CONTENTS

コマンド

Concatenate [/DL /FREE /KILL /NP[=*dim*] /O] [*typeFlags*] *waveListStr*, *destWave*

Concatenate [/DL /FREE /KILL /NP[=*dim*] /O] [*typeFlags*] {*wave1*, *wave2*, *wave3*, ...}, *destWave*

Concatenate [/DL /FREE /KILL /NP[=*dim*] /O] [*typeFlags*] {*waveWave*}, *destWave*

Concatenate コマンドは、ソースウェーブのデータを *destWave* に結合します。
destWave は、まだ存在しない場合は作成されます。
destWave がすでに存在し、上書きが指定されていない場合、ソースウェーブのデータは、宛先ウェーブ内の既存のデータと連結されます。

デフォルトでは、連結処理により、可能であれば宛先ウェーブの次元数が増加します。
例えば、同じ長さの 1D ウェーブを２つ連結すると、２列からなる 2D ウェーブが得られます。
この場合、宛先ウェーブはより高い次元へと「昇格」したと言われます。

/NP（プロモーションなし）フラグを使うと、宛先ウェーブの次元数は変更されません。
例えば、/NP を使って同じ長さの２つの 1D ウェーブを連結すると、ソースウェーブの長さの合計に等しい長さの 1D ウェーブが得られます。

ソースウェーブの長さが異なる場合、/NP を使うか否かにかかわらず、昇格は行われません。

パラメーター

waveListStr は、セミコロンで区切られた入力ウェーブの名前の一覧を含み、末尾にセミコロンが付いた文字列式です。
waveListStr に含まれるウェーブの名前数に制限はありません。

{*wave1*, *wave2*, ...} という構文では、100 ウェーブまでと制限されています。

{*waveWave*} 構文においては、*waveWave* は、入力ウェーブへの参照を含む１つのウェーブ参照です。
この構文は Igor Pro 8.0 で追加されました。

destWave は、連結結果が格納される新規または既存のウェーブの名前です。

フラグ

/DL	次元ラベルを設定します。昇格を行う時、新しい次元ラベルにはソースのウェーブ名が使われます。それ以外の場合は、既存のラベルが使われます。
/FREE	*destWave* をフリーウェーブとして作成します（ヘルプ Free Waves を参照）。 /FREE フラグは Igor Pro 8.0 で追加されました。
/KILL	ソースウェーブを Kill します。
/NP	高次元への昇格を妨げます。

/NP=*dim*　　指定された次元（0=行、1=列、2=レイヤー、3=チャンク）に沿ってデータの昇格を防止し、データを追加します。*dim* で指定された次元以外のすべての次元は、すべてのウェーブで同じでなければなりません。

/O　　*destWave* を上書きします。

型フラグ（関数のみ）

Concatenate では、ユーザー関数内でさまざまな型フラグを使って、宛先のウェーブ参照変数の型を指定することもできます。
これらの型フラグは、同名の別のウェーブ参照変数と一致させる必要がある場合や、ウェーブ代入に対してどのような式をコンパイルすべきかを指示する必要がある場合を除き、使う必要はありません。
型フラグの完全な一覧と詳細は、ヘルプ WAVE Reference Types と WAVE Reference Type Flags を参照してください。

詳細

destWave がまだ存在しない場合、または /O フラグが使われた場合、*destWave* は最初のソースウェーブの複製によって作成されます。
ウェーブは、ソースウェーブのリスト順に連結されます。
destWave が存在し、かつ /O フラグが使われていない場合、連結は *destWave* から開始されます。

destWave はソースウェーブリストでは使用できません。

ソースウェーブは、すべて数値か、すべてテキストのいずれかでなければなりません。

昇格が許可されている場合、すべてのウェーブが共通して持つ低い次元の数が *destWave* の次元数を決定し、その結果、*destWave* の次元数は 1 大きくなります。
デフォルトの挙動は、ソースウェーブのサイズによって異なります。
すべて同じ長さの 1D ウェーブを連結すると 2D ウェーブが生成されますが、長さが異なる 1D ウェーブを連結すると 1D ウェーブが生成されます。
同様に、同じサイズの 2D ウェーブを連結すると 3D ウェーブが生成されますが、ソースとなる 2D ウェーブの列数が異なる場合は *destWave* は 2D ウェーブとなり、行数が異なる場合は *destWave* は 1D ウェーブとなります。
行数が同じ 1D ウェーブと 2D ウェーブを連結すると 2D ウェーブが生成されますが、行数が異なる場合は、*destWave* は 1D ウェーブになります。
例を参照してください。

/NP フラグを使うと、次元の昇格が抑制され、*destWave* の次元数が入力ウェーブと同じままになります。

> **警告**
>
> ユーザー定義関数内のループなど、特定の状況下では、Concatenate が予期しない動作を示すことがあります。

コンパイル時に、ユーザー定義関数の中に次のような文がある場合：

```
Concatenate/O ..., DestWaveName
```

Igor Pro は DestWaveName という名前の自動ローカルウェーブ参照変数を生成します。
実行時に、このウェーブ参照変数が NULL の場合、その名前はリテラル名とみなされ、現在のデータフォルダー内にその名前のウェーブが作成されます。

ループ内で Concatenate を最初に呼び出した後のように、ウェーブ参照変数が NULL でない場合、そのウェーブがどこにあるかに関係なく、参照されているウェーブは上書きされます。

現在のデータフォルダー内にウェーブを作成または上書きする場合は、次の 2 つの方法のいずれかを使ってください：

```
Concatenate/O ..., $"DestWaveName"
WAVE DestWaveName                    // 後で宛先ウェーブを参照する場合にのみ必要
```

または

```
Concatenate/O ..., DestWaveName
// その後、DestWaveName の使用が終了したら...
WAVE DestWaveName=$""
```

例

```
// サンプルのウェーブ
Make/N=10 w1,w2,w3
Make/N=11 w4
Make/N=(10,7) m1,m2,m3
Make/N=(10,8) m4
Make/N=(9,8) m5

// 1D ウェーブを連結
Concatenate/O {w1,w2,w3},wdest          // wdest は 10x3 の行列
Concatenate {w1,w2,w3},wdest            // wdest は 10x6 の行列
Concatenate/NP/O {w1,w2,w3},wdest       // wdest は 30 ポイントの 1D ウェーブ
Concatenate/O {w1,w2,w3,w4},wdest       // wdest は 41 ポイントの 1D ウェーブ

// 2D ウェーブを連結
Concatenate/O {m1,m2,m3},wdest          // wdest は 10x7x3 のボリューム
Concatenate/NP/O {m1,m2,m3},wdest       // wdest は 10x21 の行列
Concatenate/O {m1,m2,m3,m4},wdest       // wdest は 10x29 の行列
Concatenate/O {m4,m5},wdest             // wdest は 152 ポイントの 1D ウェーブ
Concatenate/O/NP=0 {m4,m5},wdest        // wdest は 19x8 の行列

// 1D と 2D ウェーブを連結
Concatenate/O {w1,m1},wdest             // wdest は 10x8 の行列
Concatenate/O {w4,m1},wdest             // wdest は 81 ポイントの 1D ウェーブ

// 2D ウェーブに行を追加
Make/O/N=(3,2) m6, m7
Concatenate/NP=0 {m6}, m7               // m7 は 6x2 の行列

// 2D ウェーブに列を追加
Make/O/N=(3,2) m6, m7
Concatenate/NP=1 {m6}, m7               // m7 は 3x4 の行列

// 2D ウェーブにレイヤーを追加
Make/O/N=(3,2) m6, m7
Concatenate/NP=2 {m6}, m7               // m7 は 3x2x2 のボリューム

// 最後のコマンドは、次と同じ効果があります：
// Concatenate {m6}, m7
// どちらのバージョンも、m7 に第３の次元を追加しています
```

参照

Duplicate、Redimension、SplitWave コマンド

ControlBar [/EXP=*e* /L/R/B/T/W=*graphName*] *barSize*

ControlBar コマンドは、グラフ内のコントロールバーのサイズ（高さまたは幅）と位置（上/左/下/右）を設定します。

コントロールがないグラフの場合は、このコマンドを使う必要はありません。

フラグ

/EXP=*e*　　コントロールバー領域の拡大範囲を設定します。

詳細は、NewPanel コマンドの /EXP フラグを参照してください。

/EXP フラグは、Igor Pro 9.0 で追加されました。

/L/R/B/T　　コントロールバーの位置について、グラフの左端、右端、下端、または上端（既定）のいずれかでコントロールバーのサイズを変更するかどうかを指定します。

/W=*graphName*

コントロールバーを含む特定のグラフの名前を指定します。

パラメーター

barSize は、画面の解像度に応じてピクセルまたはポイントで指定されます。
詳細は、ヘルプ Control Panel Resolution on Windows を参照してください。

barSize を 0 に設定すると、指定されたコントロールバーが削除されます。

詳細

コントロールバーとは、通常、グラフの上部にあり、ボタン、チェックボックス、ポップアップメニューなどのコントロール用に確保された領域のことです。
この領域とグラフ領域の間には線が引かれています。
Ctrl キーを押しながらこの領域をクリックするか、右クリックするか、または ModifyGraph コマンドを使うことで、この領域に別の背景色を割り当てることができます。
この領域では、描画ツールを使うことはできません。

/L、/R、/B フラグを指定することで、グラフの左端、右端、下端にコントロールバーを追加することもできます。
これらのコントロールバーには、バーとグラフ領域の間に線がありません。

ControlInfo kwControlBar を使って、上部のコントロールバーの現在の高さを V_height に取得します。
kwControlBarBottom についても同様です。

ControlInfo kwControlBarLeft を使って、左コントロールバーの現在の幅を V_width で取得します。
kwControlBarRight についても同様です。

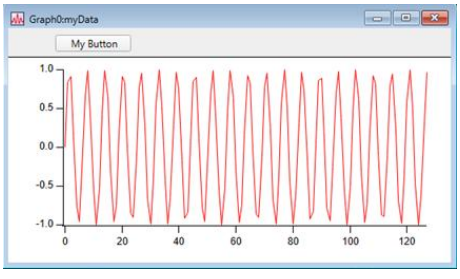
推奨

コントロールバーに配置されたコントロールは、グラフのサイズ変更時に自動的に位置が調整されません。
上部または左側のコントロールバーを使う場合は、多くの場合問題になりませんが、右側または下部のコントロールバーを使う場合は明らかに問題となります。
その場合は、グラフのサイズを固定しておくと便利です。

ControlBar を単独で使うのではなく、グラフ内の任意の場所にパネルサブウィンドウを挿入し、そのパネルサブウィンドウ内にコントロールを配置することを推奨します。
これは、ホストグラフのサイズが変更されても、コントロールがパネルサブウィンドウ内での位置を維持するためです。

例

```
Make/O myData=sin(x)
Display myData
ControlBar 35                                   // 高さ 35 ピクセル
Button button0,pos={56,8},size={90,20},title="My Button"
```



参照

ControlInfo コマンド
ヘルプ Controls in Graphs

ControlInfo [/W=*winName*] *controlName*

ControlInfo コマンドは、グラフ、パネルウィンドウ、またはサブウィンドウ内の指定されたコントロールの状態やステータスに関する情報を返します。

フラグ

/G[=*doGlobal*] *doGlobal* が 0 以外であるか、または指定されていない場合、V_top と V_left を通じて返される位置は、コントロールを含むウィンドウを基準とした相対座標ではなく、画面全体の座標系での座標となります。

