異なる言語で書かれた複数のプログラムの統合方法

異なる言語で書かれた複数の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・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

平均的な必要時間 :1 時間

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・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` オブジェクトにプログラム、パラメータ、およびオペレーティング・システムのコマンドを指定できます。`Execute Program` オブジェクトには、PC 用のバージョンと HP-UX 用のバージョンがあります。この章では、PC 用の `Execute Program` オブジェクトを使用する例題と、HP-UX 用の `Execute Program` オブジェクトを使用する例題の両方を紹介します。

Execute Program オブジェクトについて

ほかの言語を使って VEE から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. UNIX オペレーティング・システムの共用ライブラリ、または 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。共用ライブラリについての詳細は、422 ページの「コンパイル済み関数の概要」を参照してください。ダイナミック・リンク・ライブラリについての詳細は、425 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。

3. Rocky Mountain Basic プログラム専用の方法を使用します。詳細は、450 ページの「Rocky Mountain Basic プログラムを使った通信」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 7-1 と図 7-2 に示すように、PC 用のバージョンと HP-UX 用のバージョンがあります。PC バージョンの `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ないことに注意してください。

HP-UX バージョンはトランザクション I/O を使用するため、図 7-2 の例には、入力ピンと出力ピン、およびこのオブジェクトの使用方法を指示するプログラムが含まれています。

Execute Program オブジェクト (PC) の使用方法

図 7-1 は、PC の Execute Program オブジェクトを示します。

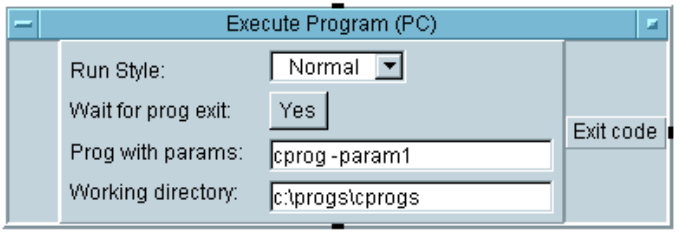


図 7-1. Execute Program オブジェクト (PC)

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出されます。
  http://www.agilent.com/find/vee が URL の一例です。

Execute Program (PC) オブジェクトのフィールドについて、以下で説明します。

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

このフィールドには、DOS プロンプトに入力する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、1 つのハイフンを置いてから、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 95 と 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 と Windows 2000 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・コマンドとして読取るように指示します。

Execute Program オブジェクト (HP-UX) の使用方法

図 7-2 は HP-UX 用の `Execute Program` オブジェクトを示します。

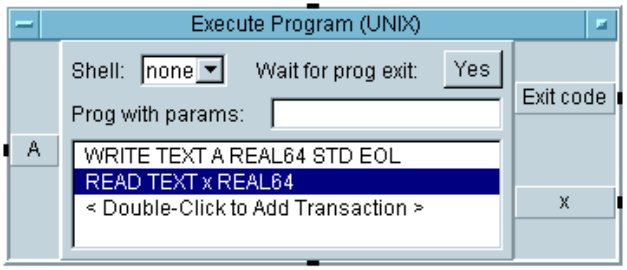


図 7-2. Execute Program オブジェクト (UNIX)

HP-UX は、プロセスと呼ばれる多数のプログラムを同時に実行するように設計されています。VEE が別のプログラムを開始した場合、VEE を*親*プロセスと呼び、開始されたプログラムを*子*プロセスと呼びます。`Execute Program` オブジェクトは、直接に、またはコマンド・シェルを介して、子プロセスを生成します。`Execute Program` (UNIX) オブジェクトのフィールドについて、以下で説明します。

[Shell] フィールド

`[Shell]` フィールドは、選択肢、`none`、`sh`、`csh`、`ksh` を使ってメニューをオープンします。

`[Prog with params]` フィールドの 1 番目のトークンは、実行可能プログラムの名前であると解釈され、その後に続くトークンはパラメータであると想定されます。

`[Prog with params]` フィールドに、標準の入力と出力のリダイレクション (`<<` と `>>`)、ワイルドカード (`*`、`?`、`[a-z]`)、またはパイプ (`|`) などの、シェル依存の機能を入力した場合、シェルを指定する必要があります。そうでない場合は、実行速度を上げるために `none` を選択します。

[Wait for prog exit] フィールド

`[Wait for prog exit]` フィールドは、`Yes` と `No` をトグルします。設定に関係なく、VEE は、子プロセスがまだ 1 つもアクティブでない場合は、子プロセスを生成します。`Execute Program` オブジェクトで指定したすべてのトランザクションは完了します。

- `Yes` に設定した場合、子プロセスは、データ出力ピンが起動される前に終了する必要があります。
- `No` に設定した場合、子プロセスは、データ出力ピンを起動し、アクティブなままです。この設定のほうが、プログラムのパフォーマンスは高くなります。

Prog with params

実行可能プログラム・ファイルの名前とコマンド・ライン・パラメータ

または

シェルに送信されて解釈されるコマンド。

入力端子または出力端子は、`Execute Program` オブジェクトに追加できます。VEE プログラムからのデータは入力ピンで受信されるので、次に、`WRITE TEXT` トランザクションを実行してそのデータを子プロセスに送信します。`READ TEXT` トランザクションは、子プロセスからデータを読取り、それを、VEE プログラムが使用するデータ出力ピンに置きます。

データ入力を追加すると使用可能になる `Command` というラベルの付いたデータ入力端子を追加すると、プログラムやシェル・コマンドの名前を `Execute Program` オブジェクトに送信することもできます。

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 7-1: システム・コマンド (PC) の使用方法

1. [I/O] ⇒ [Execute Program (PC)] 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」

メモ

Windows NT 4.0 または Windows 2000 の場合は、command.com を cmd に置換えます。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 7-3 のように表示されます。

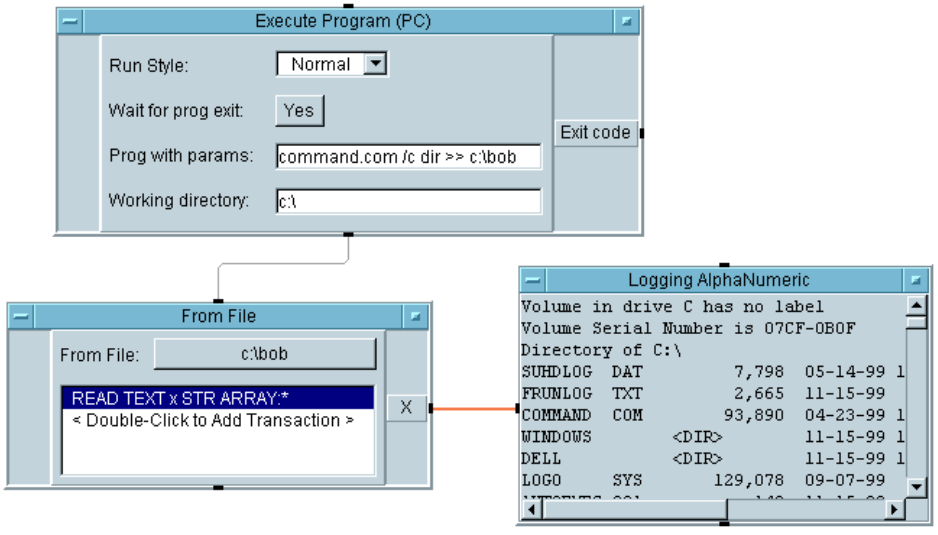


図 7-3. ディレクトリ内のファイルの一覧表示方法 (PC)

例題 7-2: ディレクトリ内のファイルの一覧表示方法 (UNIX)

この例題では、オペレーティング・システム・コマンドの ls を使用します。このコマンドは、ディレクトリ内のファイル名を一覧表示します。このコマンドはシェル依存コマンドではないので、Shell を none に設定できます。次に、シェル依存の機能であるパイプ (|) を使用して、この例題のバリエーションをプログラミングします。

HP-UX プログラムでファイルを一覧表示するには、ls コマンドを次のように使用します。

1. [I/O] ⇒ [Execute Program (UNIX)] を選択し、上部左の作業領域内に配置します。

2. [Shell] フィールドが [none] に設定され、[Wait for prog exit] フィールドが [Yes] に設定されていることを確認します。そのように設定されていれば、VEE がプログラムを継続する前に、ls コマンドは必ず終了します。

3. [Prog with params] フィールドをクリックし、「ls /tmp」と入力します。任意のディレクトリを指定できます。

4. データ出力端子を追加します。デフォルトの端子には、X という名前が付けられます。このプログラムには終了コードはないので、Exit code というラベルの付いた端子は無視してください。

5. 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。プログラムのデータはその出力端子に読込まれるので、デフォルトの変数を X に編集します。

   a. READ TEXT トランザクションを指定するため、WRITE を READ に変更します。

   b. [REAL64 FORMAT] を [STRING FORMAT] に変更します。

   c. 1 次元の配列を指定するため、データの型を SCALAR から ARRAY 1D に変更します。

   d. ディレクトリ内のファイル数はわからないので、[SIZE:] ボタンを [TO END: (*)] にトグルします。[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。

6. [Display] ⇒ [Logging AlphaNumeric] を選択し、データ入力ピンを Execute Program オブジェクトの X 端子に接続します。

7. プログラムを実行します。図 7-4 のように表示されます。

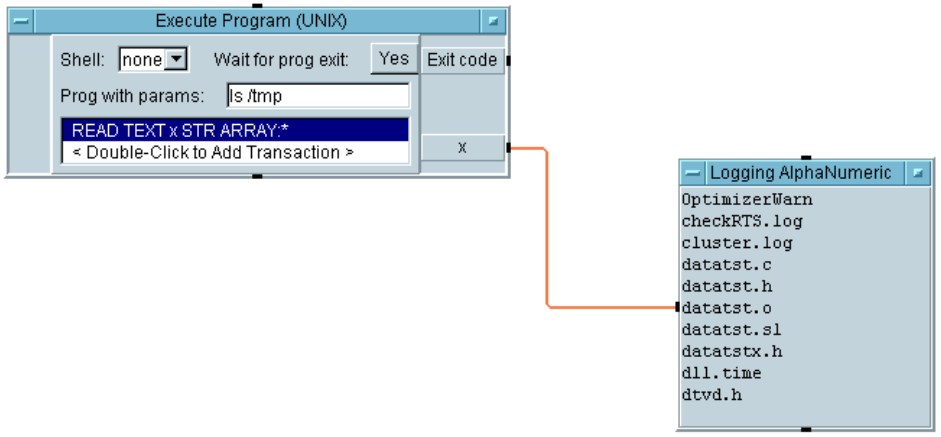


図 7-4. ディレクトリ内のファイルの一覧表示方法 (UNIX)

シェルを使ったディレクトリ内ファイルの一覧表示方法

最後の例題のこのバリエーションでは、シェル依存の機能であるパイプ (|) を使用して、1 つのオペレーティング・システム・コマンドの出力をもう 1 つのオペレーティング・システム・コマンドに送信します。受信するほうのコマンドは wc で、これは word count の略です。wc コマンドは、名前の付いているファイル内の行、語、文字をカウントします。コマンド、wc -l filename は、指定されたファイル内の行数をカウントします。この例では、ディレクトリ内のファイル数をカウントし、次にその数とファイル名を表示する方法について説明します。

1. [I/O] ⇒ [Execute Program (UNIX)] を選択します。

a. シェル機能の | と ; を使用するため、[Shell] フィールドを [sh] に設定します。

b. [Shell command] フィールドで、コマンド、「ls /tmp|wc -l;ls /tmp」を入力します。

c. 2 つのデータ出力端子を追加します。1 つは X というラベルの付いた端子、もう 1 つは Lines というラベルの付いた端子です。

d. 1 番目のトランザクションを READ TEXT Lines INT32 と設定します。デフォルト変数 a が Lines に変わります。

e. 2 番目のトランザクションを READ TEXT X STR ARRAY:Lines と設定します。[SIZE] フィールドに「Lines」と入力し、配列の長さを指定します。

2. [Display] ⇒ [AlphaNumeric] を選択し、クローンを作成します。一方の AlphaNumeric を Lines 出力に、もう一方の AlphaNumeric を X 出力ピンに接続します。

3. プログラムを実行します。図 7-5 のように表示されます。

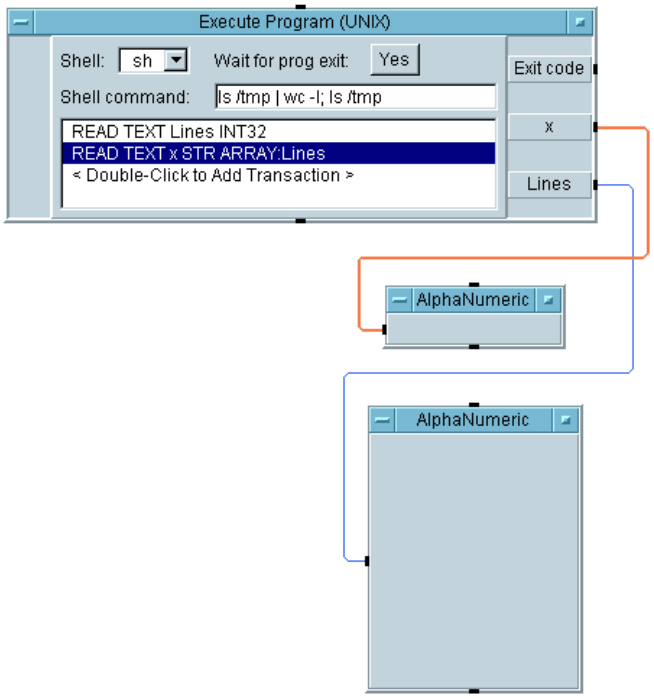


図 7-5. パイプを使ったシェル・コマンドの使用方法

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を統合することを計画している場合は、それらのプログラムをプラットフォーム間で容易に移植できるように VEE プログラムを作成します。図 7-6 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。

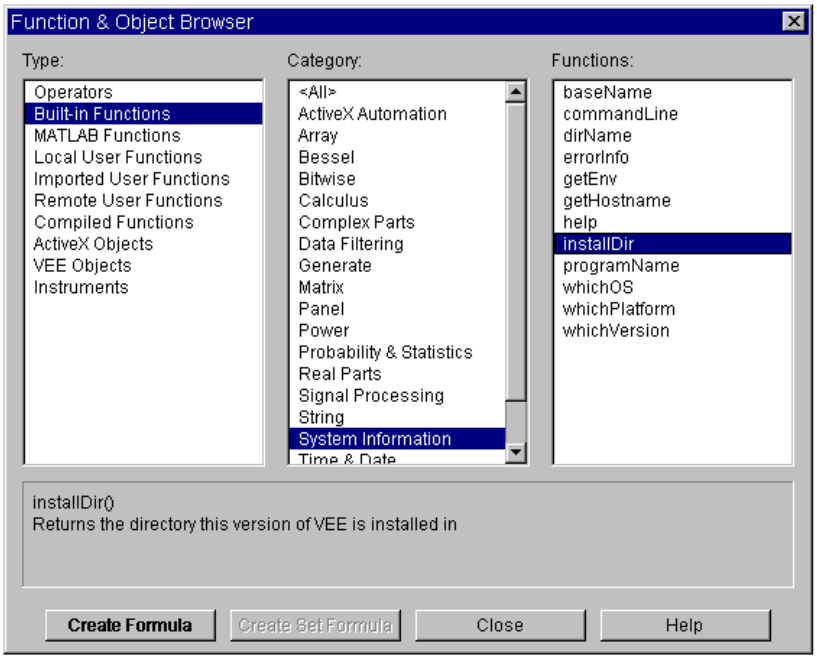


図 7-6. システム情報関数

`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。

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_95`、`Windows_98`、`Windows_2000`、`Windows_NT`、`HP-UX`。 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、必ず正しい種類のライブラリがインポートされることを目的として `whichOS()` を使用するプログラムを見たい場合は、`examples\manual` ディレクトリの `manual49.vee` を見てください。`whichOS()` は、HP-UX では、共用ライブラリをインポートし、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。
whichplatform	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の場合に役立ちます。

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、必要なら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全般的なプロセスについて説明できる。
- HP-UX プラットフォームの相違について説明できる。
- VEE からオペレーティング・システム・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。

この章の復習

8

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
- 関数を作成、呼出し、編集する方法
- 関数ライブラリを作成、マージ、インポート、削除する方法
- 大規模プログラム内の関数を検索する方法
- 既存の VEE プログラムをテストとマージする方法

平均的な必要時間 :1 時間

概要

この章では、VEE UserFunction、コンパイル済み関数、リモート関数を習得します。関数は、テストの開発に要する時間を大幅に減らすの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、既存の作業の効率を高め、プログラムのコードのサイズを減らし、テスト・プログラムの保守を容易にすることができます。関数は、グループにして使用したりライブラリに置いて使用できます。グループやライブラリを作成すると、それらを新しいプログラムにマージできます。関数は、複数のプログラム間や複数の開発者間で共用できます。

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、呼出し、編集の方法について説明します。また、関数ライブラリの作成方法も習得します。関数ライブラリは、開発フェーズでプログラムにマージしたり、実行時にプログラムにインポートできます。

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。
   - ❑ あるプログラムから別のプログラムに `UserFunction` を転送するには、プログラムの開発時に UserFunction をマージするか、実行時に UserFunction をインポートします ([Device] ⇒ [Import Library])。

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 11 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

   - リモート関数は、自分のネットワークに接続しているリモート・ホスト・コンピュータで実行される点を除き、`UserFunctions` に似ています。

UserObject と UserFunction の違い

`UserObjects` 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のコピーのみが存在します。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出されます。UserFunction に加えた変更は、その UserFunction を呼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。コードの再利用が多い場合は、UserFunction のライブラリを作成することもできます。

例題 8-1: 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 8-1 を参照してください。

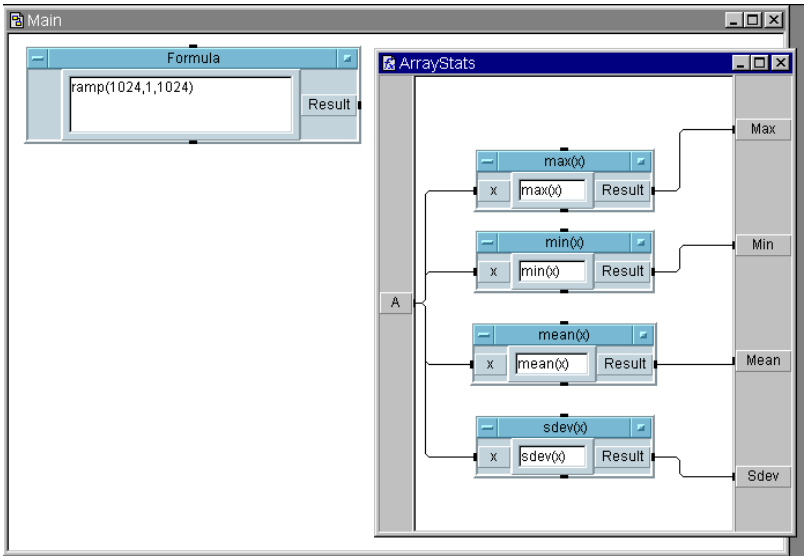


図 8-1. メイン・ウィンドウと ArrayStats ウィンドウ

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 8-2 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。

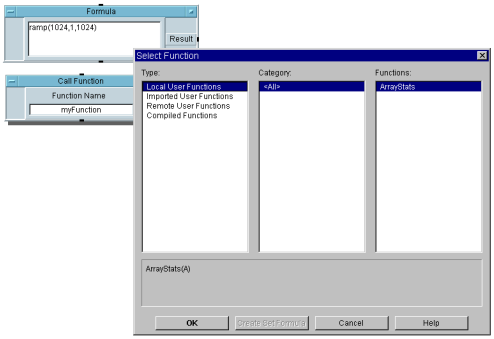


図 8-2. Call myFunction のためのピンの設定

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 8-3 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。

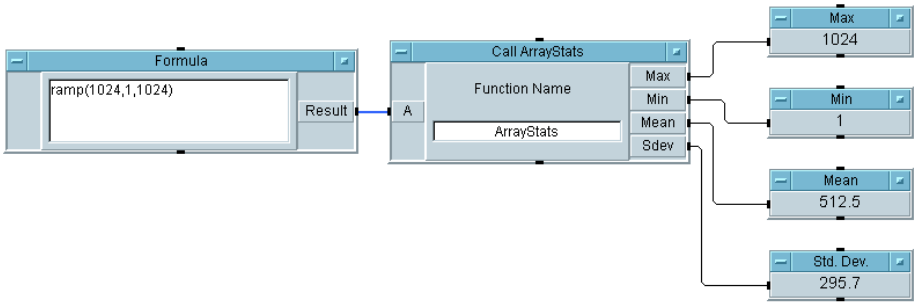


図 8-3. ユーザ関数 ArrayStats を呼出す方法

`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。

ショートカット: 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 8-4 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。

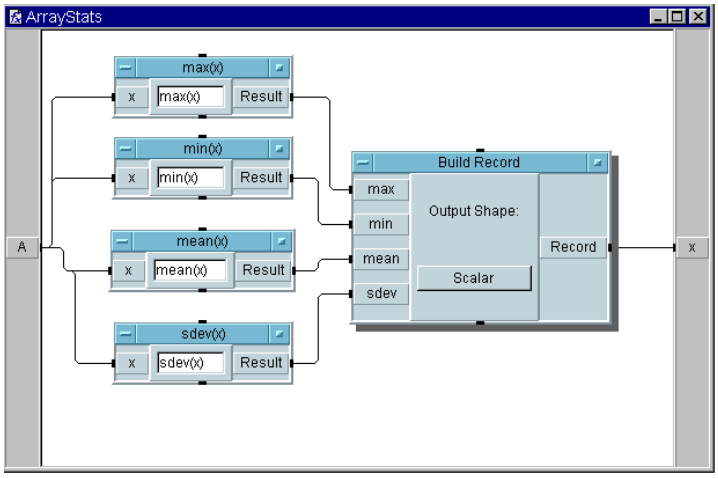


図 8-4. UserFunction ArrayStats の編集方法

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。

メモ

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、ある UserFunction を呼出す Call オブジェクトと式をすべて検索できます。詳細は、327 ページの「大規模プログラム内の関数を検索する方法」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、Call Function オブジェクトの右に配置します。

   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 8-5 のように表示されます。

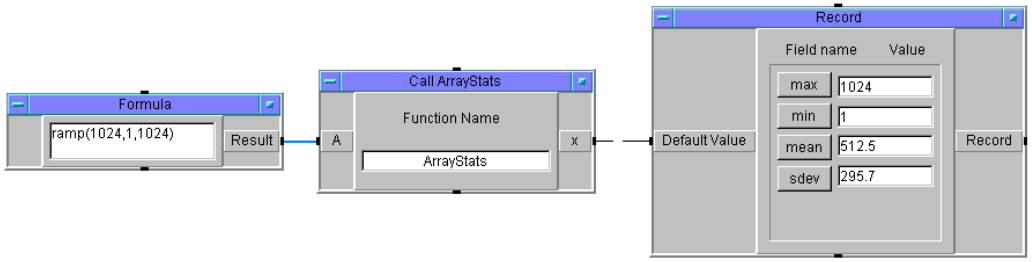


図 8-5. ArrayStats の出力を Record に編集した後

式から UserFunction を呼出す方法

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 8-6 のように表示されます。

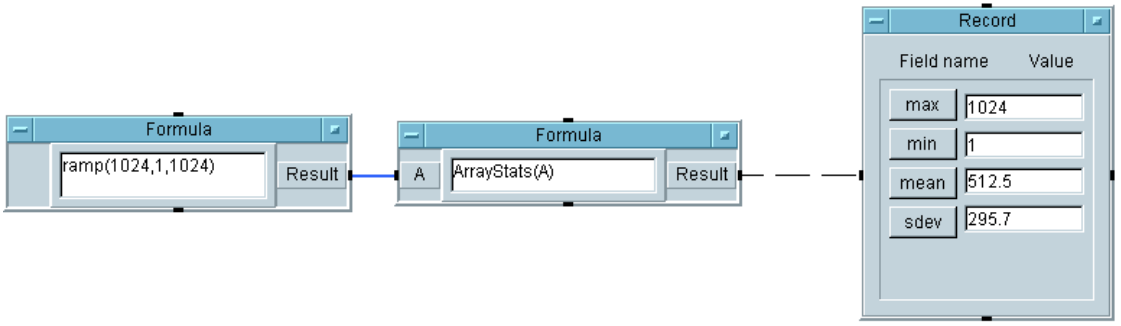


図 8-6. ユーザ関数 ArrayStats を呼出す方法

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出すことができます。

メモ

式から UserFunction を呼出した場合、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。

メモ

式から UserFunction を呼出した場合、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 オブジェクトについては、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 呼出しを生成する方法

UserFunction から呼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 を呼出すのによく使用されるオブジェクトのほとんどがあります。呼出す側のオブジェクトを選択した場合は、適切な名前とピンで適切に設定されて、メイン・プログラムなどの呼出す側のウィンドウに配置されます。

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 8-6 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 8-7 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。

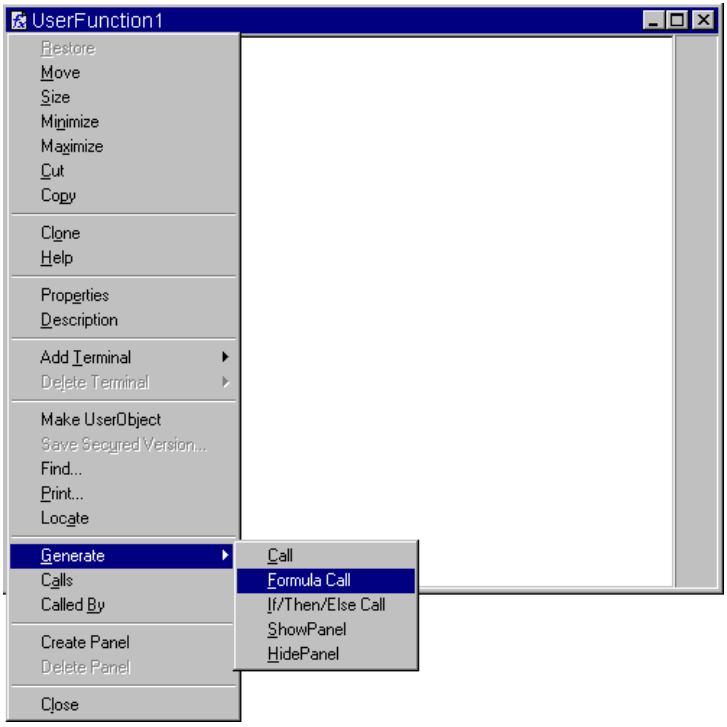


図 8-7. UserFunction の Generate メニュー

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、`ArrayStats(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 8-8 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。それらは、`Call`、(この例で使用する)`Formula Call`、`If/Then/Else Call`、`ShowPanel`、`HidePanel` オブジェクトです。

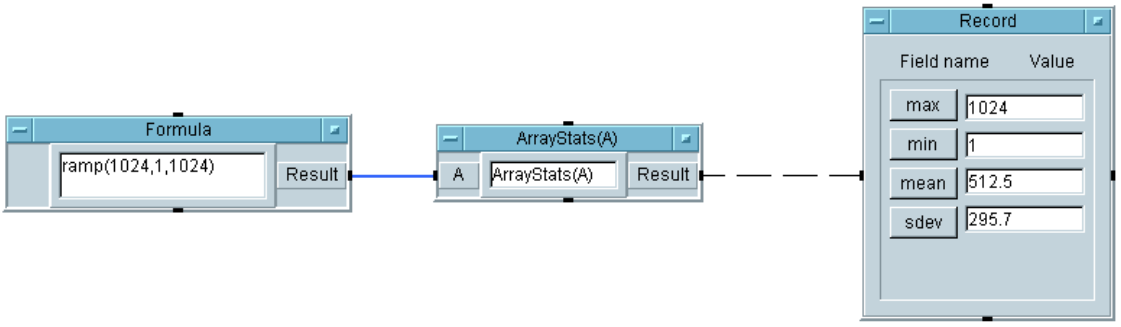


図 8-8. UserFunction から呼出しオブジェク ArrayStats(A) を生成する方法

UserFunction とプログラム・エクスプローラ

`UserFunction` と `UserObject` を使用すると、VEE プログラムがよりモジュール化されるので理解しやすくなります。プログラム・エクスプローラは、複雑なプログラムの中を探索する場合の貴重なツールです。たとえば、図 8-9 のプログラムの階層構造は、プログラム・エクスプローラを使用して表示します。プログラム・エクスプローラを表示するには、`[View]` ⇒ `[Program Explorer]` をクリックするか、ツールバーの [Program Explorer] アイコンをクリックします。

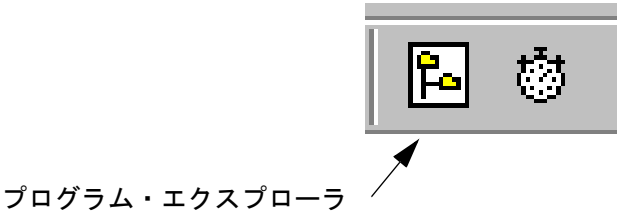


図 8-9. ツールバーの Program Explorer アイコン

図 8-10 は、使用されているプログラム・エクスプローラを示します。

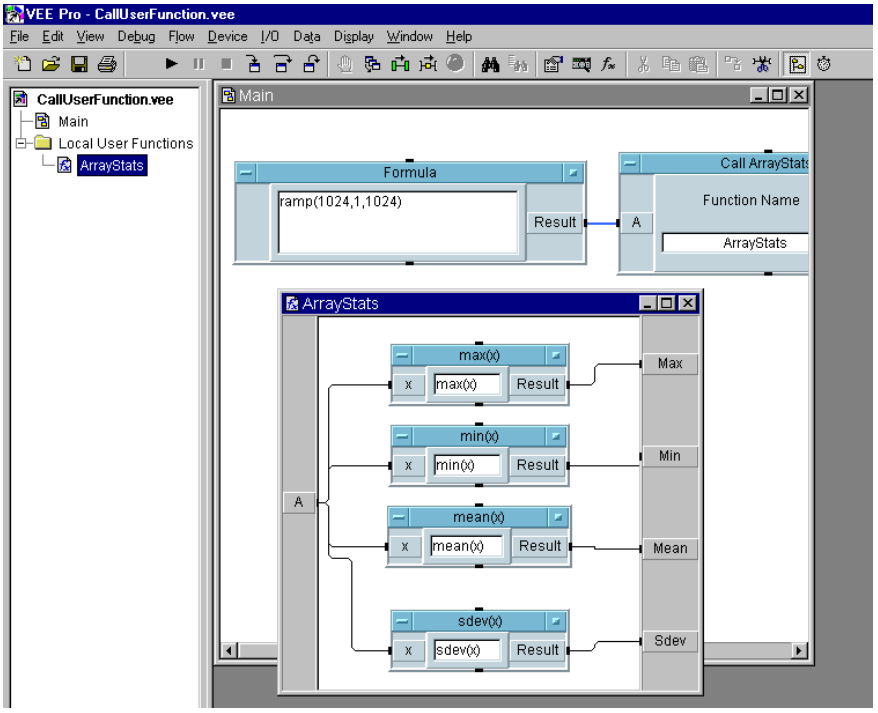


図 8-10. UserFunction の場合にプログラム・エクスプローラを使用する方法

Agilent VEE の UserFunction の場合にライブラリを使用する方法

既存の VEE テスト・プログラムの効率を高めるために、UserFunction を再利用できます。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ている複数の UserFunction から成る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の中に保管する場合、次の 2 つの方法があります。

1. [File] ⇒ [Merge Library...] コマンドを使用して、元となる UserFunction のコピーを、現在のプログラムに保管します。それぞれの UserFunction のコピーをこのプログラムに保持することになります。これらのマージされた UserFunction は編集できるので、UserFunction を変更する場合は、[File] ⇒ [Merge Library...] コマンドを使用します。

または

2. [Device] ⇒ [Import Library] オブジェクトを使用して、元となる UserFunction にアクセスします。この場合は、コピーを作成しないで、別のファイルにある元となる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。この方法では、ロードが複数回にわたって行われ、ディスク領域が浪費されず、メモリが節約されます。インポートされた UserFunction は (デバッグなどの目的で) 表示できますが、編集はできません。そのかわり、元のファイルを編集できます。[Device] ⇒ [Delete Library] オブジェクトを使用すると、インポートされた UserFunction をプログラムから削除できます。

したがって、関数の新しいコピーを作成して変更する必要がある場合や、スタンドアローン・プログラムが必要な場合は、UserFunction をマージし、関数の出所を単一にしておきたい場合やディスク領域を節約したい場合は、UserFunction を*インポート*します。

例題 8-2: UserFunction ライブラリの作成とマージの方法

この例題では、UserFunction の VEE ライブラリを保持するレポート生成プログラムを作成します。次に、その UserFunction 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 ライブラリの作成方法

1. UserFunction の細部をプログラミングするのではなく、最上位レベルのプログラムを作成します。

   a. 4 つの UserFunction を作成します。それらは、1 つの出力ピンを持つ BuildRecAry、ReportHeader、1 つの入力ピンを持つ ReportBody、ReportDisplay です。すべての UserFunction をアイコン化します。

   b. メインで、図 8-11 に示すとおりに設定して接続した 4 つの [Device] ⇒ [Call] オブジェクトを作成します。プログラムに Report.vee と名前を付けて保存します。

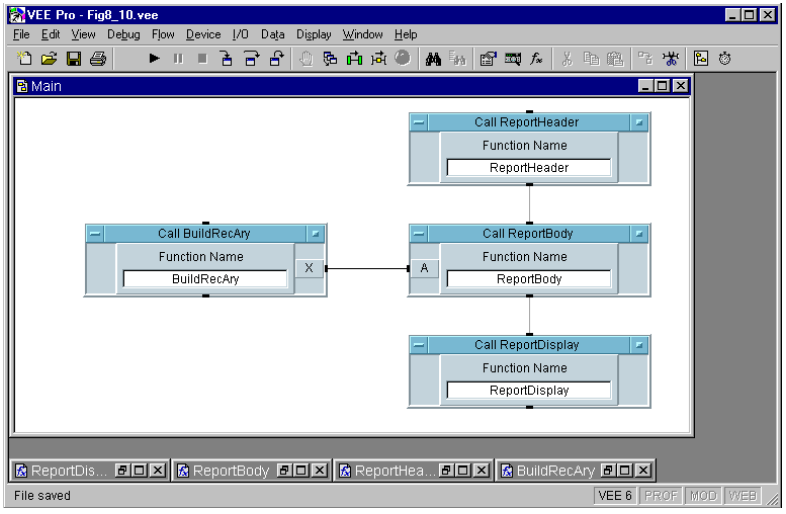


図 8-11. 最上位レベルにある Report.vee

メモ

Call オブジェクトでは、UserFunction を参照する場合、丸かっこ () は必要ありません。Formula オブジェクトから関数を呼出す場合は、関数でパラメータを使用していても使用していなくても、丸かっこ () を含める必要があります。

