Wide Studio Builder
User's Guide

目次

A はじめに

1 WideStudio とは

WideStudio は、Windows や、Linux、FreeBSD SOLARIS 上で動作する、マルチプラットフォームな GUI 開発環境です。純国産の完全フリーで、オープンソースであることが最大の特徴です。誰でもが手軽に入手でき、かつ、またオープンでマルチプラットフォームなアプリケーションが迅速に開発できることが魅力です。

WideStudio は次のような特徴を持っています。

- オープンソースでかつ、X11/MIT ライセンス
- C++ ベースでの開発
- プラットフォームに依存しないウィンドウアプリケーションの開発
- 他のライブラリに依存しない完全オリジナルのクラスライブラリ
- ウィンドウアプリケーションの編集が簡単にできるアプリケーションビルダ
- イベント駆動方式による必要最小限のソースコード記述
- 拡張性を保証する追加クラスライブラリ構築機能と、クラスライブラリインポート機能

経験のある方ですと、Windows や、X11 等の複数のプラットフォームでアプリケーションを開発したことがあるかもしれません。今まで、筆者もいろいろなアプリケーションを開発してまいりました。今思えば、小遣いをためてポケットコンピュータを手にいれて、機械語を直接入力して、ドット単位で絵をかいてゲームもどきを作っていたころが懐かしくなります。それから、MJS-DOS 機を手にいれて、それから、Windows 3.0、Windows 3.1、UNIX / X11 Window system、Xt/Motif、Windows95/NT (Win32)、MFC/ActiveX と渡り歩きました。

そう、もうプラットフォーム間の非互換に悩みたくなかったので、マルチプラットフォーム対応にしました。

まあ、それはそれで重要なんですが、適当にぷちぷちマウスをつくだけでアプリケーションができたら、素晴らしいなぁと思い、WideStudio の統合 IDE であるアプリケーションビルダを用意しました。

次の図は WideStudio の構成とファイルの構成を示します。

Wide Studio Manual Page

［WideStudio の構成］

2　C++とは

WideStudio は、C++ で構築されています。C++ とは、C 言語を拡張して、オブジェクト指向のプログラミングが行えるようにしたものです。C 言語は、基本的な言語でハードウェアを直接制御を行うようなことから、ある程度構造的で大規模なソフトウェアプロダクトまでカバーできる幅広い用途があり、もっとも良く使われる言語の内の１つです。C++ はそんな、C 言語の素晴らしいものを引き継いで、より効率的にソフトウェア構築できる機能を備えています。

しかしながら、C++ を追求した体系であればあるほど、C 言語とはかけ離れた世界になってしまい、C 言語に慣れ親しんだ方、初心者の方には難解になってしまう傾向にあります。

WideStudio では、素晴らしい C++ の機能を十分生かせるように、一見、C 言語でプログラミングしているのではないかと錯覚させるような感じで気軽にプログラミングでき、その上、一歩踏み込んで、クラスをばりばり使ったプログラミングまで、簡単にできるようになっています。

3 WideStudio の開発を通じて

よ〜く考えてみると、WideStudio と同じようなアプリケーション構築環境は、すでに世の中に同じようなものがごまんと存在します。ではなぜ、WideStudio なのでしょうか？

よく、「自分で何かアプリケーションを作ってみたいな、でも難しいんだろうなぁ」または、「もし高いアプリケーション構築ツールを購入してつかってみても、使えなかったらやだなぁ」と、プログラミングを始められないでいる方々がいらっしゃると思います。WideStudio を使えば、そういった心配はいりません。無料なので、気軽に使ってみることができます。
よく、オープンソースのあまり浸透しいない Windows の世界で「オープンソースってなに？、どうして WideStudio は無料なの？」といった声があります。

筆者の開発経験の中ではオープンソースの世界から学んだものが非常に大きく、目的の一つにそのオープンソースの恩恵を WideStudio の開発を通して還元することにあります。
WideStudio をいかに効率的に開発できるかを追求し、少ない時間で設計、開発を行いました。これから、ソフトウェアを設計・製作される方々には、どこか参考になる部分があるのではないかと思います。

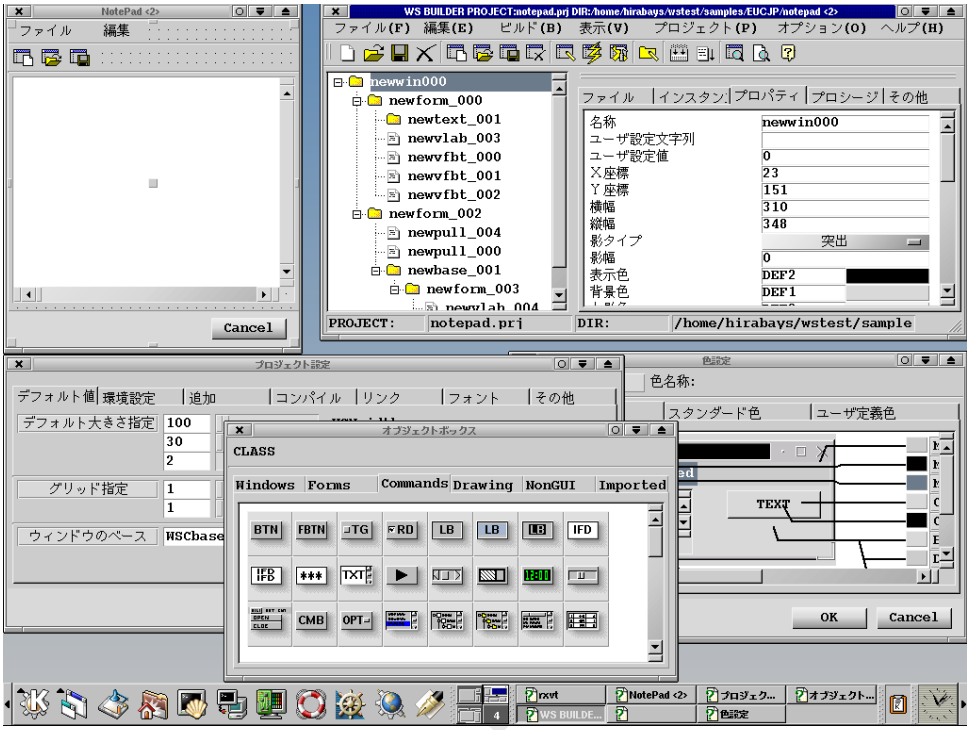
小さなものから本格的なアプリケーションの開発まで、物作りを通して、開発の楽しさが共有できたらと思います。

Wide Studio Manual Page

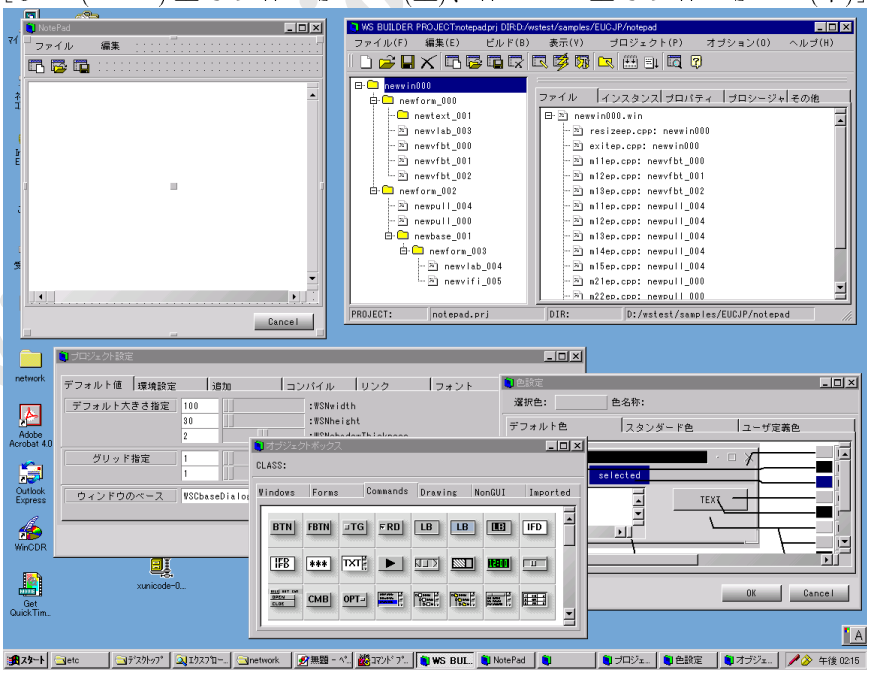
4 マルチプラットホーム

WideStudio を使うと、統一された環境で、プラットホームを気にすること無く簡単にマルチプラットホームなアプリケーションを開発することができます。そして、WideStudio で開発された同じアプリケーションを、Unix/Linux 、Windows 上で動作させることができます。

Wide Studio Manual Page



[Unix (Linux) 上での WideStudio (上)、Windows 上での WideStudio(下)]



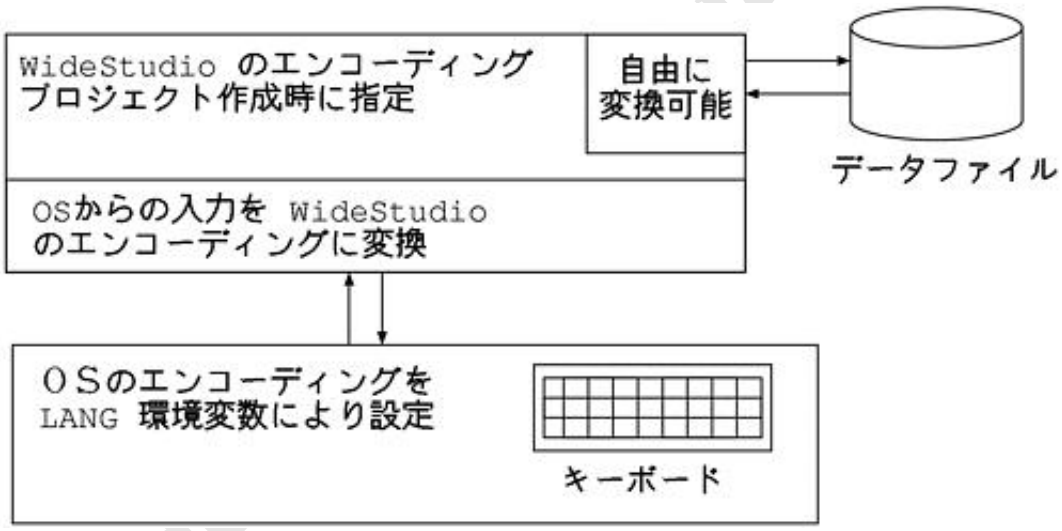
5　マルチエンコーディング

WideStudio を使うと、プラットホーム間で異なるエンコーディングを気にすること無くアプリケーションを開発することができます。

WideStudio では、ユニコード（UTF8）、EUCJP、SJIS 等のエンコーディング体系を利用することができます。

WideStudio 上ではプラットホーム毎のエンコーディングとは独立して各エンコーディングを扱えるため、例えば、Linux 上において EUCJP で作成されたアプリケーションが、EUCJP コードのまま Windows 上で動作いたします。

LANG 環境変数を指定することによって、オペレーティングシステムからのキーボード入力のコードを特定し、WideStudio 内で、自動的に変換を行います。また、外部のデータファイルの入出力において、自由にコード変換を行うことができます。



［WideStudio のエンコーディングの扱い］

［例えば、Windows 上での EUCJP コードでの表示］

また、ユニコード等を利用することにより、国際化アプリケーションを構築することができます。

Wide Studio Project Page

［ユニコードによる各国語の表示］

WideStudio でサポートしているエンコーディングは次の通りです。

Wide Studio Manual 1.0 Page

エンコーディング	
ISO8859_1	西欧
ISO8859_2	東欧
ISO8859_3	トルコ、エスペラント
ISO8859_4	バルト
ISO8859_5	キリル
ISO8859_6	アラビア
ISO8859_7	ギリシャ
ISO8859_8	ヘブライ
ISO8859_9	トルコ
ISO8859_10	バルト 2
ISO8859_13	ISO8859(13)
ISO8859_14	ISO8859(14)
ISO8859_15	西欧 2 (ユーロ)
UTF8	UTF8
KOI8R	KOI8 ロシア語
EUCJP	EUC 日本語
SJIS	SJIS 日本語
EUCKR	EUC 韓国語
EUCCN	EUC 中国語
BIG5	BIG5 中国繁体字

Wide Studio Manual Page

6 オープンソース・ライセンス

WideStudio はオープンソースです。誰もが自由にソースコードにアクセスでき、また、自由に配布することができます。
X11/MIT ライセンスですので、自由に商用アプリケーションをつくることもできます。

- 自由なソースコードの配布
- X11/MIT ライセンス
- 商用利用も含め、無償で利用可能

Wide Studio Manual Page

7 WideStudio の入手方法

WideStudio は、現在、下記の公開 URL から入手できます。

http://www.widestudio.org

下記の URL からも入手できます。

http://www.sourceforge.net/widestudio
http://www.vector.co.jp

Wide Studio Manual Page

8 WideStudio の構築とインストール

8.1 UNIX 上での構築とインストール

次に示すように、入手した WideStudio のソース、マニュアルを展開します。
(注意)vX.X は、バージョンにより異なります。

Linux の場合

```
cd /tmp
tar -zsxvf ws-vX.X.tar.gz
cd /usr/local/ws
tar -zsxvf ws-vX.X-doc.tar.gz
```

SOLARIS の場合

```
cd /tmp
gzip -cd ws-vX.X.tar.gz | tar -moxvf -
cd /opt/ws
gzip -cd ws-vX.X-doc.tar.gz | tar -moxvf -
```

次にビルドです。次の手順にしたがってビルドを行います。ビルドには、ランタイムライブラリ、スタティックライブラリがあります。下記の手順中に出て来るディレクトリ ws は、WideStudio をインストールしたディレクトリを示します。

マシン環境によっては、コンパイルでエラーが発生する場合もあります。そのような場合、configure コマンドの実行後、sys/config/mkflags の内容を編集し、パスやリンクするライブラリの設定をすることで、エラーの回避することができます。
SOLARIS 上で日本語エディションを動作させる場合、日本語 SOLARIS が必要です。FreeBSD の場合には、make コマンドの代わりに、gmake コマンドを使用します。

全てビルドする場合

```
cd ws/src
./configure
make
```

ランタイムライブラリのみの場合

```
cd ws/src
./configure
make runtime
```

Wide Studio Manual Page

デバッグ用ライブラリのみの場合

```
cd ws/src
./configure
make debug
```

ビルドが終ったら、アプリケーションビルダ (ws/bin/wsbuilder) が作成されているか、確認してみてください。
ビルドが成功していたら、インストールです。スーパーユーザーになって次のように入力します。

```
cd ws/src
make install
```

8.2 Linux 上での rpm バイナリパッケージのインストール

RPM バイナリパッケージを使用すると、ビルドせずに手軽にインストールできます。まず、URL から rpm ファイルを入手します。
スーパーユーザ権限で次の様に入力します。XXXX の部分はバージョンにより異なります。

```
rpm -i ws-runtime-vXXXX.i386.rpm
```

パッケージは、/usr/local/ws にインストールされます。rpm バイナリパッケージは、C/C++ ランタイム rpm パッケージ等に依存しています。インストール時にそれらのパッケージが足りない主旨のメッセージが出力されるようでしたら、そちらの方もインストールください。

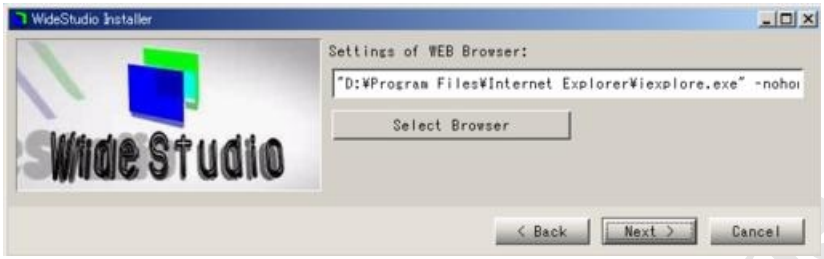
8.3 Windows 上でのインストール

URL から入手した ZIP もしくは LZH ファイルを展開します。すると、wsinst というディレクトリができます。wsinst ディレクトリ内の setup.exe を実行します。すると次のようなインストール画面が表示されますので、まず、インストールする先を指定します。

［WideStudio の setup 画面］

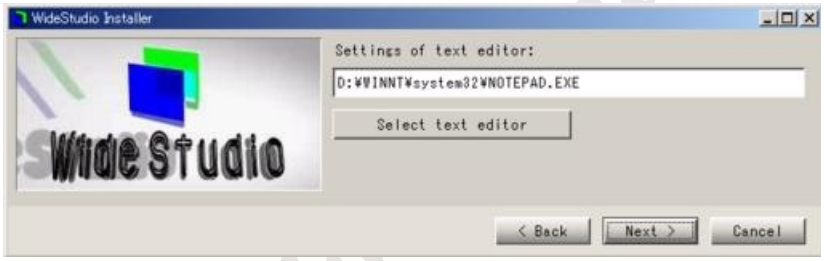
Wide Studio Manual Page

次に、Web ブラウザの指定画面が表示されます。デフォルトでは、Html ファイルを表示するアプリケーションに指定しているものが指定されます。



［ブラウザの指定画面］

次に、ソースコードエディタの指定です。お好みのエディタを御指定ください。



［ソースコードエディタの指定画面］

次に、WideStudio と gcc のライセンスの確認画面が表示され、インストールが行われます。インストールが済んだら、コンピュータをリブートします。

Wide Studio Official Page

8.4 環境変数の設定 (Windows)

WideStudio のインストーラが設定を行うので、特に、設定の必要はありません。インストールが終ったら、一度リブートしてください。

8.5 環境変数の設定 (UNIX)

WideStudio をインストールし、利用可能とするために、次の環境変数を設定します。

WSDIR	インストールディレクトリの指定
PATH	WideStudio のコマンドパスの追加
LD_LIBRARY_PATH	WideStudio のダイナミックリンクライブラリパスの追加

UNIX 上のアプリケーションビルダにおいて、WSDIR 環境変数が設定されていない場合、次の様な値として扱われます。

システム	デフォルト
Linux	/usr/local/ws/
SunOS	/opt/ws/
その他の UNIX	/usr/local/ws/

WideStudio が例に示すディレクトリに展開されている場合は、次のように環境変数を設定しましょう。
(表中の xxxxxx は、変更前の環境設定設定を示します。)

インストールディレクトリ例	/export/home/ws/
csh の環境変数の設定例	setenv WSDIR /export/home/ws setenv PATH xxxxxx:/export/home/ws/bin setenv LD_LIBRARY_PATH xxxxxx:/export/home/ws/lib
sh の環境変数の設定例	WSDIR="/export/home/ws";export WSDIR PATH="xxxxxx:/export/home/ws/bin";export PATH LD_LIBRARY_PATH="xxxxxx:/export/home/ws/lib" export LD_LIBRARY_PATH

csh 場合、.cshrc、sh の場合は、.profile などに加えて実行環境を整えましょう。なお、環境変数の設定方法の詳しい設定方法に関しては、それらのマニュアルを御参照下さい。
一度、.cshrc 等を編集した場合、rehash 等を行ない値を反映させてください。一度ログアウトするし、ログインしなおすことも有効な手段です。

Wide Studio Manual Page

8.6 その他の設定ファイル (UNIX)

アプリケーションビルダで使用する設定ファイルには、次のようなものがあります。いずれも、アプリケーションビルダの実行時に自動的に生成されます。生成される場所は、実行ユーザのホームディレクトリです。

ファイル名	内容
.wsrc	ルック&フィールやデフォルト色等の設定
.wsbuilderrc	使用するツール等の設定
.wsbuilder_defaults	アプリケーションビルダ用の形状情報等

wsreset コマンドは上記のファイルをホームディレクトリから削除し、設定をデフォルトに戻します。設定がおかしくなったりした場合に、利用しましょう。

Wide Studio Manual Page

9 WideStudio のアンインストール

9.1 Windows 上でのアンインストール

Windows 上でのアンインストールは、通常のアプリケーションと同じようにコントロールパネルのアプリケーションの追加と削除から、WideStudio を選択して、アンインストールします。または、WideStudio をインストールしたディレクトリ配下の下記のコマンドを実行してください。

```
WideStudio/ws/bin/unsetup.exe
```

9.2 UNIX 上でのアンインストール

UNIX 上でコンパイルして、make install によってインストールした場合は、make uninstall でアンインストールすることができます。ビルドしたディレクトリに移動して、スーパーユーザ権限で下記のようにコマンドを実行してください。

```
make uninstall
```

9.3 Linux 上での rpm パッケージのアンインストール

Linux 上での rpm バイナリパッケージによるインストールを行った場合は、rpm コマンドで簡単にアンインストールをすることができます。スーパーユーザ権限にて、次の様にコマンドを入力します。

```
rpm -e ws
```

Wide Studio Manual Page

B　さあ始めよう

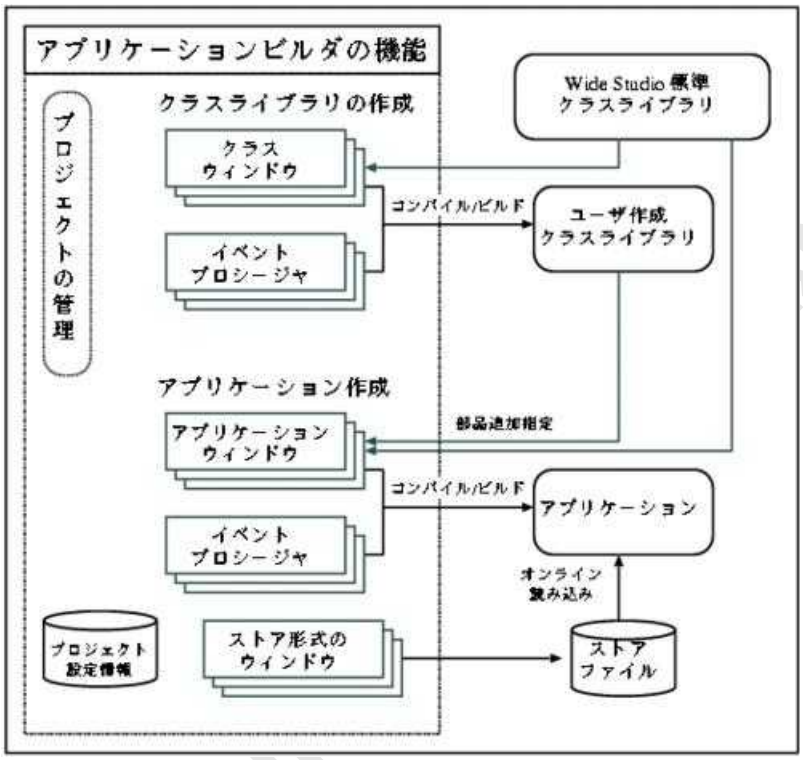
1　アプリケーションビルダとは

アプリケーションビルダは、強力でしかも有用なプログラムで、オブジェクト指向 GUI パッケージ WideStudio の機能を最大限に引き出すことができます。GUI アプリケーション作成者は、このプログラムを使用することで、面倒な GUI ソース作成や構築の作業から開放されます。
アプリケーションビルダの主な機能は次の通りです。

- アプリケーションウィンドウの作成と管理
  アプリケーションウィンドウ (アプリケーションの画面) の対話による作画とファイルの管理を行います。
- オブジェクトの設定とイベントプロシージャ(EP) の指定
  対話による GUI オブジェクトのプロパティの設定やイベントプロシージャ関数の定義などを行います。
- オブジェクトの参照と構成の表示
  作成されたアプリケーションウィンドウに存在するオブジェクトの構成一覧の表示や、プロパティの一覧、オブジェクト情報等の表示を行います。
- プロジェクトの作成と管理
  プロジェクトによるアプリケーションウィンドウやイベントプロシージャ等のファイルの管理とロードモジュールの管理、作成環境、コンパイル設定の保持などを行います。
- アプリケーションのコンパイルと構築
  プロジェクトに登録されたアプリケーションウィンドウの C++ ソースプログラム自動生成とコンパイル、指定されたライブラリとのリンクを行い、ロードモジュールを作成します。
- アプリケーションの実行とデバッグ
  構築されたアプリケーションの実行をさせたり、デバッグなどを行います。
- クラスウィンドウのサポート
  作成されたアプリケーションウィンドウを C++上のクラスとして生成し、部品として利用することをサポートします。また、それらの部品を集約してクラスライブラリ (ダイナミックリンクライブラリ) を作成することができます。
- ストアアプリケーションウィンドウのサポート
  作成されたアプリケーションウィンドウをオンラインストア形式に保存し、オンラインでアプリケーションウィンドウを読み込んだり、破棄したりすることができます。また、ストアデータを利用して、アプリケーション間でオブジェクトを共有したりすることもできます。

Wide Studio Manual Page

次の図はアプリケーションビルダの機能とファイルの構成を示します。



［アプリケーションビルダの機能概要］

Wide Studio

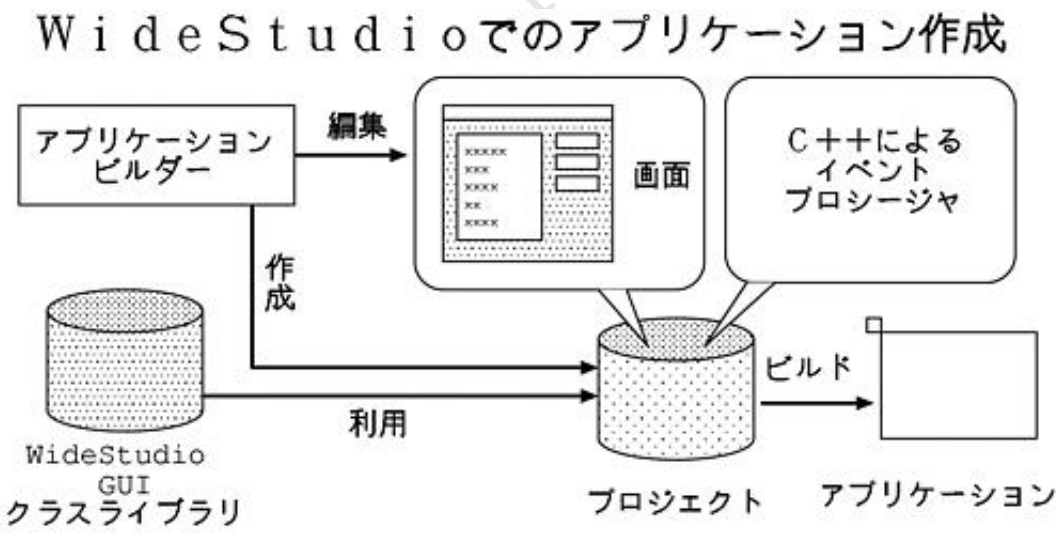
2　アプリケーション開発の手順

アプリケーションビルダーを用いたアプリケーションの開発の手順は、次の様になります。

- プロジェクトの作成
  ユーザは、アプリケーションを構築するためにまず、プロジェクトを作成します。プロジェクトとは、アプリケーションビルダで構築するアプリケーションの管理の単位です。詳しくは、第六章 プロジェクト編を御覧ください。

- アプリケーションウィンドウの作成
  アプリケーションビルダで作成したウィンドウをアプリケーションウィンドウと呼びます。ビジュアルに編集することができます。詳しくは、第三章 アプリケーションウィンドウ編を御覧ください。

- イベントプロシージャの作成
  イベントが発生すると、イベントプロシージャが実行されます。イベントプロシージャを使用することで、オブジェクトだけでは実現が難しい高度な動きをする画面の実現、オブジェクトへの機能の追加や、データの投入など、更に複雑な動作を実現することができます。詳しくは、第五章 イベントプロシージャ編を御覧ください。



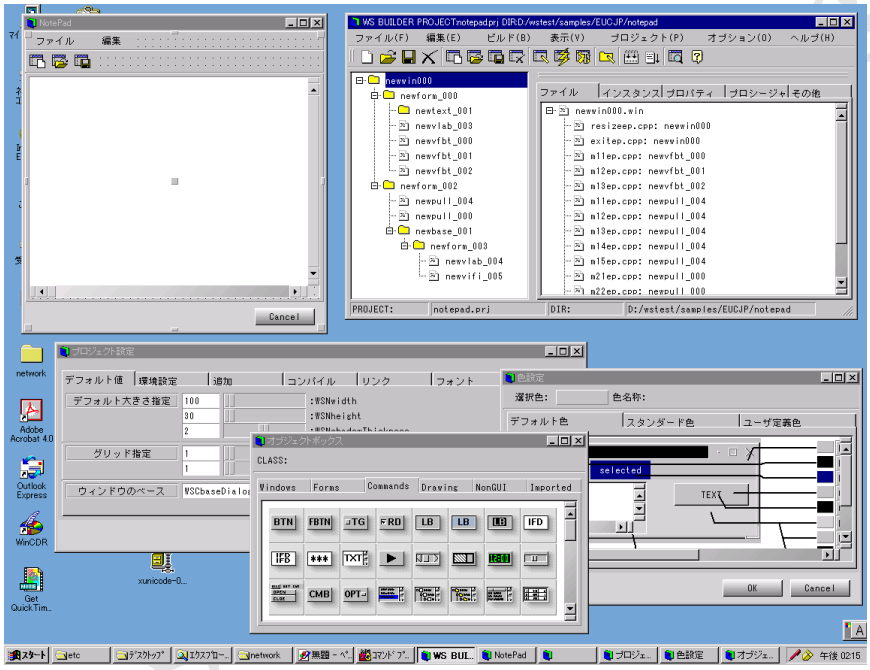
［アプリケーション開発の手順］

Manual Page

3 アプリケーションビルダの起動

3.1 Windows での起動方法

WideStudio のアイコンをクリック、またはメニューで選択するか、コマンドプロンプトで wsbuilder.exe を実行します。
すると、次のようなアプリケーションビルダー画面が立ち上がります。



［ WideStudio アプリケーションビルダー］

3.2 UNIX での起動方法

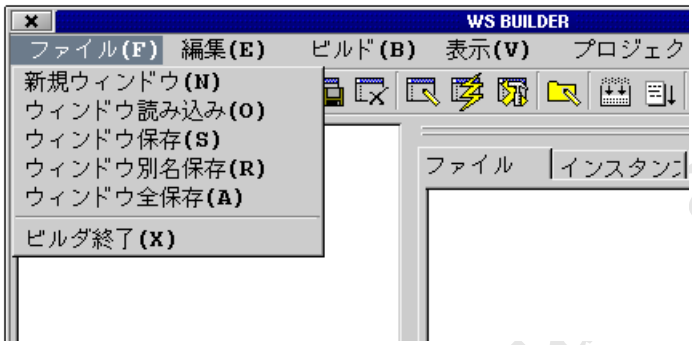
UNIX の場合、X11 ウィンドウシステムを起動した上で、コマンド wsbuilder によりアプリケーションビルダを起動することができます。C++ コンパイラがインストールされていないと、アプリケーションビルダでアプリケーションの構築作業が出来ないので注意しましょう。

［アプリケーションビルダの初期起動時の画面の状態］

Wide Studio Manual

4　アプリケーションビルダを終了するには

ピルダを終了するには、［ファイル］メニューのピルダ終了を選択します。また、ウインドウタイトル右の [×] ボタンをクリックすることでも終了させることができます。



［アプリケーションビルダの終了］

次のような確認のダイアログが表示されますので、了解を選択します。なお、保存していない編集中のものが存在する場合、保存の確認がある場合があります。

［アプリケーションビルダの終了の確認］

5 各ウィンドウの名称と機能

WideStudio アプリケーションビルダーを使うと、ビジュアルにアプリケーションを開発することができます。下記のようなビルダーで提供されている各種設定ウィンドウの機能をざっと見てみましょう。

インスペクタ

ソースインスペクタ

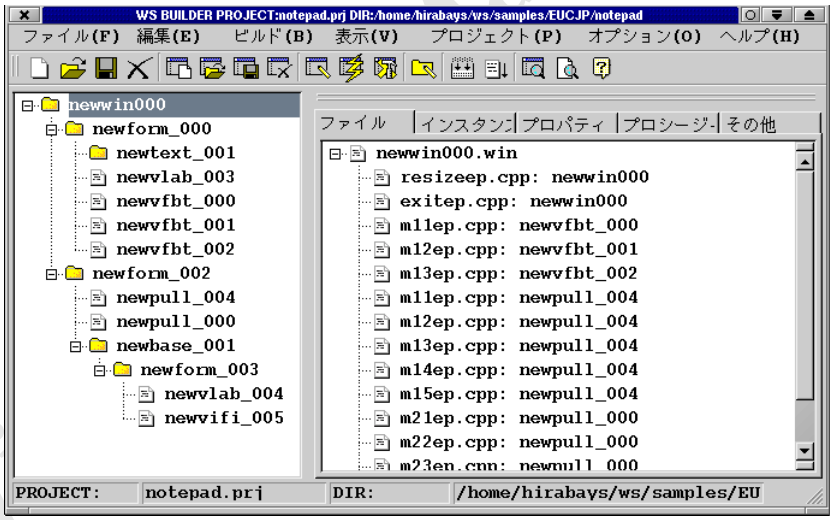
プロパティエディタ

イベントプロシージャエディタ

ウィンドウアトリビュートエディタ

5.1 インスペクタ・ソースインスペクタ

インスペクタは、作成中のアプリケーションウィンドウの構成を簡単に確認することができます。また、イベントプロシージャのソースプログラムの構成も確認できます。



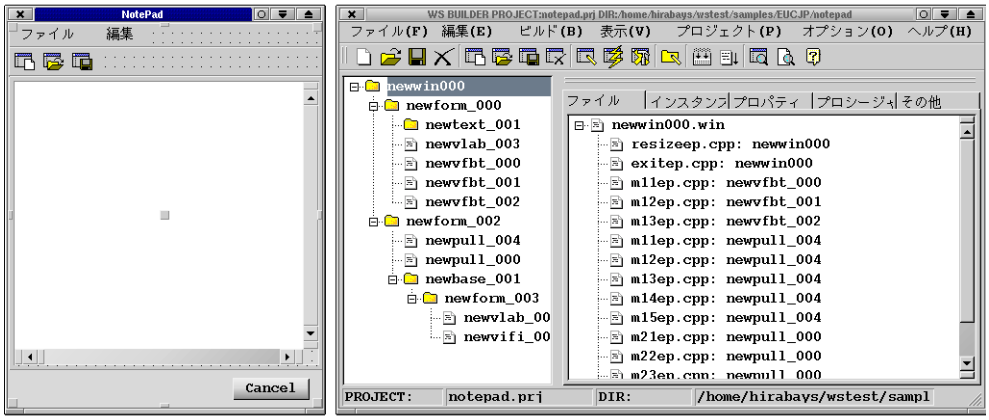
［インスタンスの構成の表示 (図左) とソースプログラムの構成の表示 (図右)］

表示されているインスタンスをマウスでクリックすると、選択された状態になります。項目の右側に、[+] や [-] が表示されている場合は、そのインスタンス上に別のインスタンスが配置されていることを示しています。その項目をダブルクリックすると、そのインスタンス上に存在する子インスタンスたちが表示されます。

Wide Studio Manual Page

5.2 プロパティエディタ

WideStudio の提供するオブジェクトは、プロパティとよばれる表示の形状や状態をいろいろと変更できる変数を持っています。そして、このプロパティは、ビルダーのプロパティーエディタで直接見ながら、編集することができます。

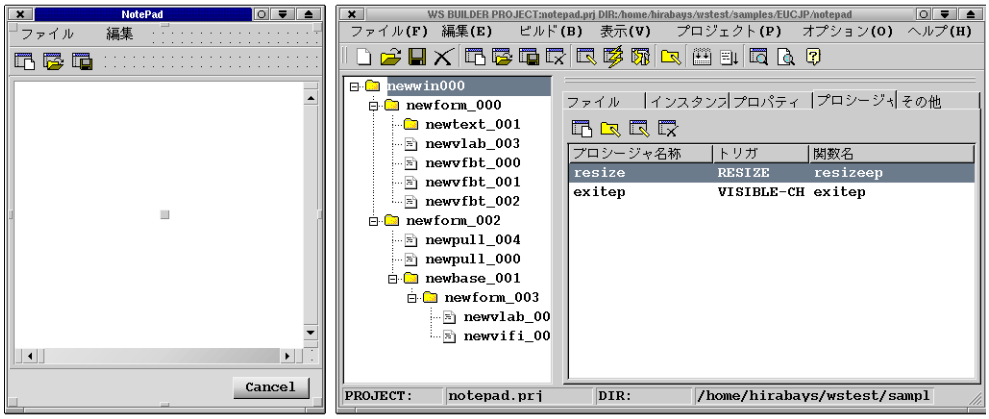


［プロパティエディタ (図右) によるプロパティの編集］

5.3 イベントプロシージャエディタ

WideStudio アプリケーションビルダーでは、イベント駆動でのプログラミングを行います。オブジェクトに対して発生したイベントをトリガーに、プロシージャを張り付けることができます。例えば、プッシュボタンに対してマウスクリックすると、ACTIVATE イベントが発生しますが、このプッシュボタンに対して ACTIVATE イベントをトリガにして、イベントプロシージャを張り付けると、マウスクリックされた時に、そのイベントプロシージャが実行されます。イベントプロシージャエディタにより、イベントプロシージャの登録、削除、編集が行えます。

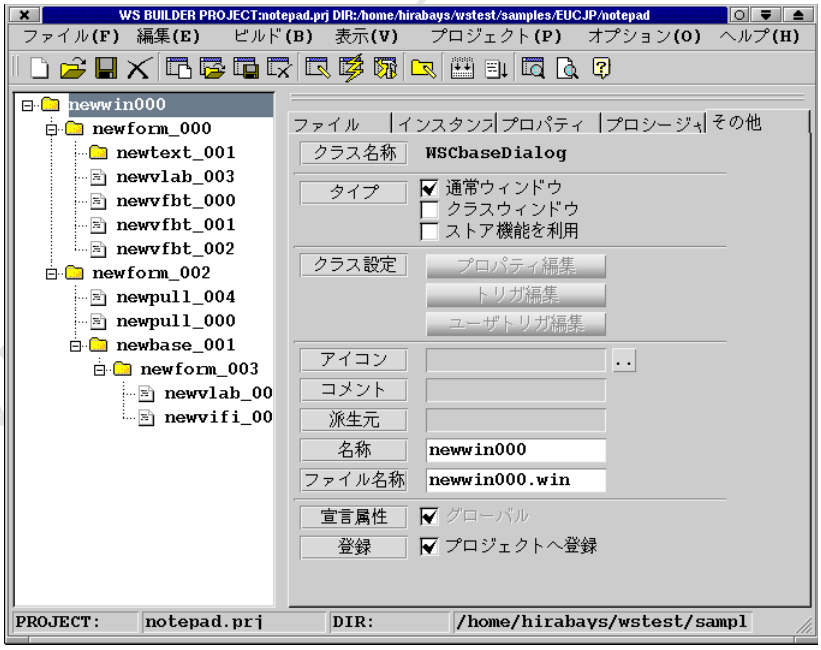
Wide Studio Ma



［イベントプロシージャエディタ (図右) によるイベントプロシージャの編集］

5.4 ウィンドウアトリビュートエディタ

ウィンドウには、ウィンドウ種別を指定したりするための各種属性を持っています。ウィンドウ属性は、ウィンドウアトリビュートエディタを使って編集します。



［ウィンドウアトリビュートエディタによる属性の設定 (図右)］

6 簡単なアプリケーション Hello の作成

WideStudio のアプリケーションビルダーを使って、簡単なアプリケーション Hello を作ってみましょう。アプリケーション Hello は、一つのボタンを持ち、このボタンを押すと、Hello! と表示します。簡単なアプリケーションを作りながら、アプリケーションビルダーの基本的な使い方を見ていきたいと思います。

ボタンを押すと Hello! と表示

［簡単なアプリケーション Hello］

次に示すような流れで hello を作成します。

hello プロジェクトの作成

hello のウィンドウの作成

イベントプロシージャの作成

hello のビルド

Wide Studio Manual Page

7 Hello プロジェクトの作成

プロジェクトの新規作成で、hello プロジェクトを作成してみましょう。アプリケーションビルダーの［プロジェクト］メニューの新規プロジェクトを選択するか、次の図に示すアイコンをクリックします。

表示されるダイアログの、プロジェクト名を hello 、作業ディレクトリにプロジェクトを格納するディレクトリを指定して、「通常のアプリケーション」を選択してください。
プロジェクトが作成されると、左下のプロジェクト名称を表示する部分「PROJECT 名称」が、hello.prj になり、hello プロジェクトになったことを確認することができます。

［Hello プロジェクトの作成］

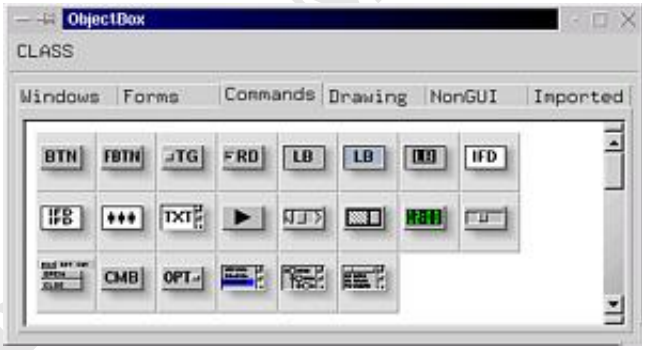
Wide Studio Manual

8 ウィンドウの作成

WideStudio では、ユーザの作成するウィンドウのことを、アプリケーションウィンドウと呼びます。hello のベースとなるアプリケーションウィンドウを一つ作成してみましょう。ビルダーの［ファイル］メニューの新規ウィンドウを選択します。すると、アプリケーションウィンドウ作成ウィザードが表示されますので、タイプは、「通常のウィンドウ」を選択し、newwin000 の名称で作成します。この場合は、ひな型選択を「なし」としましょう。

［ウィンドウ作成ウィザードダイアログ］

次は、作成したウィンドウにオブジェクトを配置してみます。配置するオブジェクトを選択するために、オブジェクトボックスを表示しましょう。［表示］メニューのオブジェクトボックスを選択します。すると次の図のようなオブジェクトボックスが表示されます。



［オブジェクトボックス］

オブジェクトボックスダイアログの Commands タブを選択して、BTN と書いたアイコン (WSCvbtn/押ボタンクラス) をアプリケーションウィンドウにドラッグアンドドロップします。

アプリケーションビルダーでは、プロパティを設定することで、オブジェクトの形状を操作することができます。この配置されたボタンオブジェクトの形を整えてみましょう。ボタンオブジェクトを選択して、アプリケーションビルダの「プロパティ」タブを選択します。そして、プロパティを次の様に設定します。

オブジェクト名称 -¿ newvbtn_000

X座標 -¿ 10

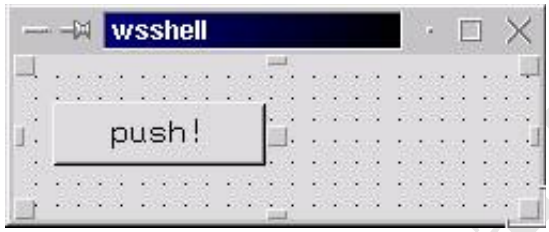
Ｙ座標 -¿ 10

横幅 -¿ 200

縦幅 -¿ 30

表示文字列 -¿ push!

