

CONTENTS

ビジュアルヘルプ - エクスperiment・ファイル・フォルダー	3
エクスperiment	3
エクスperimentの保存	3
バックされたエクスperimentファイルとして保存	3
HDF5 バックされたエクスperimentファイルとして保存	3
エクスperimentファイル保存のための HDF5 圧縮	4
バックされていないエクスperimentファイルとして保存	5
エクスperimentを開く	7
現在のエクスperimentに関する情報の入手	7
エクスperimentのマージ	8
エクスperimentを元に戻す	9
新しいエクスperiment	9
エクスperimentをテンプレートとして保存	10
エクスperimentのブラウズ	10
ファイル名とパス	10
シンボリックパス	11
New Symbolic Path ダイアログ	11
シンボリックパスの例	11
自動的に作成されるパス	13
Symbolic Path Status ダイアログ	14
Kill Path ダイアログ	14
ファイルとフォルダーへの参照	14
共有された Igor バイナリウェーブファイルの回避	15
ファイルの取り込み	15
ウェーブの取り込み	16
ノートブックとプロシージャファイルの取り込み	16
すべてを取り込む	17
すべてのスタンドアロンファイルの保存	17
エクスperimentの読み込まれ方	17
エクスperiment再作成プロシージャ	18
エクスperimentの初期化コマンド	18
エクスperimentの読み込み中のエラー	19
Igor Pro がフォルダーの欠落に対処する方法	19
エクスperimentの保存のされ方	21

実験の保存エラー.....	21
Windows ファイルのトラブルシューティング.....	22
特別なフォルダー.....	22
Igor Pro Folder.....	22
Igor Pro User Files.....	23
Igor Help Files フォルダー.....	23
Igor Extensions フォルダー.....	23
Igor Procedures フォルダー.....	24
User Procedures フォルダー.....	24
WaveMetrics Procedures フォルダー.....	25
追加の WaveMetrics ファイルの有効化.....	25
WaveMetrics のプロシージャファイルの有効化.....	25
WaveMetrics XOP の有効化.....	25
その他のファイルの有効化.....	26
複数ユーザーのシナリオでのファイルの有効化.....	26
Igor Pro のファイル処理.....	26
Open or Load File ダイアログ.....	26
Recent Files と Recent Experiments.....	27
デスクトップでのドラッグ&ドロップ.....	27
IGOR64 エクスperimentファイル.....	28
Autosave (自動セーブ).....	29
Autosave の対象.....	30
自動保存は、手動での保存やバックアップの代わりではない.....	30
Autosave のモード.....	31
実験全体の自動保存を設定する変更点.....	32
自動保存の実行を強制する.....	33
自動保存なしでスタンドアロンファイルを保存する.....	33
Indirect Autosave モード.....	33
.autosave ファイルからファイルを復旧する.....	34
Indirect Autosave モードでの問題.....	34
.autosave ファイルを明示的に開く.....	35
Indirect Autosave とバックされていない実験.....	35
Indirect Autosave の復元機能を試す.....	35

ビジュアルヘルプ – エクスペリメント・ファイル・フォルダー

本ドキュメントでは、Igor Pro の「エクスペリメント」機能と、それがファイルやフォルダーとどのように関連しているかについて説明します。

エクスペリメント

「エクスペリメント」とは、ウェーブ、変数、グラフ、テーブル、ページレイアウト、ノートブック、コントロールパネル、プロシージャなどの Igor Pro オブジェクトの集合です。

これらのオブジェクトのいずれかを作成または変更すると、現在のエクスペリメントが変更されます。

File→Save Experiment を選択すると、現在のエクスペリメントを保存できます。

デスクトップ上のアイコンをダブルクリックするか、File→Open Experiment を選択すると、エクスペリメントを開くことができます。

エクスペリメントの保存

エクスペリメントをディスクに保存するには、2つの形式があります。

- ・ パックされたエクスペリメントファイルとして。ファイルの拡張子は .pxp です。
明示的に個別に保存しない限り、すべてのウェーブ、プロシージャウィンドウ、ノートブックは、エクスペリメントファイルにまとめて保存されます。
- ・ エクスペリメントファイルとエクスペリメントファイルフォルダ（パックされていない形式）として。ファイルの拡張子は .uxp です。
すべてのウェーブ、プロシージャウィンドウ、ノートブックは、それぞれ個別のファイルとして保存されます。

ほとんどの用途では、パック形式の使用を推奨します。

パックされていない形式は、非常に多くのウェーブ（数千以上）を含むエクスペリメントに役立ちます。

パックされたエクスペリメントファイルとして保存

パックされたエクスペリメントファイルでは、エクスペリメントに関するすべてのデータが1つのファイルに保存されます。

これにより、ディスクの容量を節約できるほか、エクスペリメントデータをディスク間で簡単にコピーできるようになります。

通常は、パックされたエクスペリメントファイル形式の使用を推奨します。

パックされたエクスペリメントファイルが含まれているフォルダーは、ホームフォルダーと呼ばれます。

新しいエクスペリメントをパック形式で保存するには、File メニューから Save Experiment を選択します。

HDF5 パックされたエクスペリメントファイルとして保存

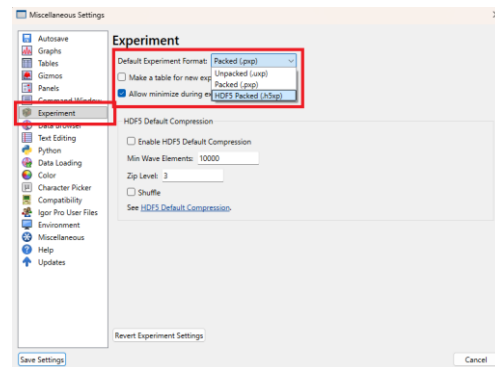
Igor Pro 9 以降では、Igor Pro エクスペリメントデータを HDF5 ファイルとして保存することができます。

主な利点は、HDF5 をサポートする多種多様なプログラムから、そのデータに即座にアクセスできることです。

主な欠点は、Igor Pro でそのファイルを開くには Igor Pro 9 以降が必要になることです。
また、ウェーブの数が非常に多いエクスペリメントの場合、HDF5 は PXP よりも処理速度がかなり遅くなります。

エクスペリメントを HDF5 パック形式で保存するには、File→Save Experiment As を選択します。
表示される Save Experiment As ダイアログで、ファイル一覧の下にあるポップアップメニューから「HDF5 Packed Experiment Files (*.h5xp)」を選択します。
その後、Save をクリックします。

新しいエクスペリメントデータを保存する時のデフォルト形式として HDF5 パック形式を設定したい場合は、メニュー Misc→Miscellaneous Settings を選択してください。
表示されるダイアログで、左側の「Experiment」カテゴリをクリックします。
次に、「Default Experiment Format」のポップアップメニューから「HDF5 Packed (.h5xp)」を選択します。