この 4 つの UserFunction の集まりが 1 つのライブラリです。[Edit] ⇒ [Edit UserFunction...] をクリックすると、一覧表示できます。[Cancel] をクリックし、リスト・ボックスをクローズします。

2. 以下の図に示すように、4 つの UserFunction をプログラミングします。

   図 8-12 は BuildRecAry UserFunction を示します。A<=5 であるかをテストする 3 項式を持つ Formula オブジェクトが保持されています。式の評価が TRUE である場合、このオブジェクトは PASS を出力します。そうでない場合は、FAIL を出力します。丸かっこ () が必要なので、注意してください。

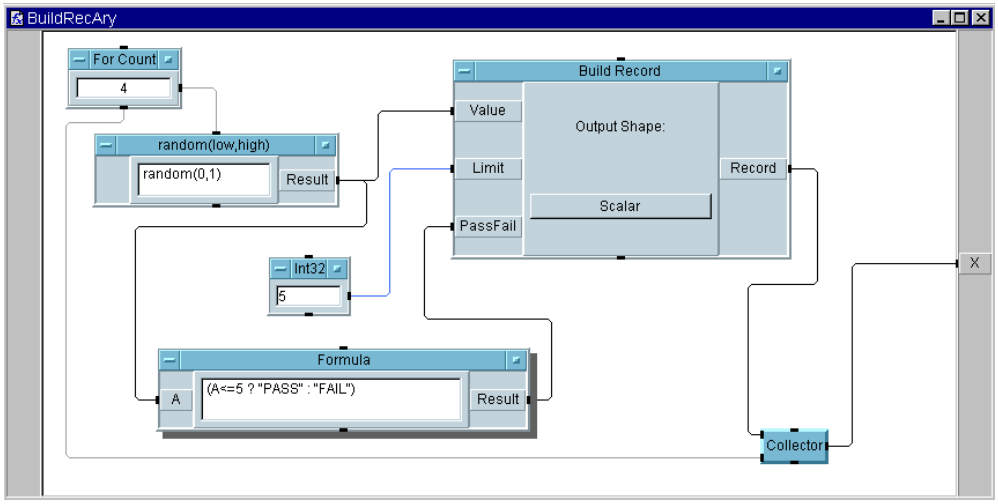


図 8-12. BuildRecAry UserFunction

図 8-13 は `ReportHeader UserFunction` を示します。

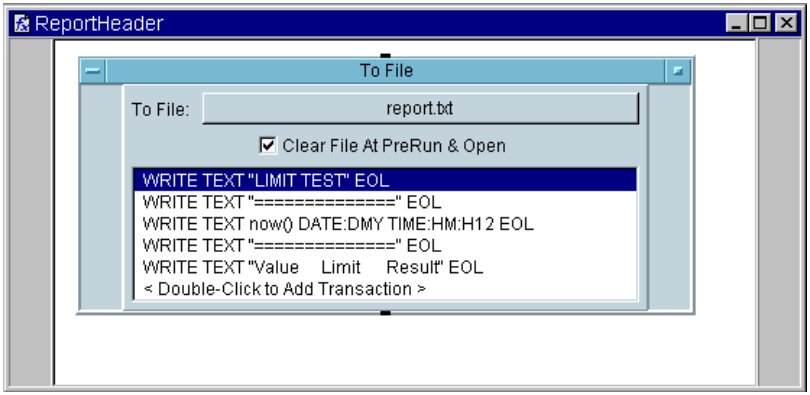


図 8-13. ReportHeader UserFunction

図 8-14 は ReportBody UserFunction を示します。レコード A[B] から成る配列であることに注意してください。< *レコード*>.< *フィールド*> 表記を使用すると、B の値が 0 から 1、2 へと変わっていくにつれ、Value、Limit、PassFail という Record 内の特定のフィールドにアクセスできます。For Count の出力はゼロから開始されるので、注意してください。最初のトランザクションが EOL off を持つことにも注意してください。

引用符で囲んだ縦棒 "|" は、縦棒の定数文字列文字を表します。FW:5 は、5 文字の幅を持つ文字列フィールドを表します。RJ は右詰めを表します。

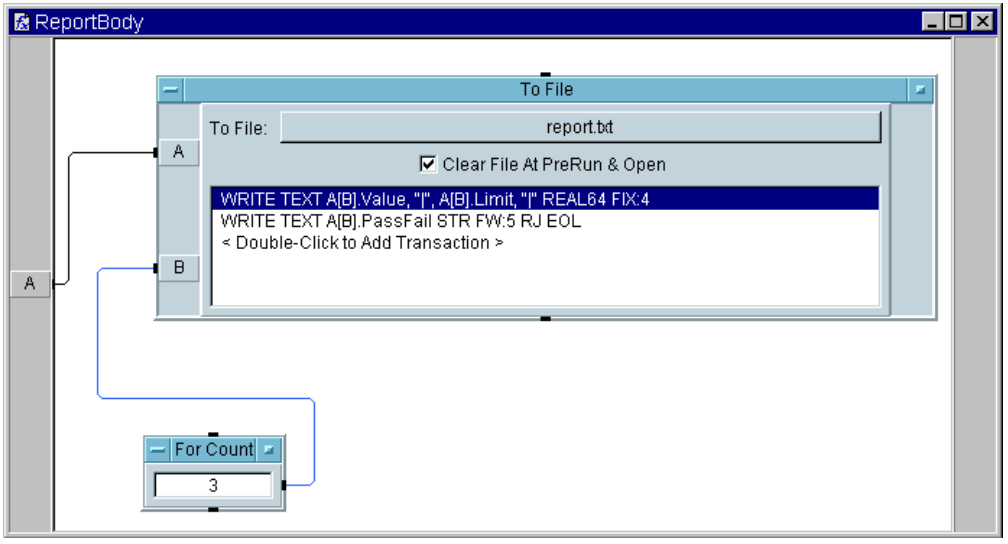


図 8-14. ReportBody UserFunction

図 8-15 は、ReportDisplay UserFunction を詳細ビューで示したものです。この関数は、アスタリスク (*) で指定した文字列配列を、STR ARRAY フォーマットに従って、ファイルの末尾に読込みます。

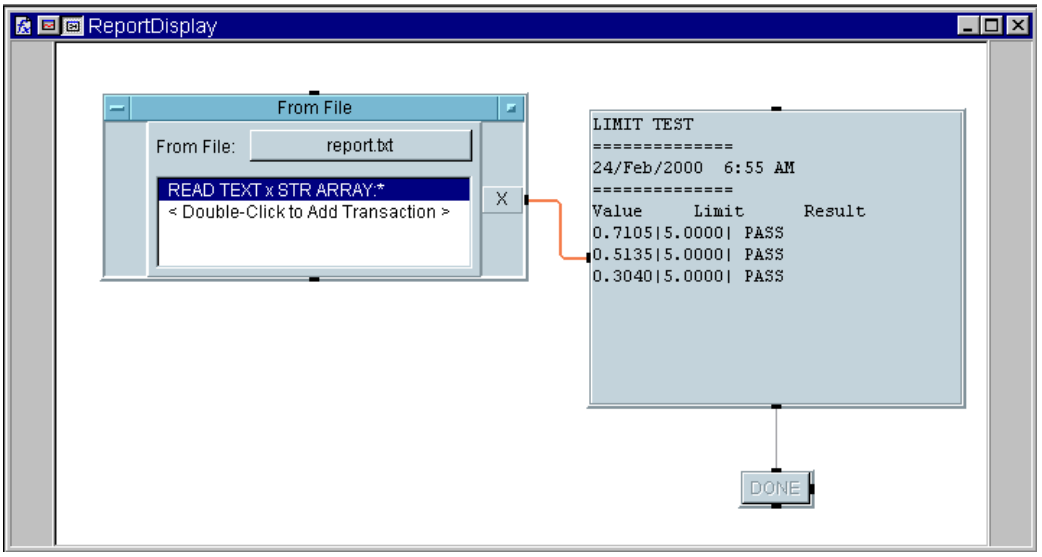


図 8-15. ReportDisplay の詳細ビュー

図 8-16 は、[Properties] ボックスで [Show Panel on Execute] が選択されている ReportDisplay UserFunction のパネル・ビューを示します。[Properties] ボックスで、Pop-up Panel Title も ReportDisplay に変更されています。パネルを作成するには、Confirm (OK) と 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 表示では、[Show Title Bar] の選択が解除されています。[Confirm (OK)] ボタンの名前が DONE に変更されています。[Confirm (OK)] ボタンがあるのは、ユーザがボタンを押すまで、Logging AlphaNumeric 表示を画面に表示し続けるためです。

3. プログラムを実行すると、ReportDisplay パネルがポップアップし、図 8-16 に示す値が表示されます。[DONE] をクリックして実行を完了します。次に、プログラムを Report.vee としてセーブします。

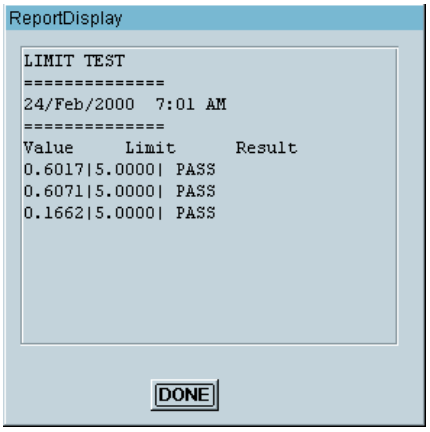


図 8-16. ReportDisplay のパネル・ビュー

別のプログラムを作成しライブラリをマージする方法

この例題では、新しいプログラムを作成し、その中にライブラリをマージします。ここでは、レポートを生成するための関数から成るライブラリを構築します。新しいプログラムは、ライブラリ内の各関数について説明する Note Pad オブジェクトを保持します。新しいプログラムには RepGen という名前を付けます。

新しいレポート生成用の User Function を作成し、プログラムとマージし、Note Pad オブジェクトを更新して関数に関する情報を最新にすることにより、RepGen を再利用できます。次に、Merge Library... コマンドを使用すると、RepGen にあるすべての関数を利用できます。

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は、ファイル・パス全体を入力します。

   [Edit] ⇒ [Edit UserFunction] を選択し (またはプログラム・エクスプローラを調べて) 、Report.vee から新しいプログラムにライブラリが転送されたかどうかを確認します。Merge Library... コマンドを使用した場合は、ローカル関数を編集するように、マージされた関数を編集できます。

3. [Display] ⇒ [Note Pad] を選択し、図 8-17 に示しているように UserFunction についての記述を入力します。次に、プログラムを RepGen.vee としてセーブします。

メモ

関数を呼出す VEE プログラムが実際には存在しない場合でも、ライブラリを作成する目的で、UserFunction 自体の「プログラム」をセーブできます。

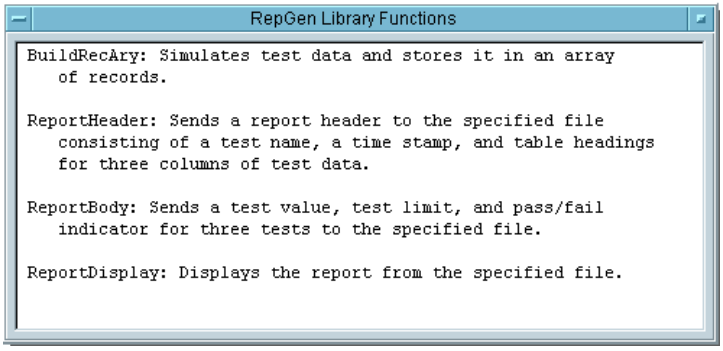


図 8-17. UserFunction から成る RepGen.vee ライブラリ

例題 8-3: ライブラリのインポートと削除の方法

UserFunction から成るライブラリを作成しても、プログラムにライブラリをマージしないこともあります。ライブラリを実行時に持ってきて、その中の一部の関数を使用したうえで、メモリを浪費しないようにライブラリを削除できます。Import Library と Delete Library オブジェクトは、この場合のために設計されています。

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 関数を呼出して、あるテスト・データをシミュレートして表示し、最後にその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、メイン内に配置します。Import Library オブジェクト内のフィールドを次のように設定します。

Library Type

[Library Type] フィールド内のメニューを使用すると、UserFunction、コンパイル済み関数、またはリモート関数を選択できます。この場合、必要なのは UserFunction ライブラリなので、デフォルトのままにしておきます。

Library Name

[Library Name] には、デフォルトで myLib が表示されています。この名前は、インポートされるそれぞれ異なるライブラリを区別するための「ハンドル」として、VEE プログラムによって使用されます。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。デフォルトの名前を使用して差支えありません。

File Name

PC では、[File Name] フィールドには、ユーザ・プログラムのデフォルトのディレクトリを選択するためのダイアログ・ボックスが表示されます。HP-UX システムは、VEE を起動したときのディレクトリにアクセスします。関数ライブラリを保持してい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。316 ページの「UserFunction ライブラリの作成とマージの方法」にあるように、[RepGen.vee] を選択します。このファイルは、インストール時にプログラムの場所として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、実行時まで待つのではなく、ただちにライブラリをインポートするために [Load Lib] を選択します。このコマンドは開発段階で役に立ちます。次のステップで、Call オブジェクト内で [Select Function] を選択したときに、myLib.BuildRecAry などと、先頭にライブラリ・ハンドルの付いた関数が示されます。

   インポートされた関数のライブラリを表示するには、プログラム・エクスプローラを使用します。

メモ

ライブラリのインポートが終わったので、UserFunction のみを表示して、ブレークポイントを設定できます。UserFunction は編集できません。編集できる 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 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、Load Lib コマンドを使ってインポートした関数の一覧を表示します。図 8-18 に示すように、[myLib.BuildRecAry] を選択します。

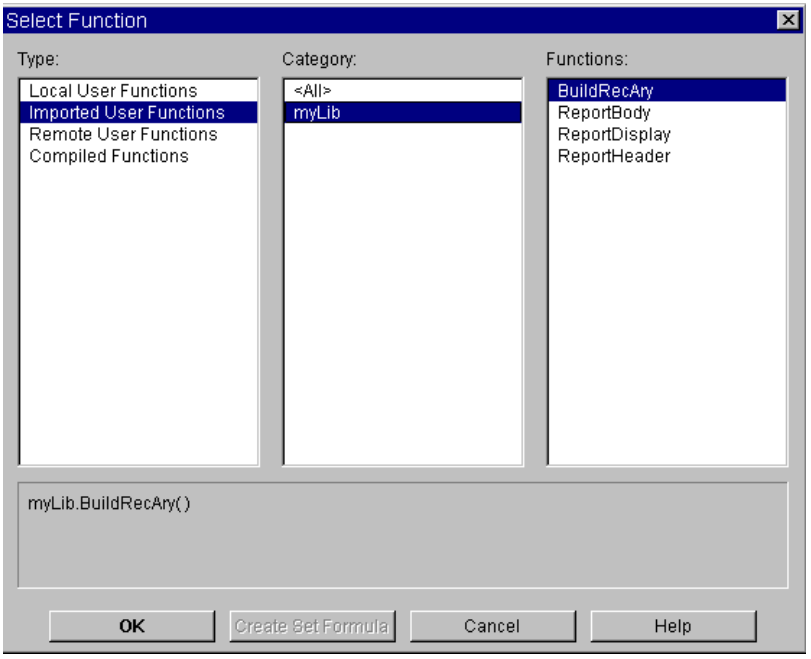


図 8-18. インポートされたライブラリから関数を選択する方法

VEE は、自動的に [Function Name] フィールドに関数を挿入し、必要な出力端子を追加します。[Function Name] フィールドに

「myLib.BuildRecAry」と入力しても、同じ結果が得られます。ライブラリ内の関数の名前を一覧表示する必要がある場合は、Select Function を使用します。

6. AlphaNumeric 表示を選択して拡大し、Call の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・ピンどうしを接続します。そうすると、BuildRecAry 関数が呼出された後でライブラリが削除されます。デフォルトの Library Name は、Import Library オブジェクトの場合に使用した名前と同じなので、そのままでかまいません。

8. プログラムを実行します。図 8-19 のように表示されます。プログラムを「libraries.vee」のファイル名で保存します。

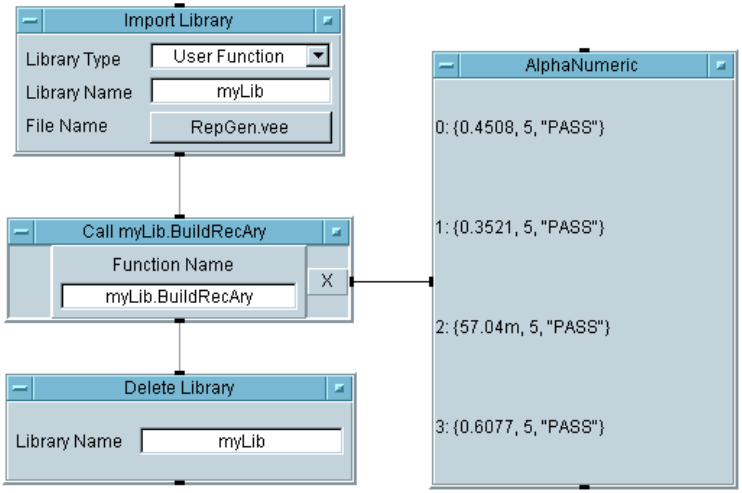


図 8-19. ライブラリから関数を呼出す方法

マージされた関数とインポートされた関数の名前については、下記の点に注意してください。

- マージされた関数の名前がローカル関数の名前と同じ場合、VEE はエラーを表示します。

- インポートされた関数の名前がローカル関数の名前と同じでもよいのですが、関数名のみを使用すると、ローカル関数が呼出されます。図 8-19 の `Call` オブジェクトにあるような `MyLib_func()` 構文を使用すると、インポートされた関数を明示的に呼出せます。.

- 2 つのインポートされたライブラリで同じ関数名を使用している場合、どうなるかはわかりません。`Call` オブジェクトは `Library` 名 `myLib.BuildRecAry` を使用しているため混乱しません。同じ名前を持つ別のローカル関数または別のインポートされた関数が存在したとしても、この名前によって、`BuildRecAry` の名前と場所が指定されるためです。

大規模プログラム内の関数を検索する方法

VEE は、大規模プログラム内でのオブジェクトやテキストの検索に役立つように Find 機能を提供します。この機能は [Edit] メニューにあります。たとえば、Examples/Games ディレクトリ内の Solitaire.vee プログラムをオープンします。詳細ビューに切替え、[Edit] ⇒ [Find...] をクリックして、図 8-20 に示すダイアログ・ボックスを表示します。

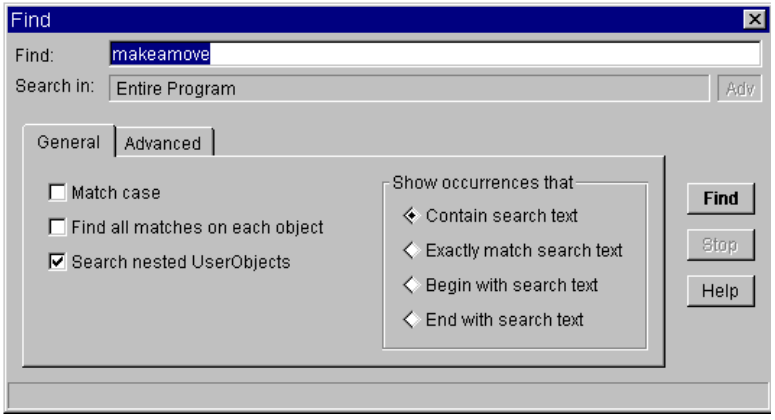


図 8-20. [Find] ダイアログ・ボックス

図に示すように、このプログラム内の UserFunction の名前を「makeamove」と入力し、[Find] をクリックします。VEE は、自動的に makeamove という名前の UserFunction を検索し、図 8-21 に示すように、この関数が呼出されたプログラムの部分を表示します。

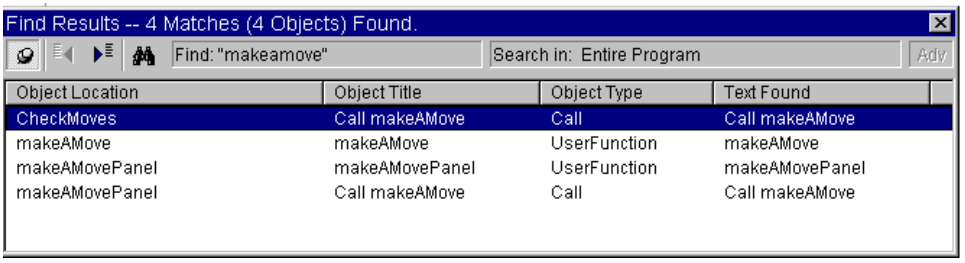


図 8-21. [Find Results] ダイアログ・ボックス

`Find` を使用すると、オブジェクトにある変数、タイトル、設定など、どのようなオブジェクトやテキストでも検索できます。`[Find Results]` ボックス内のどの行をダブルクリックしても、オブジェクトを検索できます。

メモ

プログラム・エクスプローラで、マウス・ポインタをオブジェクト上に置き、マウスの右ボタンをクリックしても、`Find` を使用できます。これにより、検索範囲が、特定の `UserFunction` や `UserObject` に限定されます。

Agilent VEE プログラムのマージ

既存のプログラムの効率を高める最も容易な方法は、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。マージしてから、現在の必要に合うように編集すると、プログラムを再利用できます。

[File] ⇒ [Merge...] コマンドは、プログラムの内容またはセーブされているオブジェクトの集合を作業領域に追加すると同時に、作業領域の既存の内容を保持します。デフォルトでは、[File] ⇒ [Merge...] は、VEE とともに出荷されたプログラムのあるディレクトリを表示します。それらのプログラムは、棒グラフ表示や (ID 番号などの) ユーザ入力用のデータ入力キーパッドなどよく使用されるプログラムです。

例題 8-4: 棒グラフ表示プログラムをマージする方法

この例題では、既存のプログラムと新しいプログラムをマージします。例では、VEELib ディレクトリにあるプログラムを使用しますが、どのプログラムでも使用できます。ramp() 関数を使用して、1 から 5 までの 5 つの値を持つ配列を作成します。その配列を VEE 内部の表示オブジェクトを使って表示するのではなく、BarChart プログラムとマージします。

1. [Device] ⇒ [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。

   1 番目のパラメータは ramp 配列内の目的の要素の数、2 番目のパラメータは開始位置、3 番目のパラメータは終了位置です。この関数の詳細は、Formula には現在 ramp 呼出しがあるため、オブジェクト・メニュー内の [Help] を選択します。または、[Help] ⇒ [Contents & Index] を選択し、次に、Index フォルダ内の Search 機能を使用します。
4. [File] ⇒ [Merge...] をクリックして、[Merge File] リスト・ボックスを表示します。BarChart.vee を選択して、Formula オブジェクトの右に配置します。2 つのオブジェクトどうしを接続します。

5. プログラムを実行します。図 8-22 のように表示されます。

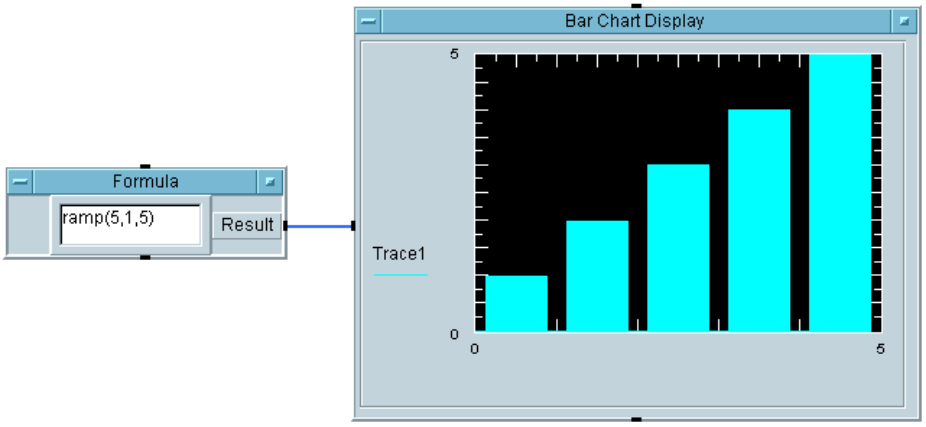


図 8-22. BarChart プログラムをマージする方法

Bar Chart Display が 1 次元の配列を受取り、値を棒グラフで表示します。このプログラムでは、配列内の値を表示するのに必要なだけのバーが使用されます。このプログラムがどのように作成されているかを理解するには、表示の詳細ビューをオープンします。プログラムの作成方法に関するアイデアをさらに知りたい場合は、library ディレクトリで実例を調べることができます。

メモ

`[File]` ⇒ `[Merge]` コマンドは、`UserObject` とオブジェクトにマージするのに使用されます。`[File]` ⇒ `[Merge Library]` コマンドは、`UserFunction` をマージするのに使用されます。

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、必要なら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とリモート関数と比較する。
- UserFunction を作成し、呼出し、編集する。
- 複数の UserFunction ライブラリを作成し、マージし、インポートし、削除する。
- いずれかのゲーム・プログラムで 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。

9

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 テストとデータを受渡す方法
- 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

平均的な必要時間 :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
- テスト間にまたがる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出せること
- テスト結果の自動ログ記録が可能なこと

メモ

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。

メモ

一連のテストによって更新されるステータス・パネルを使用する場合は、407 ページの「ステータス・パネルを作成する方法」を参照してください。

メモ

この章の例題のほかに、Sequencer の使用について学ぶ場合は、付録 A「追加の例題」の 516 ページの「テスト・シーケンサ」を参照してください。

Sequencer オブジェクトの使用方法

Sequencer オブジェクトは、複数のテストを指定された順序で実行時の結果を基にして実行します。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 9-1 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取ります。

例題 9-1: テストの設定方法

メモ

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。

メモ

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。

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。
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 を使用します。

その他のフィールドはデフォルト値のままにしておけます。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 9-1 のように表示されます。

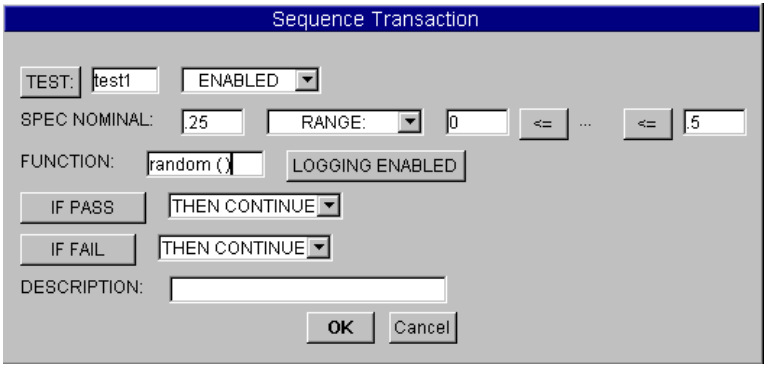


図 9-1. [Sequence Transaction] ダイアログ・ボックス

[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 9-2 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。

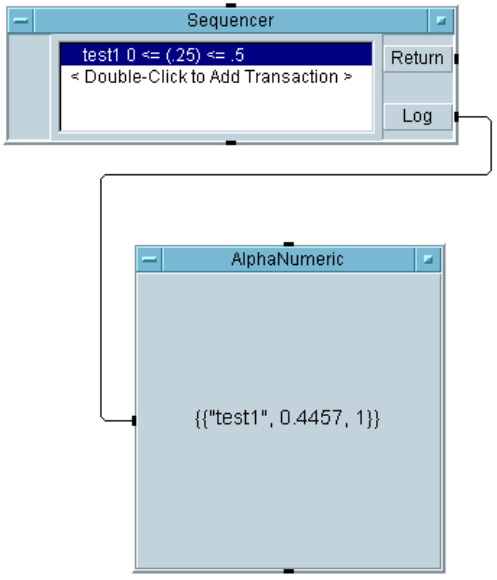


図 9-2. テストの設定方法

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 9-1 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を学んでください。

表 9-1. [Sequence Transaction] ダイアログ・ボックス

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が行われないで、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ディセーブルされます。
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 (A == 1) を保持する場合にテストを実行できます。監査テストの制御のた GI に ENABLED IF: を使用できます。たとえば、10 回テストを実行するたびに、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。

表 9-1. [Sequence Transaction] ダイアログ・ボックス (続き)

フィールド	説明
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・データの比較のた GI 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誤差のパーセンテージを指定することにより、許容範囲を指定します。
FUNCTION:	実行するテストを指定します。UserFunction、コンパイル済み関数、リモート関数を呼出すか、評価対象となる式を入力できます。呼出した関数 (または評価対象の式) の結果は、仕様と比較してテストされます。 UserFunction が 1 つの値以上を返す場合、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。 関数は組合わせたりネストすることもできます。たとえば、(random(0,myfunc()+3,100)*2)。

表 9-1. [Sequence Transaction] ダイアログ・ボックス (続き)

フィールド	説明
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。デフォルトでは、[Name]、[Result]、[Pass] にチェック・マークが付けられています。[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 9-1. [Sequence Transaction] ダイアログ・ボックス (続き)

フィールド	説明
THEN CONTINUE	テストへの分岐を決めます。[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、現在のテストを繰返します。最大回数だけ繰返した後もまだ 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 出力ピンを使って蒜 l 捉できます。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 GI ることができます。

表 9-1. [Sequence Transaction] ダイアログ・ボックス (続き)

フィールド	説明
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

この節では、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。

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。

メモ

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。

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、表示を拡大する場合もあります。

3. [Insert] トランザクション・バーをクリックして、Insert を削除し、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。

メモ

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、切取っておいたトランザクションを貼付けることもできます (ショートカットは Ctrl+Y)。同じようにして、[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 9-3 のように表示されます。

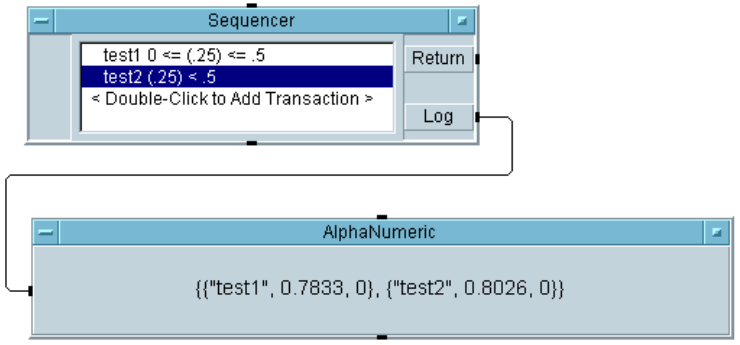


図 9-3. 単純な Sequencer の例

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 9-4 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。

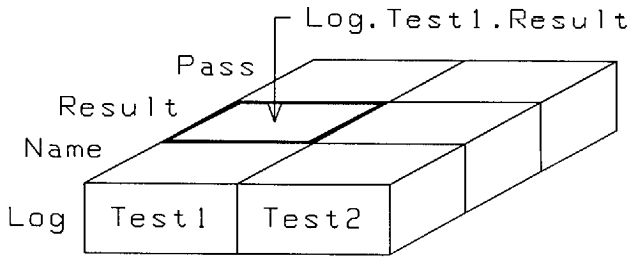


図 9-4. ログとして記録されているレコードまたは複数のレコード

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ないので、名前の中で大文字を使用しているのは、マニュアルの中で名前を明確に示すためです。

   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 9-5 のように表示されます。

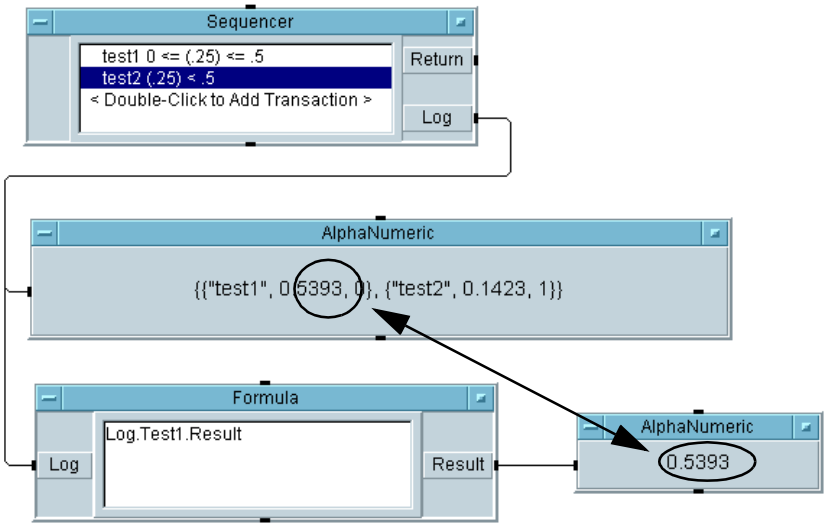


図 9-5. ログとして記録されているデータにアクセスする方法

メモ

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 9-2: 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 9-6 のように表示されます。

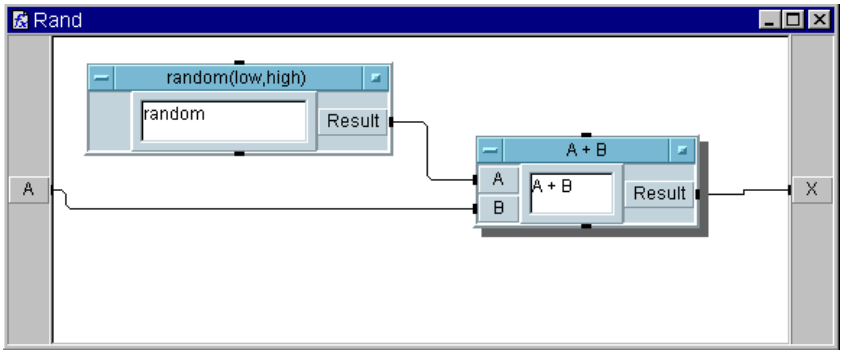


図 9-6. Rand UserFunction

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。

メモ

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 を呼出し、Sequencer の入力ピンを使って入力パラメータを Rand に与えます。

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。

メモ

Sequencer の入力端子名 A などを使用しても、[Sequence Transaction] ボックス内のどの式フィールドにもデータを渡せます。たとえば、A を使用すると、RANGE: と SPEC NOMINAL: にデータを渡せます。

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、テストをカット元の Sequencer に貼付けます。オブジェクト・メニューを使ってもカットと貼付けができます。