すると次の図の様になります。



［作成中のウィンドウの様子］

Wide Studio Manual Page

9 イベントプロシージャの作成

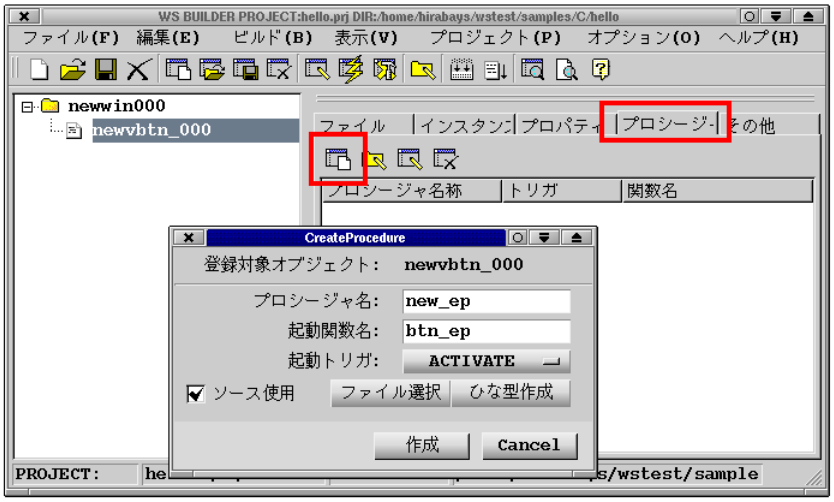
オブジェクトは実行時に、いろいろなイベントを発生します。WideStudio では、そのイベントに対し、プロシージャとよばれるプログラムを張り付けることで、プログラミングを行います。

まずは、具体的に下記の機能を持つ簡単なプロシージャを作ってみましょう。

ウィンドウ上のボタンをクリックすると「Hello!」と表示。

newvbtn_000 に対して、マウスがクリックされたら、プロシージャが実行されるように一つ、イベントプロシージャを設定してみることにします。

ウィンドウ上に配置されている newvbtn_000 を選択して、プロシージャタブを選択します。次に図のアイコンをクリックします。



［イベントプロシージャの作成］

イベントプロシージャ定義ダイアログには、次に示すように設定します。

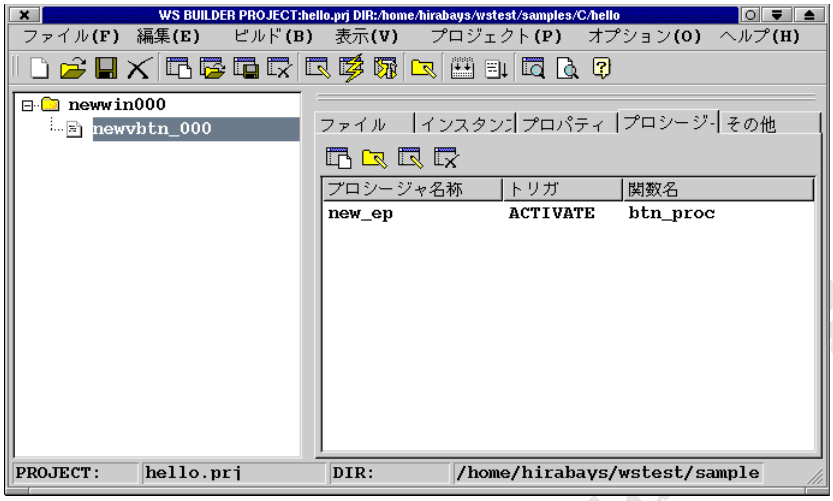
プロシージャ名 -¿ new_ep

起動関数名 -¿ btn_ep

起動トリガ -¿ ACTIVATE

プロシージャ名は、イベントプロシージャを識別するための名前です。好きな名前を付けて構いません。起動関数名は、起動される C/C++ の関数名です。この関数に処理を書きます。あとは起動トリガです。ACTIVATE トリガは、正しくボタンを押して放した時に発生します。

後は、図に示す「ひな型作成ボタン」を押して空の関数 btn_ep() がファイル名 btn_ep.cpp で作成し、「作成」ボタンを押して、イベントプロシージャを作成します。



［イベントプロシージャが作成されたところ］

図中の btn_ep をダブルクリックすると、エディタが起動して、関数が編集できるようになります。使用するエディタは、デフォルトでは、vi になっていますが、[プロジェクト] メニューのプロジェクト設定の環境設定で、お好みのエディタが指定できます。ではまず、ボタンに「Hello!」と表示するようにプロシージャを書いてみます。

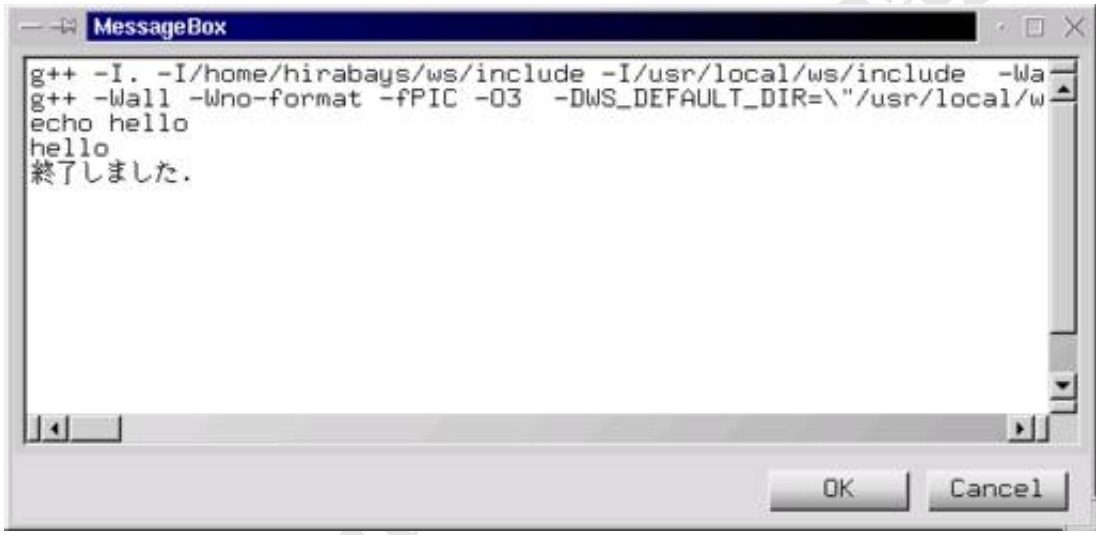
```
\#include <WScom.h>
\#include <WSCfunctionList.h>
\#include <WSCbase.h>
//-----------------------------------------------
//Function for the event procedure
//-----------------------------------------------
void btn\_ep(WSCbase* object){
  object->setProperty(WSNlabelString,"Hello!"); //A
}
static WSCfunctionRegister op("btn\_ep1",(void*)btn\_ep1);
```

A に示す行は、イベントプロシージャを張り付けたオブジェクト (この場合は、newvbtn_000 ボタンオブジェクト) のプロパティ WSNlabelString(表示文字列) を 、「Hello!」に設定しています。

Wide Studio Manual

10 Hello のビルドと実行

ここで、作成したものが失われないように、保存しておきましょう。［プロジェクト］メニューの［プロジェクトの保存］を選択し、プロジェクトを保存します。
次はいよいよ、アプリケーションのコンパイルです。アプリケーションビルダーの［ビルド］メニューのビルドオールを選択します。コンパイルメッセージダイアログが出力され、コンパイルの様子が表示されます。これでエラーが出なければ、hello は完成です。ビルドが成功すると、プロジェクトのディレクトリに hello が生成されます。［ビルド］メニューの実行を選択するか、コマンドラインから実行してみてください。



［Hello のビルド］

Wide Studio Manual

C　アプリケーションウィンドウ編

1　アプリケーションウィンドウとは

アプリケーションビルダで作成したウィンドウをアプリケーションウィンドウと呼びます。そしてアプリケーションウィンドウ単位で管理され、ファイルに保存します。

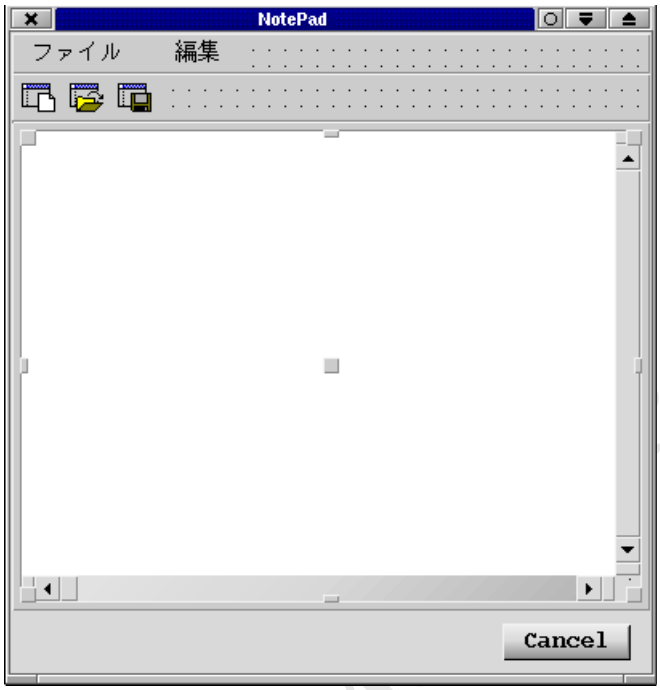
アプリケーションウィンドウは、ボタンや、エリア、テキスト入力フィールドなど、ＧＵＩオブジェクトを配置し、ウィンドウを構成する機能を持ちます。アプリケーションウィンドウには、次のようなタイプがあります。

- アプリケーションウィンドウ
  通常アプリケーションで利用するウィンドウは、アプリケーションウィンドウです。そのウィンドウ上に、エリアオブジェクトやプッシュボタンなどコントロールオブジェクトを配置することで作成します。

- クラスウィンドウ
  クラスウィンドウは、アプリケーションでよく利用されるアプリケーションウィンドウや、組み合わせた複合オブジェクトをＣ＋＋クラスにして (派生させて) 一つのオブジェクトとして利用できるようにします。アプリケーション専用のオブジェクトを作成して生産性を向上したり、再利用性を向上させるなどの効果を発揮します。単一のオブジェクトを派生させて、新しいオブジェクトを作成したり、アプリケーションウィンドウなどの複合したものを新しい一つのオブジェクトとして作成することができます。
  (詳しくはクラスウィンドウの節を参照してください)

- ストアウィンドウ
  ストアウィンドウは、オンラインストア形式に保存され、オンライン実行時にアプリケーションからロードして、オブジェクトを生成することができます。このストア機能を利用するとアプリケーション間でのオブジェクトの共有や、オンラインでのオブジェクトデータの入れ換え等を実現することができます。
  (詳しくはオンラインストア機能の節を参照してください)

Wide Studio Manual Page

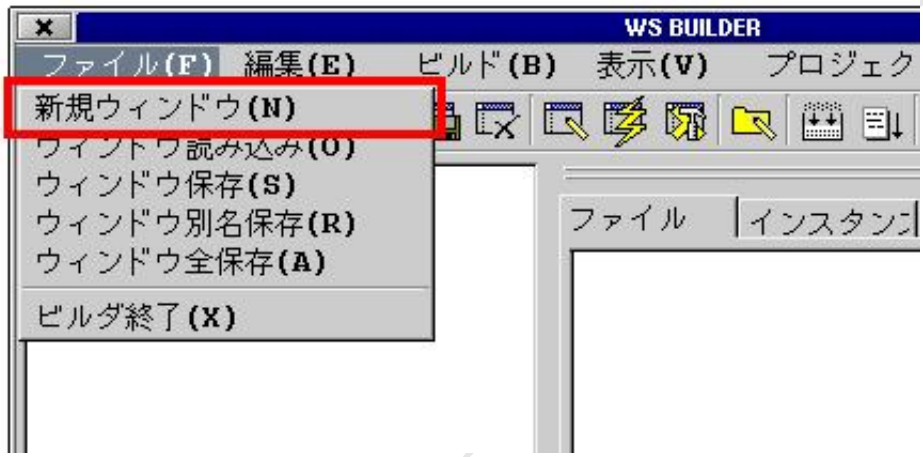


［作成中のアプリケーションウィンドウの例］

Wide Studio Manual 1.0 Page

2　アプリケーションウィンドウを新規作成/保存するには

新規アプリケーションウィンドウを作成するには、［ファイル］メニューの [新規ウィンドウ] を選択します。
すると、ウィンドウ作成ウィザードが表示されます。ウィンドウのタイプ、ウィンドウに付ける名称等が指定できます。ひな型を指定することで、あらかじめオブジェクトを配置してある定型のウィンドウを作成することができます。それらの入力の後、[生成] ボタンを選択すれば完了です。



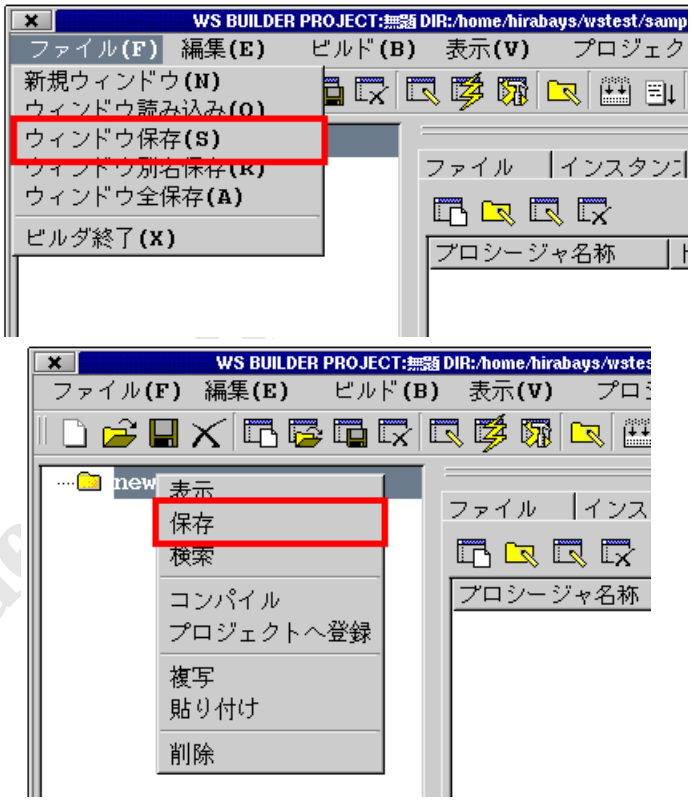
［新規アプリケーションウィンドウの作成］

［ウィンドウ作成ウィザード（タイプの指定）］

［ウィンドウ作成ウィザード（名称の指定）］

［ウィンドウ作成ウィザード（ひな型の指定）］

アプリケーションウィンドウを保存するには、保存したいアプリケーションウィンドウを、インスペクタで選択して、アプリケーションビルダの［ファイル］メニューの [ウィンドウ保存] を選択するか、インスペクタのオブジェクト上でのマウスの右ボタンメニューの [保存] を選択します。
保存されるファイル名称は、ウィンドウ名.win です。

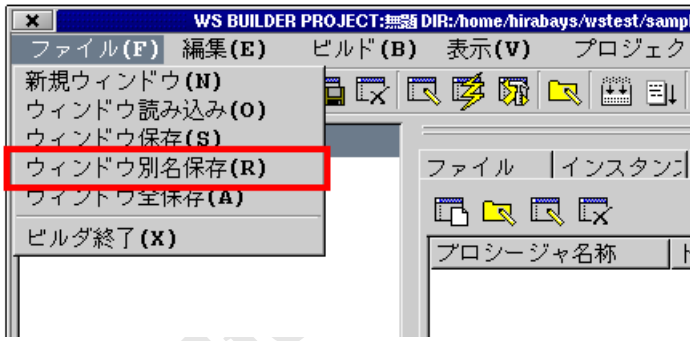


［アプリケーションウィンドウの保存］

保存したいアプリケーションウィンドウを選択しないで、保存すると、次のように警告がでます。
一括で保存したい場合は、アプリケーションビルダの［ファイル］メニューの [ウィンドウ全保存] を選択してください。

［アプリケーションウィンドウの保存］

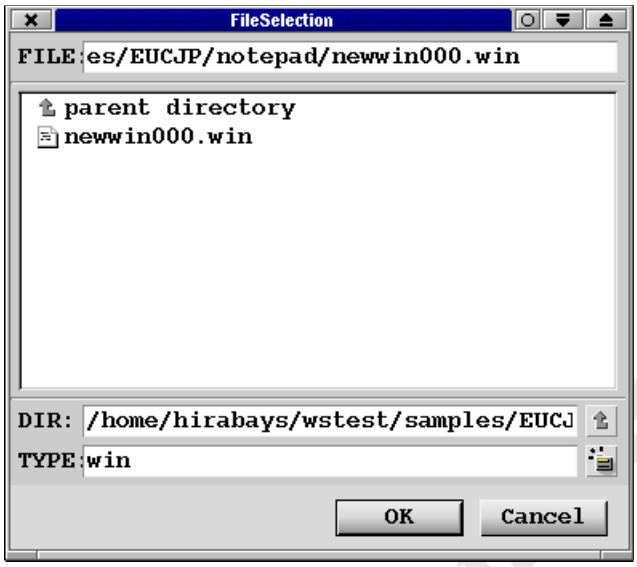
また、アプリケーションビルダの [ファイル] メニューの [ウィンドウ別名保存] を選択すると、別のファイル名で保存することができます。



［ウィンドウ別名保存メニュー］

図の示す所にファイル名称を入力して下さい。

Wide Studio Manual Page



［ファイル名称の指定］

Wide Studio Manual Page

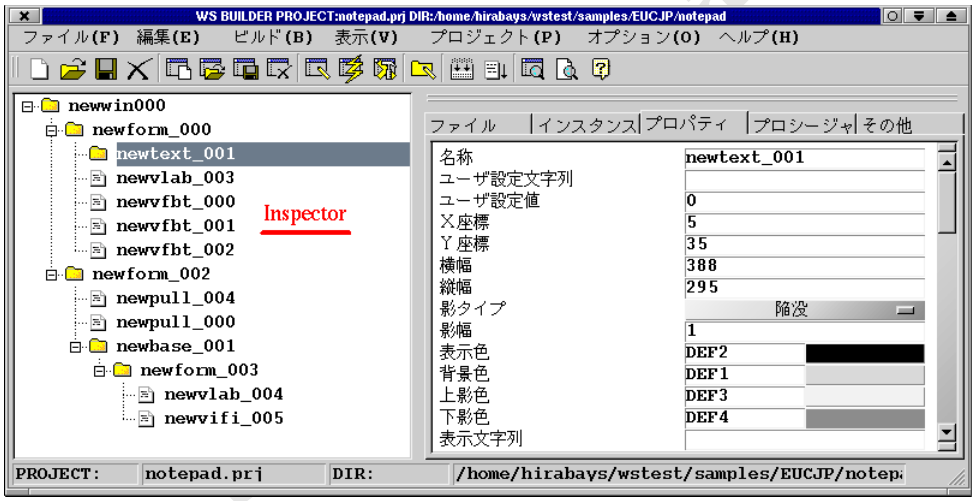
3　アプリケーションウィンドウの一覧を確認するには

3.1　アプリケーションウィンドウの一覧を確認するには

アプリケーションビルダで読み込みしたアプリケーションウィンドウの一覧を表示しコントロールするウィンドウがインスペクタです。

アプリケーションウィンドウを生成したり、読み込んだりすると表示されます。インスペクタは、アプリケーションウィンドウ上に配置されている GUI オブジェクトの構成をわかりやすくツリー状に表示します。

ツリー上に表示された一番上位のオブジェクトは、アプリケーションウィンドウを示します。例えば、図中の newwin000 は、アプリケーションウィンドウです。

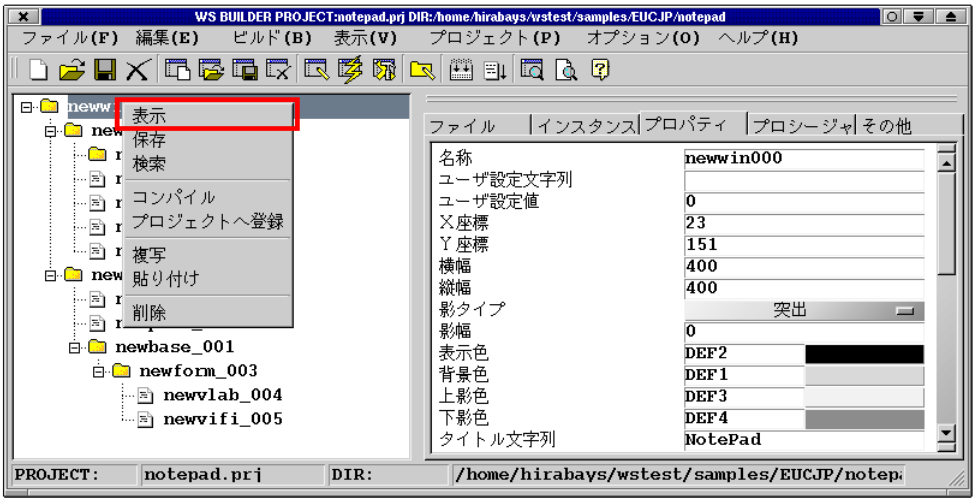


［アプリケーションウィンドウの構成を表示したところ（図左側：インスペクタ）］

3.2　アプリケーションウィンドウを表示させるには

アプリケーションウィンドウを表示するには、インスペクタ上でマウスの右ボタンを押し、[表示] メニューを選択します。

また、オブジェクトをインスペクタ上で選択しておき、アプリケーションビルダの [編集] メニューの表示を選択しても表示できます。非表示としたい場合は、マウスの右ボタンメニューで非表示を選択してください。



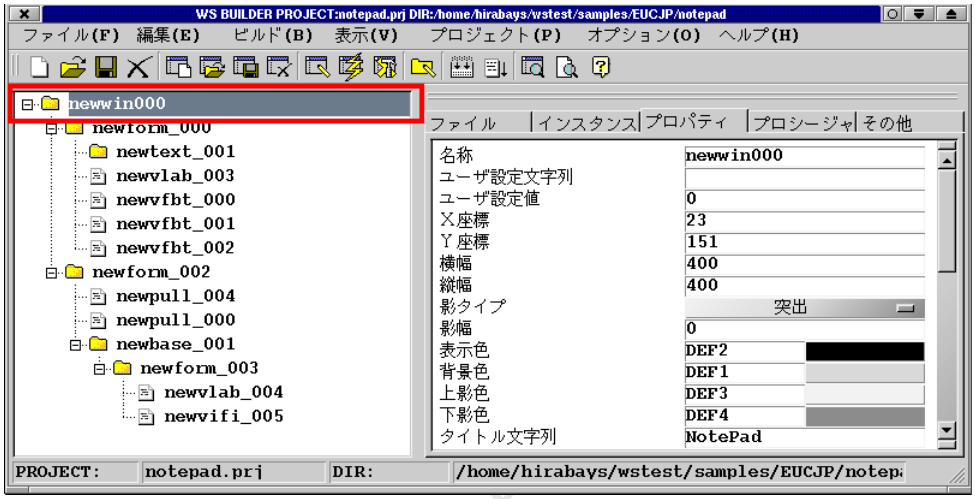
［アプリケーションウィンドウの表示/非表示］

Wide Studio Manual

4 アプリケーションウィンドウを削除するには

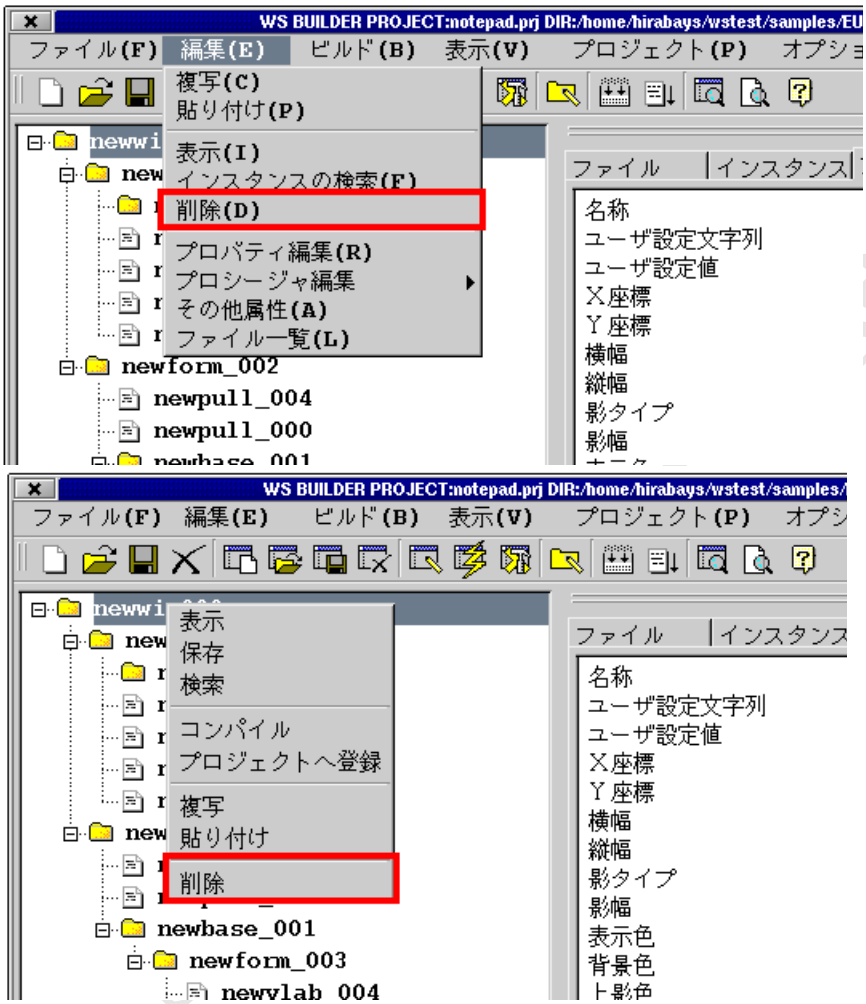
アプリケーションウィンドウを削除するには、まず、図に示すように削除したいアプリケーションウィンドウを選択状態にします。

アプリケーションビルダの [編集] メニューの [削除] を選択するか、インスペクタ上でのマウスの右ボタンメニューの [削除] を選択します。



［削除したいアプリケーションウィンドウの選択］

Wide Studio

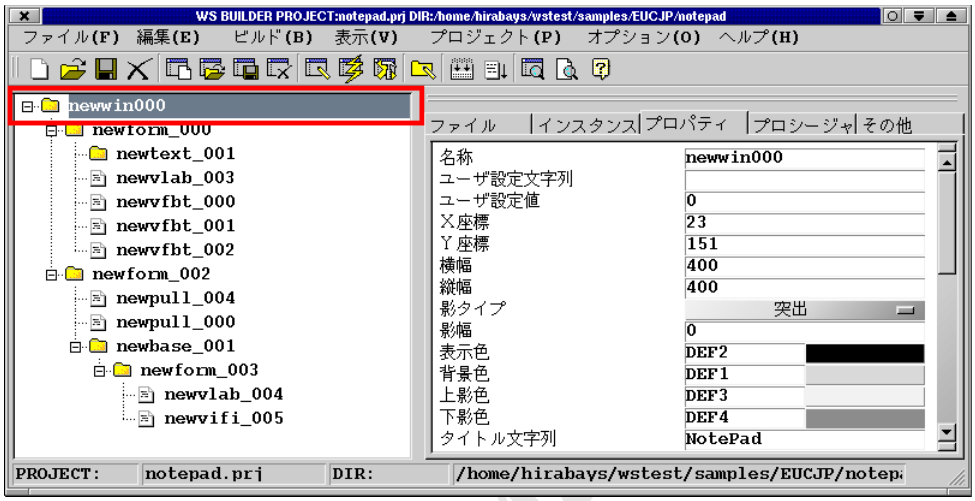


［アプリケーションウィンドウの削除］

一度、削除していいかどうか質問ダイアログがでますので、本当に削除する場合は、了解を選択してください。

5　アプリケーションウィンドウ名を変更するには

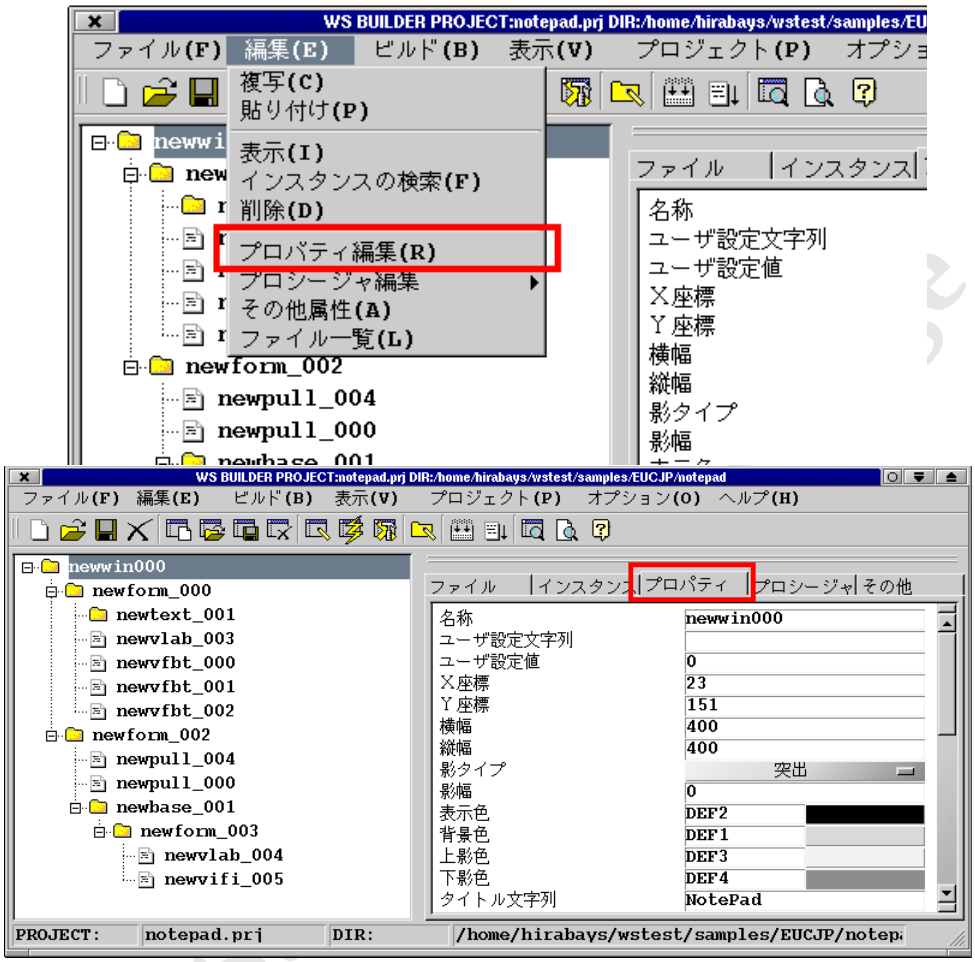
アプリケーションウィンドウの名称を変更するには、［プロパティ編集］、または、［基本設定］で変更する二通りの方法があります。まず、変更したいアプリケーションウィンドウを選択します。



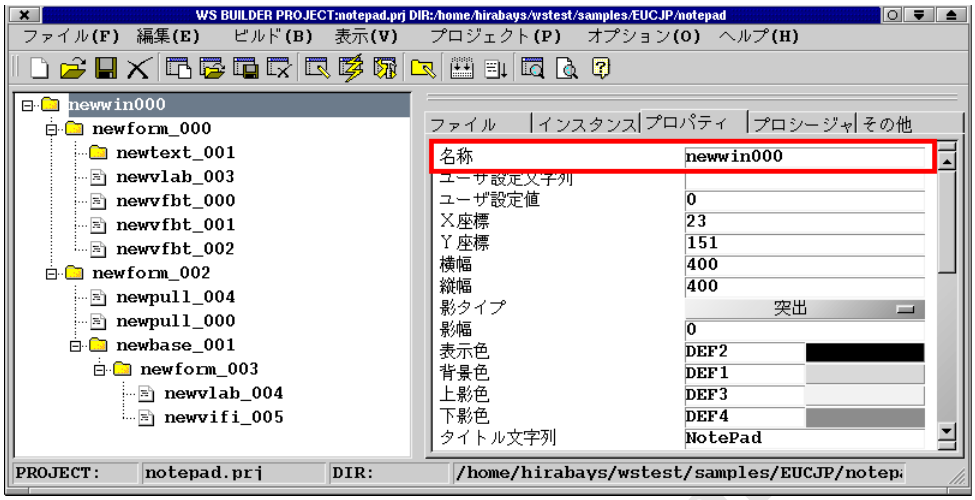
［アプリケーションウィンドウの選択］

プロパティ編集による名称変更は、次の様に行います。
アプリケーションビルダの [編集] メニューの［プロパティ編集］か、インスペクタの［プロパティ］タブを選択します。
すると、プロパティ一覧が表示され、一番上の、［名称］を変更すると完了です。
(注意) アプリケーションウィンドウに用いられる名称は、トップのウィンドウオブジェクトの名称が使われています。

Wide Studio Manual


WS BUILDER PROJECT:notepad.prj DIR:/home/hirabays/wstest/samples/EU
ファイル(F) 編集(E) ビルド(B) 表示(V) プロジェクト(P) オプショ
複写(C)
貼り付け(P)
表示(I)
インスタンスの検索(F)
削除(D)
プロパティ編集(R)
プロシージャ編集
その他属性(A)
ファイル一覧(L)
newwi
new
newform_002
newpull_004
newpull_000
ファイル
インスタンス
名称
ユーザ設定文字列
ユーザ設定値
X座標
Y座標
横幅
縦幅
影タイプ
影幅
WS BUILDER PROJECT:notepad.prj DIR:/home/hirabays/wstest/samples/EUCJP/notepad
ファイル(F) 編集(E) ビルド(B) 表示(V) プロジェクト(P) オプション(O) ヘルプ(H)
newwin000
newform_000
newtext_001
newvlab_003
newvfbt_000
newvfbt_001
newvfbt_002
newform_002
newpull_004
newpull_000
newbase_001
newform_003
newvlab_004
newvifi_005
ファイル
インスタンス
プロパティ
プロシージャ
その他
名称 newwin000
ユーザ設定文字列
ユーザ設定値 0
X座標 23
Y座標 151
横幅 400
縦幅 400
影タイプ 突出
影幅 0
表示色 DEF2
背景色 DEF1
上影色 DEF3
下影色 DEF4
タイトル文字列 NotePad
PROJECT: notepad.prj DIR: /home/hirabays/wstest/samples/EUCJP/notepa

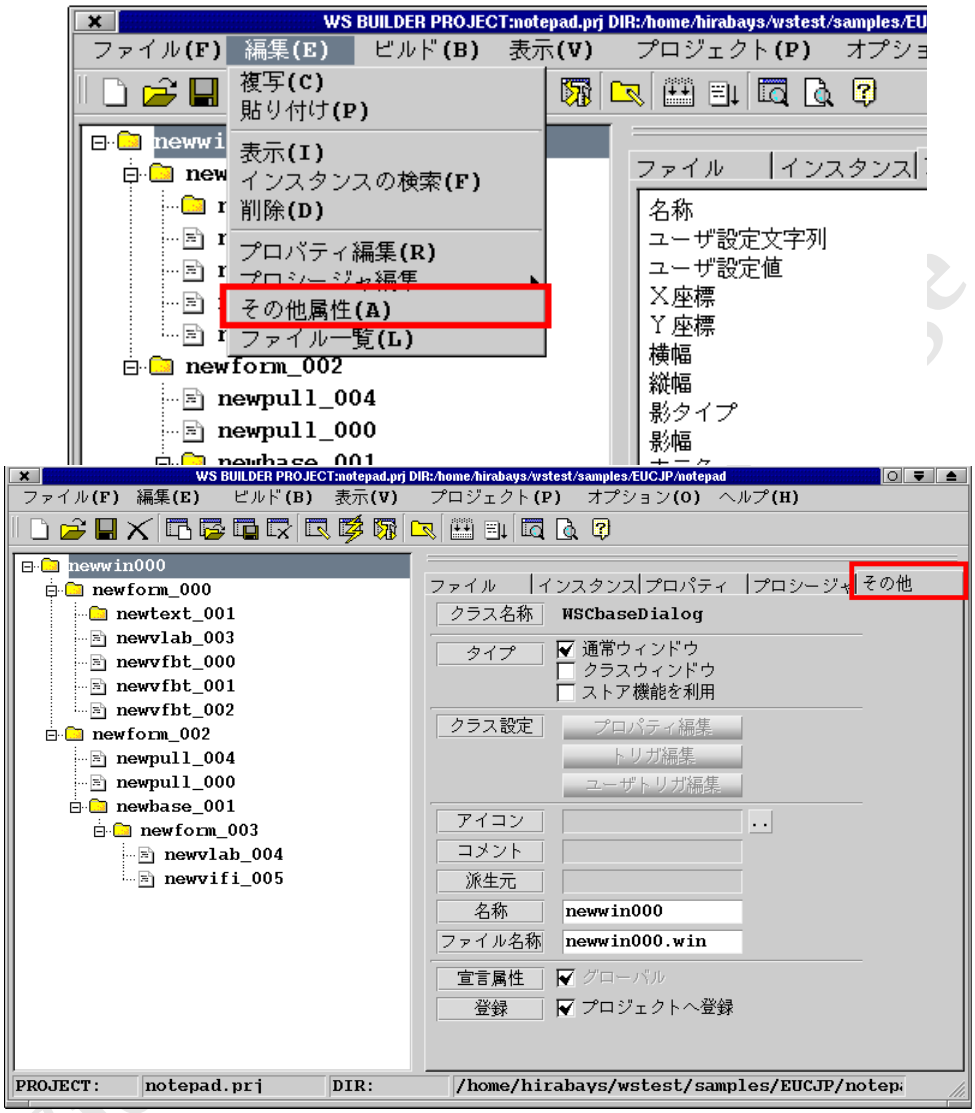
［プロパティ一覧の表示］



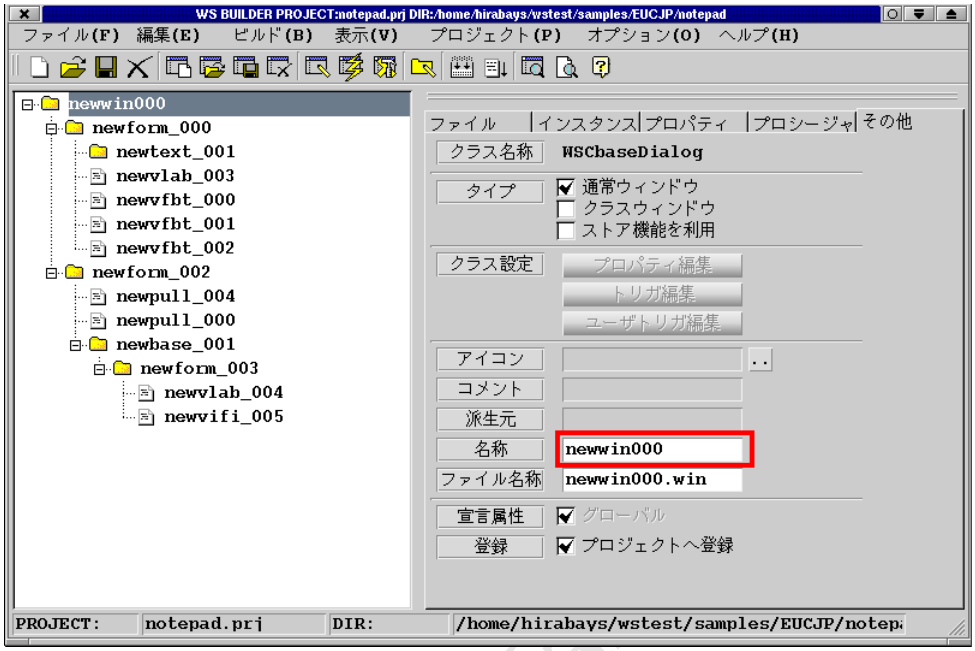
［名称プロパティによる編集］

基本設定による名称変更は、次の様に行います。
アプリケーションビルダの［編集］メニューの［基本設定］か、インスペクタの［基本設定］タブを選択します。
すると、基本設定が表示され、図中の［名称］を変更し、リターンキーを押すか、［値の反映］ボタンをクリックすると完了です。
(注意) 基本設定の名称は、名称プロパティと同一です。したがって、同時に名称プロパティも変化します。

