CONTENTS

コマンド

AutoPositionWindow [/E][/M=*m*][/R=*relWindow*] [*windowName*]

AutoPositionWindow コマンドは、*windowName* で指定されたウィンドウを、同じ種類の直下のウィンドウ、または /R フラグで指定されたウィンドウを基準として配置します。
windowName が指定されていない場合、AutoPositionWindow は対象のウィンドウに対して実行されます。

フラグ

/E　モニターの画面全体が使われます。そうでない場合は、コマンドウィンドウ用のスペースが確保されます。

/M=*m*　配置方法を指定します。

m=0：可能であれば、*windowName* を他のウィンドウの右側に配置します。スペースがない場合は、*windowName* を他のウィンドウのすぐ下、かつ表示領域の左端に配置します。それが不可能な場合は、配置位置は変更されません。

m=1：可能であれば、*windowName* を左端に揃えた状態で、他のウィンドウのすぐ下に配置します。スペースがない場合は、*windowName* を底辺に揃えた状態で、他のウィンドウのすぐ右側に配置します。どちらも不可能な場合は、*windowName* を画面の右下隅に配置します。

/R=*relWindow*　*windowName* を *relWindow* を基準として配置します。

BackgroundInfo

BackgroundInfo コマンドは、現在の名前なしバックグラウンドタスクに関する情報を返します。

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

詳細

情報は以下の変数を通じて返されます：

V_flag　0：バックグラウンドタスクが定義されていない。
1：バックグラウンドタスクは定義されているが、実行されていない（アイドル状態）。
2：バックグラウンドタスクが定義され、実行中。

V_period　CtrlBackground コマンドで設定された deltaTicks の値。これは、バックグラウンドタスクが実行される間隔。

V_nextRun　タスクが次に実行される時刻を表す値。タスクが再実行される予定がない場合は 0。

S_value　SetBackground コマンドによって設定された、バックグラウンドタスクが実行する数値式の本文。

参照

SetBackground、CtrlBackground、KillBackground、ticks コマンド
ヘルプ Background Tasks

Beep

Beep コマンドは、システムからビープ音を鳴らします。

BezierToPolygon [/DSTX=*destXWave* /DSTY=*dstYWave* /FREE /NSEG=*nseg*]
bezXWave, *bezYWave*

BezierToPolygon コマンドは、*bezXWave* と *bezYWave* で定義されたベジエ曲線を近似するウェーブの XY ペアを生成します。

BezierToPolygon コマンドは、Igor Pro 9.0 で追加されました。

フラグ

/DSTX=*destX* 作成または上書きする X 宛先ウェーブを指定します。/DSTX を省略した場合、*destX* のデフォルト値は W_PolyX になります。

/DSTY=*destY* 作成または上書きする Y 宛先ウェーブを指定します。/DSTY を省略した場合、*destY* のデフォルト値は W_PolyY になります。

/FREE 出力ウェーブを自由ウェーブとして生成します（ヘルプ Free Waves を参照）。

/FREE は関数内でのみ使用できます。/DSTX または /DSTY を使う場合、指定するパラメーターは単純な名前か、有効なウェーブ参照のいずれかでなければなりません。

/NSEG=*nseg* 各ベジェセグメントの描画に使うセグメント数を、1 から 500 の間で指定します。デフォルトの 20 で通常は十分です。

詳細

ベジエウェーブ *bezXWave* と *bezYWave* は、長さと型が同じ 1 次元の実数型浮動小数点ウェーブでなければなりません。

各ベジェ曲線は、少なくとも 1 つのセグメントからなり、そのセグメントは 4 組の XY 座標で構成されます。n 個のセグメントからなるベジエ曲線は、1 + n × 3 組の XY 座標で構成されます。

bezXWave と *bezYWave* には、ベジェセグメント間の領域に NaN 値が含まれる場合がありますが、セグメント内部には含まれません。
入力ウェーブ内のデータがこれらの要件を満たしていない場合、BezierToPolygon は実行時にエラーを発生させます。
入力ウェーブ内の NaN 値は、出力ポリゴンウェーブにコピーされます。

/DSTX を省略した場合、出力ポリゴンの X データは現在のデータフォルダー内の W_PolyX に書き込まれます。
/DSTY を省略した場合、出力ポリゴンの Y データは現在のデータフォルダー内の W_PolyY に書き込まれます。
出力ウェーブは、*bezXWave* と *bezYWave* の型に合わせて、単精度または倍精度の浮動小数点ウェーブとして作成またはリサイズされます。

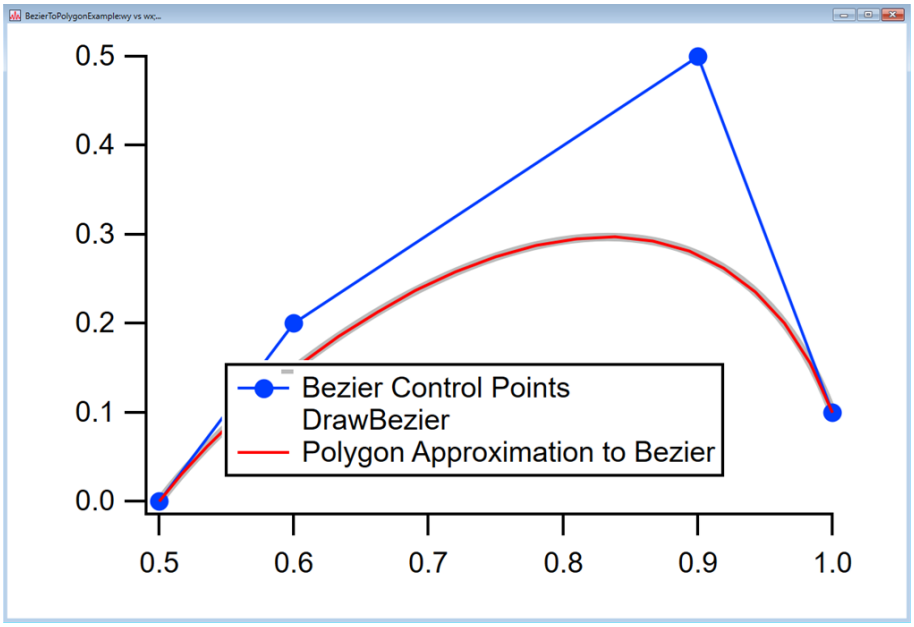
例

<プロシージャウィンドウ>

```
Function DemoBezierToPolygon()
    Make/O wx={0.5, 0.6, 0.9, 1}
    Make/O wy={0.0, 0.2, 0.5, 0.1}
    BezierToPolygon wx,wy
    Execute "BezierToPolygonExample()"
End

Window BezierToPolygonExample() : Graph
    PauseUpdate; Silent 1
    Display /W=(237,45,1419,669)/K=1  wy vs wx
    AppendToGraph W_PolyY vs W_PolyX
    ModifyGraph expand=-3
    ModifyGraph mode(wy)=4
    ModifyGraph marker(wy)=19
    ModifyGraph rgb(wy)=(1,16019,65535)
    Legend/C/N=text0/J/X=15.66/Y=68.34 \\s(wy)\\[1 Bezier Control Points
    AppendText \\K(48059,48059,48059)\\y+15\\L1700\\X1\\M\\K(0,0,0) DrawBezier \n\\s(W_PolyY)
Polygon Approximation to Bezier
    SetDrawLayer UserBack
    SetDrawEnv xcoord= bottom,ycoord= left,save
    SetDrawEnv linethick= 3,linefgc= (48059,48059,48059)
    DrawBezier wx[0],wy[0],1,1,wx,wy
    SetDrawLayer UserFront
EndMacro
```

<コマンドウィンドウ>

```
DemoBezierToPolygon()
```



参照

DrawBezier、DrawPoly コマンド
ヘルプ Drawing Polygons and Bezier Curves

BoxSmooth *box, srcWave, smoothedWave*

BoxSmooth コマンドは、*smoothedWave* を *srcWave* の平滑化されたコピーに置き換えます。
両方のウェーブが存在している必要があります。

BoxSmooth コマンドは、主に Igor テクニカルノート #20 とその派生版で使われます。
一般的な用途では、BoxSmooth の代わりに、より柔軟性のある Smooth コマンドを使ってください。

パラメーター

box は、各 *smoothedWave* ポイントを形成するために平均化される *srcWave* ポイントの数です。
偶数を指定した場合、その直上の奇数が使われます。

詳細

BoxSmooth は、/B フラグを指定した Smooth コマンドと同等ですが、Smooth のように結果をその場で計算しない点が異なります。
次のコマンド：

```
BoxSmooth box, srcWave, smoothedWave
```

は

```
Duplicate/O srcWave, smoothedWave
Smooth/B/DIM=-1/E=3/F=0 box, smoothedWave
```

と同等です。

BoundingBall [/F/Z] *scatterWave*

BoundingBall コマンドは、一連の散布ポイントに対してバウンディングサークルまたはバウンディングスフィアを計算します。
このコマンドは、2 列、3 列、またはそれ以上の列を持つ 2D ウェーブを受け付けます。
追加の列にあるデータは無視されます。

scatterWave が 2 つの列で構成されている場合、このコマンドでは外接円が計算されます。
それ以外の場合は、外接 3D 球体が計算されます。

パラメーター

scatterWave は 2 次元ウェーブであり、X 座標は 0 列目、Y 座標は 1 列目、Z 座標（任意）は 2 列目に格納されます。

フラグ

/F　このフラグは 3D 散布にのみ適用されます。これは、もともと Graphics Gems に掲載された Jack Ritter 氏の論文「An Efficient Bounding Sphere」のアルゴリズムを使っています。残念ながら、正確な境界球を生成するわけではありませんが、十分に大きな球が生成されます。このアルゴリズムは精度が低いものの、すべてのポイントを包含するのに十分な大きさの球を生成します。

/Z　エラーの報告はありません。

詳細

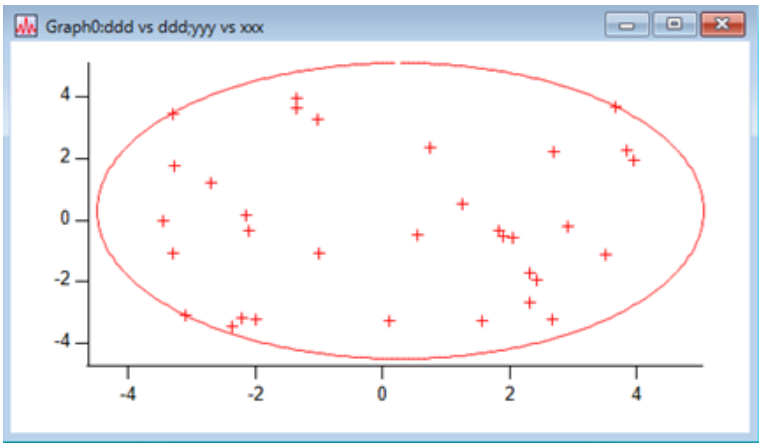
バウンディング球の中心と半径は、変数 V_CenterX、V_CenterY、V_CenterZ、V_Radius に格納されています。

/F フラグを使わない場合、このコマンドでは、平面上の外接円の中心と半径を計算するために、x,y のペアからなる 2 列のウェーブも受け付けます。

例

```
Make/N=(33,2) ddd=enoise(4)          // ランダムデータの作成
BoundingBall ddd
```

```
Display ddd[][1] vs ddd[][0]
ModifyGraph mode=3
Make/n=360 xxx,yyy
yyy=v_centerY+V_radius*cos(p*2*pi/360)
xxx=v_centerX+V_radius*sin(p*2*pi/360)
AppendToGraph yyy vs xxx
```



参考文献

Glassner, Andrew S. , (Ed.), "Graphics Gems", 864 pp., Morgan Kaufmann Publishers, 1990.