/W=*winName* 指定されたグラフ、パネルウィンドウ、またはサブウィンドウ内でコントロールを検索します。/W を省略した場合、ControlInfo は最前面のグラフ、パネルウィンドウ、またはサブウィンドウを検索対象とします。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

パラメーター

controlName は、/W で指定されたウィンドウ、あるいは最前面のグラフまたはパネルウィンドウ内のコントロールの名前です。
controlName には、V_Red、V_Green、V_Blue、V_Alpha を設定するためのキーワード kwBackgroundColor、V_Height を設定するためのキーワード kwControlBar、kwControlBarTop、または kwControlBarBottom、V_Width を設定するためのキーワード kwControlBarLeft または kwControlBarRight、または S_Value と V_flag を設定するためのキーワード kwSelectedControl を指定することもできます。

詳細

すべてのコントロールの情報は、以下の文字列と数値変数を通じて返されます。
座標はコントロールパネル単位で返されます。

S_recreation 指定されたコントロールを再作成するためのコマンド。

S_title 指定されたコントロールのタイトル。

V_disable Control の無効状態：

0： 通常（有効、表示）

1： 非表示

2： 無効、表示

V_Height, V_Width, V_top, V_left, V_right, V_pos, V_align
コントロールパネル単位における、指定されたコントロールの次元と位置です。

V_right、V_pos、V_align は Igor Pro 8.0 で追加されたもので、コントロールに align キーワードが適用されているかどうかによって異なります（例：Button name align=1）。

control に align キーワードが適用されていた場合、V_align は 1 となり、そうでない場合は 0 となります。

V_pos は、コントロールの位置を指定するために使われる水平座標です。
これは、align キーワードが省略された場合はコントロールの左端の位置を、align キーワードが指定された場合は右端の位置を表します。

V_left と V_right は、align キーワードが指定されているかどうかにかかわらず、コントロールの左右両端の水平座標を表します。

この種類のコントロールは、V_flag に正または負の整数として返されます。
負の値は、そのコントロールが未完成であるか、あるいは何らかの理由でアクティブではないことを示します。
V_flag が 0 の場合、指定されたコントロールは存在しません。
特定のコントロールタイプについて返される情報は以下の通りです：

Button

V_flag 1
V_Value 前回のマウスアップ時のティック数
S_UserData （名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

Chart

V_flag 6 または -6
V_Value 現在のポイント番号
S_Value キーワードが詰まった情報文字列
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

CheckBox

V_flag 2
V_Value 選択されていない場合は 0、選択されている場合は 1
S_UserData （名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

CustomControl

V_flag 12
V_Value 前回のマウスアップ時のティック数
S_UserData （名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

GroupBox

V_flag 9
S_Value タイトルテキスト
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

ListBox

V_flag	11
V_Value	現在選択されている行（モード 1 または 2、または selWave が使われていない場合のモード 5 および 6 で有効）。リストの行が選択されていない場合は、-1 に設定される
V_selCol	現在選択されている列（selWave を使わない場合、モード 5 および 6 で有効）
V_horizScroll	リストが右方向に水平スクロールされたピクセル数
V_vertScroll	リストが下方向に垂直スクロールされたピクセル数
V_rowHeight	行の高さ（ピクセル単位）
V_startRow	現在表示されている最上段の行
S_columnWidths	ピクセル単位の列幅をコンマ区切りで指定したリスト
S_DataFolder	listWave への完全なパス（存在する場合）
S_UserData	（名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_Value	listWave の名前（ある場合）
S_recreation, V_disable, V_Height, V_Width, V_top, V_left, V_right, V_pos, V_align	上記の説明を参照

PopupMenu

V_flag	3 または -3
V_Red, V_Green, V_Blue, V_Alpha	カラー配列のポップアップメニューでは、これらはエンコードされたカラー値
V_Value	現在の項目番号（1 から数えて）
S_UserData	（名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_Value	現在の項目のテキスト。PopupMenu がカラー配列である場合、その配列には RGBA 値としてエンコードされたカラー値が含まれる
S_recreation, V_disable, V_Height, V_Width, V_top, V_left, V_right, V_pos, V_align	上記の説明を参照

SetVariable

V_flag	5 または -5
V_Value	変数の値。SetVariable を文字列変数に対して使った場合、この値は文字列を数値として解釈した結果となる。変換に失敗した場合は NaN となる
S_DataFolder	変数の完全なパス
S_UserData	（名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_Value	変数名、または _STR: 構文を使って値が設定された場合は、その文字列値そのもの
S_recreation, V_disable, V_Height, V_Width, V_top, V_left, V_right, V_pos, V_align	上記の説明を参照

Slider

V_flag	7
V_Value	変数の数値
S_DataFolder	変数の完全なパス
S_UserData	（名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_Value	変数の名前

S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

TabControl

V_flag 8
V_Value 現在のタブの番号
S_UserData （名前が指定されていない）プライマリユーザーデータのテキスト。名前が指定されたユーザーデータについては、GetUserData 関数を使う必要がある
S_Value タブテキスト
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left, V_right, V_pos, V_align
上記の説明を参照

Operations[TitleBox]

V_flag 10
S_DataFolder テキストが文字列変数から取得される場合は、フルパスを指定する
S_Value テキストが文字列変数から取得された場合の名前
S_recreation, V_disable, V_Height, V_Width,
V_top, V_left
上記の説明を参照
V_right, V_pos, V_align
詳細は、TitleBox コマンドの「TitleBox の配置」を参照

kwBackgroundColor

V_Red, V_Green, V_Blue, V_Alpha
controlName が kwBackgroundColor の場合、それはコントロールパネルの背景色。この色は通常、「Display Properties」の「Appearance」タブで設定されたインターフェイスのデフォルトの背景色だが、ModifyPanel cbRGB によって変更されるまではそのまま

kwControlBar または *kwControlBarTop*

V_Height ControlBar で設定された、グラフの上部コントロールバー領域の高さ（ピクセル単位）

kwControlBarBottom

V_Height ControlBar/B で設定された、グラフの下部コントロールバー領域の高さ（ピクセル単位）

kwControlBarLeft

V_Width ControlBar/L で設定された、グラフの左側コントロールバー領域の幅（ピクセル単位）

kwControlBarRight

V_Width ControlBar/R で設定された、グラフの右側コントロールバー領域の幅（ピクセル単位）

kwSelectedControl

V_flag コントロールが選択されている場合は 1 に、選択されていない場合は 0 に設定する
SetVariable と ListBox コントロールは選択可能だが、その他のほとんどのコントロールは選択できない
S_Value 選択したコントロールの名前（ある場合）または ""

以下は、チャートレコーダーの S_value に格納されているキーワードが詰め込まれた情報文字列にのみ適用されます。
S_value は、「キーワード:値;」という形式のセクションの連続から構成されます。
NumberByKey 関数と StringByKey 関数を使うと、キーワードが詰め込まれた文字列から特定の値を取り出すことができます。
以下は、チャートレコーダの S_value に関するキーワードの一覧です。

キーワード	形式	意味
FNAME	文字列	FIFO チャートの名前は monitoring
LHSAMP	数字	左側のサンプル番号
NCHANS	数字	チャートレコーダーに表示されるチャンネル数
PPSTRIP	数字	チャートレコーダーの１ストリップあたりの測定点数。
RHSAMP	数字	右側のサンプル番号（V_value と同じ）

さらに、ControlInfo は、チャートレコーダーの各チャネルについて、S_value にフィールドを書き込みます。
フィールドのキーワードは、そのフィールドと、それが参照するチャネルを識別するための名前と番号の組み合わせです。
例えば、チャネル 4 の名前が「Pressure」の場合、S_value 文字列には「CHNAME4:Pressure」と表示されます。
次の表では、チャネル番号は # で表されています。

キーワード	形式	意味
CHCTAB#	数字	Chart ctab キーワードで設定された、チャネルのカラーテーブルの値
CHGAIN#	数字	Chart gain キーワードで設定されたチャネルのゲイン値
CHNAME#	文字列	FIFO で定義されたチャネル名
CHOFFSET#	数字	Chart offset キーワードで設定されたチャネルのオフセット値

例

```
ControlInfo myChart; Print S_Value
```

その結果、履歴エリアに以下の内容が出力されます：

```
FNAME:myFIFO;NCHANS:1;PPSTRIP:1100;RHSAMP:271;LHSAMP:-126229;
```

参照

コントロールパネルとコントロールに関する詳細は、ヘルプ Controls and Control Panels を参照してください。

コントロールで使われる単位については、ヘルプ Control Panel Units の項を参照してください。

コントロールの情報については ControlInfo コマンドのセクションを参照してください。

指定されたユーザーデータの取得については GetUserData コマンドのセクションを参照してください。

ControlUpdate [/A /W=*winName*] [*controlName*]

ControlUpdate コマンドは、指定されたコントロール、またはウィンドウ内のすべてのコントロールを更新します。

ここでいうウィンドウとは、最前面のグラフまたはコントロールパネル、あるいは /W オプションを使う場合は、指定されたグラフまたはコントロールパネルを指します。

フラグ

/A　ウィンドウ内のすべてのコントロールを更新します。*controlName* は省略する必要があります。

/W=*winName*　コントロールが含まれるウィンドウまたはサブウィンドウを指定します。*winName* を省略した場合、最前面のグラフまたはコントロールパネルのウィンドウもしくはサブウィンドウを使います。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

ControlUpdate は、ポップアップメニューの再構築を強制したり、ValDisplay コントロールを更新したり、SetVariable で現在編集中の値を強制的に確定させたりするのに役立ちます。

通常、ポップアップメニューはユーザーがクリックしたときにのみ再構築されます。
ポップアップメニューの内容を、グローバル文字列変数、ユーザー定義の文字列関数、または Igor 関数（例：WaveList）に依存するように設定している場合、必要に応じてポップアップメニューを強制的に更新したいことがあるかもしれません。

通常、ValDisplay コントロールは、グローバル変数の値、またはグローバル変数を含む式の値を表示します。
グローバル変数が変更されると、ValDisplay は自動的に更新されます。
ただし、グローバル変数に依存しない値を表示する ValDisplay を作成することも可能です。
例えば、外部関数の結果を表示する場合などです。
このような場合、ValDisplay は自動的に更新されません。
ControlUpdate を呼び出すことで、値を更新することができます。

SetVariable コントロールが編集中の場合、ユーザーが入力したテキストは、ユーザーが Return キーまたは Enter キーを押すまで「受け入れ」られず（処理されず）、そのままの状態になります。
ControlUpdate は、指定されたコントロールに対して、ユーザーがこれらのキーのいずれかを押したかのように動作させる効果があります。
/A が指定された場合、現在アクティブな SetVariable コントロール（存在する場合）がこの処理の対象となります。
この機能の背景には、ユーザーが Return キーを押す前に新しい値を入力し、その後、別のパネルにあるボタンをクリックして、SetVariable の値を入力として使うルーチンが実行されてしまう可能性があるという事情があります。
ユーザーは、入力した値が受け入れられたと期待していますが、変数にはまだ値が設定されていません。
最初のパネルで ControlUpdate/A を呼び出すと、変数に入力された値が読み込まれ、SetVariable コントロールの表示値と変数の実際の値との不一致が回避されます。

例

ポップアップコントロールのリストが、グローバル変数や文字列関数などの文字列式である場合、式の値が変更されても、Igor Pro がポップアップメニューを更新しないことがあります。
例えば：

```
NewPanel;DoWindow/C PanelX
String/G popupList="First;Second;Third"
PopupMenu oneOfThree value=popupList          // ポップアップに "First" を表示
popupList="1;2;3"                             // ポップアップは変更されない
ControlUpdate/W=PanelX oneOfThree             // ポップアップが "1" を表示
```

参照

コントロールパネルとコントロールに関する詳細は、ヘルプ Controls and Control Panels を参照してください。

ConvertGlobalStringTextEncoding [*flags*] *originalTextEncoding*, *newTextEncoding* , [*string* , *string*, ...]