Wide Studio Manual


WS BUILDER PROJECT:notepad.prj DIR:/home/hirabays/wstest/samples/EU
ファイル(F)
編集(E)
ビルド(B)
表示(V)
プロジェクト(P)
複写(C)
貼り付け(P)
表示(I)
インスタンスの検索(F)
削除(D)
プロパティ編集(R)
その他属性(A)
ファイル一覧(L)
newform_002
newpull_004
newpull_000
ファイル
インスタンス
名称
ユーザ設定文字列
ユーザ設定値
X座標
Y座標
横幅
縦幅
影タイプ
影幅
WS BUILDER PROJECT:notepad.prj DIR:/home/hirabays/wstest/samples/EUCJP/notepad
ファイル(F) 編集(E) ビルド(B) 表示(V) プロジェクト(P) オプション(O) ヘルプ(H)
newwin000
newform_000
newtext_001
newvlab_003
newvfbt_000
newvfbt_001
newvfbt_002
newform_002
newpull_004
newpull_000
newbase_001
newform_003
newvlab_004
newvifi_005
ファイル
インスタンス
プロパティ
プロシージャ
その他
クラス名称
WSCbaseDialog
タイプ
通常ウィンドウ
クラスウィンドウ
ストア機能を利用
クラス設定
プロパティ編集
トリガ編集
ユーザトリガ編集
アイコン
コメント
派生元
名称
newwin000
ファイル名称
newwin000.win
宣言属性
グローバル
登録
プロジェクトへ登録
PROJECT:
notepad.prj
DIR:
/home/hirabays/wstest/samples/EUCJP/notep

［基本設定一覧の表示］

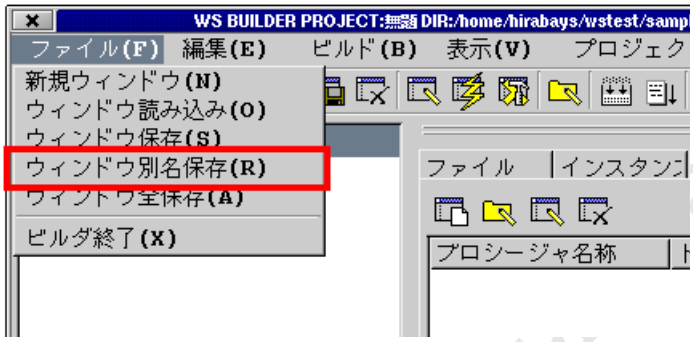


［基本設定による編集］

Wide Studio Ma

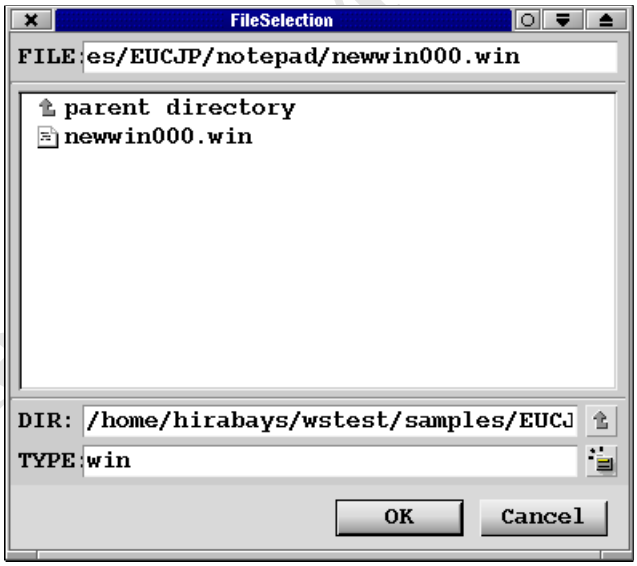
6　ファイル名称を指定して保存するには

保存したいアプリケーションウィンドウを選択状態にして、アプリケーションビルダの [ファイル] メニューの [ウィンドウ別名保存] を選択すると、別のファイル名で、保存することができます。



［ウィンドウ別名保存メニュー］

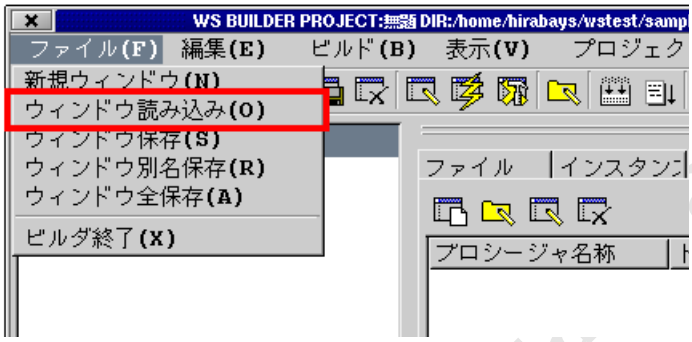
図の示す所にファイル名称を入力して下さい。



［ファイル名称の指定］

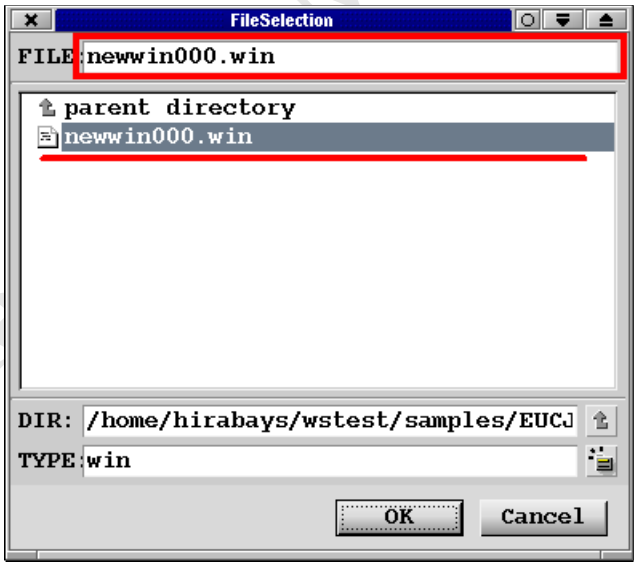
7　作成済みのアプリケーションウィンドウを読み込むには

作成済みのアプリケーションウィンドウを読み込むには、［ファイル］メニューの [ウィンドウ読み込み] を選択します。



［アプリケーションウィンドウの読み込み］

次の図のようにファイル選択ウィンドウが表示されます。読み込みたいアプリケーションウィンドウのファイルを選択します。



［ファイルの選択］

直接、ファイル名称を入力するか、ファイルをダブルクリックして選択します。

D　オブジェクト編

1　オブジェクトとは

オブジェクトは、ウィンドウやプッシュボタンなどの部品を指します。オブジェクトは C＋＋オブジェクトからなり、利用者が簡単にオブジェクトをカスタマイズ出来るよう、次の様な機能を持っています。

- プロパティ
  オブジェクトはプロパティ(一種のメンバ変数) を持ち、統一インターフェースで値を指定できます。プロパティには、色や大きさを指定するもの、形状、数値、文字列を指定するものなどがあります。
  またクラスウィンドウを利用すると、新たにプロパティを追加したオブジェクトを作成することもできます。詳しくはクラスウィンドウのプロパティの追加の節を参照下さい。

- イベントプロシージャ
  オブジェクトはイベントプロシージャ(後節を参照下さい) を持つことが出来ます。イベントプロシージャによって、更なる機能追加や動作の追加を行うことができます。イベントプロシージャによって不必要なクラス派生を抑え、簡単にオブジェクトをカスタマイズして使用できるようになっています。もちろん、クラス派生によって新型オブジェクトの生成 (クラスウィンドウの節を参照下さい) を行うこともできます。

オブジェクト (マネジメント機能を持つもの) には主に次の様なものがあります。

分類	クラス名称	説明
ウィンドウ	WSCwindow	ウィンドウでアプリケーションウィンドウのベースとなります。
	WSCbaseDialog	ダイアログです。
フォーム	WSCform	ウィンドウ内に各種コントロールを配置する矩形エリアです。
	WSCscrForm	スクロール機能を有したエリアです。
	WSCsform	セパレータによって複数の領域の大きさを自由に変えることのできる矩形エリアです。
	WSCindexForm	インデックスタブによる切替えができる矩形エリアです。

オブジェクト (マネジメント機能を持たないもの) には主に次の様なものがあります。

Wide Studio Manual Page

分類	クラス名称	説明
ラベル・ボタン	WSCvbtn	プッシュボタンです。
	WSCvtoggle	トグルボタンです。
	WSCvlabel	矩形枠を持ったテキスト表示オブジェクトです。
コントロール	WSCpulldownMenu	プルダウンメニューです。
	WSCoption	オプションメニューです。値を選択します。
	WSCpopupMenu	ポップアップメニューです。
	WSCvslider	スライダーです。値を選択するのに便利です。
	WSCvclock	時計を表示します。
	WSClist	テキストをリスト表示します。スクロール機能を有します。
	WSCvifield	キーボードからテキスト入力を行うオブジェクトです。
	WSCvmifield	複数行入力を行うオブジェクトです。
	WSCtextField	スクロールバーを持つ、複数行入力を行うオブジェクトです。
	WSCcomboBox	キーボード入力と、選択メニューの両方の機能を持ちます。
基本描画	WSCvarc	円弧を表示します。
	WSCvrect	矩形を表示します。
	WSCvline	折れ線を表示します。
	WSCvpoly	多角形を表示します。

Wide Studio Manual Page

2 ウィンドウ

ウィンドウ部品は普段トップシェルウィンドウとして利用されるウィンドウと、利用者に入力を催促するためのダイアログとに大きく分けることができます。
通常、WSCmainWindow クラスがトップウィンドウに使われますが、プロジェクト設定画面のデフォルト値設定で、ダイアログ等、トップウィンドウとして使用したいクラス名を指定することもできます。

WSCmainWindow
通常のトップシェルウィンドウとして用います。デフォルトでウィンドウでアプリケーションウィンドウを新規作成すると、このクラスのインスタンスが作成されます。

WSCdialog
ダイアログのベースとなるウィンドウです。ダイアログには、「OK」「NO」ボタン等があらかじめ用意されており、必要に応じてこれらを表示し、フォーカスをこのウィンドウに集中することができます。

WSCmessageDialog
メッセージの表示用ダイアログです。メッセージを表示し、利用者に注意を促すのに利用します。

WSCfileSelect
ファイル選択ダイアログです。ファイルやディレクトリを利用者に選択するのに用います。

WSCwizardDialog
画面きり変え機能のついたダイアログです。「¡ 戻る」「次へ ¿」ボタン等が付いており、それらをクリックすると、順次画面が切り替わっていくのが特徴です。

Wide Studio Manual Page

3 フォーム

フォームは、そのフォーム上に別のオブジェクトを配置し、親ウィンドウとは別のウィンドウ領域として管理します。フォームが移動すると、そのフォーム上のオブジェクトはフォームと共に移動します。また、フォームが不可視化されると、共に不可視状態となります。

WSCform
通常のフォームとして用いられます。このフォームは、親ウィンドウとは別座標を提供し、リサイズイベントを発生します。リサイズイベントが発生した場合は、そのイベントに対するイベントプロシージャでユーザがアンカープロパティ等では対応できない複雑な再配置が行うことができます。

WSCscrForm
スクロール機能をもったフォームです。実際よりも大きなエリアを表示するのに用いられます。スクロールバーを用いて、そのエリアをスクロールします。エリアの大きさ、スクロールする単位、をいろいろプロパティで設定できるようになっています。

WSCsform
領域を区分するセパレータを持つフォームです。このセパレータはマウスにより自由に動かすことができ、領域の大きさを自由に変えることができます。

WSCindexForm
インデックスタブを持ったフォームです。インデックスを選択すると、その選択された領域が表示されます。何層にも重なったエリアを持っているかのように振舞うフォームです。

Wide Studio Manual Past

4 コマンド

コマンドコントロールは、主に表示に使われるもの、ユーザがマウスによる操作でイベントを発生されるもの (プッシュボタン系のもの) 値を選択するもの、(値を選択させるもの) メニューで動作を選択するもの (プルダウンメニュー系のもの)、キーボードで文字列を入力、編集するもの (プルダウンメニュー系のもの) があります。

表示系
WSCvlabel(文字列、画像の表示)、WSCvclock(時計の表示)、
WSCvmeter(メータの表示)、

プッシュボタン系
WSCvbtn、WSCvfbtn
マウスでクリックすると ACTIVATE イベントを発生させます。

キーボード入力系
WSCvifield、WSCvmifield、WSCtextField
いずれもテキスト入力を行うのに使用します。

値選択系
WSCvradio、WSCoption、WSCcomboBox、
WSClist、WSCvslider、
値を選択するのに用います。

プルダウンメニュー系
WSCpulldownMenu、WSCpopupMenu、
メニュー項目に対し、イベントプロシージャを張り付けておくと、選択された場合その、プロシージャが動作します。

Wide Studio Manual Page

5 図形オブジェクト

図形コントロールは、図形の表示をおこなうものです。

円、楕円、円弧、塗りつぶし円
WSCvarc

線、折れ線
WSCvline

矩形、塗りつぶし矩形
WSCvrect

多角形、塗りつぶし多角形
WSCvpoly

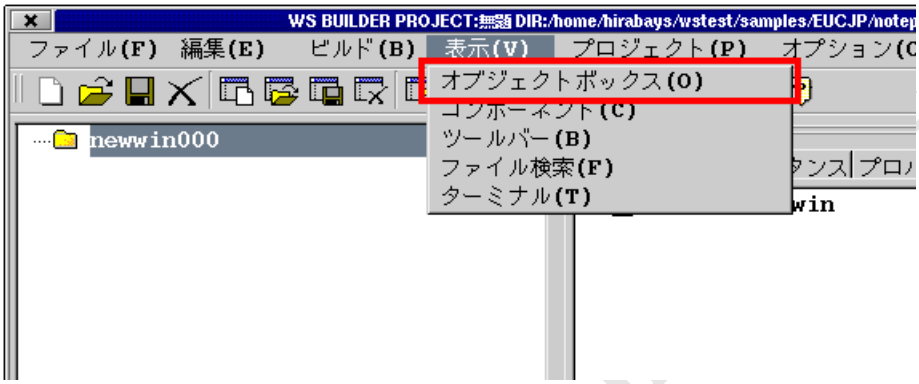
自由な図形の描画
WSCvdrawingArea

グラフ
WSCvbarGraph(棒グラフ)、WSCvlineGraph(線グラフ)、
WSCvgraphScale(グラフの目盛)、WSCvgraphMatrix(グラフの格子)

Wide Studio Manual Page

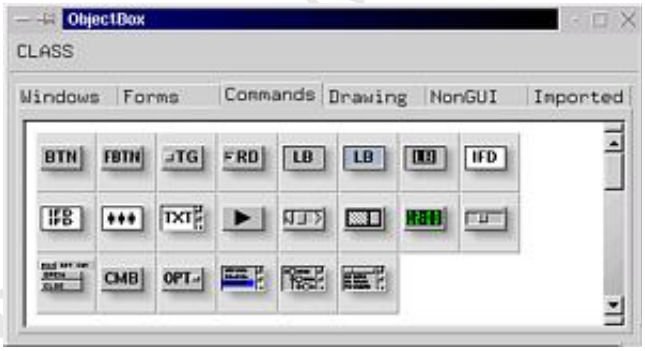
6　アプリケーションウィンドウ上にオブジェクトを配置するには

オブジェクトを配置するためのダイアログが、オブジェクトボックスです。オブジェクトボックスは、［表示］メニューのオブジェクトボックスを選択します。



［オブジェクトボックスの表示］

次の図のようなオブジェクトボックスウィンドウが表示されます。

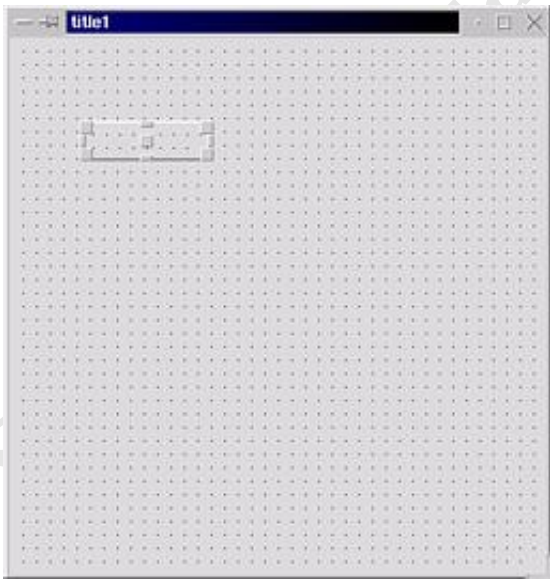


［オブジェクトボックス］

各アイコン上にマウスポインタを持っていくと、オブジェクトのクラス名と簡単な説明 (バルーンヘルプ) がでます。好みのオブジェクトを配置する前に、まず配置先のアプリケーションウィンドウを表示しましょう。アプリケーションウィンドウの表示の方法は、アプリケーションウィンドウの表示の節を参照下さい。アプリケーションウィンドウの表示が成功したら、配置したいオブジェクトに該当するオブジェクトボックス内のアイコンをクリックしたまま移動し、そのアプリケーションウィンドウ上に配置します。

オブジェクトは、次の様に種類別に、分類されています。

分類	説明
Windows	ウィンドウや、ダイアログなどです。枠を持った独立したウィンドウです。
Forms	他のオブジェクトを配置するマネジメント機能を持った矩形エリアです。 ウィンドウ資源を持ちます。
Commands	プッシュボタン等、個別の機能を持った部品です。ウィンドウ資源を持ちません
Drawing	図形や、グラフ等の主に表示を行う部品です。
NonGUI	タイマー等、実行時には表示されない部品です。
Imported	追加モジュールによって追加された部品です。



［プッシュボタンをアプリケーションウィンドウ上に配置したところ］

オブジェクトを内部に配置可能なオブジェクトはマネージャオブジェクトと呼びます。マネージャオブジェクトには、主に次のようなものがあります。

分類	クラス名称	説明
ウィンドウ	WSCmainWindow	アプリケーションウィンドウのベースとなります。 メインのウィンドウとして利用されます。
	WSCwindow	アプリケーションウィンドウのベースとなります。
	WSCdialog	ダイアログです。
フォーム	WSCform	ウィンドウ内に各種コントロールを配置する矩形エリアです。
	WSCscrForm	スクロール機能を有したエリアです。
	WSCsform	セパレータによって複数の領域の大きさを自由に変えることのできる矩形エリアです。
	WSCindexForm	インデックスタブによる切替えができる矩形エリアです。

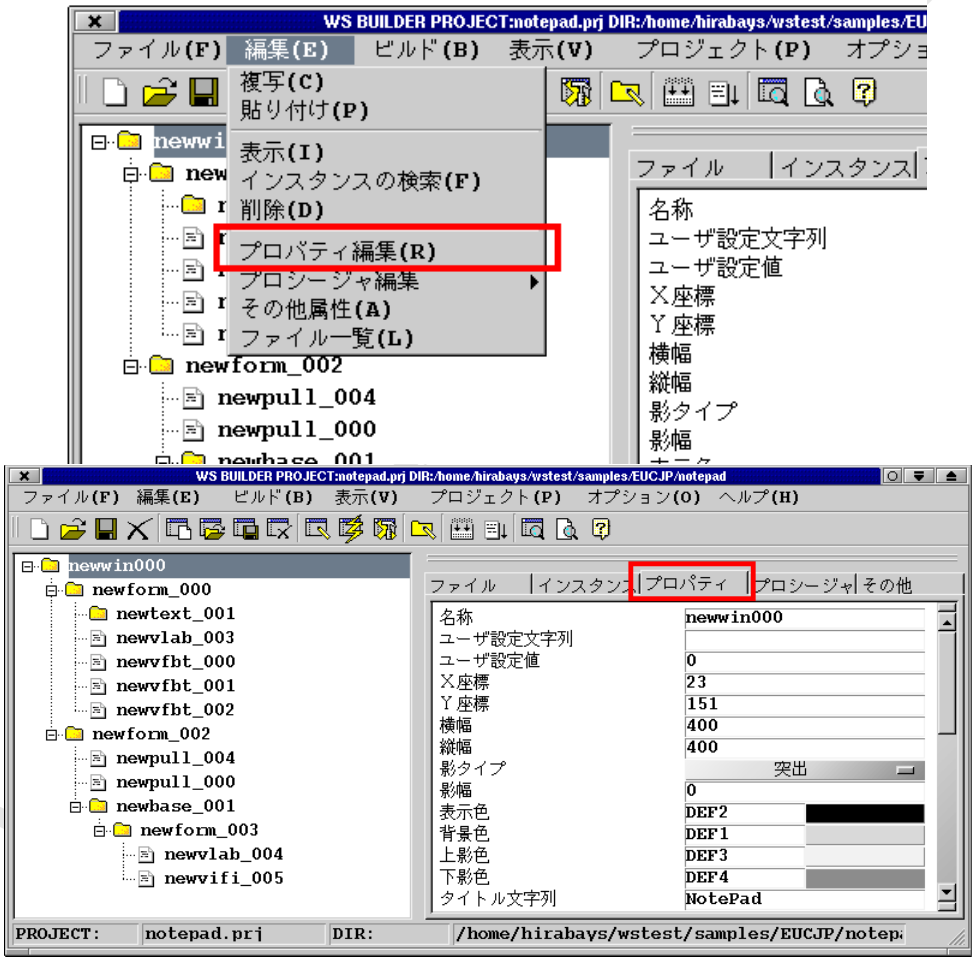
マネージャオブジェクトは、編集モードのとき (ビルダで編集中のとき) は、ドットマトリックス状態で表示されます。ドットとドットの間隔は 10 ドットです。

Wide Studio Manual Page

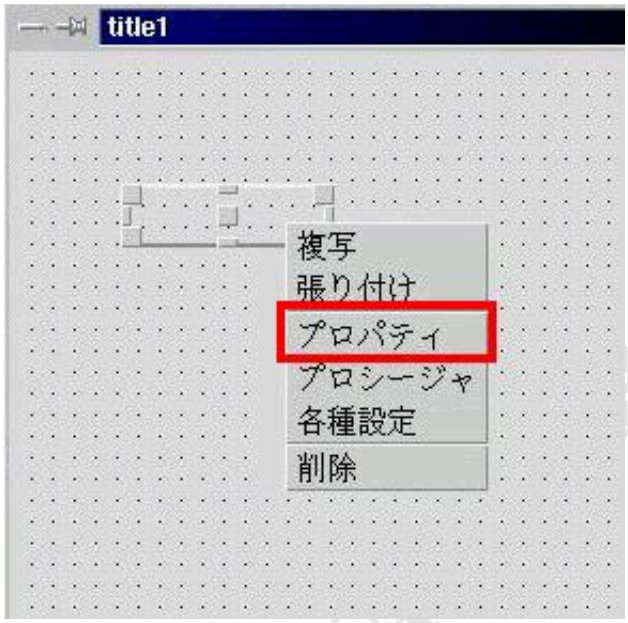
7　プロパティを設定するには

7.1　通常のプロパティを設定するには

オブジェクトにはプロパティとよばれる一種のメンバ変数を内部に持っています。その値を自由に設定することで、いろいろな表示や動作を行うことができます。
そのプロパティを設定するためのフォームがインスペクタのプロパティ設定です。プロパティ設定は、アプリケーションビルダの [編集] メニューの [プロパティ編集] を選択するか、インスペクタの [プロパティ] タブを選択します。すると、プロパティ一覧が表示され、編集可能となります。

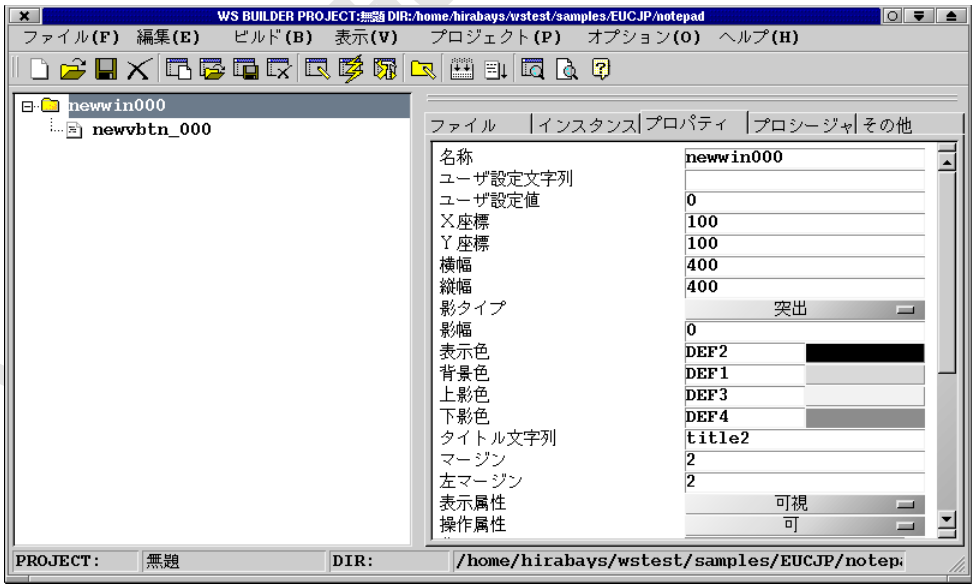


［プロパティ設定ウィンドウの表示］



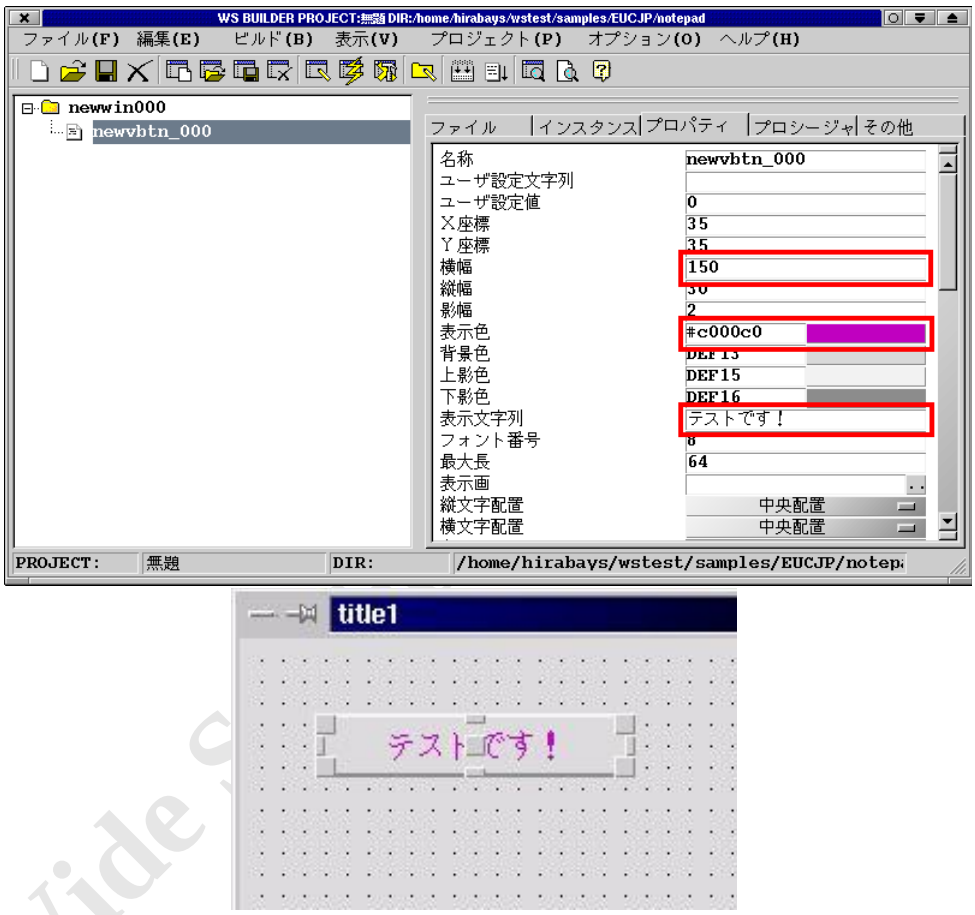
［オブジェクト上でのマウスの右ボタンメニューによる表示］

次の図のようなプロパティ設定ウィンドウが表示されます。



［プロパティ設定ウィンドウ (インスペクタ右)］

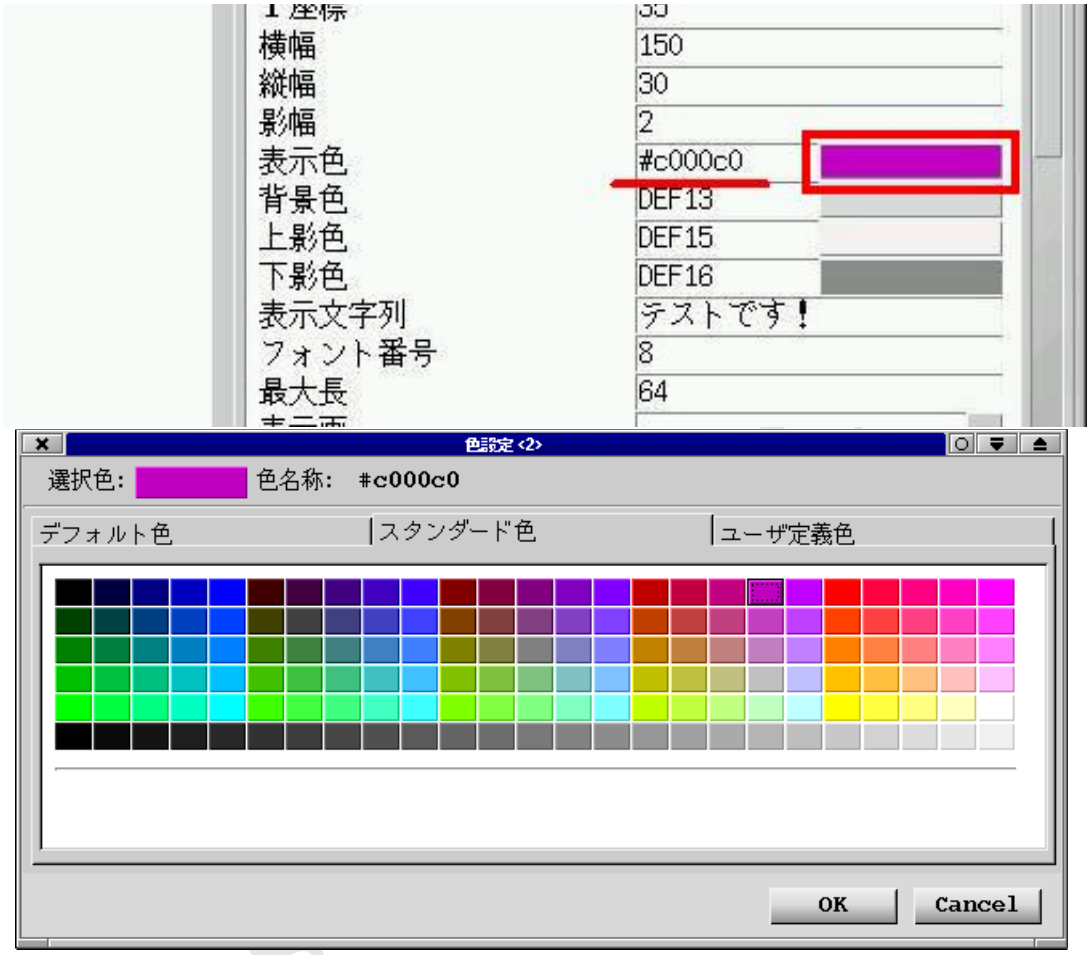
上図のプロパティ設定ウィンドウはプッシュボタンオブジェクト (WSCvbtn) のプロパティを表示しています。変更したいプロパティの値の入力を行い、インプットフィールドならば、リターンキーを押すと反映されます。
次の図は、下線部の値を変更したところです。図中のオブジェクトの表示状態が変化しているのがわかります。



［プロパティ設定ウィンドウによる値の変更］

7.2 色プロパティを設定するには

色を設定する場合は、次の図の様に、下線部に色名または 16 進数の RGB 値を入力するか、色ボタンを押して色選択ダイアログで色を選択してください。



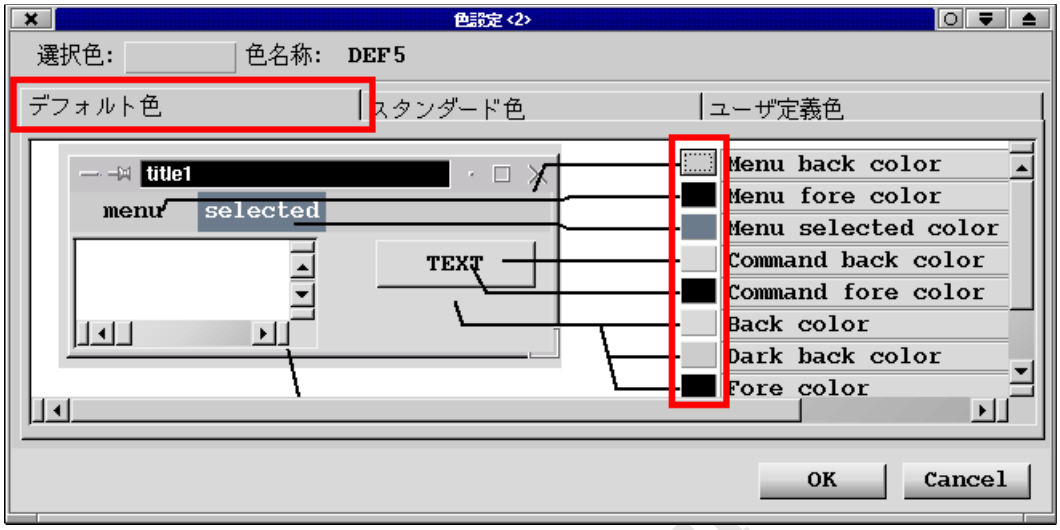
［色の変更/選択］

RGB 値で色名を指定する場合、 ＃を頭につけて、赤２桁、緑２桁、青２桁の順でそれぞれ 00〜ff（16 進数）の値をもちます。黒は #000000、白は #ffffff です。英数半角文字で指定しましょう。

有効な値は、次の様な範囲のものです。

#000000〜#ffffff

また、システムデフォルト色を指定することもできます。特に Windows sysmtem 向けにアプリケーションを開発する場合は、このデフォルト色を使うようにしましょう。



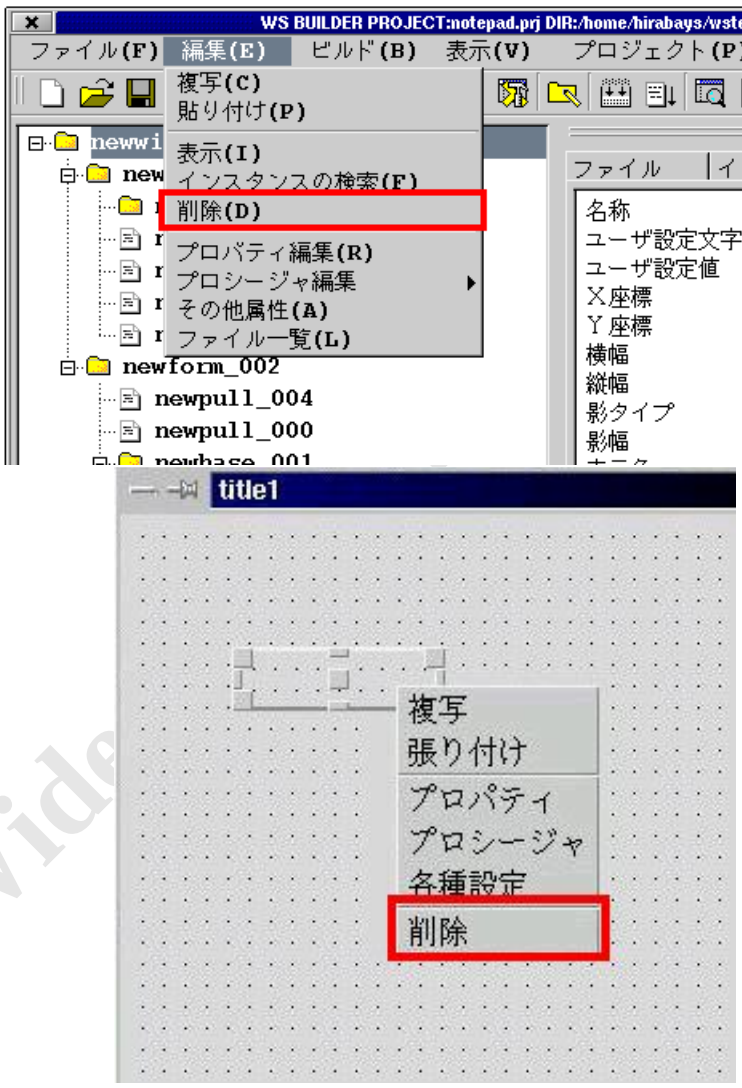
［色の変更/選択］

Wide Studio Manual

8　配置したオブジェクトを削除するには

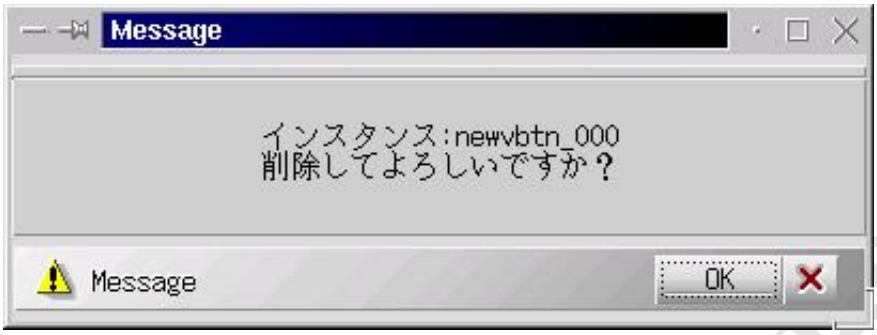
8.1　配置したオブジェクトを削除するには

オブジェクトを削除するには、まず削除したいオブジェクトをクリックして選択状態にします。そして［編集］メニューの削除を選択するか、オブジェクト上での右マウスボタンのクリックにより表示されるポップアップメニューの削除を選択します。



［オブジェクトの削除］

次の図のような確認のダイアログが表示されるので了解を選択します。



［削除の確認］

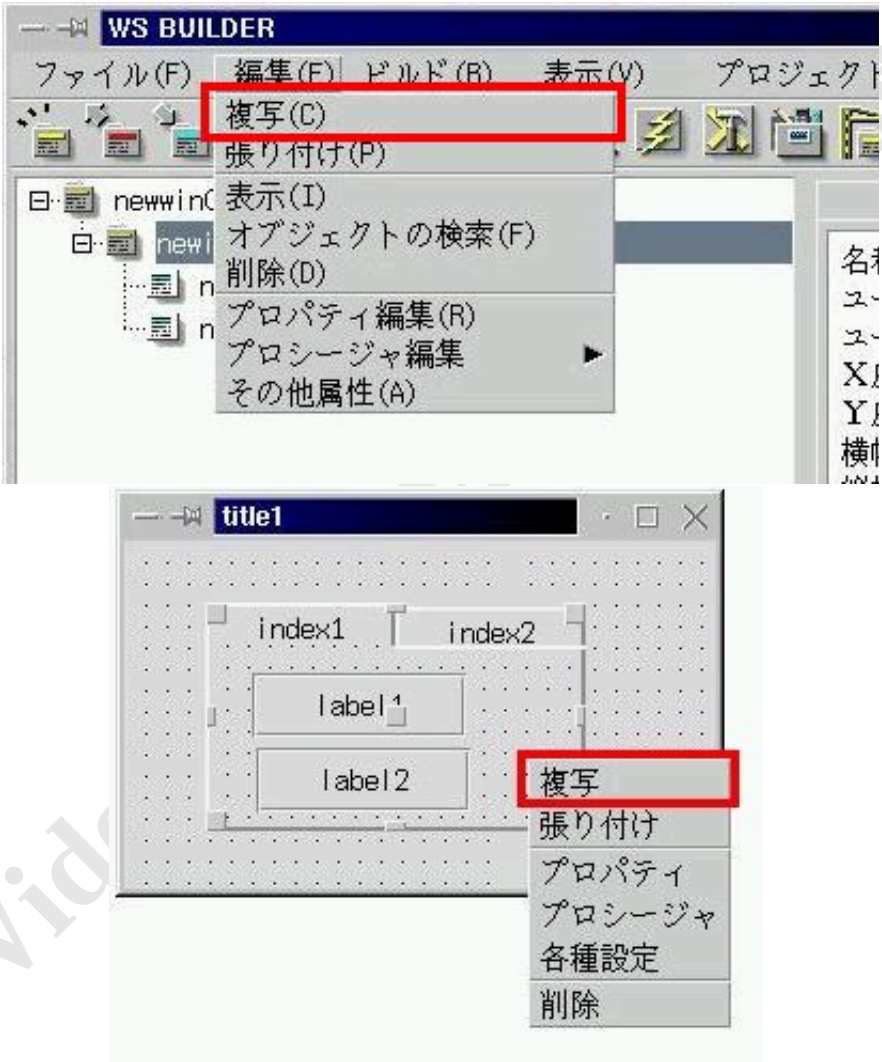
8.2 アプリケーションウィンドウを削除には

アプリケーションウィンドウ編のアプリケーションウィンドウの削除方法の他に、オブジェクト上のマウスの右ボタンメニューで、通常のオブジェクトと同じように削除が出来ます。
次の図のようにアプリケーションウィンドウを選択した状態で削除をするとアプリケーションウィンドウごと削除されます。