通常の使用においては、Igor Pro が HDF5 形式のパックされたエクスペリメントファイルにデータをどのように保存しているかといった詳細を知る必要はありません。

もし興味があれば、HDF5 ブラウザ（Data→Load Waves→New HDF5 Browser）を使ってそのようなファイルを開くことで、その仕組みを把握することができます。

他のプログラムから HDF5 パック形式のエクスペリメントファイルを読み書きしたいプログラマーの方は、ヘルプ HDF5 Packed Experiment Files の項目で技術的な詳細を確認できます。

エクスペリメントファイル保存のための HDF5 圧縮

Igor Pro を初めて使う場合は、慣れてくるまで、このセクションは飛ばしておくことをお勧めします。

Igor Pro 9 以降では、HDF5 形式のエクスペリメントファイルを圧縮形式で保存することができます。
このセクションでは、エクスペリメントファイルの保存に関連する HDF5 圧縮について簡単に説明します。
圧縮の詳細については、ヘルプ HDF5 Compression を参照してください。

圧縮を行うとファイルサイズは小さくなりますが、処理に時間がかかります。
圧縮を行う価値があるかどうかは、データの性質や、ファイルサイズと処理時間のバランスをどう取るかによって異なります。

エクスペリメントデータに、ノイズの多い比較的小さなウェーブが多数含まれている場合、圧縮を行ってもディスク容量の節約効果はほとんどありません。

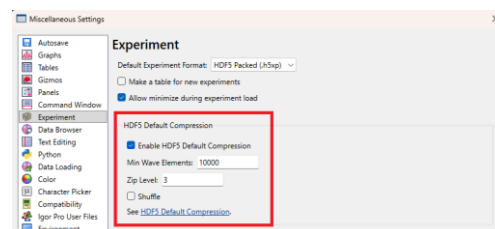
エクスペリメントデータに、1つの値で占められる部分が多い大きなウェーブ（例えば、大部分が黒で構成される画像など）が含まれている場合、圧縮によってかなりのディスク容量を節約できます。

圧縮が自分にとって有益かどうかを判断する唯一の方法は、実際に試してみることです。

File メニューから、圧縮された HDF5 形式のエクスペリメントデータを保存するには、HDF5 のデフォルト圧縮を有効にし、その後、Miscellaneous Settings ダイアログの Experiment セクションで3つのパラメータを指定する必要があります。

パラメータは以下の通りです：

- ・ 圧縮が適用されるためのウェーブの最小サイズ
- ・ 圧縮レベルは 0（圧縮なし）から 9（最大）まで
- ・ シャッフル（圧縮前のオプション処理）を有効にするかどうか



エクスペリメントファイルを使って HDF5 の圧縮をテストする方法について説明します。

この手順では、File→Save Experiment ではなく、TestHDF5ExperimentCompression 関数による SaveExperiment コマンドを使うため、Miscellaneous Settings ダイアログのデフォルトの圧縮設定とは無関係です。

1. 「Test HDF5 Experiment Compression.ipf」プロシージャファイルをグローバル・プロシージャファイルとして有効にし、Igor Pro を再起動します。
このファイルは、Igor Pro フォルダ内の「WaveMetrics Procedures/File Input Output」ディレクトリにあります。
やり方がわからない場合は、「WaveMetrics プロシージャファイルの有効化」のセクションを参照してください。
2. 一般的なエクスペリメントファイルを開きます。
3. 次を実行します。

```
TestHDF5ExperimentCompression(10000, 2, 0)
```

圧縮対象となるウェーブの要素数は、10000 以上である必要があります。

これより少ない要素数のウェーブは、圧縮されずに保存されます。

2 は ZIP 圧縮レベルです。

0 はシャッフルがオフであることを意味し、通常はこれが望ましい設定です。

TestHDF5ExperimentCompression コマンドは、現在のエクスペリメントデータを、指定されたパラメータを使って非圧縮形式と圧縮形式の両方で保存し、コマンドウィンドウの履歴エリアに、圧縮がエクスペリメントデータの保存時間やファイルサイズに与える影響を示すメッセージを表示します。

このコマンドは保存したデータを削除するため、不要なデータが残ることはありません。

最小ウェーブ長や ZIP 圧縮レベルについて、さまざまな設定を試して、保存時間や圧縮率にどのような影響があるかを確認してください。

多くの場合、ZIP 圧縮レベルを高くしても圧縮率はわずかに向上するに過ぎず、それに要する時間を割く価値はありません。

大幅な圧縮率が見込めない限り、圧縮は控えることを推奨します。

通常、ディスク容量は豊富にあり、時間のほうが貴重です。

File メニューから HDF5 形式のエクスペリメントファイルを保存する場合、デフォルトでは圧縮は無効になっています。

圧縮を使いたい場合は、ヘルプ HDF5 Default Compression を参照し、Miscellaneous Settings ダイアログから圧縮を有効にする方法を確認してください。

パックされていないエクスペリメントファイルとして保存

パックされていない形式では、エクスペリメントはエクスペリメントファイルとエクスペリメントフォルダーとして保存されます。

ファイルには、Igor Pro がエクスペリメントを再現するために使う指示が含まれており、フォルダーにはデータを読み込むためのファイルが含まれています。

エクスペリメントフォルダーは、ホームフォルダーとも呼ばれます。

この形式の主なメリットは、非常に多くのウェーブ（数千以上）を含むエクスペリメントにおいて処理が高速であることです。

ただし、パックされていない形式はデータが破損しやすいため、日常的な使用には推奨されません。

新しいエクスペリメントをパックされていない形式で保存するには、File メニューから Save Experiment As を選択します。

表示される Save Experiment As ダイアログの下部にあるポップアップメニューから、Unpacked Experiment Files を選択します。

Save をクリックすると、は拡張子が「.uxp」のパックされていないエクスペリメントファイルを保存します。

Igor Pro は、エクスペリメントファイル名に「Folder」またはその日本語相当語を付加することで、エクスペリメントフォルダー名を自動的に生成します。

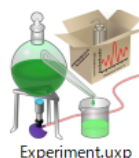
その後、ユーザーによる追加の操作を必要とせずに、解凍されたエクスペリメントフォルダーを作成します。

例えば、パックされていないエクスペリメントファイル名として「Test.uxp」を入力した場合、自動的に「Test Folder」またはその日本語相当語を、パックされていないエクスペリメントフォルダー名として使います。

「Test Folder」という名前のフォルダーがすでに存在する場合、そのフォルダーをパックされていないエクスペリメント用に再利用するかどうかの確認メッセージを表示します。

パックされていないエクスペリメントフォルダー名の自動生成に問題がある場合は、SaveExperiment /F コマンドを使って、任意の名前でエクスペリメントを保存することができます。