BrowseURL [/Z] *urlStr*

BrowseURL コマンドは、使っているコンピューター上の Web ブラウザーまたは FTP ブラウザーを開き、特定の Web ページを表示するか、FTP サーバーに接続するよう指示します。

BrowseURL は、コマンドが成功した場合は V_flag という名前の変数を 0 に設定し、失敗した場合は 0 以外を設定します。
これを /Z フラグと組み合わせて使うことで、エラーが発生した場合でもプロシージャの実行を継続させることができます。

パラメーター

urlStr は、閲覧する Web ページまたは FTP サーバーのディレクトリを指定します。
これは、命名規則（「http://」や 「ftp://」）、コンピューター名（例：「www.wavemetrics.com」、「ftp.wavemetrics.com」、または 「38.170.234.2」）、パス（例：「/Test/TestFile1.txt」）で構成されます。
例については以下を参照してください。

フラグ

/Z　エラーは致命的ではないときに使います。URL が不正であったり、サーバーがダウンしていたりしても、プロシージャの実行は中止されません。プロシージャ内で V_flag 変数を調べることで、転送が成功したかどうかを確認できます。V_flag は、成功した場合は 0、失敗した場合は 0 以外になります。

URL を完全に省略したり、引用符を省略したりといった構文エラーは、依然として致命的なエラーとなります。

例

```
//  Web ページのブラウズ
    String url = http://www.wavemetrics.com/News/index.html
    BrowseURL url
```

```
//  FTP サーバーのブラウズ
    String url = ftp://ftp.wavemetrics.com/pub/test
    BrowseURL url
```

参照

URLRequest コマンド

BuildMenu *menuNameStr*

BuildMenu コマンドは、ユーザーが次にメニューバーをクリックした時に、指定されたメニュー内のユーザー定義メニュー項目を再構築します。

パラメーター

menuNameStr は、メニュー名または “All” を含む文字列式です。

詳細

メニュー項目に文字列変数を使ってカスタムメニューを定義した場合は、BuildMenu を呼び出す必要があります。
文字列変数を変更した後、BuildMenu を呼び出してメニューを更新してください。

BuildMenu “All” を実行すると、すべてのメニュー項目とタイトルが再構築され、メニューバーが更新されます。

現在の実装では、*menuNameStr* が「All」でない場合、任意のユーザー定義メニューに対して BuildMenu が呼び出されると、すべてのユーザー定義メニュー項目を再構築します。

参照

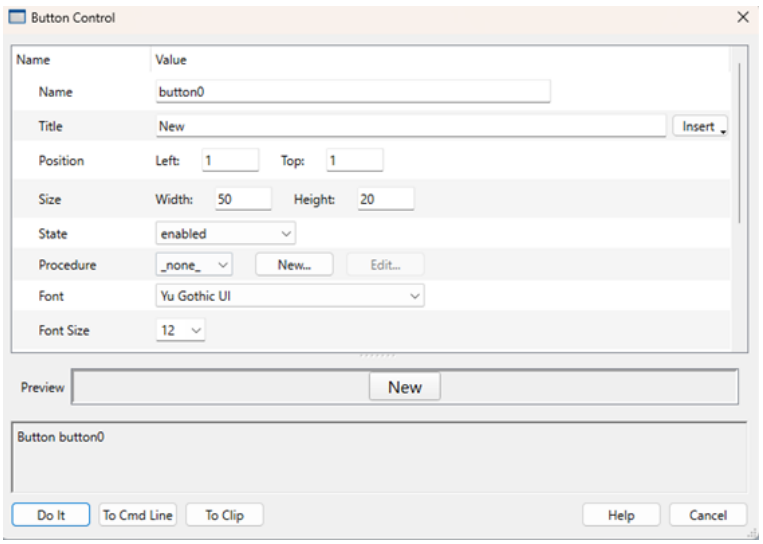
ヘルプ Dynamic Menu Items

Button [/Z] *ctrlName* [*keyword=value* [, *keyword=value* ...]]

Button コマンドは、対象ウィンドウ内の指定されたボタンコントロールを作成または変更します。

コントロールの状態やステータスに関する情報を確認するには、ControlInfo コマンドを使ってください。

メニュー Panel→Add Control→Add Button のダイアログに相当します。



パラメーター

ctrlName は、作成または変更する Button コントロールの名前です。

align=*alignment*　コントロールの配置モードを設定します。配置モードは、pos キーワードに対する *leftOrRight* パラメーターの解釈をコントロールします。
align キーワードは Igor Pro 8.0 で追加されました。

alignment=0（デフォルト）の場合、*leftOrRight* はコントロールの左端の位置を指定し、コントロールのサイズが変更されても左端の位置は固定されたままになります。

alignment=1 の場合、*leftOrRight* はコントロールの右端の位置を指定し、コントロールのサイズが変更されても右端の位置は固定されたままになります。

appearance={*kind* [, *platform*]}
コントロールの外観を設定します。*platform* は省略可能です。どちらのパラメーターも文字列ではなく、名前です。

kind=default：　外観は、DefaultGUIControls コマンドによって決定されます。

kind=native：　現在のコンピュータープラットフォームに合わせて、標準的な外観のコントロールを作成します。

kind=os9：　Igor Pro 5 の外観（Macintosh OS 9 風の操作画面）。

platform=Mac：　Igor Pro 9 またはそれ以前のバージョンを実行している Macintosh コンピューター上のコントロールの外観を変更します。

platform=Win：　Windows コンピューター上のコントロールの外観を変更します。

platform=All：　Windows コンピューターと Igor Pro 9 以前を実行している Macintosh コンピューター上のコントロールの外観を変更します。

disable=*d*　コントロールの状態を設定します。*d* はビットフィールドです。コントロールが非表示の場合、ビット 0（最下位ビット）がセットされます。コントロールが無効化されている場合、ビット 1 がセットされます。

d=0：　通常（表示）、有効

d=1：　非表示

d=2：　表示、無効。グレー表示で描画され、アクションプロシージャも無効になります。

d=3：　非表示、無効

ビットを個別に設定する方法については、ModifyControl コマンドの例を参照してください。

fColor=(*r*,*g*,*b*[,*a*])　ボタンの色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。

デフォルトのボタン色にするには、fColor=(0,0,0) を指定します。
黒いボタンにしたい場合は、fColor=(1,1,1) を使います。
デフォルトの青いボタンの外観にするには、fColor(0,0,65535) を使います。

タイトルテキストの色を設定するには、valueColor を参照してください。

focusRing=*fr*　キーボードのフォーカスを示す四角形の表示を有効または無効にします：

fr=0：　フォーカス矩形は描画されません。

fr=1：　フォーカス矩形が表示されます（デフォルト）。

font="*fontName*"　フォントを設定します。例：font="Helvetica"

fsize=*s*　フォントサイズを設定します。

fstyle=*fs*　フォントスタイルを指定します。*fs* はビット単位のパラメーターであり、各ビットは以下のようにフォントスタイルをコントロールします：

ビット 0：　Bold（太字）

ビット 1：　Italic（斜体）

ビット 2：　Underline（下線）

ビット 4：　Strikethrough（取消線）

ビット設定の詳細は、ヘルプ Setting Bit Parameters を参照してください。

guides={*left*, *hcenter*, *right*, *top*, *vcenter*, *bottom*}
レイアウトガイドへのコントロールポイントの配置をコントロールします。詳細や例は、ヘルプ Laying Out Controls in Guides Mode を参照してください。

left、*hcenter*、*right*、*top*、*vcenter*、*bottom* の各々は、適切な方向を持つレイアウトガイドの名前で置き換えられます。つまり、*left*、*hcenter*、*right* は垂直ガイドへの配置を示し、*top*、*vcenter*、*bottom* は水平ガイドへの配置を示します。配置されていないアンカーポイントを表すには、特別な名前 *kwNone* を使います。

コントロールのレイアウトに過度な制約を課すと、エラーが発生します。３つの水平または垂直のコントロールポイントのうち、レイアウトガイドに固定できるのは最大２つまでです。３つのうち１つを固定すると、レイアウトガイドが移動する時にコントロールも移動します。２つを固定すると、コントロールは移動するとともにサイズも変更されます。したがって、*kwNone* という名前は少なくとも２回出現することになります。

すべてのアンカーポイントに *kwNone* を設定すると、コントロールがレイアウトガイドから完全に切り離されます。

なお、ボタンの外観に画像を設定した場合、ボタンのサイズを変更するレイアウトガイドへの配置により、画像が歪んでしまうことがあります。

help={*helpStr*}　コントロールのヘルプを設定します。

helpStr の最大長は 1970 バイトです（Igor Pro 8 とそれ以前のバージョンでは 255 バイト）。

引用符で囲んだ文字列の中に「\r」を入れると、改行を挿入できます。

labelBack=(*r*, *g*, *b*[, *a*]) または 0
画像を使ってボタンの外観を定義する場合、コントロールの背景色を設定します（picture キーワードを参照）。

labelBack キーワードは、Igor Pro 9.0 で追加されました。

r、*g*、*b*、*a* は、RGBA 値として色と（任意の）不透明度を指定します。

透明化が機能するためには、画像自体が本質的に透明でなければなりません。例えば、PNG 画像の各ピクセルには独自の内部アルファ値が設定されており、それによって本質的に透明になることも、本質的に不透明になることも可能です。

ボタンの背景を実際に透明にしたい場合は、labelBack の色を使って画像の背景色を設定してください。ほとんどの場合、透明な白を使います：labelBack=(65535, 65535, 65535, 0)。

labelBack を省略するか、labelBack=0 を指定した場合、ボタンの背景色は、そのボタンが描画されるウィンドウの背景色になります。

noproc　ボタンをクリックしても、プロシージャが実行されないように指定します。

picture=*pict*　指定された画像を使ってボタンを描画します。この画像は、通常状態、マウスボタンが押下されている状態、無効状態におけるコントロールの外観を示す、３つのフレームが並んだものとみなされます。画像は、グローバル（インポートされた）画像、またはプロシージャ画像（ヘルプ Proc Pictures を参照）のいずれかです。

デフォルトでは、ボタンが描画される前に、その長方形は描画先のウィンドウの背景色で塗りつぶされます。labelBack キーワードを使うと、ボタンの背景色や透明度をコントロールできます。