Wide Manual Page

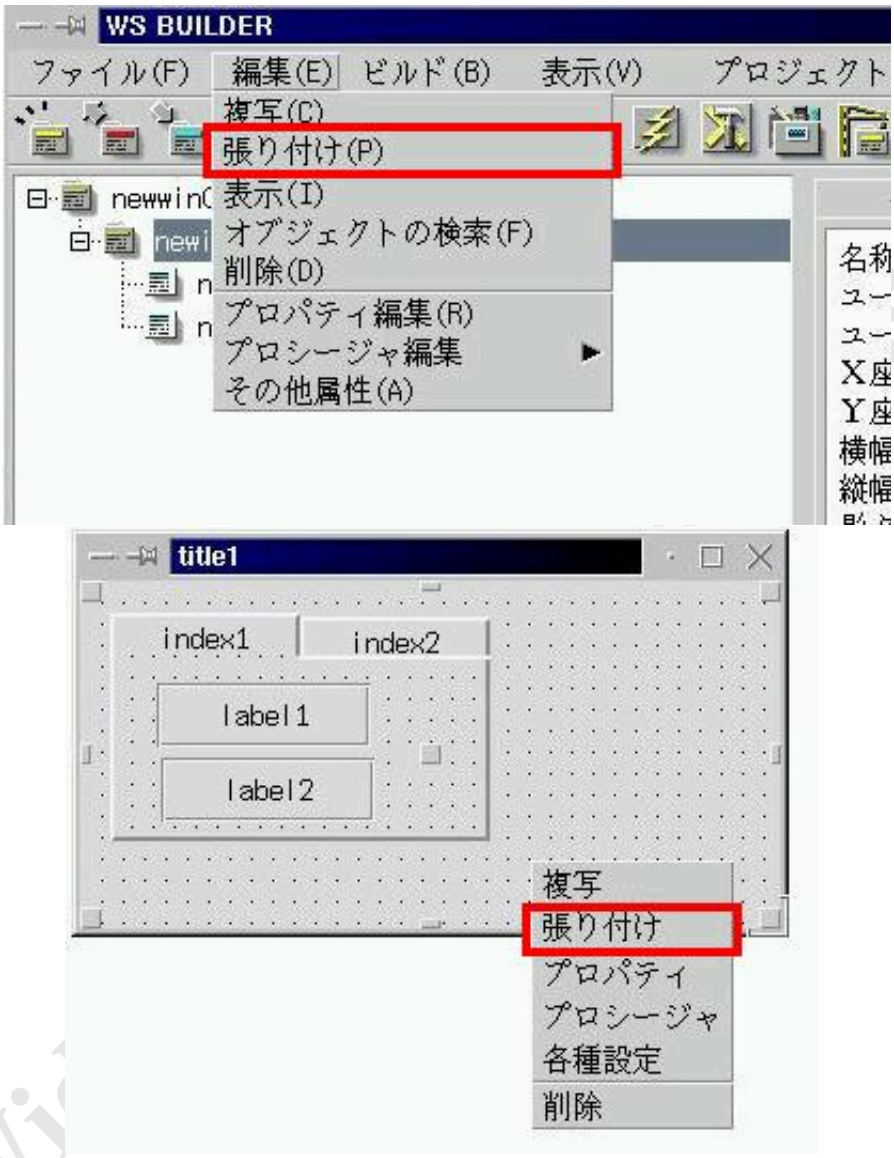
9　オブジェクトをコピーするには

オブジェクトをコピーするには、まずコピーしたいオブジェクトをクリックして選択状態にし、アプリケーションビルダの［編集］メニューの［複写］を選択するか、オブジェクト上での右マウスボタンのクリックにより表示されるポップアップメニューの［複写］を選択します。



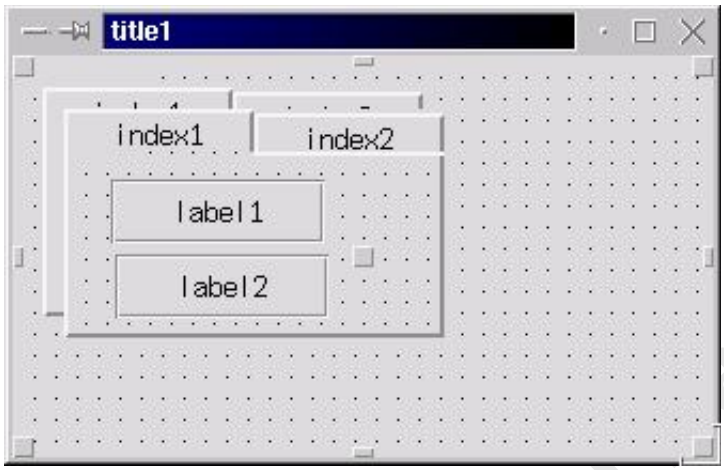
［コピーするオブジェクトの指定］

次にコピー先のオブジェクトを選択状態にして、［編集］メニューの［張り付け］を選択するか、オブジェクト上での右クリックにより表示されるポップアップメニューの［張り付け］を選択します。



［コピー先のオブジェクトの指定］

次の図のようにオブジェクトがコピーされます。

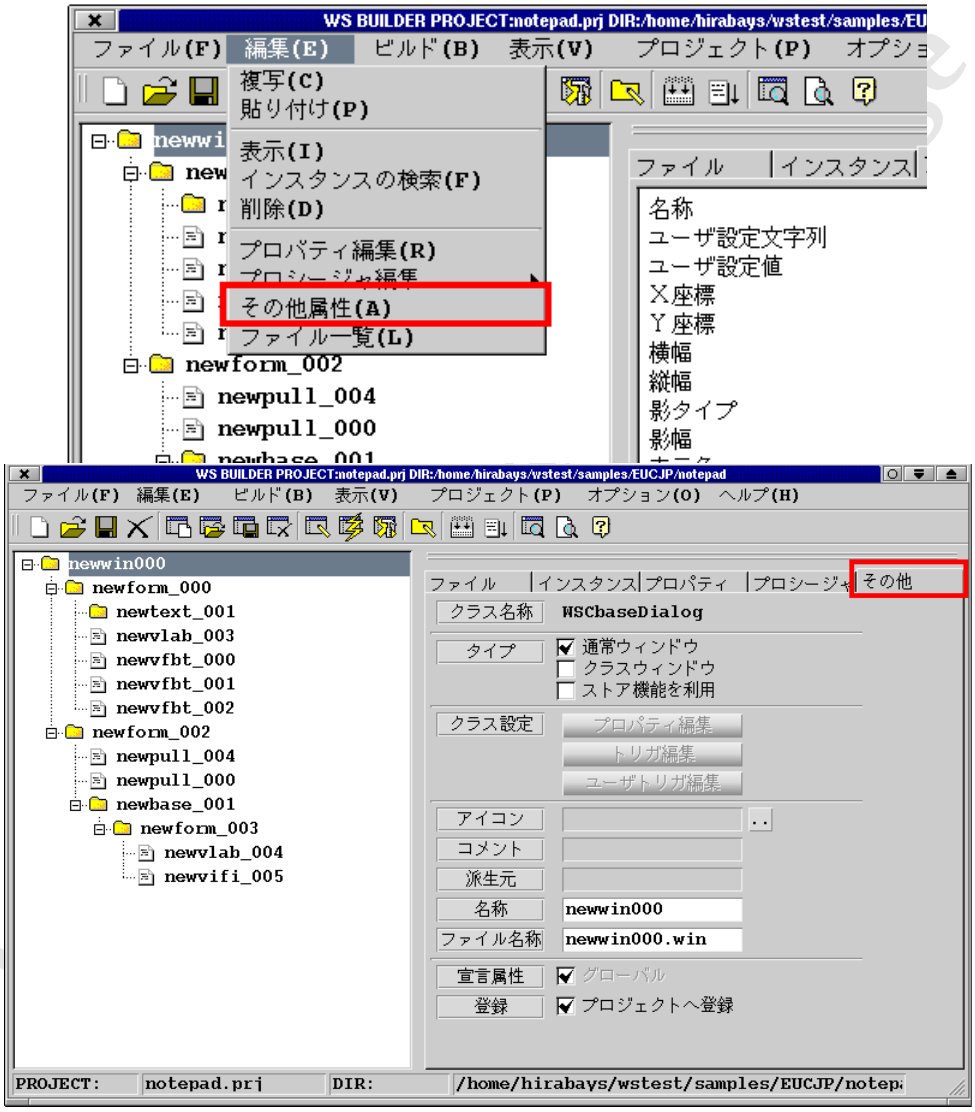


［コピーされたオブジェクト］

Wide Studio Manual Image

10　外部変数として参照可能なオブジェクトとするには

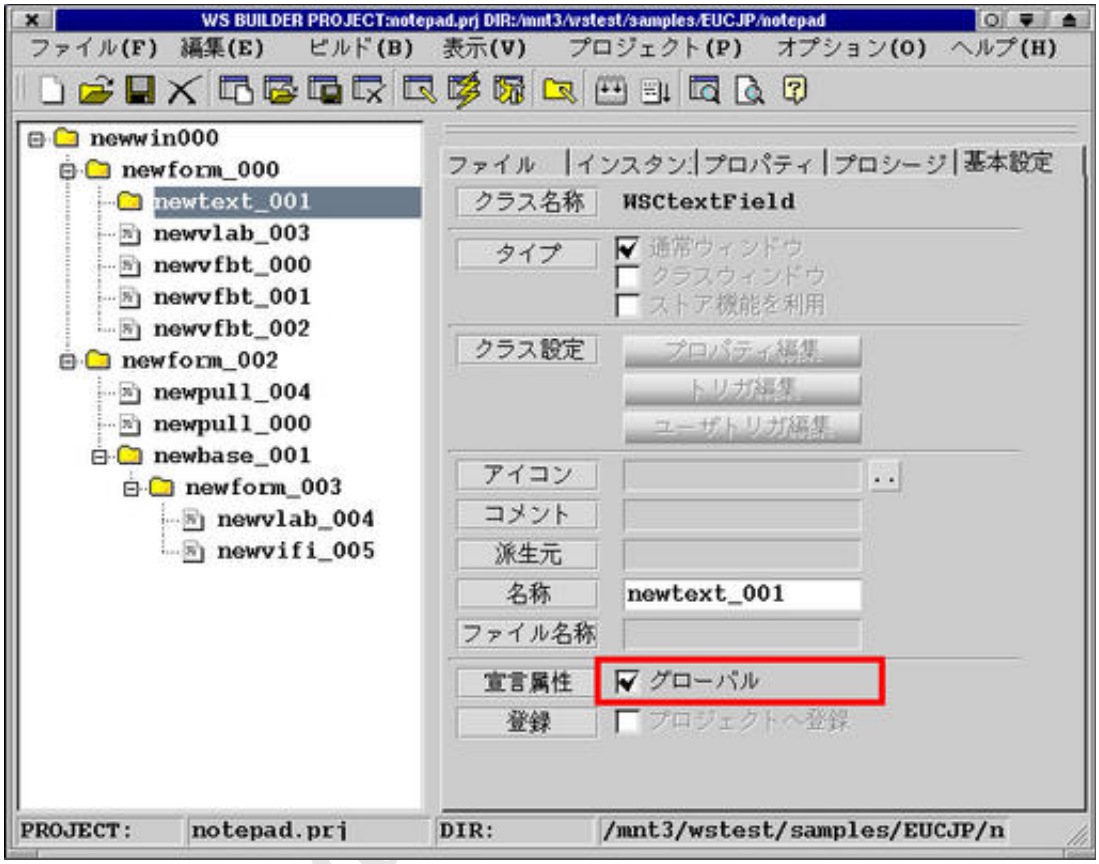
オブジェクトは、イベントプロシージャなどの C++ プログラムの中から、外部変数として参照することができます。外部変数として参照可能なオブジェクトにするには、まずオブジェクトを選択してインスペクタの基本設定を表示します。



［外部変数定義の指定］

図中の［変数宣言］の項目をチェックして、グローバル属性とします。
なお、アプリケーションウィンドウ、または配列オブジェクトは、自動的に外部変数参照の対象にな

ります。



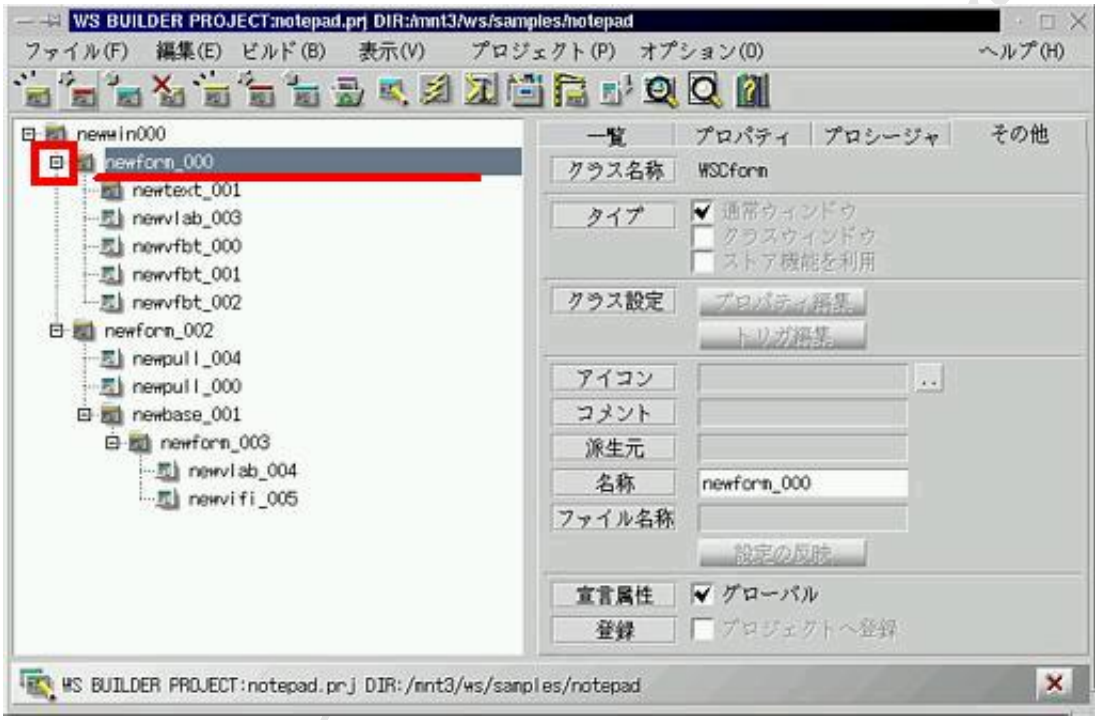
［外部変数定義の指定］

Wide St

11　オブジェクトの一覧を表示するには

11.1　アプリケーションウィンドウ内のオブジェクト構成を表示するには

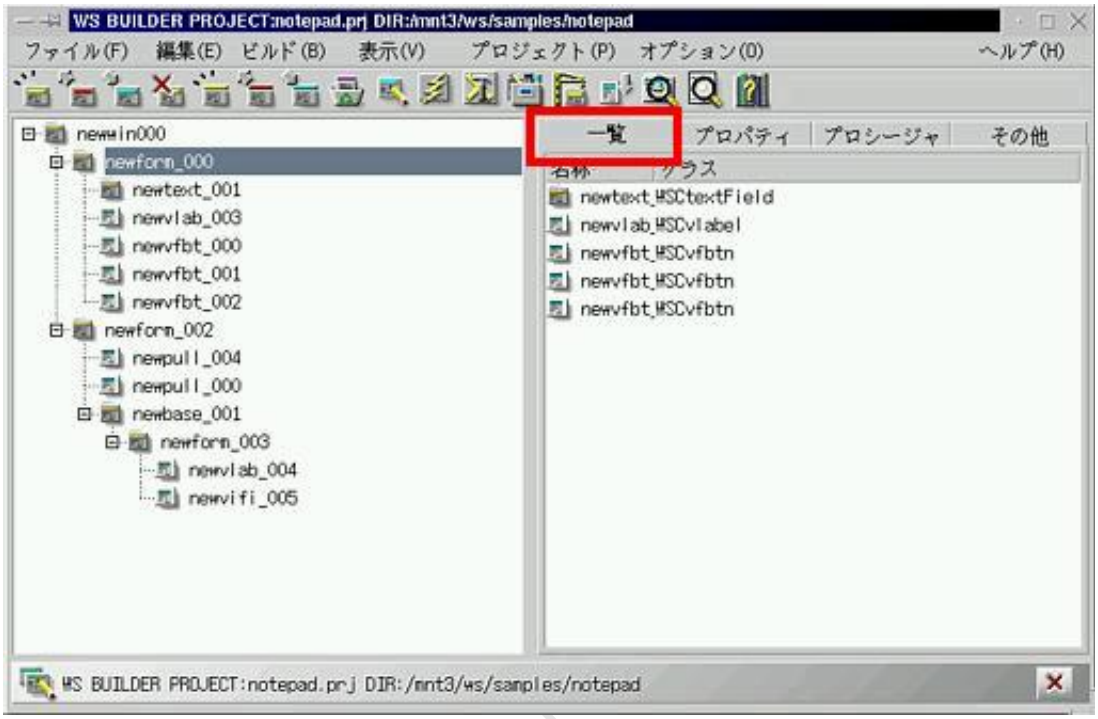
アプリケーションウィンドウ内のオブジェクトの構成の一覧を表示するには、インスペクタで、図に示すボタンをクリックするか、名称をダブルクリックします。そのインスタンス配下の子インスタンスが展開されます。もう一度行うと、今度は閉じます。



［オブジェクト構成の表示］

11.2　指定したマネージャオブジェクト内の一覧を表示するには

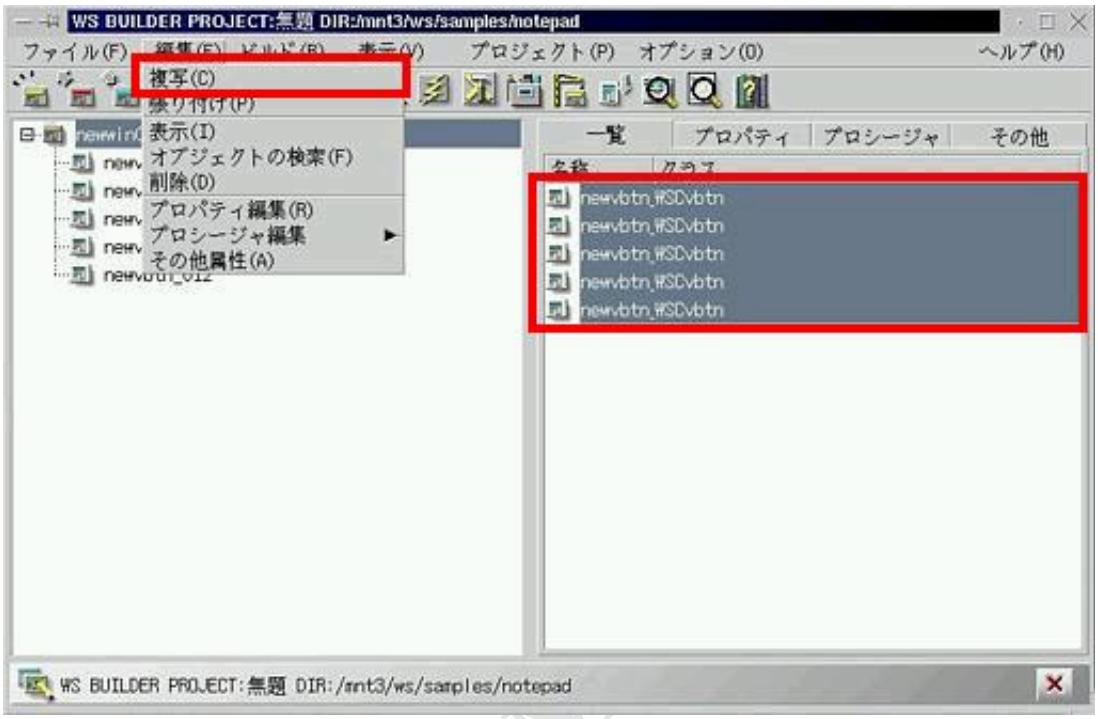
インスペクタでマネージャオブジェクトを選択して、［一覧］タグを選択すると、そのオブジェクト配下のオブジェクトの一覧が表示されます。一覧表示で、オブジェクトを複数選択して、コピーの対象とすることもできます。



［一覧の表示］

一覧表示で、複数選択してコピー等するには、次の図の様に複数選択し、アプリケーションビルダの［編集］メニューの［複写］を選択するか、一覧上での右マウスボタンのクリックにより 表示されるポップアップメニューの［複写］を選択します。

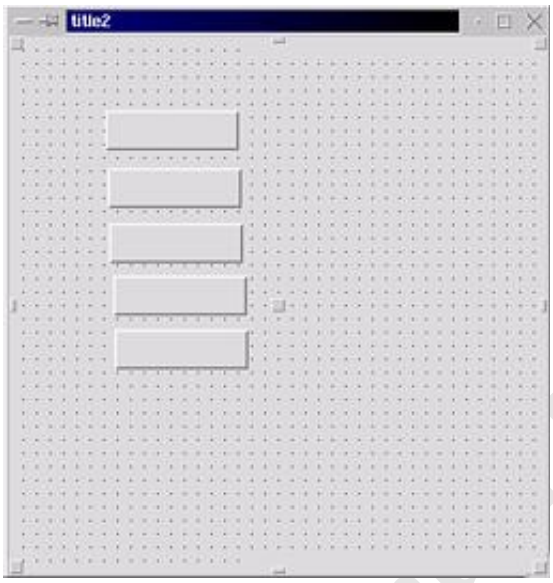
Wide Studio



［一覧表示での複数選択］

そこで、［張り付け］を行うと次の図の様に、一度にコピーされます。

Wide Studio

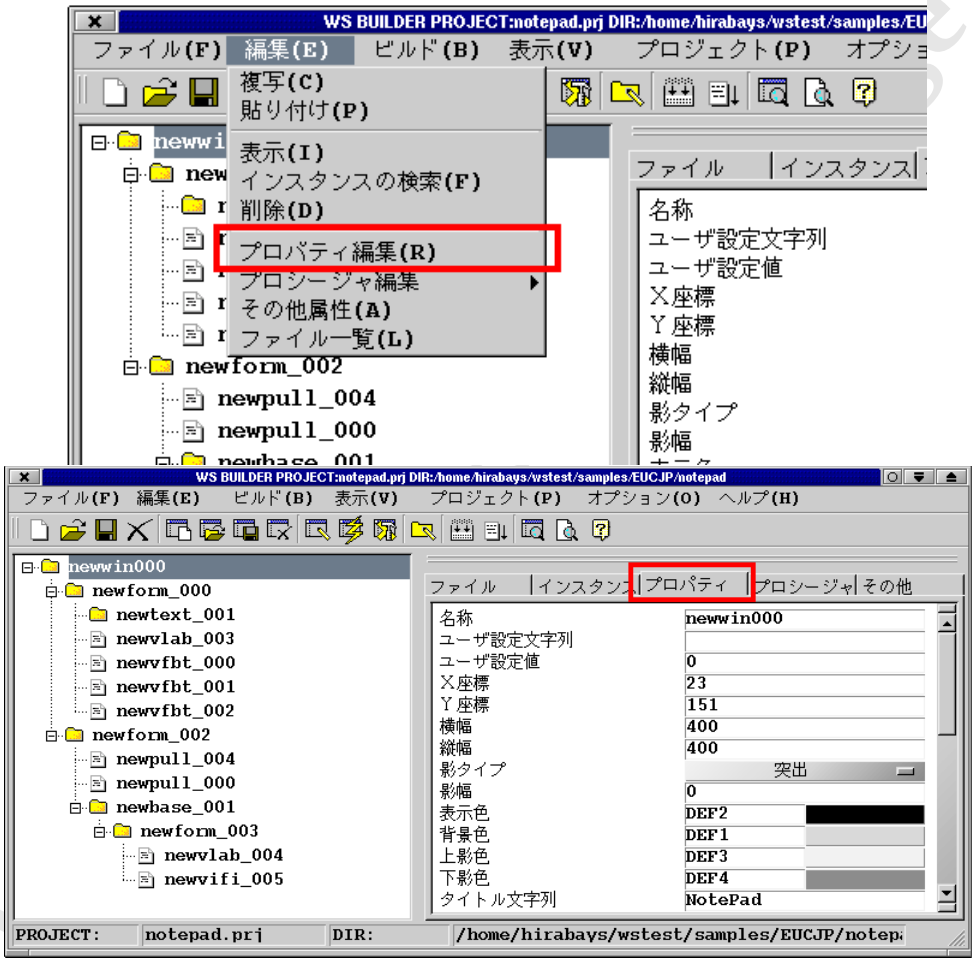


［複数選択による張り付け］

Wide Studio Manual Page

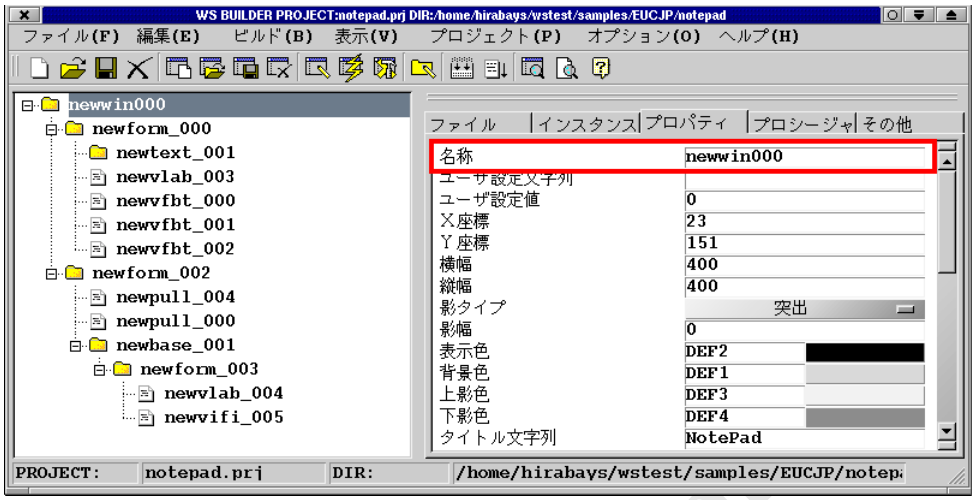
12　オブジェクトの名称を変更するには

新しくオブジェクトを配置した場合には生成時にアプリケーションビルダが自動的に名称を割り振ります。その名称を変更したい場合、プロパティ設定で行うことができます。まず名称を変更したいオブジェクトを選択します。次にプロパティ設定ウィンドウを表示させます。図に示す名称プロパティに新しい名称を入力します。



［プロパティ設定の表示］

また、アプリケーションウィンドウのベースとなるウィンドウの名称を変更するとアプリケーションウィンドウ名称を変更することができます。［アプリケーションウィンドウ名を変更するには］の節を御参照下さい。



［オブジェクトの名称の指定］

Wide Studio Manual

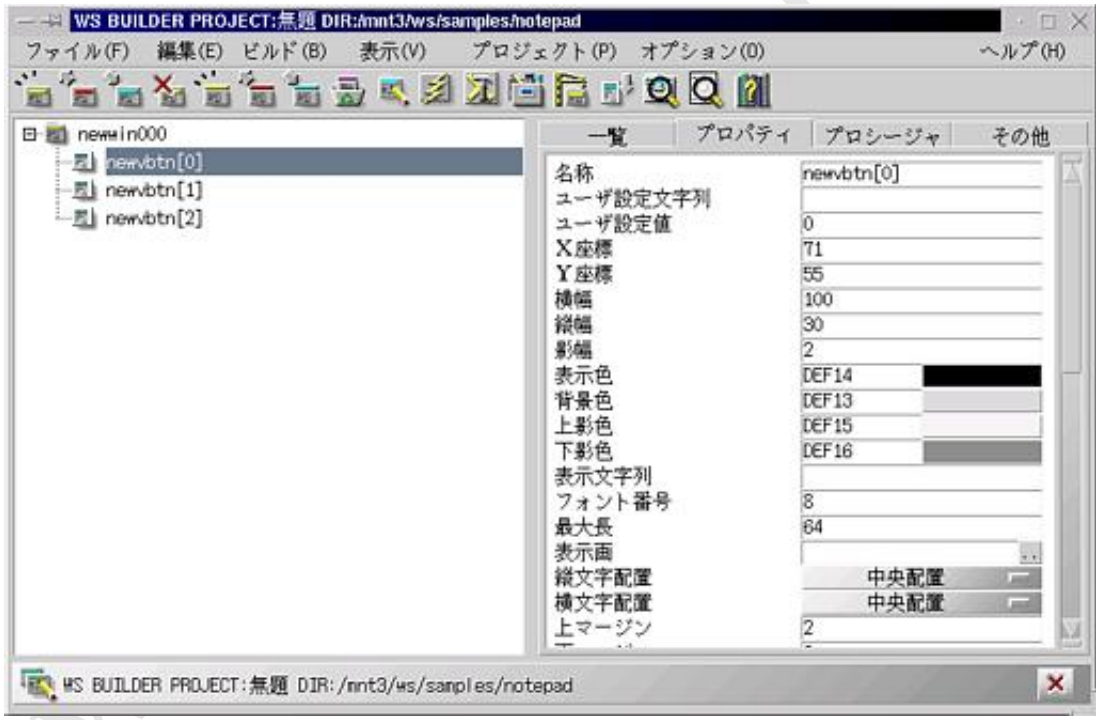
13　オブジェクトを配列として定義するには

オブジェクトに配列名称を指定することで配列定義します。まず、配列定義したいオブジェクトを選択します。次にプロパティ設定ウィンドウを表示させます。図に示す項目に配列名称を入力します。

配列名称は次構成を取ります。

arrayname[NO]

arrayname は英数半角文字、NO は、半角の０から始まる整数を指定します。注意事項としては、配列は同じアプリケーションウィンドウ内であること、配列のメンバーは同じクラスであることです。もしこれらの制限を越えると警告ダイアログが表示されます。



［オブジェクトの配列名称の指定］

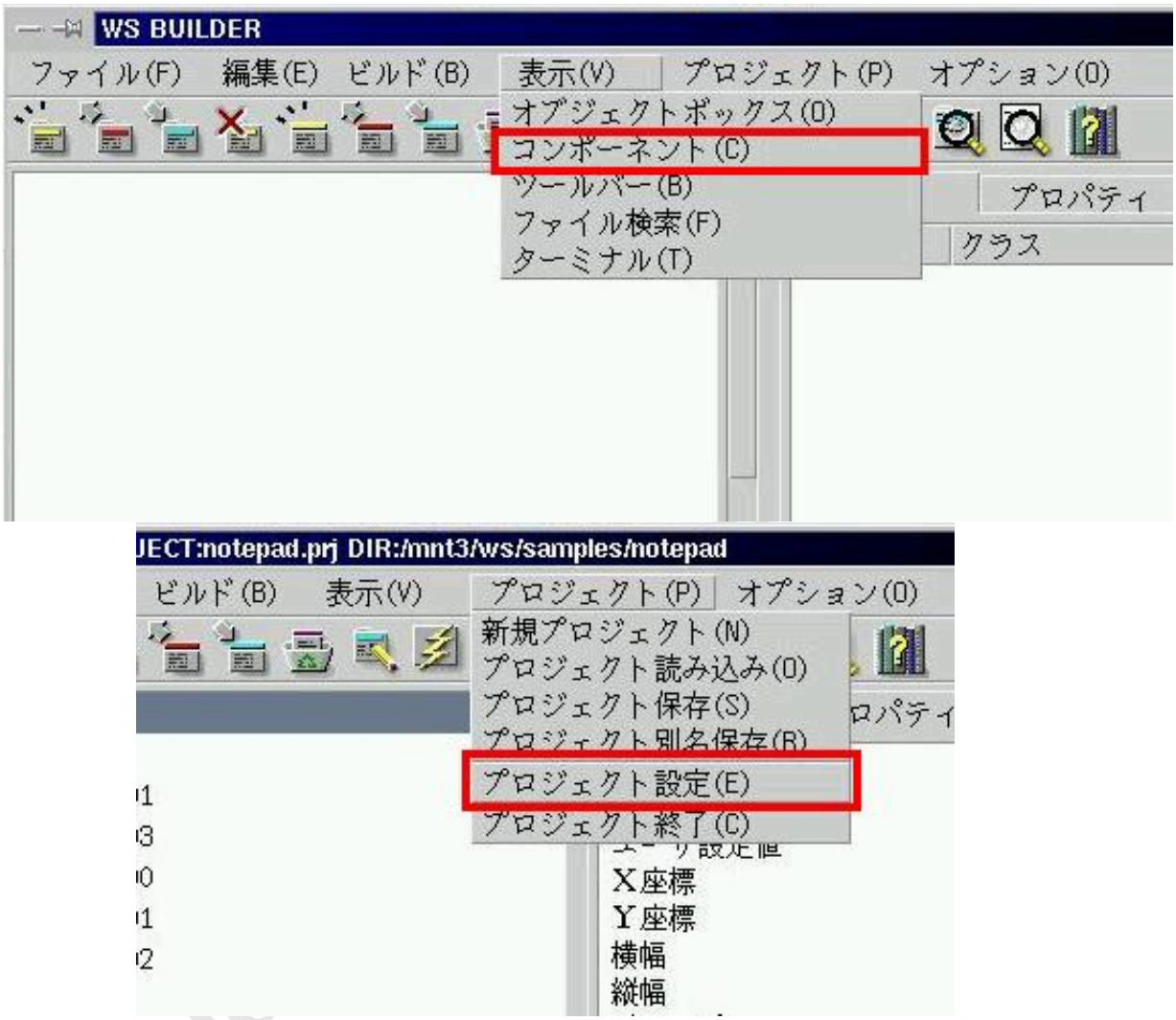
なおアプリケーションウィンドウは、配列定義できないのでご注意ください。また、外部変数で配列オブジェクトにアクセスする場合は、プログラミング編の［オブジェクトにアクセスするには］の節を参照下さい。

14　追加ライブラリによる新たなオブジェクトを利用するには

アプリケーションビルダでは、新たなオブジェクトを追加ライブラリによって利用することができます。追加ライブラリの指定は、プロジェクト設定ウィンドウで行います。［表示］メニューの［コンポーネント］を選択するか、［プロジェクト］メニューの［プロジェクト設定］を選択してください。後者の場合は、［追加］タブを選択すると、表示されます。

また j 追加ライブラリを利用する場合は、リンク用のライブラリ設定も行ってください。詳しくは、プロジェクトのコンパイルのライブラリの指定の節を参照下さい。

Wide Studio Manual Page



［環境設定ウィンドウの表示］

ライブラリを追加するには、次に示す追加ボタンをクリックします。ファイル選択ウィンドウが表示されますので、追加ライブラリのファイル名を選択し設定します。 UNIX のシステムの場合は、シェアードライブラリ (libXXXX.so) Windows の場合は、DLL (libXXX.dll) を指定します。

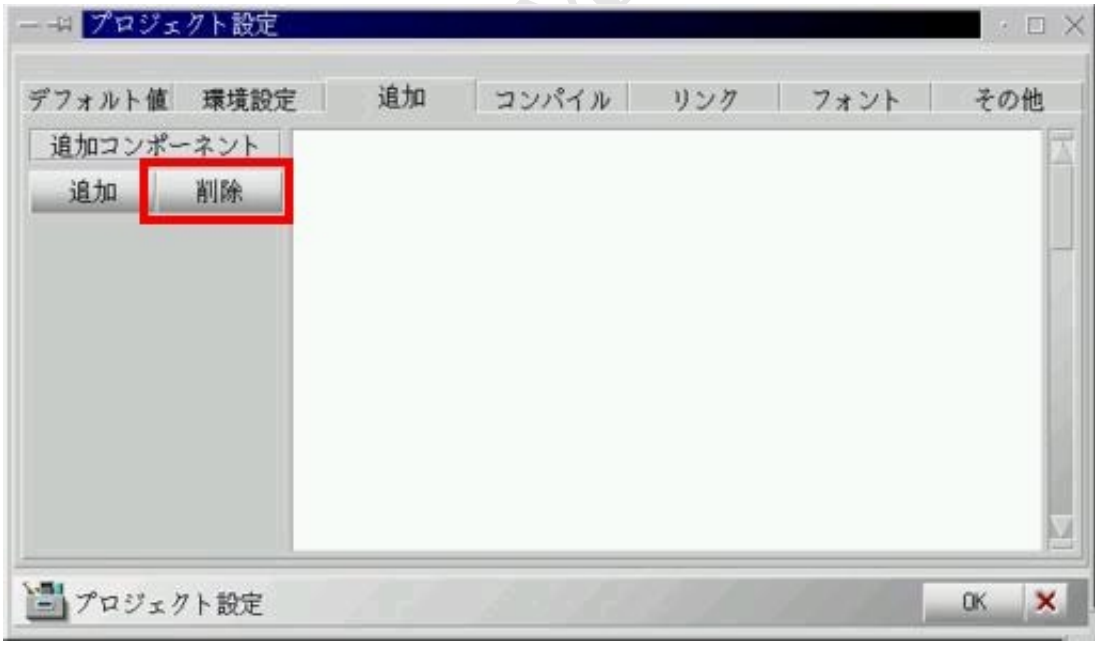
［ライブラリの追加］

［ライブラリが追加されたところ］

［ライブラリが追加され、オブジェクトボックスにオブジェクトが増えたところ］

また追加したライブラリを削除または変更したい場合、次の図のようにそのライブラリをクリックして選択状態 (反転表示状態) にします。

変更したい場合も一度、削除してから、新たに追加しなおしてください。削除ボタンをクリックして削除します。



［追加したライブラリの削除］

15　配置したオブジェクトのクラス名を確認するには

配置したオブジェクトのクラス名を確認するには、［編集］メニューの基本設定を選択します。クラス名称の欄にクラス名が表示されます。

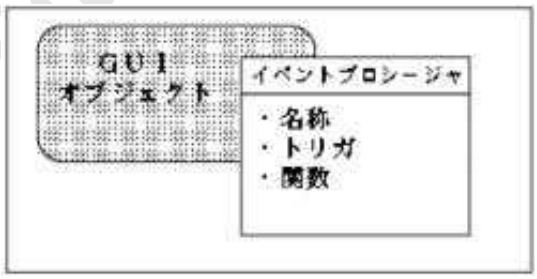
Wide Studio Manual Page

E　イベントプロシージャ編

1　イベントプロシージャとは

イベントプロシージャは、C++ 言語で記述された、オブジェクトの動作を補佐するためのプログラムです。イベントプロシージャを使用することで、オブジェクトだけでは実現が難しい高度な動きをする画面の実現、オブジェクトへの機能の追加や、データの投入など、更に複雑な動作を実現することができます。
イベントプロシージャは、C++ 言語の知識があれば自由に記述定義することができます。プログラムの記述方法については、プログラミング編を参照下さい。
イベントプロシージャは次の要素を持ちます。

- プロシージャ名称
  オブジェクトにはイベントプロシージャが多数貼られることがあります。従ってイベントプロシージャは自らを識別するための名称を持っています。
- プログラム関数
  イベントプロシージャは、C++ 言語で記述された関数を持ちます。イベントプロシージャは起動されると、この関数を実行します。したがって、イベントプロシージャ本体 = プログラム関数と捉えて良いでしょう。
- トリガ
  イベントプロシージャは、起動されるトリガを持ちます。オブジェクトに対しそのトリガが発生するとプログラム関数が実行されます。



［イベントプロシージャ］

1.1　イベントプロシージャ関数

イベントプロシージャの関数は、引数に、張り付けたオブジェクトが渡されて来ます。
次に示すものは、アプリケーションビルダが生成した、イベントプロシージャ関数のサンプルです。

Wide Studio Manual Page

```
\#include <WScom.h>
\#include <WSCfunctionList.h>
\#include <WSCbase.h>

//-------------------------------------------------------------
//Function for the event procedure
//-------------------------------------------------------------
void sample(WSCbase* object){
  object->setProperty(WSNlabelString,"Hello."); //A

}
static WSCfunctionRegister  op("sample",(void*)sample);
```

A は、オブジェクトの表示文字列プロパティ、WSNlabelSgtring を、Hello. に設定しています。

Wide Studio Manual Page

2 トリガとは

トリガとはオブジェクトに対して発生するイベントのようなものです。イベントプロシージャには、トリガを設定することができ、そのトリガの発生によって起動されます。例えば、オブジェクト上でマウスがクリックされた場合、WSEV_MOUSE_PRESS が発生します。従って WSEV_MOUSE_PRESS を設定してあるイベントプロシージャが起動されます。
イベントプロシージャで利用できるトリガには、大きく分けて次のように４種類あります。

- オブジェクトの状態の変化で発生するもの
  表示状態などオブジェクトの状態が変化することで発生します。
- マウスの変化で発生するもの
  操作されるなどマウスが動作することで発生します。
- キーボードの変化で発生するもの
  キーが押下されるなどキーボードに関することで発生します。
- その他