この図は、パックされていないエクスペリメントで使われるアイコンを示し、各要素がどこに保存されているかを説明しています。



これには、Igor Pro がエクスペリメントを再作成する時に実行する起動コマンドが含まれており、すべてのエクスペリメントウィンドウも含まれます。



Experiment
Folder

これには、ウェーブ、変数、履歴、プロシージャ、ノートブック、その他のオブジェクト用のファイルが含まれています



history



miscellaneous



Procedure



variables

通常、エクスペリメントフォルダー内のファイル进行操作する必要はありません。

エクスペリメントを保存すると Igor Pro が自動的にそれらを書き込み、エクスペリメントを開くと自動的に読み込みます。

エクスペリメントにルートデータフォルダー以外のデータフォルダー（ヘルプ Data Folders を参照）が含まれている場合、エクスペリメントフォルダー内に、エクスペリメントに含まれるデータフォルダーごとに1つのサブフォルダーを作成します。

各ウェーブにつき1つのファイルが作成される点に注意してください。

これらは Igor バイナリデータファイルであり、ウェーブデータをコンパクトな形式で保存しています。

プログラマーの参考として、Igor バイナリウェーブファイル形式については、テクニカルノート #003 に詳細が記載されています。

「procedure」ファイルには、エクスペリメントの組み込みプロシージャウィンドウに表示されるテキストが格納されています。

「variables」ファイルには、エクスペリメントの数値変数および文字列変数がバイナリ形式で保存されます。

「miscellaneous」ファイルには、画像、ページ設定の記録、XOP 設定、その他のデータが保存されます。

このバックされていない形式のエクスペリメントのメリットは以下の通りです。

- ・ 変更されていないウェーブ、プロシージャ、またはノートブックのファイルを更新する必要があるため、エクスペリメントデータをより迅速に保存できます。
- ・ あるエクスペリメントに保存されたファイルを、別のエクスペリメントと共有することができます。ただし、エクスペリメントを別のディスクに移動すると、ファイルの共有によって問題が発生する可能性があります。詳細については、「ファイルとフォルダへの参照」のセクションを参照してください。

バックされていないエクスペリメント形式のデメリットは以下の通りです。

- ・ 特に小さなウェーブが多いエクスペリメントの場合、より多くのディスク容量が必要になります。
- ・ エクスペリメントを別のディスクに移動する時は、エクスペリメントファイルとフォルダーを一緒に保存する必要があります。

エクスペリメントを開く

File メニューから Open Experiment を選択すると、ディスクに保存されているエクスペリメントを開くことができます。

現在のエクスペリメントに修正を加えている場合は、まず保存してください。

その後、Select an experiment ダイアログを表示します。

エクスペリメントファイルを選択して Open ボタンをクリックすると、Igor Pro はそのエクスペリメントをロードします。

これには、エクスペリメントを構成するすべてのウェーブ、変数、グラフ、テーブル、ページレイアウト、ノートブック、プロシージャ、その他のオブジェクトが含まれます。

エクスペリメントの読み込み方法の詳細については、「エクスペリメントの読み込む方法」のセクションを参照してください。

現在のエクスペリメントに関する情報の入手

File→Experiment Info を選択すると、現在のエクスペリメントに関する概要情報を確認できます。

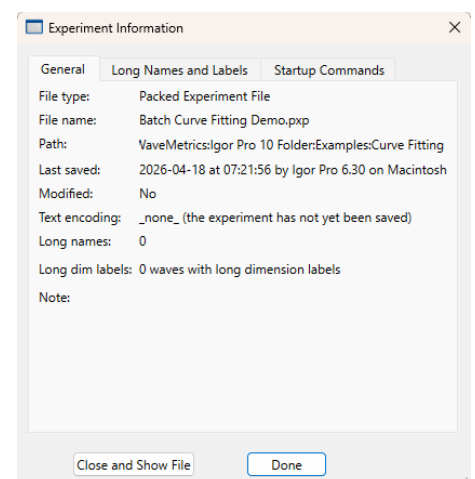
これにより、Experiment Information ダイアログが表示されます。

このダイアログには、ファイルが最後に保存された日時、前回の保存以降に変更があったかどうか、およびその他の一般的な情報が表示されます。

このダイアログには、エクスペリメントで長い名前、可変名、データフォルダー、ターゲットウィンドウ、またはシンボリックパス名が使われているのかも表示されます。

長い名前を使うエクスペリメントには、Igor Pro 8.0 以降が必要です。

詳細については、ヘルプ Long Object Names を参照してください。



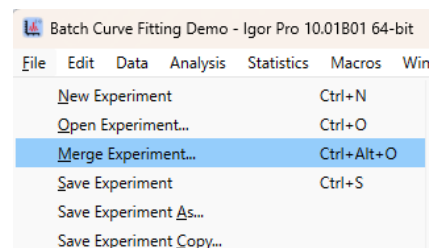
エクスペリメントのマージ

通常、Igor Pro は新しいエクスペリメントを開く前に、現在開いているエクスペリメントを閉じます。

しかし、エクスペリメントファイルの内容を現在のエクスペリメントに統合することも可能です。

これは、例えば2つ以上のエクスペリメントのグラフを含むページレイアウトを作成したい場合などに便利です。

これを行うには、Alt キーを押しながら、File メニューから Merge Experiment を選択します。



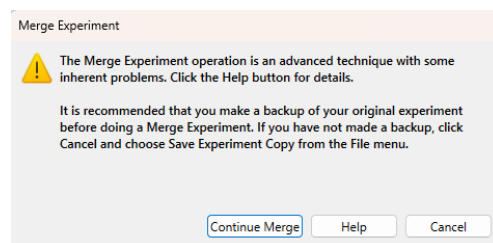
注記：

エクスペリメントのマージは高度な機能であり、固有の問題も伴うため、慎重に使う必要があります。

Igor Pro の使い方を学び始めたばかりの方は、操作に習熟するまではエクスペリメントのマージを避けるべきです。

このセクションの残りの部分はざっと目を通すか、あるいは完全にスキップすることをお勧めします。

このセクションは、Igor Pro に精通していることを前提としています。



最初の問題は、マージ操作を行うと、バックされたエクスペリメントファイルに保存されているデータやその他のオブジェクト（グラフ、プロシージャファイル、ノートブックなど）のコピーが作成されてしまうことです。コピーを作成するたびに、それらの内容が異なるものになり、どちらが「本来の」データやオブジェクトなのかが分からなくなる可能性があります。

この問題を回避する一つの方法は、目的を果たした後に、マージされたエクスペリメントを破棄することです。

2つ目の問題は、Igor Pro があらゆる種類のデータ、プロシージャ、その他のオブジェクトを参照するために名前を使っている点に関係しています。

エクスペリメント B をエクスペリメント A にマージする時、名前の競合が発生する可能性があります。

Igor Pro は、エクスペリメント B のデータを格納するための新しいデータフォルダーを作成することで、データ（ウェーブ、数値変数、文字列変数）の名前衝突を防ぎます。