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。

   ヒント: 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 9-7 のように表示されます。

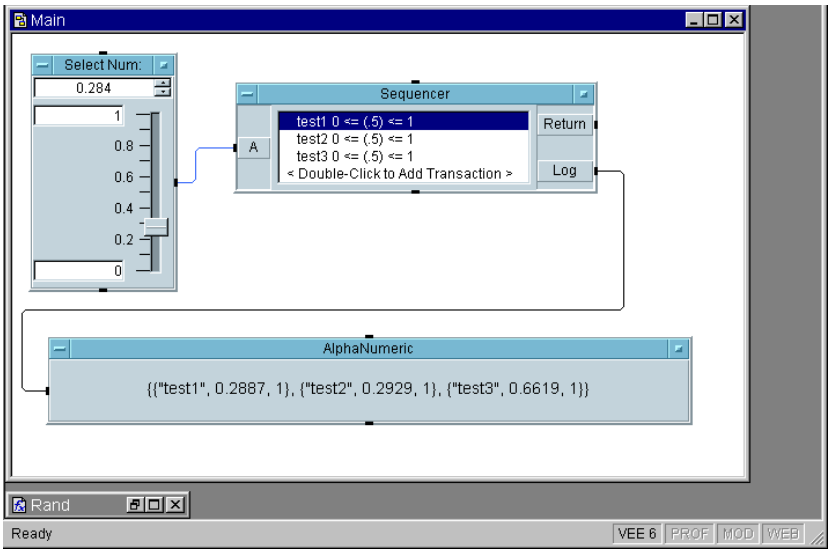


図 9-7. 入力端子を使用してデータを渡す方法

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 を使用しても、グローバル変数をセットアップできます。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。