状態の変化に関するもの	説明
WSEV_INITIALIZE	オブジェクトが初期かされたとき
WSEV_DELETE	オブジェクトが削除されるとき
WSEV_ACTIVATE	ボタンなど押して離したとき
WSEV_VALUE_CH	トグルボタンなど値が変化したとき
WSEV_VISIBLE_CH	オブジェクトの表示状態が変化したとき
WSEV_PARENT_VISIBLE_CH	所属している親オブジェクトの表示状態が変化したとき
WSEV_EXPOSE	オブジェクトが露出して描画されるとき
WSEV_RESIZE	ウィンドウなどのサイズが外部から変更されたとき
WSEV_SENSITIVE_CH	オブジェクトの選択属性が変化した場合
WSEV_PARENT_SENSITIVE_CH	所属している親オブジェクトの選択属性が変化した場合

マウスに関するもの	説明
WSEV_MOUSE_IN	マウスがオブジェクト内に入ったとき
WSEV_MOUSE_OUT	マウスがオブジェクト外に出たとき
WSEV_MOUSE_PRESS	マウスボタンがおされたとき
WSEV_MOUSE_RELEASE	マウスボタンがはなされたとき
WSEV_MOUSE_MOVE	マウスが動いたとき

キーボードに関するもの	説明
WSEV_FOUCS_CH	フォーカスが変化した場合
WSEV_KEY_PRESS	キーボードが押されたとき
WSEV_KEY_RELEASE	キーボードが離されたとき
WSEV_KEY_HOOK	キーボードが押される直前

Wide Studio Manual Page

その他	説明
WSEV_NONE	トリガなし

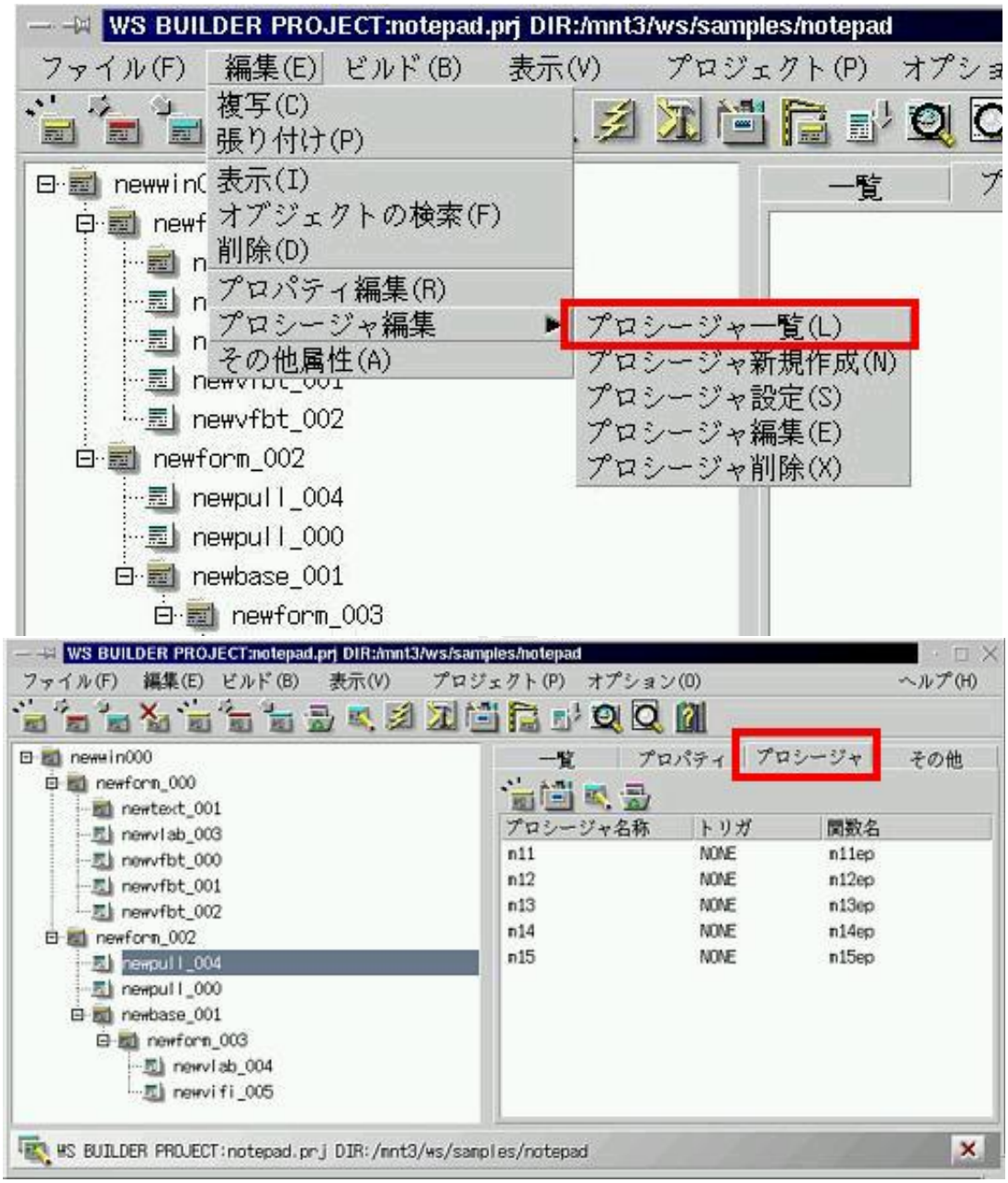
Wide Studio Manual Page

3　イベントプロシージャを作成/設定/削除するには

3.1　イベントプロシージャを一覧表示するには

イベントプロシージャを作成/設定するのに使用するのが、インスペクタのプロシージャ設定です。イベントプロシージャの対象となるオブジェクトを選択してから、アプリケーションビルダの［編集］メニューの［プロシージャ編集/プロシージャ一覧］を選択するか、インスペクタの［プロシージャ］タブを選択します。すると、その選択されているオブジェクトに対して設定されているイベントプロシージャの一覧が表示されます。

Wide Studio Manual Page

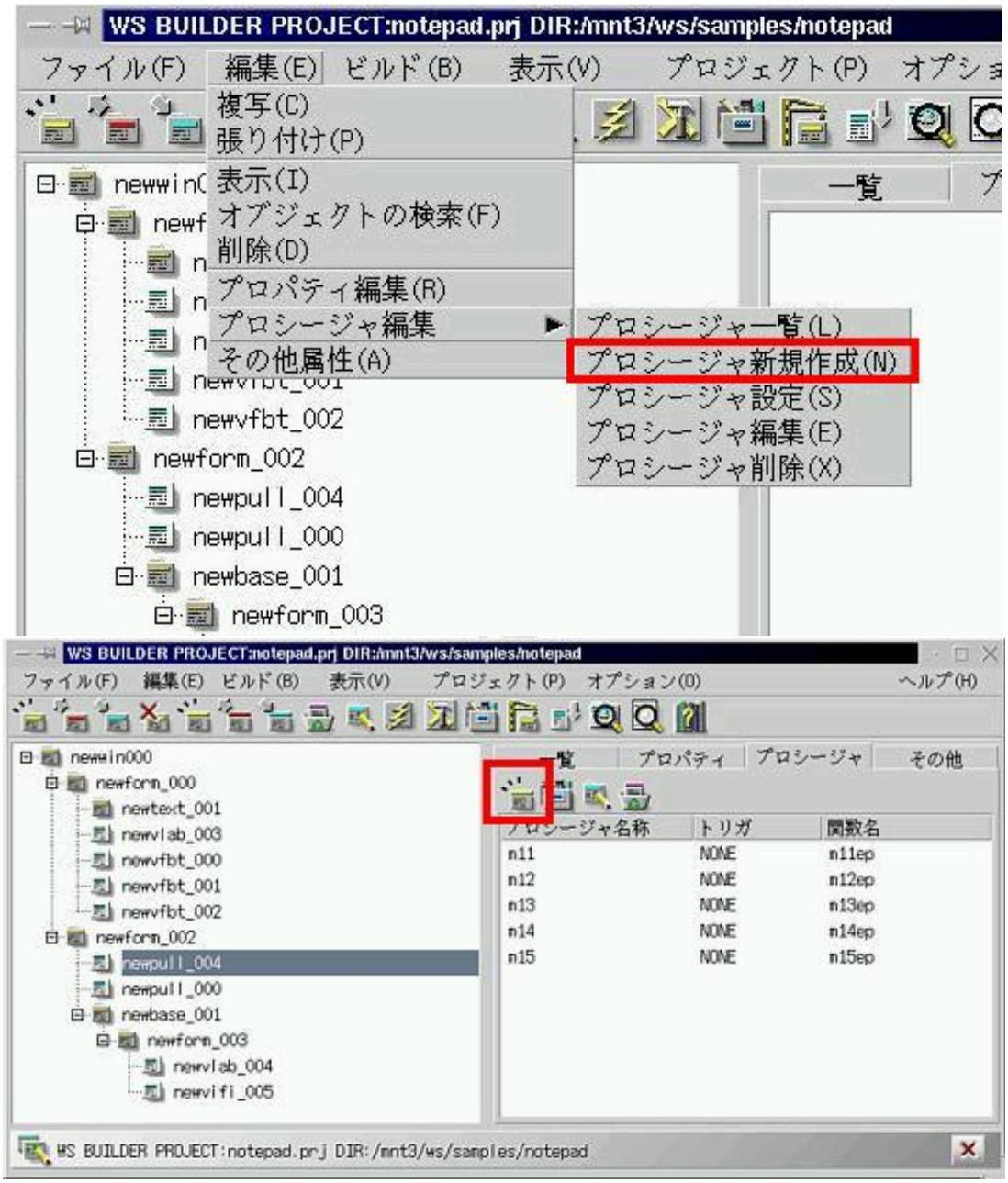


［プロシージャ一覧の表示］

3.2 イベントプロシージャを新規に作成するには

イベントプロシージャを新規に作成する場合、イベントプロシージャの対象となるオブジェクトを選択してから、アプリケーションビルダの［編集］メニューの［プロシージャ編集/プロシージャ新規作成］を選択するか、次の図のアイコンをマウスでクリックして下さい。

Wide Studio Manual Page



［新規イベントプロシージャの作成］

表示されたプロシージャ編集ウィンドウに対して、プロシージャ名称、起動トリガ、起動関数名を指定してください。また、ひな型の関数を生成したい場合は、［ひな型作成］ボタンをクリックして

ください。

- プロシージャ名称:
  イベントプロシージャの識別名称を指定します。漢字仮名交じり文字列も指定できます。実際に行われる処理の内容に応じて自由に付けることができます。

- 起動トリガ:
  イベントプロシージャの起動トリガを指定します。

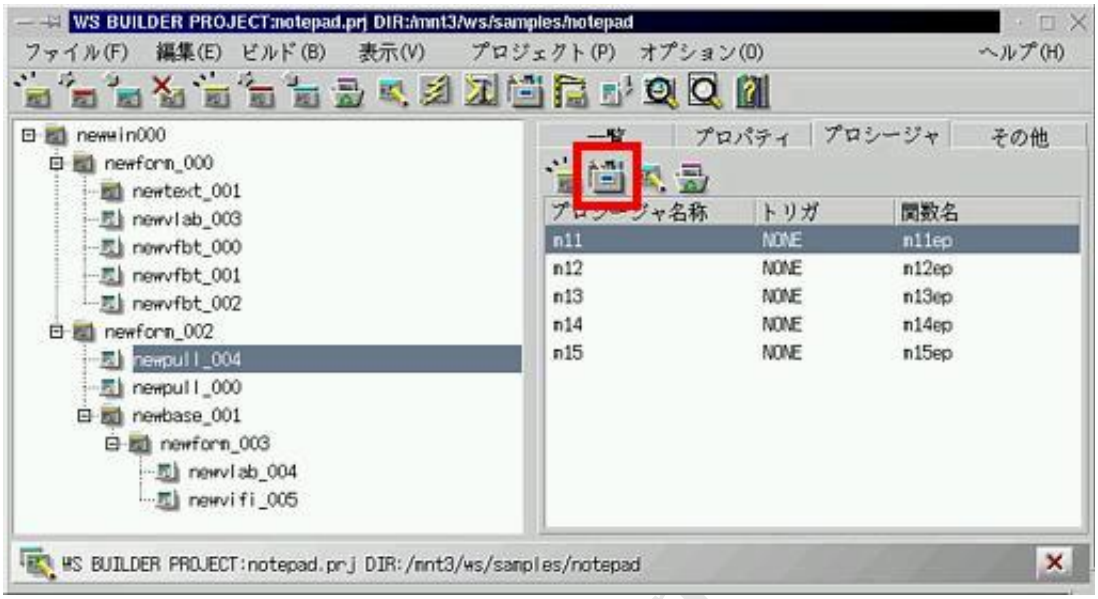
- 起動関数名:
  C++ 関数名を指定します。この関数名は、実際の C++ のシンボルとしてコンパイルされますので、C++ の文法に沿った名称を指定しなければなりません。

［プロシージャ設定ウィンドウ］

3.3 イベントプロシージャの設定を行うには

下線部のように編集したいイベントプロシージャをクリックして選択状態にし、
アプリケーションビルダの［編集］メニューの［プロシージャ編集/プロシージャ設定］を選択するか、図に示すアイコンをマウスクリックします。イベントプロシージャの新規作成と同じ画面がでてきますので、変更箇所を入力し、［設定］ボタンをクリックします。

Wide Studio Manual Page

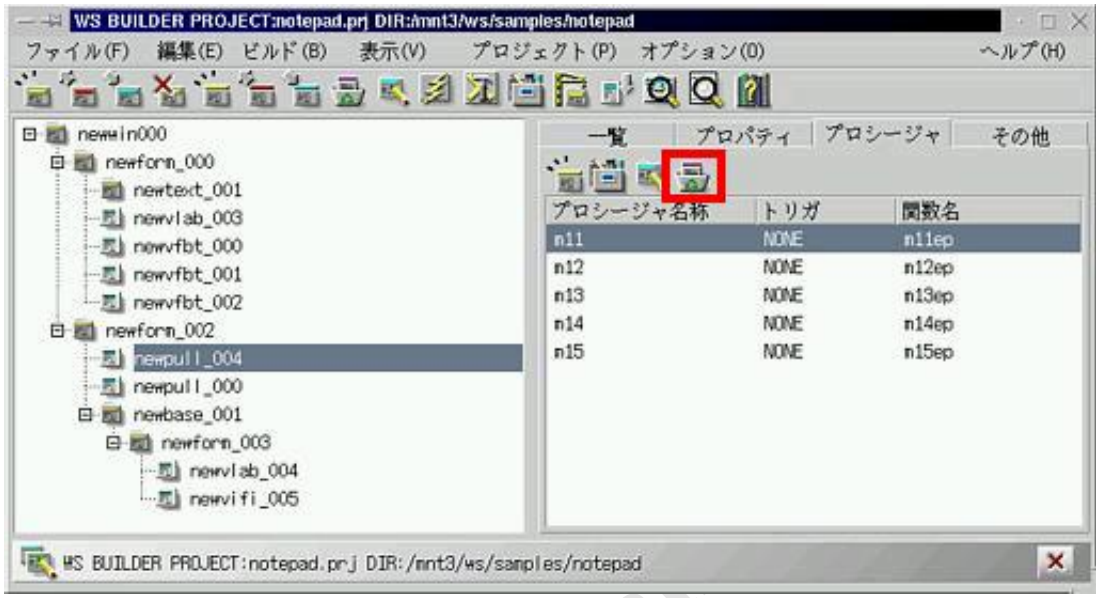


［イベントプロシージャの設定］

3.4 イベントプロシージャを削除するには

下線部のように削除したいイベントプロシージャをクリックして選択状態にし、アイコンをクリックします。

Wide Studio Manual



［イベントプロシージャを削除するには］

Wide Studio Manual

4 関数を作成/編集するには

4.1 関数のひな型ファイルを作成するには

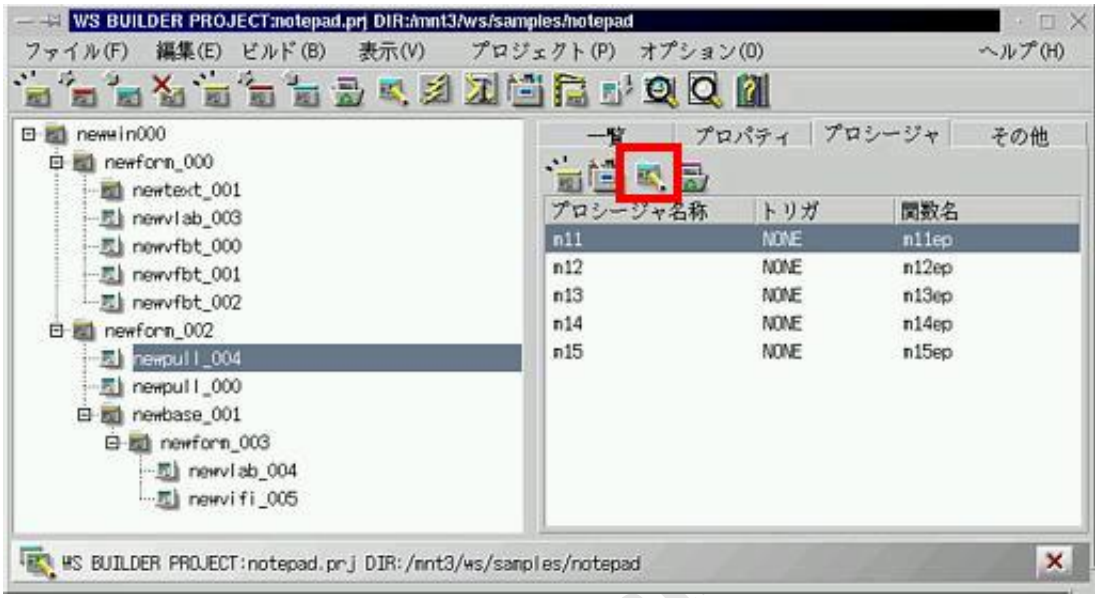
イベントプロシージャに設定する関数のひな型を作成するには、起動関数名を指定してからひな型作成ボタンをクリックします。

［関数のひな型作成］

4.2 関数を編集するには

イベントプロシージャに設定された関数を編集するには、図のイベントプロシージャ表示の部分をダブルクリック、または、図に示すアイコンをクリックします。関数を編集するテキストエディタが起動され、編集状態になります。

Wide Studio Manual Page

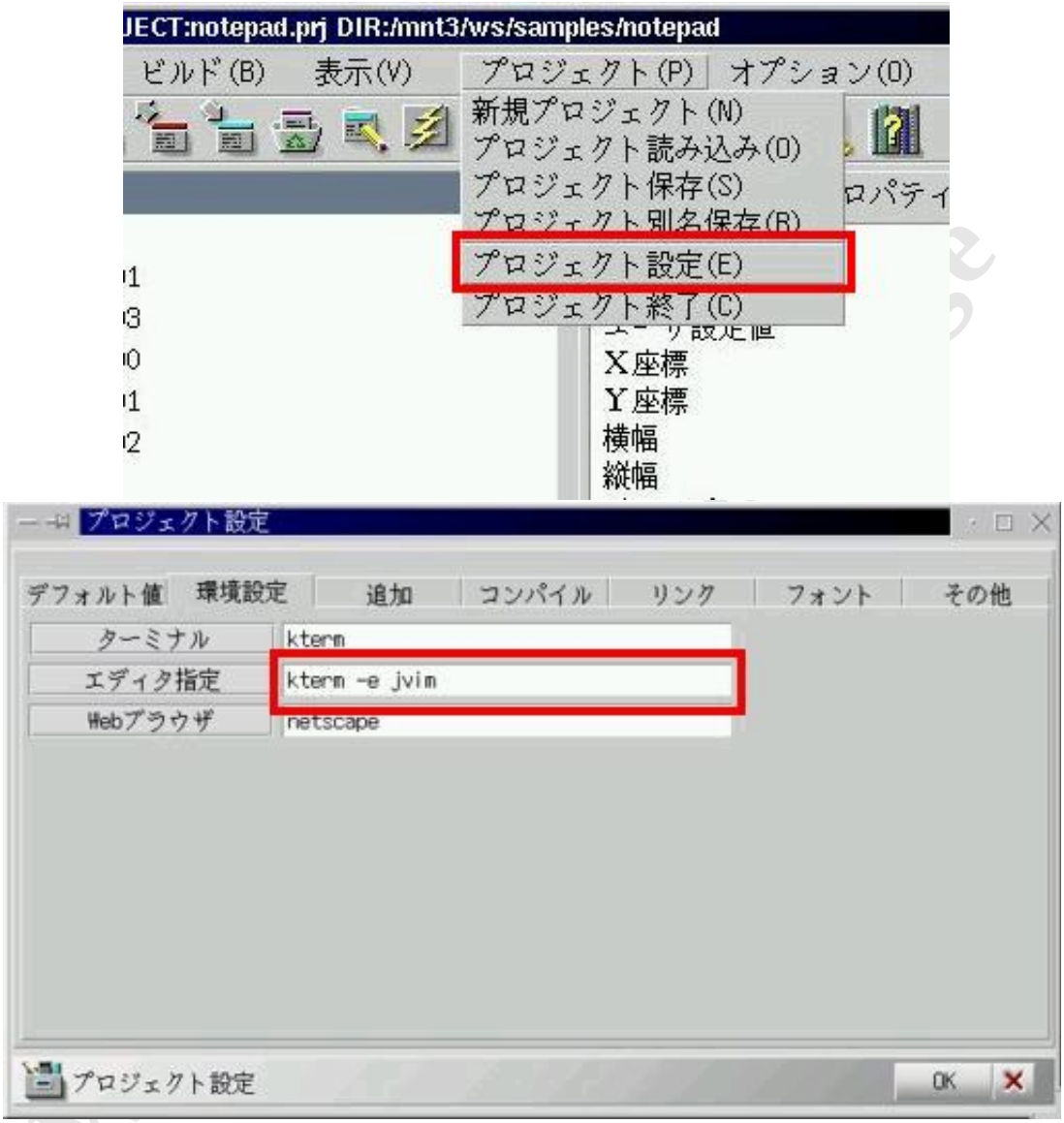


［関数の編集］

［エディタが立ち上がったところ］

4.3 関数を編集するエディタを指定するには

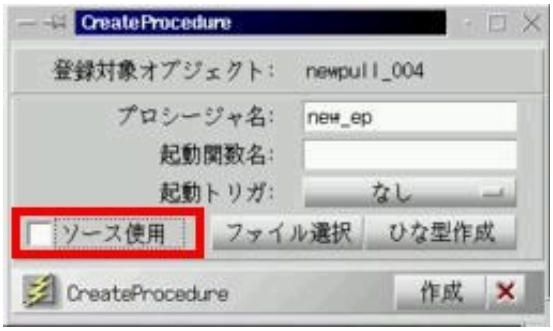
環境設定ウィンドウで関数を編集するエディタを指定することができます。アプリケーションビルダの［プロジェクト］メニューの［プロジェクト設定］を選択し、［環境設定］タブを選択します。図に示されるところにエディタのコマンド名やその起動のために必要な引数を入力してください。



［環境設定ウィンドウの表示／エディタの指定］

4.4 ライブラリ中に存在する関数を利用するには

イベントプロシージャに設定する関数には、コンパイル済みのライブラリ中の関数を指定することができます。図の示すソース使用の項目をチェックを外します。この場合、関数のソースプログラムのコンパイルや、そのオブジェクトファイルのリンクを行いません。



［ライブラリ中の関数の指定］

次にコンパイル時にリンクするライブラリの指定で、そのイベントプロシージャで使用されている関数の入ったライブラリを指定します。詳しくは、プロジェクトのコンパイルの節を参照下さい。

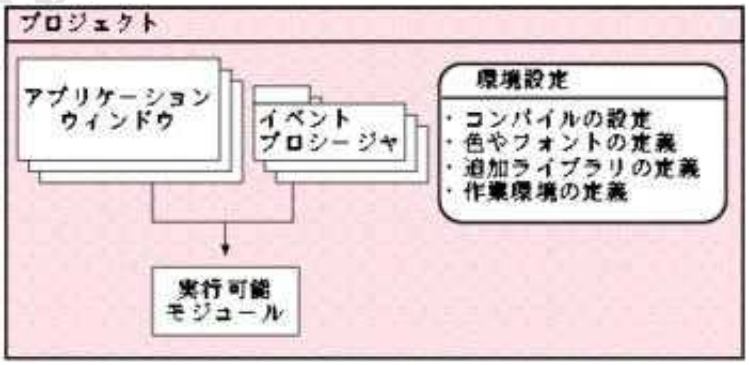
Wide Studio Manual Page

F　プロジェクト編

1　プロジェクトとは

プロジェクトは、ユーザがアプリケーションビルダで構築するアプリケーションの管理の単位です。あるアプリケーションを一つ構築する場合、それに対するプロジェクトを定義します。
構築するアプリケーションには、たくさんのアプリケーションウィンドウが存在したり、たくさんのイベントプロシージャが存在する場合があります。アプリケーション構築者が、これらのソースプログラムを一つ一つ管理をしなくても良いように、プロジェクトが次の事柄について管理します。

- プロジェクトに登録されたアプリケーションウィンドウの管理
  プロジェクトでは、アプリケーションウィンドウを登録することが出来ます。登録を行うと、コンパイル時に自動的にコンパイルされ、アプリケーションの一部となります。
- アプリケーションウィンドウに存在するイベントプロシージャの管理
  プロジェクトは登録されたアプリケーションウィンドウに存在するイベントプロシージャの管理を行います。それらはコンパイル時に自動的にコンパイルされ、アプリケーションの一部となります。
- ソースプログラムのコンパイルの管理
  プロジェクトは、管理されているアプリケーションウィンドウ、イベントプロシージャなどをコンパイル、メイクしてアプリケーションを作成します。また更新されたソースプログラムをコンパイルして、ロードモジュールを再構築する機能などがあります。
- ビルダの作業環境設定の管理
  プロジェクトは、作業環境、たとえば、デフォルトの色や大きさなどの設定、コンパイルのための、インクルードファイルやライブラリのパスの設定など、環境設定を管理します。



［プロジェクト］

Wide Studio Manual Page

2 開発要素

アプリケーションビルダーで作成されるファイルをみてみよう。プロジェクトを作成して、アプリケーションを開発する場合、いろいろなファイルが作成されます。例えば、プロジェクト project1 があり、そこに、アプリケーションウィンドウ newwin000 が存在したとすると、下記の表のようなファイルが作成されます。また、イベントプロシージャを作成した場合は、そのプログラムを記述したファイルも構成要素として扱われます。

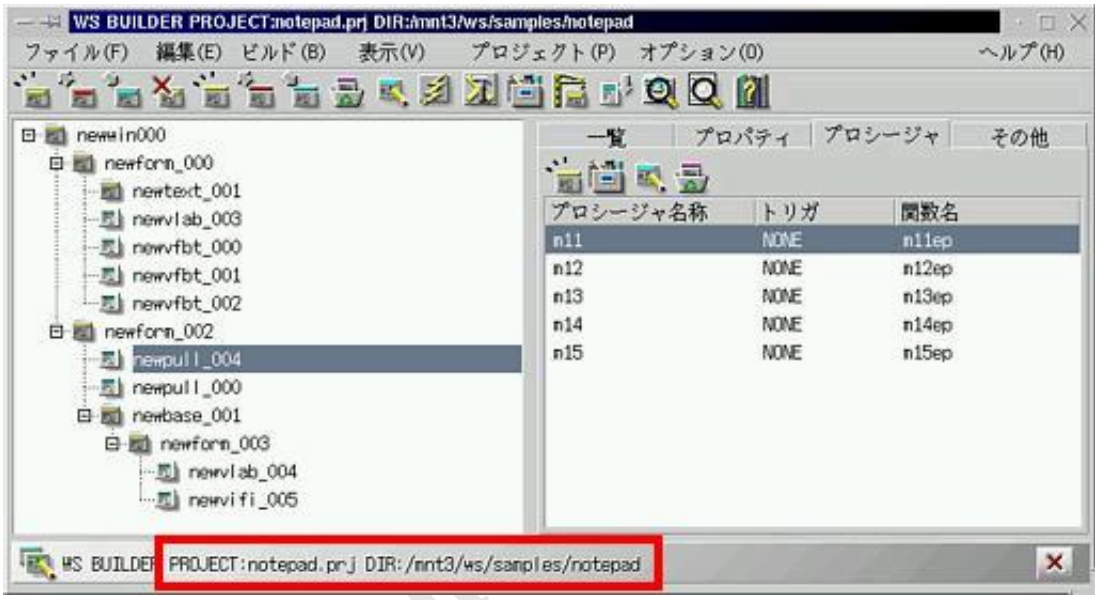
ファイル名	説明
project1.prj	プロジェクトの設定
project1.cpp	コンパイル時に自動作成されるソースファイル
project1.wns	プロジェクトに登録されているウィンドウの一覧の保持
project1.col	プロジェクトのユーザ定義色の設定の保持
newwin000.wiun	newwin000 ウィンドウの構成情報の保持
newwin000.cpp	コンパイル時に自動作成されるソースファイル

Wide Studio Manual Page

3　プロジェクトを新規作成・保存・別名保存するには

3.1　プロジェクト名称を確認するには

現在編集中のプロジェクト名称を確認するには、次の図に示すところをご覧下さい。もし空欄であれば、まだプロジェクトを定義またはロードしていないことを示します。

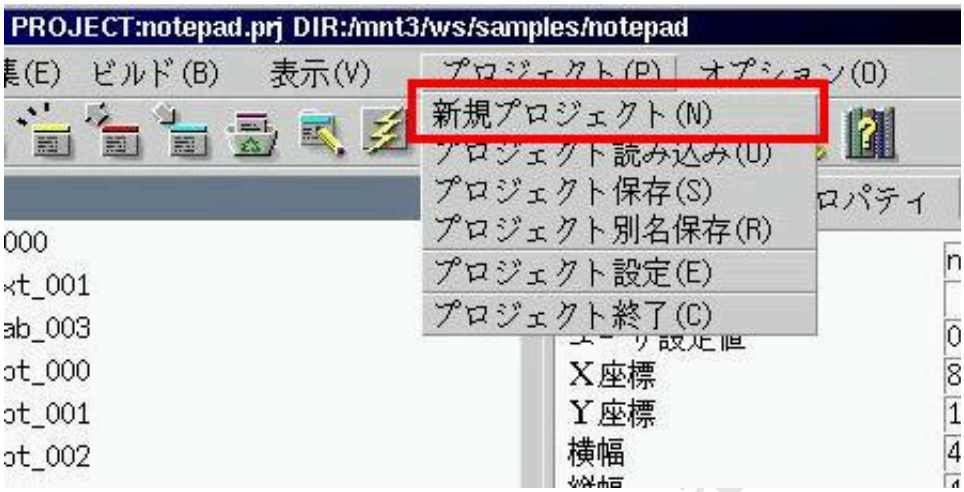


［プロジェクト名称の確認］

3.2　プロジェクトを新規作成するには

プロジェクトの新規作成をするには、［プロジェクト］メニューの新規プロジェクトを選択するか、次に示すアイコンをクリックします。

Wide Studio



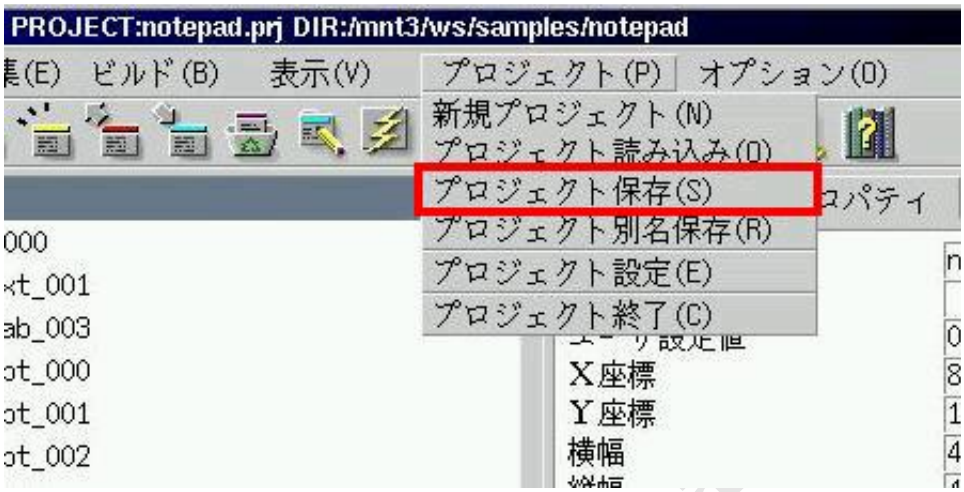
［新規プロジェクトの作成］

プロジェクト作成ウィザードが表示されますので、英数小文字でプロジェクト名称を入力や、タイプを指定し、最後にアプリケーションで使用する言語エンコーディングタイプを指定します。

［プロジェクト作成ウィザード］

3.3 プロジェクトを保存するには

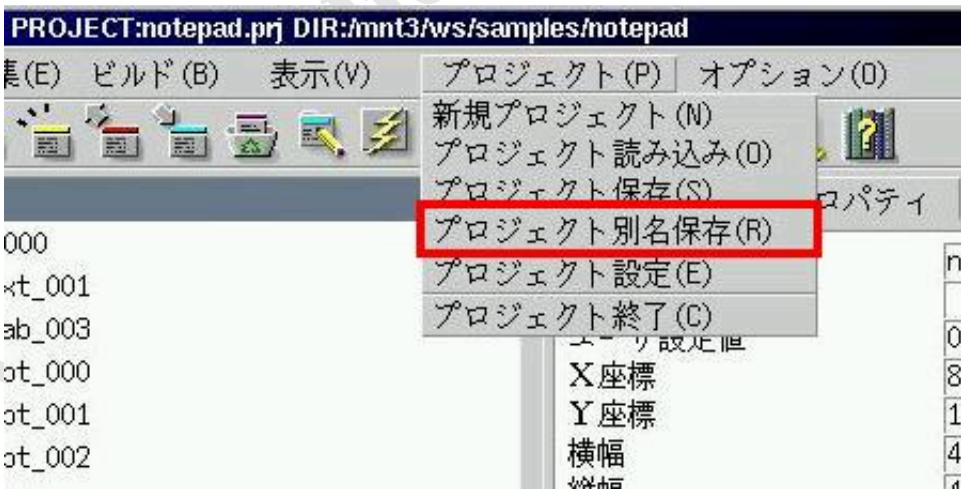
プロジェクトを保存するには、［プロジェクト］メニューの［プロジェクト保存］を選択します。



［プロジェクトの保存］

3.4 プロジェクトを別名保存するには

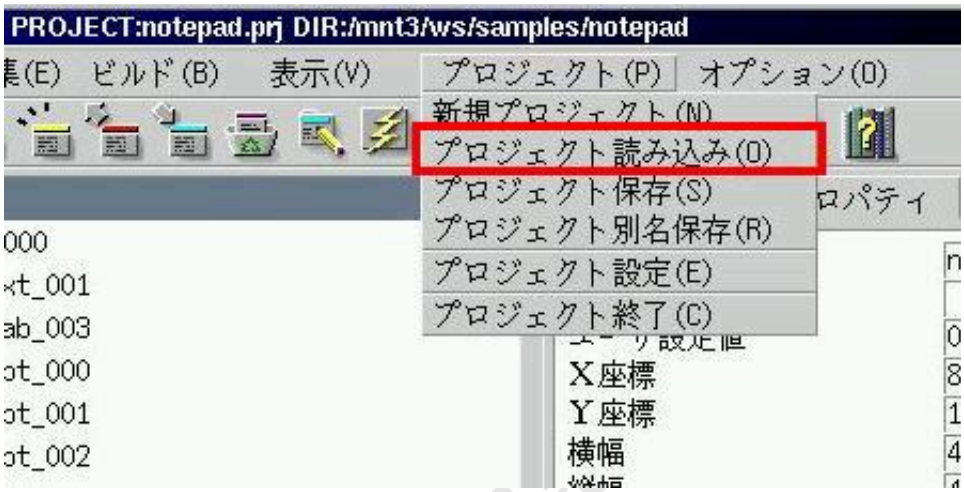
プロジェクトの名称を変更する場合、または別名保存するには、［プロジェクト］メニューのプロジェクト保存を選択します。ファイル選択ダイアログが表示され、ファイル名を指定して保存します。



［プロジェクトの別名保存］

3.5 プロジェクトを読み込むには

作成済みのプロジェクトを読み込むには、［プロジェクト］メニューのプロジェクトの読み込みを選択します。

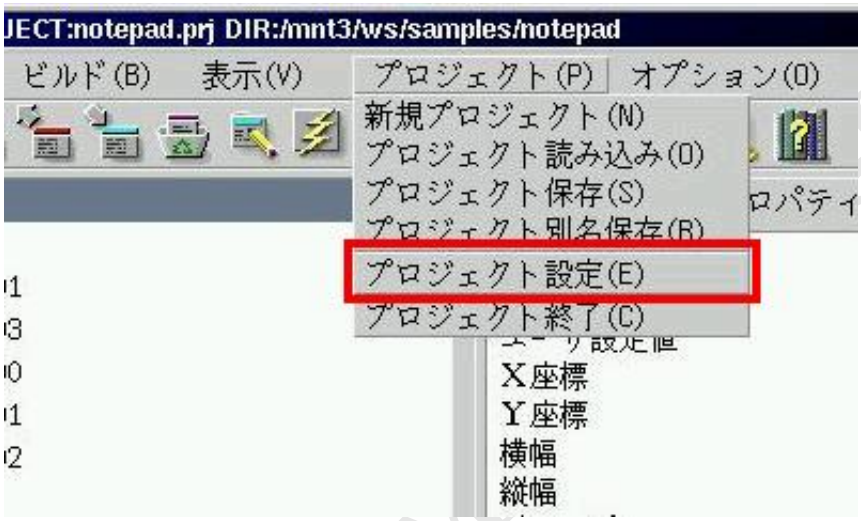


［プロジェクトの読み込み］

Wide Studio Manual

4　プロジェクトの環境を設定するには

プロジェクトの作業環境の設定は、プロジェクト設定で行います。［プロジェクト］メニューの［プロジェクト設定］を選択します。



［プロジェクト設定ダイアログの表示］

次のような項目の環境設定を行うことができます。

- デフォルト値の指定
  オブジェクトを配置する際、オブジェクトにあらかじめデフォルトの大きさ、色等を指定することができます。

- 環境設定
  ターミナル、ブラウザ、エディタ等のコマンドを指定します。

- 追加モジュール指定
  クラスライブラリを追加し、あらたなオブジェクトの利用ができます。オブジェクト編の［追加ライブラリによる新たなオブジェクトを利用するには］の節を参照下さい。

- コンパイル設定
  コンパイル時の、インクルードパスの設定、コンパイルフラグの設定、使用するコンパイラ、コンパイルモードの指定等の設定を行います。

- リンク設定
  リンク時の、ライブラリパスの設定、リンクするライブラリ、リンクフラグの設定、使用するリンカ、使用するデバッガ等の設定を行います。

Wide Studio Manual

- フォントの設定
  フォントの設定を行います。

［プロジェクト設定ダイアログ］

各項目を設定したら、［OK］ボタンを押して反映させます。なお環境設定の情報は、ファイル［プロジェクト名称.prj］に出力されます。このファイルはテキストファイルなので、テキストエディタで編集可能です。