新しいデータフォルダーは、現在のエクスペリメント（この場合はエクスペリメント A）の現在のデータフォルダー内に作成されます。

グラフ、テーブル、ページレイアウト、コントロールパネル、ノートブック、Gizmo プロット、シンボリックパス、ページ設定、画像など、その他のグローバルに名前が付けられたオブジェクトについては、Igor Pro は名前の競合を避けるため、必要に応じてエクスペリメント B のオブジェクトの名前を変更します。

ページレイアウトは、グラフ、テーブル、Gizmo プロット、画像を名前で参照しているため、これらのオブジェクトのいずれかの名前を変更すると、ページレイアウトに影響が出る可能性があります。

エクスペリメントのマージ操作は、エクスペリメント B で開かれているページレイアウトについて、この問題に対処します。

ただし、エクスペリメント B 内のページレイアウト再作成マクロで、対応する開いているウィンドウがないものについては、この問題に対処しません。

ウィンドウ再作成マクロ以外のプロシージャで名前の競合がある場合、エクスペリメントのマージ操作の完了後にプロシージャをコンパイルする時、エラーを通知します。

競合しているプロシージャを削除するか名前を変更し、手動で名前の競合を解決する必要があります。

プロシージャウィンドウにはタイトルがありますが、標準的な Igor 名はありません。

エクスペリメントのマージ操作では、同じタイトルを持つプロシージャウィンドウのタイトルを変更しようとはしません。

エクスペリメント B のメインプロシージャウィンドウの内容が、エクスペリメント A のメインプロシージャウィンドウの内容に追加されます。

通常のエクスペリメントの開始時、Igor Pro はエクスペリメントの初期化コマンドを実行します。
エクスペリメントのマージの時には、この処理は行われません。

各エクスペリメントには、グラフやページレイアウトに影響を与えるデフォルトのフォント設定が含まれています。
エクスペリメントをマージすると、エクスペリメント B のデフォルトのフォント設定は無視され、エクスペリメント A のデフォルトのフォント設定がそのまま維持されます。
これにより、エクスペリメント B のグラフやレイアウトの外観に影響が出る可能性があります。

エクスペリメント B の履歴はエクスペリメント A にマージされません。
その代わりに、履歴エリアにエクスペリメントのマージプロセスに関するメッセージが表示されます。

エクスペリメント B のシステム変数 (K0...K19) は無視され、エクスペリメント A にはマージされません。

エクスペリメントのマージ操作では、ほとんどの一般的な名前の競合の問題は処理できますが、処理できない問題もいくつかあります。

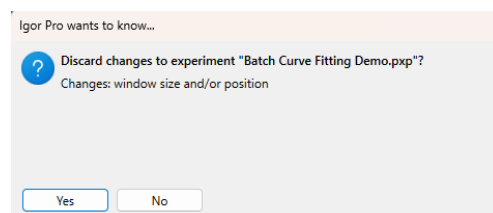
例えば、エクスペリメント B のデータにフルパスを使って参照しているプロシージャ、依存関係式、またはコントロールは、マージ中にエクスペリメント B のデータが新しいデータフォルダーに読み込まれるため、期待どおりに動作しない可能性があります。

別の例として、名前衝突のためにマージ操作によって名前が変更されたウィンドウ、シンボリックパス、または画像を参照するプロシージャが挙げられます。

名前衝突によって予期しない動作を引き起こす可能性のある状況は、他にも間違いなく数多く存在します。

エクスペリメントを元に戻す

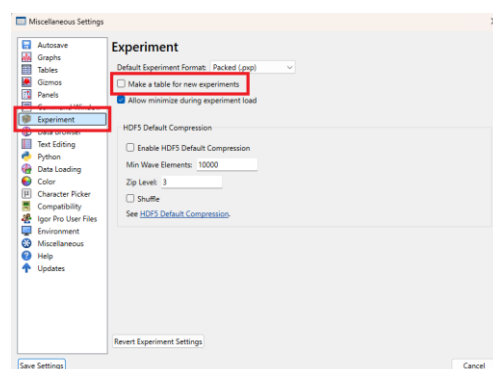
File メニューから Revert Experiment を選択すると、現在のエクスペリメントへの変更を破棄してもよいか確認します。
「Yes」と答えると、ディスクから現在のエクスペリメントを読み込み、最後に保存した時点の状態に復元します。



新しいエクスペリメント

File メニューから New Experiment を選択すると、現在のエクスペリメントが前回保存してから変更が加えられている場合、まずそれを保存するかどうかを確認します。
その後、Igor Pro は新しい空のエクスペリメントを作成します。
新しいエクスペリメントは、保存するまではエクスペリメントファイルとして保存されません。

デフォルトでは、新しいエクスペリメントを作成すると、自動的に新しい空のテーブルを作成します。
通常、手動でデータを入力することから作業を始める場合には、これが便利です。
ただし、Igor Pro では、テーブルに表示されなくても、データがメモリ内に存在することがあります。
必要に応じて、Miscellaneous Settings ダイアログ (Misc メニュー) の Experiment カテゴリから、テーブルの自動作成を無効にすることができます。



エクスペリメントをテンプレートとして保存

テンプレートエクスペリメントを使うと、新しいエクスペリメントの初期内容をカスタマイズすることができます。テンプレートエクスペリメントを開くと、Igor Pro は通常通りそれを開きますが、タイトルは未設定のままにし、テンプレートエクスペリメントファイルとの関連付けを解除します。

これにより、プロトタイプに基づいた新しいエクスペリメントが作成されます。

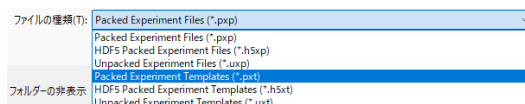
この未タイトルのエクスペリメントを保存すると、Igor Pro は新しいエクスペリメントファイルを作成します。

バックされたテンプレートエクスペリメントのファイル名拡張子は「.pxp」ではなく「.pxt」です。

バックされていないテンプレートエクスペリメントのファイル名拡張子は「.uxp」ではなく「.uxt」です。

テンプレートエクスペリメントを作成するには、まず新しいエクスペリメントで、ウェーブ、変数、プロシージャ、その他のオブジェクトを自由に配置してプロトタイプエクスペリメントを作成します。

次に、File→Save Experiment As を選択し、ファイル形式のポップアップメニューから Packed Experiment Template または Unpacked Experiment Template を選んで、テンプレートエクスペリメントを保存します。



既存のエクスペリメントファイルの拡張子を変更することで（「.pxp」を「.pxt」に、または「.uxp」を「.uxt」に）、テンプレートファイルに変換することができます。