ConvertGlobalStringTextEncoding コマンドは、指定されたグローバル文字列変数の内容、または /DF フラグで指定されたデータフォルダー内のすべてのグローバル文字列変数の内容を、あるテキストエンコーディングから別のテキストエンコーディングに変換します。

ConvertGlobalStringTextEncoding コマンドは、Igor Pro 7.0 で追加されました。

デフォルトでは、キャリッジリターン (13)、ラインフィード (10)、タブ (9) 以外の制御文字（コードが 32 未満のもの）を含むグローバル文字列変数の内容は変換されません。
制御文字を含むグローバル文字列変数の内容を変換するには、/SKIP=0 フラグを指定する必要があります。

また、Igor Pro 9.0 以降では、デフォルトで UTF-8 への変換を行う時、すでに UTF-8 として有効な文字列はスキップされます。

Igor Pro 7 以降では、すべてのグローバル文字列変数の内容は UTF-8 でエンコードされているものとみなされます。
例えば、Igor Pro 7 より前に作成されたエクスペリメントにグローバル文字列変数が含まれており、そこに非 ASCII 文字が含まれている場合は、そのグローバル文字列変数のテキストエンコーディングを変換する必要があります。
これらの文字列を正しく表示および解釈するためには、その内容を UTF-8 に変換する必要があります。

ほとんどのグローバル文字列変数には非 ASCII 文字が含まれていないため、ほとんどのユーザーは Igor Pro のグローバル文字列変数のテキストエンコーディングについて心配する必要はありません。
テキストエンコーディングの問題について十分に理解しているか、あるいはその分野に精通した人物から使用を指示された場合を除き、このコマンドを使うべきではありません。

基本的な背景情報については、ヘルプ String Variable Text Encodings を参照してください。

ConvertGlobalStringTextEncoding は、特定のグローバル文字列変数のリストに対して、あるいはデータフォルダー内のすべてのグローバル文字列変数に対して処理を行うことができます（/DF フラグ）。
データフォルダーを処理する場合、データフォルダー自体のみを対象とすることも、サブデータフォルダーに対しても再帰的に処理を行うことも可能です。

文字変換によって、テキストを構成する文字そのものが変わるわけではありません。
単に、それらの文字を表すために使われる数値コードが変わるだけです。

パラメーター

originalTextEncoding は、指定されたグローバル文字列変数で使用される元の（現在の）テキストエンコーディングを指定します。
コードの一覧については、ヘルプ Text Encoding Names and Codes を参照してください。

newTextEncoding は、出力テキストのエンコーディングを指定します。
コードの一覧については、ヘルプ Text Encoding Names and Codes を参照してください。

string, *string*, ... は、対象となるグローバル文字列変数のリストです。
パス指定と名前を組み合わせた形式ではなく、文字列変数の名前のみを指定できます。
したがって、文字列変数は現在のデータフォルダー内に存在している必要があります。
このリストはオプションであり、/DF フラグを使う場合は省略する必要があります。
/DF フラグと文字列変数のリストを併用すると、エラーとして扱われます。
このリストにローカル文字列変数を含めることもエラーとなります。
ローカル文字列変数を変換するには、ConvertTextEncoding を使ってください。

フラグ

/CONV={*errorMode*[, *diagnosticsFlags*]}

errorMode は、テキストを指定されたテキストエンコーディングにマッピングできないために変換が行えない場合、*ConvertGlobalStringTextEncoding* がどのように動作するかを決定します。これは、テキストに指定されたテキストエンコーディングで表現できない文字が含まれている場合、または *originalTextEncoding* の値が間違っている場合に発生します。

errorMode は、以下のいずれかの値をとります：

1： エラーを生成します。ConvertGlobalStringTextEncoding がエラーを返します。

2： マッピングできない文字に対して、代替文字を使います。Unicode の代替文字は、Unicode 置換文字「□」（U+FFFD）です。Unicode 以外のほとんどのテキストエンコーディングでは、Control-Z または疑問符が使われます。

3： マッピングできない入力文字をスキップします。マッピングできない文字は、出力から除外されます。

4： マッピングできない文字や無効なソーステキストを表すには、エスケープシーケンスを使ってください。

ソーステキストがソーステキストのエンコーディングでは有効であるが、宛先テキストのエンコーディングでは表現できない場合、マッピング不可能な文字は ¥uXXXX に置き換えられます。ここで、XXXX は、マッピング不可能な文字の UTF-16 コードポイントを 16 進数で表したものです。

ソーステキストがソーステキストのエンコーディングで有効でないために変換ができない場合、無効なバイトは ¥xXX に置き換えられます。ここで、XX は無効なバイトの値を 16 進数で表したものです。

diagnosticsFlags は、次のように定義されるオプションのビット単位のパラメーターです。

Bit 0： テキストの変換に成功した場合に、診断メッセージを出力します。

Bit 1： テキストの変換に失敗した場合に、診断メッセージを出力します。

Bit 2： テキストの変換がスキップされた場合に、診断メッセージを出力します。

Bit 3： 診断メッセージの概要を出力します。

その他のビットはすべて、将来の使用のために予約されています。

ビット設定の詳細については、ヘルプ Setting Bit Parameters を参照してください。

/DF フラグが指定されていない場合、diagnosticsFlags のデフォルト値は 2（ビット 1 がセット）となり、/DF フラグが指定されている場合は 10（ビット 1 および 3 がセット）となります。

グローバル文字列変数の内容がバイナリであると検出された場合、または *newTextEncoding* が *originalTextEncoding* と同じである場合、テキストの変換がスキップされることがあります。

/DF={*dfr*, *recurse*, *excludedDFR*}

dfr はデータフォルダーへの参照です。ConvertGlobalStringTextEncoding は、指定されたデータフォルダー内のすべてのグローバル文字列変数に対して処理を行います。*dfr* が null ($"") の場合、ConvertGlobalStringTextEncoding は /DF が省略されたかのように動作します。/DF フラグを使う場合は、オプションの文字列リストパラメーターを使ってはなりません。

recurse が 1 の場合、ConvertGlobalStringTextEncoding はすべてのサブデータフォルダーに対して再帰的に処理を行います。それ以外の場合は、*dfr* で参照されるデータフォルダーに対してのみ処理が行われます。

excludedDFR は、ConvertGlobalStringTextEncoding によってスキップされるデータフォルダーへのオプションの参照です。
例えば、次のコマンドは、root:Packages とそのサブデータフォルダーを除く、すべてのデータフォルダー内のグローバル文字列変数の文字列データを変換します。

```
ConvertGlobalStringTextEncoding /DF={root:,1,root:Packages} 4, 1
```

excludedDFR が null ($"") の場合、ConvertGlobalStringTextEncoding は、*excludedDFR* が省略され、データフォルダーが除外されていないかのように動作します。

/SKIP=*skip*　次のように、文字列変数の型変換をスキップします。

skip=0：　変換を省略しない

skip=1：　バイナリデータを含む文字列の変換をスキップする

skip=2：　UTF-8 に変換する時、すでに UTF-8 として有効な文字列の変換はスキップする。これには、ASCII 文字のみを含む文字列も含まれる

skip=3：　バイナリデータを含む文字列と、UTF-8 への変換時にすでに UTF-8 として有効な文字列の両方をスキップする。これは、/SKIP を省略した場合のデフォルトの動作

/SKIP=2 と /SKIP=3 を使うには、Igor Pro 9.0 以降が必要

/Z[=*z*]　ConvertGlobalStringTextEncoding でエラーが発生した場合でも、プロシージャの実行が中断されないようにします。実行を中断させるのではなく、プロシージャ内でエラーを処理したい場合は、/Z またはそれに相当する /Z=1 を指定してください。

/Z オプションは、無効なパラメーターに関するエラーを抑制しません。テキストのエンコード変換を行う時のエラーのみを抑制します。

詳細

テキストエンコーディングに関する一般的な背景情報は、ヘルプ Text Encodings を参照してください。

文字列変数のテキストエンコーディングに関する背景情報は、ヘルプ String Variable Text Encodings を参照してください。

ConvertGlobalStringTextEncoding の使用

ConvertGlobalStringTextEncoding は、テキストを表す数値コードを変更する、つまり、コンテンツを別のテキストエンコーディングに変換するために使われます。
その主な用途は、Igor Pro 6 のグローバル文字列変数を、使われているテキストエンコーディングから UTF-8 に変換することです。
UTF-8 は Unicode の一種であり、より新しい形式ですが、Igor Pro 6 との下位互換性はありません。
Igor Pro 7 以降では、文字列変数は UTF-8 でエンコードされているものとみなされます。

例えば、Igor Pro 6 のグローバル文字列変数が日本語（Shift JIS）でエンコードされている場合は、それらを UTF-8 に変換する必要があります。
そうしないと、文字列を出力または表示した際にエラーが発生したり、文字化けしたりします。
これは、MacRoman や Windows-1252 でエンコードされた非 ASCII 文字を含む欧米語のテキストにも当てはまります。

Igor Pro 6 のエクスペリメントに、例えば MacRoman などでエンコードされた非 ASCII 文字列変数が含まれており、Igor Pro 7 以降で非 ASCII 文字列変数を追加した場合、MacRoman と UTF-8 の非 ASCII 文字

列変数が混在することになります。
Igor Pro 9 以降で UTF-8 に変換する場合、デフォルトでは、ConvertGlobalStringTextEncoding は、すでに UTF-8 として有効な文字列の変換をスキップします。
これにより、すでに UTF-8 である文字列変数を破損させることなく、MacRoman の文字列変数を UTF-8 に変換することができます。

出力変数

ConvertGlobalStringTextEncoding コマンドは、以下の変数に情報を返します。

V_*numConversionsSucceeded* テキスト変換に成功した件数に設定します。

V_*numConversionsFailed* テキスト変換に失敗した回数に設定されます。

V_*numConversionsSkipped* スキップされるテキスト変換の回数に設定します。文字列変数にバイナリデータが含まれており、かつ /SKIP=0 フラグが使われていない場合、テキスト変換がスキップされることがあります。

文字列変数にバイナリデータが含まれている場合、デフォルトではテキスト変換はスキップされます。また、*newTextEncoding* が 1 (UTF-8) で、かつ文字列変数がすでに UTF-8 として有効である場合も、デフォルトではテキスト変換はスキップされます。詳細は、/SKIP フラグを参照してください。

例

```
// 以下の例では、1 は UTF-8、4 は Shift JIS を表します。

// 特定の文字列の内容を Shift JIS から UTF-8 に変換する
ConvertGlobalStringTextEncoding 4, 1, string0, string1

// すべての文字列の内容を Shift JIS から UTF-8 に変換する
ConvertGlobalStringTextEncoding /DF={root:,1} 4, 1

// 上記と同じですが、root:Packages データフォルダーを除外する
ConvertGlobalStringTextEncoding /DF={root:,1,root:Packages} 4, 1

// バイナリデータを含む文字列を除き、すべての文字列の内容を Shift JIS から UTF-8 に変換する
ConvertGlobalStringTextEncoding /DF={root:,1}/SKIP=0 4, 1
```

参照

ConvertTextEncoding、SetWaveTextEncoding コマンド
ヘルプ Text Encodings、String Variable Text Encodings、Text Encoding Names and Codes