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される結果は生成しないことにあります。

3. [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。

   次に、UserFunction global() を作成します。

4. [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。

5. [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。

メモ

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。

6. [Display] ⇒ [Note Pad] を選択し、[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.
   ```

7. 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 画面内の空いている領域、UserObject、または UserFunction の詳細ビューで右ボタン・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。

メモ

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、[Show Panel on Execute] をクリックして選択します。図 9-8 は詳細ビューの UserFunction を示し、図 9-9 はそのパネル・ビューを示します。

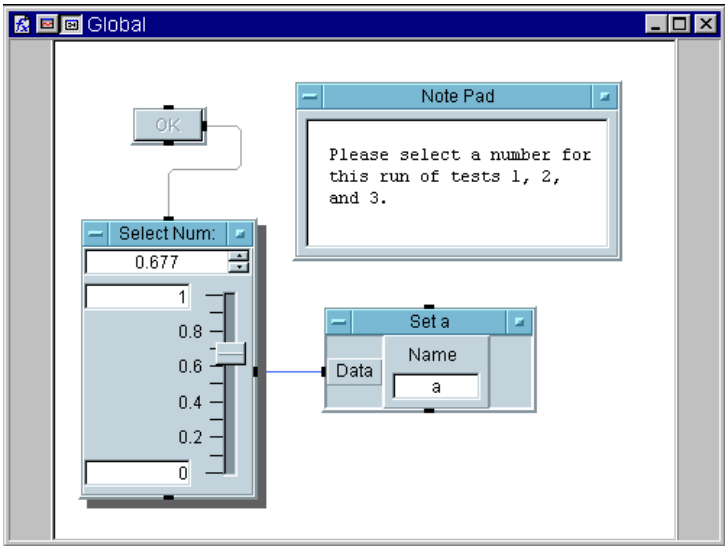


図 9-8. Global UserFunction (詳細)

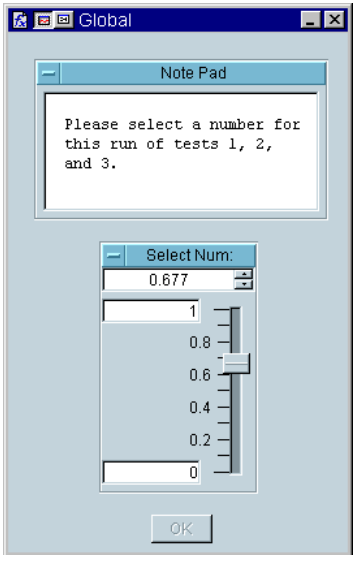


図 9-9. Global UserFunction (パネル)

8. 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 9-10 のように表示されます。

メモ

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。

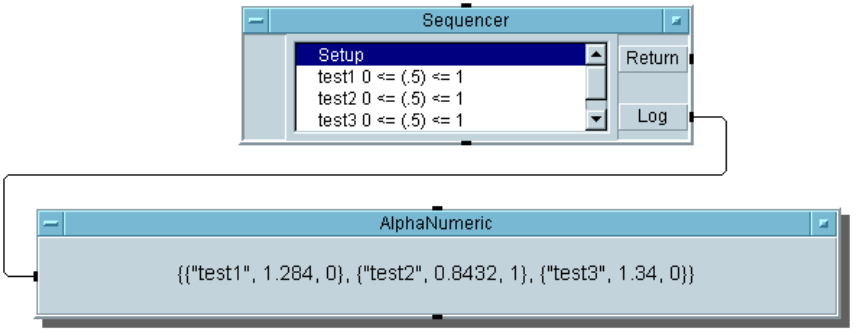


図 9-10. グローバル変数を使用してデータを渡す方法

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波形をテストします。

1. 図 9-11 の詳細ビューで示されているように noisyWv という名前の UserFunction を作成します。

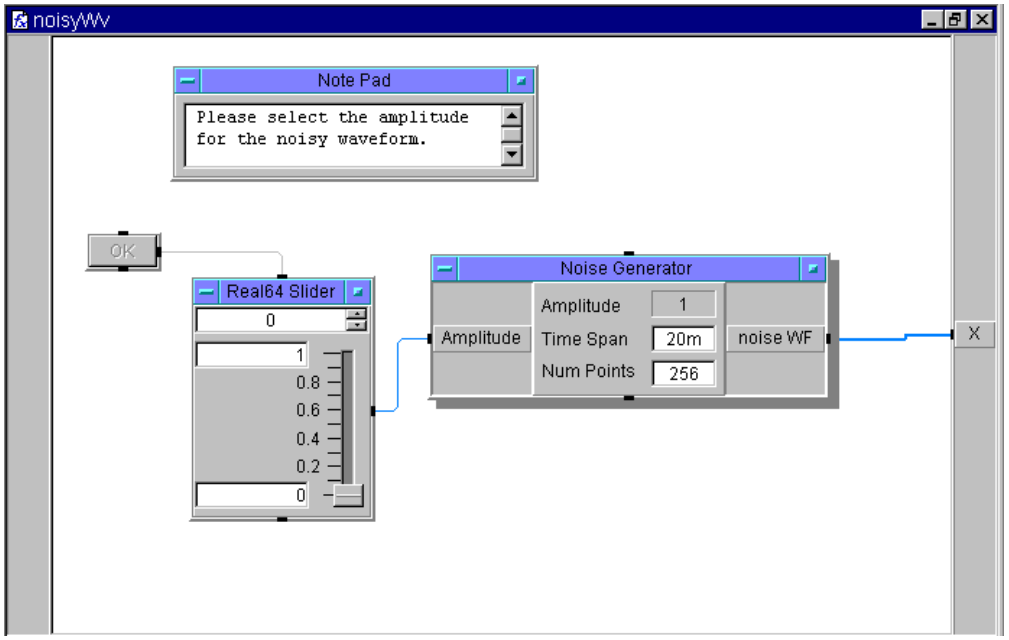


図 9-11. noisyWv UserFunction(詳細)

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 9-12 のように表示されます。

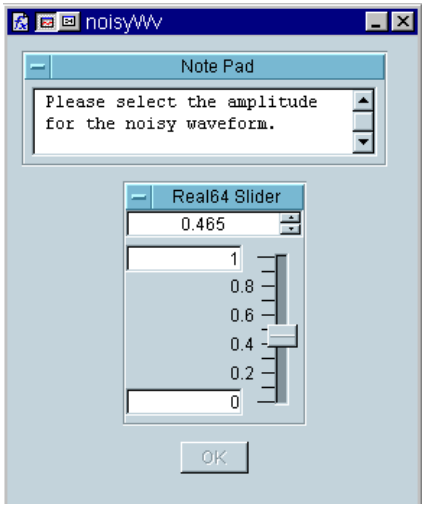


図 9-12. noisyWv UserObject(パネル)

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、[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。

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の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波がマスクと比較して小さいか等しい場合、ノイズ波のテストは成功 (PASS) です。そうでない場合、ノイズ波のテストは失敗 (FAIL) です。

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。

Configuration	1D Array に設定します。
Size	「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。

   [OK] をクリックします。

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。コ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。

メモ

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、幅を拡げ、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 9-13 のように表示されます。

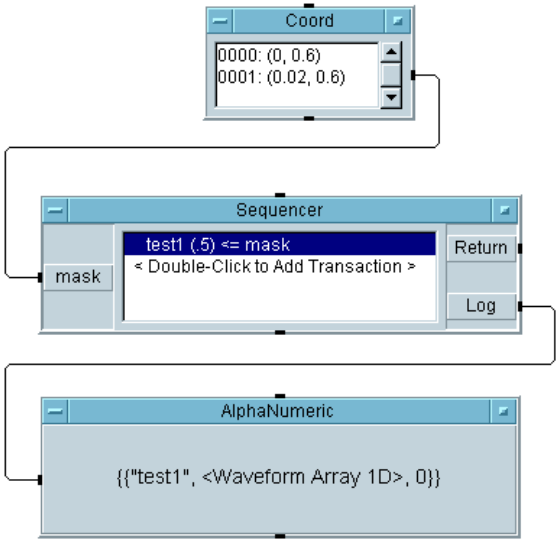


図 9-13. 波形をマスクと比較する方法

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 9-14 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。

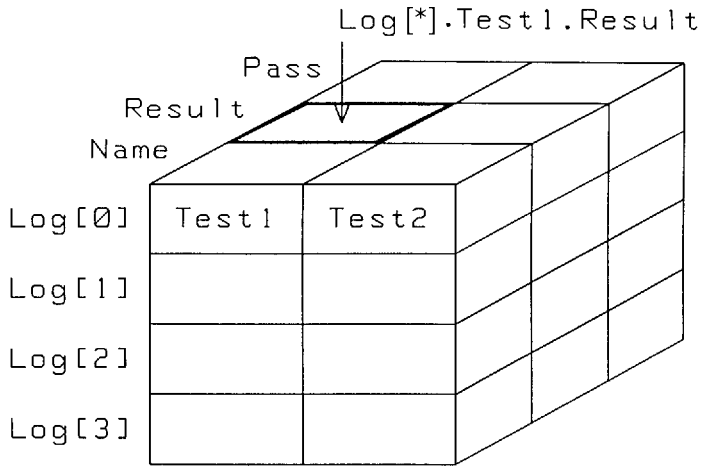


図 9-14. ログとして記録されたレコードから成るレコードを要素とする配列

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 9-3: 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芳香に少し拡大します。

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、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 9-14 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 9-15 のように表示されます。

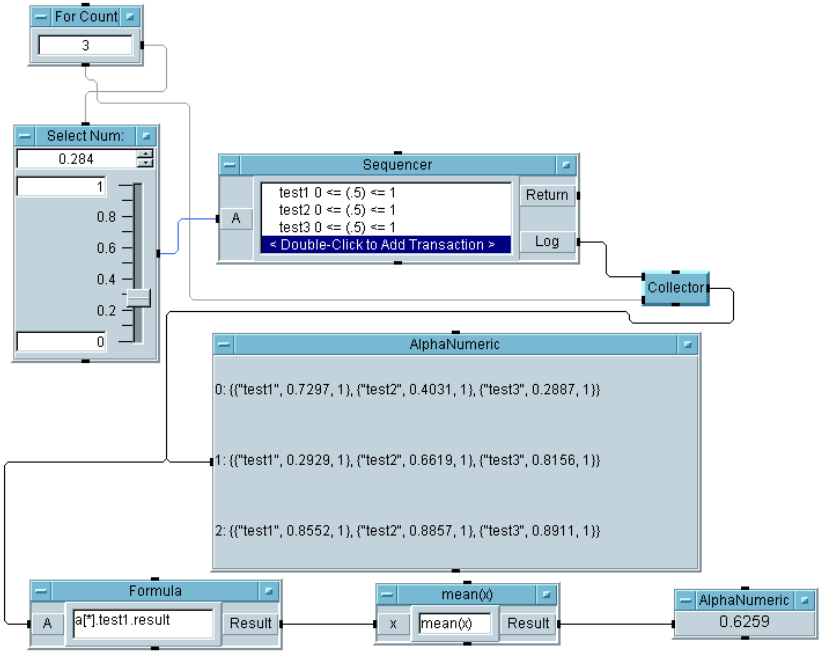


図 9-15. Sequencer を数回実行して得られたデータを解析する方法

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 9-4: 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 9-16 のように表示されます。

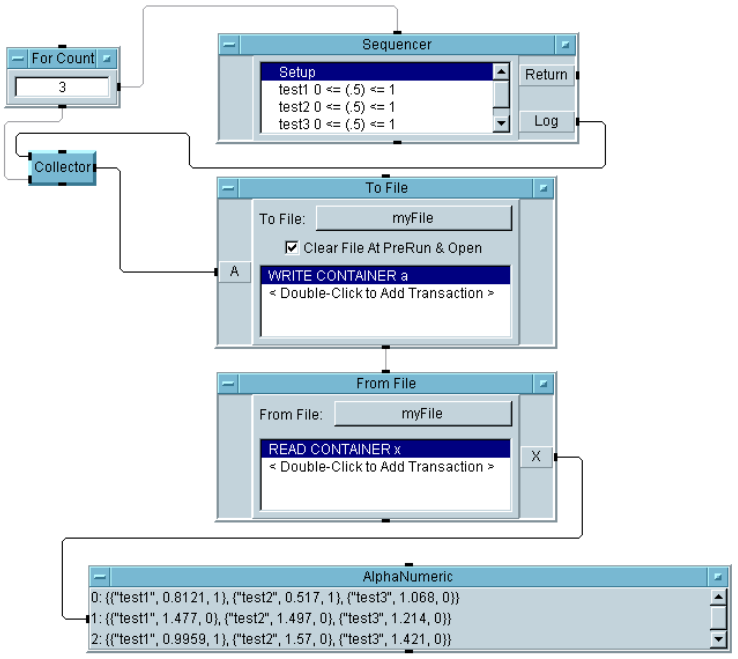


図 9-16. To/From File を使用して、ログとして記録されているデータを保管する方法

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 9-17 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。

メモ

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。

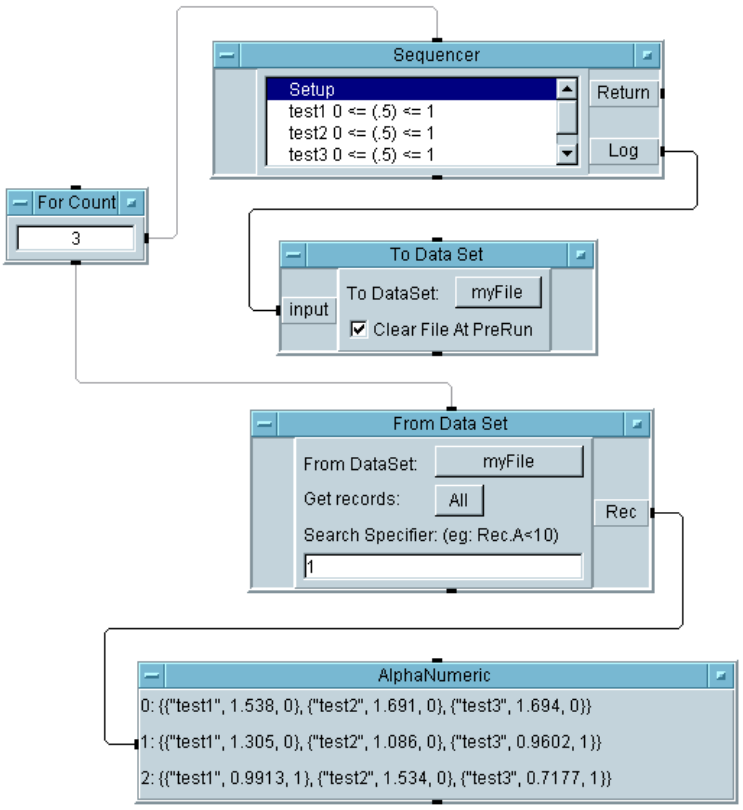


図 9-17. To/From DataSet を使用して、ログとして記録されているデータを保管する方法

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 によるテストについて操作を追加、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。
- Global variable 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。

10

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

平均的な必要時間 :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・インタフェースの詳細を習得します。この章では、前のほうの章でオペレータ・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詳細にしています。

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

この節では、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 10-1 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。

メモ

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、`UserObject` 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。

.

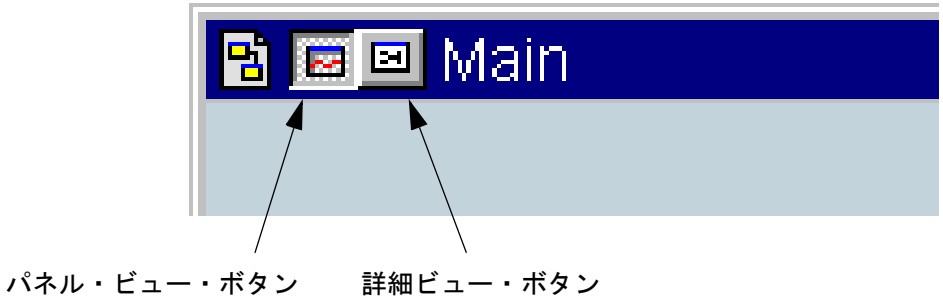


図 10-1. タイトル・バーのパネル・ビュー・ボタンと詳細ビュー・ボタン

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、強調を加え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、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て、タイトル・バーを編集します。

図 10-2 は、使用可能な VEE インジケータを示したものです。

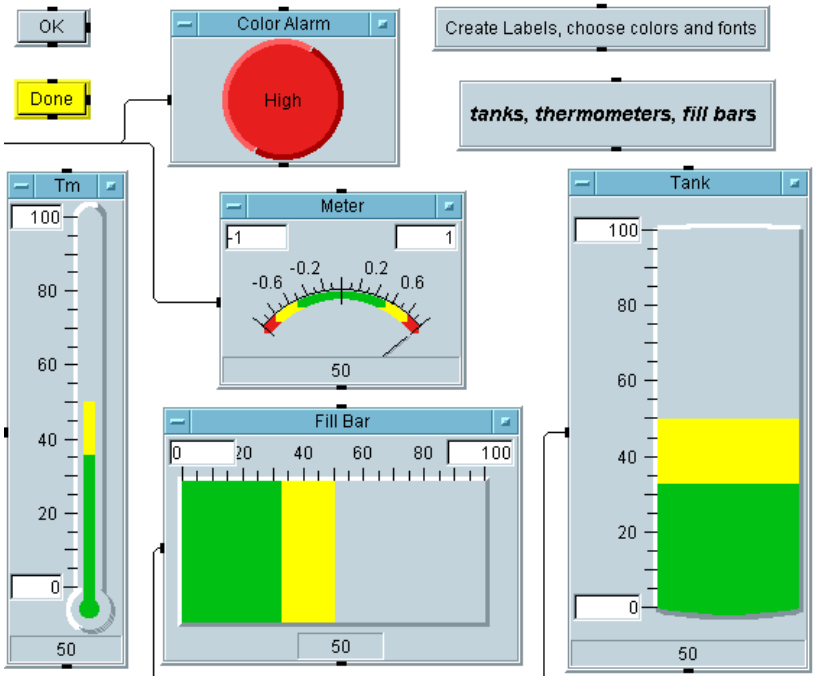


図 10-2. 選択の対象となる VEE インジケータ

オペレータ・インタフェース・オブジェクトの使用方法

この節では、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。この節を一とおり読むと、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、通常のタスクの一部のオペレータ・インタフェースをセットアップする方法を理解するための実習を行います。

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 Alarm　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、または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は、[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、デフォルトの 3 つの異なる範囲を保持するように設定できます。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、*.xwd、*.GIF、*.bmp、*.icn で、メイン、UserObject、または 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、397 ページの「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、切取ったり、中央寄せできます。図 10-3 は、サイズを設定して背景イメージとして使用している VEE のロゴ・マークを示します。

図 10-3. 背景ピクチャとして使用しているロゴ・マーク

図 10-4 は、タイルにした背景ピクチャを示します。

図 10-4. 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 10-5 は、[Display] ⇒ [Picture] を選択して組込んだ上で、VEE で切取ったピクチャを示します。

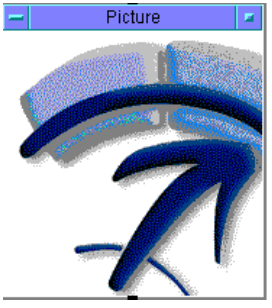


図 10-5. VEE で切取られたイメージ

メモ

[Properties] ⇒ [Icon] タブを使用すると、ビットマップをアイコン用に変更できます。

オペレータ入力のためのコントロールの表示方法

入力を行うことによってオペレータが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 10-6 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。

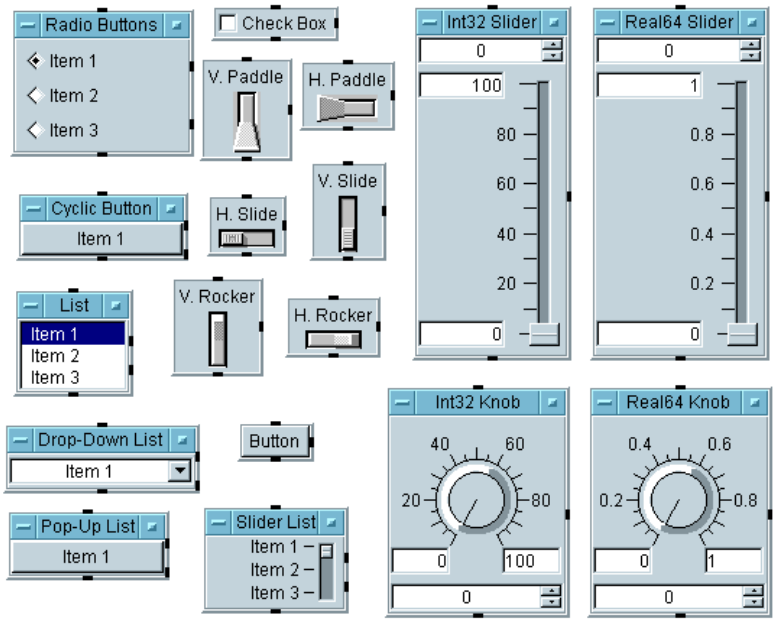


図 10-6. [Data] のさまざまなサブメニューにあるコントロール

図 10-6 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 10-7 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、Colors などのフォルダを選択して行います。

メモ

ActiveX を使用すると、404 ページの「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。

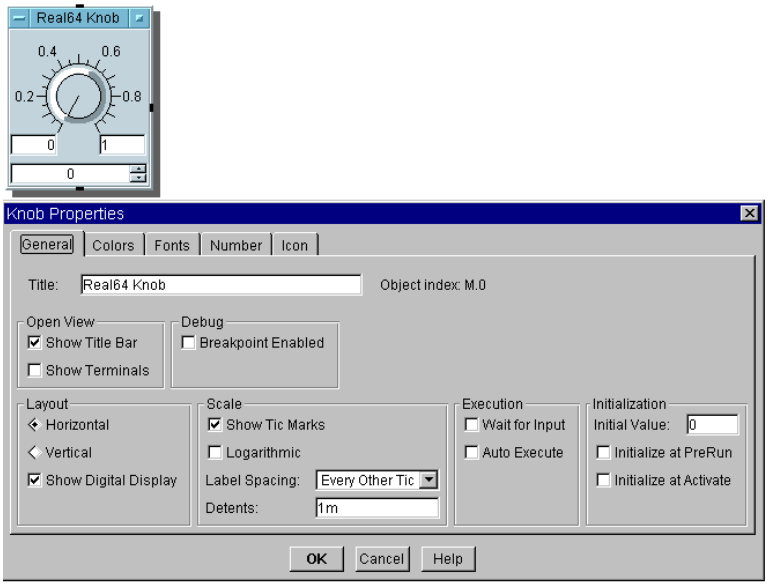


図 10-7. [Properties] ダイアログ・ボックス

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 10-8 は、プログラムに組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。

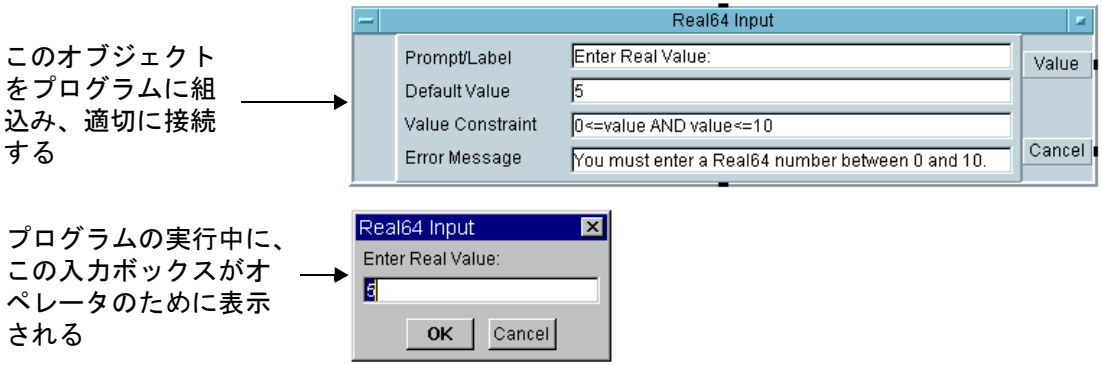


図 10-8. テキスト入力ボックス

図 10-9 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 10-9. 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 10-10 は、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。

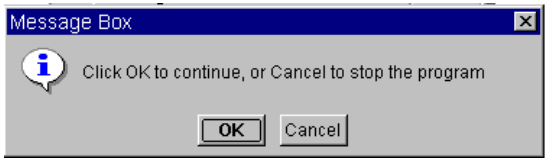


図 10-10. ポップアップ・メッセージ・ボックス

図 10-11 は、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。

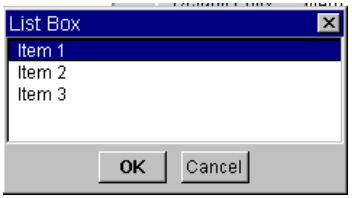


図 10-11. リスト選択ボックス

図 10-12 は、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。

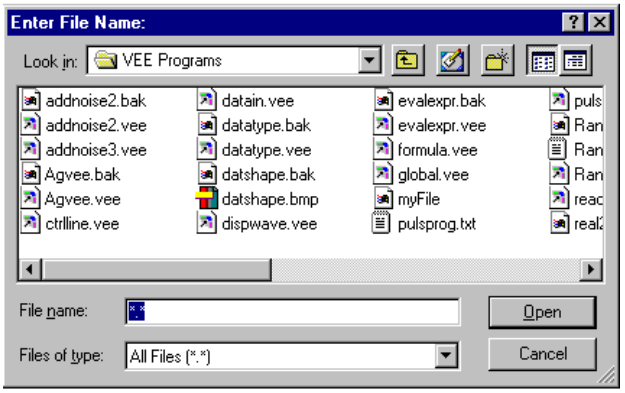


図 10-12. ポップアップ・ファイル選択ボックス

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 10-13 は、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。

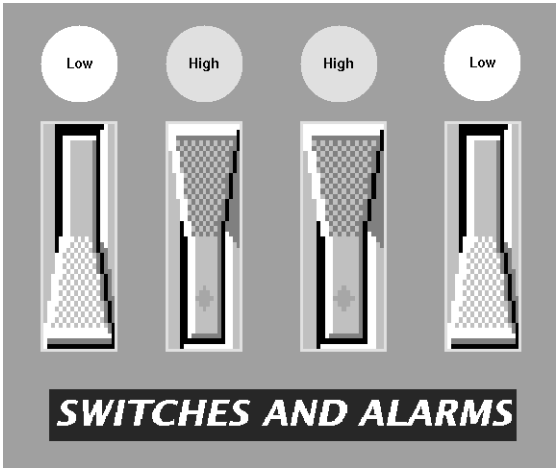


図 10-13. 組合わせたスイッチと Alarm

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、オブジェクトの位置合わせに役立つ「snap-to-grid」という機能があります。図 10-14 に示すように、グリッドのサイズを 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` フォルダを選択するダイアログ・ボックスを表示するためのパネル・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。

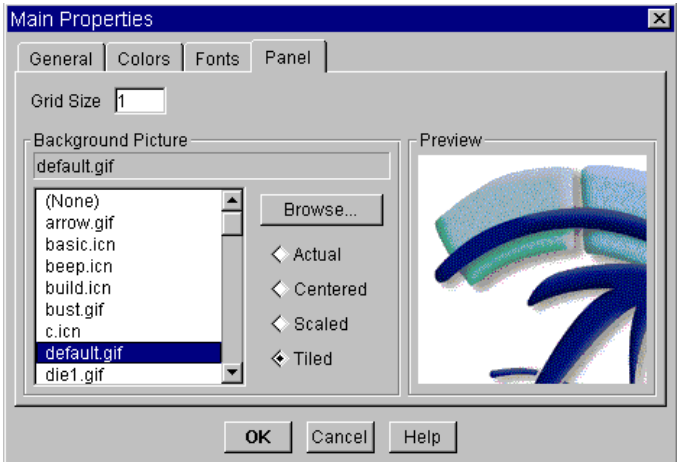


図 10-14. パネルのプロパティを設定する方法

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 10-15 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。

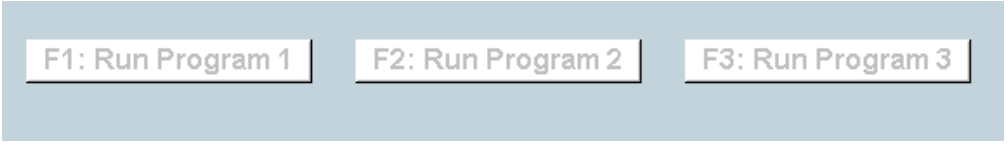


図 10-15. UserFunction を実行するソフトキー

図 10-16 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。

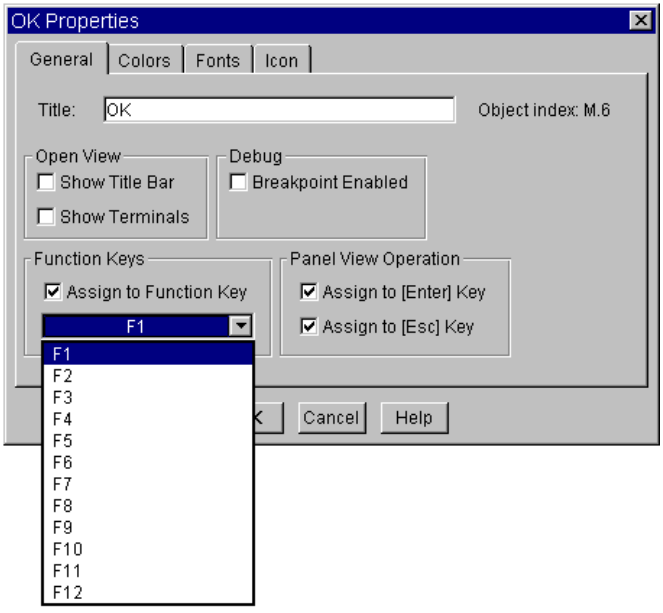


図 10-16. Confirm (OK) オブジェクトをソフトキーとして設定する方法

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。

Ctrl+G	実行または継続 (再開)
Ctrl+P	休止
Ctrl+T	停止

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 10-17 と図 10-18 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。

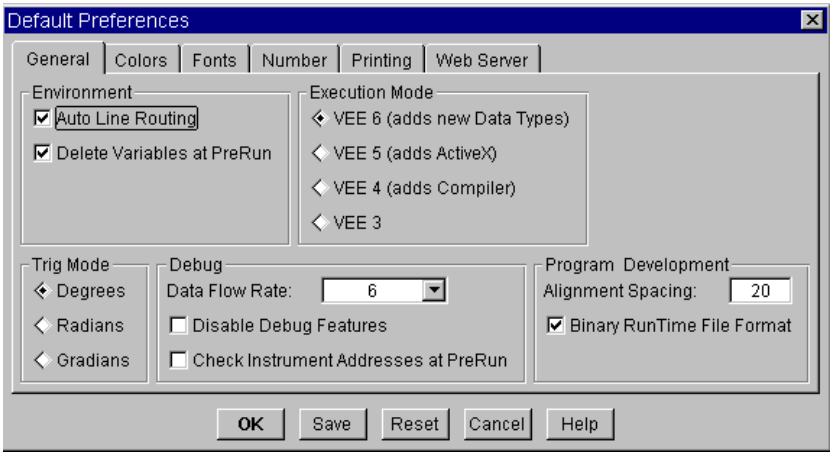


図 10-17. [Default Preferences] ダイアログ・ボックス

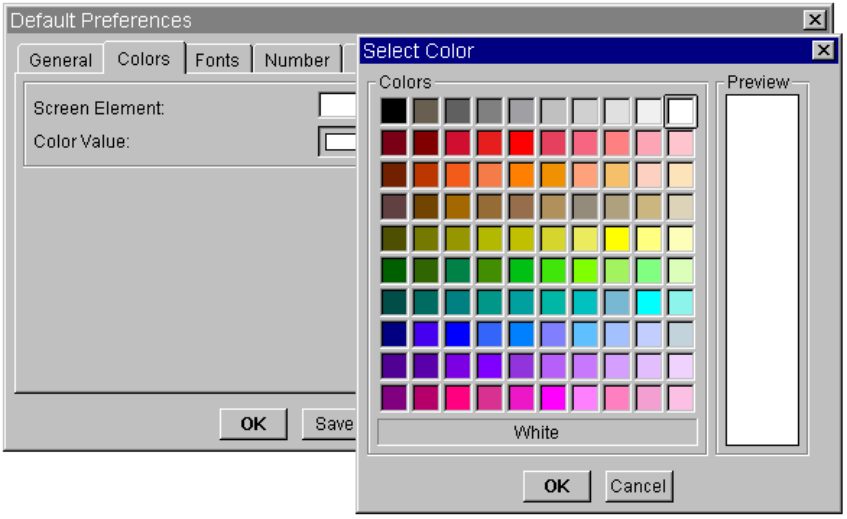


図 10-18. 画面要素の色の選択

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)

不用意にプログラムが変更されるのを防ぐために、またはプログラムの機密を保護するために、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ます。*

メモ

VEE プログラムの実行時バージョンを作成した場合、実行時バージョンは編集できません。詳細ビューを表示できません。したがって、実行時バージョンの作成を開始する前に、元のプログラムのコピーを作成したうえで、次の手順説明に注意深く従ってください。

VEE プログラムの `RunTime` バージョンを作成するには、次の手順に従います。

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. 編集できるコピーを作成するためにプログラムをセーブします。

3. [File] ⇒ [Create RunTime Version...] を選択します。実行時バージョンであることを示す *.vxe 拡張子が自動的に使用されます。

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 [Show Panel on Execute] 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、パネルは表示されなくなります。

UserFunction を複数回呼出す間、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場合があります。例が必要な場合は、次の節を参照してください。

ステータス・パネルを作成する方法

VEE では、複数回のテストや関数の結果を監視するためのステータス・パネルを実装できます。この機能は、図 10-19 に示すように、ShowPanel() と HidePanel() 関数を使用して実装します。詳細は、407 ページの「ステータス・パネルを作成する方法」を参照してください。

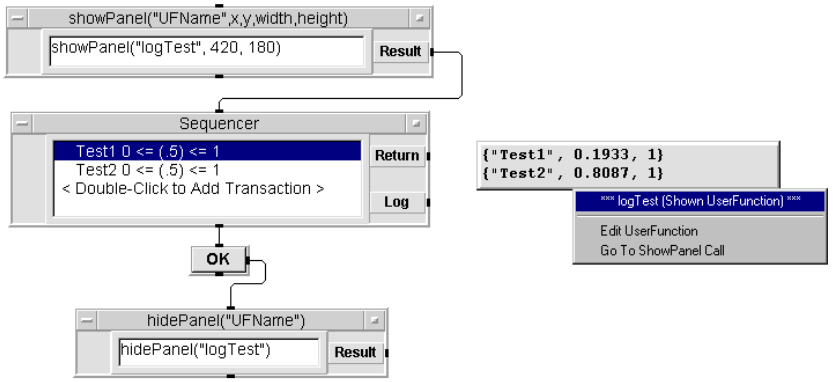


図 10-19. ステータス・パネルを作成する方法

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。これらにより、オペレータ・インタフェースをカスタマイズします。

例題 10-1: 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の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、[Open View] の [Show Title Bar] の選択を解除します。[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。

メモ

[Show Title Bar] をオフにするときにオブジェクト・メニューにアクセスするには、オブジェクトの右上のボタンをクリックします。

メモ

デフォルトでは bitmaps サブディレクトリが使用されますが、任意のディレクトリにあるビットマップも使用できます。

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。

   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。

5. 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。

メモ

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。

   [Properties] ダイアログ・ボックスで [Show Panel on Execute] 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、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。

6. 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 10-20 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。

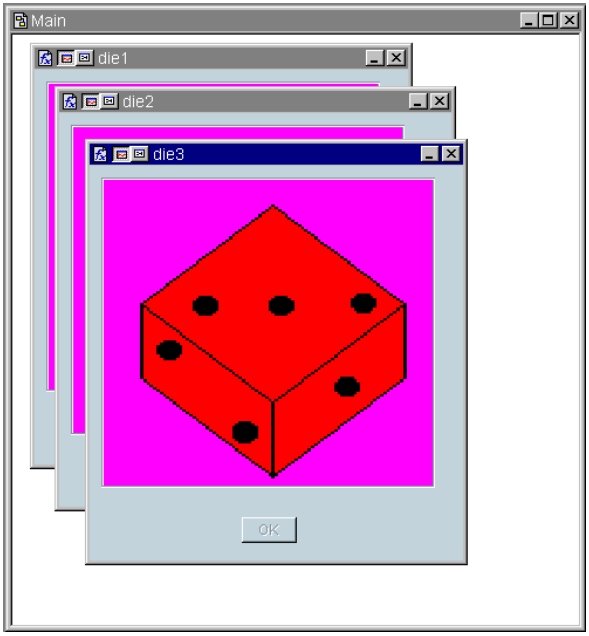


図 10-20. Dice Program の初期段階

3 つの関数から呼出す関数を選択するメニューを作成します。

7. [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、序数を自分に関連付け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。

8. 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。

メモ

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。

9. 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。

10. [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 10-21 に示すように、Call Function Name が die2 に変わることに注目します。

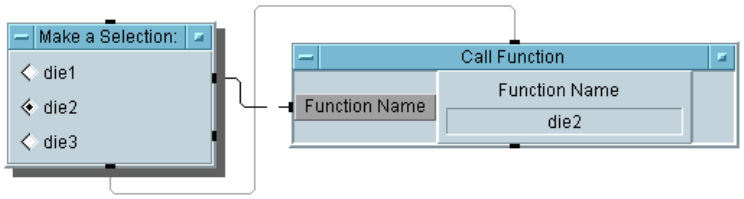


図 10-21. Dice Program(詳細ビュー)

メモ

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。

波線はコントロール・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出しません。

メモ

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。

11. 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。[Edit] ⇒ [Add To Panel] を次に選択します。

12. 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。

13. 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 10-22 のように表示されます。

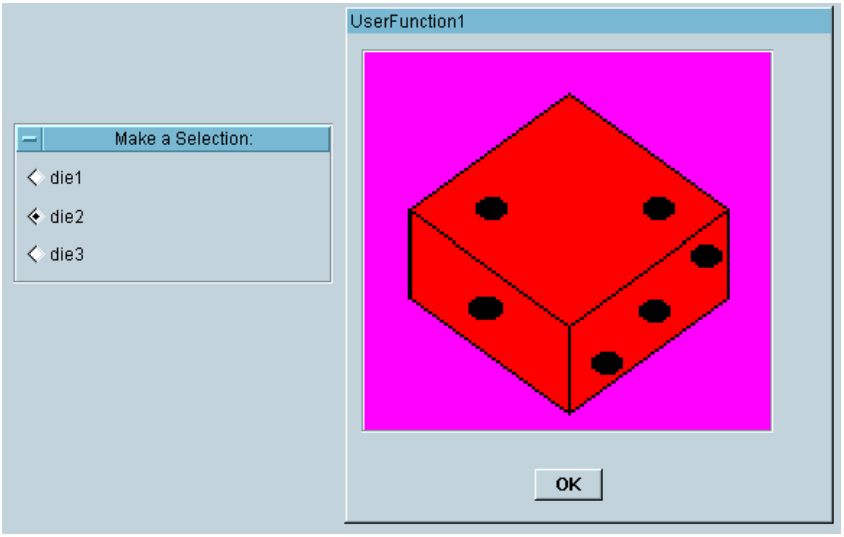


図 10-22. Dice Program(パネル・ビュー)

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 10-2: 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)] ⇒ [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、[Show Panel on Execute] 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するので、注意してください。[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。

   General/Title: [Bitmap Function] に変更します。

Label Justification	[Center Justify] に変更します。
Colors/Object/ Background	[Light Gray] を選択し、[OK] をクリックします。
Fonts/Object/Text:	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。
Appearance/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 10-23 のように表示されます。

図 10-23. Bitmap Function

例題 10-3: 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返し生成する alarm を呼出します。

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。

メモ

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 95、Windows 98、Windows 2000、and Windows NT 4.0 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。

メモ

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。[Open View] で [Show Title Bar] の選択を解除し、[Layout] で [Rectangular] をクリックします。[Limits/High Test] で、[High Text] の横にテキストがあれば削除します。[OK] をクリックします。

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。これにより、Alarm は、常に、高い音域とデフォルト色である赤に設定されます。

   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。

5. [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。

6. [Display] ⇒ [Note Pad] を選択し、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。

7. メイン・ウィンドウに行きます。[Device] ⇒ [Call] をクリックしてから、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 10-24 のように表示されます。

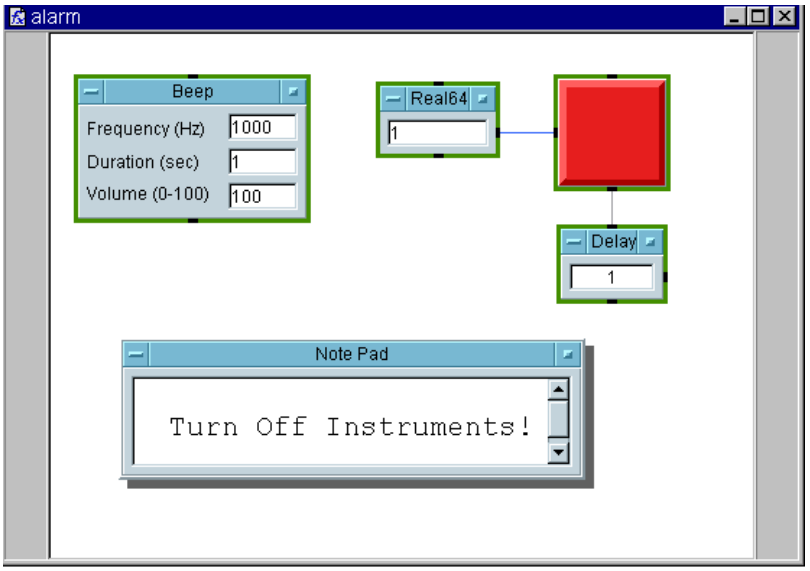


図 10-24. UserFunction alarm (詳細ビュー)

8. 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のサイズと

配置を調整します。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。

Open View/ Show Title Bar	選択を解除します。
Editing/Enabled	選択を解除します。
Fonts/Text size	テキスト・サイズを拡大し、[Font Style: Bold] を選択します。
Fonts	[Automatically Resize Object on Font Change] を選択します。
Appearance/Border	[Raised border] に変更します。

[OK] をクリックし、[Properties] ダイアログ・ボックスを閉じます。

9. [Color Alarm] も [Raised Border] に変更します。

10. UserFunction 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、[Show Panel on Execute] を選択します。[Show Title Bar] の選択を解除します。[Panel Title] を [alarm] に変更します。alarm をアイコン化します。

11. 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。

alarm 関数を繰返し呼出す関数を作成します。

12. [Device] ⇒ [UserFunction] を選択し、UserFunction の名前を warning に変更します。

13. [Flow] ⇒ [Repeat] ⇒ [Until Break] を選択します。

14. [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。

15. [Data] ⇒ [Toggle Control] ⇒ [Check Box] を選択します。[Check Box Properties] ボックスをオープンし、名前を Turn off alarm? に変更し、[Layout] で [Scaled] を選択し、[Initialize at PreRun] を選択して、その値が [0] であることを確認し、名前のフォント・サイズを大きくしてから、[OK] をクリックします。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。

16. [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。

17. [Flow] ⇒ [Repeat] ⇒ [Break] を選択し、図 10-25 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。

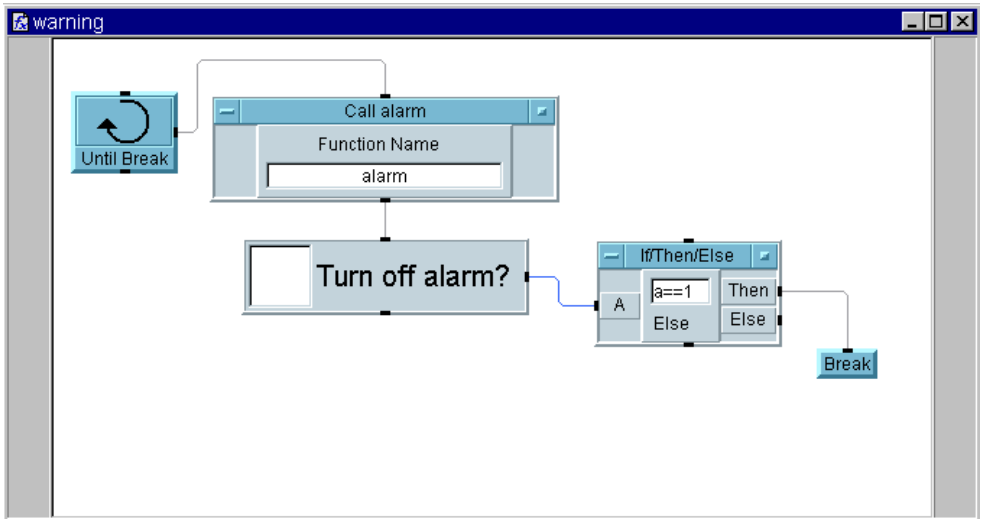


図 10-25. Warning UserFunction (詳細ビュー)

18. 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。

19. warning UserFunction Properties ボックスをオープンし、[Show Panel on Execute] を選択し、[Show Title] の選択を解除します。タイトルはオペレータにとって意味がないためです。[OK] をクリックします。

20. 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。

21. 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、位置の変更はしづらくなります。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 10-26 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 10-26. Warning プログラム

例題 10-4: 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 プログラムに取込むことができます。この例題では、`ProgressBar` コントロールを取込み、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、次のように設定します。[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 10-27 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。

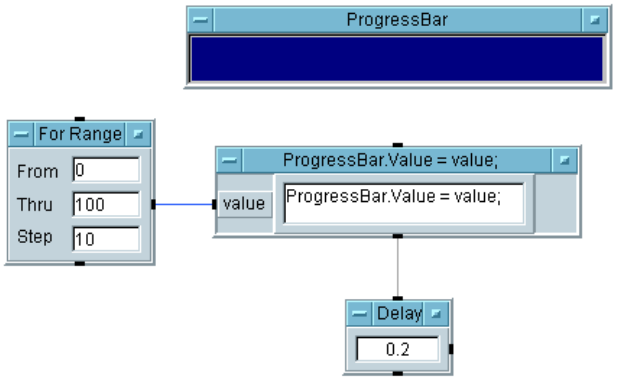


図 10-27. ActiveX コントロール ProgressBar の使用方法

メモ

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 10-28 は、MS Chart からコントロールを取込んだ 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。

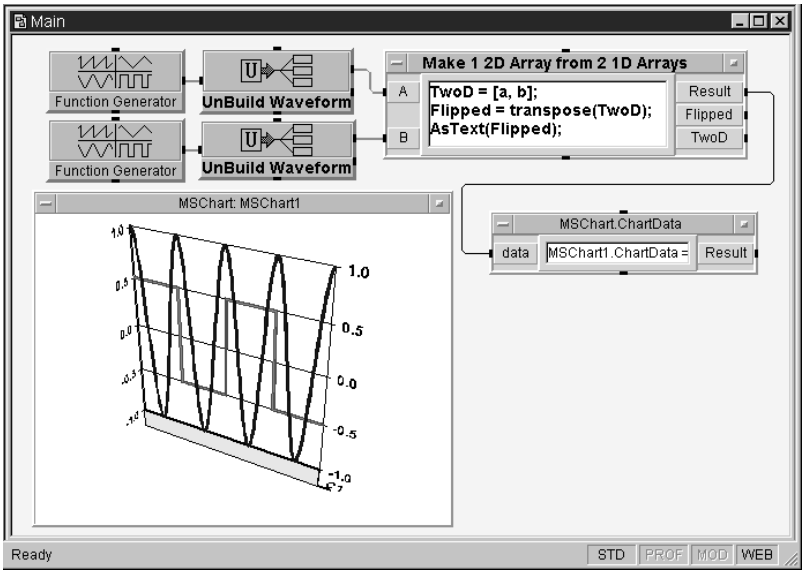


図 10-28. MSChart を使用する ActiveX コントロールの例

例題 10-5: ステータス・パネルを作成する方法

この例題では、dice program の関数を使ってステータス・パネルを作成する方法を習得します。dice program の例題は 391 ページの「メニューを使用する方法」にあります。通常、このパネルは、テストが多数あり、結果が返されたときに結果を表示する場合に、`Sequencer` オブジェクトとともに使用されます。デフォルトの設定を使用する場合は、`0` から `1` までの real 値を返す関数 `random()` を使用します。

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` を使ってテストを設定し、`[FUNCTION:]` フィールドを `[random()]` に置換えます。図 10-29 を参照してください。

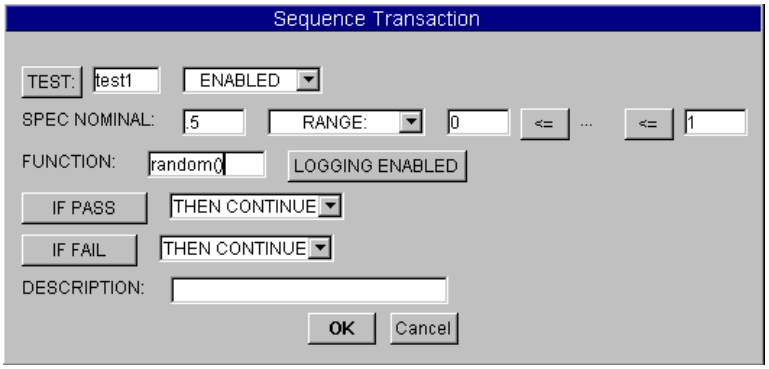


図 10-29. Test1 の設定

2. 2 番目のテストに test2 という名前を付けて、同じように設定します。