4.1 配置するオブジェクトのデフォルト値の設定

［デフォルト値］タブを選択し、デフォルトの大きさで、横幅、縦幅、影幅を指定することができます。

4.2 配置するオブジェクト座標のグリッド値の設定

［デフォルト値］タブを選択し、グリッド指定のところで、横グリッド単位、縦グリッド単位をそれぞれ指定することができます。

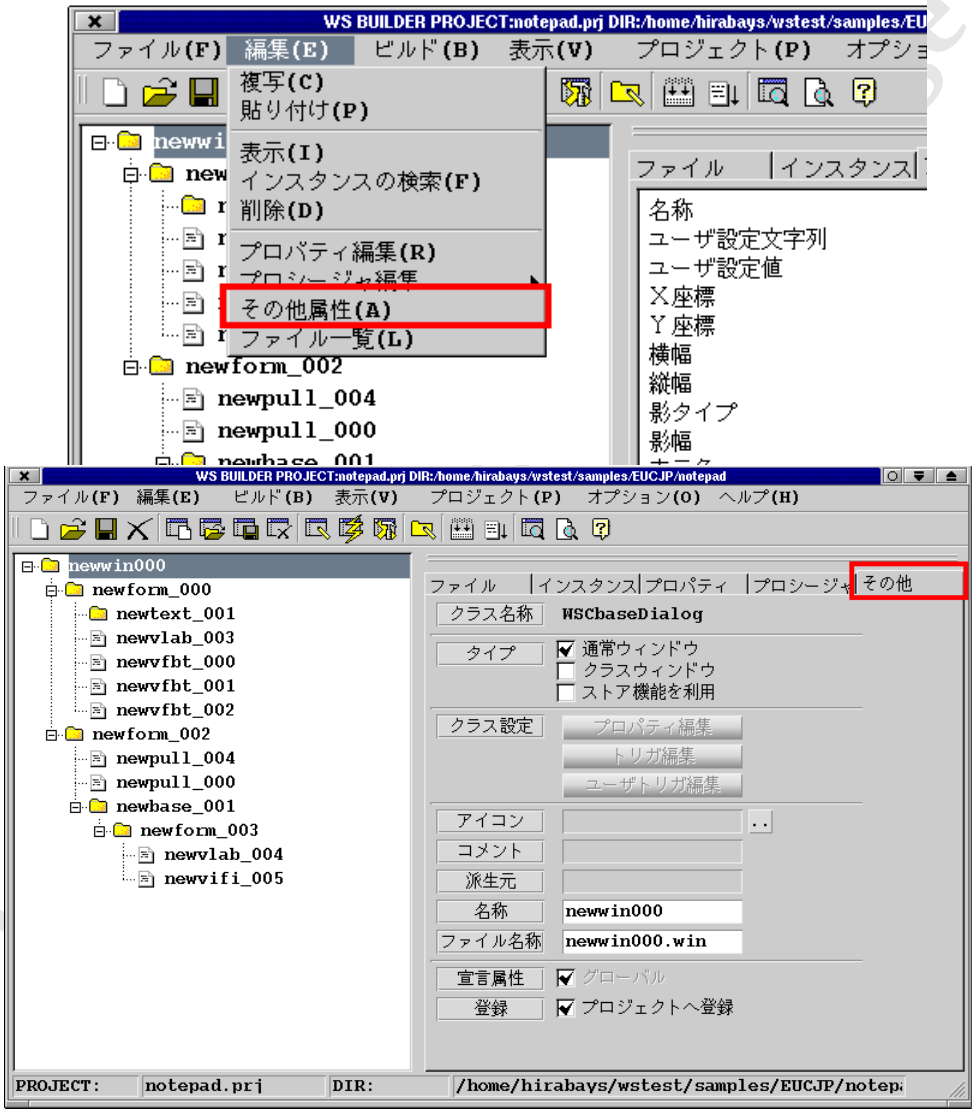
4.3 ヘルプブラウザ・ソースコードエディタの設定

［環境設定］タブを選択し、Web ブラウザの指定項目で、ブラウザのコマンドを設定します。

Wide Studio Ma

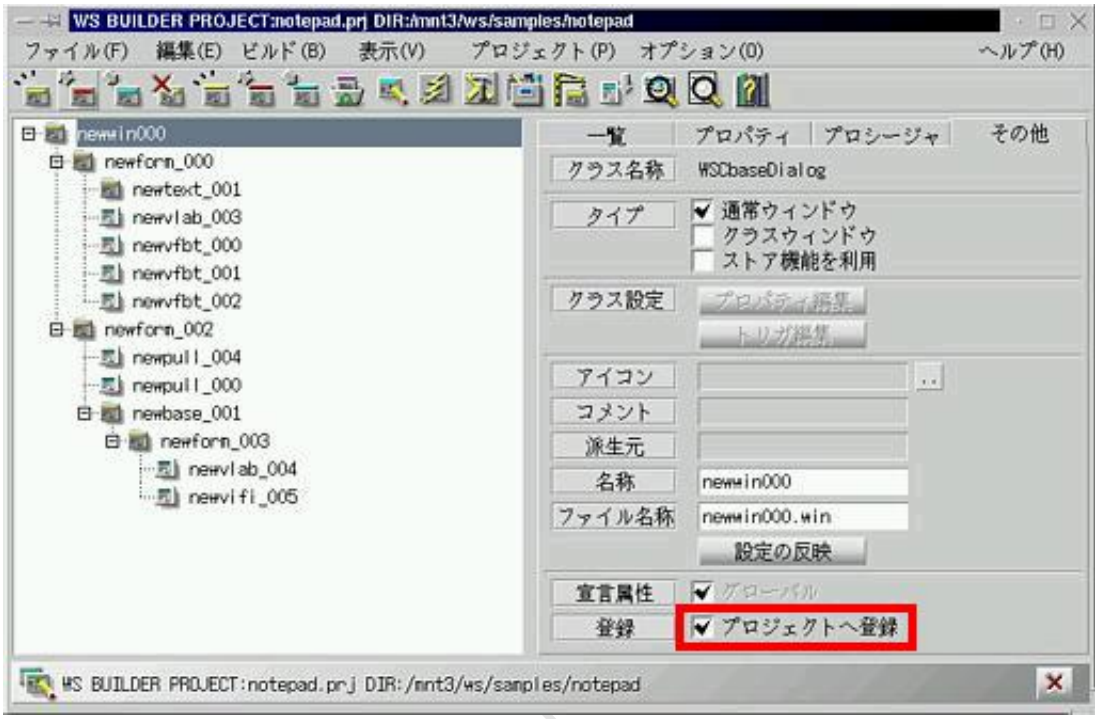
5　プロジェクトにアプリケーションウィンドウを登録するには

アプリケーションウィンドウをプロジェクトに登録することができます。登録されたアプリケーションウィンドウは、プロジェクトの読み込み時に自動的に読み込みされます。
アプリケーションウィンドウをプロジェクトに登録するには、
まず、登録したいアプリケーションウィンドウを選択して、インスペクタの基本設定を表示します。



［基本設定の表示］

次に図に示す［登録］の項目をチェックします。



［アプリケーションウィンドウの登録］

プロジェクトの保存を行うと、アプリケーションウィンドウの登録情報は、ファイル［プロジェクト名称.wns］に出力されます。このファイルはテキストファイルなので、テキストエディタで編集可能です。

Wide Studio

6　色を追加するには

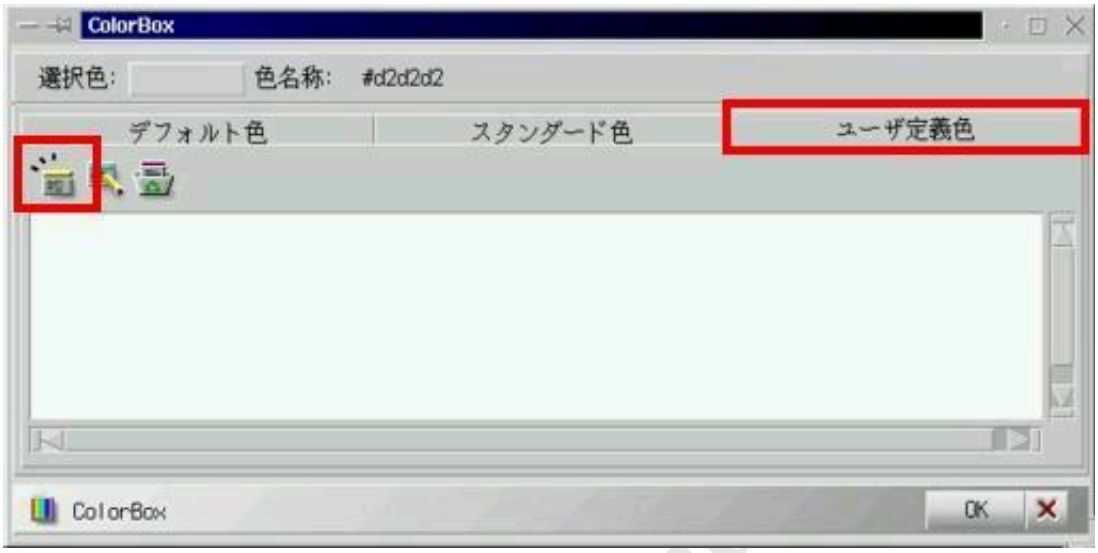
6.1　色を追加するには

色選択ダイアログに色を追加することができます。［オプション］メニューの［色設定］を選択して色選択ダイアログを表示してください。

［色選択ダイアログの表示］

色を追加したい場合、図に示すアイコンをクリックします。

Wide Studio



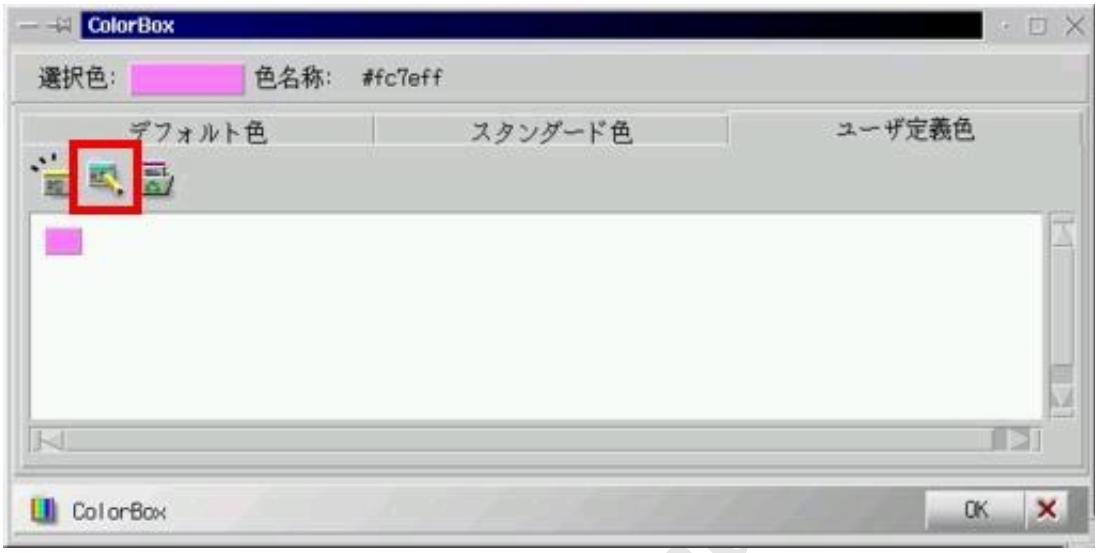
［色編集ウィンドウの表示］

色編集ウィンドウで色の設定を行います。左の色マップで、直接色を選択するか、右側のスライダで数値を調整して色を調整します。もちろん、直接色名称を指定しても構いません。色が決まったら、［ＯＫ］をクリックします。

［色編集ウィンドウ］

7　色を編集するには

編集したい色を選択した状態で、編集ボタンをクリックします。ただし、ユーザが追加したものに限ります。

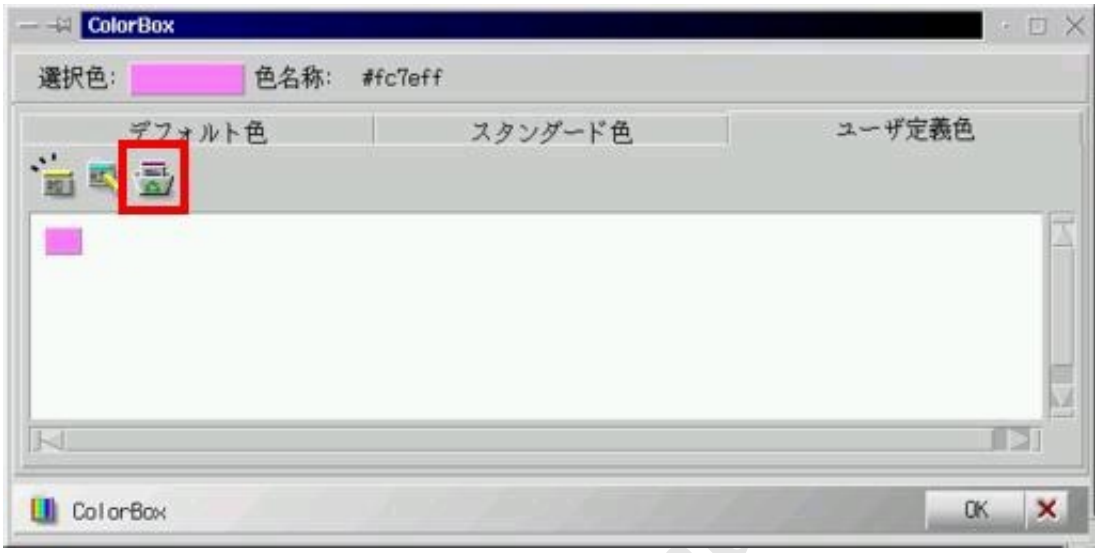


［色の編集］

8　色を削除するには

削除したい色を選択した状態で、削除ボタンをクリックします。ただし、ユーザが追加したものに限ります。

Wide Studio Manual

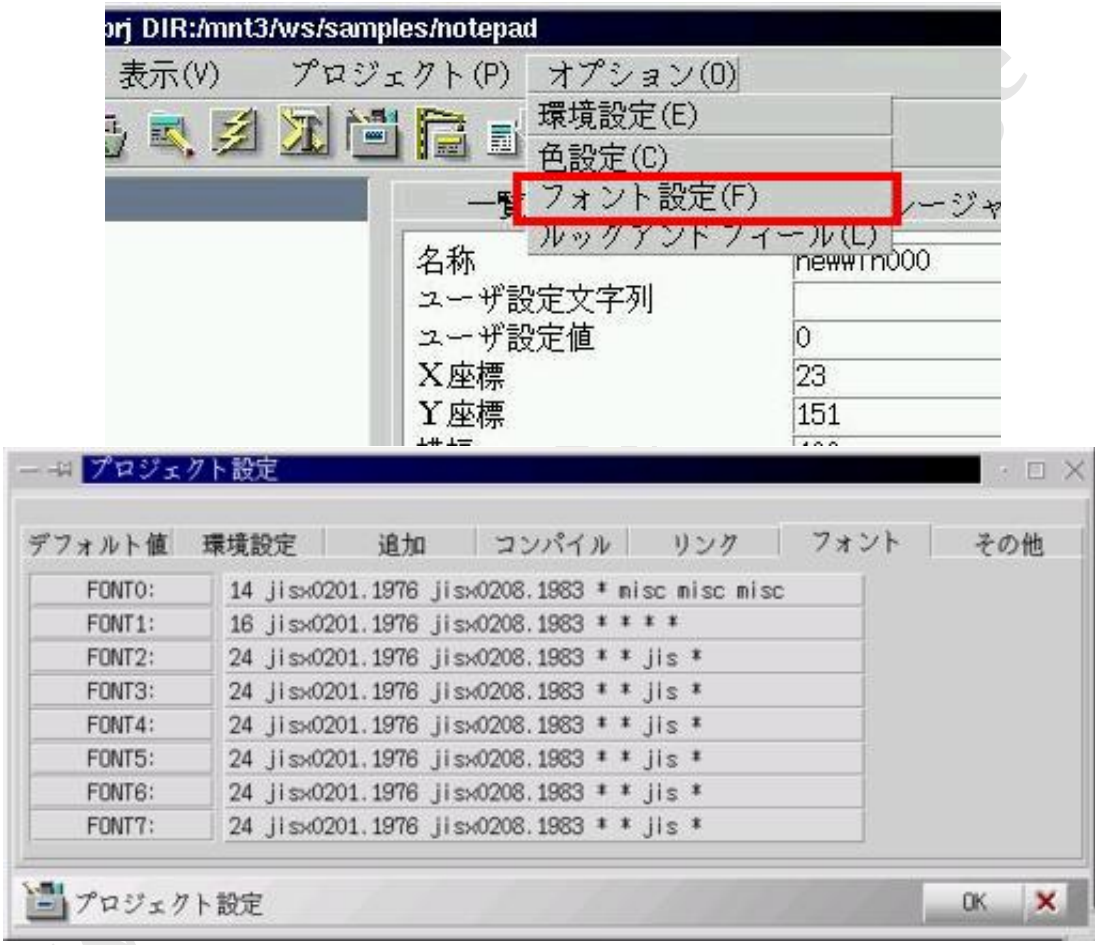


［色の削除］

Wide Studio Manual

9 フォントテーブルを設定するには

フォントテーブルを編集することができます。［オプション］メニューの［フォント設定］を選択し次のようにフォント設定ウィンドウを表示してください。次に編集したいフォントのボタンをクリックします。



［フォント設定ウィンドウの表示］

フォントウィンドウでフォントの設定を行います。図の様に設定したいフォントを選んでください。次の場合は、UNIX のシステムの例です。

［UNIX システムでのフォントの選択］

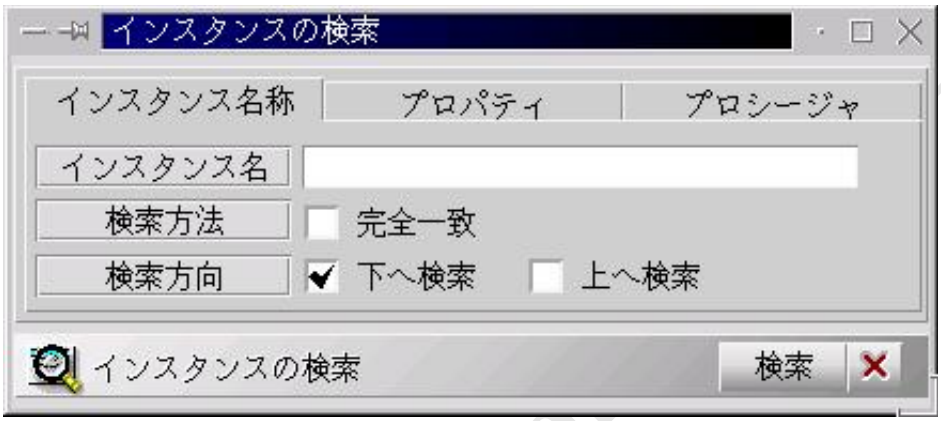
UNIX システムでのフォントの選び方は、次の順で行いましょう。

- １． 英数ベンダ/漢字ベンダ/外字ベンダを * にします。
- ２． 好きなドットを選びます。
- ３． 選択可能な、英数ベンダ/漢字ベンダ/外字ベンダを選択します。
- ４． 選択が終わったら設定ボタンをマウスクリックします。

Wide Studio Manual Page

10　ウィンドウ/オブジェクトの検索

プロジェクトに存在するアプリケーションウィンドウやオブジェクトを、インスタンス名称やプロパティ値、イベントプロシージャ等をキーにして、検索することができます。
［編集］メニューの［オブジェクトの検索］を選択します。



［ウィンドウ/オブジェクトの検索］

オブジェクトの検索には、次にあげる３通りの方法があります。

- インスタンス名称による検索方法
  インスタンス名称を検索キーとして与えます。
- プロパティ値による検索方法
  プロパティ値を検索キーとして与えます。プロパティ名が与えられた場合、インスタンスがそのプロパティを持っていて、その値が、指定された値であるものが検索対象となります。
  プロパティ名には、WSNxxx のものを指定します。
  プロパティ値による検索を行う場合、必ず、プロパティ値が指定されなければなりません。
- プロシージャによる検索
  プロシージャ名、または、トリガ、関数名のうち、組み合わせて検索キーとして与えます。

検索した結果、ヒットするインスタンスが存在した場合、インスペクタ上に表示されます。また、そのまま、検索ボタンを押すと、次のインスタンスが検索されます。

Wide Studio Manual

［プロパティによる検索］

［プロシージャによる検索］

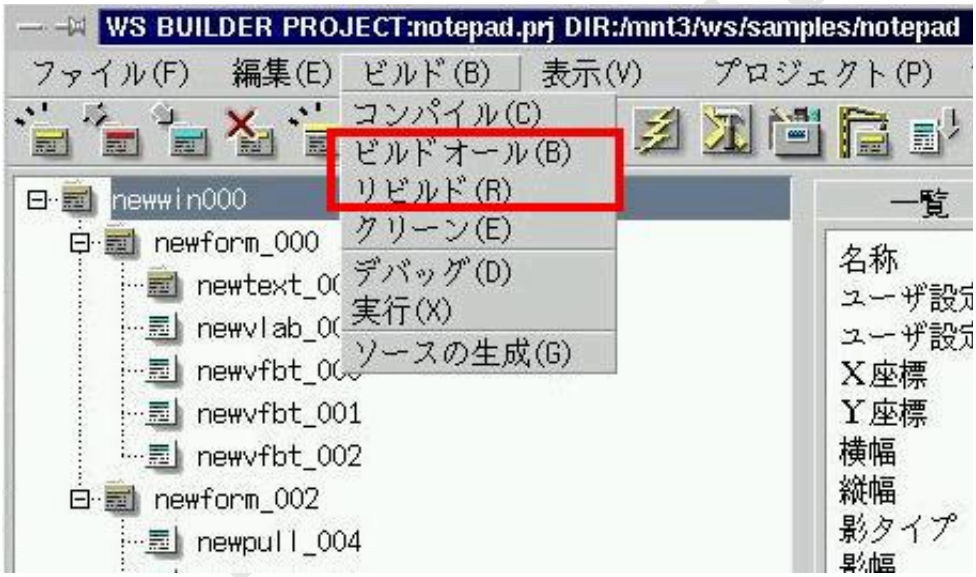
Wide Studio Manual

G　コンパイル・ビルド編

1　プロジェクトをビルドするには

1.1　プロジェクトをビルドするには

アプリケーションビルダで作成したプロジェクトを実行可能にするには、コンパイルし、リンクする必要があります。ビルドでは、自動的にコンパイルし、作成されたアプリケーションウィンドウ、オブジェクトのクラスライブラリ、ユーザの指定するライブラリなどをリンクして、ロードモジュールを作成します。［ビルド］メニューの［ビルドオール］、［リビルド］を選択してください。なお、［リビルド］は、一度、オブジェクトファイル等を削除し、最初からコンパイルしなおします。

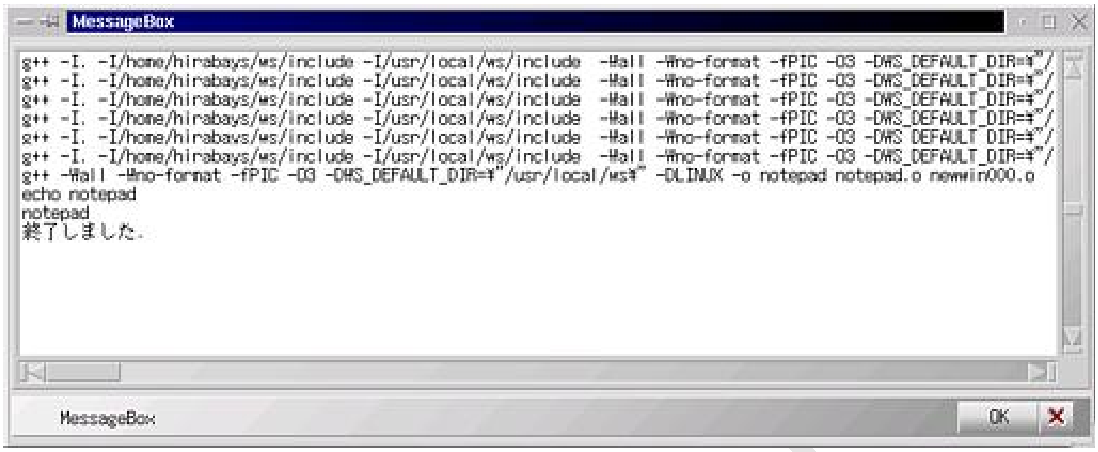


［プロジェクトのビルド］

- ビルドオール
  変更された部分のみ、コンパイルを走らせます。再度の必要のないコンパイルをなるべく避けて、高速にロードモジュールを作成します。

- リビルド
  全ファイルに関して、コンパイルを走らせます。ヘッダーファイルや、バージョンなどが変更されて、コンパイルをやり直さなければならない場合に指定します。

コンパイルが始まると次のようにコンパイル情報ウィンドウが表示されます。このウィンドウはコンパイルした情報を表示します。エラーが出た場合などこの情報を確認します。
またコンパイルを中止する場合は、中止ボタンをクリックして下さい。

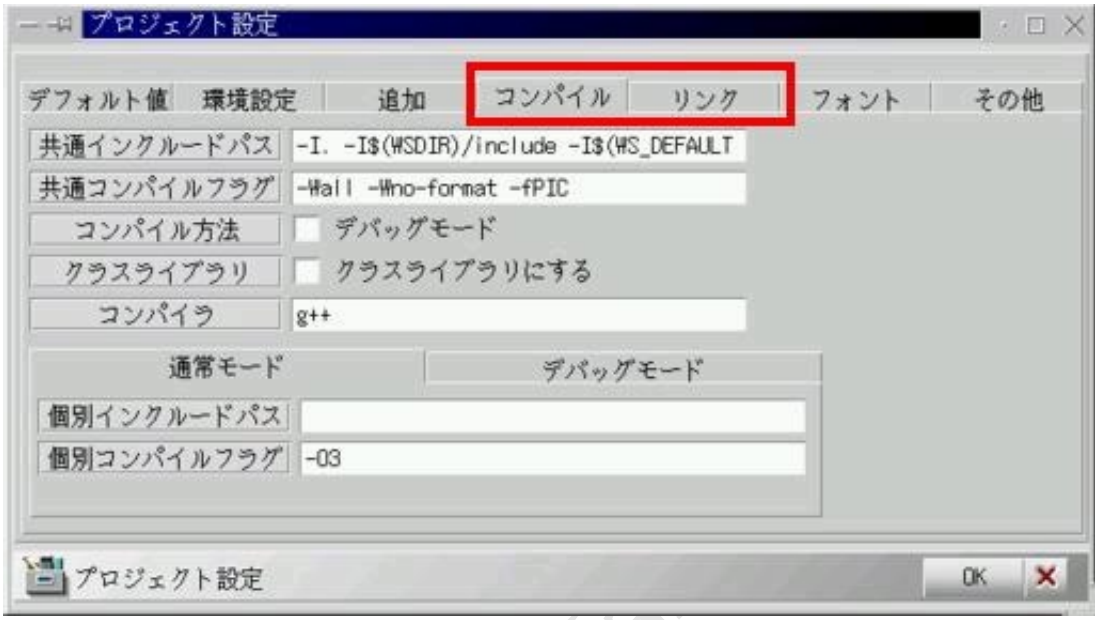


［コンパイル情報ウィンドウ］

1.2 インクルードパスやリンクするライブラリを指定するには

コンパイルする場合のインクルードパス、コンパイルのモード、フラグの指定、リンクするライブラリを指定することができます。［プロジェクト］メニューの［プロジェクト設定］を選択して、コンパイル、リンクをそれぞれ選択してください。

Wide Studio Manual



［コンパイル、リンクの設定］

次の項目が設定できます。

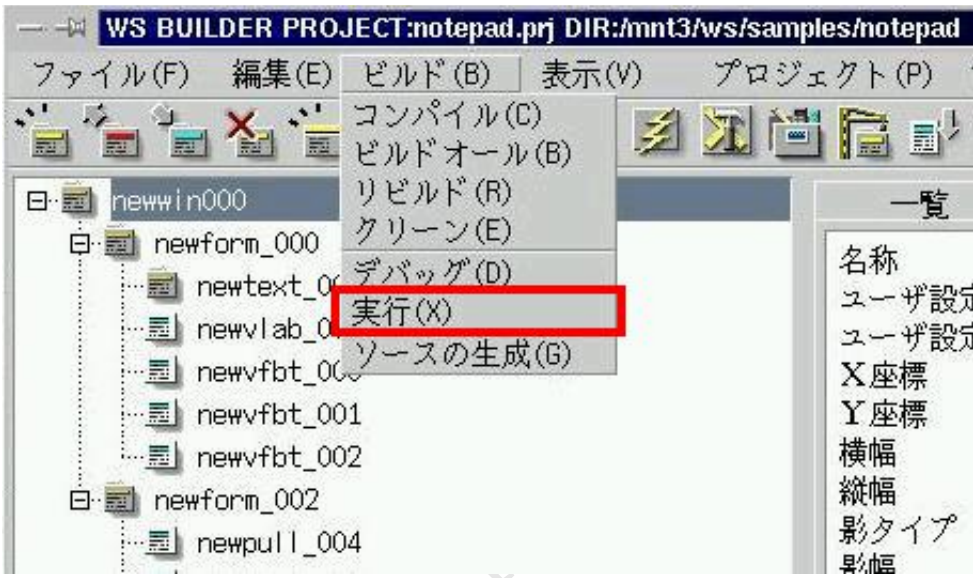
- インクルードパス指定
  追加オブジェクト用のヘッダや、ユーザの指定するインクルードの存在するディレクトリを指定します。一般のＵＮＩＸのコンパイラの場合、次の書式に従って指定します。
  -I インクルードパス１　-I インクルードパス２　...
- コンパイルフラグの指定
  コンパイラに引き渡す引数を指定します。
- コンパイル方法の指定
  通常のコンパイルか、デバッグモードかを指定します。
- クラスライブラリの指定
  ロードモジュールを作成するか、ダイナミックリンクライブラリを作成するかを指定します。
- コンパイラの指定
  使用するコンパイラを指定します。
- ライブラリの指定
  追加オブジェクト用のライブラリや、ユーザの指定するライブラリの存在するディレクトリを指定します。一般のＵＮＩＸのリンカの場合、次の書式に従って指定します。
  -L ライブラリパス１　-l ライブラリ１　-L ライブラリパス２　-l ライブラリ２ ...

- リンカフラグの指定
  リンカに引き渡す引数を指定します。
- リンカの指定
  使用するリンカを指定します。
- デバッガの指定
  使用するデバッガを指定します。

Wide Studio Manual Page

2　作成したアプリケーションを実行するには

アプリケーションビルダで作成したプロジェクトがビルドされてアプリケーションが実行可能であれば、実行することができます。［ビルド］メニューの［実行］を選択してください。



［作成したアプリケーションの実行］

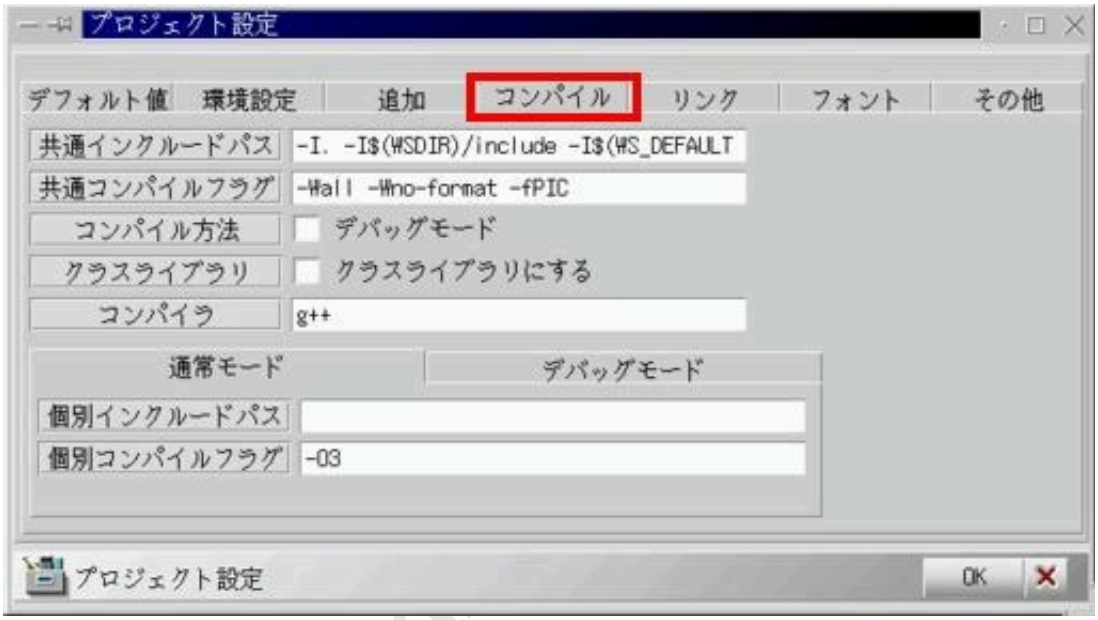
実行を中止するには、同じメニューの実行中止を選択してください。

Wide Studio

3　コンパイルオプションの設定

3.1　コンパイルオプションの設定

コンパイル時のインクルードパス、コンパイルのモード、フラグの指定等の設定を行う場合は、［プロジェクト］メニューの［プロジェクト設定］を選択して、コンパイルを選択してください。



［コンパイルの設定］

次の項目が設定できます。

- インクルードパス指定
  追加オブジェクト用のヘッダや、ユーザの指定するインクルードの存在するディレクトリを指定します。一般のＵＮＩＸのコンパイラの場合、次の書式に従って指定します。
  -I インクルードパス１　-I インクルードパス２　...

- コンパイルフラグの指定
  コンパイラに引き渡す引数を指定します。

- コンパイル方法の指定
  通常のコンパイルか、デバッグモードかを指定します。

- クラスライブラリの指定
  ロードモジュールを作成するか、ダイナミックリンクライブラリを作成するかを指定します。

Wide Studio

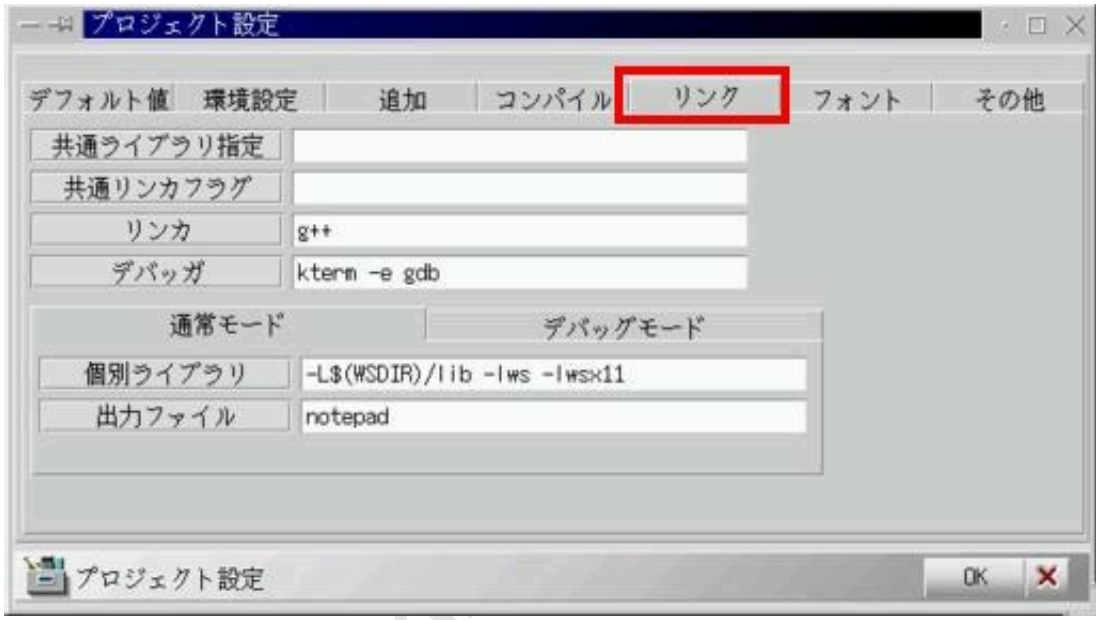
- コンパイラの指定
  使用するコンパイラを指定します。
- 追加オブジェクトの指定
  追加したいソースコードのオブジェクトファイル名を列挙して指定します。
- 通常モード
  通常モードでの、個別に指定するインクルードパス、コンパイルオプション等を指定します。
- デバッグモード
  デバッグモードでの、個別に指定するインクルードパス、コンパイルオプション等を指定します。

Wide Studio Manual Page

4　リンクするライブラリの設定

4.1　リンクするライブラリの設定

リンク時のライブラリの指定や、オプションの指定は、［プロジェクト］メニューの［プロジェクト設定］を選択して、リンクを選択してください。



［リンクの設定］

次の項目が設定できます。

- ライブラリの指定
  追加オブジェクト用のライブラリや、ユーザの指定するライブラリの存在するディレクトリを指定します。一般のＵＮＩＸのリンカの場合、次の書式に従って指定します。
  -L ライブラリパス１　-l ライブラリ１　-L ライブラリパス２　-l ライブラリ２ ...
- リンカフラグの指定
  リンカに引き渡す引数を指定します。
- リンカの指定
  使用するリンカを指定します。
- デバッガの指定
  使用するデバッガを指定します。

Wide Studio

- 通常モード
  通常のリンクで個別に使用するライブラリや、ユーザの指定するライブラリの存在するディレクトリ、生成するモジュール名を指定します。

- デバッグモード
  デバッグモードで個別に使用するライブラリや、ユーザの指定するライブラリの存在するディレクトリ、生成するモジュール名を指定します。

Wide Studio Manual Page

5　ソースファイル追加の設定

5.1　ソースファイル追加の設定

プロジェクトに自作したソースファイルを追加することができます。［プロジェクト］メニューの［プロジェクト設定］を選択して、コンパイルを選択します。
次に、［追加オブジェクト］の欄に、追加したいソースコードのオブジェクト名を列挙して指定します。

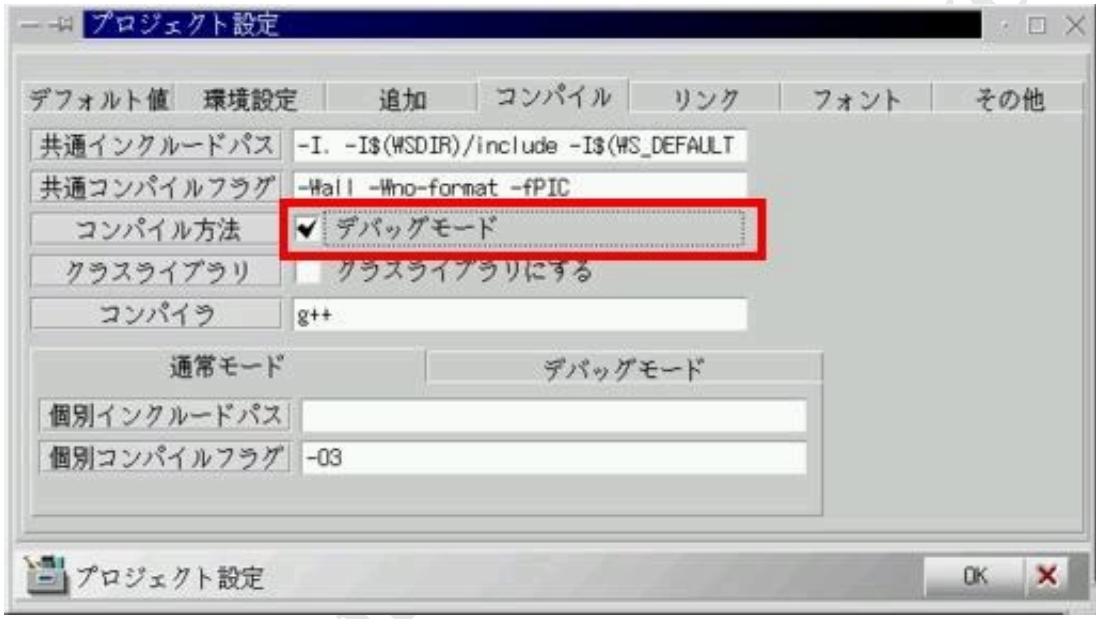
例えば src1.c 、src2.c 、src3.c をプロジェクトに追加したい場合は、次のように指定します。

```
src1.o src2.o src3.o
```

Wide Studio Manual Page

6　デバッグモードの指定

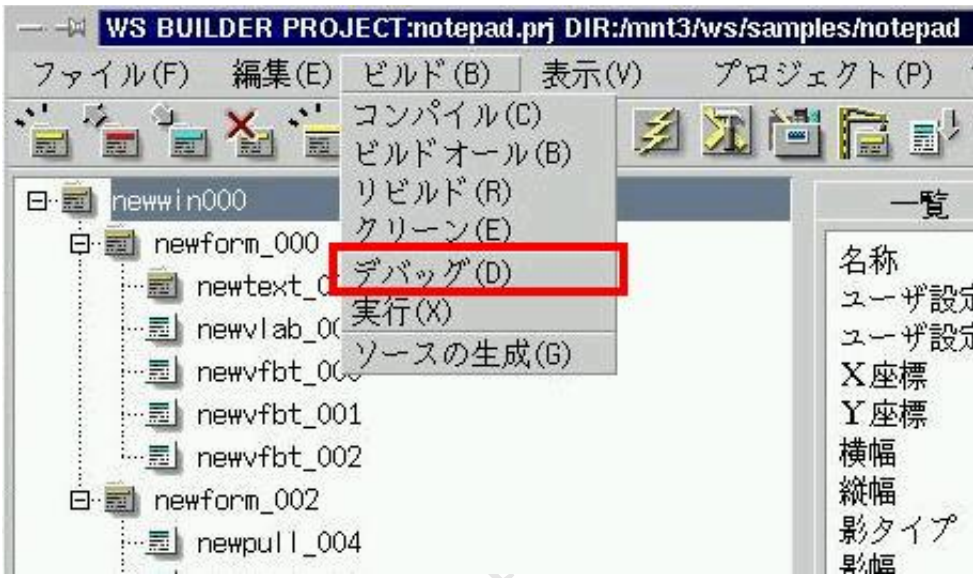
プロジェクトをデバッグモードでビルドすることができます。デバッグモードでビルドすると、デバッガなどでのデバッグ作業が効率良く行えます。
［プロジェクト］メニューの［プロジェクト設定］を選択して、コンパイルを選択してください。デバッグモードの欄をチェックすると、デバッグモードとなります。
（注意）デバッグモードを指定した場合は、一度、リビルドしてください。