ConvexHull [/C/E/I/S/T=*tolerance* /V/Z] *xwave*, *ywave*

ConvexHull [/C/E/I/S/T=*tolerance* /V/Z] *tripletWave*

ConvexHull コマンドは、２次元または３次元の凸包を計算します。
次元数は、入力ウェーブから推定されます。
入力が同じ長さの２つの 1D ウェーブで構成されている場合、次元数は２とみなされます。
入力が１つのトリプレットウェーブ（３列のウェーブ）で構成されている場合、次元数は３となります。

2D の場合、この演算は凸包を計算し、その結果を x ウェーブと y ウェーブのペアである W_XHull と W_YHull として出力します。

3D の場合、この演算は凸包を計算し、それを凸包の順序付きファセットを表すトリプレットウェーブ M_Hull に格納します。

入力ウェーブに含まれるデータポイントが 3 つ未満の場合、ConvexHull はエラーを返します。

フラグ

/C　　（2D の凸包にのみ適用）W_XHull と W_YHull のウェーブの末尾に最初のポイントを追加し、最初と最後のポイントが同じになるようにします。

/DEST=*destWave*

このコマンドによって生成される出力ウェーブを指定します。2D 領域に対して凸包が計算された場合、結果として得られる *destWave* は 2 列のウェーブとなります。3D 領域に対して凸包が計算された場合、*destWave* は 3 列のウェーブとなります（デフォルトで生成される M_Hull に対応します）。

srcWave と *destWave* の両方に同じウェーブを指定するとエラーになります。

関数内で使われた場合、ConvexHull コマンドは、宛先ウェーブに対して実数ウェーブ参照を作成します。詳細は、ヘルプ Automatic Creation of WAVE References を参照してください。

/E　　（3D の場合のみ）このフラグを使うと、このコマンドによって、凸包に含まれない頂点、つまり凸包の内部にある頂点のインデックスを列挙したウェーブも生成されます。出力はウェーブ W_HullExcluded になります。

/FREE　　*destWave* をフリーウェーブとして作成します。

/FREE は関数内でのみ使用可能で、かつ /DEST で指定された *destWave* が単純な名前またはウェーブ参照構造体のフィールドである場合に限り使用できます。

詳細は、ヘルプ Free Waves を参照してください。

/FREE フラグは、Igor Pro 10.0 で追加されました。

/I　　（3D 凸包にのみに適用）このフラグを使うと、M_Hull ウェーブの 4 列目に、対応する頂点のインデックスが格納されます。

/S　　（3D 凸包にのみに適用）結果の M_Hull において、各三角形を区切る線に NaN を含めたい場合は、このフラグを使ってください。

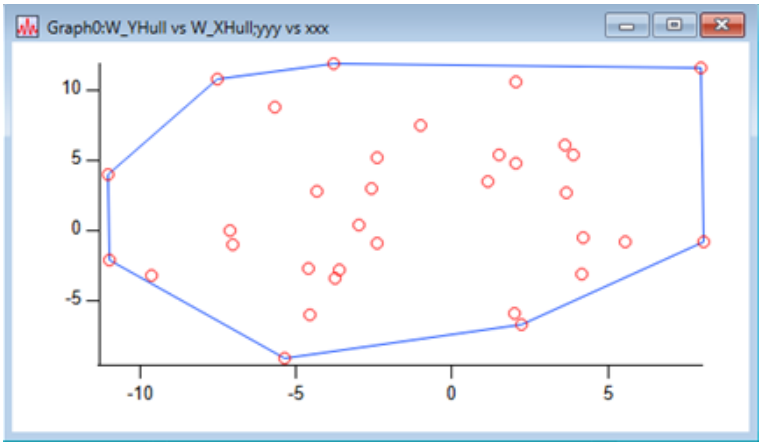
/T=*tolerance*　　（3D の場合のみ）ポイントが凸包の内側にあるか外側にあるかを判定する時のデフォルトの許容誤差は 1.0e-20 です。この値の代わりに、任意の正の値を使うことができます。

/V　　（3D の場合のみ）このフラグを使うと、このコマンドによって、出力データを頂点インデックスのリストとして含むウェーブも生成されます。ウェーブ M_HullVertices には、三角形ごとに 1 行が割り当てられ、各行の各エントリは入力頂点のインデックスに対応しています。

/Z　　エラーの報告はありません。

例

```
Make/O/N=33 xxx=gnoise(5),yyy=gnoise(7)
Convexhull/c  xxx,yyy
Display W_Yhull vs W_Xhull
Appendtograph yyy vs xxx
ModifyGraph mode(yyy)=3,marker(yyy)=8,rgb(W_YHull)=(0,15872,65280)
```



参照

Triangulate3D コマンド

Convolve [/A/C] *srcWaveName*, *destWaveName* [, *destWaveName*]...

Convolve コマンドは、*srcWaveName* と各宛先ウェーブを畳み込み、各畳み込みの結果を対応する宛先ウェーブに格納します。

Convolve は多次元に対応していません。
一部の多次元畳み込みについては、MatrixConvolve、MatrixFilter、MatrixOp コマンドで対応しています。

フラグ

/A　　非因果的線形畳み込み

/C　　循環畳み込み

詳細

Convolve は、/C または /A フラグが指定されていない限り、線形畳み込みを実行します。
詳細は、ヘルプ Convolution を参照してください。

畳み込みの種類によっては、宛先ウェーブの長さが増加する場合があります。
srcWaveName は、宛先ウェーブとしても指定されていない限り、変更されません。

srcWaveName が実数型の場合、各宛先ウェーブも実数型でなければならず、*srcWaveName* が複素数ウェーブ型の場合、各宛先ウェーブも複素数ウェーブ型でなければなりません。倍精度と単精度のウェーブは自由に混在させることができます。計算は倍精度で行われます。

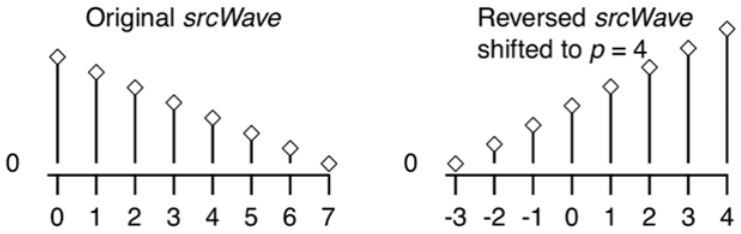
線形畳み込み方程式は次のとおりです：

$$destWaveOut[p] = \sum_{m=0}^{N-1} destWaveIn[m] \cdot srcWave[p-m]$$

ここで、N は *destWaveIn* と *srcWave* のうち長い方のポイント数です。
循環畳み込みの場合、インデックス [*p*-*m*] が [0, numpts(*srcWave*)-1] の範囲を超えるときは、そのインデックスがループバックされます。
非因果畳み込みの場合、[*p*-*m*] が範囲を超えるときは、srcWave [*p*-*m*] の値が 0 に置き換えられます。
同様の処理が *destWaveIn* [*m*] に対しても適用されます。

この式を別の見方をすれば、すべての *p* について、*destWaveOut*[*p*] は、宛先ウェーブと、*p* だけ右にシフトされ端から端まで反転されたソースウェーブとの、0 から *p* までの各ポイントにおける積の和に等しいということになります。

次の図は、*destWaveIn* と合成される、反転／シフトされた *srcWave* を示しています。
反転された *srcWave* の 0 から 4 までの各ポイントは、*destWaveIn*[0...4]と乗算され、それらを合計することで *destWaveOut*[4] が生成されます。



線形畳み込みおよび非因果畳み込みの場合、まず、対象ウェーブに対して、ソースウェーブの長さの 1 つ分短い長さだけゼロパディングが行われます。
これにより、循環畳み込みで発生する「ラップアラウンド」効果が防止されます。
ゼロパディングされたポイントは、非因果畳み込みの後は削除され、線形畳み込みの後は保持されます。
ウェーブの X 軸スケーリングは無視されます。

畳み込み処理は、ソースウェーブと宛先ウェーブを高速フーリエ変換（FFT）で変換し、周波数領域でそれらを乗算した後、逆変換を行って宛先ウェーブを生成することで実行されます。

結果のウェーブが 256 ポイントを超え、かつ宛先ウェーブのポイント数がソースウェーブの 2 倍である場合、畳み込みはセグメント単位で実行されます。
非因果畳み込みの場合、この計算において、結果のウェーブの長さは numpnts(*srcWaveName*) + numpnts(*destWaveName*) - 1 とみなされます。

応用

畳み込みの一般的な用途は、インパルス応答によって定義される線形システムが、入力信号に対してどのような応答を示すかを計算することです。
srcWaveName にはインパルス応答が格納され、宛先ウェーブには当初、入力信号が格納されています。
畳み込み演算が完了すると、宛先ウェーブには出力信号が格納されます。

srcWaveName の最初のポイントが遅延なし（t = 0）に対応するインパルス応答（またはフィルター係数）がソースウェーブに含まれている場合は、線形畳み込みを使ってください。

srcWaveName と *destWaveName* のデータが無限に繰り返される（あるいは「ループして」終わりからまた先頭に戻る）とみなされる場合は、循環畳み込みを使います。
これにより、ゼロパディングは不要となります。

srcWaveName の中央のポイントが遅延なし（t = 0）に対応するインパルス応答がソースウェーブに含まれている場合は、非因果畳み込みを使ってください。

参照

Correlate、Convolution、MatrixOp コマンド

参考文献

循環畳み込みおよび線形畳み込みに関する非常に詳細な説明は、Rabiner and Gold 著『Theory and Application of Digital Signal Processing』（Prentice Hall、1975 年）の第 2.23 節および第 2.24 節に掲載されています。

CopyDimLabels [*flags*] *srcWave*, *destWave* [, *destWave*]...

CopyDimLabels コマンドは、ソースのウェーブから宛先のウェーブ（1 つまたは複数）へ、次元ラベルをコピーします。

CopyDimLabels は Igor Pro 8.0 で追加されました。

Igor Pro 9.0 では、複数の宛先ウェーブへの対応が追加されました。

フラグ

以下のフラグにおいて、行次元の場合は *dim* が 0、列次元の場合は 1、レイヤー次元の場合は 2、チャンク次元の場合は 3 となります。

/ROWS=*dim*　　*srcWave* の行次元ラベルを、*dim* で指定した *destWave* の次元にコピーします。

/COLS=*dim*　　*srcWave* の列次元ラベルを、*dim* で指定した *destWave* の次元にコピーします。

/LAYR=*dim*　　*srcWave* のレイヤー次元ラベルを、*dim* で指定した *destWave* の次元へコピーします。

/CHNK=*dim*　　*srcWave* のチャンク次元ラベルを、*dim* で指定した *destWave* の次元へコピーします。

詳細

すべてのフラグを省略した場合、CopyDimLabels は、*srcWave* 内のすべての次元ラベルを、*destWave* に存在する次元について、*destWave* の対応する次元ラベルにコピーします。

/ROWS、/COLS、/LAYR、/CHNK を使うと、既存の任意のソースウェーブの次元ラベルから、既存の任意の宛先ウェーブの次元ラベルへ次元ラベルをコピーできます。
例えば、wave1 の列次元ラベルを wave2 の行次元ラベルにコピーするには、次のように指定します。

```
CopyDimLabels /COLS=0 wave1, wave2
```

wave1 の列次元ラベルを wave2 の行次元ラベルに、wave1 の行次元ラベルを wave2 の列次元ラベルにコピーするには、次のように入力します。

```
CopyDimLabels /COLS=2 /ROWS=1 wave1, wave2
```

ソース次元が N 個の要素を持ち、宛先の次元が M（N より大きい）個の要素を持つ場合、宛先には最初の N 個の次元ラベルのみが設定されます。
宛先の残りの次元ラベルは変更されません。

ソース次元が N 個の要素を持ち、宛先の次元が M < N 個の要素を持つ場合、ソースから宛先へは最初の M 個の次元ラベルのみがコピーされます。

目的先の存在しない次元に次元ラベルをコピーしようとすると、エラーが発生します。

参照

FindDimLabel、GetDimLabel、SetDimLabel コマンド
ヘルプ Dimension Labels

CopyScales [/I/P] *srcWaveName*, *waveName* [, *waveName*]...