3. Sequencer Properties ボックスをオープンし、[Logging] タブを選択し、[Logging Mod] で [Log Each Transaction To: logTest(thisTest)] を選択してから、[OK] をクリックします。

   Sequencer は各トランザクションを実行したとき、thisTest と呼ばれる各テストのレコードを作成します。レコードのフィールドは同じタブで設定できます。これにより、logTest という名前 (または別の名前) の UserFunction を作成できるので、Sequencer は、実行される各トランザクションの末尾にある Record thisTest を使用して、LogTest() UserFunction を呼出します。このようにして、ステータス・パネルを更新できます。

4. [Device] ⇒ [Function & Object Browser] ⇒ [Built-in Functions] ⇒ [Panel] ⇒ [showPanel] を選択し、Sequencer の上に配置します。入力ピンをすべて削除してから、パラメータを "logTest",420,180 と編集し、最後の 2 つのパラメータは無視します。showPanel の Result 出力ピンを Sequencer のシーケンス入力ピンに接続します。成功した場合、ShowPanel は 1 を出力します。

   LogTest は UserFunction の名前です。ほかの 2 つのパラメータは、左上の隅から開始する画面上の X と Y 座標です。これらのパラメータは、UserFunction パネルを表示する場合に、どこに配置するかを VEE に指示するものです。パネルの次元はこの例にはありません。

5. 下に示すように、logTest という名前の UserFunction を作成します。入力ピンを追加します。ログを記録する Record が入力となります。Logging AlphaNumeric をパネル上に配置し、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・ビュー内で、必要に応じて、サイズと配置を調整します。表示とパネルのタイトル・バーの選択を解除します。

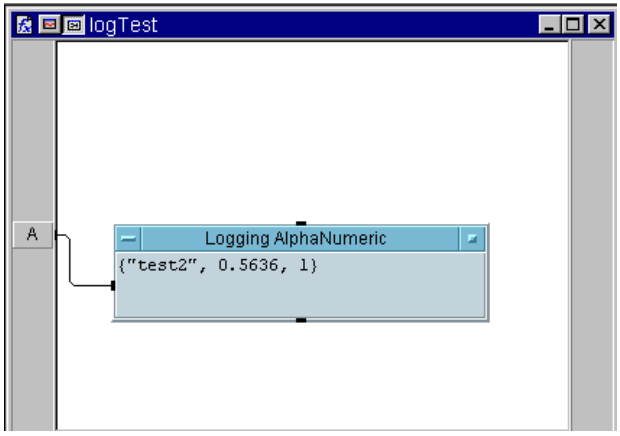


図 10-30. UserFunction LogTest(詳細)

図 10-31 はプログラム実行後のパネル・ビューを示します。これによりデータがどのように表示されるかがわかります。

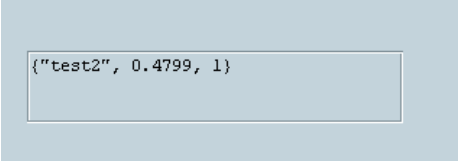


図 10-31. UserFunction LogTest(パネル)

6. Function & Object Browser ボックスで [hidePanel] を選択し、[Flow] ⇒ [Confirm(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 10-32 のようにオブジェクトを接続します。

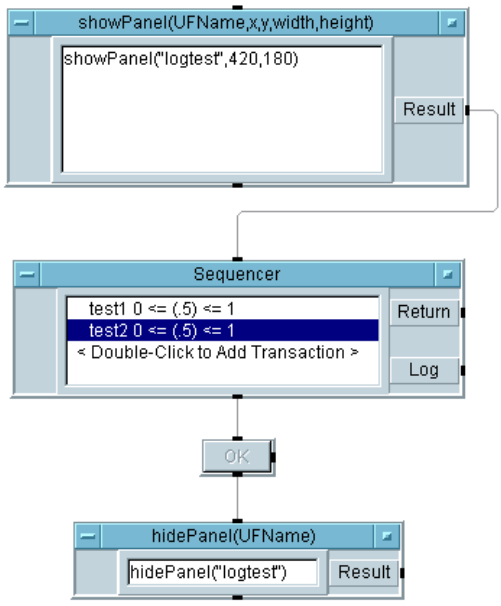


図 10-32. ステータス・パネル・プログラム (実行前)

7. プログラムを実行します。図 10-33 のように表示されます。

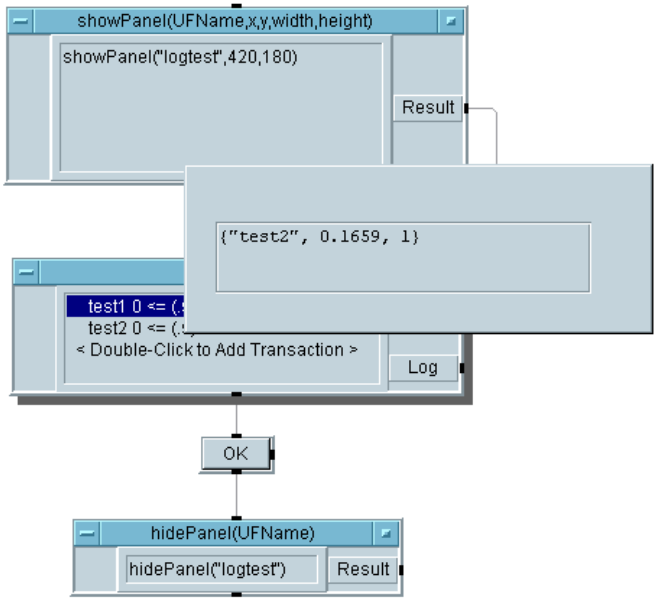


図 10-33. ステータス・パネル・プログラム (実行後)

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 を呼出しません。Sequencer は logging 関数を介して、UserFunction を 2 回呼出し、それぞれの呼出しでパネルが更新されます。次に、オペレータは作業が終わると、[OK] を押すことができ、これによりパネルは表示されなくなります。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ていますが、OK オブジェクトは、プログラムの実行時間を計測するためにプログラム内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・パネルが表示され、Test1 と Test2 の結果を使って更新され、次に表示されなくなるのを確認します。

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、必要なら復習してください。

- 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。

11

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
- PC にあるダイナミック・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
- ほかの言語で記述されたコンパイル済みの関数を UNIX プラットフォームで使用する方法
- VEE コンパイラの使用法
- VEE プロファイラの使用法

平均的な必要時間 :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。

最初の節では、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。PC のダイナミック・リンク・ライブラリ (DLL) についても学びます。次の節では、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。UNIX プラットフォームで、ほかの言語で記述されたコンパイル済みの関数をリンクすることにより、プログラムの各部を最適化する方法も習得します。次に、VEE コンパイラの概要について説明します。さらに、VEE プロファイラについて説明します。

メモ

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、この節の情報を参照してください。この節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 つずつ測定値を受取り、平方根を計算します。VEE では、そのようにしないで、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 11-1 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。

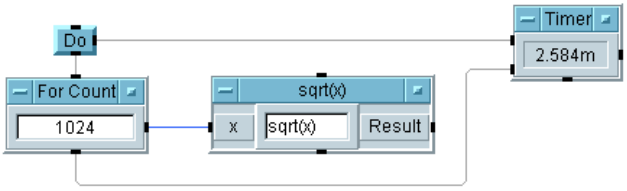


図 11-1. 測定値ごとに平方根を計算する

図 11-2 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 11-2 のプログラムは、図 11-1 のプログラムより 6 倍速く実行されます。この例では、HP Pavilion PC (300MHz) を使用しています。

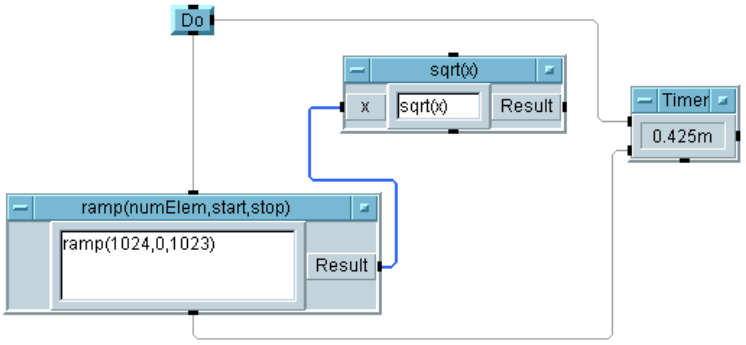


図 11-2. 配列演算を使って平方根を計算する

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。

メモ

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 11-3 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。

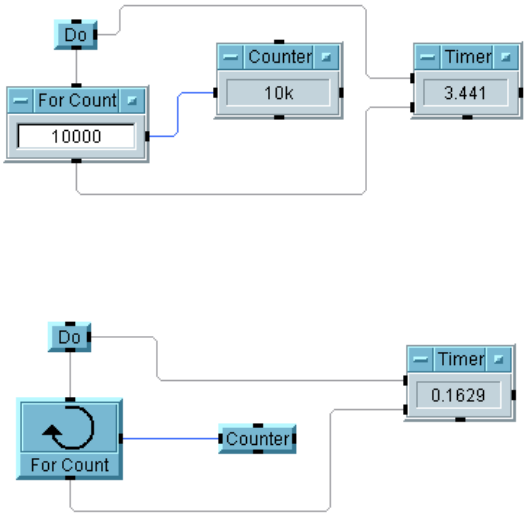


図 11-3. アイコンを使ったプログラムの最適化

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出しをネストします。たとえば、図 11-4 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 11-5 では、`ramp` 関数呼出しが `randomize` 呼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。

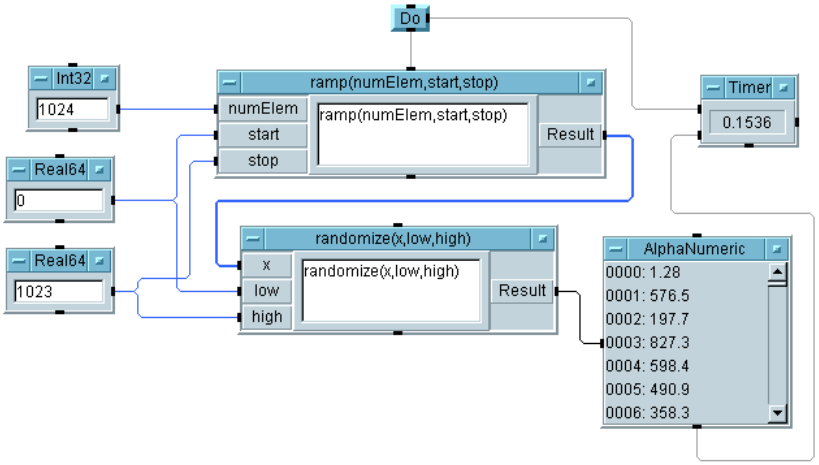


図 11-4. 最適化されていない関数呼出し

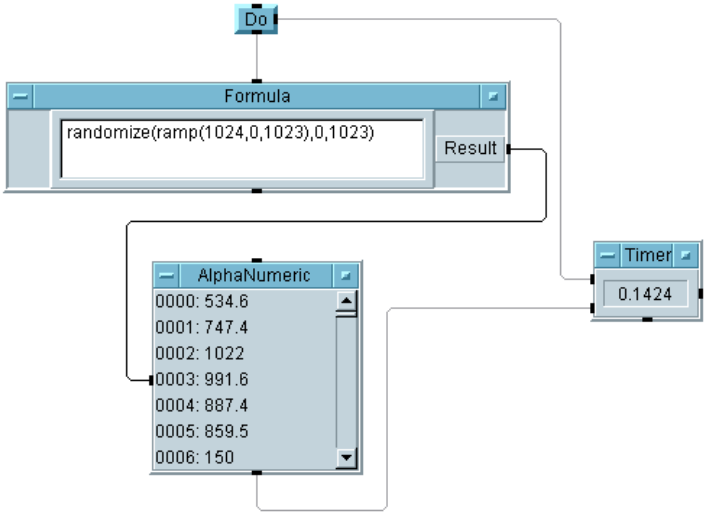


図 11-5. 最適化されている関数呼出し

Agilent 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、VEE コンパイラを使用していることを確認します。詳細は、435 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) を受渡すかわりに、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は、個々のスカラ点をプロットするのではなく、データを配列の収集して、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。必要な場合に AutoScale コントロール・ピンだけを使用し、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には、別々の Call オブジェクトではなく、Sequencer を使用します。

- Sequencer を使用する場合は、Log レコードが必要なトランザクションについてのみログを有効にします。

- 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、色をすばやく切替えるには、[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 VEE 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 (DLL) を使用する方法については、次の節で説明します。UNIX プラットフォームでコンパイル済み関数を使用する方法については、432 ページの「C を使ったコンパイル済み関数 (UNIX)」で説明しています。

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。

メモ

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`、`Execute Program` (オペレーティング・システムへのエスケープ)、または `ActiveX` オートメーション (別のプログラムの呼出し) 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の使用する予定の場合は、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 VEE 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取る場合は、検査するデータ型の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。

- `UserFunction` を呼出す場合とまったく同様に、コンパイル済み関数を呼出したり、設定します。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、その関数を呼出すことができます。

メモ

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 VEE プログラムの一部として使用できます。DLL は、独自に記述した関数をコンパイルして作成したり、購入または Web からダウンロードすることができます。DLL の作成方法については、Microsoft に問合わせてください。

メモ

VEE は、"_cdecl" と "_stdcall" の両方の呼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、_cdecl 呼出し規則を使用しています。Win32 API 呼出しのほとんどは、_stdcall 呼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

この節では、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 11-6 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。

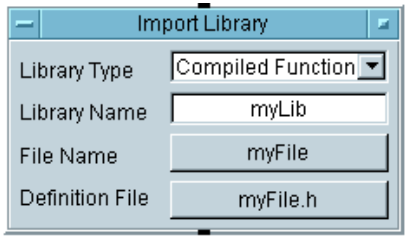


図 11-6. コンパイル済み関数のライブラリのインポート

次のフィールドがあります。

[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。

メモ

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出すことができます。たとえば、図 11-7 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出します。

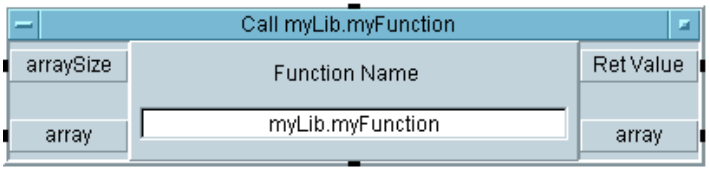


図 11-7. コンパイル済み関数の Call オブジェクト

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。

メモ

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 11-8 のように表示されます。

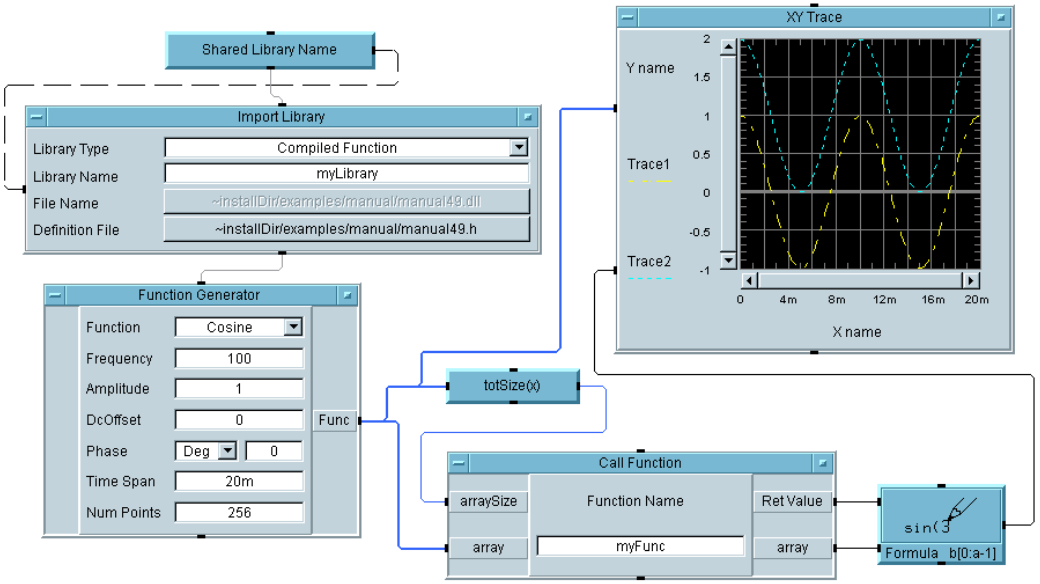


図 11-8. DLL を使ったプログラム (MANUAL49)

Import Library

コンパイル済み関数 Call Function を最初に呼出す前に、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 です。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。
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ため、`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 VEE プラットフォームで使用できるようにする方法です。HP-UX プラットフォームでは、ファイル名拡張子 `*.sl` で示される共有ライブラリが使用されます。Windows 95、Windows 98、Windows 2000、and Windows NT 4.0 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 11-9 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。

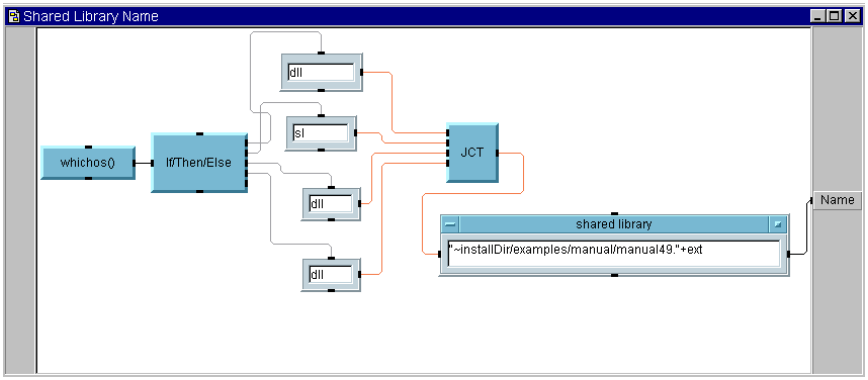


図 11-9. Shared Library Name UserObject

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。

メモ

データベースにデータを送信するなど、混合環境でデータを共有する場合は、To/From Socket を調査します。また、サンプル・ゲーム、Battleship と Euchre は、複数の VEE プロセス間で通信するために、ソケットを広範に使用します。

Execute Program オブジェクトとコンパイル済み関数の比較

コンパイル済み言語プログラムを VEE と統合するために、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を決定する場合は、次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
- 起動時間が長い
- パイプを介した通信 (HP-UX のみ)
- アドレス空間の保護
- 同期実行と非同期実行の選択
- 非同期イベントのサービス
- 安全性が高い (呼出されたプログラムがクラッシュすると、エラー・メッセージが表示される)
- 連続的なデータの取得に便利

コンパイル済み関数

`Import Library` と `Call` オブジェクトを使った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 と共有されたスタックとメモリ空間上での通信
- 同期実行
- 信号と例外はブロックまたはキャッチされない (GPF メッセージなど)
- テキスト言語のコンパイラが必要
- 使い方が複雑。使用上のリスクが大きい (範囲外の配列エラーやメモリの上書きがあると、VEE がクラッシュする)

C を使ったコンパイル済み関数 (UNIX)

ほかの言語で記述されたコンパイル済み関数を使用する場合は、HP-UX プラットフォーム上に共有ライブラリが必要です。UNIX ワークステーションでは、C、C++、Fortran、Pascal で記述されたプログラムを VEE プログラムと動的にリンクできます。Pascal のコンパイル済み関数は、HP 9000, Series 700 ワークステーションでのみサポートされているので、注意してください。

図 11-10 に示すプログラムは、ライブラリをインポートし、C 関数を呼出します。この C 関数は、実数の配列を受取り、配列内の各要素に 1 を加算します。変更された配列は、VEE の `Call Function` オブジェクトの `Array` 端子に返され、配列のサイズは `Ret Value` 端子に返されます。このサンプルは、次の VEE ディレクトリにあります。

< インストール先ディレクトリ >`/examples/manual/manual49.vee`

メモ

ファイル拡張子は次のように使用されます。`".vee"` 拡張子はプログラム、`".c"` 拡張子は C ソース・コードを保持するファイル、`".h"` または `".def"` 拡張子は定義ファイル、`".sl"` 拡張子は共有ライブラリ・ファイルをそれぞれ表します。

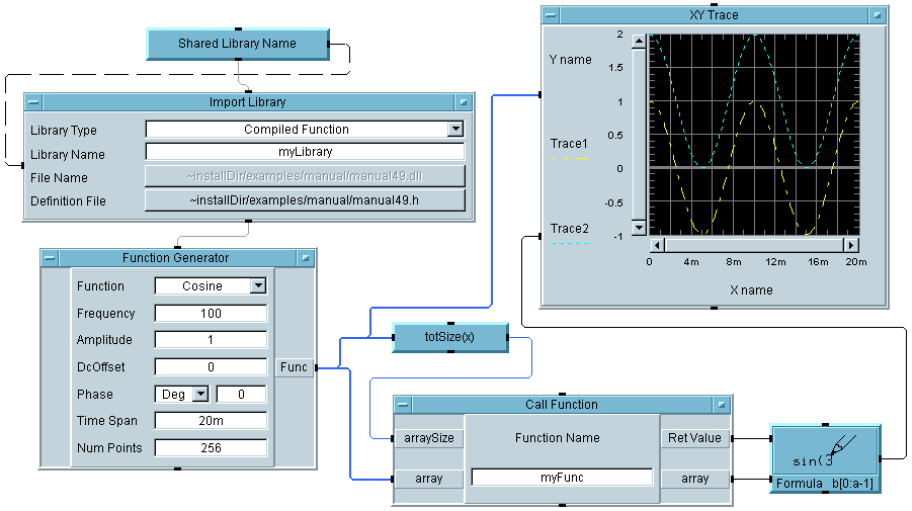


図 11-10. コンパイル済み関数を呼出すプログラム

このプログラムでは、次の点に注意してください。

配列のサイズ

C 関数内の 1 つの変数 (および、対応する `Call` オブジェクト内の 1 つのデータ入力端子) は、配列のサイズを示すために使用されます。`arraySize` 変数は、配列の末尾を超えてデータが書込まれることを防ぐために使用されます。

Call Function オブジェクト

配列は参照によって渡されるため、VEE は、入力ピンと出力ピンの両方を `Call Function` オブジェクトに自動的に作成します。

`arraySize` 変数は値によって渡されるため、VEE は、入力端子だけを作成します。ただし、関数の戻り値は、出力配列のサイズを VEE に返すために使用されます。この手法は、入力配列より要素数の少ない配列を返す必要がある場合に便利です。

C のルーチン	この C ルーチンは関数であり、プロシージャではありません。`Compiled Function` は戻り値を必要とするため、プロシージャと関数を区別する言語を使用する場合は、必ず関数としてルーチンを作成してください。
実行順序	プログラムの中で、`Import Library` オブジェクトは `Call` オブジェクトの前に実行されます。これらの 2 つのオブジェクトの実行順序が明確でない場合は、シーケンス・ピンを使用して、正しい順序を確保してください。
変数の渡し方	参照によって関数に渡されるパラメータ変数 array には、入力端子と出力端子の両方がありますが、値によって渡される変数 `arraySize` には。入力端子だけがあります。
配列の要素数	`Formula` オブジェクトは、`Ret Value` 端子にある配列サイズを使用して、配列の正しい要素数を表示に送信します。
XY Trace	`XY Trace` は、2 つの波形を自動的にスケーリングします。

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。

メモ

*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 11-11 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。

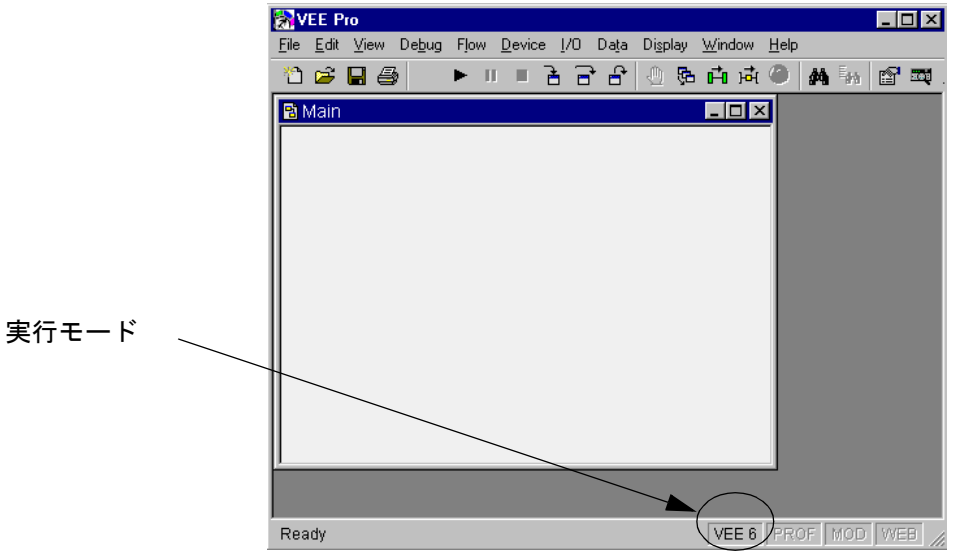


図 11-11. VEE のステータス・バーに表示される実行モード

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 VEE 6.0 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 11-12 に示すツールバーの [Default Preferences] ボタンを押します。

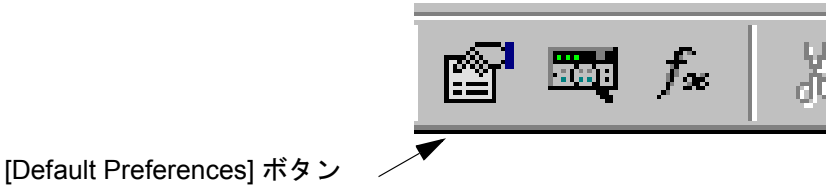


図 11-12. ツールバーの [Default Preferences] ボタン

2. 図 11-13 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。

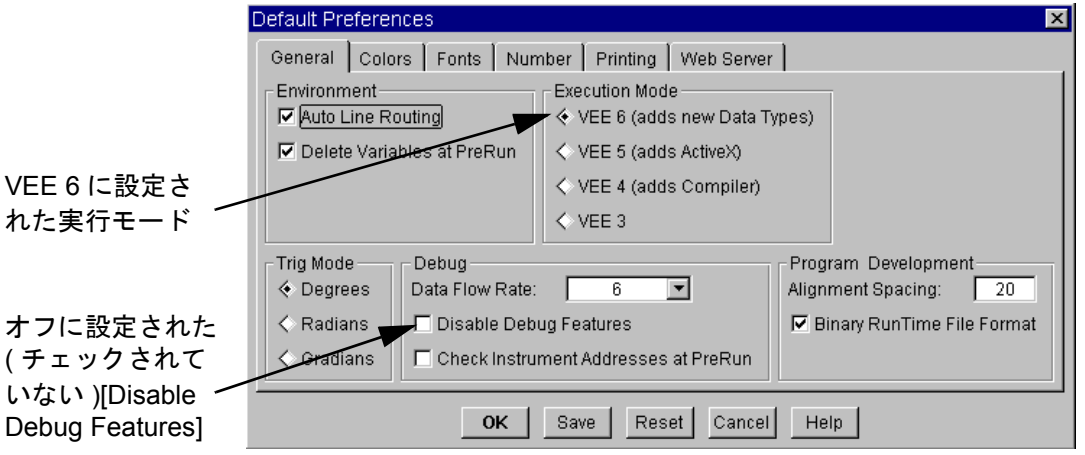


図 11-13. [Default Preferences] ダイアログ・ボックスで実行モードを変更

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。これらの例は、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、`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、HP Pavilion PC (300MHz) を使って Windows 95 上でプログラムが実行され、同時にほかに 2 つの大きなアプリケーションを実行しています。

図 11-14 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。

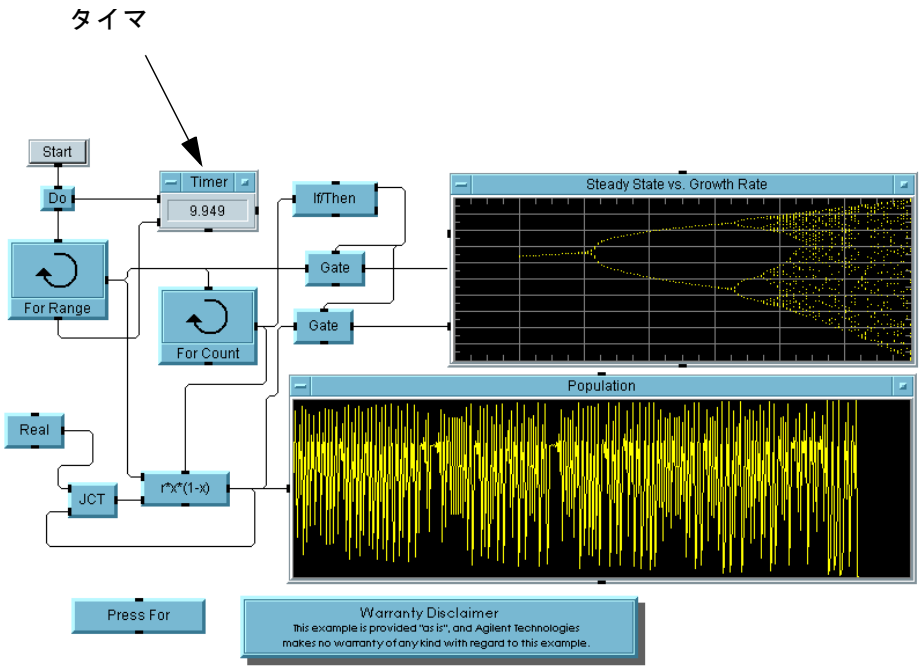


図 11-14. VEE 3 モードで、表示を開いた Chaos.vee

図 11-15 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。

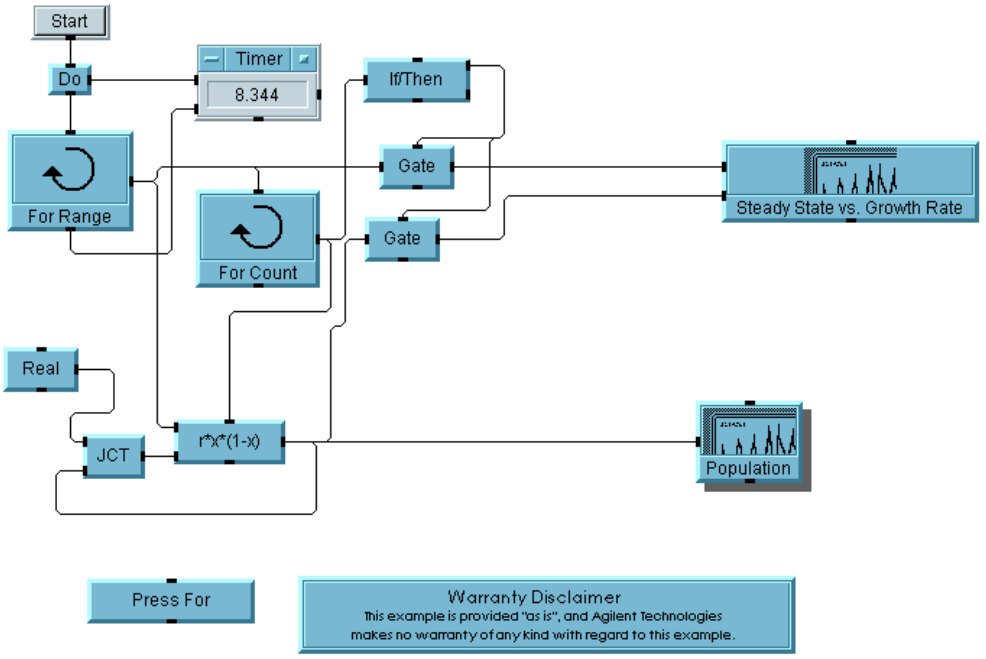


図 11-15. VEE 3 モードで、表示を閉じた Chaos.vee

最後に、図 11-16 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合わせて、最大の処理速度が得られることがわかります。

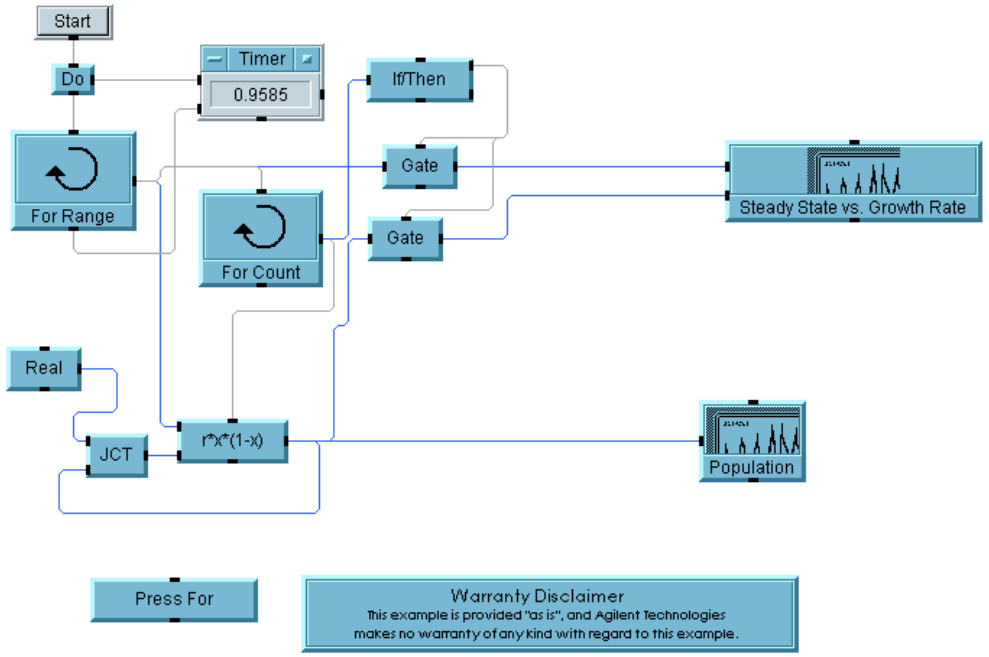


図 11-16. VEE 4 以上のモードで、デバッグを無効にした Chaos.vee

図 11-17 と図 11-18 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。

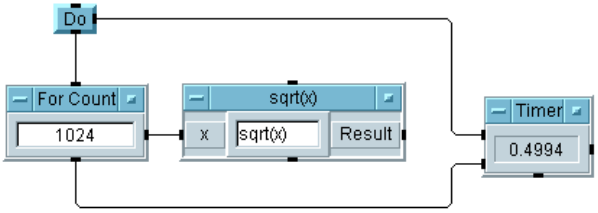


図 11-17. VEE 3 モードでの反復演算ルーチン

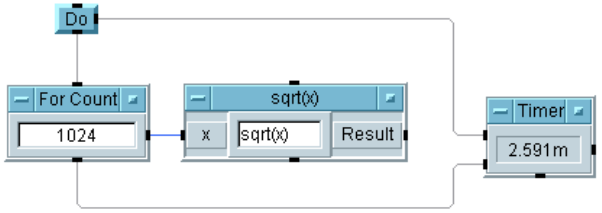


図 11-18. VEE 4 以上のモードでの反復演算ルーチン

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 のプロフェッショナル開発環境における機能の 1 つです。プロファイラには、プログラム内にある `UserFunction` や `UserObject` の実行速度が表示されるため、プログラムの最適化に役立ちます。

プロファイラを使用すると、プログラム内の速度の遅い場所を識別し、この章で説明している手法を適用して、プログラムの速度を向上させることができます。図 11-19 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。プロファイラは、各 `UserObject` と `UserFunction` の実行に要した時間を比較できるように情報を一覧表示します。

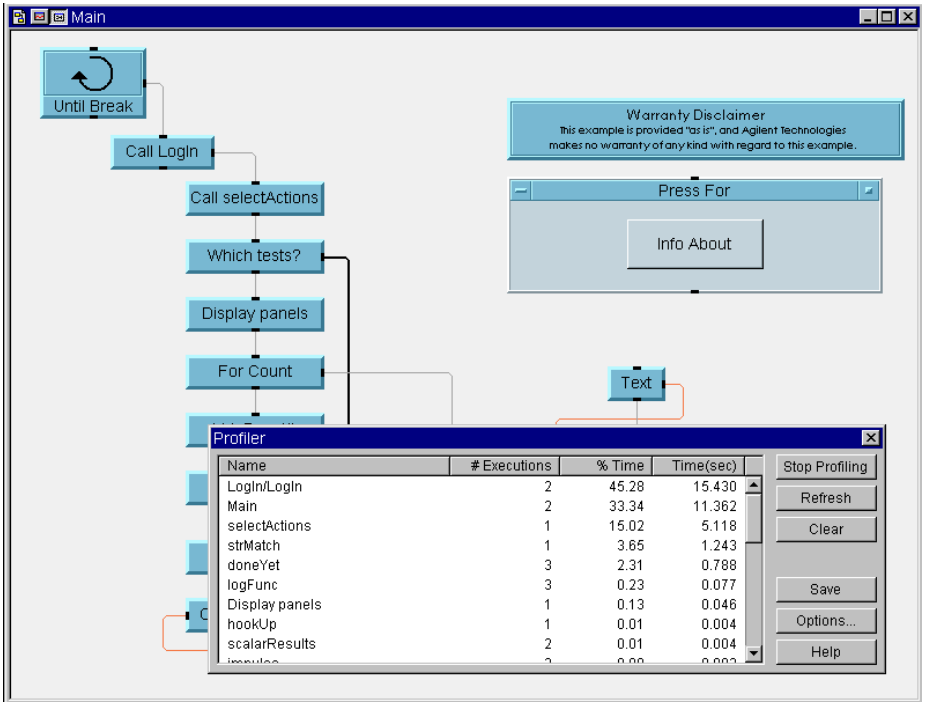


図 11-19. プロファイラの例

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。必要に応じて、各項目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、関数を呼出し、その DLL を削除する。
- ほかの言語で記述されたコンパイル済み関数をHP-UXプラットフォームで使用する方法について説明する。
- VEE 6 実行モードを使用して、または VEE 5、VEE 4、VEE3 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。

12

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

- PC と HP-UX プラットフォームの相違点
- Rocky Mountain Basic プログラムとの通信方法
- VEE ActiveX Automation Server を使用して、ほかのアプリケーションから VEE 関数を呼出す方法
- Web モニタ管理

平均的な必要時間 :2 時間

概要

この章では、オペレーティング・システム間の重要な相違点と、それらの相違点を処理するオブジェクトが VEE でどのように設計されているかについて学びます。次に、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取込むための重要な手法ついて学びます。最後に、Web モニタ管理の主要な概念について学びます。

VEE プログラムは、サポート対象のプラットフォーム間を移動しますが、特定のオペレーティング・システムに固有のオブジェクトもあります。たとえば、PC では、コンパイル済み関数としてダイナミック・リンク・ライブラリ (DLL) が使用され、HP-UX では、共有ライブラリが使用されます。プロセス間通信の場合、PC では ActiveX オートメーションが使用され、HP-UX では名前付きパイプが使用されます。

PC と HP-UX プラットフォーム間の相違点

PC における VEE の使用法と、HP-UX における VEE の使用法の間には、いくつかの相違点があります。

プログラム

VEE のすべての機能をすべてのプラットフォーム上のプログラムで使用できます。ただし、オブジェクトがオペレーティング・システムに依存している場合、そのオブジェクトはそのオペレーティング・システム上でしか実行できません。プログラムを移植する場合は、[Function & Object Broswer] ボックスの System Information オブジェクト (whichOS() または whichPlatform()) を使用すると、これらの相違を考慮できます。これらのオブジェクトは、オペレーティング・システム (OS) またはプラットフォームを出力するため、プログラムは、「OS 依存」のオブジェクトを使用するかどうかを検出できます。「OS 依存」のオブジェクトは、メニュー内のオブジェクト名の末尾に "PC" または "UNIX" が付いています。

名前付きパイプと ActiveX の機能

To/From Named Pipe (UNIX) オブジェクトと、PC の ActiveX Automation は、それぞれのオペレーティング・システムにおいて、別のプログラムまたはアプリケーションと通信するという同じ機能を実現します。

Rocky Mountain Basic

Initialize Rocky Mountain Basic (UNIX) と To/From Rocky Mountain Basic (UNIX) は、HP-UX 上の Rocky Mountain Basic だけと機能するように設計されています。

Execute Program オブジェクト

Execute Program オブジェクトには、HP-UX 用と PC 用の 2 つのバージョンがあります。どちらも、ほかのプログラムまたはアプリケーションを起動するために使用されます。

To/From Stdout、Stderr (UNIX)

これらのオブジェクトは、PC 上で動作しますが、ファイルを使って実装されており、一般のプログラミングにはお勧めできません。これらのオブジェクトは、VEE プログラムを HP-UX プラットフォームから PC に移植する場合にのみ使用してください。

フォントと画面の解像度

PC 上の VEE では、画面の解像度に合わせて見栄えのよいフォント・サイズが選択されます。プログラムの外見を変更するには、[File] ⇒ [Default Preferences] ダイアログ・ボックスを使用します。個々のオブジェクトをカスタマイズするには、オブジェクト・メニューの [Properties] のオプションを使用します。VEE は、必要に応じて、プログラム内に保存されているフォントを移植先のマシンで利用できるフォントに変換します。最良の結果を得るには、画面の解像度とフォントを移植先のマシンと同じように設定したコンピュータでプログラムを構築する必要があります。プログラムを移植する場合は、問題の発生を避けるため、画面の解像度が同じであることを確認してください。詳細は、オンライン・ヘルプのインデックスで、「fonts, using」を参照してください。

データ・ファイル

To File オブジェクトを使って生成される ASCII データ・ファイルは、HP-UX と Windows のどちらのプラットフォームでも From File オブジェクトを使って読取り可能です。バイナリ・ファイルは、HP-UX と Windows でバイト・オーダーが逆のため、プラットフォームにまたがっては機能しません。

Rocky Mountain Basic プログラムを使った通信

VEE には、HP-UX 上で VEE と Rocky Mountain Basic 間の通信を容易にするオブジェクトが用意されています。Series 700 ワークステーション (HP-UX 10.20) では、VEE 6.0 を使用できます。Series 300 と Series 400 ワークステーション (HP-UX 9.0) では、HP VEE 3.1 を使用する必要があります。

Initialize Rocky Mountain Basic オブジェクトの使用法

図 12-1 に示すように、`Initialize Rocky Mountain Basic` オブジェクトにはフィールドが 1 つだけあり、ここで、実行する Rocky Mountain Basic (RMB) プログラムを指定します。

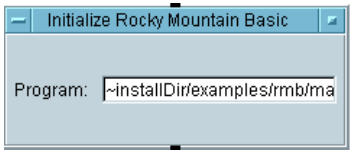


図 12-1. Initialize Rocky Mountain Basic オブジェクト

パス全体とプログラムのオプションを入力します。たとえば、プログラム `~installDir/examples/rmb/man34a.bas` が RMB で保存されているとします。このオブジェクトは、RMB のプロセスを生成し、そのプログラムを実行します。現在の作業ディレクトリからの相対パスを使ってプログラムを指定することもできます。このオブジェクトは、RMB への、または RMB からのデータ・パスを何も提供しません。プログラムを指定するには、`To/From Rocky Mountain Basic` オブジェクトを使用します。1 つの VEE プログラムで複数の `Initialize Rocky Mountain Basic` オブジェクトを使用できます。

メモ

VEE プログラムから RMB プロセスを直接終了する方法はありません。そのかわり、VEE プログラムから一定のデータ値を受信した場合に RMB プログラムで `QUIT` ステートメントを使用します。`Execute Program` オブジェクトを使用すると、`rmbkill` などのシェル・コマンドを使って RMB プロセスを強制終了できます。VEE を終了したとき、まだ接続されている RMB プロセスがあれば、すべて自動的に終了されます。

To/From Rocky Mountain Basic オブジェクトの使用法

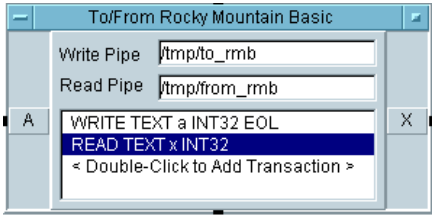


図 12-2. To/From Rocky Mountain Basic オブジェクト

図 12-2 に、`[I/O]` ⇒ `[Rocky Mountain Basic (UNIX)]` にある `To/From Rocky Mountain Basic (UNIX)` オブジェクトを示します。このオブジェクトは、RMB プログラムとの間のデータ転送を可能にします。このオブジェクトは、プロセス間通信のために名前付きパイプを作成して使用します。処理を簡単にするため、VEE は、`READ` トランザクションと `WRITE` トランザクションのために、パイプを 1 つずつ実装します。

図 12-2 に示す `To/From Rocky Mountain Basic (UNIX)` オブジェクトには、2 つのトランザクションがあります。つまり、`Int32` 整数の書込みと、`Int32` 整数の読取りです。デフォルトのパイプを使用することもできますが、`Write Pipe` と `Read Pipe` の新しいパスとファイル名を入力して、独自のパイプを作成することもできます。トランザクションは、VEE のその他のトランザクション・オブジェクトと同様に設定されます。

メモ

`Write Pipe` と `Read Pipe` のフィールドは、コントロール・ピンとしてオブジェクトに追加できます。

次に、Rocky Mountain Basic との通信についてさらに詳細に説明します。

読取りパイプと書込みパイプのデフォルト名

すべての To/From Rocky Mountain Basic オブジェクトでは、読取りパイプと書込みパイプが同じデフォルト名を持ちます。したがって、READ や WRITE に正しいパイプを指定するようにしてください。異なるプログラムに対するパイプは一意な名前を持つようにしてください。RMB プログラムでは、正しいパイプに対して OUTPUT と ENTER ステートメントを指定してください。

自動的に作成されるパイプ

To/From Rocky Mountain Basic が動作する前にパイプが存在していない場合は、次のパイプが自動的に作成されます。