Igor Pro 6 では、ボタンコントロールで画像を使う場合、size キーワードは無視されていました。高解像度画面向けにグラフィックのサイズ調整を容易にするため、Igor Pro 7 以降では、この場合でも size キーワードが反映されるようになりました。

pos={*leftOrRight*, *top*}
コントロールの配置モードが 0 の場合は、コントロールの左上隅の位置を、配置モードが 1 の場合は、コントロールの右上隅の位置を、コントロールパネル単位で設定します。詳細は、前述の align キーワードを参照してください。

pos+={*dx*, *dy*}　ボタンの位置をコントロールパネル単位でオフセットします。

proc=*procName*　ボタンをクリックした時に実行するプロシージャの名前を指定します。

rename=*newName*　ボタンの名前を変更します。

size={*width*, *height*}
コントロールパネル単位でボタンの幅と高さを設定します。

title=*titleStr*　ボタンのタイトルを指定された文字列式に設定します。タイトルとは、ボタンに表示されるテキストのことです。指定しない場合、タイトルは “New” になります。”” を指定すると、ボタンにはテキストが表示されません。

エスケープコードを使うと、タイトルのフォント、サイズ、スタイル、色を変更できます。詳細は、ヘルプ Annotation Escape Codes を参照してください。

userdata(*UDName*)=*UDStr*
名前なしのユーザーデータを *UDStr* に設定します。オプションの（*UDName*）を使って、作成する名前付きユーザーデータを指定します。

userdata(*UDName*)+=*UDStr*
現在の無名のユーザーデータに *UDStr* を追加します。オプションの（*UDName*）を使うと、名前付きユーザーデータに追加できます。

valueColor=(*r*, *g*, *b*[, *a*])
ボタンのテキスト（タイトル）の初期色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。デフォルトは不透明な黒です。

タイトルテキストの色をさらに変更するには、title=*titleStr* で説明しているようにエスケープシーケンスを使ってください。

win=*winName*　指定された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。指定がない場合は、最前面のグラフまたはパネルウィンドウ、あるいはそのサブウィンドウが対象となります。

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。

フラグ

/Z　　エラーを報告しません。

詳細

対象ウィンドウは、グラフまたはパネルでなければなりません。

ボタンアクションプロシージャ

Button コントロールのプロシージャでは、関数のパラメーターとして、あらかじめ定義された構造体 WMButtonAction を受け取ります。

```
Function ActionProcName(B_Struct) : ButtonControl
    STRUCT WMButtonAction &B_Struct
    ...
    return 0
End
```

「: ButtonControl」という指定により、Igor Pro はこのプロシージャを「Button Control」ダイアログの「Procedure」ポップアップメニューに含めるようになります。

WMButtonAction 構造体の詳細は、WMButtonAction のセクションを参照してください。

現在、戻り値は使われていませんが、アクションプロシージャは常に 0 を返すようにしてください。

古いコードの中に、次のような古い形式のボタンアクションプロシージャが見られるかもしれません。

```
Function procName (ctrlName) : ButtonControl
    String ctrlName
    ...
    return 0
End
```

この古い形式は、新しいコードでは使わないでください。

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。

デモ

メニュー File→Example Experiments→Feature Demos 2→All Controls Demo

cd *dataFolderSpec*

cd コマンドは、現在のデータフォルダーを指定されたデータフォルダーに設定します。
これは、より長い名前の SetDataFolder コマンドと全く同じ動作をします。

cd は、UNIX の「ディレクトリ変更」コマンドにちなんで名付けられました。

参照

SetDataFolder、pwd、Dir コマンド
ヘルプ Data Folders

Chart *ctrlName* [*keyword=value* [, *keyword=value ...*]]

Chart コマンドは、チャートコントロールを作成または変更します。
チャートは通常、データ取得と組み合わせて使われます。
チャートは FIFO に接続する必要はありませんが、接続されるまでは実用的な機能を発揮しません。

コントロールの状態やステータスに関する情報を確認するには、ControlInfo コマンドを使ってください。

パラメーター

ctrlName は、作成または変更するチャートの名前です。

align=*alignment*　コントロールの配置モードを設定します。配置モードは、pos キーワードに対する *leftOrRight* パラメーターの解釈をコントロールします。
align キーワードは Igor Pro 8.0 で追加されました。

alignment=0（デフォルト）の場合、*leftOrRight* はコントロールの左端の位置を指定し、コントロールのサイズが変更されても左端の位置は固定されたままになります。

alignment=1 の場合、*leftOrRight* はコントロールの右端の位置を指定し、コントロールのサイズが変更されても右端の位置は固定されたままになります。

chans={*ch#*, *ch#*,...}
チャートが監視する FIFO チャネル番号の一覧です。

color(*ch#*)=(*r*, *g*, *b*[, *a*])
指定したトレースの色を設定します。*r*、*g*、*b*、*a* は、色と（オプションで）不透明度を RGBA 値として指定します。

ctab=*colortableName*
チャンネルが画像ストリップ FIFO チャネルに接続されると、データはこの組み込みカラーテーブルを使って画像として表示されます。有効な名前は、画像で使われるものと同じです。無効な名前を指定した場合、デフォルトの Grays カラーテーブルが使われます。

disable=*d*　コントロールのユーザーによる編集可否を設定します。

d=0：　通常

d=1：　隠す

d=2：　ユーザー入力を無効にします。チャートは読み取り専用であるため、外観は変化しません。無効にすると、手の形をしたカーソルは表示されません。

d=3：　コントロールを非表示にし、無効にします。これは、非表示のタブにあるために非表示になっているコントロールを無効にする場合に便利です。

fbkRGB=(*r*, *g* ,*b*[, *a*])
フレームの背景色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。

fgRGB=(*r*, *g*, *b*[, *a*])
前景色（テキストなど）を設定します。*r*、*g*、*b*、*a* は、色と（オプションで）不透明度を RGBA 値として指定します。

fifo=*FIFOName*　そのチャートが監視する FIFO を設定します。

font="*fontName*"　チャートのフォントを設定します。例：font=”Helvetica”

fsize=*s*　チャートのフォントサイズを設定します。

fstyle=*fs*　フォントスタイルを指定します。*fs* はビット単位のパラメーターであり、各ビットは以下のようにフォントスタイルをコントロールします：

ビット 0：　Bold（太字）

ビット 1：　Italic（斜体）

ビット 2：　Underline（下線）

ビット 4：　Strikethrough（取消線）

ビット設定の詳細は、ヘルプ Setting Bit Parameters を参照してください。

gain(*ch#*)=*g*　指定したチャネルの表示ゲインを基準値に対して設定します。1 より大きい値を設定すると、表示範囲が広がります。

gridRGB=(*r*, *g*, *b*[, *a*])
グリッドの色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。

guides={*left*, *hcenter*, *right*, *top*, *vcenter*, *bottom*}
レイアウトガイドへのコントロールポイントの配置をコントロールします。詳細や例には、ヘルプ Laying Out Controls in Guides Mode を参照してください。

left、*hcenter*、*right*、*top*、*vcenter*、*bottom* の各々は、適切な方向を持つレイアウトガイドの名前で置き換えられます。つまり、*left*、*hcenter*、*right* は垂直ガイドへの配置を示し、*top*、*vcenter*、*bottom* は水平ガイドへの配置を示します。配置されていないアンカーポイントを表すには、特別な名前 *kwNone* を使います。

コントロールのレイアウトに過度な制約を課すと、エラーが発生します。３つの水平または垂直のコントロールポイントのうち、レイアウトガイドに接続できるのは最大２つまでです。３つのうち１つを接続すると、レイアウトガイドが移動する時にコントロールも移動します。２つを接続すると、コントロールは移動するとともにサイズも変更されます。したがって、*kwNone* という名前は少なくとも２回出現することになります。

すべてのアンカーポイントに *kwNone* を指定すると、コントロールがレイアウトガイドから完全に切り離されます。

help={*helpStr*}　コントロールのヘルプを設定します。

helpStr の最大長は 1970 バイトです（Igor Pro 8 およびそれ以前のバージョンでは 255 バイト）。

引用符で囲んだ文字列の中に “\r” を入れると、改行を挿入できます。

jumpTo=*p*　チャートがポイント番号 *p* にジャンプします。これはレビューモードでのみ機能します。

lineMode(*ch#*)=*lm*　指定されたチャンネルの表示行モードを設定します。

lm=0：　ドットモード。値をドットとして描画します。ただし、ストリップ内のドット数が maxDots を超える場合、ストリップに詰め込まれた値の最小値から最大値までを結ぶ垂直線を描画します。

lm=1： ラインモード。指定されたストリップ内のポイントの最小値と最大値、直前のストリップの最後のポイントを含む垂直線を描画します。どのストリップが「直前のストリップ」にあたるかは移動方向によって異なるため、チャートの移動方向によっては表示位置がわずかにずれる場合があります。

lm=2： ゼロモードを維持します。指定されたストリップ内のポイントの最小値と最大値、値「0」を含む垂直線を描画します。

mass=*m* マウスでチャート用紙を動かす時の動きの感触を設定します。質量 *m* が大きいほど、チャートレコーダーの反応は遅くなります。奇数値を設定すると、マウスをクリックした瞬間に用紙の動きが止まりますが、偶数値を設定すると、質量があるかのような動きが続きます。

maxDots=*md* チャートの指定された垂直範囲内のポイントが、点線として表示されるか実線として表示されるかをコントロールします。上記の lineMode を参照してください。デフォルトは 20 です。

offset(*ch#*)=*o* 指定したチャネルの表示オフセットを設定します。オフセット値は、ゲインが適用される前にデータから差し引かれます。

oMode=*om* チャートの動作モード。

om=0： ライブモード。

om=1： レビューモード。

pbkRGB=(*r*, *g*, *b*[, *a*])
プロット領域の背景色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションで）不透明度を指定します。

pos={*leftOrRight*, *top*}
コントロールの配置モードが 0 の場合は、コントロールの左上隅の位置を、配置モードが 1 の場合は、コントロールの右上隅の位置を、コントロールパネル単位で設定します。詳細は、align キーワードを参照してください。

ppStrip=*pps* チャートの各垂直ストリップに詰め込まれるデータポイントの数です。

rename=*newname* コントロールに新しい名前を付けます。

rSize(*ch#*)=*rs* 指定されたチャンネルに割り当てられる相対的な垂直サイズを設定します。デフォルト値は 1 です。*rs* の値が 0 の場合、このチャネルは前のチャネルとスペースを共有します。

sMode=*sm* ステータスラインモードです。

sm=0： 装飾的なステータスラインと位置調整バーを非表示にします。

sm=1： 通常モードです。

sm=2： バーに別のスタイルを適用します。

sRate=*sr* *sr* はスクロール速度（1 秒あたりの垂直ストリップ数）です。チャートレコーダーが再生モードの場合、負の数値は逆方向にスクロールします。

title=*titleStr* グラフのタイトルを指定します。タイトルを付けない場合は “” を使ってください。

uMode=*um* 更新モードです。

um=1： 余計な機能は一切ありません、素早い更新がされます。

um=2： ステータスラインと位置調整バー。

um=3：　ステータスライン、位置調整バー、アニメーション付きペン。

win=*winName*　指定された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。指定がない場合は、最前面のグラフまたはパネルウィンドウ、あるいはそのサブウィンドウが対象となります。