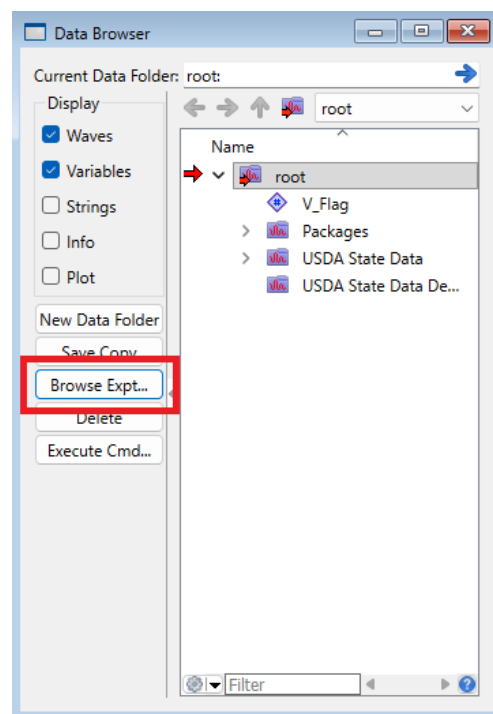
エクスペリメントのブラウズ

データブラウザーを使うと、現在のエクスペリメントのデータだけでなく、ディスクに保存されているエクスペリメントのデータも確認できます。

ブラウザーを開くには、Data→Data Browser を選択します。

次に、Browse Expt ボタンをクリックします。

詳細については、ヘルプ The Data Browser を参照してください。



ファイル名とパス

Igor Pro は、最大 255 バイトまでのファイル名に対応しています。

ファイルパスは最大 2000 バイトまで指定できます。

オペレーティングシステムがそれより長いパスをサポートしている場合でも、Igor Pro の 2000 バイトという制限が適用されます。

Windows では、2016 年 8 月にリリースされた Windows 10 ビルド 1607 以前のバージョンにおいて、オペレーティングシステムはパスの長さを 260 バイトに制限していました。

ビルド 1607 では、この制限が 32767 文字に引き上げられました。

ヘルプ Path Separators の項で説明されているように、Igor Pro 9 以前の Igor Pro はどちらのプラットフォームでも、コロンまたはバックスラッシュを含むパスを受け付けます。

Igor Pro ではリテラル文字列内でバックスラッシュがエスケープ文字として使われるため、バックスラッシュの扱いは複雑になります。

これについては、ヘルプ Path Separators の項でも詳しく説明しています。

この問題に対する最も簡単な解決策は、Windows 上で実行している場合でも、パス要素の区切り文字としてコロン(:)を使うことです。

ファイルパスの一部を抽出したり、ファイルパスを操作したりする必要があるプロシージャを作成する場合、ParseFilePath 関数が役立つことがあります。

シンボリックパス

シンボリックパスとは、ディスクドライブ上のフォルダーに短い名前を関連付ける Igor オブジェクトのことです。ファイルの読み込み、開く、保存を行う時、フルパスの代わりにこの短い名前を使ってフォルダーを指定することができます。

ディスクの構成が変更された場合や、ディスク上の別の場所にアクセスしたい場合に、変更が容易であるため、可能な限りフルパスではなくシンボリックパスを使うことを推奨します。

Igor Pro は、「Igor Pro Folder」フォルダー、「Igor Pro User Files」フォルダー、現在のエクスペリメントのホームフォルダーへのシンボリックパスを自動的に作成します。

これらのシンボリックパスは、すべてのエクスペリメントに存在します。

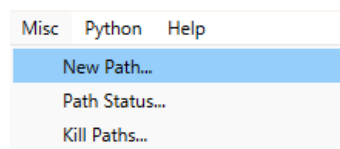
ディスク上の任意のフォルダーを指すカスタムシンボリックパスを作成することもできます。

これらのシンボリックパスは現在のエクスペリメントの一部であり、新しいエクスペリメントを作成すると削除されます。

New Symbolic Path ダイアログ

New Symbolic Path ダイアログを開くには、Misc メニューから New Path を選択します。

このダイアログの使い方については、次のセクションで説明します。



シンボリックパスの例

この例では、シンボリックパスを使うべき理由とその使い方を説明します。

ここでは、Igor Pro でグラフ化したいデータを含むテキストファイルが格納されたフォルダーがあり、ハードディスクの構成が以下の通りであると仮定します。

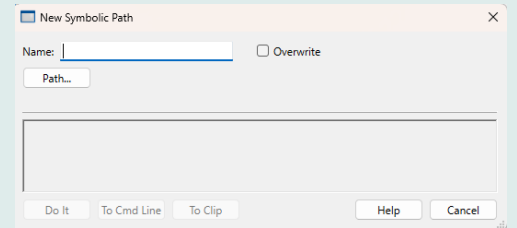
```
C:
  Users
    Jack
      Documents
        Data
```

2016
May
June
July

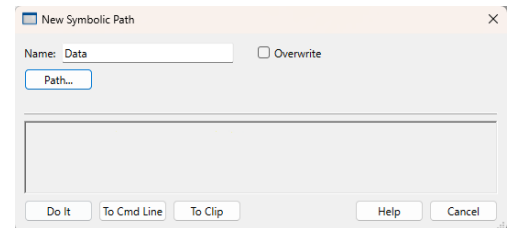
「June」フォルダー内のファイルにアクセスしたいと仮定します。

フォルダーへのシンボリックパスを作成するには：

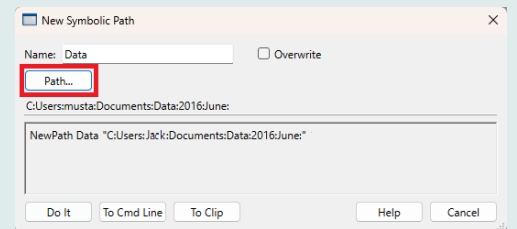
1. Misc メニューから New Path を選択します。これにより、New Symbolic Path ダイアログが表示されます。



2. Coefficients タブを選択します。Name フィールドに Data と入力します。



3. Path ボタンをクリックし、June フォルダーを選択します。



4. Overwrite チェックボックスをオンにします。



5. Do It をクリックすると、シンボリックパスが作成されます。

ダイアログで作成された NewPath コマンドは、「Data」という名前のシンボリックパスを作成し、以下を参照します。

```
C:\Users\Jack\Documents\Data\2024\June
```

このダイアログで生成されるコマンドは、各要素をコロンで区切る Macintosh 形式のパスを使います。

```
NewPath/O Data, "C:Users:Jack:Documents:Data:2024:June"
```