```
/tmp/to_rmb

/tmp/from_rmb
```

ただし、あらかじめパイプが存在している方が、プログラムはすばやく実行されます。

追加のパイプの作成

追加のパイプを作成するには、オペレーティング・システム・コマンド mknod を使用します。

パイプを開くおよび閉じる

Rocky Mountain Basic のパイプ (単純な名前付きパイプ) は、PreRun 後、パイプに対する最初の READ または WRITE トランザクションが動作したときに開かれます。すべての名前付きパイプは、PostRun 時に閉じられます。PreRun と PostRun についての詳細は、『*VEE Pro Advanced Techniques*』を参照してください。EXECUTE CLOSE READ PIPE と EXECUTE CLOSE WRITE PIPE トランザクションを使用すると、いつでもパイプを閉じることができます。

パイプを持つトランザクションの構築

名前付きパイプの動作により、既知の、または容易に解析できるデータ・ブロックを送信するトランザクションの構築はたいへん簡単です。たとえば、文字列を送信する場合は、送信するブロックの最大長を判定し、それより短い文字列には空白を補います。これにより、パイプから利用できる以上のデータを読取ろうとしたり、パイプ内に不要なデータを残してしまう問題を回避できます。

DATA READY トランザクションの使用

データが使用可能になるまで READ トランザクションが停止してしまうことをできるだけ防ぐには、それぞれの To/FromRocky Mountain Basic オブジェクト内で READ IOSTATUS DATA READY トランザクションを使用します。このトランザクションは、読取り対象のデータが少なくとも 1 バイトある場合は 1 を返し、まったくない場合は 0 を返します。読取りパイプにある使用可能なすべてのデータを読取るには、READ ... ARRAY ID TO END:(*) トランザクションを使用します。

ディスクのない環境内での実行

ディスクのない環境で実行する場合は、必ず一意な名前付きパイプとの間で WRITE と READ を使用してください。そうでない場合は、ディスクのない同じクラスタ上のいくつかのワークステーションが同じ名前付きパイプにアクセスしようとして、競合の問題が発生する可能性があります。

図 12-3 に、RMB プログラムとの通信をセットアップする方法を示します。このプログラムは HP-UX プラットフォームだけにあり、examples/rmb サブディレクトリの manual34.vee に収録されています。

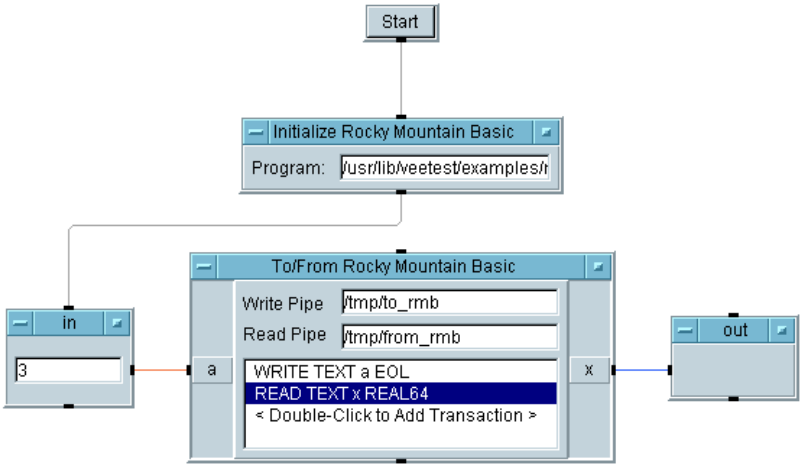


図 12-3. Rocky Mountain Basic との通信

Callable VEE ActiveX Automation Server

Windows 95、Windows 98、Windows 2000、and Windows NT 4.0 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 オブジェクトを取込むことができます。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取込きます。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、オートメーション・サーバを呼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。

メモ

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。Callable VEE ActiveX Automation Server の呼出し元となる Visual Basic や C などの環境により、呼出し方法が決まります。Callable VEE ActiveX Automation Server には、その呼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。

メモ

Callable VEE ActiveX Automation Server は、VEE 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。この節では、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 95, Windows 98, Windows NT 4.0, or Windows 2000)、MS Office、MS Internet Explorer を使用します。図 12-4 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。

メモ

この例では、PC 画面ダンプを使用します。VEE は、HP-UX 上でも Web に対応します。

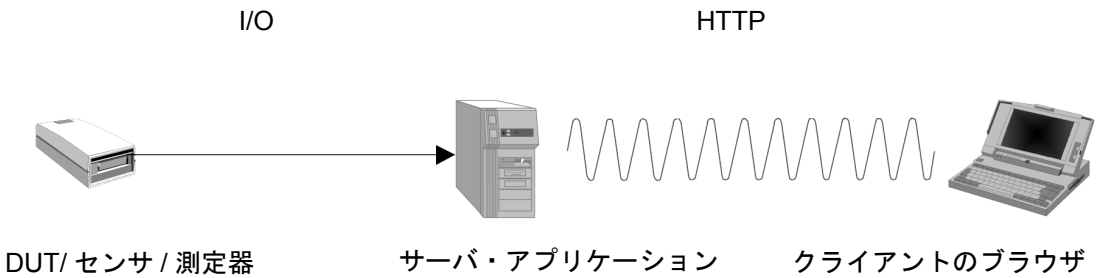


図 12-4. Web 測定アプリケーションのモデル

図 12-4 には、次の情報の流れが示されています。

- *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、VIX、MXI、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 や Netscape Navigator 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、Agilent Technologies 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、JBscript 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 12-5 に図示します。

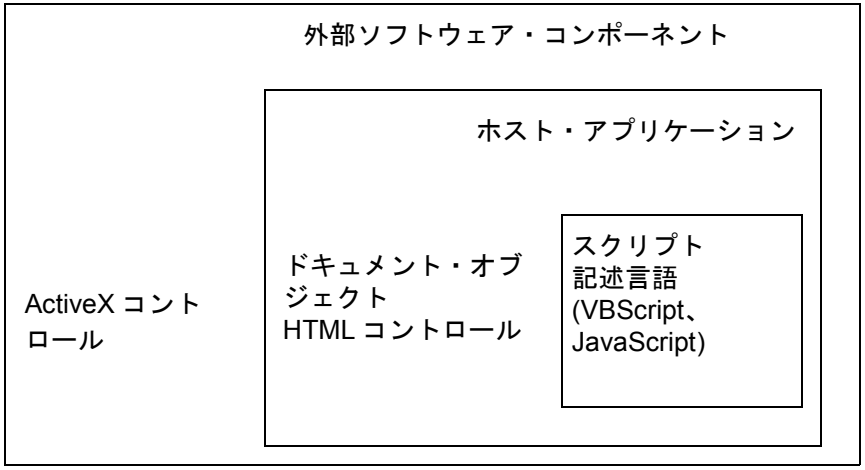


図 12-5. スクリプト記述言語ホスト・モデル

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ある必要があります。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。この節では、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 12-6 に示すように、[Web Server] ダイアログ・ボックスが表示されます。

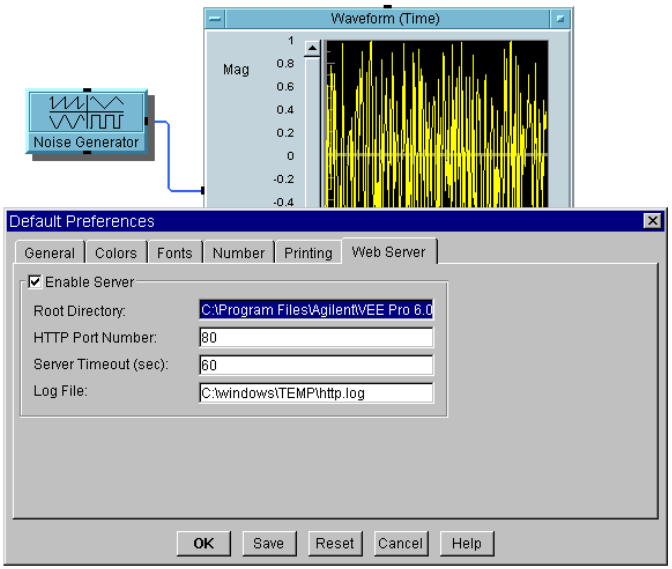


図 12-6. [Default Preferences] の [Web Server] ダイアログ・ボックス

[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。

[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。Windows 95、Windows 98、Windows 2000、and Windows NT 4.0 において、VEE Pro 6.0 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 6.0\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 6.0\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。

[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。
- HP-UX では、1024 より大きいポート番号だけを指定できます。デフォルトのポート番号は 8080 です。
- 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い、ログ・ファイルがまだ存在しない場合は、それが作成されます。

[Save]

表示されている属性を永続的な選択として保存します。これで、Windows では `vee.rc` ファイルに、HP-UX では `.veerc` ファイルに設定値が保存されます。

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。

メモ

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。

メモ

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。

ホスト名

(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。

ポート

(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。

ファイル	(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。
コマンドとパラメータ	(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。

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。

VEE ウィンドウ全体の表示	`ViewVEE` 例 : `http://` *ホスト名* /ViewVEE
メイン VEE プログラムのパネル・ビュー	`ViewMainPanel` 例 : `http://` *ホスト名* `/ViewMainPanel`

メイン VEE プログラムの詳細ビュー	`ViewMainDetail`
	
	例 :
	`http://` *ホスト名* `/ViewMainDetail`

実行時の VEE 実行ウィンドウ	`ViewExecWindow`
	
	ó·:
	`http://` *ホスト名* `/ViewExecWindow`

パネル・ビュー内の VEE UserFunction	`http://` *ホスト名* /ViewPanel? <*UserFunction 名* > 例 (`UserFunction` が `AddNoise` の場合): `http://` *ホスト名* /ViewPanel? AddNoise

UserFunction の詳細ビュー	`http://` *ホスト名* /ViewDetail? `UserFunctionName` 例 (`UserFunction` が `AddNoise` の場合): `http://` *ホスト名* /ViewPanel? AddNoise
現在のプログラムのエラー・ウィンドウ	`http://` *ホスト名* /ViewError
URL で使用できるコマンドの一覧表示	`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 12-7 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。

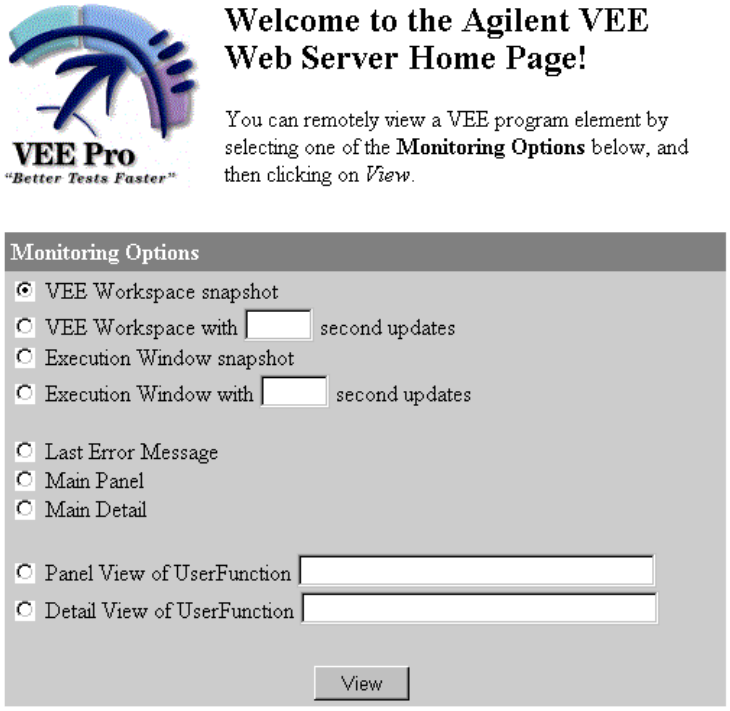


図 12-7. デフォルトの Index.html ページ

メモ

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 12-7 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 12-7 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。

Lab 12-1: 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、-466 ページの図 12-7 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 12-8 に示すビューが表示されます。

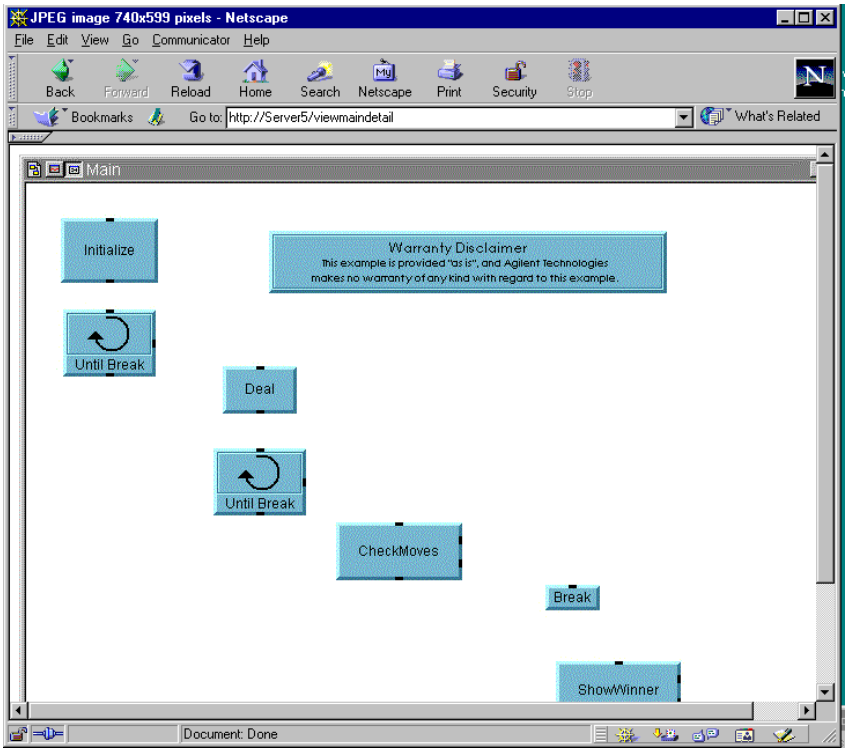


図 12-8. ブラウザに表示された Solitaire.vee メイン・プログラム

図 12-8 に、VEE 内のメイン・プログラムを示します。

メモ

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 12-9 に示すエラー・メッセージが表示されます。

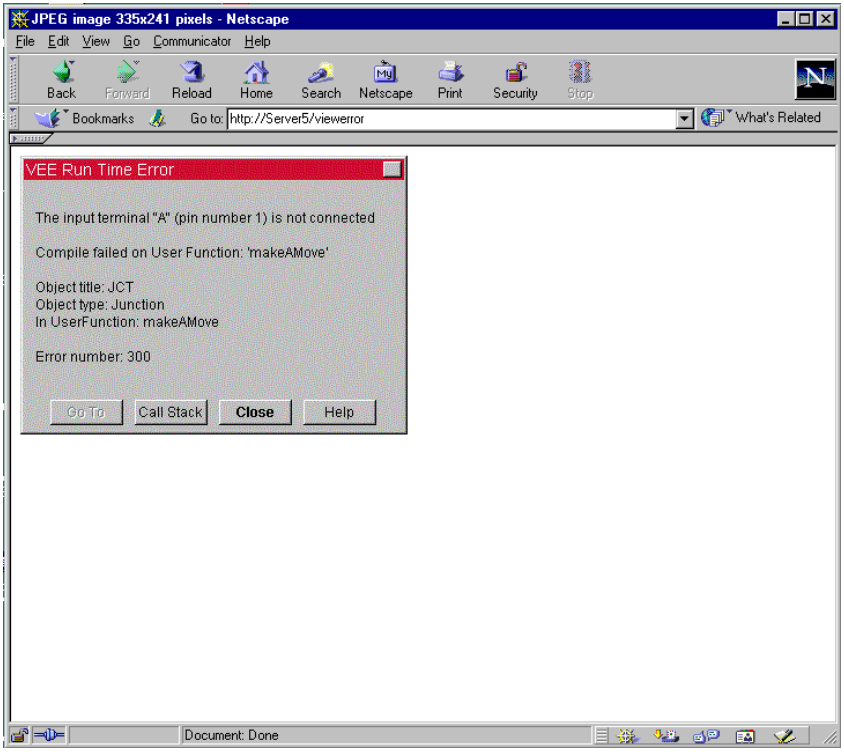


図 12-9. ブラウザを使った VEE エラー・メッセージの表示

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 12-10 に示すように、ブラウザには、`UserFunction makeAMove` が表示されます。

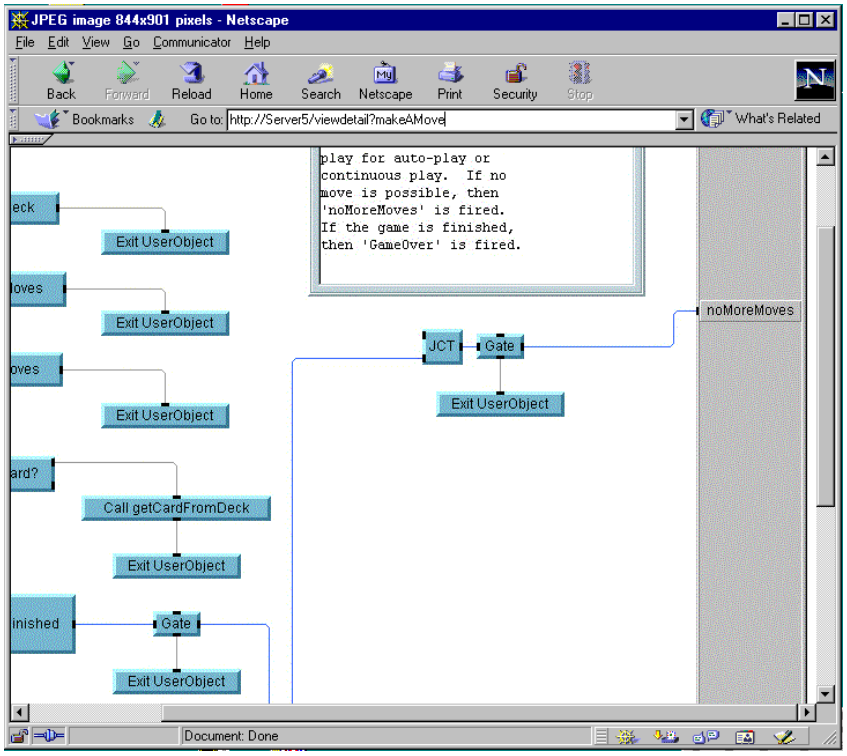


図 12-10. ブラウザに表示された UserFunction の詳細ビュー

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 12-10 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したり、合同して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 とおりあります。

1. 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、388 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 12-11 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。

メモ

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。

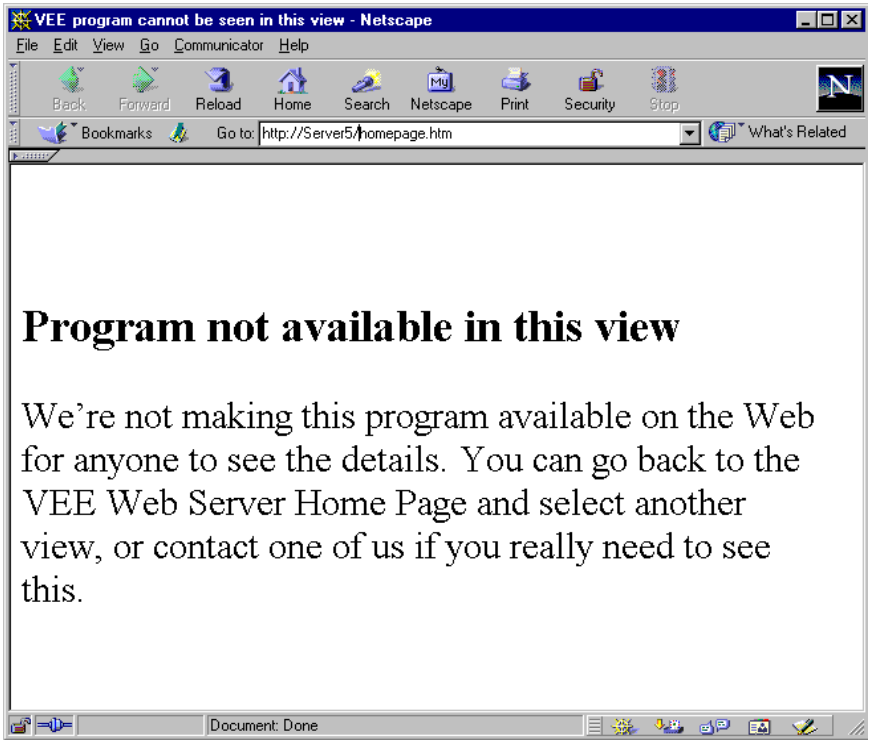


図 12-11. VEE プログラムのかわりに表示する HTML メッセージの例

`*.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できます。図 12-12 に、パスワードによる保護の例を示します。

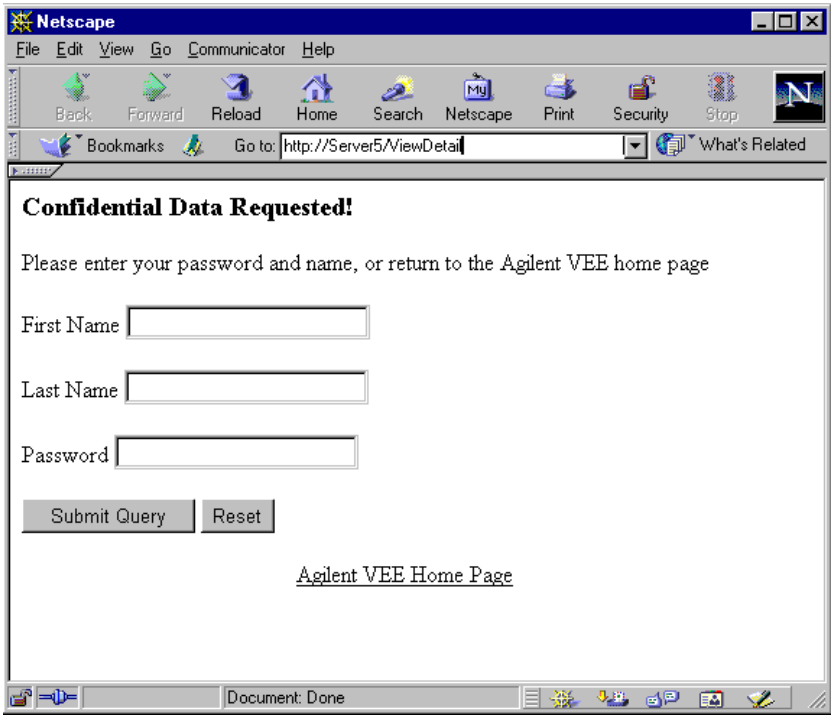


図 12-12. パスワード・ウィンドウの例

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。

- PC と HP-UX プラットフォームの重要な相違点と、プログラムを移植する際に発生する問題について説明する。
- Rocky Mountain Basic プログラムを呼出し、そのプログラムと通信する方法について説明する。
- 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キー・ポイントについて説明する。

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 ポンドだとします。

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

メモ

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 A-1 に、練習「りんごのかご入れ」の解答例を示します。

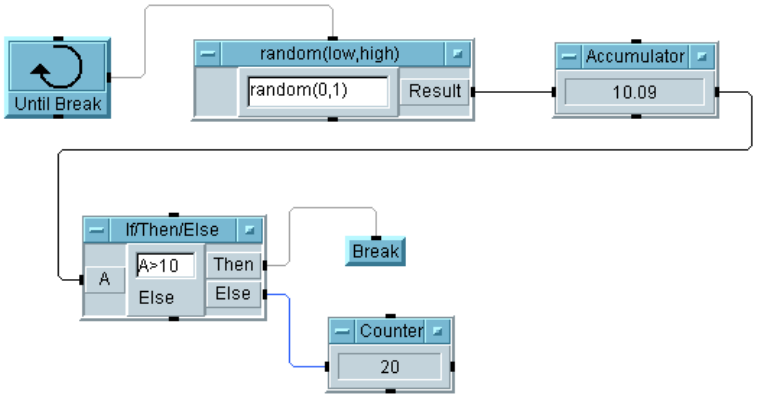


図 A-1. 「りんごのかご入れ」解答 1

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 A-2 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 A-2 に、もう少しオブジェクトを使った解答例を示します。

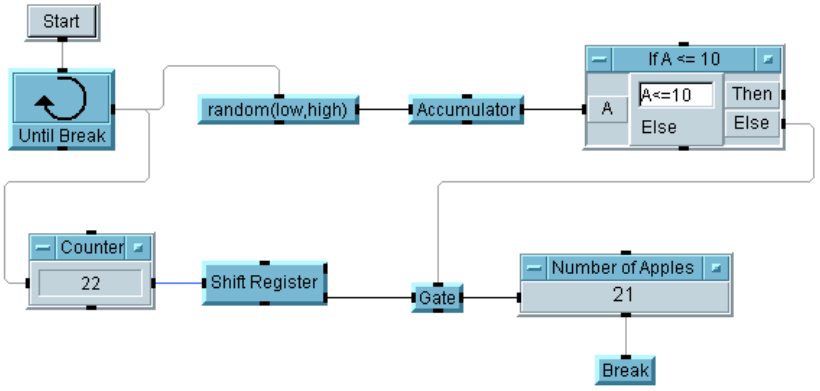


図 A-2. 「りんごのかご入れ」解答 2

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 A-3 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。

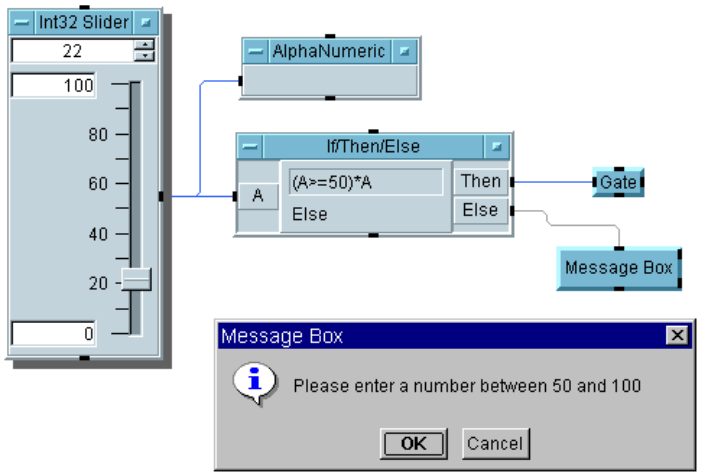


図 A-3. 「数の判定」ステップ 1 (ポップアップが表示されている)

「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 A-4 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。

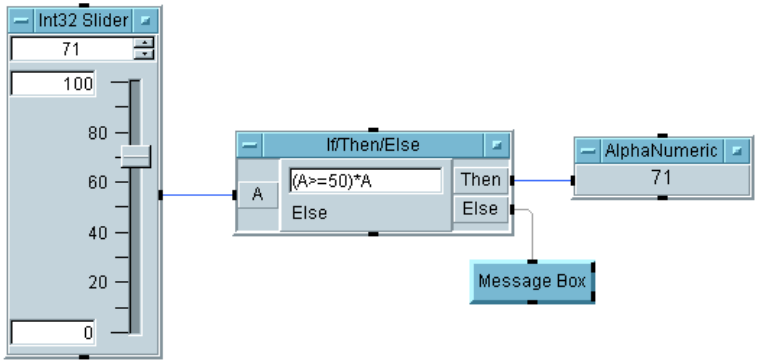


図 A-4. 「数の判定」ステップ 2