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

対象ウィンドウは、グラフまたはパネルでなければなりません。

一部のチャートキーワードの動作は、データ取得が行われているかどうかによって異なります。
チャートがレビューモードの場合、すべてのキーワードによってチャートが再描画されます。
データ取得が行われており、チャートがライブモードの場合、一部のキーワードは新しいデータに影響を与えますが、すでに描画済みの「紙面」の部分は更新しようとしません。
以下のキーワードは、ライブモード中に新しいデータにのみ影響を与えます：

ppStrip, maxDots, gain, offset, color, lineMode

参照

ヘルプ FIFOs and Charts

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。

デモ

メニュー File→Example Experiments→Feature Demos→FIFO Chart Demo FM

メニュー File→Example Experiments→Feature Demos→Wave Review Chart Demo

メニュー File→Example Experiments→Imaging→Image Strip FIFO Demo

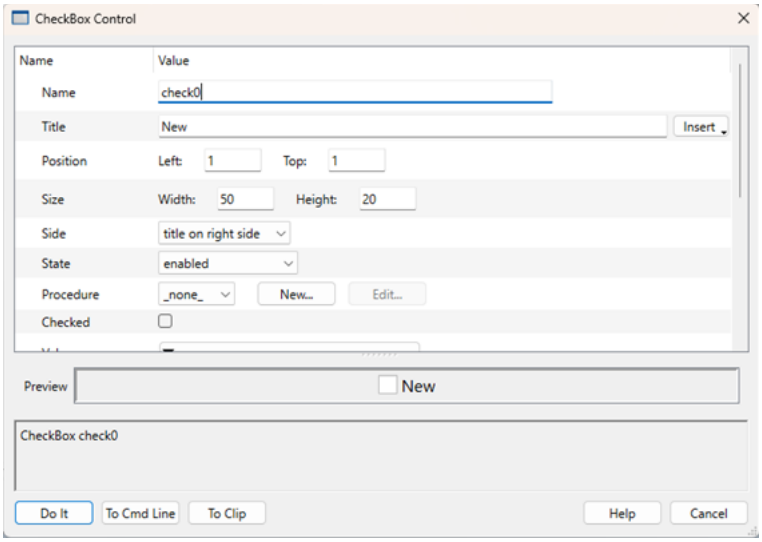
CheckBox [/Z] *ctrlName* [*keyword=value* [, *keyword=value*...]]

CheckBox コマンドは、グラフまたはコントロールパネルである必要がある対象ウィンドウまたは指定されたウィンドウ内に、チェックボックス、ラジオボタン、または展開三角形を作成または変更します。

ctrlName はチェックボックスの名前です。

コントロールの状態やステータスに関する情報を確認するには、ControlInfo コマンドを使ってください。

メニュー Panel→Add Control→Add Check Box のダイアログに相当します。



パラメーター

以下の keyword=value パラメーターがサポートされています。

ctrlName は、作成または変更する CheckBox コントロールの名前です。

align=*alignment*

コントロールの配置モードを設定します。配置モードは、pos キーワードに対する *leftOrRight* パラメーターの解釈をコントロールします。
align キーワードは Igor Pro 8.0 で追加されました。

alignment=0（デフォルト）の場合、*leftOrRight* はコントロールの左端の位置を指定し、コントロールのサイズが変更されても左端の位置は固定されたままになります。

alignment=1 の場合、*leftOrRight* はコントロールの右端の位置を指定し、コントロールのサイズが変更されても右端の位置は固定されたままになります。

appearance={*kind*[, *platform*]}

コントロールの外観を設定します。*platform* は省略可能です。両方のパラメーターは名前であり、文字列ではありません。

種類は、default、native、または os9 のいずれかになります。

platform は、Mac、Windows、または All のいずれかを選択できます。

外観に関する詳細については、Button と DefaultGUIControls コマンドを参照してください。

disable=*d*

コントロールのユーザーによる編集可否を設定します。

d=0： 通常

d=1： 隠す

d=2： ユーザー入力を無効にします。

fColor=(*r*, *g*, *b*[, *a*])

タイトルの初期色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。デフォルトは不透明な黒です。

タイトルテキストの色をさらに変更するには、title=*titleStr* で説明しているようにエスケープシーケンスを使ってください。

focusRing=*fr*　　キーボードのフォーカスを示す四角形の表示を有効または無効にします。

fr=0：　フォーカスの四角形は描画されません。

fr=1：　フォーカスの四角形が表示されます（デフォルト）。

fsize=*s*　　チェックボックスのフォントサイズを設定します。

guides={*left*, *hcenter*, *right*, *top*, *vcenter*, *bottom*}
レイアウトガイドへのコントロールポイントの配置をコントロールします。詳細や例には、ヘルプ Laying Out Controls in Guides Mode を参照してください。

left、*hcenter*、*right*、*top*、*vcenter*、*bottom* の各々は、適切な方向を持つレイアウトガイドの名前で置き換えられます。つまり、*left*、*hcenter*、*right* は垂直ガイドへの配置を示し、*top*、*vcenter*、*bottom* は水平ガイドへの配置を示します。配置されていないアンカーポイントを表すには、特別な名前 *kwNone* を使います。

コントロールのレイアウトに過度な制約を課すと、エラーが発生します。チェックボックスコントロールの幅と高さは、フォント、フォントサイズ、タイトルの長さによって決定されます。そのため、各方向のレイアウトガイドにはアンカーを 1 つしか配置できません。2 つのアンカーポイントを配置すると、チェックボックスコントロールのサイズ変更が試みられる可能性が高いためです。

すべてのアンカーポイントに *kwNone* を指定すると、コントロールがレイアウトガイドから完全に切り離されます。

help={*helpStr*}　　コントロールのヘルプを設定します。

helpStr の最大長は 1970 バイトです（Igor Pro 8 およびそれ以前のバージョンでは 255 バイト）。

引用符で囲んだ文字列の中に “\r” を入れると、改行を挿入できます。

labelBack=(*r*, *g*, *b*[, *a*]) または 0
チェックボックスの背景色を設定します。

labelBack キーワードは、Igor Pro 9.0 で追加されました。

r、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。

labelBack を省略するか、labelBack=0 を指定した場合、背景色は CheckBox が表示されるウィンドウの背景色になります。

mode=*m*　　チェックボックスの外観を指定します。

m=0：　デフォルトのチェックボックスの表示。

m=1：　ラジオボタンとして表示。

m=2：　ツリービューの展開ノードとして表示。

noproc　　チェックボックスがクリックされた時に、プロシージャを実行しないことを指定します。

picture=*pict*　　指定された画像を使ってチェックボックスを描画します。この画像は、通常状態、マウスボタンが押下されている状態、無効状態におけるコントロールの外観を示す、6 つのフレームが並んだものとみなされます。最初の 3 つのフレームはチェック状態が false のときに使われ、次の 3 つのフレームは true の状態を示します。画像は、グローバル（インポートされた）画像、またはプロシージャ画像のいずれでもかまいません（ヘルプ Proc Pictures を参照）。

pos={*leftOrRight*, *top*}

コントロールの配置モードが 0 の場合は、コントロールの左上隅の位置を、配置モードが 1 の場合は、コントロールの右上隅の位置を、コントロールパネル単位で設定します。詳細は、前述の align キーワードを参照してください。

pos+={*dx*, *dy*}　コントロールパネルの単位で、チェックボックスの位置をオフセットします。

proc=*procName*　チェックボックスがクリックされたときに実行するプロシージャを指定します。

rename=*newname*　チェックボックスの名前を変更します。

side=*s*　*s*=0：　チェックボックスは左側に、タイトルは右側に表示されます（デフォルト）。

s=1：　チェックボックスは右側にあり、タイトルは左側にあります。

size={*width*, *height*}

コントロールの幅と高さを、コントロールパネル単位で設定します。

title=*titleStr*　チェックボックスのタイトルを、指定された文字列式に設定します。タイトルとは、チェックボックスに表示されるテキストのことです。指定がない場合、または "" の場合、タイトルは "New" になります。

エスケープコードを使うと、タイトルのフォント、サイズ、書式、色を変更できます。詳細は、ヘルプ Annotation Escape Codes を参照してください。

userdata(*UDName*)=*UDStr*

名前のないユーザーデータを *UDStr* に設定します。オプションの（*UDName*）を使って、作成する名前付きユーザーデータを指定します。

userdata(*UDName*)+=*UDStr*

現在の名前のないユーザーデータに *UDStr* を追加します。オプションの（*UDName*）を使うと、名前付きのユーザーデータに追加することができます。

value=*v*　チェックボックスがオン（*v*=1）かオフ（*v*=0）かを指定します。

variable=*varName*　チェックボックスがクリックされたとき、または「value」キーワードによって設定されたときに、その現在の状態が格納されるグローバル数値変数を指定します。この変数は双方向です。つまり、変数に値を設定すると、チェックボックスの状態も変更されます。

wave=*waveName*　チェックボックスがクリックされたとき、または「value」キーワードによって設定されたときに、そのチェックボックスの現在の状態に設定するウェーブのポイントを指定します。ポイントは、標準の角括弧表記を使って、数値のポイント番号または行ラベルのいずれかで指定します。例：value=awave[4] または value=awave[%alabel]。
また、2D、3D、または 4D のウェーブを使い、行インデックスに加えて、列、レイヤー、チャンクインデックス、または次元ラベルを指定することもできます。
この機能は Igor Pro 9.0 で追加されました。

win=*winName*　指定された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。指定がない場合は、最前面のグラフまたはパネルウィンドウ、あるいはそのサブウィンドウが対象となります。

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。

フラグ

/Z　エラーを報告しません。

詳細

対象ウィンドウは、グラフまたはパネルでなければなりません。

チェックボックスアクションプロシージャ

CheckBox コントロールのプロシージャでは、関数のパラメーターとして、あらかじめ定義された構造体 WMCheckboxAction が渡されます。

```
Function ActionProcName(CB_Struct) : CheckBoxControl
    STRUCT WMCheckboxAction &CB_Struct
    ...
    return 0
End
```

「: CheckBoxControl」という指定により、このプロシージャを CheckBox Control ダイアログの Procedure ポップアップメニューに含めるよう指示します。

WMCheckboxAction 構造体の詳細は、WMCheckboxAction を参照してください。

現在、戻り値は使われていませんが、アクションプロシージャは常に 0 を返すようにしてください。

古いコードには、次のような旧形式のチェックボックスプロシージャが含まれている場合があります。

```
Function procName (ctrlName,checked) : CheckBoxControl
    String ctrlName
    Variable checked    // 1 if selected, 0 if not
    ...
    return 0
End
```

この古い形式は、新しいコードでは使わないでください。

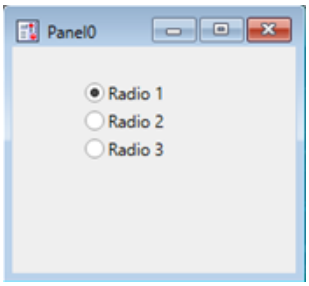
ラジオボタンコントロールを使う場合、ラジオボタンのグループのうち 1 つが押された時に、他のラジオボタンを無効にする処理を追加してください。

例

<プロシージャウィンドウ>

```
// ラジオボタンのグループを作成

Window Panel0() : Panel
    PauseUpdate; Silent 1       // ウィンドウの構築
    NewPanel /W=(150,50,353,212)
    Variable/G gSelectedRadioButton = 1
    CheckBox radioButton1,pos={52,25},size={78,15},title="Radio 1",value=
1,mode=1,proc=MyRadioButtonProc
    CheckBox radioButton2,pos={52,45},size={78,15},title="Radio 2",value=
0,mode=1,proc=MyRadioButtonProc
    CheckBox radioButton3,pos={52,65},size={78,15},title="Radio 3",value=
0,mode=1,proc=MyRadioButtonProc
EndMacro

static Function HandleRadioButtonClick(controlName)
String controlName

    NVAR gSelectedRadioButton = root:gSelectedRadioButton

    strswitch(controlName)
        case "radioButton1":
```

```
            gSelectedRadioButton = 1
            break
        case "radioButton2":
            gSelectedRadioButton = 2
            break
        case "radioButton3":
            gSelectedRadioButton = 3
            break
    endswitch
    CheckBox radioButton1, value = gSelectedRadioButton==1
    CheckBox radioButton2, value = gSelectedRadioButton==2
    CheckBox radioButton3, value = gSelectedRadioButton==3
End

Function MyRadioButtonProc(cb) : CheckBoxControl
    STRUCT WMCheckboxAction& cb

    switch(cb.eventCode)
        case 2:     // マウスを上げる
            HandleRadioButtonClick(cb.ctrlName)
            break
    endswitch

    return 0
End
```



<コマンドウィンドウ>

```
Panel0()
```

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。

CheckDisplayed [/A/W=*winName*] *waveName* [, *waveName*]...