NewPath 関数では、バックスラッシュを含む Windows 形式のパスも指定できますが、問題が発生する可能性があるため、推奨されません。

詳細については、ヘルプ Path Separators を参照してください。

作成が完了すると、ファイルを選択する必要があるダイアログで、そのデータパスを選択できるようになります。

例えば、Load Waves ダイアログでは、[Symbolic Path] リストから [Data] を選択できます。
その後、[File] ボタンをクリックして、読み込むファイルを選択します。
Igor Pro は、例えば、次のようなコマンドを生成します。

```
LoadWave /J /P=Data "Data1.txt"  
LoadWave /J /P=Data "Data1.txt"  
LoadWave /J /P=Data "Data3.txt"
```

これらのコマンドは、June フォルダーからデータファイルを読み込みます。

読み込むファイルへのフルパスではなくシンボリックパスを使うことで、データファイルの場所を「シンボリックパス」という単一のオブジェクトに集約しました。

これにより、シンボリックパスを使うコマンドやプロシージャの宛先を簡単に変更できるようになります。
例えば、NewPath コマンドを再実行する時、June を July に置き換えることができます。

```
NewPath/O Data, "C:\User:Jack:Documents:Data:2024:July"
```

LoadWave コマンドを再度実行すると、June ではなく July のデータが読み込まれます。

シンボリックパス内でディスク上の特定の場所を指定しておけば、ユーザー間やマシン間で作業を切り替える時にも非常に便利です。

多くのコマンドでフルパスを変更する必要がなく、シンボリックパスを変更するだけで済みます。

自動的に作成されるパス

Igor Pro は、Igor Pro 10 Folder を指す「Igor」という名前のシンボリックパスを自動的に作成します。
この Igor シンボリックパスは、Igor Pro Folder 内のファイルにアクセスしたいという、ごく稀な場合にのみ役立ちます。

また、Igor Pro は「IgorUserFiles」という名前のシンボリックパスを自動的に作成します。
これは Igor Pro の「User Files」フォルダーを指しています。
詳細については、「特別なフォルダー」のセクションを参照してください。

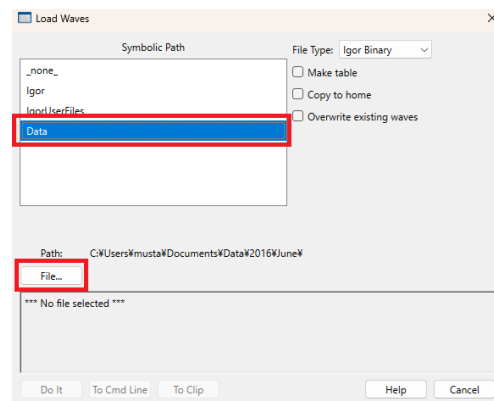
Igor Pro は、ホームのシンボリックパスも自動的に作成します。
このパスは、現在のエクスペリメントのホームフォルダーを指します。
バックされていないエクスペリメントの場合、これはエクスペリメントフォルダーになります。
バックされたエクスペリメントの場合、これはエクスペリメントファイルが含まれるフォルダーになります。
まだ保存されたことのない新しいエクスペリメントの場合、ホームは未定義となります。

最後に、現在のエクスペリメントで、そのエクスペリメントの一部として保存されていないファイルを参照するような操作を行った場合、Igor Pro は自動的にシンボリックパスを生成します。
これは、次のような場合に発生します：

- ・ 別のエクスペリメントから Igor バイナリウェーブファイルを現在のエクスペリメントに読み込む
- ・ 現在のエクスペリメントとは別の場所に保存されているノートブックファイルを開く
- ・ 現在のエクスペリメントと一緒に保存されていないプロシージャファイルを開く

これらのパスを作成しておくと、参照先のファイルが名前変更や移動された場合でも、それらを見つけやすくなります。

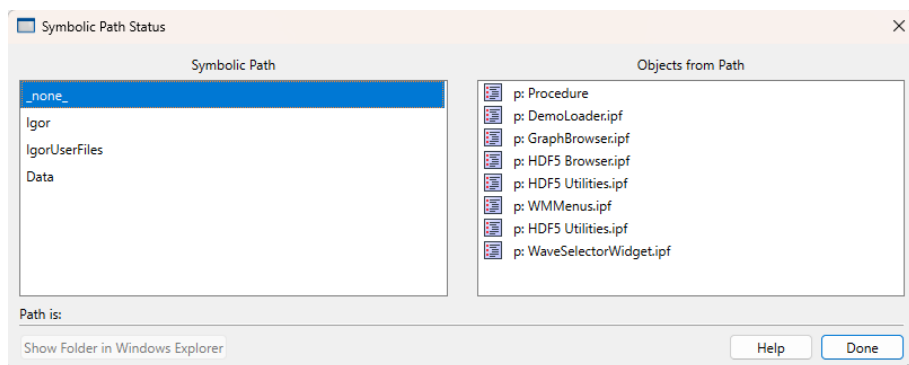
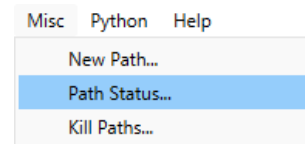
詳細については、「ファイルとフォルダーへの参照」のセクションを参照してください。



Symbolic Path Status ダイアログ

Symbolic Path Status ダイアログには、現在のエクスペリメントに存在するパスが表示されます。

このダイアログを表示するには、Misc メニューから Path Status を選択してください。



このダイアログには、指定されたシンボリックパスを通じて現在のエクスペリメントで参照されているウェブ、ノートブックファイル、プロシージャファイルも表示されます。

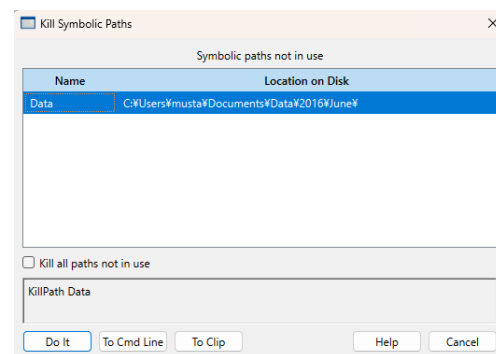
Kill Path ダイアログ

Kill Symbolic Paths ダイアログでは、現在のエクスペリメントから不要になったシンボリックパスを削除します。このダイアログを表示するには、Misc メニューから Kill Paths を選択します。

シンボリックパスを削除しても、そのパスが指すフォルダーには何の影響もありません。

単に、Igor Pro のシンボリックパス一覧からそのシンボリックパス名が削除されるだけです。

エクスペリメントに、シンボリックパスが指すフォルダー内のファイルにリンクされたウェブ、ノートブックウィンドウ、またはプロシージャウィンドウが含まれている場合、そのシンボリックパスは使用中となり、Igor Pro ではそれを削除することはできません。



ファイルとフォルダーへの参照

エクスペリメントでは、そのエクスペリメントと一緒に保存されていないファイルを参照することができます。これは、別のエクスペリメントと一緒に保存されている、あるいはどのエクスペリメントとも関連付けられていない Igor Pro バイナリデータファイルを読み込む場合に行います。