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。

- Gate の削除 : `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` が `Else` ピンに出力されます。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 A-5 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。

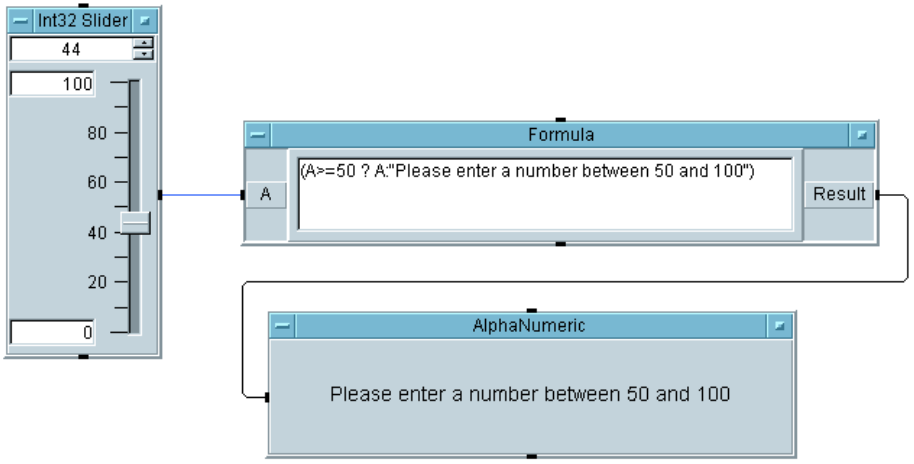


図 A-5. 「数の判定」ステップ 3

メモ

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 A-6 に、練習「乱数の収集」の解答例を示します。

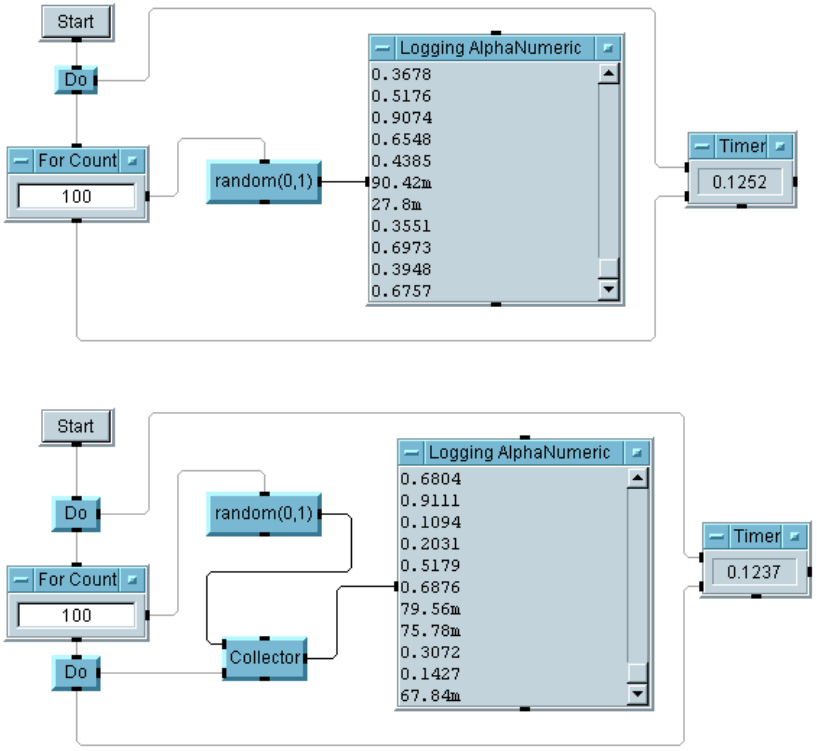


図 A-6. 「乱数の収集」

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 A-7 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。

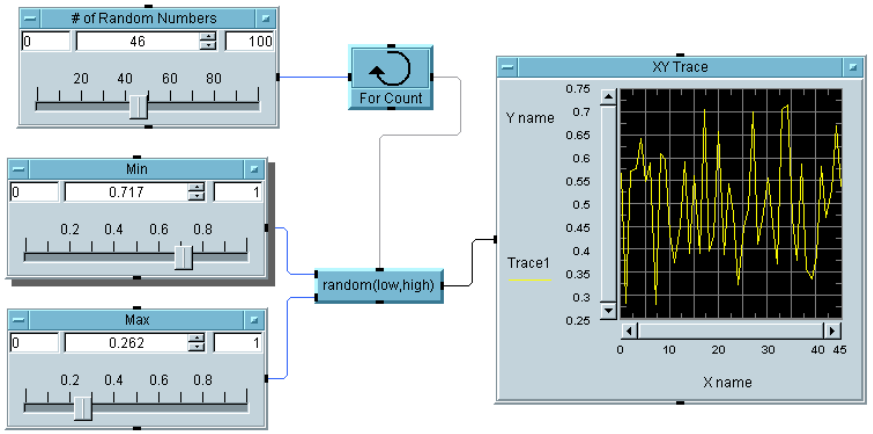


図 A-7. 「乱数生成プログラム」ステップ 1

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 A-8 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。

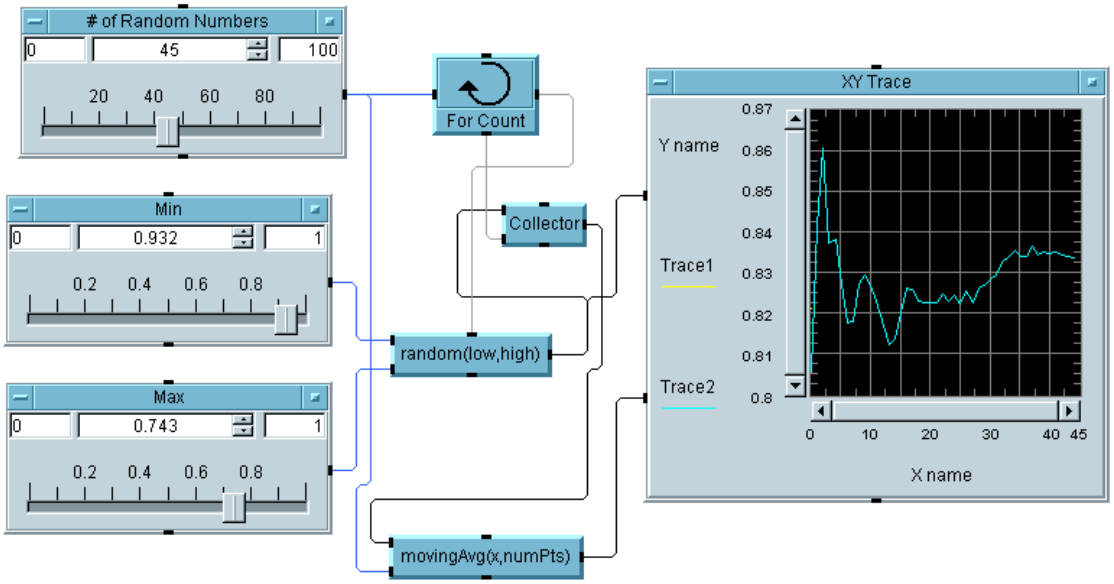


図 A-8. 「乱数生成プログラム」ステップ 2

- 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、ただの点、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 A-9 に、ステップ 1 の解答例を示します。

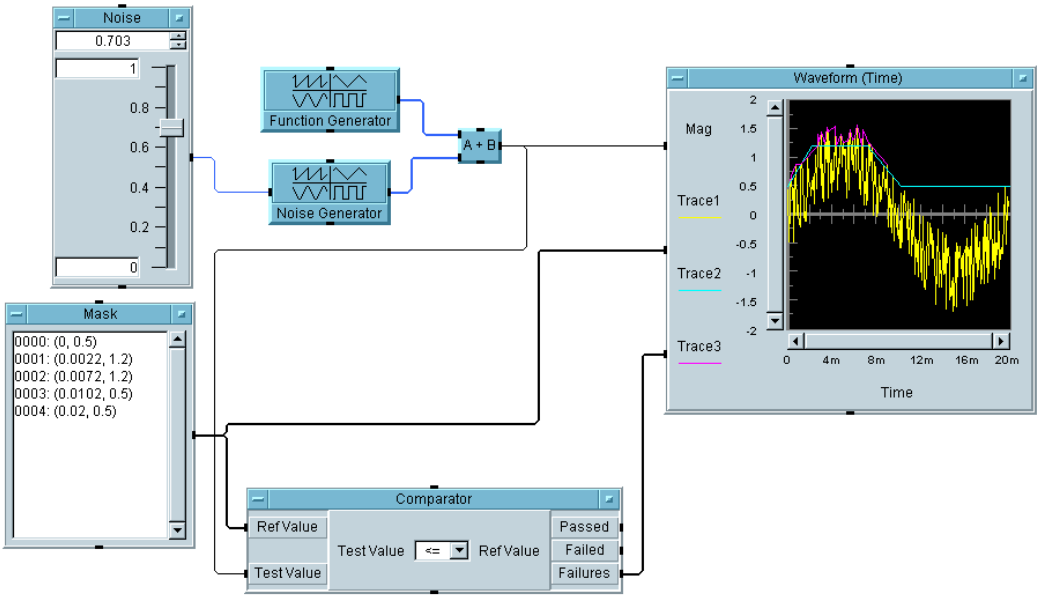


図 A-9. 「マスク・テスト」ステップ 1

「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 A-10 に、ステップ 2 の解答例を示します。

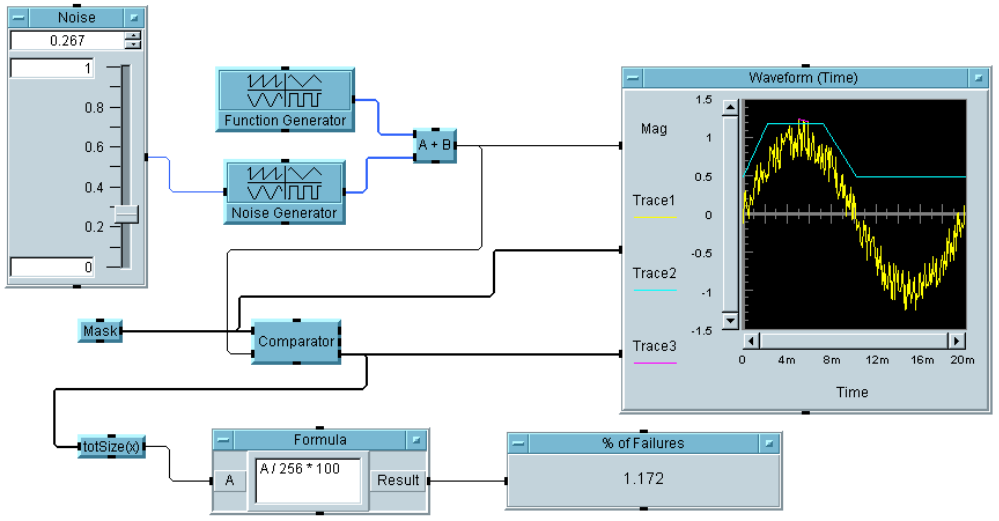


図 A-10.「マスク・テスト」ステップ 2

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。座標の組をコ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するため、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。これは失敗の数を保持するため、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 A-11 に、練習「文字列とグローバルの操作」の解答例を示します。

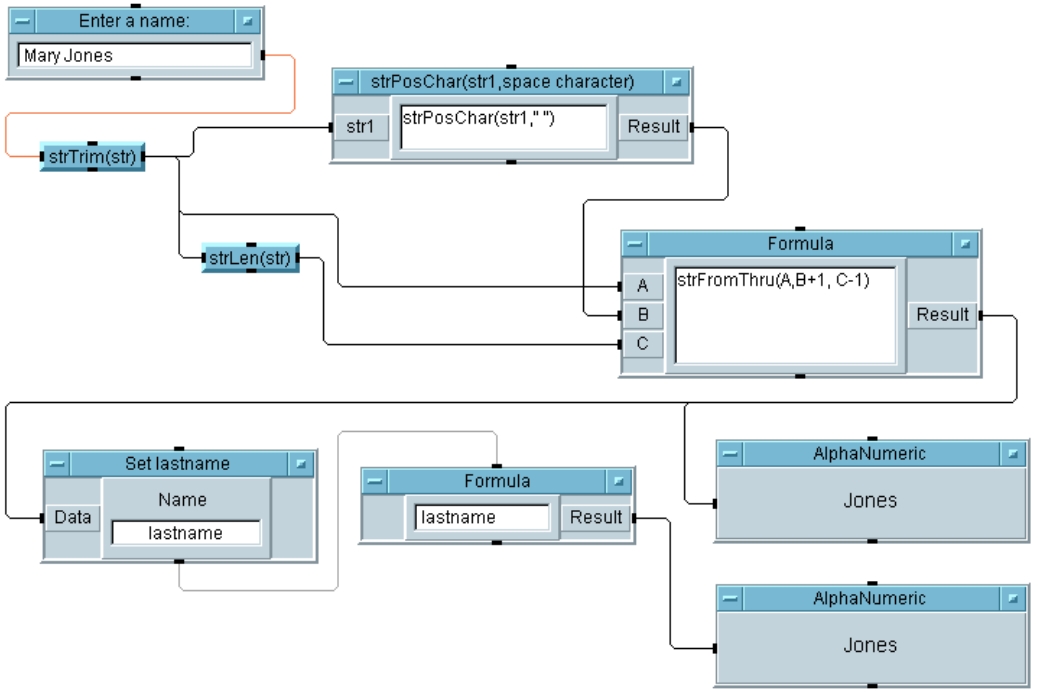


図 A-11. 「文字列とグローバル変数の操作」

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 A-12 に、ステップ 1 の解答例を示します。

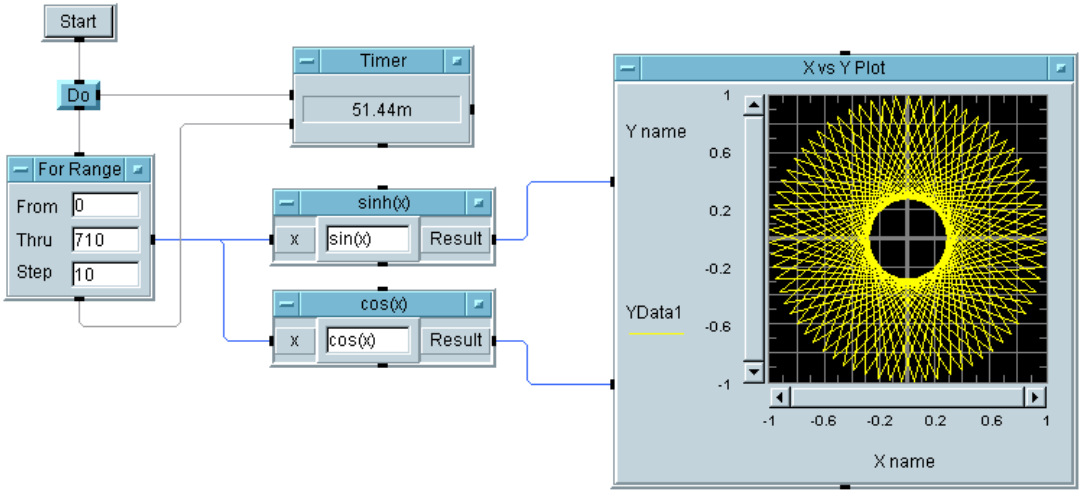


図 A-12. 「VEE プログラムの最適化」ステップ 1

「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 A-13 に、ステップ 2 の解答例を示します。

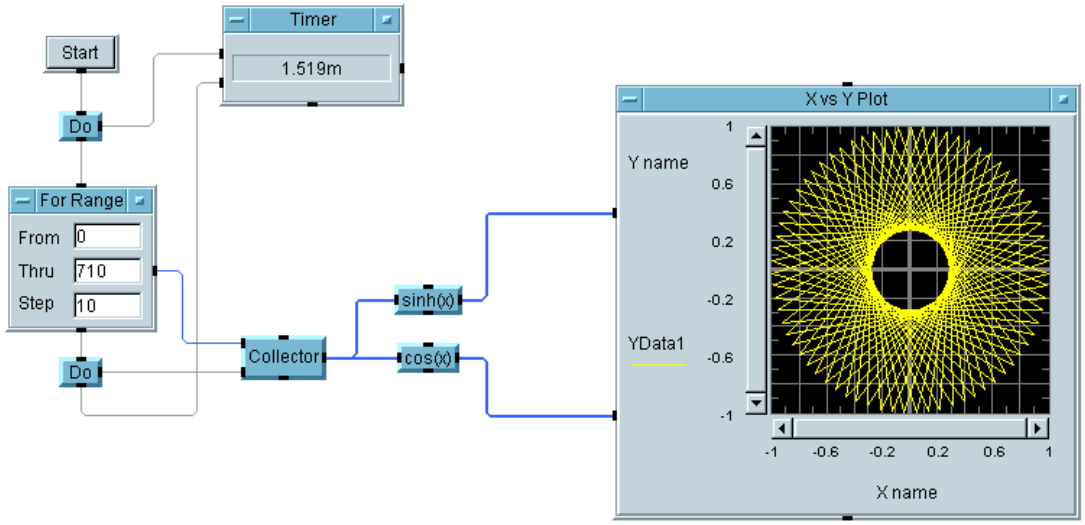


図 A-13. 「VEE プログラムの最適化」ステップ 2

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。

メモ

UserObject の中で仮想ソースを使用しないでください。UserObject 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 A-14 に、「不規則ノイズ UserObject」の解答例を示します。

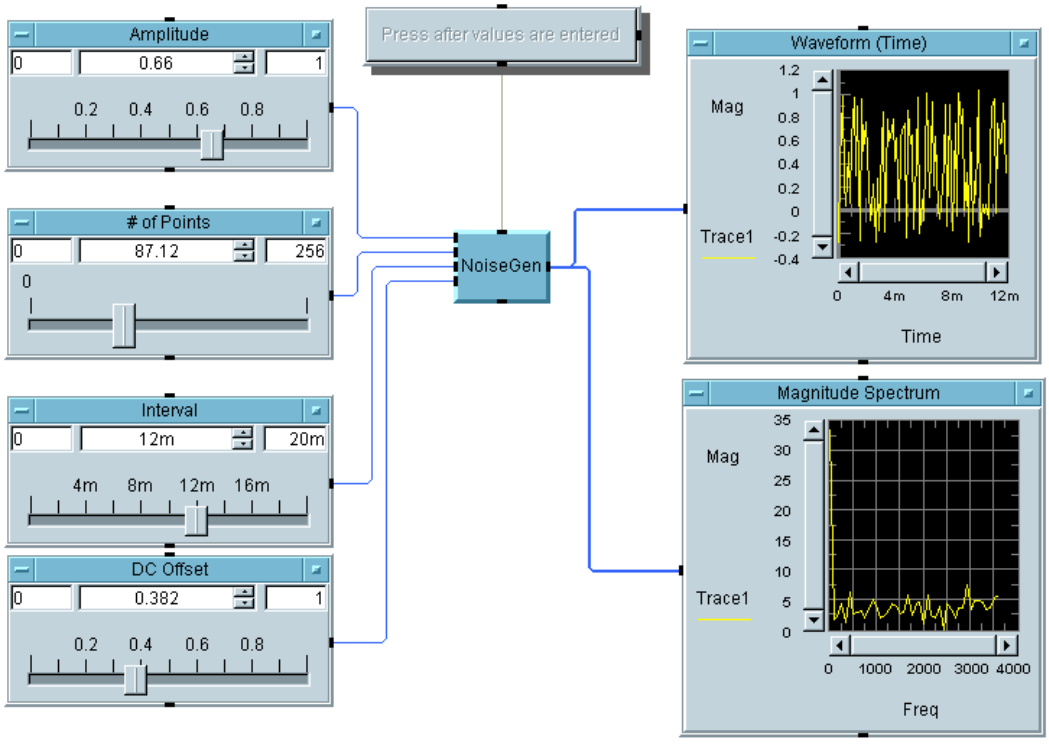


図 A-14. 「不規則ノイズ UserObject」

解答 —「不規則ノイズの NoiseGen オブジェクト」

図 A-15 に、「`NoiseGen UserObject`」の解答例を示します。

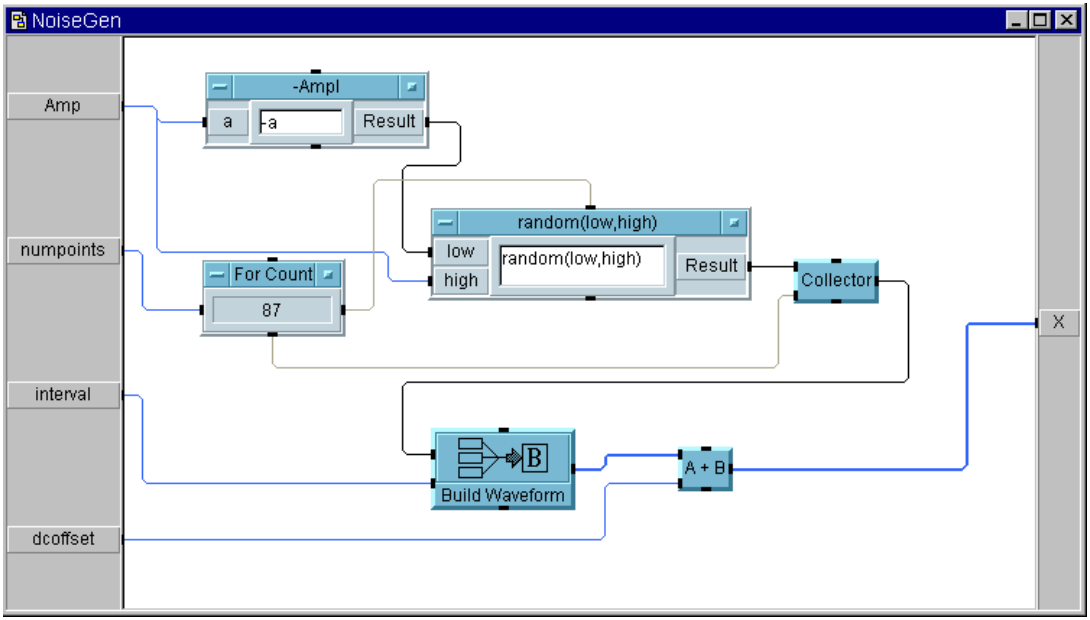


図 A-15. 「NoiseGen UserObject」

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 A-16 に、ステップ 1 の解答例を示します。

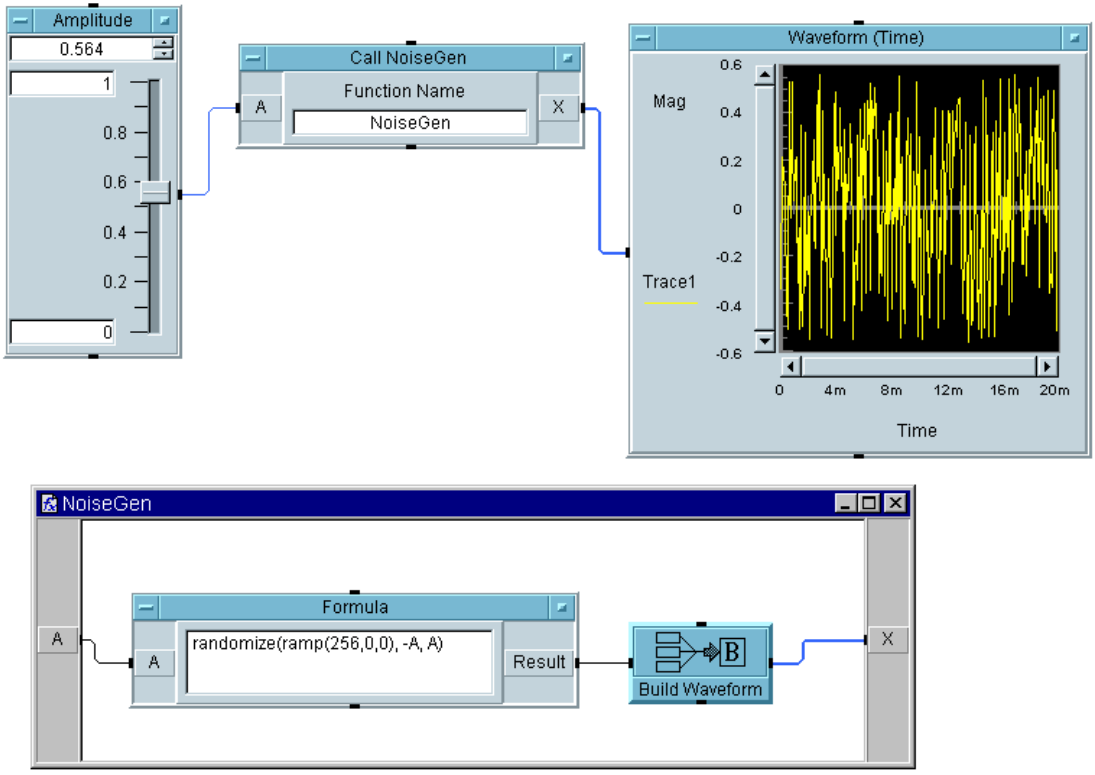


図 A-16. 「User Function」ステップ 1

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 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 A-17 に、ステップ 2 の解答例を示します。

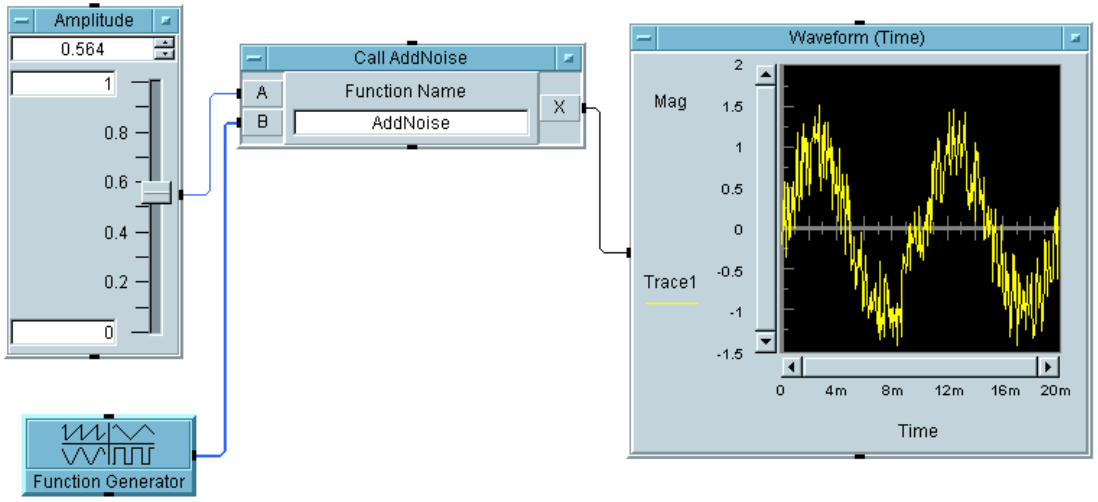


図 A-17. 「User Function」ステップ 2

「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 関数を呼出し、その結果の絶対値を受取ります。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 A-18 に、ステップ 3 の解答例を示します。

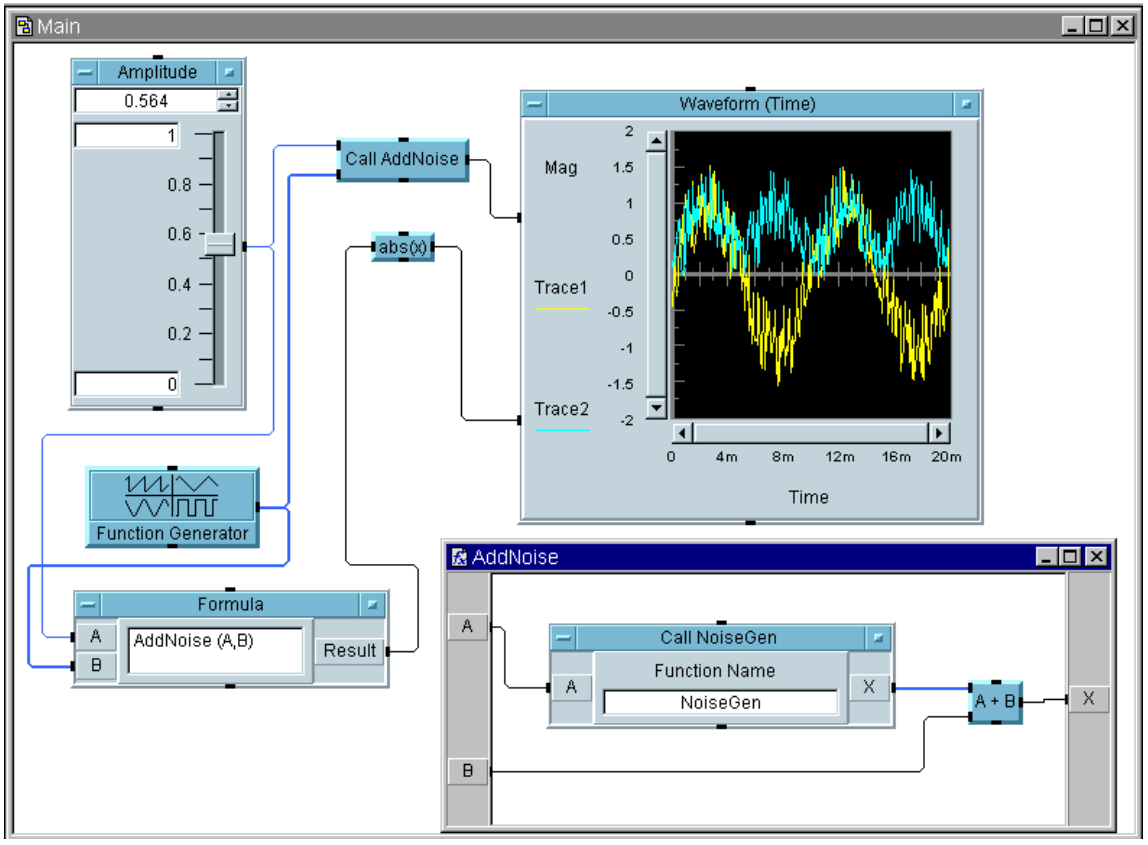


図 A-18. 「User Function」ステップ 3

「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 -「UserFunctions の使用方法」ステップ 4

図 A-19 に、ステップ 4 の解答例を示します。

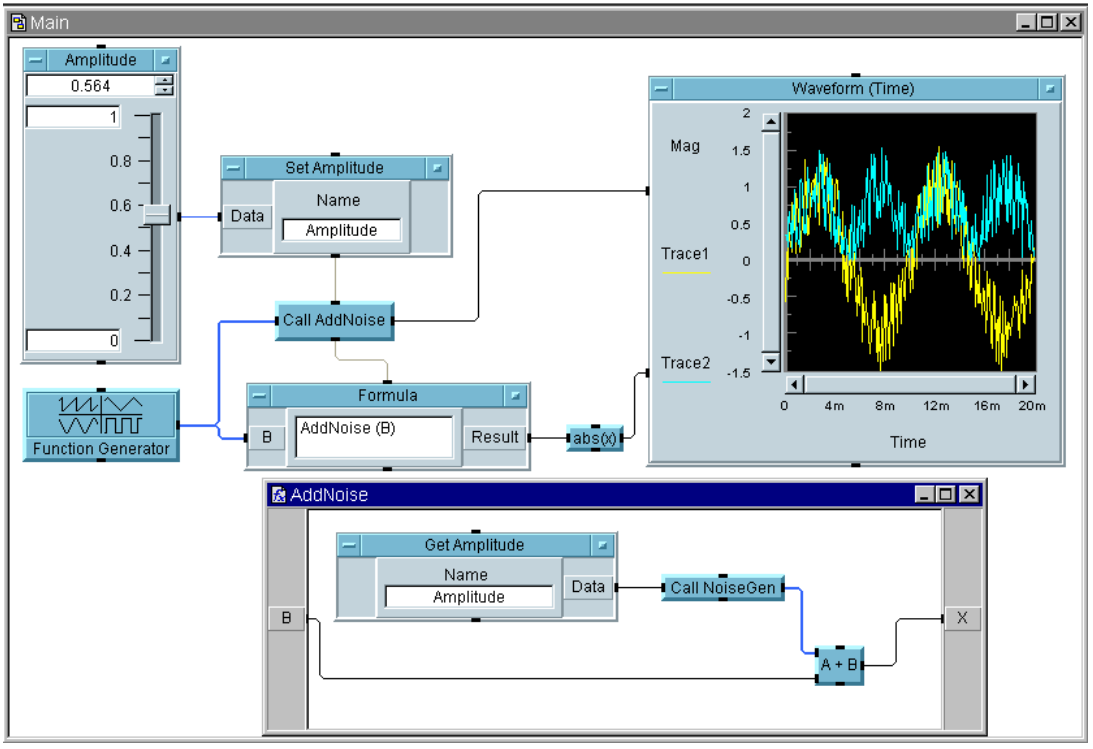


図 A-19. 「User Function」ステップ 4

ヒント：Call AddNoise と Formula オブジェクトはグローバル Amplitude を使用するため、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 ライブラリのインポートと削除

前の練習の uflab.vee 関数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出した後、プログラムからライブラリを削除します。Call オブジェクトのオブジェクト・メニューにある [Select Function] を使用します。

ヒント：Import Library のオブジェクト・メニューにある [Load Lib] をクリックし、指定したライブラリを手動でロードします。そうすると、Call 内の Select Function 機能を使用できます。

解答 -「ライブラリをプログラムからインポートおよび削除する方法」

図 A-20 に、「ライブラリをプログラムからインポートおよび削除する方法」の解答例を示します。

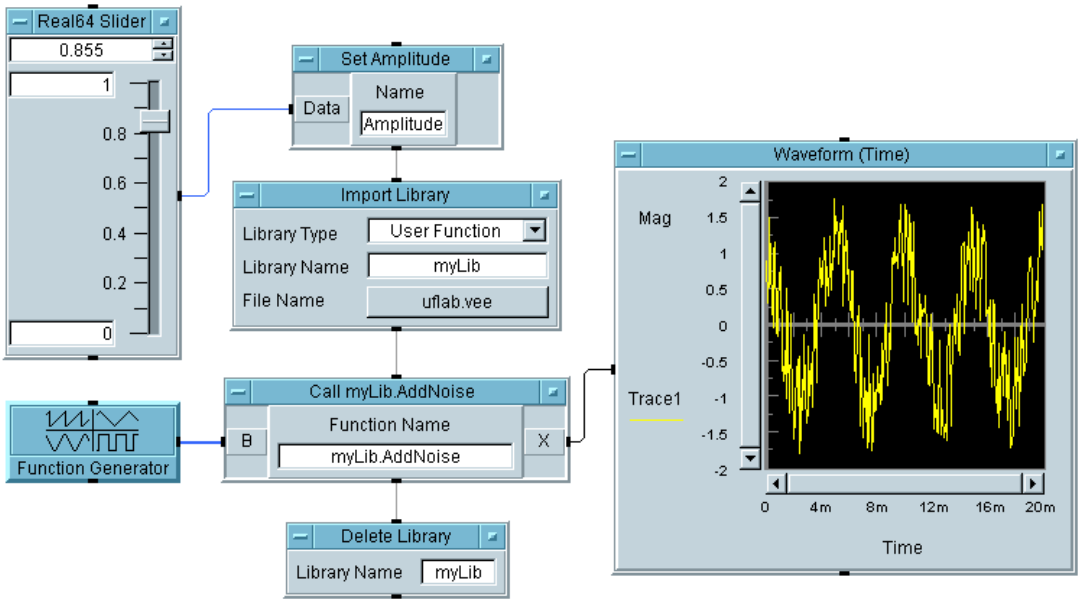


図 A-20. ライブラリのインポートと削除

キー・ポイント

- Select Function: 関数を選択すると、選択した関数に対する正しい入力ピンと出力ピンが設定されます。
- UserFunction の編集: プログラムを使って UserFunction のライブラリをインポートした場合、UserFunction を編集することはできません。それらを表示したり、ブレークポイントを設定してデバッグすることはできます。インポートした UserFunction を編集する必要がある場合は、Merge Library コマンドを使用します。

- **変数設定の注意**：関数内でグローバル変数を使用している場合、その関数をほかのプログラムで使用するには、そのグローバル変数を作成する必要があることに注意してください。入力と出力を明示的に作成する利点の 1 つは、それらの管理が容易になることです。

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 A-21 に、詳細ビューでの解答例を示します。図 A-22 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。

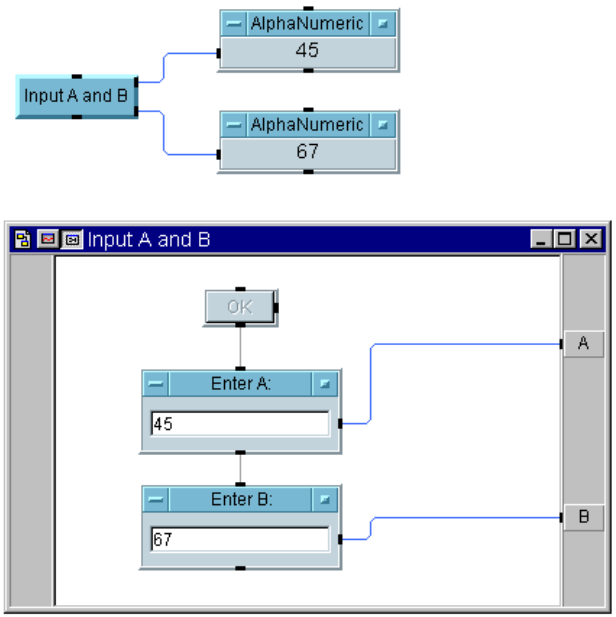


図 A-21. オペレータに A と B の入力を求める UserObject

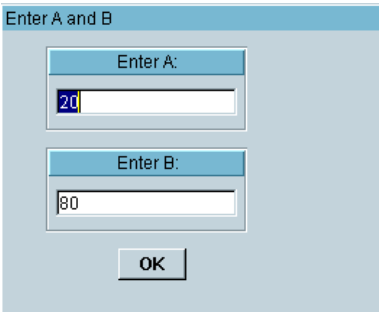


図 A-22. オペレータが A と B を入力するためのパネル

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 A-23 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 A-24 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。

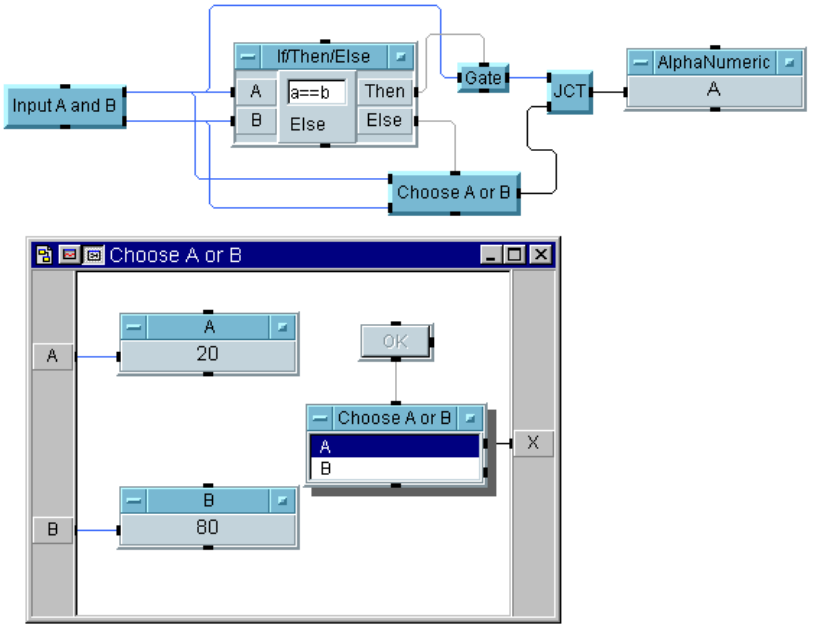


図 A-23. オペレータに A または B のどちらを表示するかをたずねる UserObject

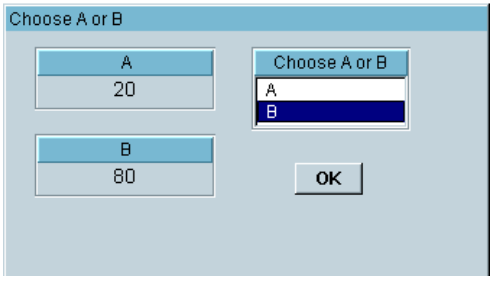


図 A-24. A と B のどちらを表示するかをオペレータが選択するためのパネル

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 A-25 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。

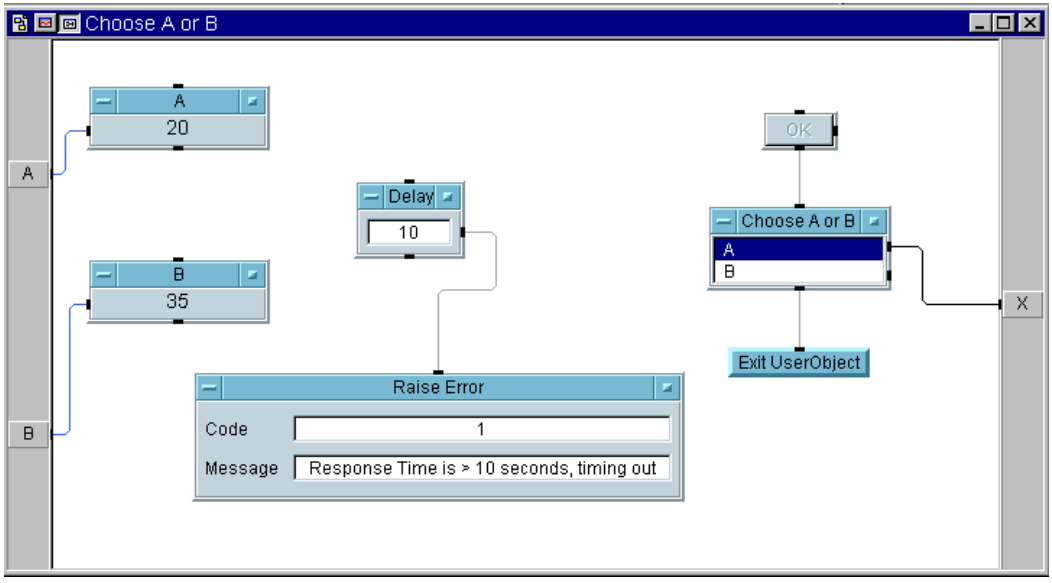


図 A-25. オペレータが選択を入力しなかった場合にエラーを生成

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取ります。図 A-26 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 A-26 に、「ファイルとの間のデータの移動」の解答例を示します。

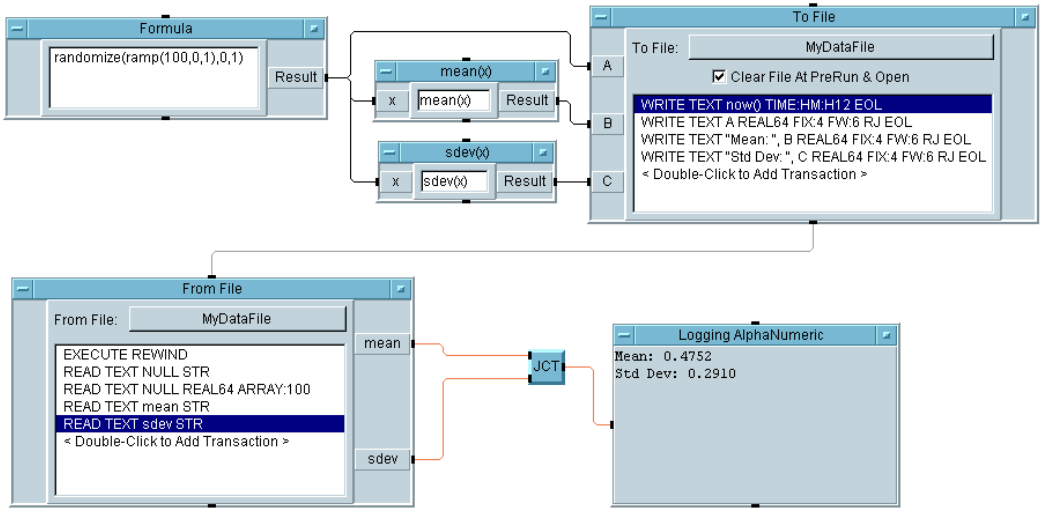


図 A-26. ファイルとの間のデータの移動

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() 関数が 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・ボックス内の式フィールドで、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 配列を読飛ばします。これで、読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 A-27 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。

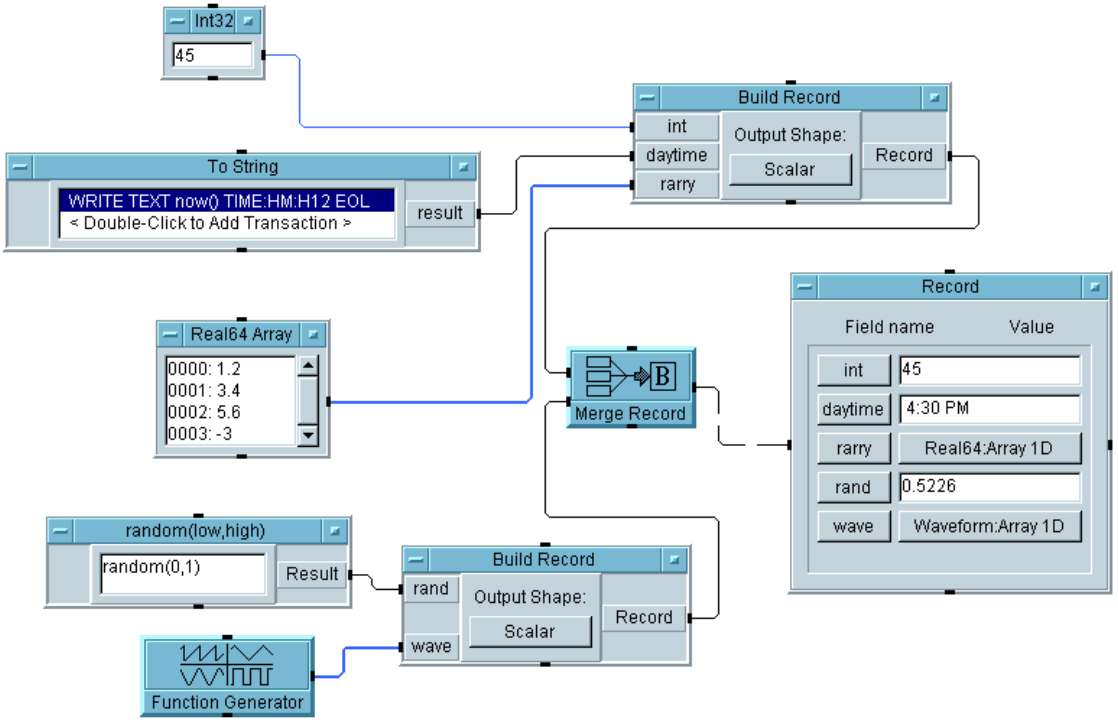


図 A-27. 「レコードの操作」ステップ 1

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します。または、値の入力して 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 A-28 に、「レコードの操作」ステップ 2 の解答例を示します。

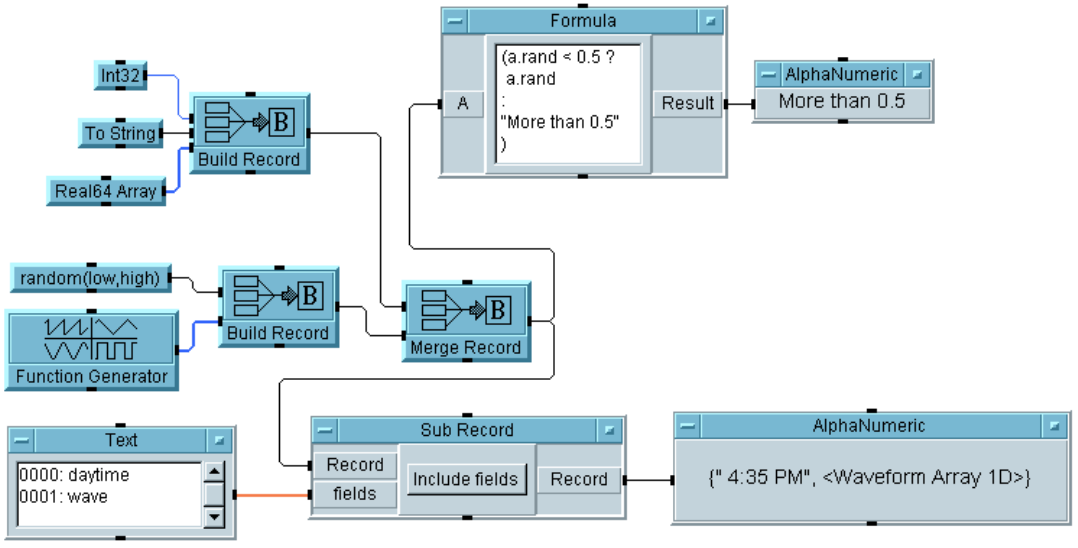


図 A-28. 「レコードの操作」ステップ 2

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 オブジェクトを設定した場合、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取ります。読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 A-29 に、「レコードの操作」ステップ 3 の解答例を示します。

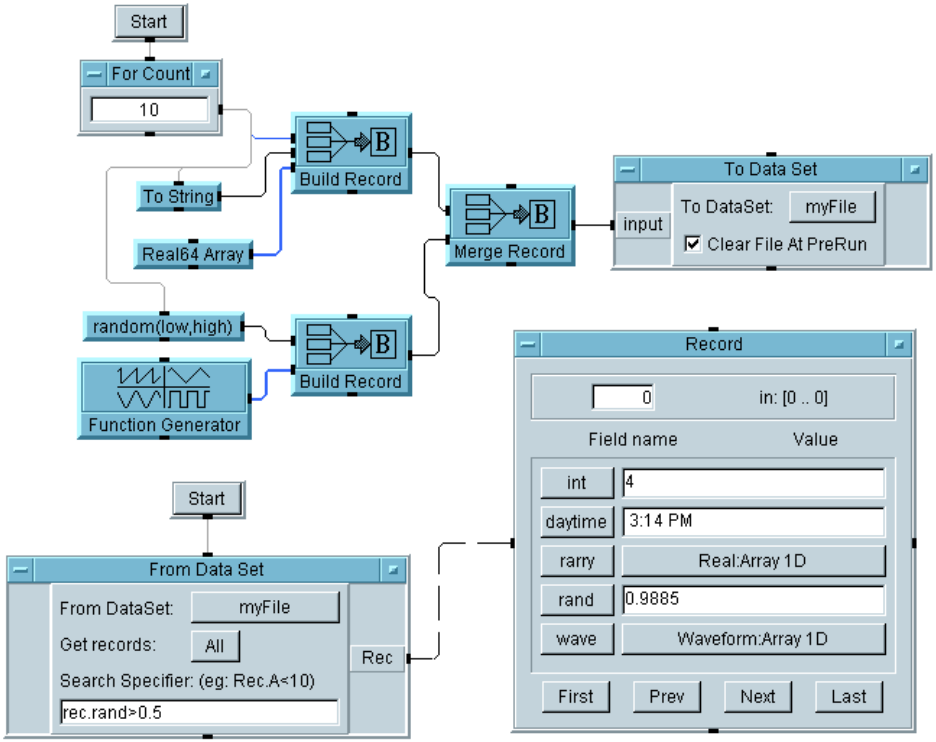


図 A-29. 「レコードの操作」ステップ 3

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・シーケンサ

「テスト・シーケンサの使用方法」ステップ 1

Real64 Slider と Confirm (OK) オブジェクトを持ち、UpperLimit という名前のポップアップ・パネル UserFunction を作成します。UpLimit という名前のグローバル変数と出力端子の 1 つにスライダの出力を送信します。Sequencer オブジェクトを作成します。その Sequencer の test1 に、UpperLimit を呼出す EXEC トランザクションを設定します。

呼出すテストをシミュレートするために AddRand という名前の関数を別に作成します。この関数は、入力値に乱数値 (0 ～ 1) を加算し、入力ピンと出力ピンを 1 つずつ持ちます。

Sequencer で、AddRand を呼出し、0 を送信する test2 を作成します。戻り値をテストして、グローバル UpLimit 値より小さいかどうかを比較します。戻り値が条件を満たす場合は、"PASS" + test2.result を返します。そうでない場合は、"FAILED" + test2.result を返します。Sequencer の Return ピンに Alphanumeric 表示を接続します。

Sequencer オブジェクトの後に、Get Variable オブジェクト (UpLimit) と、別の Alphanumeric 表示を起動します。プログラムを数回実行します。

解答 -「テスト・シーケンサの使用方法」ステップ 1

図 A-30 に、「テスト・シーケンサの使用方法」ステップ 1 の解答例を示します。

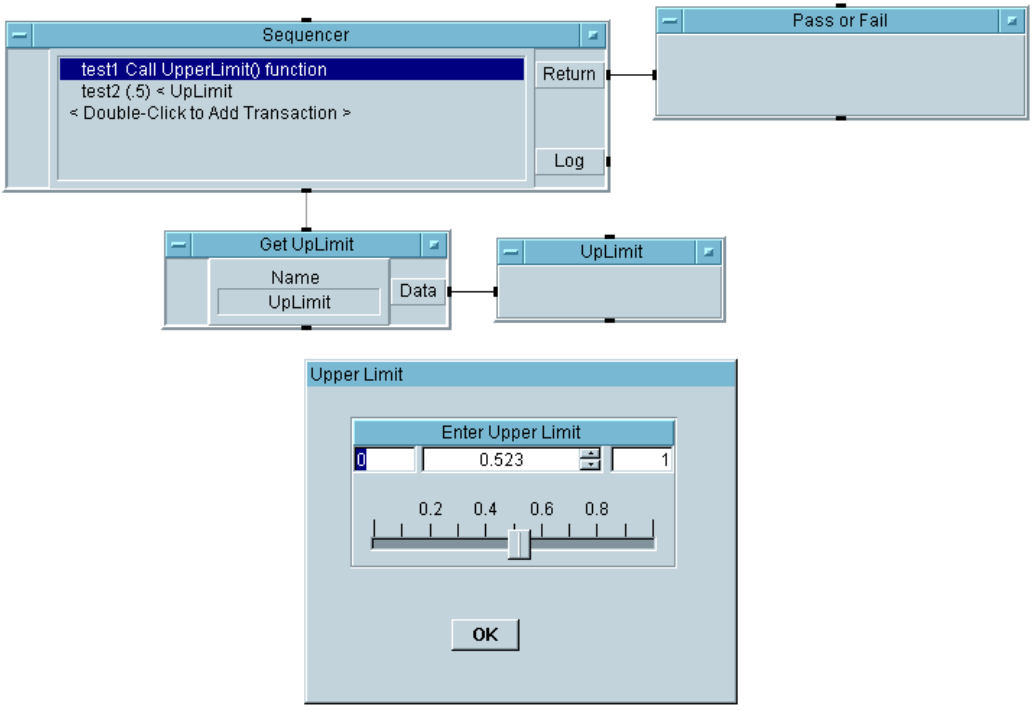


図 A-30. 「テスト・シーケンサの使用方法」ステップ 1

キー・ポイント

- **UserFunction を使ったグローバル変数の設定**: `Sequencer` の最初のトランザクションは、この場合のように、`UserFunction` を呼出してグローバル変数を設定するためによく使用されます。これで、この例のように、後続のテストでもそれらの変数を利用できます。
- **Sequencer の Return ピン**: この例の `Return` ピンは、PASS または FAIL メッセージとテスト値を渡します。このピンを使用すると、特定のテストからメッセージを渡すことができます。

「テスト・シーケンサの使用方法」ステップ 2

テスト 1 を無効にします。特にほかの場所でグローバルが必要ないとすると、`UpperLimit` 関数を直接呼出すことができます。`test2` によって

AddRand(0) の戻り値と UpperLimit 関数の結果が比較されるように、test2 を変更します。

ヒント：テスト 1 を無効にするには、図 A-31 に示すように、[Sequencer Transaction] ダイアログ・ボックスを使用します。

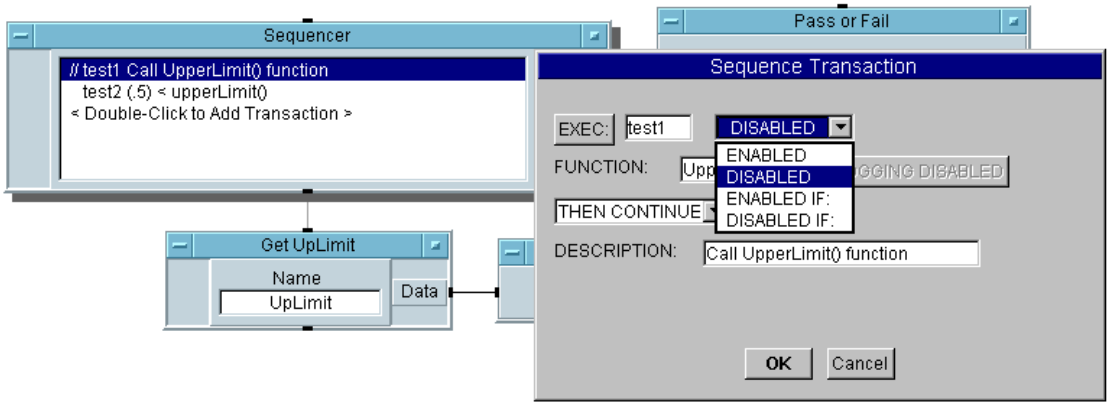


図 A-31. シーケンサのテスト 1 を無効にする

図 A-31 で、Sequencer のテスト 1 が無効であることを示すために、テスト 1 が 2 つのスラッシュで「コメント・アウト」されていることに注意してください。

解答 -「テスト・シーケンサの使用方法」ステップ 2

図 A-32 に、「テスト・シーケンサの使用方法」ステップ 2 の解答例を示します。

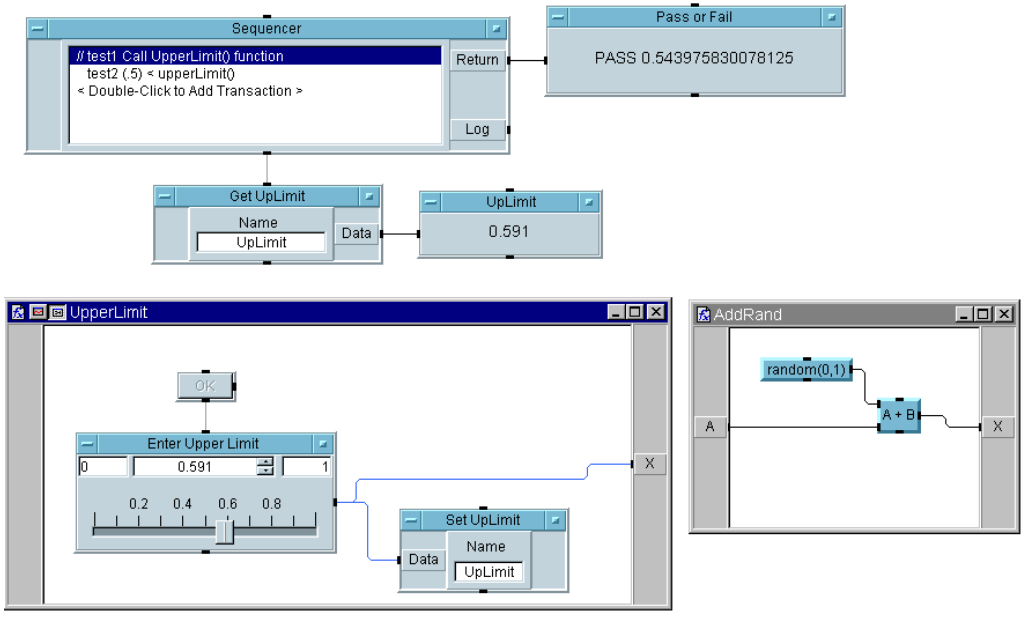


図 A-32. 「テスト・シーケンサの使用方法」ステップ 2

キー・ポイント

- **式フィールドの UserFunction:** この例では、テスト結果を `UpLimit` 変数と比較するのではなく、式フィールド内の変数のあった位置に関数名 `UpperLimit()` を入力できます。