CheckDisplayed コマンドは、指定されたウェーブがホストウィンドウまたはサブウィンドウに表示されているか、あるいはその他の方法で使われているかを判定します。

フラグ

/A　すべてのグラフ、テーブル、ページレイアウト、パネル、Gizmo ウィンドウをチェックします。

/W=*winName*　指定されたウィンドウのみをチェックします。

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。

詳細

/A も /W も指定されていない場合、CheckDisplayed は最前面のグラフ、テーブル、ページレイアウト、パネル、または Gizmo ウィンドウのみをチェックします。

CheckDisplayed は、どのウェーブを表示するかに応じて、変数 V_flag をビット単位で設定します。

ウィンドウに直接表示されていないウェーブであっても、そのウィンドウで使うことは可能です。
例えば、CheckDisplayed は、非表示のトレースに関連付けられたウェーブ、カラーインデックスウェーブ、エラーバーの指定に使われるウェーブ、グラフだけでなくページレイアウトやコントロールパネル内を含むポリゴンの描画に使われるウェーブなどについては、1 を返します。

例

```
// Graph0 を確認し、aWave、bWave、cWave が表示されているかどうかを調べます。
// aWave が表示されている場合、V_flag のビット 0 をセットします。
// bWave が表示されている場合、V_flag のビット 1 をセットします。
// cWave が表示されている場合、V_flag のビット 2 をセットします。
CheckDisplayed/W=Graph0 aWave,bWave,cWave
```

参照

ビットパラメーターの詳細は、ヘルプ Setting Bit Parameters を参照してください。

ChooseColor [/A[=*a*]/C=(*r*,*g*,*b*[,*a*])]

ChooseColor コマンドを実行すると、色を選択するためのダイアログが表示されます。

/C オプションで別の色を指定しない限り、最初に表示される色は黒です。

フラグ

/A[=*a*]　　*a*=1：　アルファ（不透明度）チャンネルを表示します。/A は /A=1 と同じです。

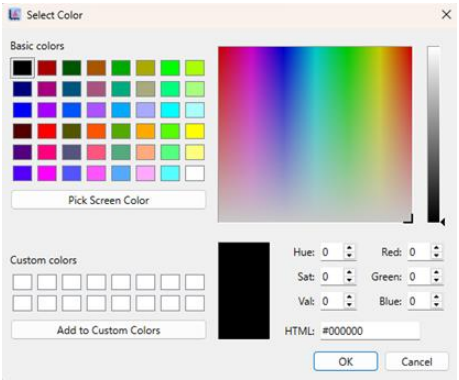
a=0：　アルファチャンネルが非表示になります。これがデフォルトの設定です。

/A フラグは、Igor Pro 7.0 で追加されました。

/C=(*r*, *g*, *b*[, *a*])

ダイアログに最初に表示される色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。

詳細



ChooseColor は、ユーザーがダイアログで「OK」をクリックした場合は変数 V_Flag を 1 に、それ以外の場合は 0 に設定します。

V_Flag が 1 の場合、V_Red、V_Green、V_Blue、および V_Alpha は、選択された色を 0 から 65535 までの整数として設定されます。

完全に不透明な色では、V_Alpha=65535 に設定されます。完全に透明な色では、V_Alpha=0 に設定されます。

参照

ImageTransform の *rgb2hsl* と *hsl2rgb* キーワード

Close [/A] [*fileRefNum*]

Close コマンドは、Open コマンドによって以前に開かれたファイルを 1 つ閉じるか、/A オプションが指定されている場合は、そのようなファイルをすべて閉じます。

パラメーター

fileRefNum は、閉じるファイルのファイル参照番号です。この番号は Open コマンドによって割り当てられます。/A オプションを使う場合は、*fileRefNum* を省略する必要があります。

フラグ

/A　すべてのファイルを閉じます。主に、プロシージャの実行中にエラーが発生した場合のクリーンアップに役立ち、通常の Close コマンドが実行されないようにします。

参照

Open コマンド

CloseHelp [/ALL /FILE=*fileNameStr* /NAME=*helpNameStr* /P=*pathName*]

CloseHelp コマンドは、ヘルプウィンドウを閉じます。

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

フラグ

/ALL　開いているヘルプウィンドウをすべて閉じます。

/FILE=*fileNameStr*
　ディスク上のヘルプファイルの場所を使って、閉じるヘルプウィンドウを特定します。ファイルは、*fileNameStr* と /P=*pathName* によって指定されます。ここで、pathName は Igor Pro のシンボリックパスの名前です。*fileNameStr* には、ファイルへのフルパスを指定できます（この場合、/P は不要です）。また、*pathName* に関連付けられたフォルダーを基準とした部分パス、あるいは *pathName* に関連付けられたフォルダー内のファイル名を指定することもできます。

　fileNameStr 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はヘルプ Path Separators を参照してください。

/NAME=*helpNameStr*
　helpNameStr で指定されたウィンドウのタイトルに基づいて、閉じるヘルプウィンドウを特定します。これは、ヘルプウィンドウのタイトルバーに表示されるテキストです。

/P=*pathname*　/FILE で指定された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 Igor Pro シンボリックパスの名前です。

詳細

次のフラグのいずれかを指定する必要があります：/ALL、/FILE、/NAME

参照

OpenHelp コマンド

CloseMovie

CloseMovie コマンドは、現在開いているムービーを閉じます。
ムービーを再生するには、まずそれを閉じる必要があります。

フラグ

/Z　　エラーの表示を抑制します。/Z オプションを使う場合は、コマンドが成功したかどうかを確認するために、出力変数 V_Flag を確認してください。

出力変数

V_Flag　　コマンドが成功した場合は 0 に、失敗した場合は 0 以外のエラーコードに設定します。

V_Flag は、/Z フラグを使った場合にのみ設定されます。

参照

Movies、NewMovie コマンド

CloseProc /NAME=*procNameStr* [/P=*PathName*] [/COMP=[*compile*] [/D=[*delete*]] [/SAVE[=*savePathStr*]]

CloseProc /FILE=*procFileStr* [/P=*PathName*] [/COMP=[*compile*] [/D=[*delete*]] [/SAVE[=*savePathStr*]]

CloseProc コマンドは、プロシージャウィンドウを閉じます。
メインのプロシージャウィンドウでは、CloseProc を呼び出すことはできません。

CloseProc を使うと、プロシージャファイルをプログラムで作成、変更することができます。
これは、内容が変化するユーザー定義のメニューバーを作成する場合などに利用できます。

注記

CloseProc は、Procedure ウィンドウを変更し、関数やマクロの実行中は呼び出せないようにします。関数やマクロからこれを呼び出したい場合は、Execute/P を使ってください。

注意

ソースファイルがない、または保存先ファイルを指定せずにプロシージャウィンドウを閉じると、ウィンドウの内容は永久に失われます。

フラグ

/COMP[=*compile*]
　　プロシージャウィンドウを閉じた後に、プロシージャをコンパイルするかどうかを指定します。

　　compile=0：　プロシージャをコンパイルされていない状態に保ちます。

　　compile=1：　プロシージャをコンパイルします（/COMP と同じ動作です）。

/D[=*delete*]　　プロシージャウィンドウを閉じた後、そのプロシージャファイルを削除するかどうかを指定します。

　　delete=0：　関連するファイルには一切影響を与えません。

　　delete=1：　プロシージャファイルを削除します（/D オプションのみと同じ動作）。

　　注意

　　/D=1 を使って削除されたファイルは復元できません。

/FILE=*fileNameStr*
　　fileNameStr で指定されたファイル名とファイルへのパスを使って、閉じるプロシージャウ

ィンドウを特定します。/P オプションを使って親フォルダーのシンボリックパス名を指定している場合は、文字列はファイル名のみでも構いません。/P オプションが、その部分パスの先頭を含む親フォルダーを指している場合は、部分パスでも構いません。また、ファイルへの完全なパスでも構いません。

/NAME=*procNameStr*

文字列式 *procNameStr* を使って、閉じるプロシージャウィンドウを特定します。これは、ウィンドウのタイトルに表示されるテキストと同じものです。プロシージャウィンドウがファイルに関連付けられている場合、そのファイル名と拡張子になります。

独立モジュールの一部であるプロシージャファイルを閉じるには、*procNameStr* にその独立モジュールの名前を含める必要があります。例えば：

```
CloseProc /NAME="GraphBrowser.ipf [WM_GrfBrowser]"
```

ファイル名の後にスペースを入れ、その後に括弧で囲んだ独立したモジュール名を続けることに注意してください。

/P=*pathname*　/FILE で指定された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。

/SAVE[=*savePathStr*]

ウィンドウを閉じる前にプロシージャを保存します。引数を指定せずにこのフラグを使うと、プロシージャウィンドウへの変更内容を、ウィンドウを閉じる前にそのソースファイルに保存します。*savePathStr* が指定されている場合、それはプロシージャウィンドウの内容を保存するファイルの完全なパスでなければなりません。/P フラグは *savePathStr* と併用できないため、*savePathStr* は完全なパスでなければなりません。

詳細

CloseProc は、マクロや関数内からは呼び出すことができません。
コマンドラインから、または Execute/P を使って呼び出してください（ヘルプ Operation Queue を参照）。

/NAME または /FILE フラグのいずれかを使って、閉じるウィンドウを指定します。
どちらか一方を使う必要があります。
通常は /NAME を使います。
こちらの方が便利な場合が多いためです。
２つのプロシージャの名前が同じ場合は、/FILE を使ってそれらを区別することができます。

誰かが「Procedure」という、まったくセンスのない名前を付けてしまったメインプロシージャ以外のプロシージャウィンドウに対しては、CloseProc を呼び出すことはできません。