［デバッグモードの指定］

7　デバッグの方法

作成したモジュールをデバッガにかけてデバッグすることができます。まず、デバッグモードでビルドします。つぎに、［ビルド］メニューの［デバッグ］を選択します。



［デバッグの実行］

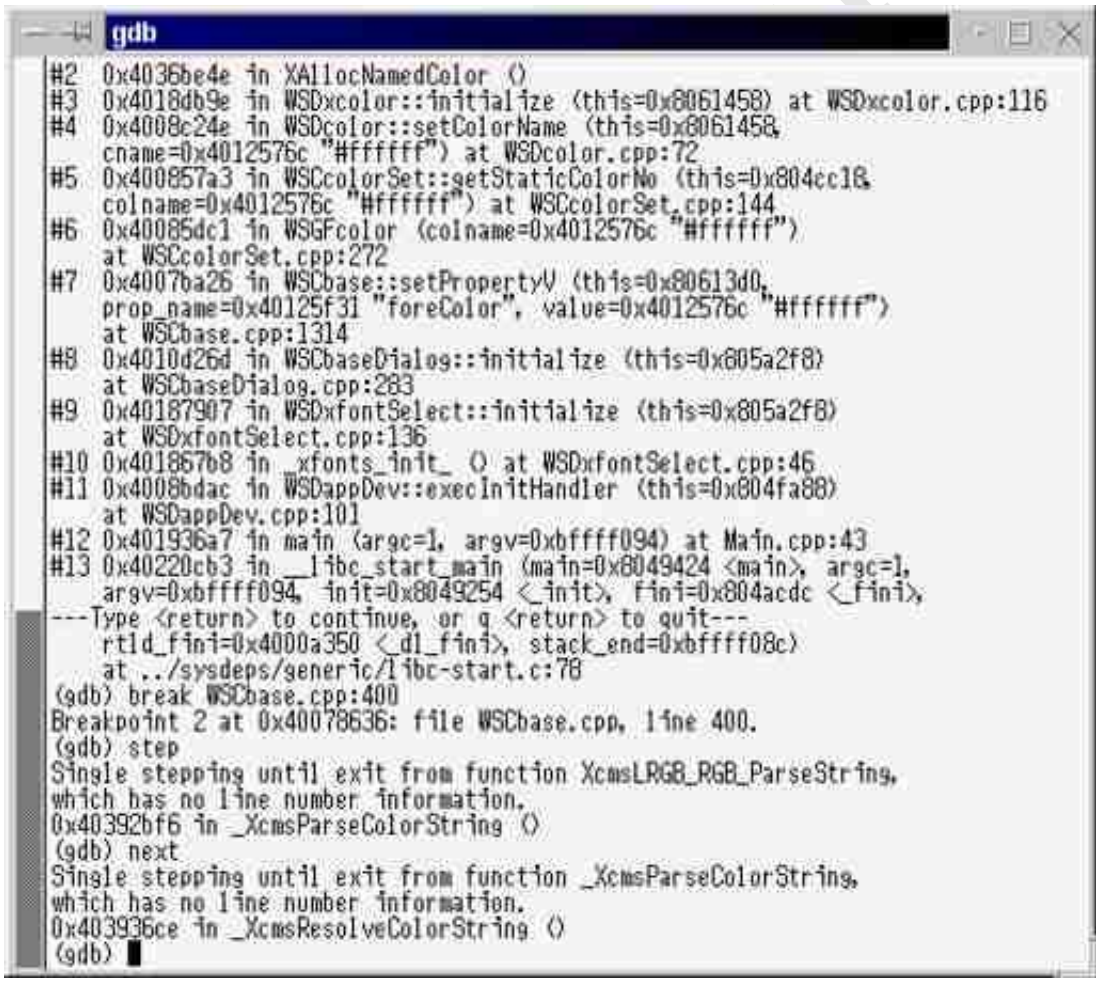
ここでは簡単に、gdb の使い方について触れてみます。まず、run とキー入力してアプリケーションを実行します。次に、プログラムにバグがあり、エラーが発生して停止した場合は、where と入力してスタックトレースを参照し、どこの関数で落ちているかを調べます。list と入力すると、エラーの発生したソースの位置を見ることができます。
主なコマンドは次の通りです。

- run
  実行を開始します。
- cont
  中断した実行を再開します。
- step/next
  中断した実行を１行ずつ実行します。
- where
  現在の関数呼出状況を表示します。
- print
  変数の内容を表示します。

- list
  現在の位置のソースの内容を表示します。

- break
  指定した箇所にブレークポイントを設けます。書式は
  break file.cpp:XXX
  break CLASS::FUNCTION()
  等です。XXX には、行数を指定します。

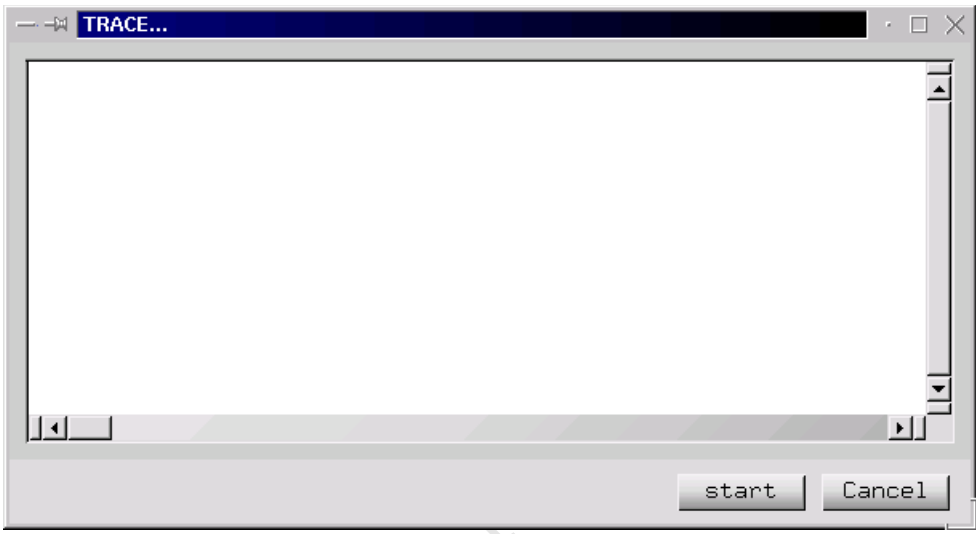
- cntl-C
  実行中のプログラムを停止させます。



［デバッグをしている様子］

8　トレースデバッグの方法

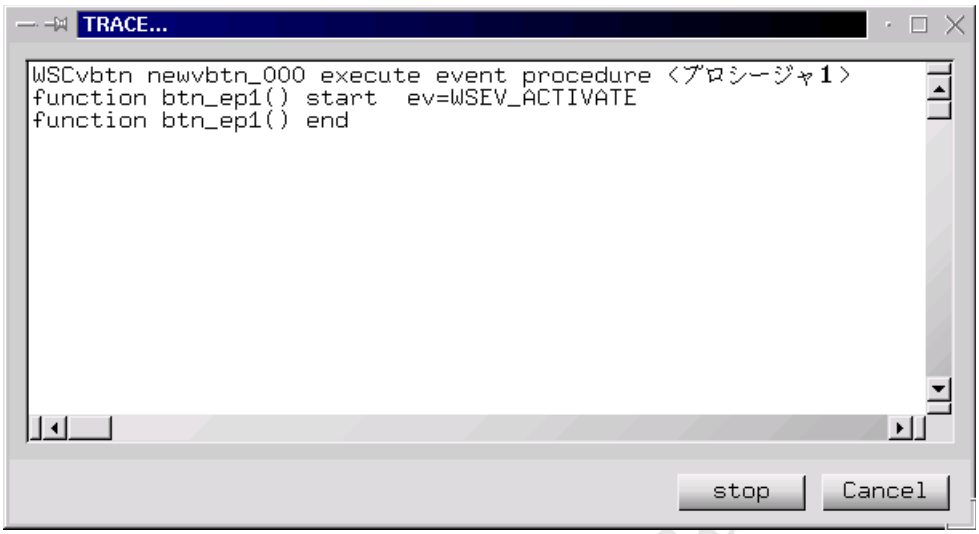
作成したモジュールのイベントプロシージャの実行状況をトレースしデバッグすることができます。まず、アプリケーションをビルドします。次に、［ビルド］メニューの［トレース実行］を選択します。すると次の図のようなトレースダイアログが表示されます。



［トレースダイアログ］

そして、トレースを開始するには、トレースダイアログの［start］ボタンを押下してください。イベントプロシージャが起動されると、トレースダイアログに次のように出力されます。また、トレース出力を停止するには、［stop］ボタンを押下してください。

Wide Studio



［トレース出力］

もし、トレース出力中に、あるイベントプロシージャの

function 関数名 ( ) end

が出力されず、アプリケーションが異常終了する場合、そのイベントプロシージャのプログラムに異常があることがわかります。

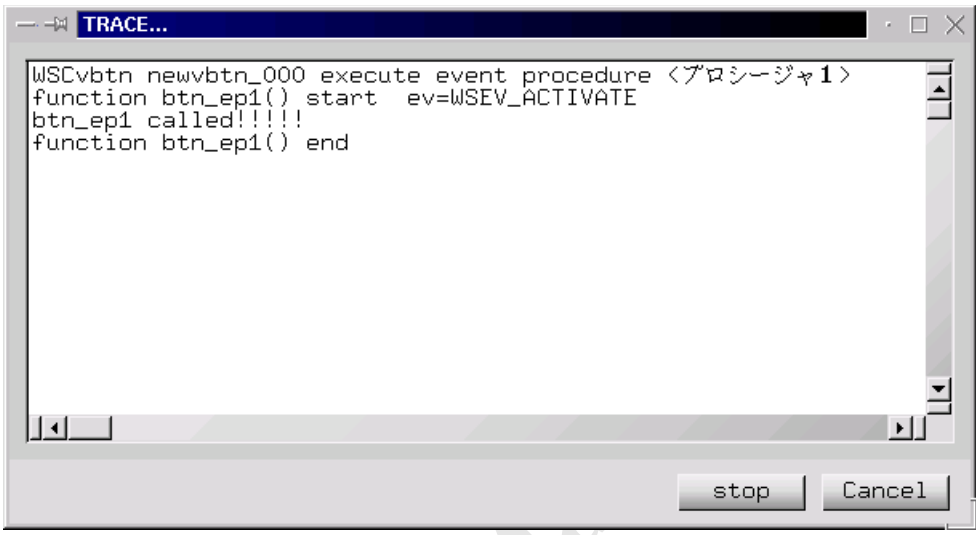
また、次のように (A) の WSGFtrace() 関数によってアプリケーションプログラムからもトレース出力を行うことができます。

```
\#include <WScom.h>
\#include <WSCfunctionList.h>
\#include <WSCbase.h>
//-----------------------------------------------
//Function for the event procedure
//-----------------------------------------------
void btn\_ep1(WSCbase* object){

  object->setProperty(WSNlabelString,"Hello!");
  WSCstring string;
  string = "btn\_ep1 called!!!!!\n";
  WSGFtrace(string);                              //(A)
}
```

```
static WSCfunctionRegister  op("btn\_ep1",(void*)btn\_ep1);
```

なお、WSGFtrace( ) による出力は、トレース出力時のみ有効となります。



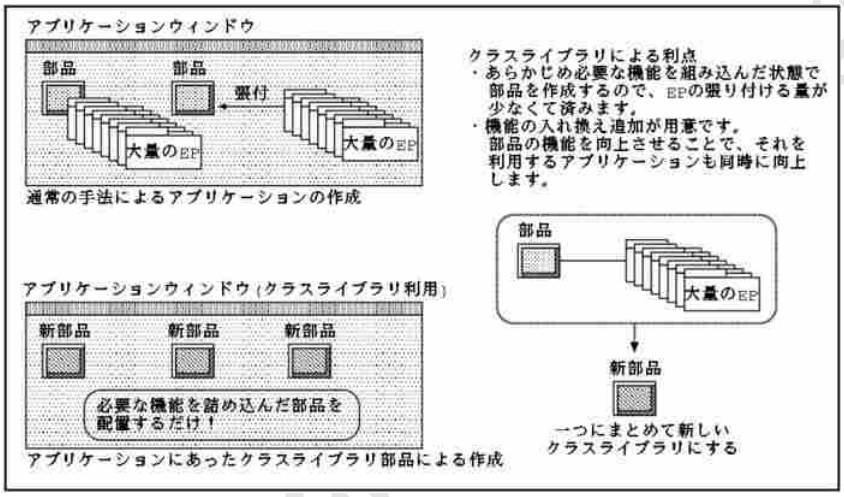
［アプリケーションによるトレース出力］

Wide Studio Manual

H　クラスウインドウ編

1　クラスアプリケーションウィンドウとは

WideStudio 用の C++ オブジェクトクラスをもとに、新たにより複雑なオブジェクトを派生して作成することができます。

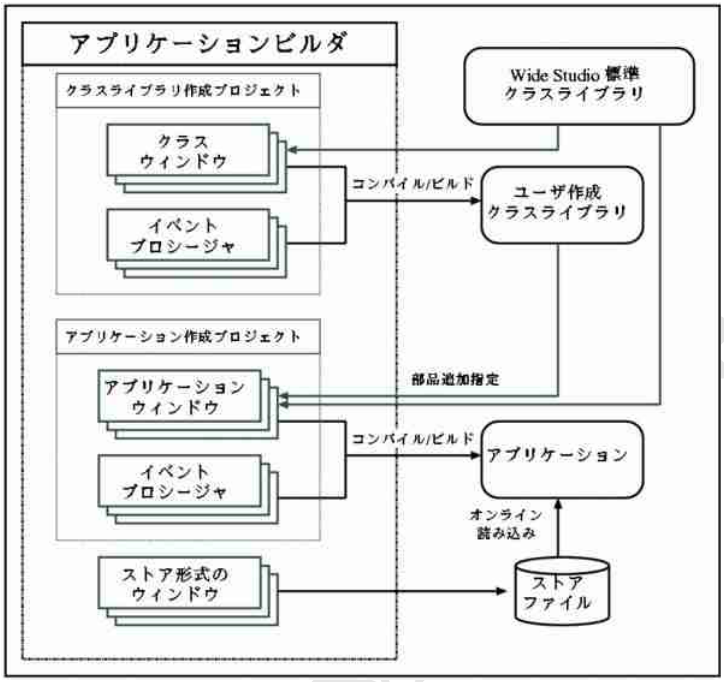


アプリケーションビルダでは、クラス化の機能を使うことで本来であればソースプログラムで記述しなければならない C++ クラスの派生を、自動で生成します。作成したクラスに対し、ユーザは新たなメンバ関数などの追加を行う形で、新しいクラスを定義します。
新しいオブジェクトの派生をサポートするのがクラスウィンドウです。クラスウィンドウは次の要素を持ちます。

- 通常のアプリケーションウィンドウと全く変わらないオブジェクトの編集
  通常のアプリケーションウィンドウと全く変わらない操作方法で、クラスウィンドウは編集できます。

- 新しいプロパティの定義
  マウス操作で、クラスウィンドウに対して、新たにプロパティを定義することができます。ソースプログラムは自動生成されます。

- クラスライブラリ構築
  プロジェクトをクラスライブラリとしてコンパイルする指定を行うことができます。プロジェクトはこの場合、実行可能モジュールではなく、オブジェクトのライブラリとしてコンパイル・リンクされます。



新しいオブジェクトは次の様な形態で派生されます。

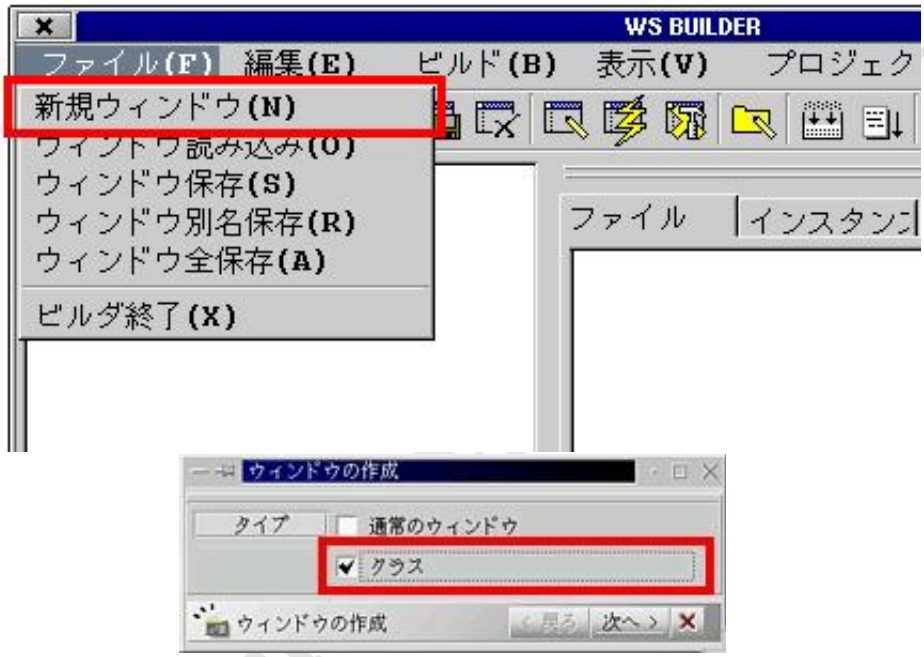
アプリケーションウィンドウ派生	一つのアプリケーションウィンドウ全体を一つの部品として派生します。
複合派生	アプリケーションウィンドウ上に配置された一部のオブジェクト群を一つの部品として派生します。
部品派生	一つのオブジェクトを基に新しい部品を派生します。

Wide Studio Manual

2 クラスウィンドウ (C++ クラス) を作成するには

2.1 クラスウィンドウ (C++ クラス) を作成するには

クラスウィンドウを作成する場合は、［ファイル］メニューの [新規ウィンドウ] を選択します。 次に、ウィンドウ作成ウィザードで、クラスを指定して、ウィンドウを作成します。

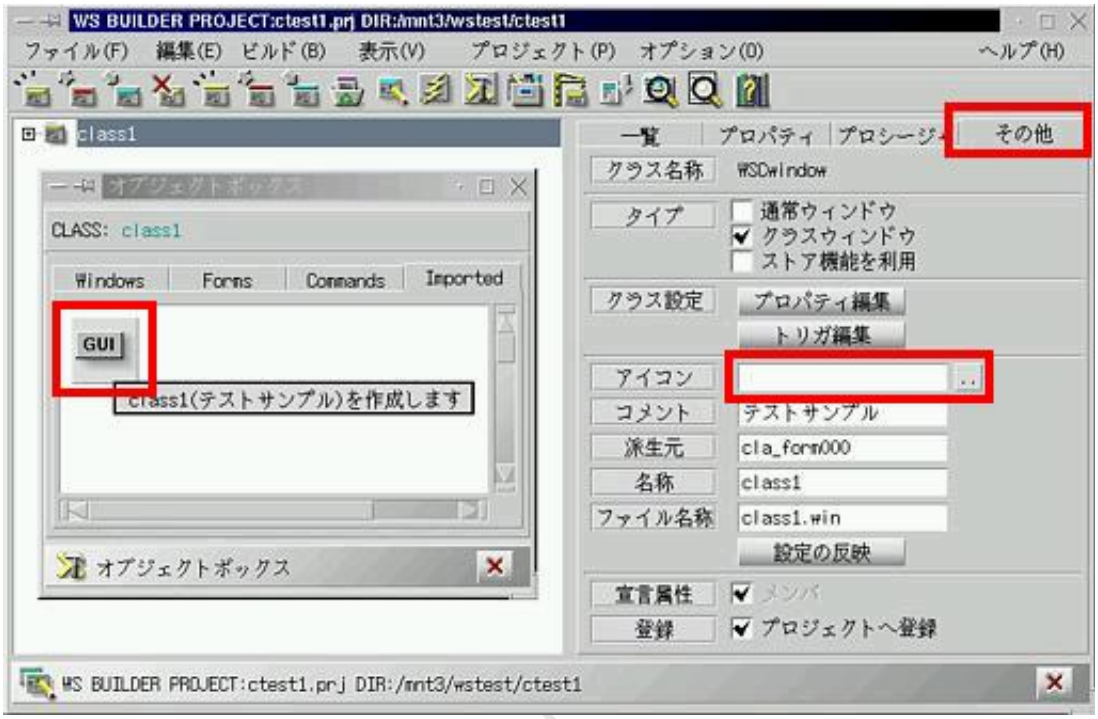


［クラスウィンドウの作成］

2.2 部品アイコン・部品タイトルを指定するには

クラスアプリケーションウィンドウは追加部品ライブラリとしてまとめられて、新たな部品として使用されます。追加ライブラリを読み込むとオブジェクトボックスに新しくオブジェクトが追加され、使用できるようになります。そのときのアイコンを指定することができます。何も指定しない場合は、デフォルトのアイコンが使用されます。

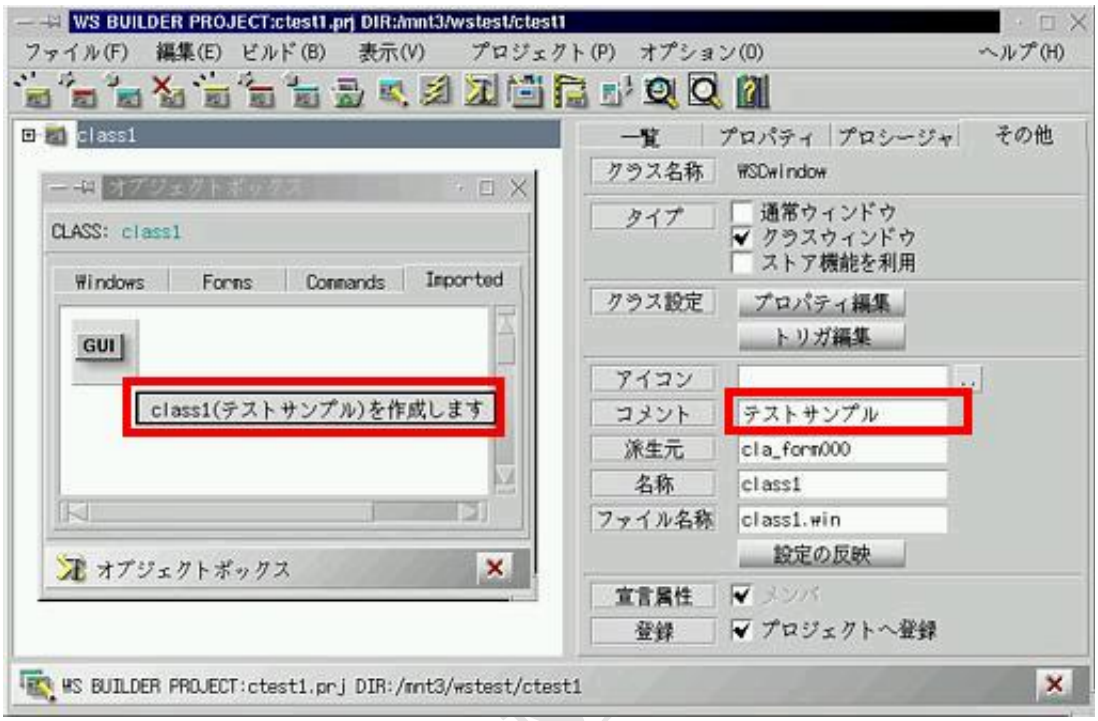
アイコンには、xpm 形式か、jpg 形式を指定してください。次の図は、デフォルトのアイコン（オブジェクトボックス内左)、アイコンを指定するところ（中央）を示しています。



［デフォルトのアイコンとアイコンの指定］

また、オブジェクトボックス上に表示されるバルーンヘルプ (マウスポインタ右下に表示されるコメント) の部品タイトル文字列を指定することができます。

Wide Studio



［オブジェクトボックスのバルーンヘルプとその文字列の指定］

Wide Studio

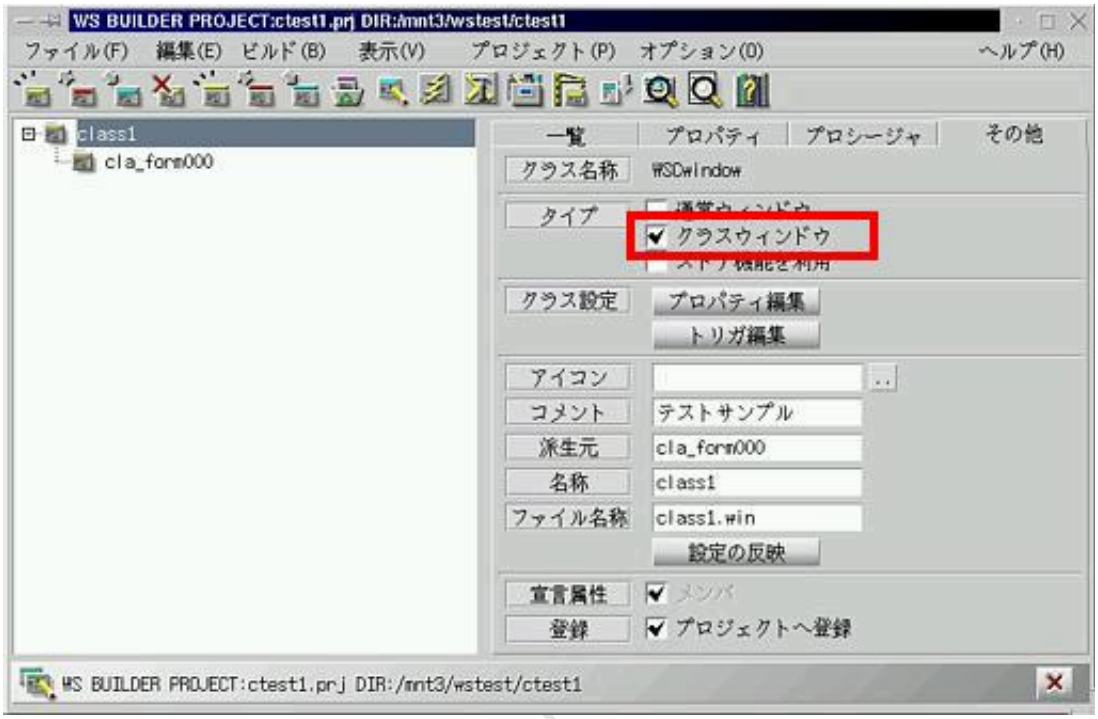
3　新たなクラスを派生するには

3.1　アプリケーションウィンドウをそのままクラスにするには

通常のアプリケーションウィンドウをクラスウィンドウ指定を行うと、そのアプリケーションウィンドウはそのままクラスウィンドウになります。クラスウィンドウをコンパイルを行うと、次のような C++ ソースプログラムが生成されます。[classwin] はアプリケーションウィンドウ名称です。

- ［classwin］.h
  クラスアプリケーションウィンドウ本体のヘッダーファイルです。新たなメンバ変数や関数を追加する場合は、このヘッダに追加します。
- ［classwin］.cpp
  クラスアプリケーションウィンドウ本体のソースファイルです。処理を追加する場合は、このソースプログラムに記述します。
- ［classwin］P.h
  プロパティなどの情報を定義します。ユーザはこのヘッダを編集してはいけません。
- ［classwin］P.cpp
  プロパティなどの資源を初期化したりします。ユーザはこのソースプログラムを編集してはいけません。

Wide Studio Manual Page



［クラスアプリケーションウィンドウの指定］

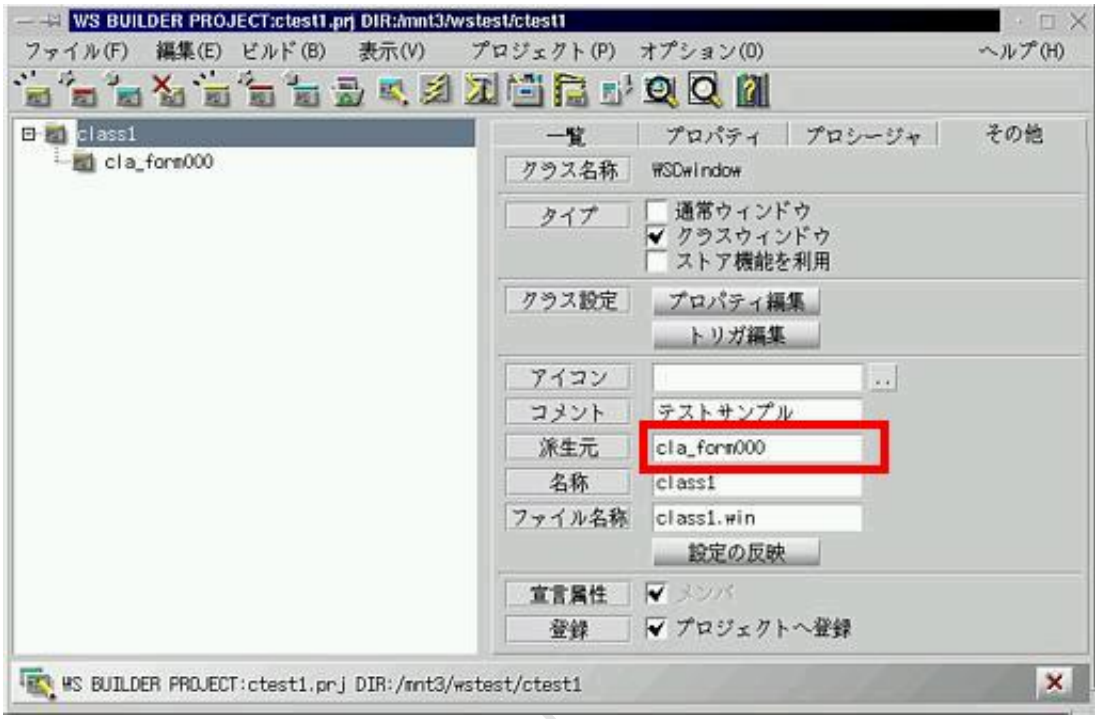
前節でクラスアプリケーションウィンドウ指定した図中の下線のアプリケーションウィンドウ class1 は、そのまま class1 型のオブジェクトクラスになります。クラス class1 は、WSCwindow を継承したクラスとなります。

3.2 アプリケーションウィンドウの一部をクラスにするには

アプリケーションウィンドウの一部、ここでは、フォーム cla_form000 をベースとしたクラスを作成しようとしています。通常、何も指定しないでクラスを派生させると、一番上位のウィンドウをベースとして、派生しますが、図に示す派生元に、トップレベルのウィンドウではない、子オブジェクトを指定すると指定すると、それをベースとして派生します。

（注意）派生元を変更した場合は、ソースプログラムを生成しなおす必要があります。次に示すファイルがすでに存在する場合は削除してください。［classwin］はアプリケーションウィンドウ名称です。

- ［classwin］.h
- ［classwin］.cpp



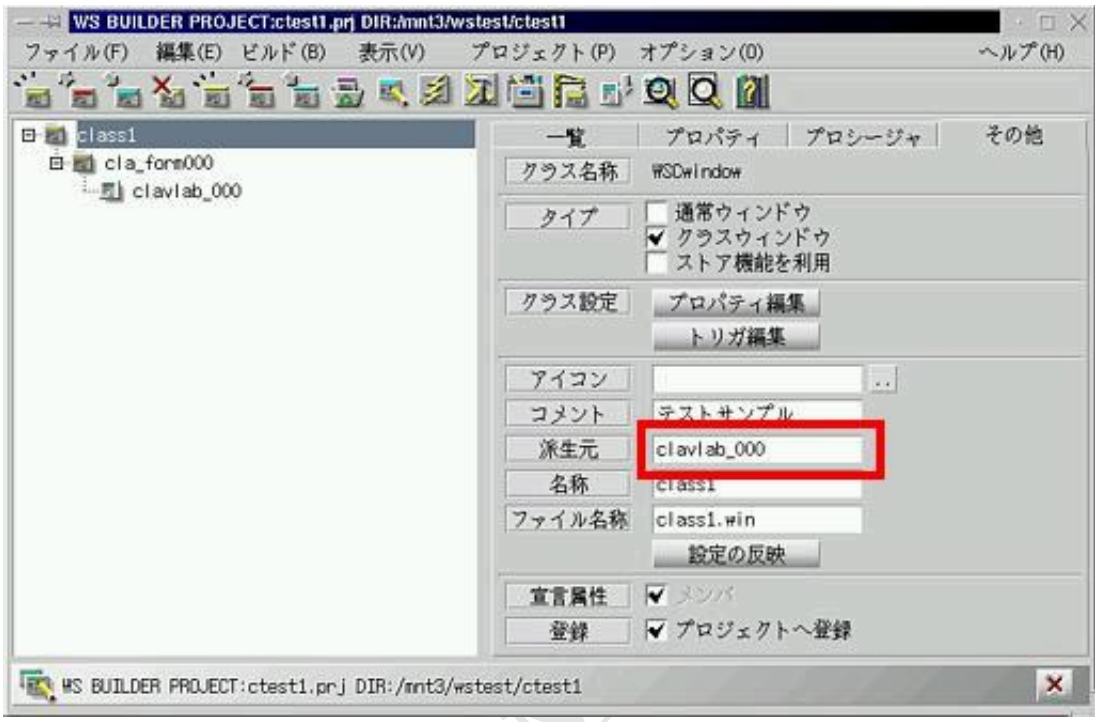
［ベースとなるオブジェクトの指定］

この場合 cla_form000（WSCform クラス）をベースとした、新しいクラス (型) class1 が作成されます。 すなわち、class1 は、WSCform を継承したクラスとなります。もし、cla_form000 上に、子オブジェクトが配置されている場合は、一緒にクラスに含まれます。

3.3 既存のオブジェクトを基に新しいクラスを派生するには

既存のオブジェクト、例えばラベル (WSCvlabel) を基にあらたなラベルとして部品にしたい場合、ベースとなるオブジェクトにそのラベルを指定します。
ここでは、ラベル clavlab_000 をベースとしたクラスを作成しようとしています。図の派生元に clavlab_000 を指定します。
（注意）派生元を変更した場合は、ソースプログラムを生成しなおす必要があります。次に示すファイルがすでに存在する場合は削除してください。［classwin］はクラスウィンドウ名称です。

- ［classwin］.h
- ［classwin］.cpp



［ベースとなるオブジェクトの指定 (2)］

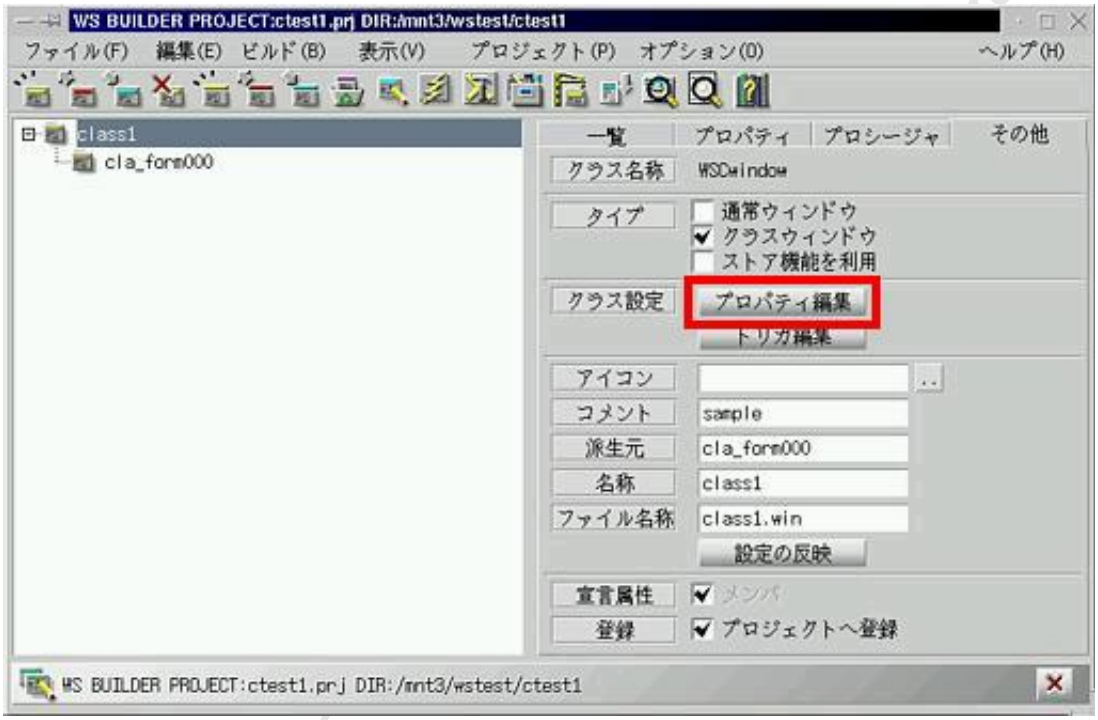
この場合 clavlab_000 をベースとした、新しいクラス (型) class1 が作成されます。 clavlab_000 が WSCvlabel 型のオブジェクトである場合、class1 は、WSCvlabel を継承したクラスとなります。通常はプロパティやメンバ関数 (ＡＰＩ) などを新たに追加してオブジェクトの機能アップするのに利用します。基本的には、アプリケーションウィンドウの一部をクラスにする方法と何ら変わりはありませんが、子オブジェクトを含まないで、単体で派生します。

Wide Studio

4 新たなプロパティを追加・編集・削除するには

4.1 プロパティ設定ウィンドウを表示するには

クラスウィンドウには、プロパティを新たに追加することができます。プロパティの編集は、プロパティ定義ウィンドウで行います。図のプロパティ編集ボタンをクリックしてプロパティ定義ウィンドウを表示させます。



［プロパティ定義ウィンドウの表示］

4.2 新たなプロパティを追加するには

新たなプロパティを追加するには、プロパティ定義ウィンドウの図に示すアイコンをクリックして、プロパティ作成ウィンドウを表示します。

［プロパティ作成ウィンドウ］

指定する項目は次のようなものがあります。

- プロパティ名称
  WSN で始まる英数で名称を付けます。名称が他のプロパティ名称と重複してはいけません。
- プロパティ型
  プロパティのデータ型を指定します。プロパティに割り当てられる変数の型になります。
- メンバ変数
  プロパティに割り当てられる変数を指定します。変数が他のプロパティのものと重複してはいけません。
- デフォルト値
  プロパティに設定される初期値です。文字列型の場合は、入力されたスペース文字全てが、デフォルトにセットされる文字列として扱われますので、入力には注意して下さい。
- 定義属性
  プロパティの定義属性には次のようなものが存在します。

分類	説明
通常定義	通常のプロパティを生成します。
不可視定義	不可視化されたプロパティを生成します。
削除	派生基のクラスに存在するプロパティを削除します。
初期値変更	派生基のクラスに存在するプロパティの初期値を変更します。
不可視化変更	派生基のクラスに存在するプロパティを不可視化します。

Wide Studio Manual

- ビルダタイトル
  アプリケーションビルダ上の編集時に表示される図の下線部の説明文字列を指定します。

［プロパティの作成］

4.3 プロパティを編集するには

プロパティを編集するには、プロパティ定義ウィンドウの下線部のように編集したいプロパティをマウスで選択状態にして、アイコンをクリックし編集します。

［プロパティの編集］

Wide Studio Manual Page

4.4 プロパティを削除するには

プロパティを削除するには、プロパティ定義ウィンドウの下線部のように削除したいプロパティをマウスで選択状態にして、アイコンをクリックし削除します。

［プロパティの削除］

4.5 不可視プロパティを作成するには

クラスアプリケーションウィンドウでは、アプリケーションビルダ上からは見えないプロパティを作成することができます。プロパティ定義ウィンドウで不可視定義を指定します。

Wide Studio Manual

［不可視プロパティの定義］

Wide Studio Manual Page

5　派生元のクラスに存在するプロパティを削除/不可視化するには

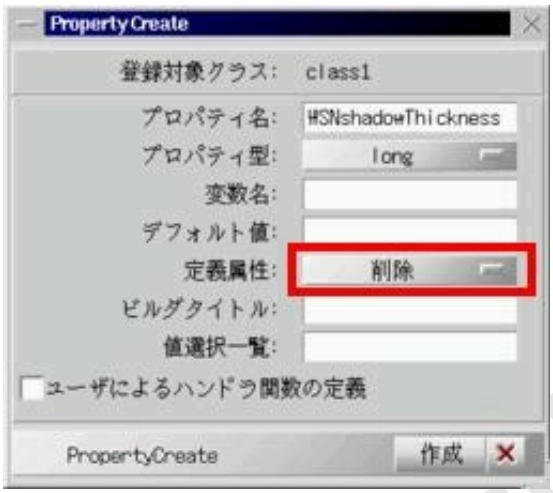
5.1　派生元のクラスに存在するプロパティを削除するには

クラスアプリケーションウィンドウでは、派生基のクラスに存在するプロパティを削除することができます。派生基プロパティを削除する削除属性プロパティを作成します。この削除属性プロパティは、同じ名前の派生基のプロパティを削除します。削除属性のプロパティを作成するには、まず新しいプロパティを作成する要領でプロパティ作成ウィンドウを表示させます。

［プロパティ作成ウィンドウの表示］

次に、図の下線部のように削除したいプロパティ名称を入力し、定義属性を削除にして、作成ボタンをクリックします。図の例では、WSNshadowThickness プロパティを削除するよう設定しています。

Wide Studio Manual

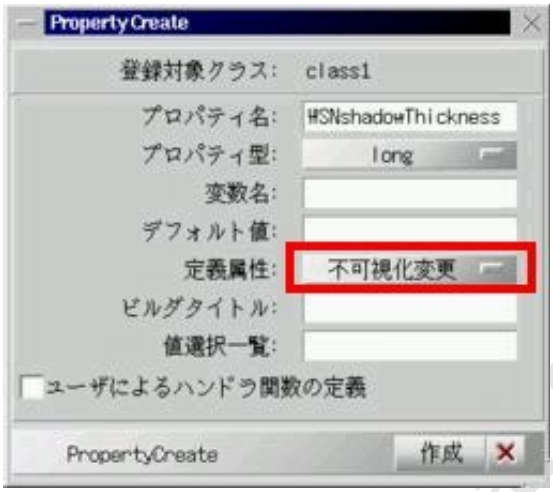


［削除プロパティの作成］

5.2 派生基のクラスに存在するプロパティを不可視属性にするには

クラスアプリケーションウィンドウでは、派生基のクラスに存在するプロパティを不可視化することができます。派生基プロパティを不可視化する不可視化変更属性プロパティを作成します。この不可視化変更属性プロパティは、同じ名前の派生基のプロパティを不可視化します。不可視化変更属性のプロパティを作成するには、削除属性のプロパティと同じ要領で作成します。
次に、図の下線部のように不可視化したいプロパティ名称と型を入力し、定義属性を不可視化変更にして、作成ボタンをクリックします。図の例では、WSNshadowThickness プロパティを不可視化するよう設定しています。