また、現在のエクスペリメントと一緒に保存されていないノートブックファイルやプロシージャファイルを開く場合にも行います。

このような場合、現在のエクスペリメントがそのウェブファイル、ノートブックファイル、またはプロシージャファイルを共有していると言います。

例えば、既存のテキストファイルをノートブックとして開き、そのエクスペリメントを保存したとします。このノートブックのデータは、ハードディスク上のどこかにあるテキストファイルの中に格納されています。エクスペリメントファイル自体にはデータは保存されません。エクスペリメントファイルに保存されるのは、そのファイルへの参照情報です。

具体的には、エクスペリメントファイルには、次にそのエクスペリメントを開いた時にノートブックファイルを再表示するためのコマンドが含まれています。

注記：

エクスペリメントが、そのエクスペリメントの一部として保存されていないファイルを参照している場合、問題が発生する可能性があります。

例えば、エクスペリメントを別のコンピュータに持ち出すために外部ドライブにコピーした場合、外部ドライブ上のエクスペリメントファイルには、ハードディスク上のファイルへの参照が含まれることになります。

別のコンピュータでそのエクスペリメントを開くと、Igor Pro は参照されているファイルを探すよう求めます。

参照されているファイルを別のコンピュータにもコピーし忘れた場合、Igor Pro はそのエクスペリメントを完全に再現することができません。

そのため、参照は必要な場合にのみ使用し、この潜在的な問題に留意することをお勧めします。

異なるプラットフォーム間でファイルを転送する場合、ファイルへの参照が特に厄介になることがあります。ヘルプ Experiments and Paths を参照してください。

共有された Igor バイナリウェーブファイルの回避

別のエクスペリメントに保存されている Igor バイナリウェーブファイルからウェーブを読み込む時は、そのウェーブを元のエクスペリメントと共有するか、新しいエクスペリメントにコピーするかを選択する必要があります。

共有を選択すると、現在のエクスペリメントからウェーブファイルへの参照が作成されますが、この参照が前述の問題を引き起こす可能性があります。

したがって、複数のエクスペリメントから同じデータにアクセスする必要があり、かつ前述の問題が発生するリスクを承知の上で共有を行う場合を除き、共有は避けるべきです。

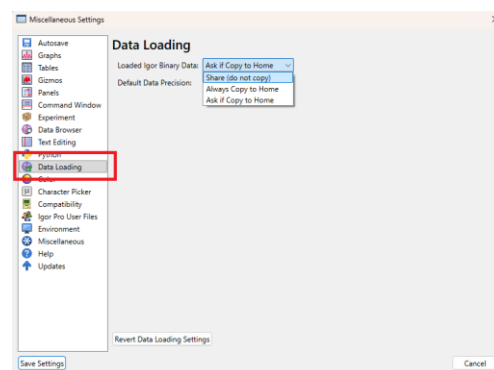
Load Igor Binary ダイアログからウェーブを読み込むと、Igor は「共有」するか「コピー」するかを尋ねてきます。

Miscellaneous Settings ダイアログ (Data Loading カテゴリ) を使えば、確認を求めずに常に「共有」または「コピー」するように設定できます。

コマンドラインまたはプロシージャから LoadWave コマンドを使ってウェーブを読み込む場合、操作の確認を求めません。

通常は LoadWave の /H フラグを使うべきです。

これにより、「ウェーブをホームにコピーする」ように指示され、共有が回避されます。



Data Browser を使ってあるエクスペリメントから別のエクスペリメントへウェーブを転送する場合、Igor Pro は常にウェーブをコピーします。

詳細については、ヘルプ Home Versus Shared Waves と Home Versus Shared Text Files を参照してください。

ファイルの取り込み

取り込み機能を使うと、共有されているウェーブファイル、ノートブック、またはプロシージャファイルを現在のエクスペリメントにコピーし、元のファイルとの関連付けを解除することができます。

取り込み機能により、エクスペリメントはより独立した構成となるため、障害に強くなります。

別のコンピュータに移行したり、同僚に送信したりする場合でも、エクスペリメントを再現するために必要なすべてのファイルがエクスペリメント自体に保存されます。

ウェーブの取り込み

ウェーブファイルを取り込むには、Data Browser でウェーブを選択し、右クリックして Adopt Wave を選択します。

また、File→Adopt All を使ってウェーブを適用することもできます。
以下の「すべてを取り込む」のセクションを参照してください。

ウェーブを取り込むと、そのウェーブを元のスタンドアロンファイルから切り離します。
元のファイルはそのまま残りますが、現在のエクスペリメントからは参照されなくなります。
エクスペリメントを保存するまで、取り込みは確定しません。

現在のエクスペリメントがパックされた形式で保存されている場合、ウェーブを取り込むと、そのウェーブはパックされたエクスペリメントファイルに保存されます。
パックされていないエクスペリメントの場合、そのウェーブはエクスペリメントフォルダ内の「waves」データフォルダーに対応するディスク上のフォルダーに保存されます。

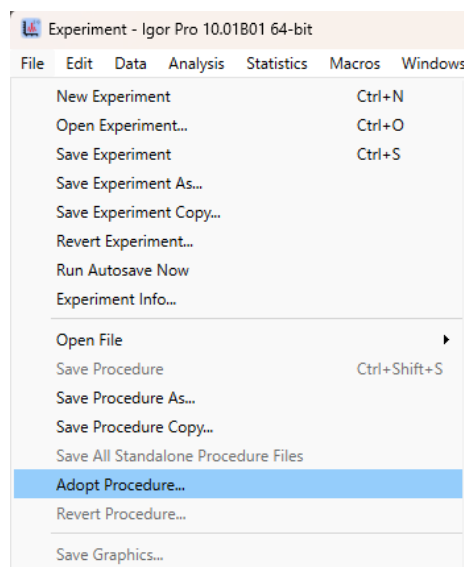
ノートブックとプロシージャファイルの取り込み

ノートブックまたはプロシージャファイルを取り込むには、File メニューから Adopt Notebook または Adopt Procedure を選択します。

この項目は、アクティブなウィンドウが現在のエクスペリメントとは別に保存されているノートブックまたはプロシージャファイルであり、かつ現在のエクスペリメントがディスクに保存されている場合にのみ利用可能です。

(右はプロシージャファイルを取り込む場合のメニュー)

現在のエクスペリメントがパックされた形式で保存されている場合、ファイルをインポートすると、Igor Pro は一時ファイルとして save-as を行います。
その後、エクスペリメントを保存すると、一時ファイルの内容がパックされたエクスペリメントファイルに保存されます。
従って、エクスペリメントを保存するまで、インポートは完了しません。