参照

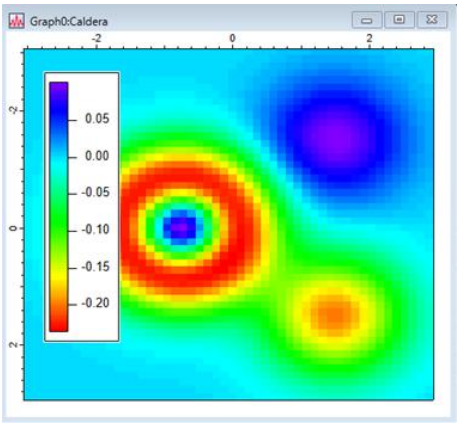
Execute コマンド
ヘルプ Procedure Windows

ColorScale [flags] [, keyword=value, ...] [axisLabelStr]

ColorScale コマンドは、グラフ、Gizmo プロット、またはページレイアウトにカラースケール（または「色の凡例」）の注釈を追加します。
詳細は、ヘルプ Color Scales を参照してください。

Igor Pro 8.0 で、Gizmo プロットにカラースケールを追加する機能が追加されました。

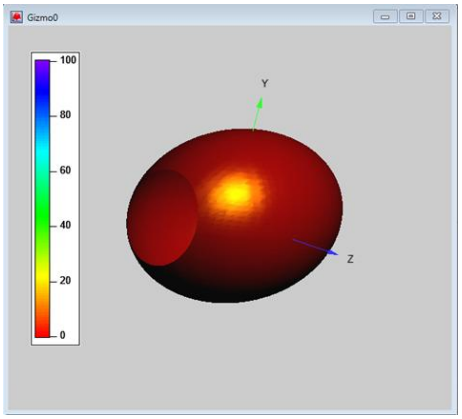
ColorScale コマンドは、フラグやパラメーターを指定せずに実行できます。
このコマンドは、現在のターゲットウィンドウ、アクティブなサブウィンドウ、または /W フラグで指定されたウィンドウもしくはサブウィンドウに対して実行されます。



ターゲットがグラフの場合、カラースケールは、そのグラフに最初に追加された画像プロットに関連付けられた色と値を表します。
グラフに画像プロットが存在しない場合、カラースケールは、そのグラフに最初に追加されたコンタープロットまたは f(z) トレースのいずれかを表します。

パラメーターを指定せずにコマンドを実行する時、エラーなしで実行されるためには、これらいずれかが存在している必要があります。

ターゲットが Gizmo プロットの場合、カラースケールは最初のサーフェスオブジェクトに関連付けられた色と値を表し、サーフェスのグリッドラインのカラーテーブルよりも、サーフェスの塗りつぶしカラーテーブルを優先します。
そのようなサーフェスオブジェクトが存在しない場合、カラースケールは、カラーテーブルを使って最初の散布オブジェクトを表します。
そのような散布オブジェクトが存在しない場合、カラースケールは、カラーテーブルを使って最初のパスオブジェクトを表します。
そのようなパスオブジェクトが存在しない場合、カラースケールは、カラーテーブルを使って最初のリボンオブジェクトを表します。



対象がページレイアウトの場合、パラメーターを指定せずに ColorScale を実行すると、ctab={0,100,Rainbow} というパラメーターが指定されたかのようにカラーバーが表示されます。

フラグ

/W=win フラグを使って、特定のグラフまたはレイアウトウィンドウを指定します。
コマンドライン、マクロ、またはプロシージャで使う場合は、/W フラグを他のすべてのフラグよりも先に指定する必要があります。

カラースケールを変更するには、/C/N=*name* フラグを使います。
注釈は、作成時に名前が指定されていない場合、自動的に「text0」、「text1」などの名前が付けられます。
既存の注釈を変更する時は、その名前を使う必要があります。
そうしないと、新しい注釈が作成されてしまいます。

すべてのフラグの説明は、TextBox コマンドを参照してください。

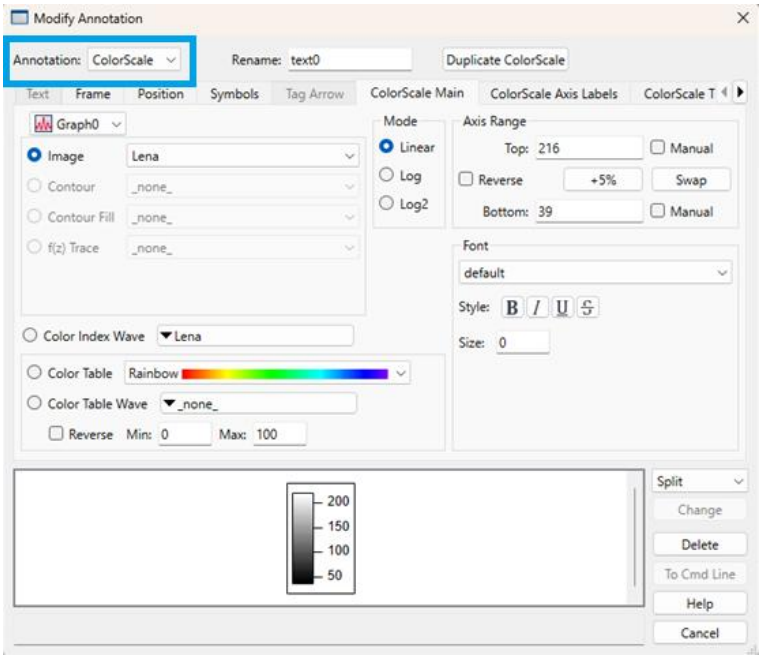
パラメーター

以下のキーワードと値のペアは、ColorScale コマンドにおける重要なパラメーターです。
これらは、カラースケールがどのオブジェクトを表しているかを指定するためです。

注記

ページレイアウトで使う場合、keyword ={graphName,...} の形式が必須となります。

グラフの場合は、image=imageInstanceName という形式（graphName を省略）を使う方が簡単ですが、$”” を使って最前面のグラフを指定したり、長い形式で別のグラフの名前を指定したりすることもできます。例を参照してください。



axisLabelStr　　カラースケールの主軸の横に表示されるテキストが含まれています。このテキストは、Label コマンドの場合と同様に解釈されます。

エスケープコードを使うと、テキストのフォント、サイズ、書式、色を変更したり、上付き文字や下付き文字を作成したり、その他の効果を適用したりすることができます。詳細は、ヘルプ Annotation Escape Codes を参照してください。

この軸ラベルの文字列を変更するには、AppendText または ReplaceText コマンドを使用できます。*axisLabelStr* のデフォルト値は "" です。

cindex=cindexMatrixWave

表示されている色は、指定されたカラーインデックスウェーブに含まれる色であり、その軸の値は、そのウェーブの X 軸（行）のスケーリングと単位に基づいて算出されています。

画像の色は、指定された行列ウェーブを参照して決定されます。ModifyImage コマンドで使われる cindex キーワードを参照してください。

contour=*contourInstanceName*
contour={*graphName*, *contourInstanceName*}

各色は、指定されたコンタープロットの線の色、それに対応するコンター（Z）値とコンターデータの単位を示しています。カラーテーブル、カラーインデックス（cindex）、固定色など、画像プロットのすべての特性が表されています。

graphName には、最前面のグラフウィンドウの名前、または Panel0#G0 のようなサブウィンドウの指定を指定できます。サブウィンドウの指定に関する詳細は、ヘルプ Subwindow Syntax を参照してください。

contourFill=*contourInstanceName*
contourFill={*graphName*, *contourInstanceName*}

各色は、指定されたコンタープロットの塗りつぶし色、それに対応するコンター（Z）値とコンターデータの単位を示しています。カラーテーブル、カラーインデックス（cindex）、固定色など、画像プロットのすべての特性が表されています。

graphName には、最前面のグラフウィンドウの名前、または Panel0#G0 のようなサブウィンドウの指定を指定できます。サブウィンドウの指定に関する詳細は、ヘルプ Subwindow Syntax を参照してください。

contourFill キーワードは、Igor Pro 8.0 で追加されました。

ctab={*zMin*, *zMax*, *ctName*, *reverse*}

ctName で指定されたカラーテーブルが、色凡例に描画されます。*ctName* には、CTabList 関数が返す値（Grays や Rainbow など）のいずれかを指定できます。ヘルプ Color Tables も参照してください。

カラーテーブルを変更しない場合は、カラーテーブル名を省略できます。*zMin* と *zMax* は、マッピングする Z 値の範囲を設定します。パラメーター *mode* を 1 に設定すると、カラーテーブルが反転します。0 または値が指定されていない場合は、カラーテーブルは反転しません。

ctName で指定されたカラーテーブルが、色凡例に描画されます。

zMin と *zMax* は、マッピングする Z 値の範囲を設定します。

カラーテーブルの名前を変更しない場合は、省略することができます。

ctName には、CTabList 関数によって返される任意の色テーブル名（Grays や Rainbow など。詳細はヘルプ Color Tables を参照）や、3 列または 4 列のカラーテーブルウェーブの名前（詳細はヘルプ Color Table Waves を参照）を指定できます。

ctab に指定するカラーテーブルのウェーブ名は、組み込みのカラーテーブルの名前であってはなりません（CTabList コマンドを参照）。3 列または 4 列のカラーテーブルのウェーブの値は、0 から 65535 の範囲でなければなりません。列 0 は赤、1 は緑、2 は青です。列 3 において、値 65535 は完全に不透明、0 は完全に透明です。

カラーテーブルの順序を逆にするには、*reverse* を 1 に設定します。0 に設定するか、この引数を省略した場合は、カラーテーブルの順序は逆転されません。

image=*imageInstanceName*

image={*graphName*, *imageInstanceName*}

各色は、指定された画像プロットの色、関連する画像（Z）値、画像データの単位を示しています。カラーテーブル、カラーインデックス（cindex）、ルックアップウェーブ、評価色、NaN の透明度など、画像プロットのすべての特性が表されています。

注記

ColorScale では、偽色画像のプロットのみを使用できます（ヘルプ Indexed Color Details を参照）。

graphName には、トップレベルのグラフウィンドウの名前、または Panel0#G0 のようなサブウィンドウの指定を指定できます。

logLabel=*t*

t は、補助目盛の目盛ラベルが非表示になるまでの最大デケード数です。デフォルト値は 3 です。

logLabel キーワードは、Igor Pro 7.0 で追加されました。

logTicks=*t*

t=0 は「自動」を意味します。これがデフォルト値です。

t が 0 以外の場合、これは補助目盛が抑制されるまでの最大デケード数を表します。

logTicks キーワードは、Igor Pro 7.0 で追加されました。

lookup=*waveName*　ctab キーワードで指定されたカラーテーブルへの、スケーリングされた Z 値のマッピングを変更するために使われる、オプションの 1D ウェーブを指定します。値は 0.0 から 1.0 の範囲で指定する必要があります。0 から 1 への線形ランプは効果をもたらしませんが、1 から 0 へのランプは画像を反転させます。グレースケール画像へのガンマ補正や特殊効果の適用に使われます。このオプションを無効にするには、lookup=$"" を指定してください。