CopyScales コマンドは、*srcWaveName* の x、y、z、t のスケーリング、x、y、z、t の単位、データのフルスケール、データ単位を、他のウェーブにコピーします。

フラグ

/I　　x、y、z、t のスケーリングを包括形式でコピーします。

/P　　x、y、z、t のスケーリングを、傾き／切片形式（x0、delta x 形式）でコピーします。

詳細

通常、x、y、z、t（次元）のスケーリングは、min/max 形式でコピーされます。
ただし、/P を使うと、次元のスケーリングは傾き／切片形式でコピーされるため、*srcWaveName* と他のウェーブとで次元のサイズ（1D ウェーブの場合はポイント数）が異なっていても、共通するポイントについては次元の値が一致するようになります。
同様に、/I オプションでは、min/max 形式の「包含型」が使われます。
これらの次元スケーリング形式に関する詳細については、SetScale を参照してください。

ウェーブが 1 ポイントしかない場合、/I モードは /P モードに戻ります。

CopyScales は、*srcWaveName* と *waveName* に共通する次元のスケールのみをコピーします。

CopyFile [/D /I[=*i*]/M=*messageStr* /O/P=*pathName* /S=*saveMessageStr* /Z[=*z*]] [*srcFileStr*] [*as destFileOrFolderStr*]

CopyFile コマンドは、ディスク上のファイルをコピーします。

パラメーター

srcFileStr には、コピーするファイルへの完全なパス（この場合、/P は不要）、*pathName* に関連付けられたフォルダーからの相対パス、または *pathName* に関連付けられたフォルダー内のファイル名を指定できます。

srcFileStr と *pathName* からソースファイルの場所を特定できない場合、ソースファイルを指定できるダイアログが表示されます。

/D が指定された場合、*destFileOrFolderStr* は既存のフォルダーの名前（またはパス）として解釈されます。それ以外の場合は、存在する場合のあるファイルの名前（またはパス）として解釈されます。

destFileOrFolderStr が部分パスである場合、そのパスは *pathName* に関連付けられたフォルダーを基準とした相対パスとなります。

/D を指定した場合、ソースファイルは、そのファイル名と同じ名前のフォルダー内にコピーされます。

pathName、*srcFileStr*、*destFileOrFolderStr* から保存先のファイルの場所を特定できない場合、保存先ファイル（およびフォルダー）を指定できる Save File ダイアログが表示されます。

srcFileStr または *destFileOrFolderStr* に完全パスまたは部分パスを使う場合は、パスの構成方法の詳細についてヘルプ Path Separators を参照してください。

フォルダーパスは、単一のパス区切り文字で終わってはいけません。
MoveFolder コマンドの「詳細」セクションを参照してください。

フラグ

/D　　*destFileOrFolderStr* を、既存のフォルダー（または「ディレクトリ」）の名前（またはパス）として解釈します。/D オプションを指定しない場合、*destFileOrFolderStr* はファイルの名前（またはパス）として解釈されます。

destFileOrFolderStr がフォルダーの完全なパスでない場合、そのパスは *pathName* に関連付けられたフォルダーを基準とした相対パスとなります。

/I[=*i*]　　ユーザーとの対話性のレベルを指定します。

i=0：　*srcFileStr* または *destFileOrFolderStr* のいずれかが指定されていない場合、あるいはソースファイルが存在しない場合にのみ、対話モードになります。（/I が指定されていない場合と同じです。）

i=1： *srcFileStr* が指定され、ソースファイルが存在する場合でも、対話モードで実行されます。

i=2： *destFileOrFolderStr* が指定されている場合でも、対話モードになります。

i=3： *srcFileStr* が指定され、ソースファイルが存在し、かつ *destFileOrFolderStr* が指定されている場合でも、対話モードになります。/I オプションのみと同じ動作です。

/M=*messageStr*

Open File ダイアログに表示されるプロンプトメッセージを指定します。/S が指定されていない場合、*messageStr* は Open File ダイアログと Save File ダイアログの両方で使われます。

/O 既存の保存先ファイルがある場合は、それを上書きします。

/P=*pathname* ソースファイルの検索対象となるフォルダーと、ファイルがコピーされるフォルダーを指定します。*pathName* は、既存のシンボリックパスの名前です。

/P オプションを使う場合、*srcFileStr* と *destFileOrFolderStr* の両方が、単純なファイル名またはフォルダー名、あるいは *pathName* で指定されたフォルダーからの相対パスでなければなりません。

/S=*saveMessageStr*

Save File ダイアログに表示されるプロンプトメッセージを指定します。

/Z[=*z*] 存在しないファイルをコピーしようとした場合でも、プロシージャの実行が中断されないようにします。実行を中断させるのではなく、プロシージャ内でこのケースを処理したい場合は、/Z オプションを使ってください。

/Z=0： /Z を一切指定しない場合と同じです。

/Z=1： ファイルが存在する場合にのみ、それをコピーするために使います。/Z のみを指定した場合、/Z=1 を指定した場合と同じ効果があります。

/Z=2： ファイルが存在する場合はそのファイルをコピーし、存在しない場合はダイアログを表示するために使われます。

変数

CopyFile コマンドでは、以下の変数に情報が返されます。

V_flag ファイルがコピーされた場合は V_flag が 0 に、ユーザーが Open File または Save File ダイアログをキャンセルした場合は -1 に、指定されたファイルが存在しないなどのエラーが発生した場合は 0 以外の値に設定されます。

S_fileName コピーされたファイルへのフルパスを格納します。エラーが発生した場合、またはユーザーがキャンセルした場合は、空の文字列に設定されます。

S_path ファイルのコピーのフルパスを格納します。エラーが発生した場合、またはユーザーがキャンセルした場合は、空の文字列に設定されます。

例

```
// 同じフォルダー内のファイルを、新しい名前でコピーする
CopyFile/P=myPath "afile.txt" as "destFile.txt"

// 元のファイル名をそのままにして、ファイルをサブフォルダーにコピーする（/P オプションを使用）
CopyFile/D/P=myPath "afile.txt" as ":subfolder"
Print S_Path    // prints "Macintosh HD:folder:subfolder:afile.txt"
```

```
// ファイルをサブフォルダーに、元のファイル名（フルパス指定）のままコピーする
CopyFile/D "Macintosh HD:folder:afile.txt" as "Server:archive"

// あるフォルダーから別のフォルダーへファイルをコピーし、そのコピーに新しい名前を付ける
CopyFile "Macintosh HD:folder:afile.txt" as "Server:archive:destFile.txt"

// ユーザーが選択したファイルを、任意のフォルダーから「myPath」フォルダー内に「destFile.txt」という
// 名前でコピーする（destFile.txt が存在しない場合でも保存するかどうかを確認する）
CopyFile/I=2/P=myPath as "destFile.txt"

// ユーザーが選択したファイルを、任意のフォルダー内の任意のフォルダーに「destFile.txt」という名前で
// コピーする
CopyFile as "destFile.txt"
```

参照

Open、MoveFile、DeleteFile、CopyFolder、IndexedFile コマンド
ヘルプ Symbolic Paths

CopyFolder [/D/I[=*i*]/M=*messageStr* /O/P=*pathName* /S=*saveMessageStr* /Z[=*z*]] [*srcFolderStr*] [*as destFolderStr*]

CopyFolder コマンドは、ディスク上のフォルダー（およびその内容）をコピーします。

注意

CopyFolder コマンドは、別のフォルダーやその中身を上書きしてしまうことで、データを破壊する恐れがあります。

ディスク上の既存のフォルダーを上書きする場合、CopyFolder はユーザーから許可が得られた場合にのみ上書きを行います。デフォルトでは、許可を求めるダイアログが表示されます。ユーザーは、Miscellaneous Settings ダイアログの Misc カテゴリから、この動作を変更することができます。

アクセスが拒否された場合、そのフォルダーはコピーされず、V_Flag は 1088（コマンドが無効です）または 1275（フォルダーの上書きを拒否しました）を返します。/Z フラグが指定されていない限り、コマンドの実行は中止されます。

srcFolderStr には、コピー対象のフォルダーの完全なパス（この場合、/P は不要）、*pathName* で指定されたフォルダーを基準とした相対パス、または *pathName* で指定されたフォルダー内のフォルダー名を指定できます。

srcFolderStr と *pathName* からフォルダーの場所を特定できない場合、ソースフォルダーを指定できるダイアログが表示されます。

/P=*pathName* が指定され、*srcFolderStr* が指定されていない場合、*pathName* に関連付けられたフォルダーがコピーされます。

destFolderStr には、出力（保存先）フォルダーの完全なパスを指定することも（その場合は /P は不要）、*pathName* で指定されたフォルダーを基準とした相対パスを指定することもできます。

宛先フォルダーがソースフォルダー内にある場合、エラーが返されます。

destFolderStr と *pathName* から保存先のフォルダーの場所を特定できない場合、保存先のフォルダーを指定または作成できるダイアログが表示されます。

いずれかのフォルダーに対して完全パスまたは部分パスを使う場合は、パスの構成方法の詳細についてヘルプ Path Separators を参照してください。

フラグ

/D　　*destFolderStr* を、ソースフォルダーをコピーする先の既存のフォルダー（または「ディレクトリ」）の名前（またはパス）として解釈します。/D オプションを指定しない場合、*destFolderStr* はコピー先のフォルダーの名前（またはパス）として解釈されます。

destFolderStr がフォルダーの完全なパスでない場合、そのパスは *pathName* に関連付けられたフォルダーを基準とした相対パスとなります。

/I[=*i*]　　ユーザーとの対話性のレベルを指定します。

i=0：　ソースフォルダーまたは宛先フォルダーが指定されていない場合、またはソースフォルダーが存在しない場合にのみ、対話モードになります。（/I が指定されていない場合と同じです。）

i=1:　ソースフォルダーが指定されており、そのフォルダーが存在する場合でも、対話形式で実行されます。

i=2:　*destFolderStr* が指定されている場合でも、対話モードになります。

i=3:　ソースフォルダーが指定され、ソースファイルが存在し、かつ *destFolderStr* が指定されている場合でも、対話モードで実行されます。/I オプションのみを指定した場合と同じ動作です。

/M=*messageStr*

Select (source) Folder ダイアログに表示されるプロンプトメッセージを指定します。/S が指定されていない場合、*messageStr* が Select Folder ダイアログと Create Folder ダイアログで使われます。

/O　　既存の保存先フォルダーがある場合は、そのフォルダーを上書きします。

Windows スタイルの「ミックスイン移動」が行われます。これは、ソースフォルダーの内容を宛先フォルダーに移動し、同名のファイルは置き換える一方で、それ以外のファイルは元の場所に残す操作です。