「テスト・シーケンサの使用方法」ステップ 3

VEE 関数 `random (0,1)` を呼出す `test2 Sequencer` トランザクションを編集します。結果が `0.5` より小さいかどうかを比較します。テストが全部で 4 つになるまで、`test2` トランザクションを切取って貼付けます。

`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、そのデータを配列に収集します。その配列を使用して、テスト 2 の結果の最小値、最大値、平均、標準偏差をそれぞれ計算します。

解答 -「テスト・シーケンサの使用方法」ステップ 3

図 A-33 に、「テスト・シーケンサの使用方法」ステップ 3 の解答例を示します。

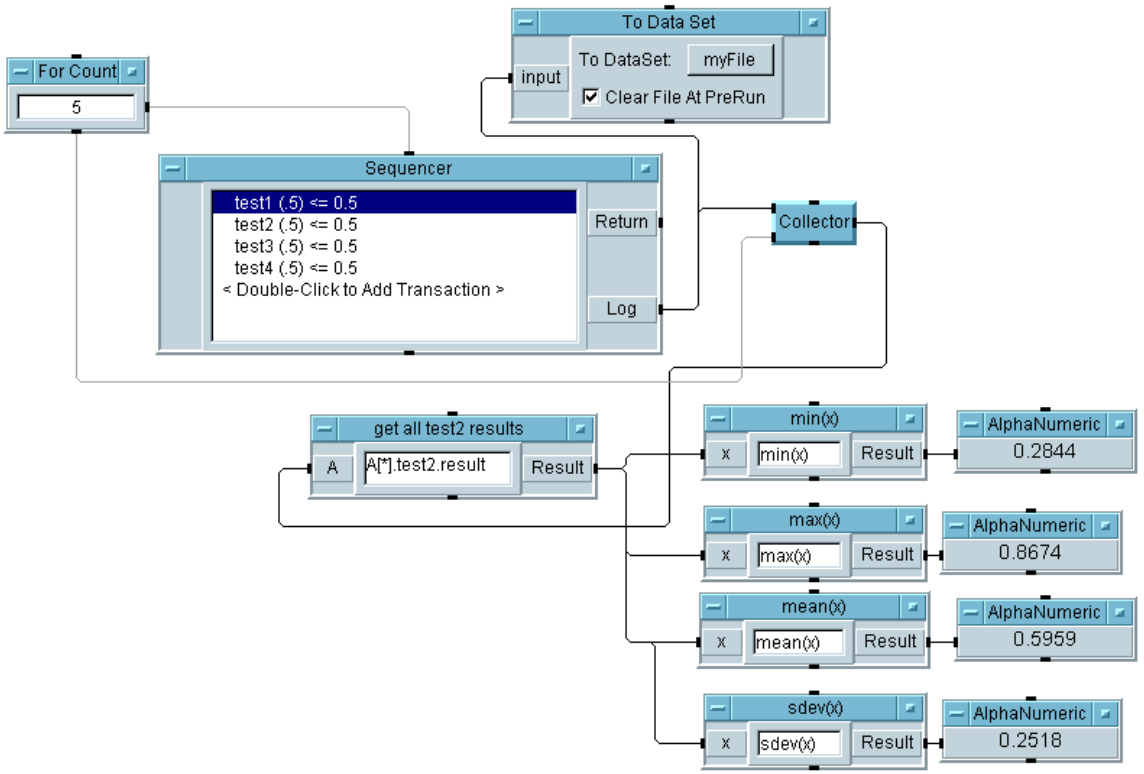


図 A-33.「テスト・シーケンサの使用方法」ステップ 3

キー・ポイント

- **シーケンサを複数回の実行するためのデータ・フォーマット (最初のスレッド)**: `Sequencer` が 1 回実行されると、レコードから成るレコードが出力されます。最初のレコードは、テスト名に一致するフィールド名を持ち、次に各フィールドが、その特定のテストに対するそれぞれ異なるデータを保持するレコードを 1 つずつ保持します。`Sequencer` が数回実行されると、レコードから成るレコードがそれぞれ配列に追加されます。そこで、その配列を調査できます。`Formula` オブジェクト内で

「*< レコード >[*].< レコード >.< フィールド >*」フォーマットを使用すれば、データの配列を取得できます。この例では、test2 の 5 回の実行結果を示す Real 値の配列を得ることができます。次に、この配列から、その最小値、最大値、平均、標準偏差を計算できます。「レコードから成るレコード」の配列内の特定の要素を指定すると、test2 の 1 回の実行を指定できます。たとえば、test2 の初回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。各ステップが 1 秒間隔で実行されるように Delay を追加します。別のスレッドで、test2 のすべての結果を取得し、それらをレコード定数に送信します。

ヒント

- **Delay オブジェクト (最初のスレッド)**: このオブジェクトは、指定された秒数の間、実行フローを保留します。この例では、Sequencer が実行されるたびにタイム・スタンプが変わるように、このオブジェクトが使用されます。
- **タイム・スタンプの追加** : 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、[Record Fields to Log] ⇒ [Time Stamp] をチェックします。図 A-34 に、[Properties] ダイアログ・ボックスの [Logging] タブを示します。

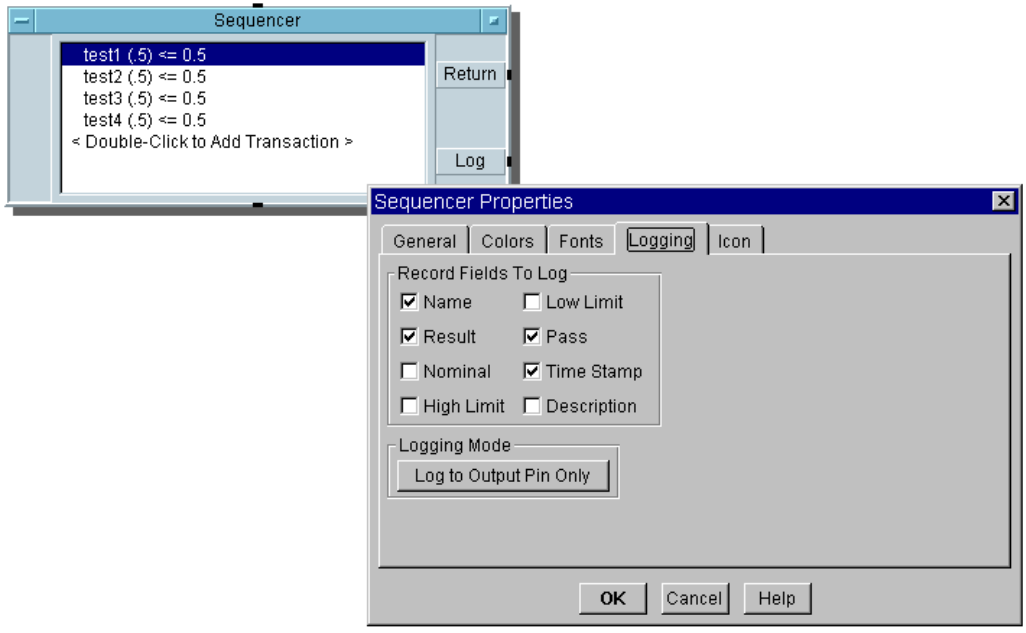


図 A-34. 記録レコードへのタイム・スタンプの追加

解答 -「テスト・シーケンサの使用方法」ステップ 4

図 A-35 に、「テスト・シーケンサの使用方法」ステップ 4 の解答例を示します。

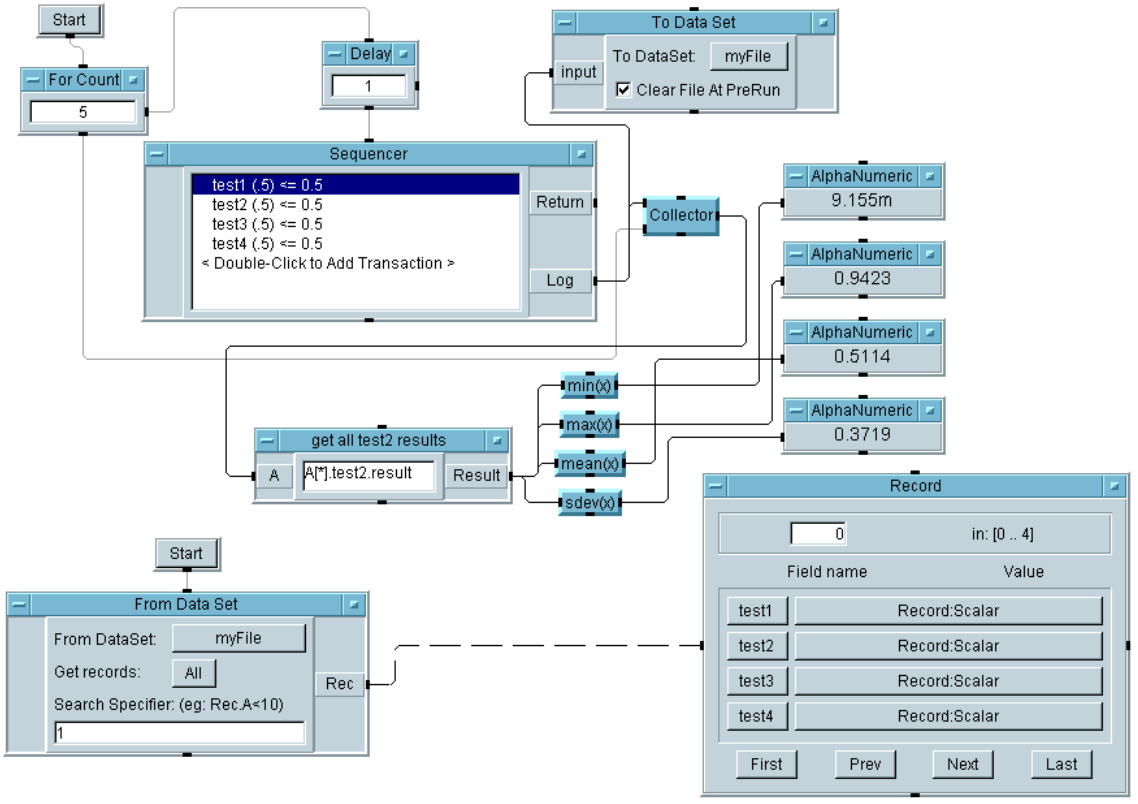


図 A-35.「テスト・シーケンサの使用方法」ステップ 4

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て、[Record Field Data] ダイアログ・ボックスを表示します。図 A-36 に、[Record Field Data] ダイアログ・ボックスを示します。

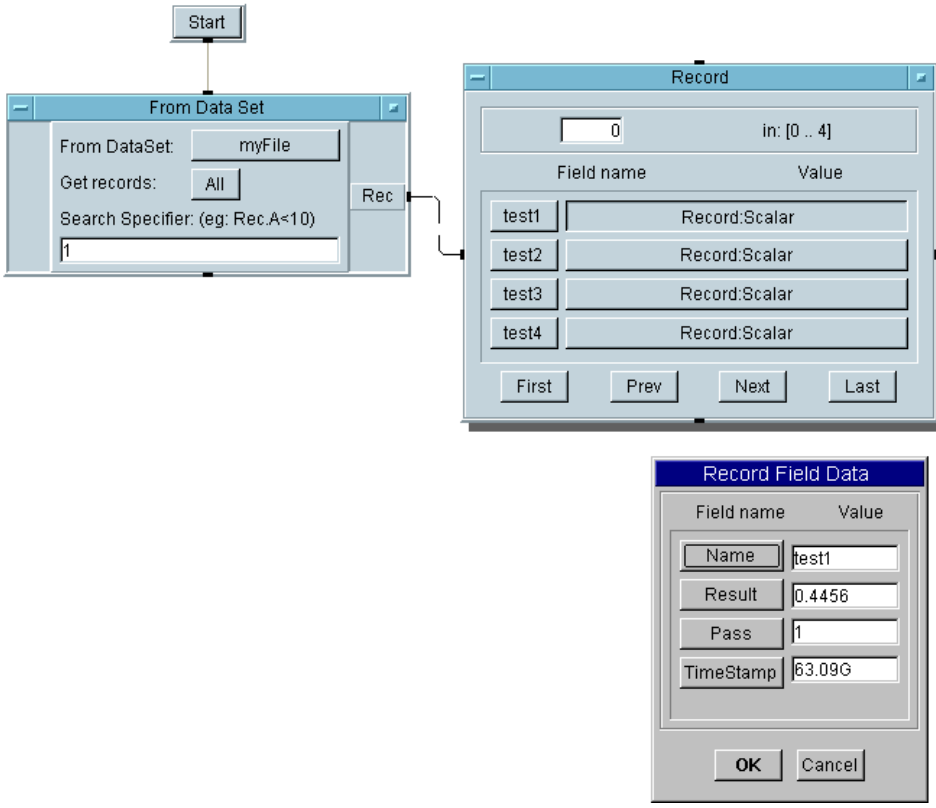


図 A-36. レコードの確認

「テスト・シーケンサの使用方法」ステップ 5

レコードのタイム・スタンプ・フィールドを `Logging Alphanumeric` 表示に出力します。

ヒント：各テストごとに 1 つの 4 つの `Formula` オブジェクトを使用します。4 つの `Formula` の結果すべてを 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・シーケンサの使用方法」ステップ 5

図 A-37 に、「テスト・シーケンサの使用方法」ステップ 5 のタイム・スタンプを表示に出力するプログラム・スレッドを示します。

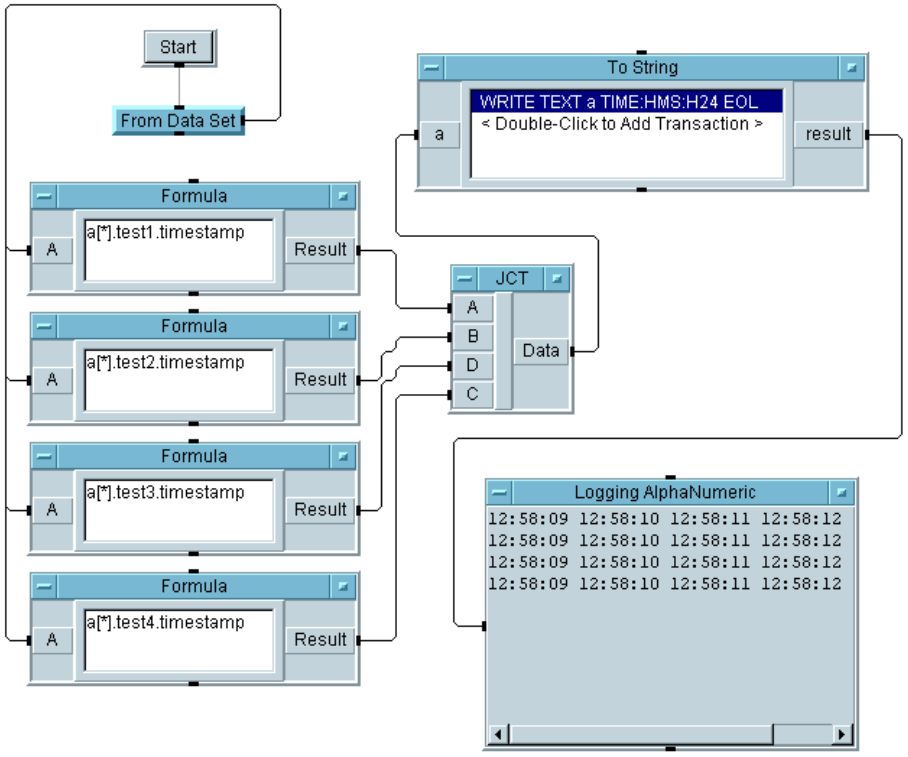


図 A-37. 「テスト・シーケンサの使用方法」ステップ 5

キー・ポイント

- **タイム・スタンプ・フォーマットの変換**: タイム・スタンプを読みやすくするため、`Logging AlphaNumeric` の前の `To String` オブジェクトにより、タイム・スタンプが `Real` フォーマットから `Time Stamp` フォーマットに変換されます。

「テスト・シーケンサの使用方法」ステップ 6

Sequencer に多数のテストがある場合、多くの Formula オブジェクトを使って Junction に個々に接続すると、扱いにくくなることがあります。そのかわり、式を保持する Formula を使用し、実行時に式を生成して、考えられる式をすべてループします。

最初に、例として式の文字列を生成します。

別のスレッドで、Loop と Formula を使用して、テスト式の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" であり、<x> は 1 ～ 4 です。

解答 -「テスト・シーケンサの使用方法」ステップ 6

図 A-38 に、ステップ 6 の解答例を示します。

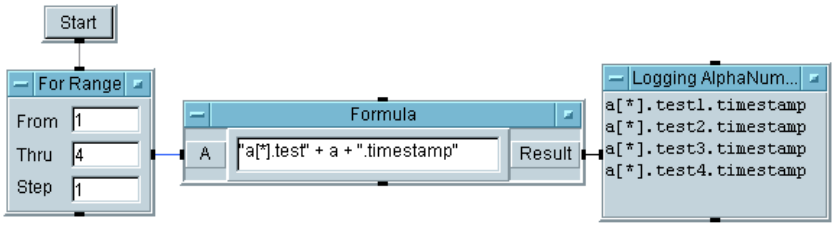


図 A-38. 「テスト・シーケンサの使用方法」ステップ 6

「テスト・シーケンサの使用方法」ステップ 7

次に、ステップ 6 で構築した Loop と Formula を利用して、以前のステップの 4 つの Formula と Junction をこの Loop と Formula で置換えます。さらに、構築した文字列を評価する必要があります。生成した文字列 (式 "a[*].test<x>.timestamp") が実行時に評価されるように、それらを Formula に送信します。

ヒント

- Formula オブジェクトの Formula 制御ピン : 評価する 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別の Formula ボックスを作成します。2 つ目の Formula が評価する式は、実行時に生成されます。

解答 —「テスト・シーケンサの使用方法」ステップ 7

図 A-39 に、ステップ 7 の解答例を示します。

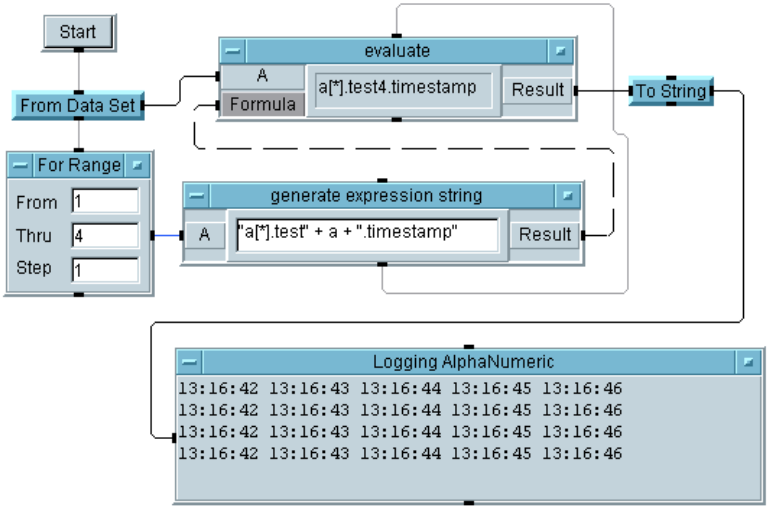


図 A-39.「テスト・シーケンサの使用方法」ステップ 7

キー・ポイント

- `To String` オブジェクトは、やはり `Real64` をタイム・スタンプのフォーマットに変換するために使用されています。
- 最初の「generate」用の `Formula` と 2 番目の「evaluate」用の `Formula` を結ぶシーケンス線に注目してください。これにより、2 番目の `Formula` は、評価対象の新しい文字列を受取るまで、実行されなくなります。

「テスト・シーケンサの使用方法」ステップ 8

テスト 1 に成功し、テスト 2 に失敗したレコードだけを表示します。

解答 -「テスト・シーケンサの使用方法」ステップ 8

図 A-40 に、「テスト・シーケンサの使用方法」の最後のステップの解答例を示します。

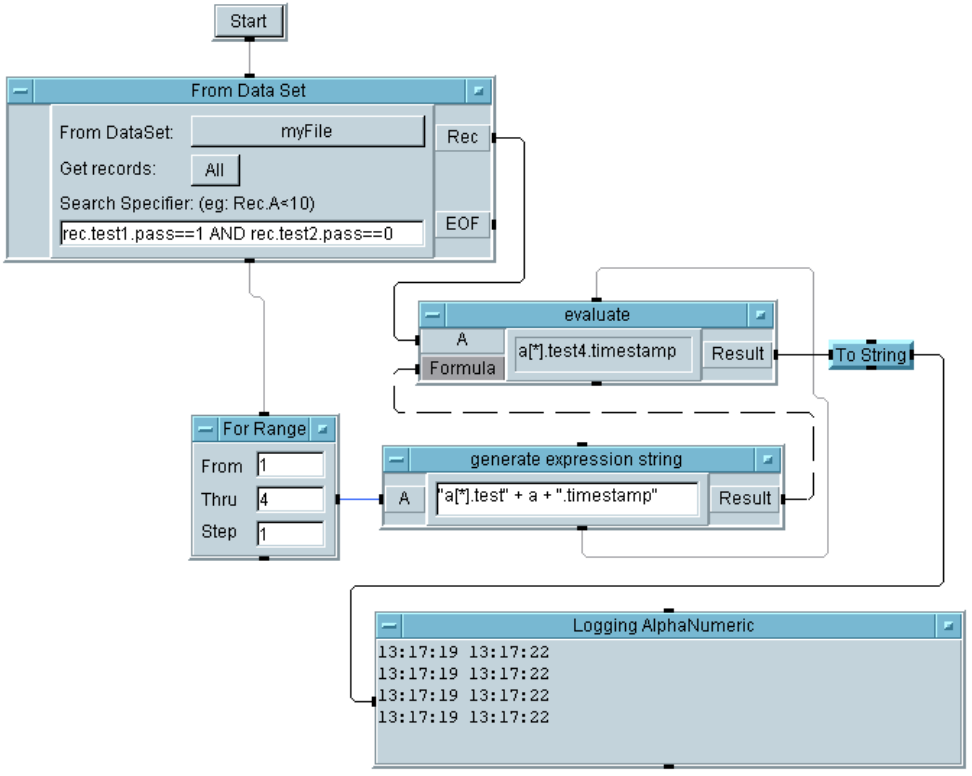


図 A-40. 「テスト・シーケンサの使用方法」ステップ 8

キー・ポイント

- From Data Set オブジェクトの EOF ピン (2 番目のスレッド): 条件に一致するレコードがない場合に、EOF ピンが追加されます。この状態が起こると、EOF ピンが起動されます。エラー・メッセージとともに VEE が停止することはありません。
- From Data Set オブジェクト内の条件式 (2 番目のスレッド): この式は「*(Rec.test1.pass==1) OR (Rec.test2.pass==0)*」であり、同じ「*< レコード >.< レコード >.< フィールド >* 」フォーマットを持っています。Rec は、読取りおよびテストされているデータセット内の各レコードの名前です。test1 と test2 は、VEE が検査するテストを指定します。フィールド名 pass は、VEE によって割当てられている成功 / 失敗

インジケータ (1 または 0) のデフォルト名です。すべてのテストについて、それぞれのフィールドを有効または無効にする場合は、[Sequencer Properties] ダイアログ・ボックスの [Logging] タブを選択します。

用語集

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 6.0 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。

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。

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。

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。

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。

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。

コンテナ
「**データ・コンテナ**」を参照してください。

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。

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付けることができます。

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込みます。

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。

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。

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。

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。

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。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。

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。

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。

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。

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。

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。

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。

HP-UX

Hewlett-Packard 社によって開発された UNIX オペレーティング・システムの派生システムの 1 つ。

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。

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 と HP-UX (with VUE) 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。

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。

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。

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。

最小化ボタン

オブジェクトの 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。

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。

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。

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。

ピン
線を接続できるオブジェクトの外部接続点。

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。

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。

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。

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。

プログラム・エクスプローラ
プログラムを探索するために VEE ウィンドウに用意された機能。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。

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。

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] を使用って変更できます。

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。

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。

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。

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。

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。

ステータス・バー
VEE 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。

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。

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。

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。

トランザクション
VEE 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。To File、From File、Sequencer、Direct I/O などがオブジェクトの例です。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。

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。

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。

Windows 95、Windows 98、Windows NT 4.0、Windows 2000

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。

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。

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。

索引

し

す

せ

そ

た

ち

つ

て

と

な

に

ぬ

の

は

へ

ほ

れ

ろ

わ