画像プロットでルックアップウェーブを使う場合でも、このキーワードは image キーワードと併用する必要はありません。ColorScale のルックアップウェーブの代わりに、画像プロットのルックアップウェーブが使われます。

path=*gizmoObjectSpec*
path={*gizmoName*, *gizmoObjectSpec*}

各色は、指定された Gizmo パスプロットの色と、それに対応する X、Y、または Z の値、単位を示しています。表示されるのはパスプロットのカラーテーブルのみです。カラーウェーブや単色などの他のカラーモードはサポートされていません。

gizmoName には、最前面の Gizmo ウィンドウの名前、または Panel0#GZ0 のようなサブウィンドウの指定を指定できます。サブウィンドウの指定に関する詳細は、ヘルプ Subwindow Syntax を参照してください。

gizmoObjectSpec には、path0 のようにオブジェクトの名前だけを指定することも、group0:path0 のように、グループ内のオブジェクトへのコロン区切りのパスを指定することもできます。ModifyGizmo setCurrentGroup とは異なり、*gizmoObjectSpec* は Gizmo ウィンドウの名前で始まることはありません。

path キーワードは、Igor Pro 9.0 で追加されました。

ribbon=*gizmoObjectSpec*
ribbon={*gizmoName*, *gizmoObjectSpec*}

各色は、指定された Gizmo リボンプロットの色と、それに対応する X、Y、または Z の値、単位を示しています。表示されるのはリボンプロットのカラーテーブルのみです。カラーウェーブや単色などの他のカラーモードはサポートされていません。

{*gizmoName*, *gizmoObjectSpec*} の詳細は、path キーワードを参照してください。

ribbon キーワードは、Igor Pro 9.0 で追加されました。

scatter=*gizmoObjectSpec*
scatter={*gizmoName*, *gizmoObjectSpec*}

色は、指定された Gizmo 散布図のマーカーの色、それに対応する X、Y、または Z の値と単位を示しています。表示されるのは、散布図のカラーテーブルのみです。カラーウェーブや単色などの他のカラーモードはサポートされていません。

{*gizmoName*, *gizmoObjectSpec*} の詳細は、path キーワードを参照してください。

scatter キーワードは、Igor Pro 9.0 で追加されました。

surface=*gizmoObjectSpec*
surface={*gizmoName*, *gizmoObjectSpec*}

色は、指定された Gizmo サーフェスプロットのマーカーの色、それに対応する X、Y、または Z の値と単位を示しています。表示されるのは、サーフェスプロットのカラーテーブルのみです。カラーウェーブや単色などの他のカラーモードはサポートされていません。

{*gizmoName*, *gizmoObjectSpec*} の詳細は、path キーワードを参照してください。

surface キーワードは、Igor Pro 8.0 で追加されました。

surfaceFill=*gizmoObjectSpec*
surfaceFill={*gizmoName*, *gizmoObjectSpec*}

色は、指定された Gizmo サーフェスプロットの塗りつぶし色、それに対応するサーフェスの X、Y、Z の値とサーフェスマトリクスの単位を示しています。画像プロットとは異なり、サーフェスプロットのカラーテーブルのみが表現されます。カラーウェーブや単色などの他のカラーモードはサポートされていません。

gizmoName には、最前面の Gizmo ウィンドウの名前、または Panel0#GZ0 のようにサブウィンドウを指定できます。サブウィンドウの指定に関する詳細は、ヘルプ Subwindow Syntax を参照してください。

{*gizmoName*, *gizmoObjectSpec*} の詳細は、path キーワードを参照してください。

surfaceFill キーワードは、Igor Pro 8.0 で追加されました。

trace=*traceInstanceName*
trace={*graphName*, *traceInstanceName*}

色は、指定されたトレースの色を示しています。これは、ModifyGraph zColor(traceName)=... 機能を使って「z ウェーブ」によってトレースの色が設定されている場合に役立ちます。Modify Trace Appearance ダイアログでは、これは Set as f(z) サブダイアログで選択されます。カラースケールの主軸には、z ウェーブの値の範囲が表示され、そのウェーブに設定されているデータ単位がある場合はそれらが表示されます。

gizmoName には、最前面の Gizmo ウィンドウの名前、または Panel0#GZ0 のようにサブウィンドウを指定できます。サブウィンドウの指定に関する詳細は、ヘルプ Subwindow Syntax を参照してください。

サイズパラメーター

以下のキーワードパラメーターは、カラースケール注釈のサイズを変更します。
これらのキーワードは、Slider コントロールで使われるものと似ています。
注釈のサイズは、「カラーバー」のサイズや各種軸パラメーターを設定することで間接的にコントロールされます。
注釈は、カラーバー、目盛ラベル、軸ラベルを収めるように自動的にサイズが調整されます。

height=*h*　カラーバーの高さをポイント単位で設定し、heightPct の設定を上書きします。デフォルトの高さは、カラースケールが垂直の場合はプロット領域の高さの 75%、水平の場合は 15 ポイントの定数となります。height=0 を指定すると、デフォルト値に戻ります。heightPct の値を指定すると、height はこのデフォルト値にリセットされます。

heightPct=*hpct*　高さをグラフのプロット領域に対する割合（パーセンテージ）として設定し、他の高さ設定を上書きします。カラースケールが垂直の場合、デフォルトの高さはプロット領域の高さの 75%となり、カラースケールが水平の場合は 15 ポイントの定数となります。heightPct=0 を設定すると、デフォルトの高さに戻ります。

side=*s*　主軸の目盛をどの軸に描画するかを選択します。

s=1：　vert=1 の場合はカラーバーの右側、vert=0 の場合は下側に表示されます。

s=2：　vert=1 の場合はカラーバーの左側、vert=0 の場合は上側に表示されます。

vert=*v*　カラースケールの向きを指定します。

v=1：　水平

v=2：　垂直（デフォルト）

width=*w*　　カラーバーの幅をポイント単位で設定し、widthPct の設定を上書きします。デフォルトの幅は、カラースケールが垂直の場合は常に 15 ポイント、カラースケールが水平の場合はプロット領域の幅の 75%となります。width=0 を指定すると、デフォルト値に戻ります。widthPct の値を指定すると、width はこのデフォルト値にリセットされます。

widthPct=*wpct*　　幅をグラフのプロット領域のパーセンテージとして設定し、他の幅の設定を上書きします。カラースケールが垂直の場合、デフォルトの幅は定数の 15 ポイントとなり、カラースケールが水平の場合はプロット領域の幅の 75%となります。widthPct=0 を設定すると、デフォルト値に戻ります。幅の値を指定すると、widthPct はこのデフォルト値にリセットされます。

カラーバーパラメーター

以下のキーワードとパラメーターのペアは、カラースケールのカラーバーの外観を変更します。

colorBoxesFrame=*on*

カラーバー上の最大 99 個の色見本を囲む枠を描画します（*on*=1）。

カラーバーで 99 色以上を指定した場合（100 色の「レインボー」カラーテーブルなど）、ボックスには枠が表示されません。カラーボックスに枠を表示するのは、色の数が少ない場合にのみ有効です。frame キーワードを使って、枠の幅を設定してください。

カラーボックスの枠線を非表示にするには、*on*=0 を指定してください。

frame=*f*　　カラーバーの周囲に描画される枠線の太さをポイント単位で指定します（*f* の値は 0～5 ポイントの範囲です）。デフォルトは 1 ポイントです。小数点以下の値も指定可能です。

f=0 に設定すると、枠線を非表示にします。0.5 未満の値は画面上には表示されませんが、細い枠線が印刷されます。

frameRGB=(*r*, *g*, *b*[, *a*]) または 0

カラーバーを囲む枠の色を設定します。*r*、*g*、*b*、*a* は、RGBA 値として色と（オプションの）不透明度を指定します。colorBoxesFrame=1 の場合、枠には個々のカラーバーの色が含まれます。

frameRGB=0 の場合、フレームには /G フラグで設定されたカラースケールの前景色が使われます。

軸パラメーター

以下のキーワード値パラメーターは、カラースケールの軸の外観を変更します。
これらのキーワードは、メインまたはセカンダリのカラースケールの軸を変更するため、「ModifyGraph for Axes」セクションで説明しているのキーワードに基づいています。

axisRange={*zMin*, *zMax*}

カラーバーの軸範囲を、*zMin* と *zMax* で指定された値に設定します。いずれか一方または両方の値についてデフォルトの軸範囲を使うには、* を使ってください。

zMin または *zMax* を省略すると、その範囲の端の値は変更されません。例えば、{*zMin*, } を使って *zMin* を変更して zMax はそのままにしたり、{ ,*} を使って軸の最大値のみをデフォルト値に設定したりできます。

dateInfo={*sd*, *tm*, *dt*}

日付/時刻軸の書式設定をコントロールします。

sd=0：　date を date&time の形式で表示します。

tm=2（経過時間）の場合、date は常に非表示になります。

1 日あたりのティック数が 1 未満で、かつ *tm*=0（12 時間制）または *tm*=1（24 時制）の場合、時刻は常に非表示になります。

sd=1： date を非表示にします。

tm=2（経過時間）の場合、date は常に非表示になります。

sd=2： time を非表示にします。

tm=2（経過時間）の場合、time は常に表示されます。

sd=2 を使うには、Igor Pro 9.0 以降が必要です。

tm=0： 12 時間制（AM/PM）

tm=1： 24 時間制（軍用表記）

tm=2： 経過時間

dt=0： 短い日付（2/22/90）

dt=1： 長い日付（Thursday, February 22, 1990）

dt=2： 省略形の日付（Thurs, Feb 22, 1990）

軸が「dat」データ単位を持つウェーブによってコントロールされていない限り、これらは効果を発揮しません。

これらの日付形式は、0001-01-01 より前の日付では機能せず、その場合は軸に空の文字列が表示されます。そのような場合は、dateFormat キーワードを使ってカスタムの日付形式を適用することができます。

dateFormat キーワードで指定されたカスタム日付形式を使うには、dateInfo の *dt* パラメーターに -1 を指定する必要があります。

f(z) のカラースケールの場合：

```
SetScale d, 0,0, "dat", fOfZWave
```

コンタープロットや画像プロットのカラースケールの場合：

```
SetScale d, 0,0, "dat", ZorXYZorImageWave
```

日付/時刻軸の仕組みの詳細については、ヘルプ Date/Time Axes と Date, Time, and Date&Time Units を参照してください。

font=*fontNameStr*　フォント名を文字列式で指定します。そのフォントが存在しない場合は、デフォルトのフォントが使われます。「default」を指定しても同様の効果があります。ModifyGraph とは異なり、fontName は実行時に評価されるため、システムにそのフォントが存在しなくてもエラーにはなりません。

fsize=*s*　*s*=0： 目盛ラベルおよび軸ラベルには、デフォルトの（グラフ用）フォントサイズを使います。

s はポイント単位のフォントサイズです。デフォルトは 0（フォントサイズを自動設定）です。

fstyle=*fs*　フォントスタイルを指定します。*fs* はビット単位のパラメーターであり、各ビットは目盛ラベルのフォントスタイルの以下の各要素をコントロールします。