/P=*pathname*　　ソースフォルダを検索するフォルダーを指定します。*pathName* は、既存のシンボリックパスの名前です。

srcFolderStr が指定されていない場合、*pathName* に関連付けられたフォルダーがコピーされます。

/P オプションを使用する場合、*srcFolderStr*（指定されている場合）と *destFolderStr* は、単純なフォルダー名、または *pathName* で指定されたフォルダーを基準とした相対パスのいずれかでなければなりません。

/S=*saveMessageStr*

Create Folder ダイアログに表示されるプロンプトメッセージを指定します。

/Z[=*z*]　　存在しないファイルをコピーしようとした場合でも、プロシージャの実行が中断されないようにします。実行を中断させるのではなく、プロシージャ内でこのケースを処理したい場合は、/Z オプションを使ってください。

/Z=0：　/Z を一切指定しない場合と同じです。

/Z=1：　そのフォルダーが存在する場合にのみコピーします。/Z のみを指定した場合、/Z=1 を指定した場合と同じ効果があります。

/Z=2：　フォルダーが存在する場合はそのフォルダーをコピーし、存在しない場合はダイアログを表示します。

変数

CopyFolder コマンドでは、以下の変数に情報が返されます。

V_flag　フォルダーがコピーされた場合は 0 に、ユーザーが Select Folder または Create Folder ダイアログをキャンセルした場合は -1 に、指定されたファイルが存在しないなどのエラーが発生した場合は 0 以外の値に設定されます。

S_fileName　コピーされたフォルダーのフルパスを、末尾にコロンを付けて格納します。エラーが発生した場合や、ユーザーがキャンセルした場合は、空の文字列に設定されます。

S_path　フォルダーのコピー先のフルパスを、末尾にコロンを付けて格納します。エラーが発生した場合やユーザーがキャンセルした場合は、空の文字列に設定されます。

詳細

コピーするソースフォルダーを指定するには、/P=pathName（*srcFolderStr* を指定しない）のみを使用できます。

フォルダーパスは、1 つのパス区切り文字で終わってはいけません。MoveFolder コマンドの「詳細」セクションを参照してください。

参照

Open、MoveFile、DeleteFile、MoveFolder、NewPath、IndexedDir コマンド
ヘルプ Symbolic Paths

Correlate [/AUTO/C/NODC] *srcWaveName*, *destWaveName* [, *destWaveName*]...

Correlate コマンドは、*srcWaveName* と各宛先ウェーブとの相関を計算し、各相関の結果を対応する宛先ウェーブに格納します。

フラグ

/AUTO　自己相関スケーリング。これにより、宛先ウェーブの中心点の X 軸スケーリングが x=0 になるように強制され、中心値が正確に 1.0 になるよう、宛先ウェーブを中心点の値で除算します。

srcWaveName と *destWaveName* のデータポイント数が一致しない場合、このフラグは無視されます。

/AUTO は /C と互換性がありません。

/C　循環相関。（以下の「互換性に関する注意事項」を参照）。自己相関（*srcWaveName* と *destWaveName* が同じ場合）には /C を使わないでください。

/NODC　相関を計算する前に、ソースウェーブと宛先ウェーブから平均値を除去します。平均値を除去すると、正規化されていない自己共分散または交差共分散が得られます。

「DC」は「直流（direct current）」の略語であり、信号のうち平均値が変化しない成分を指す電子工学用語です。

詳細

注記
単一値の相関係数を計算するには、StatsCorrelation 関数を使います。この関数は、長さが同じである 2 つのウェーブのピアソンの相関係数を返します。

相関の種類によっては、宛先ウェーブの長さが増加する場合があります。
srcWaveName は、宛先ウェーブとしても指定されていない限り、変更されません。

ソースウェーブが実数である場合、各宛先ウェーブも実数でなければならず、ソースウェーブが複素数ウェーブである場合、各宛先ウェーブも複素数ウェーブでなければなりません。
倍精度と単精度のウェーブは自由に混在させることができ、計算はより高い精度で行われます。

線形相関式は次のとおりです：

$$destWaveOut[p] = \sum_{m=0}^{N-1} srcWave[m] \cdot destWaveIn[p+m]$$

ここで、N は *destWaveIn* と *srcWave* のうち、より長い方のポイント数です。

循環相関の場合、インデックス [*p* + *m*] が [0, numpts(*destWaveIn*) - 1] の範囲を超えるときは、その値がループバックされます。
線形相関の場合、*p* + *m* が範囲を超えるときは、*destWaveIn* [*p* + *m*] の値が 0 に置き換えられます。
m が numpts(*srcWave*) - 1 を超えるときは、*srcWave* [*m*] の代わりに 0 が使われます。

これを、線形畳み込み方程式が以下のようになる畳み込み演算と比較すると：

$$destWaveOut[p] = \sum_{m=0}^{N-1} destWaveIn[m] \cdot srcWave[p-m]$$

ご覧の通り、主な違いは、相関処理の場合、ソースウェーブをシフトして宛先ウェーブと合成する前に反転させないという点にあります。

Correlate コマンドは多次元に対応していません。
詳細は、ヘルプ Multidimensional Waves、特に Analysis on Multidimensional Waves を参照してください。

互換性に関する注意事項

Igor Pro 5 以前のバージョンでは、Correlate/C による結果の拡大縮小および回転が不適切に行われていました（結果が左方向に 1 回転していることが多く、X 軸の拡大縮小係数がすべて負の値になっていました）。

現在、宛先ウェーブの X 軸スケーリングは変更されず、結果も回転されません。
以前の挙動に依存する古いプロシージャとの互換性を確保するために、root:V_oldCorrelationScaling=1 を設定することで、以前の挙動を強制的に適用することができます。

Igor Pro のすべてのバージョンで同一の Correlate/C の結果を得るためのより良い方法は、次のコードを使うことです。
このコードは、どのバージョンの Igor Pro で実行しても、x=0 が常に destWave の最初のポイントになるように結果を回転させます（現時点では、回転処理を行わないため、何も変更されず、実行も極めて高速です）。

```
Correlate/C srcWave, destWave
Variable pointAtXEqualZero= x2pnt(destWave,0)      // Igor Pro 5 に対しては 0
Rotate -pointAtXEqualZero,destWave
SetScale/P x, 0, DimDelta(destWave,0), "", destWave
```

応用

相関の一般的な用途として、 2 つの入力信号を互いにずらしながら、それらの類似度を測定することが挙げられます。

多くの場合、 2 つの入力が最も類似している最大値において、相関結果を 1.0 に正規化することが望ましいです。
destWaveOut を正規化するには、入力ウェーブの RMS 値と各ウェーブのデータポイント数を算出します。

```
WaveStats/Q srcWave
Variable srcRMS= V_rms
Variable srcLen= numpnts(srcWave)

WaveStats/Q destWave
Variable destRMS= V_rms
Variable destLen= numpnts(destWave)

Correlate srcWave, destWave        // destWave を上書き

// 最大値が 1.0 になるように正規化
destWave /= (srcRMS * sqrt(srcLen) * destRMS * sqrt(destLen))
```

もう 1 つの一般的な用途として、自己相関（*srcWaveName* と *destWaveName* が同じ場合）を用いてパワースペクトル密度を算出することが挙げられます。
この場合は、より多くのオプションが用意されている DSPPeriodogram コマンドを使うことを推奨します。

参照

Convolve、Correlation、MatrixOp、StatsCorrelation、StatsCircularCorrelationTest、StatsLinearCorrelationTest、DSPPeriodogram コマンド

参考文献

自己相関とパワースペクトル密度（PSD）に関する説明は、William H. Press et.al. 著 Numerical Recipes in C（Cambridge University Press、1988 年）の第 12 章に記載されています。

WaveMetrics 社は、ウィンドウ処理やセグメンテーションなどのオプションを使って PSD を計算する「Igor テクニカルノート 006：DSP サポートマクロ」を提供しています。
「Technical Notes」フォルダーを参照してください。
そこで解説されている手法の一部は、WaveMetrics Procedures:Analysis フォルダー内に Igor プロシージャファイルとして用意されています。

Wikipedia: http://en.wikipedia.org/wiki/Correlation
Wikipedia: http://en.wikipedia.org/wiki/Cross_covariance
Wikipedia: http://en.wikipedia.org/wiki/Autocorrelation_function

デモ

メニュー File→Example Experiments→Analysis→PSD Demo

CreateAliasShortcut [/D/I[=i]/M=*messageStr* /O/P=*pathName* /S=*saveMessageStr* /Z[=*z*]] [*targetFileDirStr*] [*as shortcutFileStr*]

CreateAliasShortcut コマンドは、ディスク上にショートカットファイルを作成します。
ショートカットは、ファイルまたはフォルダーのいずれかを指定できます。
ショートカットが指すファイルまたはフォルダーは、ショートカットの「ターゲット」と呼ばれます。

Igor Pro 10 以前の Igor Pro Macintosh 版では、Windows のショートカットに相当するエイリアスがサポートされていました。

パラメーター

targetFileDirStr には、ショートカットを作成するファイルまたはフォルダーの完全なパス、/P=*pathName* で指定されたフォルダーを基準とした相対パス（一部のみ）、あるいは *pathName* で指定されたフォルダー内のファイルまたはフォルダー名を指定できます。

targetFileDirStr と *pathName* からファイルまたはフォルダーの場所を特定できない場合、ターゲットファイルを指定できるダイアログが表示されます。
代わりに、/D オプションを使って、ショートカットのターゲットとしてフォルダーを選択してください。

shortcutFileStr には、作成されるショートカットファイルへのフルパス、*pathName* で指定されたフォルダーを基準とした相対パス（指定されている場合）、または *pathName* で指定されたフォルダー内のファイル名を指定できます。
Windows 形式と Macintosh 形式のファイルパスの両方がサポートされています。

shortcutFileStr と *pathName* からショートカットファイルの場所を特定できない場合、ファイルを作成できる File Save ダイアログが表示されます。

targetFileDirStr または *shortcutFileStr* にフルパスまたは部分パスを使う場合は、パスの構成方法の詳細についてヘルプ Path Separators を参照してください。

フォルダーパスは、単一のパス区切り文字で終わってはいけません。
MoveFolder コマンドの「詳細」セクションを参照してください。

フラグ

/D　*targetFileDirStr* が完全に指定されていない場合は、Open File ダイアログではなく、Select Folder ダイアログを使います。

/I[=*i*]　ユーザーとのインタラクティブ性をどの程度にするかを指定します。

- *i*=0：　*targetFileDirStr* または *shortcutFileStr* のいずれかが指定されていない場合、またはターゲットファイルが存在しない場合にのみ、対話モードになります（/I が指定されていない場合と同じです）。
- *i*=1：　*targetFileDirStr* が完全に指定されており、対象ファイルが存在する場合でも、対話モードで実行されます。
- *i*=2：　*targetFileDirStr* が指定されている場合でも、対話モードになります。
- *i*=3：　*targetFileDirStr* が指定され、かつ対象ファイルが存在する場合でも、対話モードで実行されます。/I オプションのみを指定した場合と同じ動作です。

/M=*messageStr*
Open File または Select Folder ダイアログに表示されるプロンプトメッセージを指定します。/S が指定されていない場合、*messageStr* は Open File（または Select Folder）ダイアログと Save File ダイアログで使われます。

/O　既存のファイルがあれば、ショートカットファイルで上書きします。

/P=*pathName*　ファイルを検索するフォルダーを指定します。*pathName* は、既存のシンボリックパスの名前です。

/S=*saveMessageStr*
ショートカットファイルを作成する時、Save File ダイアログに表示されるプロンプトメッセージを指定します。

/Z[=*z*]　プロシージャが、存在しないファイルやフォルダーのショートカットを作成しようとした場合でも、プロシージャの実行が中断されないようにします。実行を中断させるのではなく、プロシージャ内でこのケースを処理したい場合は、/Z オプションを使ってください。