Wide Studio Manual Page

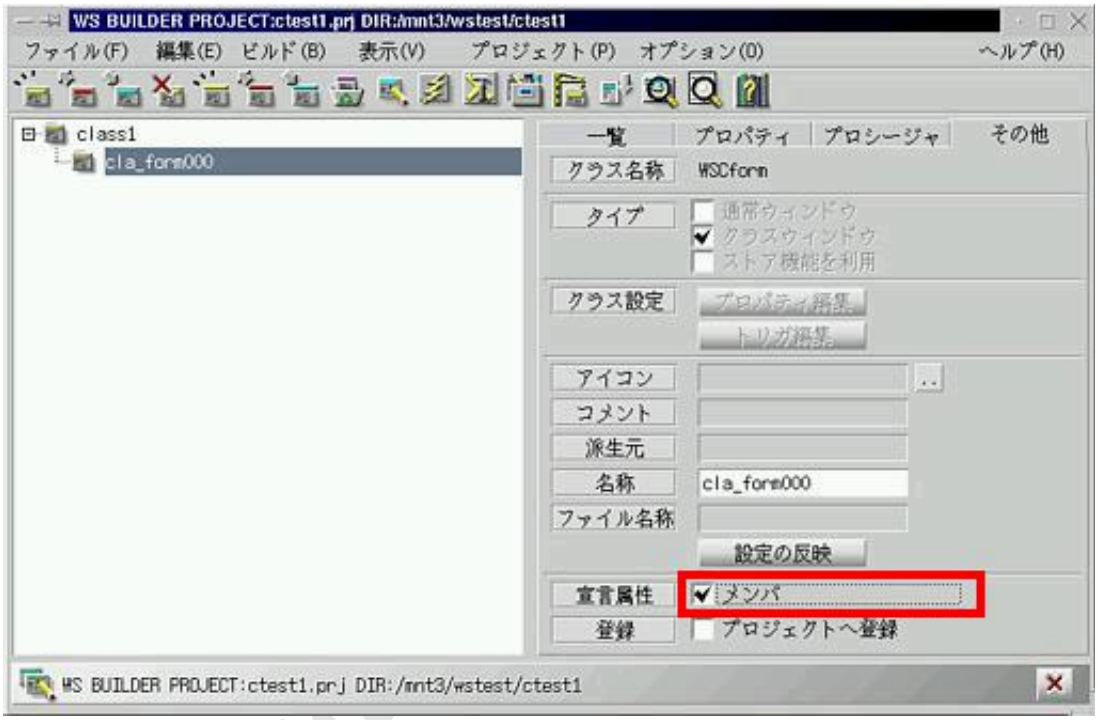


［不可視化プロパティの作成］

Wide Studio Manual Page

6　配置されたオブジェクトをメンバ変数にするには

クラスアプリケーションウィンドウ上に配置されたオブジェクトは、メンバ変数として定義することができます。まずオブジェクトを選択し、インスペクタの基本設定を表示し、宣言属性の［メンバ］の項目をチェックします。



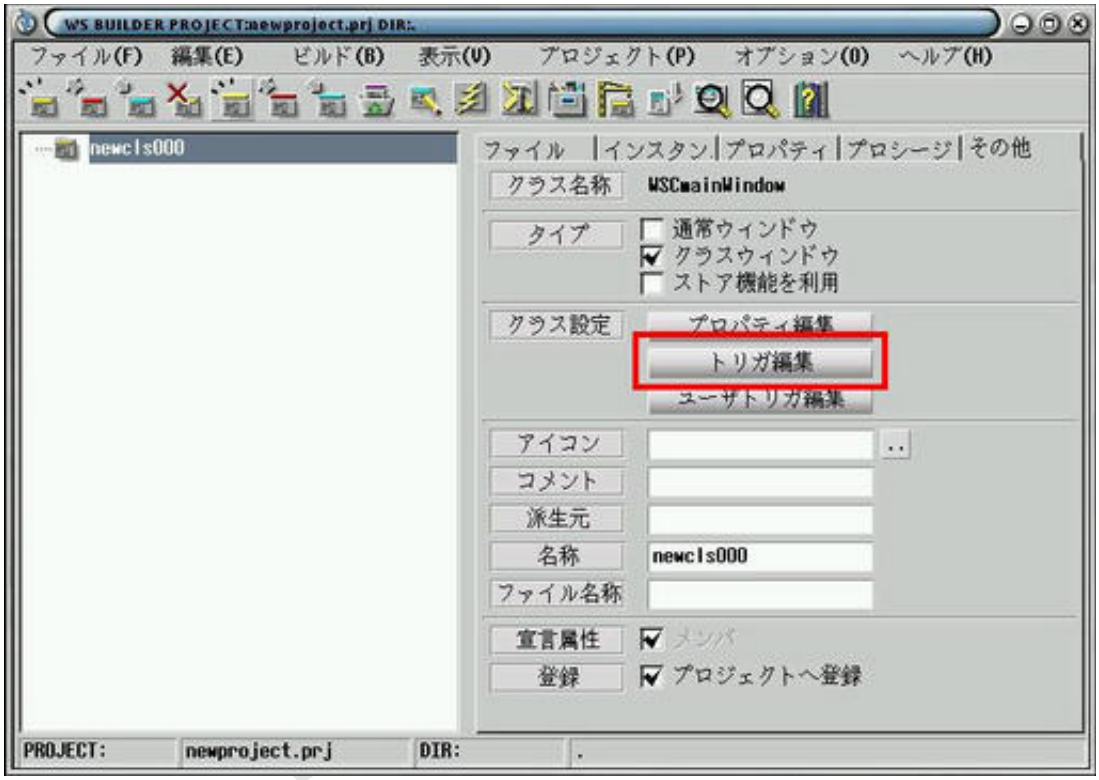
［メンバ属性の指定］

Wide Studio

7　新たなトリガを追加/編集/削除するには

7.1　トリガ設定ウィンドウを表示するには

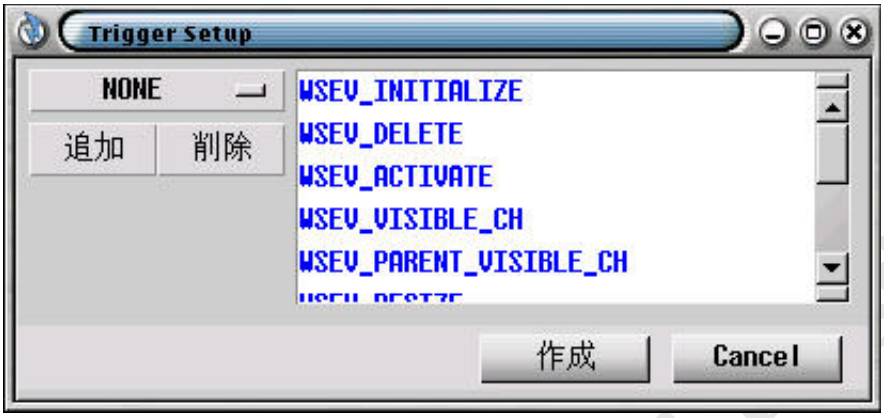
クラスアプリケーションウィンドウには、トリガを新たに追加することができます。トリガの編集は、トリガ設定ウィンドウで行います。図のトリガ編集ボタンをクリックして表示します。



［トリガ設定ウィンドウの表示］

7.2　新たなトリガを追加するには

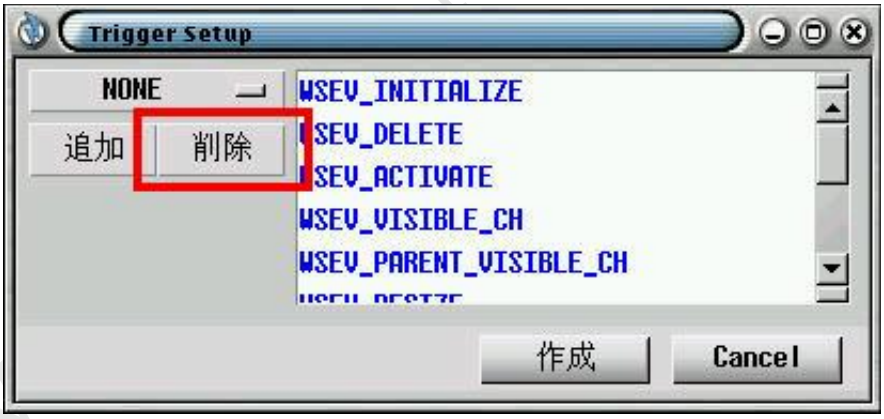
新たなトリガを追加するには、図中のメニューでトリガを選択し、追加ボタンをクリックします。追加作業が終わったら設定ボタンをクリックして終了します。



［トリガの追加］

7.3 追加したトリガを削除するには

一度追加したトリガを削除するには、図の下線部のように削除したいものをマウスで選択状態にして、削除ボタンをクリックし削除します。作業が終わったら設定ボタンをクリックして終了します。

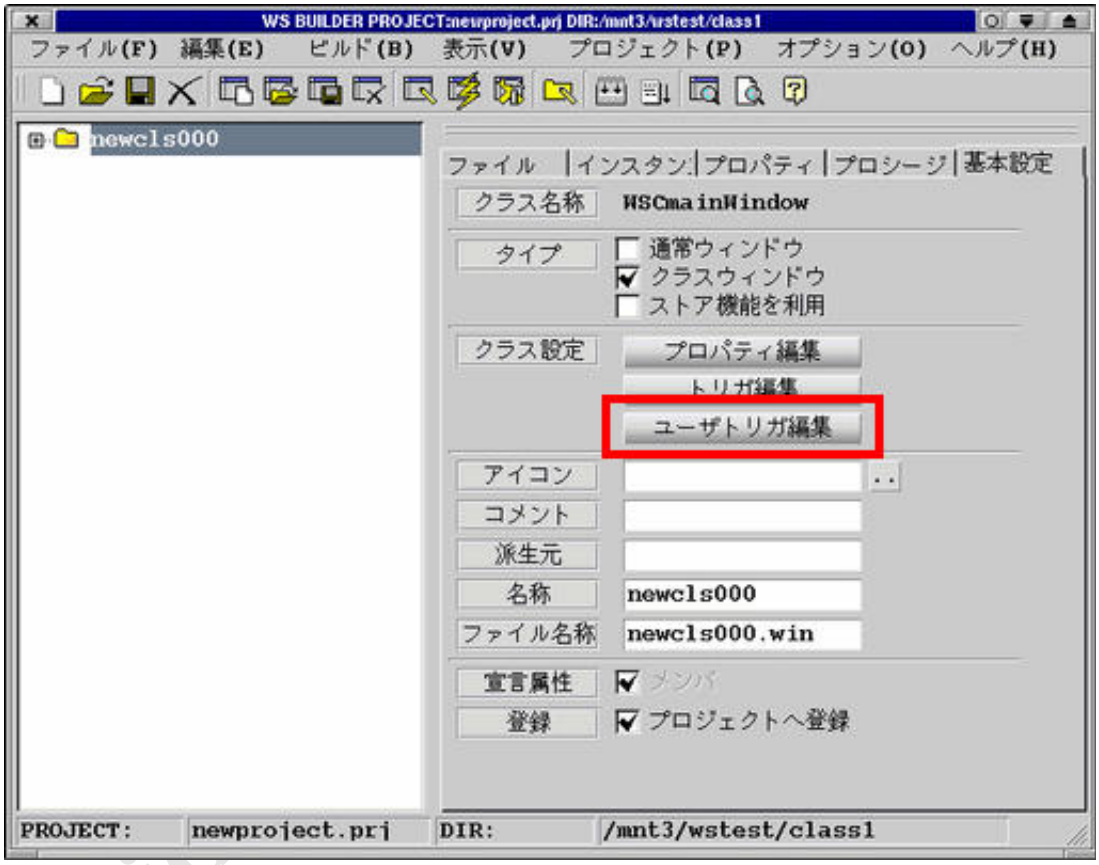


［トリガの削除］

8　新たなユーザトリガを追加/編集/削除するには

8.1　トリガ設定ウィンドウを表示するには

クラスアプリケーションウィンドウには、トリガを新たに追加することができます。トリガの編集は、トリガ設定ウィンドウで行います。図のトリガ編集ボタンをクリックして表示します。



［ユーザトリガ設定ウィンドウの表示］

8.2　新たなユーザトリガを追加するには

新たなユーザトリガを追加するには、図中のアイコンを選択し、追加したいトリガ名、定義値を入力し作成ボタンをクリックします。

［ユーザトリガの追加］

8.3 追加したユーザトリガを削除するには

一度追加したユーザトリガを削除するには、図のように削除したいものをマウスで選択状態にして、削除アイコンをクリックし削除します。

Wide Studio Manual

［ユーザトリガの削除］

Wide Studio Manual

9　クラスライブラリを作成するには

アプリケーションビルダには、ユーザが作成した新たなオブジェクトをまとめて、クラスライブラリを作成する機能があります。プロジェクトは通常、メイク後に実行可能なロードモジュールを作成しますが、ライブラリ作成オプションを設定すると、クラスライブラリを作成するようになります。クラスライブラリにしたいクラスウィンドウ (部品) をプロジェクトに登録し、コンパイルすることで、好みのクラスライブラリを作成することができます。

［クラスライブラリ作成の指定］

図のようにプロジェクト設定ウィンドウ中の［コンパイル］で［クラスライブラリ］の項目をチェックして、メイクすると次のような名称でライブラリファイルが作成されます。［output］は、リンクで指定した、出力ファイルの名称です。

システム	ライブラリ
ＵＮＩＸ	lib[output].so (シェアード)
	lib[output].a (スタティック)
Ｗｉｎｄｏｗｓ－ＮＴ	lib[output].dll

これらのクラスライブラリを使用する場合は、オブジェクト編の［追加ライブラリによる新たなオブジェクトを利用するには］の節を参照下さい。

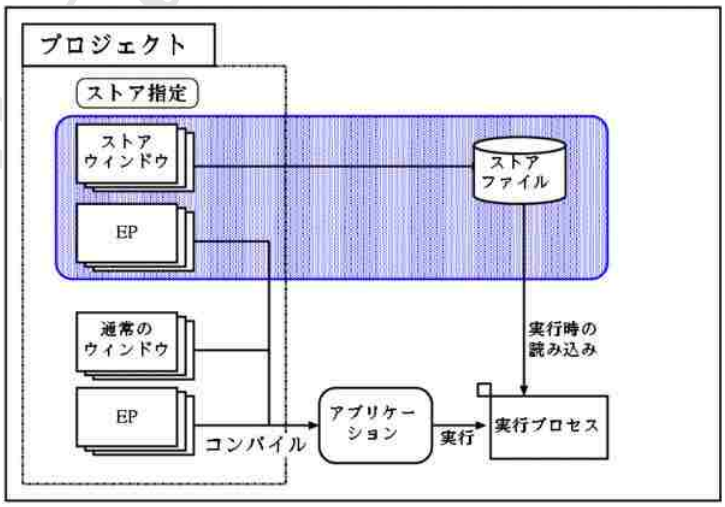
I　オンラインストア編

1　オンラインストア機能とは

WideStudio には、アプリケーション実行中に、動的にオブジェクトを読み込む機能があります。オンラインストア機能を利用すると、次のようなことが可能となります。

- メモリの節約や立ち上げの高速化
  あらかじめ、立ち上げ時に全てのアプリケーションウィンドウを展開せず、オンライン実行中にアプリケーションウィンドウをロードすることで、不必要なメモリの消費や、立ち上げの時間の短縮化が図れます。
- アプリケーションウィンドウの一部分の入れ換え表示
  ストアされた部分アプリケーションウィンドウを他のアプリケーションウィンドウ上にロードすることで、アプリケーションウィンドウの一部分の入れ換え表示が可能となります。図面表示画面など、中央の図面部分のみを入れ替え表示するような使用方法も、可能です。
- アプリケーション実行中におけるオンラインメンテナンスの実現
  オンライン実行中にアプリケーションウィンドウをメンテナンスし、ロードすることでオンラインメンテナンス（コンパイルのいらないメンテナンス）に対応したアプリケーションを構築することが可能となります。

プロジェクトに登録された通常のアプリケーションウィンドウをアプリケーションビルダでストア指定を行うと、自動的にストアファイルに保存され、作成されるロードモジュールから外されます。ストアされたアプリケーションウィンドウはオンライン実行時に、WSGFloadWindow() グローバル関数を用いてロードします。

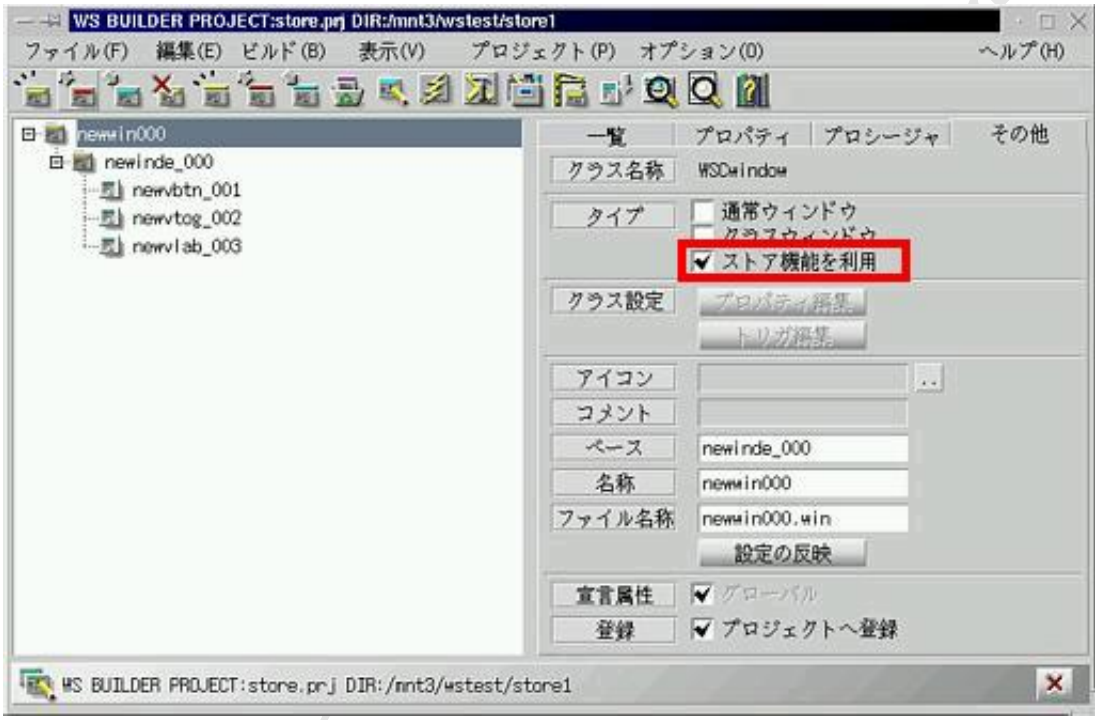


Wide Studio Manual Page

2　アプリケーションウィンドウをオンラインストア形式にするには

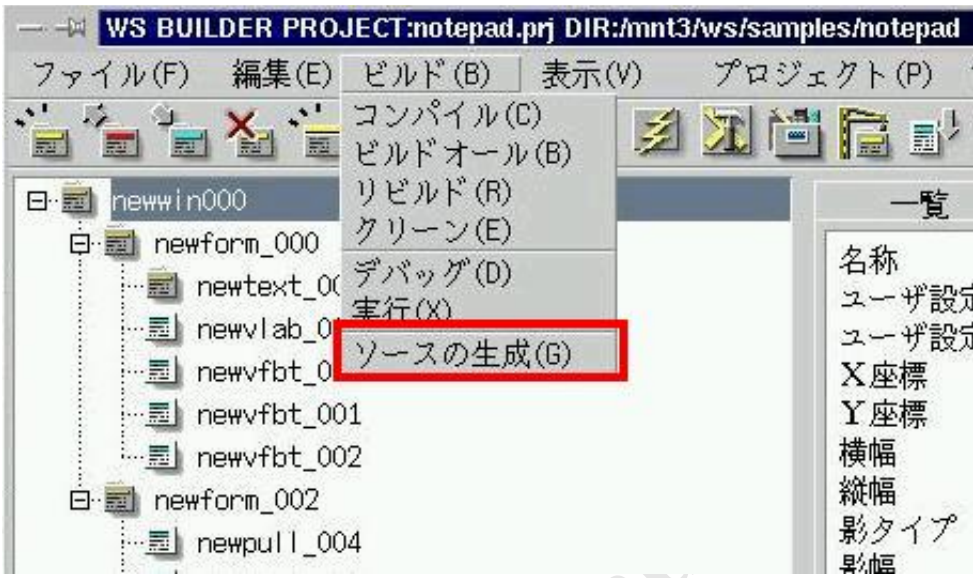
2.1　アプリケーションウィンドウをオンラインストア形式で出力するには

通常のアプリケーションウィンドウをオンラインストア形式にすることができます。インスペクタの基本設定において図中の下線のように保存したいアプリケーションウィンドウを選択した状態で、［ストア機能を利用］をチェックします。



［ストア形式の指定］

次の図のように、［ビルド］の［ソースの生成］を選択します。



［オンラインストア形式の出力］

2.2 プロジェクトでオンラインストア形式に指定するには

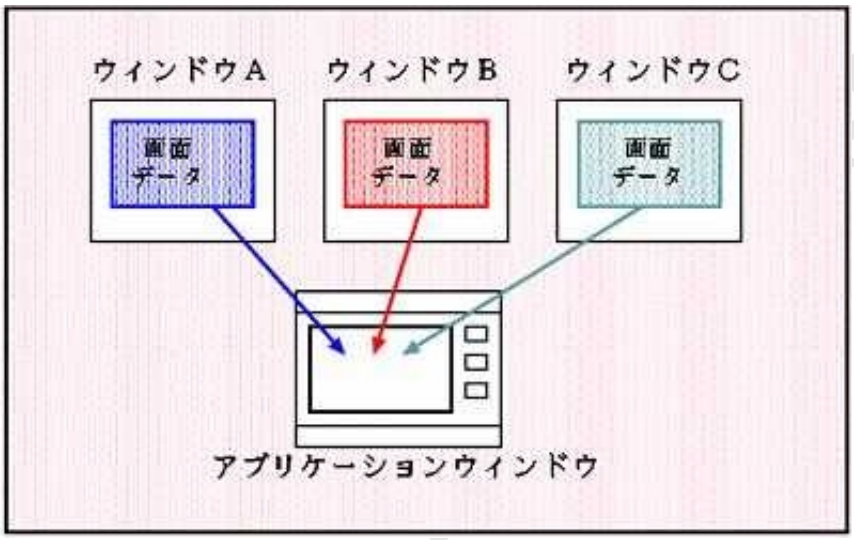
アプリケーションウィンドウの変更の都度、手動でオンラインストア形式で保存する労力は、できれば避けたいものです。アプリケーションビルダでは、前項の指定により、オンラインストア形式での取扱の指定をアプリケーションウィンドウ毎に設定することができます。あるアプリケーションウィンドウをストア扱いに指定した場合、そのアプリケーションウィンドウはコンパイルされずに、オンラインストア形式に保存されます。作成される実行可能モジュール内には、もはやそのアプリケーションウィンドウは存在せず、オンライン実行中にそのストアファイルを読み込むことで、初めてそのアプリケーションウィンドウは表示可能となります。
生成されるファイル名は、［ウィンドウ名称］.oof です。

なお、ストアされたアプリケーションウィンドウをオンラインでロードする場合は、プログラミング編の［ストアされたアプリケーションウィンドウをロードするには］の節を参照下さい。

Wide Studio Manual

3　アプリケーションウィンドウの一部分をオンラインストアするには

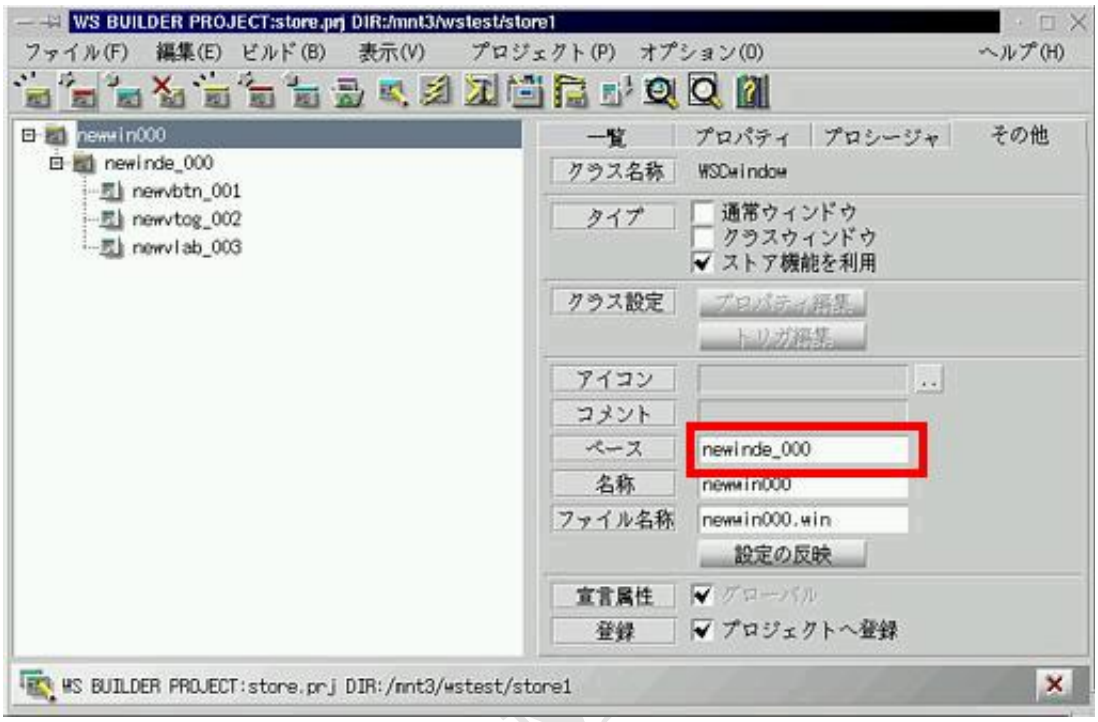
アプリケーションウィンドウの一部分をオンラインストア形式で保存することができます。アプリケーションウィンドウの一部分をストアする最大のメリットは、ストアされた異なるエリアをすり替え表示することが可能となることです。



［すり替え表示］

上の図の例では、アプリケーションウィンドウ A、B、C のそれぞれにおいて表示したい内容、すなわち網かけのエリアの部分を基底としてストア形式にしておきます。それぞれのストアされた部分をオンラインで、ロードまたは破棄して、メインアプリケーションウィンドウ (全く別のアプリケーションウィンドウ) 上に入れ換え表示させます。
アプリケーションウィンドウの一部分をストアしたい場合、インスペクタの基本設定のベースに項目にストアしたいオブジェクトの名称を指定します。
この項目が空白の場合は、アプリケーションウィンドウがベースとしてストアされます。

Wide Studio - Close



［ストアするオブジェクトの指定］

なお、ストアされた部分アプリケーションウィンドウをオンラインでロードする場合は、プログラミング編の［ストアされた部分アプリケーションウィンドウをロードするには］の節を参照下さい。

Wide Studio

J　リモートインスタンス編

1　リモートインスタンス機能とは

WideStudio には、他のマシンで動作している WideStudio アプリケーション中に存在するインスタンス (オブジェクト) を、あたかも自分のプロセスに存在するインスタンス (オブジェクト) と同じように呼び出すことができます。リモートインスタンス機能を利用すると、次のようなことが可能となります。

- 分散コンピューティングが可能
  他の WideStudio アプリケーションに存在するインスタンス (オブジェクト) に、あたかも自分のプロセスに存在するインスタンス (オブジェクト) と同じようにアクセスすることで簡単に、分散コンピューティングが可能となります。大きな WideStudio アプリケーションをあらかじめ、機能別に複数のプロセスに分離してアプリケーションを構築し、複数のプロセスとして運用することができます。

- ネットワーク上のシームレスな分散
  ネットワーク上に存在する複数台のマシンに分散された WideStudio アプリケーションに存在するインスタンスをどこに存在するかを気にすること無くシームレスに参照することができます。各マシン上で動作するエージェントがマシン上に存在するリモートインスタンスを管理し、エージェント同士が、その情報を交換するしあうことで、自動的に、どこのマシン上にどんなリモートインスタンスが存在するか管理されています。

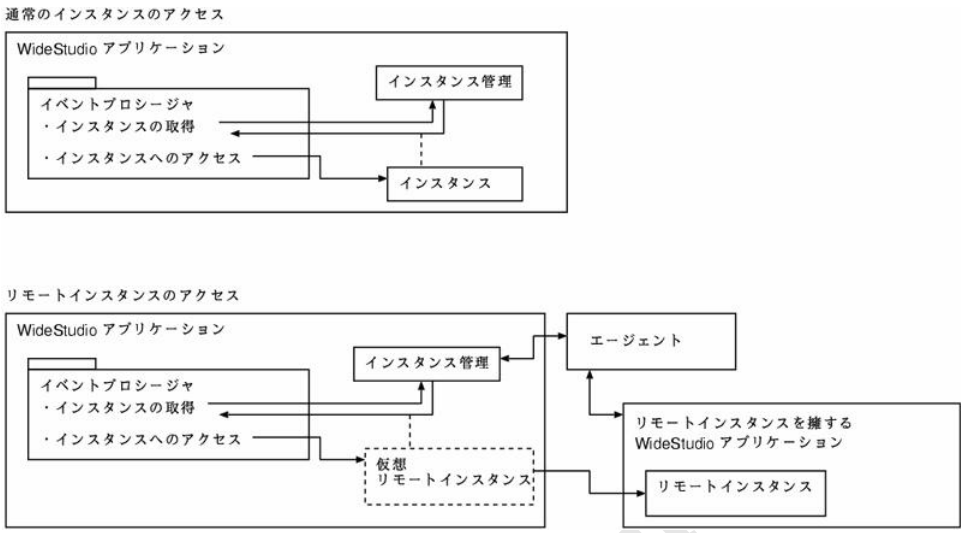
  WideStudio アプリケーションは、そんなリモートインスタンスの存在位置を気にすること無く、アクセスするだけで自動的に該当するリモートインスタンスにアクセスできます。

- フォールトトレランスの強化
  WideStudio アプリケーションを多重化して動作させることができます。例えば、多重化されたアプリケーションの内の１つがダウンしても、エージェントがそれを察知して、別のアプリケーションに存在するリモートインスタンスに切り替えます。

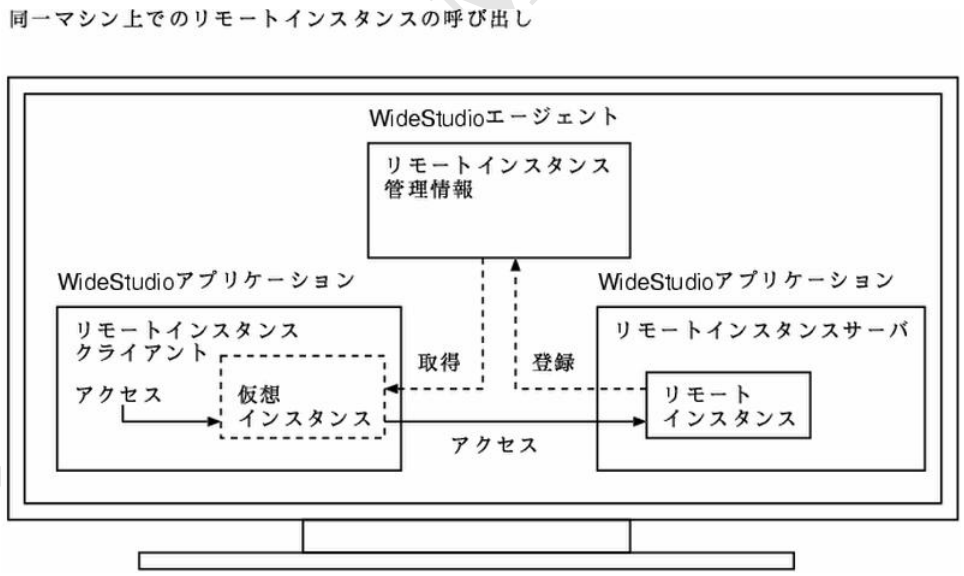
  WideStudio アプリケーションは、相手が多重化されていてアクセス切り替えられたことを意識することなくリモートインスタンスにアクセスすることができます。

リモートインスタンスへのアクセスは、WSCGIappObjectList() オブジェクト管理を通して取得された通常のインスタンスへのアクセスと同じように、オブジェクト管理を通して取得されたリモートインスタンスを使用して行います。

Wide Studio Manual Page

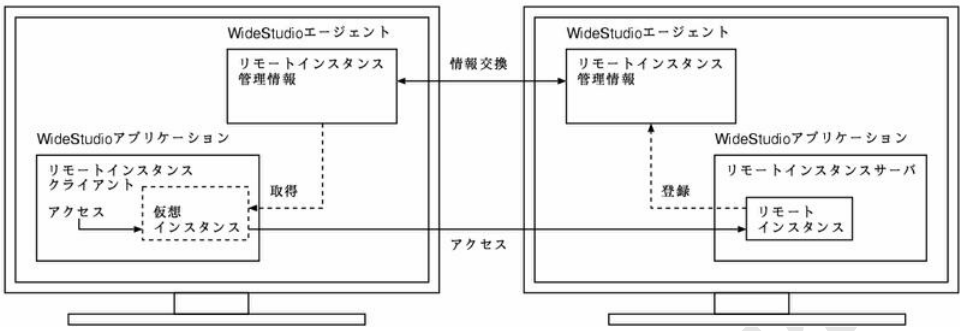


同じマシン上のリモートインスタンスの呼び出しの場合、同じマシン上に存在するエージェントを通して、リモートインスタンスが取得され、リモートインスタンスの呼び出しを行います。



異なるマシン上のリモートインスタンスの呼び出しの場合、それぞれのマシン上に存在するエージェント同士がリモートインスタンス情報の交換を行い、リモートインスタンスが取得され、リモートインスタンスの呼び出しを行います。

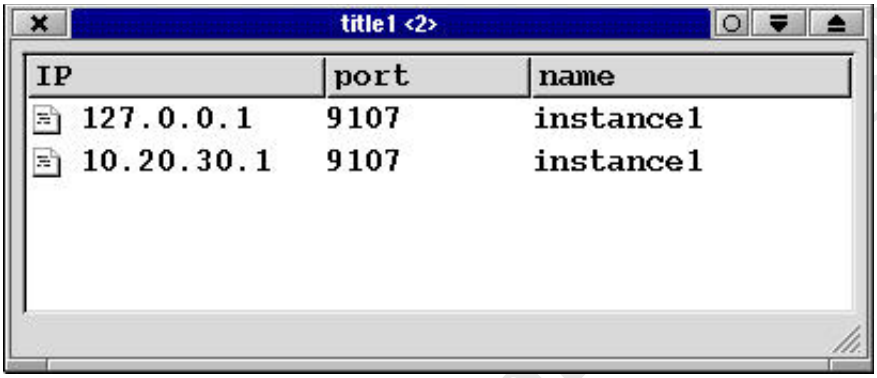
異なるマシン上でのリモートインスタンスの呼び出し



Wide Studio Manual Pr

2　エージェントを起動するには

リモートインスタンス機能を利用する上で、エージェントは重要な位置を占めます。エージェントは、公開されたリモートインスタンスを管理し、リモートインスタンスにアクセスを行いたいクライアントに対してリモートインスタンスの存在位置の情報を提供します。リモートインスタンス機能を利用する場合、１つのマシン上に１つのエージェント wsagent を起動しておく必要があります。



［エージェント］

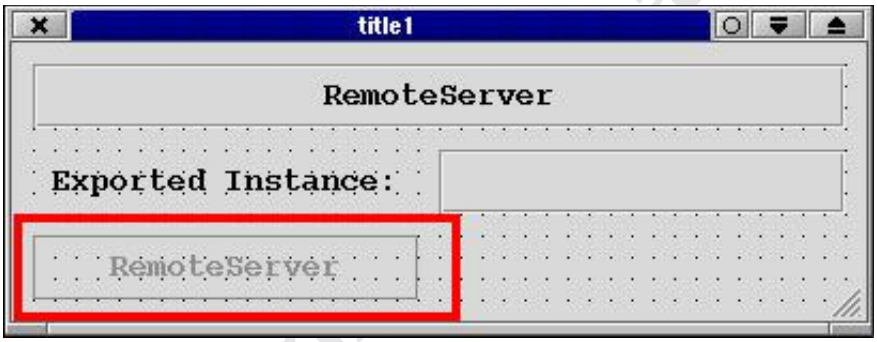
Wide Studio Manual

3　リモートインスタンスサーバを構築するには

リモートインスタンスサーバとは、通常の WideStudio アプリケーションとなんら変わりは無く、アプリケーション内に存在する通常のインスタンスをリモート公開を行ったものです。リモート公開すると、公開されたインスタンスは、自動的にエージェントに登録され、外部のアプリケーションからアクセスできるようになります。

3.1　アプリケーションをリモートインスタンスサーバにするには

WideStudio アプリケーションをリモートインスタンスサーバにするには、WSCvremoteServer クラスを使用します。オブジェクトボックスの NonGUI セクションに存在する WSCvremoteServer クラスのインスタンスをいずれかのアプリケーションウィンドウ上に一つ配置するだけで簡単にリモートインスタンスサーバとすることができます。

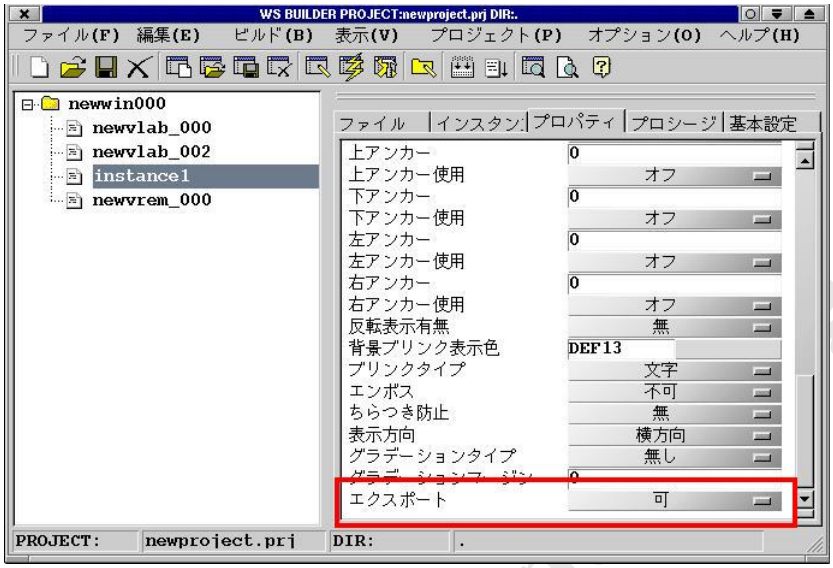


［WSCvremoteServer の配置］

3.2　インスタンスをリモートインスタンスとして公開するには

WideStudio アプリケーション上に存在するインスタンスを外部公開し、リモートインスタンスとしたい場合、そのインスタンスのプロパティ「エクスポート」を可に設定します。

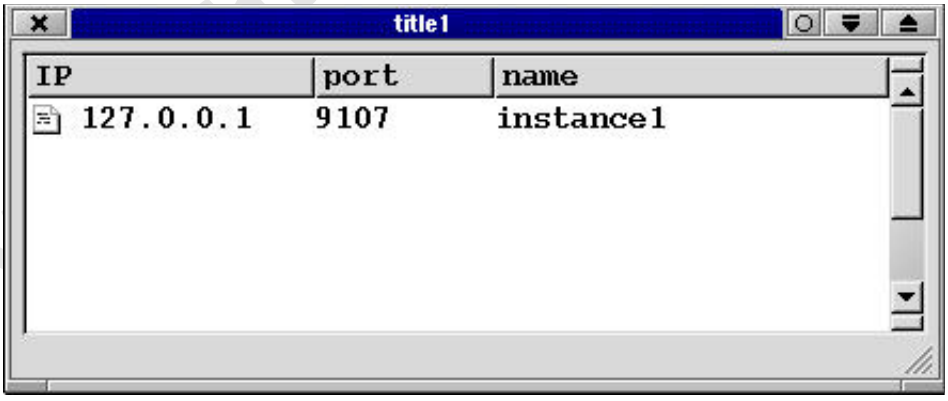
WideStudio Manual Page



［インスタンスをリモートインスタンスとして公開するところ］

3.3 エージェントをするには

リモートインスタンスサーバに存在する公開されているリモートインスタンスを管理するためのエージェントを起動することで、リモートインスタンスにアクセスすることができるようになります。エージェントは wsagent コマンドで起動します。



［エージェント wsagent を起動したところ］

4 エージェントとは

WideStudio アプリケーションは、自分自身のプロセスに存在するアプリケーションインスタンスと同じようにリモートインスタンスサーバに存在するリモートインスタンスにアクセスすることが出来ます。

まず、リモートインスタンスにアクセスするために、そのリモートインスタンスがどのマシン上のリモートインスタンスサーバに存在するかをエージェントに問い合わせる必要があります。

エージェントは、リモートインスタンスサーバに存在するリモートインスタンスを管理します。

5 リモートインスタンスにアクセスする前に

リモートインスタンスにアクセスするために、WSCvremoteClient クラスを使用します。WSCvremoteClient クラスは、リモートインスタンスを管理するエージェントとの通信をサポートします。

エージェントと通信を行うために行わなければならないことは、オブジェクトボックスの NonGUI セクションに存在する WSCvremoteClient をアプリケーションのいずれかのアプリケーションウィンドウ上に配置するのみです。

6 リモートインスタンスにアクセスするには

名称を指定して、オブジェクト管理を通して得られた仮想リモートインスタンスを通してリモートインスタンスにアクセスします。詳細はプログラミングガイドのリモートインスタンス編を参照ください。

Wide Studio Manual Page