Bit 0： 太字

Bit 1： 斜体

Bit 2： 下線

Bit 3： 取り消し線

ビット設定の詳細は、ヘルプ Setting Bit Parameters を参照してください。

highTrip=*h*　軸の極値が lowTrip と highTrip の間に収まる場合、目盛ラベルには固定小数点表記が使われます。それ以外の場合は、指数表記（科学表記または工学表記）が使われます。デフォルトの highTrip は 100,000 です。

lblLatPos=*p*　主軸ラベルの横方向のオフセットを設定します。これは軸と平行なオフセットです。*p* はポイント単位です。正の値は、縦軸の場合は下方向、横軸の場合は右方向を示します。デフォルトは 0 です。

lblMargin=*m*　主軸のラベルを、通常の位置から *m* ポイント（デフォルトは 0）だけ移動させます。デフォルト値は -5 で、これにより軸ラベルが軸に近づきます。軸ラベルを軸から遠ざけるには、より大きな正の値を使用してください。

lblRot=*r*　メイン軸のラベルを、通常の向きを基準として反時計回りに *r* 度回転させます。*r* の値は -360 から 360 までの範囲です。

log=*l*　軸の種類を指定します。

l=0： 線形（デフォルト）

l=1： 対数（底 10）

l=2： 対数（底 2）

logHTrip=*h*　highTrip と同様ですが、対数軸用です。デフォルト値は 10,000 です。

logLTrip=*l*　lowTrip と同様ですが、対数軸用です。デフォルト値は 0.0001 です。

logTicks=*t*　*t* は、対数軸上で補助目盛が省略されるまでの最大デケード数です。

lowTrip=*l*　軸の極値が lowTrip と highTrip の間に収まる場合、目盛ラベルには固定小数点表記が使われます。それ以外の場合は、指数表記（科学表記または工学表記）が使われます。デフォルトの lowTrip は 0.1 です。

minor=*m*　*m*=0： 補助目盛を無効にします（デフォルト）。

m=1： 補助目盛を有効にします。

notation=*n*　*n*=0： 工学表記（デフォルト）。

n=1： 科学表記

nticks=*n*　*n* は、主軸に沿って配置される目盛の概算数です。目盛ラベルは、グラフの軸に使われるのと同じ自動アルゴリズムで作成されます。デフォルト値は 5 です。

n=0： 目盛なし

prescaleExp=*exp*　目盛ラベル表示において、軸の範囲を 10^exp 倍し、exp を軸ラベルの指数から差し引きます。つまり、指数が目盛ラベルから軸ラベルへと移動されます。デフォルトは 0（変更なし）です。ModifyGraph コマンドの「詳細」セクションの説明を参照してください。

tickExp=*te*　　*te*=1 に設定すると、単位に接頭辞が付いている場合、目盛ラベルが指数表記で表示されます。*te*=0 に設定すると、この機能は無効になります。

tickLen=*t*　　目盛の長さを設定します。*t* は、ポイント単位での主目盛の長さです。この値は -100 から 50 の間でなければなりません。

t=0～50：　目盛ラベルとカラーボックスの間に目盛線を引きます。

t=-1：　自動目盛長（デフォルト）は、目盛ラベルのフォントサイズの 70% に相当します。目盛ラベルと色ボックスの間に目盛を引きます。

t=-2～-50：　目盛ラベルに最も近いカラーボックスの端を横切る目盛を引きます。目盛の実際の合計長さは -t です。

t=-100～-51：　目盛ラベルに最も近いカラーボックスの内側に目盛を引きます。実際の目盛の長さは -(t +50) です。例えば、-58 を指定すると、長さ 8 ポイントの内側の目盛が描画されます。

tickThick=*t*　　*t* は、目盛の太さをポイント単位で指定します（0～5 ポイント）。デフォルトは 1 ポイントです。小数点以下の値も指定可能です。

t=0 に設定すると、目盛は非表示になりますが、目盛ラベルは表示されたままになります。

tickUnit=*tu*　　*tu*=0：　単位のラベルを再び表示します。

tu=1：　目盛に付いている単位ラベルを非表示にします。

userTicks={*tvWave*, *tlblWave*}
主軸に対して、ユーザー定義の目盛位置とラベルを提供します。*tvWave* には数値の目盛位置が格納されており、テキストウェーブ *tlblWave* にはそれに対応するラベルが格納されています。

nticks で指定された通常の目盛間隔を上書きします。

詳細は、ヘルプ User Ticks from Waves を参照してください。

目盛ラベルは複数行にまたがって表示でき、書式設定されたテキストを使うこともできます。詳細は、Fancy Tick Mark Labels を参照してください。

ZisZ=*z*　　*z*=1 に設定すると、ゼロの目盛ラベル（存在する場合）は、他のラベルに使われる桁数に関係なく、1 桁の「0」として表示されます。デフォルトは *z*=0 です。

第 2 の軸パラメーター

axisLabel2=*axisLabelString2*
第 2 の軸の軸ラベルです。この軸ラベルは、userTicks2 が有効な場合にのみ描画されます。\r 文字以降のテキストは、\r 文字と同様に無視されます。デフォルトは "" です。

lblLatPos2=*p*　　第 2 の軸のラベルの横方向のオフセットを設定します。これは軸と平行なオフセットです。*p* はポイント単位です。正の値は、縦軸の場合は下方向、横軸の場合は右方向を示します。デフォルトは 0 です。

lblMargin2=*m*　　*m* は、グラフの通常の位置から副軸ラベルを移動させる距離（単位：ポイント、デフォルトは 0）です。デフォルト値は -5 で、これにより軸ラベルが軸に近づきます。軸ラベルを軸から遠ざけるには、正の値を使ってください。

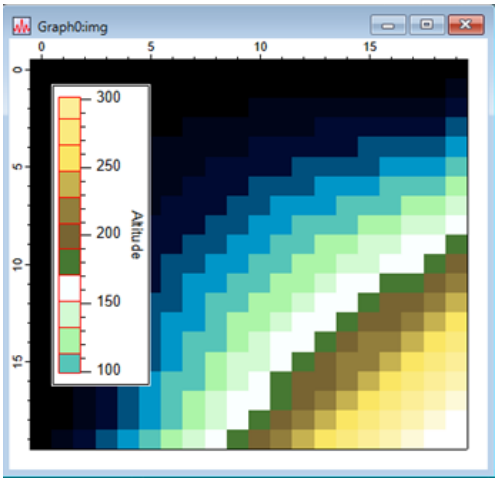
lblRot2=*r*　　第 2 の軸のラベルを、通常の向きを基準として反時計回りに *r* 度回転させます。*r* の値は -360 から 360 までの範囲です。

userTicks2={*tvWave*, *tlblWave*}

主軸とは常にカラーバーを挟んで反対側にある第２軸に対して、ユーザー定義の目盛位置とラベルを設定します。目盛ラベルは複数行にまたがることができ、スタイル付きテキストを使用できます。詳細は、ヘルプ Fancy Tick Mark Labels を参照してください。これが第２軸を描画する唯一の方法です。

画像プロットの ColorScale の例

```
Make/O/N=(20,20) img=p*q; NewImage img                  // 画像を作成して表示
ColorScale                                              // デフォルトのカラースケールを作成
ColorScale/C/N=text0 nticks=3,minor=1,"Altitude"        // 最初の注釈は text0
ModifyImage img ctab= {*,*,Relief19,0}                  // 19 色のカラーテーブル
ColorScale/C/N=text0 axisRange={100,300}                // 100-300 の範囲の詳細
ColorScale/C/N=text0 colorBoxesFrame=1                  // カラーボックスにフレームを付ける
ColorScale/C/N=text0 frameRGB=(65535,0,0)               // 赤いフレーム
```



Gizmo プロットの ColorScale の例

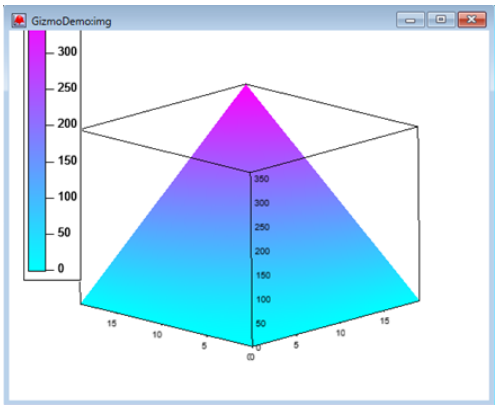
<プロシージャウィンドウ>

```
Window GizmoDemo() : GizmoPlot
    PauseUpdate; Silent 1
    // Gizmo ウィンドウの構築
    NewGizmo/W=(35,45,435,353)
    ModifyGizmo startRecMacro=700
    ModifyGizmo scalingOption=63
    AppendToGizmo Surface=root:img,name=surface0
    ModifyGizmo ModifyObject=surface0,objectType=surface,property={ srcMode,0}
    ModifyGizmo ModifyObject=surface0,objectType=surface,property={ surfaceCTab,CyanMagenta}
    AppendToGizmo Axes=boxAxes,name=axes0
    ModifyGizmo ModifyObject=axes0,objectType=Axes,property={-1,axisScalingMode,1}
    ModifyGizmo ModifyObject=axes0,objectType=Axes,property={-1,axisColor,0,0,0,1}
    ModifyGizmo ModifyObject=axes0,objectType=Axes,property={0,ticks,3}
    ModifyGizmo ModifyObject=axes0,objectType=Axes,property={1,ticks,3}
    ModifyGizmo ModifyObject=axes0,objectType=Axes,property={2,ticks,3}
    ModifyGizmo modifyObject=axes0,objectType=Axes,property={-1,Clipped,0}
    ModifyGizmo setDisplayList=0, object=surface0
    ModifyGizmo setDisplayList=1, object=axes0
    ModifyGizmo autoscaling=1
    ModifyGizmo currentGroupObject=""
    ModifyGizmo showInfo
```

```
    ModifyGizmo infoWindow={551,23,1368,320}
    ModifyGizmo endRecMacro
    ModifyGizmo SETQUATERNION={0.531194,-0.220652,-0.320213,0.752734}
    ColorScale/C/N=text0/X=2.75/Y=20.45 surfaceFill=surface0
EndMacro
```

<コマンドウィンドウ>

```
Make/O/N=(20,20) img=p*q; NewImage img
GizmoDemo()
```



ColorTab2Wave *colorTableName*

ColorTab2Wave コマンドは、組み込みのカラーテーブルから色を抽出し、それらを M_colors という名前の、赤、緑、青の列からなる N×3 の行列に格納します。
値は符号なし 16 ビット整数で、0 から 65535 までの範囲です。

N は通常 100 ですが、9 から 476 までの範囲になる場合があります。
実際の色数を決定するには：

```
Variable N= DimSize(M_colors,0)
```

を使います。

ウェーブ M_colors は、現在のデータフォルダー内に作成されます。
赤は列 0、緑は列 1、青は列 2 にあります。

パラメーター

colorTableName には、CTabList 関数が返す値（Grays や Rainbow など）のいずれかを指定できます。

colorTableName には「Igor」または「IgorRecent」を指定することもでき、これにより、Igor Pro のカラーメニューから 128 色の標準色、あるいは 0～32 のユーザー指定色をそれぞれ取得できます。

詳細

ヘルプ Color Tables