- /Z=0：　/Z が指定されていないときと同じです。
- /Z=1：　ファイルまたはフォルダーが存在する場合にのみ、そのショートカットを作成します。/Z のみを指定した場合、/Z=1 を指定した場合と同じ効果があります。

/Z=2： ファイルまたはフォルダーが存在する場合にのみそのショートカットを作成し、存在しない場合はダイアログを表示します。

変数

CreateAliasShortcut コマンドでは、以下の変数に情報が返されます。

V_Flag 0： ショートカットファイルを作成しました。

1： ユーザーが Open File、Select Folder、Save File のいずれかのダイアログをキャンセルしました。

その他： ターゲットファイルが存在しないなど、エラーが発生しました。

S_path 作成されたショートカットファイルへのフルパス。エラーが発生した場合、またはユーザーがキャンセルした場合は、空の文字列になります。

S_fileName 対象のファイルまたはフォルダーのフルパス。エラーが発生した場合、またはユーザーがキャンセルした場合は、空の文字列になります。

例

```
// デスクトップに現在のエクスペリメントへのショートカットを作成する
// エクスペリメント用ファイルの拡張子は .pxp
String target= Igorinfo(1)+".pxp"
// Windows 形式の完全なファイルパス
CreateAliasShortcut/O/P=home target as "C:\\Users\\myName\\Desktop\\"+target
// ターゲットへのフルパス（Mac 形式のファイルパス）
Print S_fileName
// 作成されたショートカットファイルへのフルパス（Mac 形式のファイルパス）
Print S_path

// SpecialDirPath から返される Mac 形式のファイルパスを使った同じ例
// 末尾に「:」区切り記号が付いた Mac のファイルパス
String desktopPath = SpecialDirPath("Desktop", 0, 0, 0)
String shortcutPath = desktopPath+target
CreateAliasShortcut/O/P=home target as shortcutPath
// ターゲットへのフルパス（Mac 形式のファイルパス）
Print S_fileName
// 作成されたショートカットファイルへのフルパス（Mac 形式のファイルパス）
Print S_path

// Windows エクスプローラーでデスクトップを表示する
NewPath/O/Q Desktop, desktopPath
PathInfo/SHOW Desktop
```

参照

Open、MoveFile、DeleteFile、GetFileFolderInfo、IgorInfo、Symbolic Paths、ParseFilePath コマンド

CreateBrowser [/M] [*keyword=value* [, *keyword=value* ...]]

CreateBrowser コマンドは、Data Browser ウィンドウを作成します。
このコマンドには 2 つの用途があります。

- 通常の Data Browser を表示する

・ プロシージャの入力として使うデータオブジェクトを 1 つ以上ユーザーに選択してもらうために、モーダルデータブラウザーを表示する

通常、通常の Data Browser をいつ表示するかはユーザーに任せるため、CreateBrowser コマンドの主な目的は、プロシージャの実行中にユーザーインターフェース要素として使うモーダルデータブラウザーを作成することです。

モーダルデータブラウザーは、通常の Data Browser とは別のものであり、通常は入力を求めるために使っている間だけ、短時間だけ存在します。

CreateBrowser コマンドは、prompt キーワードを使う場合、または /M フラグを使う場合、モーダルデータブラウザーを対象とします。
/M フラグの目的は、モーダルデータブラウザーを表示する前に、CreateBrowser の実行に続いて 1 つ以上の ModifyBrowser コマンドを実行できるようにすることです。
これについては、例の中の SelectWavesUsingDataBrowser() 関数で説明しています。

CreateBrowser を、キーワード prompt やフラグ /M を指定せずに呼び出すと、通常の Data Browser がまだ存在しない場合はそれが作成され、表示されます。
この場合、すべての CreateBrowser キーワードは無視されます。

ユーザーは、モーダルデータブラウザーを使って、現在のデータフォルダーを変更することができます。
これを許可したくない場合は、以下の例の中の SelectDataObjects() 関数に示すように、現在のデータフォルダーを保存、復元することができます。

executeMode キーワードと関連するキーワードを使うことで、ユーザーが OK をクリックした時に、モーダルデータブラウザーにコマンドを実行させるよう指示することができます。
これについては、以下の例の中の SelectWavesAndDisplayInGraph() 関数で説明しています。

モーダルデータブラウザーは、以下に説明する通り、ユーザーの選択に基づいて出力変数を作成します。

フラグ

/M　まだモーダルブラウザーが作成されていない場合、モーダルブラウザーを作成します。/M フラグを使う場合、許可されるキーワードは prompt のみです。モーダルブラウザーは作成されますが、ModifyBrowser/M showModalBrowser が呼び出されるまで表示されません。以下の例の SelectWavesUsingDataBrowser() を参照してください。

キーワード

prompt=*string*　*string* は、モーダルブラウザのプロンプトとして使う任意の文字列です。/M フラグが指定されていない場合、プロンプトキーワードが含まれていると、コマンドの実行時にモーダルブラウザーが表示されます。

以下のキーワードは、/M フラグを指定せずに prompt キーワードを使って即時にモーダルデータブラウザーを作成する時に使います。
/M が指定されている場合は、これらのキーワードは使用できません。
通常のデータブラウザを表示するために CreateBrowser が呼び出された場合（/M と prompt が省略されている場合）、これらのキーワードは無視されます。

command1=*cmdStr*

cmdStr には、ユーザーが OK ボタンを押した時に、モーダルデータブラウザーに実行させたいコマンドが格納されています。これは、選択時に実行されるメインのコマンドです。以下の例の SelectWavesAndDisplayInGraph() を参照してください。

詳細は、ヘルプ Data Browser Command String Limitations を参照してください。

command2=*cmdStr*

cmdStr には、ユーザーが OK ボタンを押した時に、モーダルデータブラウザーに実行させたい二次コマンドが格納されます。これは、選択された項目のリストがコマンドの最大長を

超え、かつ executeMode=2 の場合にのみ実行されます。command1 が空の場合、command2 は無視されます。以下の例の SelectWavesAndDisplayInGraph() を参照してください。

詳細は、ヘルプ Data Browser Command String Limitations を参照してください。

executeMode=*mode*

mode=1　選択された各項目に対して、command1 で指定されたコマンドを 1 つずつ実行します。これがデフォルトの設定です。

mode=2　可能であれば、選択された項目のリスト全体に対してコマンドを一度に実行します。まず、command1 で指定されたコマンドが実行されます。選択された項目のリスト全体がコマンドの最大長を超える場合、残りの選択項目に対して command2 で指定されたコマンドが実行されます。以下の例の SelectWavesAndDisplayInGraph() の例を参照してください。

詳細は、ヘルプ Data Browser Command String Limitations を参照してください。

exitCommand=*cmdStr*

cmdStr には、モーダルデータブラウザーが閉じられる前に実行させたいコマンドが格納されています。このコマンドは選択範囲に対して作用するものではなく、command1 の文字列が空の場合にのみ実行されます。

expandAll　すべてのデータフォルダを完全に展開して表示します。

select=*findStr*　*findStr* に一致するすべての項目名を選択します。

一致の判定は、Data Browser で有効になっている大文字小文字の区別と単語単位の検索設定によって異なります。ModifyBrowser の case と wholeWord キーワードを参照してください。

showWaves=*mode*

mode=1 に設定すると、モーダルデータブラウザーのリストにウェーブが表示されます。0 に設定すると、ウェーブは非表示になります。

showVars=*mode*

mode=1 に設定すると、モーダルデータブラウザーのリストに数値変数が表示され、0 に設定すると非表示になります。

showStrs=mode

mode=1 に設定すると、モーダルデータブラウザーのリストに文字列変数が表示され、0 に設定すると非表示になります。

詳細

モーダルデータブラウザーを使っている間は、通常のメニューにはアクセスできません。
ただし、以下のキーボードショートカットは引き続き使用できます：

アクション	**Windows**
すべてを選択	Ctrl-A
検索	Ctrl-F
検索に選択を使用	なし
再検索	Ctrl-G

出力変数

CreateBrowser は、V_Flag と S_BrowserList という 2 つの出力変数を作成します。
これらの出力の値は、ユーザーが「OK」または「キャンセル」ボタンをクリックしてモーダルデータブラウザーを閉じた後にのみ意味を持ちます。

プロシージャから CreateBrowser が実行された場合、これらの出力変数はローカル変数となります。コマンドラインからモーダルデータブラウザーを作成する理由はありませんが、もし作成した場合、出力変数はグローバル変数となり、CreateBrowser コマンドが呼び出された時点の現在のデータフォルダー内に作成されます。

V_flag　　ユーザーが「OK」ボタンをクリックした場合は、1 に設定します。

ユーザーが「キャンセル」ボタンまたはウィンドウの閉じるボタンをクリックした場合は、0 に設定します。

S_BrowserList　　選択された各項目の完全なパス（必要に応じて引用符で囲まれたもの）を、セミコロンで区切ったリストとして含みます。リスト内の項目の順序は未定義です。

例

```
// モーダルデータブラウザーを表示し、ユーザーが選択した項目の文字列リストを返す
Function/S SelectDataObjects()
    DFREF saveDF = GetDataFolderDFR            // 事前に現在のデータフォルダーを保存
    String promptStr = "Select Items"
    CreateBrowser prompt=promptStr, showWaves=1, showVars=0, showStrs=0
    SetDataFolder saveDF                       // 現在のデータフォルダーを復元

    if (V_Flag == 0)
        return ""                              // ユーザーがキャンセル
    endif
    return S_BrowserList
End

// モーダルデータブラウザーを表示し、ユーザーが選択したウェーブをグラフに表示する
Function SelectWavesAndDisplayInGraph()
    String cmd1 = "Display %s"
    String cmd2 = "AppendToGraph %s"
    String promptStr="Select Waves for Graph"
    CreateBrowser prompt=promptStr, executeMode=2, command1=cmd1, command2=cmd2
End

// この例では、モーダルデータブラウザーを表示する前に、そのブラウザーに対して
// ModifyBrowser を呼び出せるようにするための /M フラグの使用方法を示す
Function/S SelectWavesUsingDataBrowser()
    // モーダルデータブラウザーを作成するが、表示はしない
    CreateBrowser/M prompt="Select a wave:"

    // モーダルデータブラウザーにはウェーブを表示し、変数は表示しない
    ModifyBrowser/M showWaves=1, showVars=0, showStrs=0

    // モーダルデータブラウザーを名前順で並べ替えるように設定する
    ModifyBrowser/M sort=1, showWaves=1, showVars=0, showStrs=0

    // モーダルデータブラウザーの情報ペインとプロットペインを非表示にする
    ModifyBrowser/M showInfo=0, showPlot=0

    // モーダルデータブラウザーを表示し、ユーザーが選択できるようにする
    ModifyBrowser/M showModalBrowser

    if (V_Flag == 0)
        return ""                              // ユーザーがキャンセル
    endif
```

```
    return S_BrowserList
End
```

参照

ModifyBrowser、GetBrowserSelection、GetBrowserLine コマンド
ヘルプ The Data Browser

Cross [/DEST=*destWave* /FREE /T /Z] *vectorA*, *vectorB* [, *vectorC*]

Cross コマンドは、*vectorA* × *vectorB* と *vectorA* × (*vectorB* × *vectorC*) の外積を計算します。
各ベクトルは、3行からなる 1D の実数型ウェーブです。
結果を、現在のデータフォルダー内のウェーブ W_Cross に保存します。

フラグ

/DEST=*destWave*
destWave で指定されたウェーブに、外積を格納します。

宛先ウェーブが存在する場合、そのウェーブは上書きされます。

宛先ウェーブは、入力ウェーブとは異なっていなければなりません。

このコマンドは、ユーザー定義関数内で呼び出された場合、宛先ウェーブに対するウェーブ参照を作成します。詳細は、ヘルプ Automatic Creation of WAVE References を参照してください。

/DEST を省略した場合、この操作の結果は現在のデータフォルダー内のウェーブ W_Cross に保存されます。

Igor Pro 7 以降が必要です。

/FREE　/DEST オプションと併用すると、宛先ウェーブはフリーウェーブとして作成されます。
フリーウェーブの詳細は、ヘルプ Free Waves を参照してください。

/FREE は、ユーザー定義関数でのみ使用できます。

Igor Pro 7 以降が必要です。

/T　W_Cross では、出力を列ではなく行として保存します。

/Z　入力が不適切な場合でも、エラーは発生しません。

CtrlBackground [start[=*startTicks*], *period=deltaTicks*, *dialogsOK=d*, *noBurst=n*, *stop*]

CtrlBackground コマンドは、名前のないバックグラウンドタスクをコントロールします。

CtrlBackground は、名前が付けられていないバックグラウンドタスクでのみ機能します。
新しいコードでは、代わりに名前付きバックグラウンドタスクを使う必要があります。
詳細は、ヘルプ Background Tasks を参照してください。

パラメーター

period=*deltaTicks*
バックグラウンドタスクの呼び出し間隔として、経過しなければならない最小ティック数を設定します。

dialogsOK=1 または 0

1 の場合、ダイアログが表示されている間も、タスクの実行が許可されます。タスクの挙動が適切でない限り、これによりクラッシュが発生する可能性があります。

noBurst=1 または 0

通常（または noBurst=0 の場合）、遅延によって通常の実行時間を超過したときは、タスクは最大レートで呼び出されます。noBurst=1 を指定すると、このバーストキャッチアップモードが抑制されます。

start[=*startTicks*]

ティック数が *startTicks* に達した時点で、（SetBackground で指定された）バックグラウンドタスクを開始します。*startTicks* を省略した場合、タスクは直ちに開始されます。

stop　バックグラウンドタスクを停止します。

参照

SetBackground、KillBackground、CtrlNamedBackground、BackgroundInfo コマンド
ヘルプ Background Tasks

CtrlNamedBackground *taskName*, *keyword=value* [, *keyword=value* ...]

CtrlNamedBackground コマンドは、名前付きバックグラウンドタスクを作成およびコントロールします。

バックグラウンドタスクを扱う前に、ヘルプ Background Tasks を確認することを推奨します。

パラメーター

taskName　*taskName* はバックグラウンドタスクの名前、またはすべての指定されたバックグラウンドタスクをコントロールする場合は _all_ を指定します。バックグラウンドタスク名としては、有効な標準 Igor オブジェクト名であればどれでも使用できます。

burst [=*b*]　バーストキャッチアップモードを有効にします（デフォルトではオフ、b=0）。オン（b=1）の場合、遅延によって通常の実行時間を超過したときは、タスクが最大レートで呼び出されます。

dialogsOK [=*d*]

dialogsOK=1 を指定すると、ダイアログウィンドウがアクティブな状態でもバックグラウンドタスクの実行が可能になります。デフォルトでは、dialogsOK=0 が有効になっています。詳細は、ヘルプ Background Tasks and Dialogs を参照してください。

dialogsOK を 1 に設定すると、バックグラウンドタスクがダイアログの想定外の動作を行った場合に、クラッシュが発生する可能性があります。詳細は、ヘルプ Background Tasks and Dialogs を参照してください。

kill [=*k*]　タスクメモリを停止して解放し、再利用できるようにする（k=1；デフォルト）か、継続します（k=0）。

noEvents=*mode*

バックグラウンドタスク関数の実行中に、イベント処理をコントロールします。
noEvents キーワードは、Igor Pro 8.04 で追加されました。

バックグラウンドタスク関数が利用可能なプロセッサ時間の大部分を占有している場合、コマンドラインやその他の場所で入力を行うのが困難になることがあります。
これは、通常、ユーザー定義関数の実行中に到着したイベントを破棄してしまうため、イベントが失われてしまうからです。
これはデフォルトの動作モードであり、noEvents=0 に相当します。

noEvents=1 を指定すると、バックグラウンドタスク関数が返るまでイベント処理を延期するため、ユーザーインターフェースの信頼性が向上します。noEvents=1 を指定した場合、ユーザーによる中止キーの組み合わせを使って、異常な動作をしているバックグラウンドタスク関数を中止することはできなくなります。

noEvents キーワードは、名前のないバックグラウンドタスクを含め、すべてのバックグラウンドタスクに適用されます。taskName パラメータには _all_ を指定してください。

起動時、モードはデフォルトで 0 に設定されます。noEvents を使って設定されたモードは、Igor Pro が終了するまで有効です。

period=*deltaTicks*
: バックグラウンドタスクの呼び出し間隔として、経過しなければならない最小ティック数（*deltaTicks*）を設定します。*deltaTicks* は整数に切り捨てられ、0 より大きい値にクリップされます。詳細は、ヘルプ Background Task Period を参照してください。

proc=*funcName*
: バックグラウンドユーザー関数の名前を指定します（詳細は後述）。

start [=*startTicks*]
: ティック数が *startTicks* に達した時点で開始されます。*startTicks* を指定しない場合、タスクは直ちに開始されます。

status
: S_info 文字列変数にバックグラウンドタスクの情報を返します。

stop [=*s*]
: バックグラウンドタスクを停止します（*s*=1、デフォルト）または継続します（*s*=0）。

詳細

proc キーワードを介して指定するユーザー関数は、以下の形式でなければなりません。

```
Function myFunc(s)
    STRUCT WMBackgroundStruct &s
    ...
```

WMBackgroundStruct のメンバーは以下の通りです：

メンバー	**説明**
char name[MAX_OBJ_NAME+1]	バックグラウンドタスクの名前
uint32 curRunTicks	タスクが呼び出された時点でのティック数
int32 started	CtrlNamedBackground start コマンドが発行されたときに TRUE。この値をクリアしたり、任意の値に設定したりすることができる
uint32 nextRunTicks	次回の実行に向けた事前計算値。ユーザー関数によってこれが変更される可能性がある

また、構造体の最初の要素が WMBackgroundStruct と一致する場合、あるいは（できれば）最初の要素が WMBackgroundStruct のインスタンスである場合に限り、ユーザー定義の STRUCT を受け取るユーザー関数を指定することもできます。
追加のフィールドをいつ初期化するかは、started フィールドを使って決定してください。
構造体には、String、WAVE、NVAR、DFREF、または構造体自体の一部ではないメモリを参照するその他のフィールドを含めることはできません。

WMBackgroundStruct のインスタンスを含むのではなく、最初のフィールドと一致するユーザー定義構造体を指定した場合、将来、組み込み構造体のサイズが変更されると、その関数は失敗します。
MAX_OBJ_NAME の値は 255 ですが、これも変更される可能性があります。
Igor Pro 8.0 で、31 から 255 に変更されました。

ユーザーの関数は、停止させたい場合を除き、0 を返す必要があります。
停止させたい場合は、1 を返す必要があります。

バックグラウンド関数内で CtrlNamedBackground を呼び出すことができます。
必要に応じて、別の関数に切り替えることも可能です。

status キーワードを使って、S_info 変数を通じてバックグラウンドタスクの情報を取得します。
その形式は次のとおりです：

```
NAME:name;PROC:fname;RUN:r;PERIOD:p;NEXT:n;QUIT:q;FUNCERR:e;
```

S_info を解析する際は、キーと値のペアの数やその順序に依存しないでください。
RUN、QUIT、FUNCERR の値は 1 または 0 であり、NEXT はタスクが次に実行されるまでのティック数です。
関数が 0 以外の値を返した場合、QUIT は 1 に設定され、何らかの理由で関数が使用できなかった場合は FUNCERR が 1 に設定されます。

参照

例については、ヘルプ Background Tasks を参照してください。

デモ

メニュー File→Example Experiments→Programming→Background Task Demo

CtrlFIFO *FIFOName*, [*deltaT=dt*, *note=noteStr*, *file=oRefNum*, *rfile=rRefNum*, *rdfile=rRefNum*, *doffset=dataOffset*, *dsize=dataSize*, *swap*, *size=s*, *start*, *stop*, *close*, *flush*]

CtrlFIFO コマンドは、指定された FIFO のさまざまな側面をコントロールします。

パラメーター

close　FIFO の出力ファイルまたは確認ファイル（存在する場合）を閉じます。

deltaT=*dt*　データ取得レートを記録します。

doffset=*dataOffset*
　rdfile でのみ使われます。データに対するオフセットです。指定がない場合は、オフセットは 0 となります。

dsize=*dataSize*　rdfile でのみ使われます。データのサイズ（バイト単位）です。指定しない場合、データサイズはファイルの残りの部分とみなされます。この仮定が当てはまらない場合、予期しない結果が生じる可能性があります。

flush　FIFO 内の新しいデータは、直ちにディスクに書き出されます。

file=*oRefNum*　FIFO の出力ファイルのファイル参照番号です。この参照番号は、ファイルの作成に使った Open コマンドから取得します。

note=*noteStr*　ファイルヘッダーにメモの文字列を格納します。最大 255 バイトまでです。

rfile=*rRefNum*　FIFO のレビューファイルの参照番号です。既存のデータをレビューするために FIFO を使う場合、レビューファイルを使います。この参照番号は、ファイルを開く時に使う Open/R コマンドから取得します。ファイルは、ヘッダーとデータが統合された形式、あるいはヘッダーに生データを含むファイル名が記載された分割形式のいずれかです。

rdfile=*rRefNum*

rfile と同様ですが、生データのレビュー用です（Open/R コマンドを使用）。チャネルデータは、ファイル内の生データと一致している必要があります。ファイルの先頭からデータの先頭までのオフセットは、同じコマンド内で doffset を指定することで指定できます。データがファイルの末尾まで広がっていない場合は、同じコマンド内で dsize を指定して、データのバイト数を指定できます。

size=*s*　FIFO 内のチャンク数を設定します。デフォルトは 10000 です。1 つのデータチャンクは、FIFO の各チャンネルからの 1 つのデータポイントで構成されます。

start　FIFO ヘッダー内の日時情報フィールドを設定し、そのヘッダーを出力ファイルに書き込み、FIFO をアクティブとしてマークすることで、FIFO の実行を開始します。

stop　データをディスクに書き出し、FIFO を非アクティブとしてマークすることで、FIFO を停止します。

swap　rdfile でのみ使われます。生データファイルを読み込む時にバイト順の反転が必要であることを示します。これは、Mac で実行している場合に PC からのバイナリファイルを読み込む場合、あるいはその逆の場合に該当します。

詳細

開始コマンドが発行されると、FIFO は停止コマンド以外のコマンドを受け付けなくなります。

FIFO のデータにアクセスするには（チャートコントロールを使う場合も、FIFO2Wave コマンドを使う場合も）、FIFO が有効な状態である必要があります。NewFIFO を使って FIFO を作成すると、初期状態では無効です。CtrlFIFO コマンドを介して開始コマンドを発行すると、有効になります。CtrlFIFO を使って FIFO のパラメータを変更するまでは、有効な状態が維持されます。

参照

Open コマンド
ヘルプ FIFOs and Charts