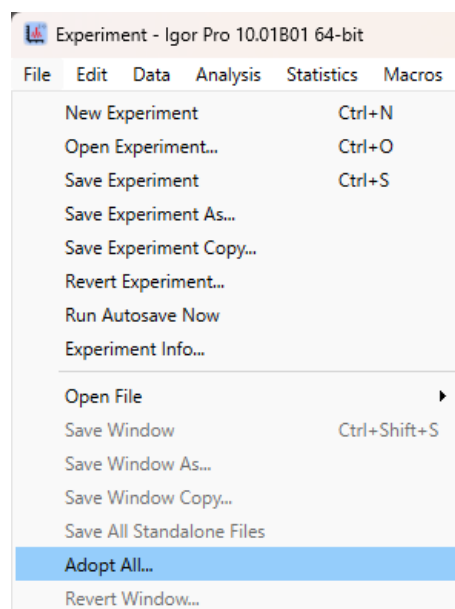
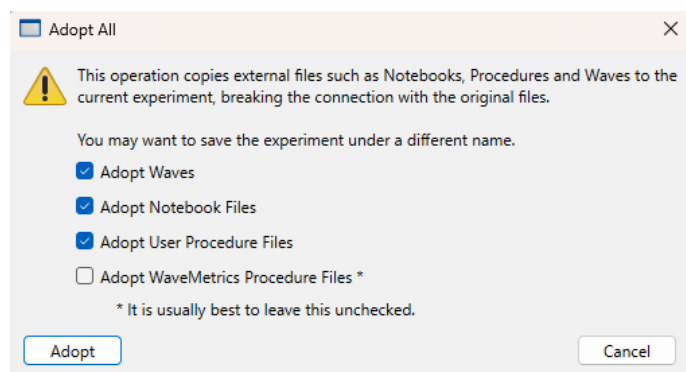
現在のエクスペリメントがパックされていない状態で保存されている場合、ファイルを取り込むと、そのファイルをエクスペリメントのホームフォルダーに save-as します。
その後、エクスペリメントを保存すると、エクスペリメントの再作成手順を更新し、元のファイルではなくホームフォルダー内の新しいファイルを開くようにします。
なお、パックされていないエクスペリメントでファイルを取り込んだ後、エクスペリメントを保存しなかった場合、新しいファイルはホームフォルダー内に残りますが、エクスペリメントの再作成手順は依然として元のファイルを参照し続けます。
したがって、ファイルを取り込んだ後は、エクスペリメントを保存する必要があります。

プロシージャまたはノートブックファイルを「Unadopt (解除)」するには、File メニューから Save Procedure File As または Save Notebook As を選択します。

すべてを取り込む

Shift キーを押しながら File→Adopt All を選択すると、参照されているすべてのノートブック、プロシージャファイル、ウェーブを取り込みます。

これは、他の人に送信するための、すべてが揃ったパッケージ化されたエクスペリメントを作成したい場合に便利です。



Adopt をクリックした後、File→Save Experiment As を選択して、パックされたエクスペリメントを保存します。

すべてのスタンドアロンファイルの保存

プログラマーは、特定のタスクに取り組んでいる時に、複数のプロシージャファイルを編集することがあります。File→Save All Standalone Procedure Files を選択すると、変更されたすべてのスタンドアロン・プロシージャファイルを一度に保存できます。

この操作では、スタンドアロン・プロシージャファイルのみが保存され、組み込みのプロシージャウィンドウや、まだファイルに保存されていない新しいプロシージャウィンドウを含むパックされたプロシージャファイルは保存されません。

Save All Standalone Procedure Files メニュー項目は、アクティブなウィンドウがプロシージャウィンドウの場合にのみ利用可能です。

同様に、File→Save All Standalone Notebook Files を選択することで、変更を加えたすべてのノートブックウィンドウを一括で保存できます。

この操作では、スタンドアロン・ノートブックファイルのみが保存され、パックされたノートブックファイルや、まだファイルとして保存されていない新しいノートブックウィンドウは保存されません。

Save All Notebook Procedure Files メニュー項目は、アクティブなウィンドウがノートブックウィンドウの場合にのみ利用可能です。

エクスペリメントの読み込まれ方

Igor Pro がエクスペリメントデータをどのように保存しているか、あるいはどのように再現しているかを知る必要は必ずしもありません。

しかし、これを理解しておくことで、いくつかの落とし穴を回避でき、Igor Pro に対する全体的な理解を深めることができるでしょう。

エクスペリメント再作成プロシージャ

エクスペリメントを保存すると、Igor Pro は「エクスペリメント再作成プロシージャ」と呼ばれるプロシージャとコマンドを作成します。

これらは、次回そのエクスペリメントを開いた時に Igor Pro によって実行されます。

通常、これらのプロシージャはユーザーには表示されません。

これらはエクスペリメントファイル内に保存されます。

バックされていないエクスペリメントファイルにはプレーンテキストが含まれていますが、拡張子が「.txt」ではないため、一般的なワープロソフトやエディタでは開くことができません。

例として、ごく単純なバックされていないエクスペリメントの実行手順を見てみます。

// ホームのシンボリックパスを作成

```
NewPath home ":Unpacked Experiment Folder:"
```

// "variables" ファイルからエクスペリメントの変数を読み込む

```
ReadVariables
```

// エクスペリメントのウェーブを読み込む

```
LoadWave/C /P=home "wave0.ibw"
```

```
LoadWave/C /P=home "wave1.ibw"
```

```
LoadWave/C /P=home "wave2.ibw"
```

```
DefaultFont "Helvetica"
```

```
MoveWindow/P 5,62,505,335 // プロシージャウィンドウを配置
```

```
MoveWindow/C 2,791,1278,1018 // コマンドウィンドウを配置
```

```
Graph0() // Graph0 ウィンドウを再作成
```

// Graph0 のグラフ再作成マクロ

```
Window Graph0() : Graph
```

```
    PauseUpdate; Silent 1 // ウィンドウの構築
```

```
    Display /W=(35,44,430,252) wave0,wave1,wave2
```

```
EndMacro
```

エクスペリメントを開くと、Igor Pro はエクスペリメントファイルからエクスペリメント再構築プロシージャを読み込み、プロシージャウィンドウに表示して実行します。

これらのプロシージャは、エクスペリメントを構成するすべてのオブジェクトとウィンドウを再構築します。

その後、エクスペリメント再構築プロシージャはプロシージャウィンドウから削除され、エクスペリメントのプロシージャファイルからユーザー自身のプロシージャが読み込まれ、プロシージャウィンドウに表示されます。

バックされたエクスペリメントの場合、手順は同じですが、エクスペリメントの再実行手順を含むすべてのデータがエクスペリメントファイルにバックされます。

エクスペリメントの初期化コマンド

エクスペリメントの再実行手順を実行し、その手順をプロシージャウィンドウに読み込んだ後、Igor Pro はプロシージャウィンドウの内容を確認します。

最初のマクロ、関数、またはメニューの宣言の前にあるコマンドは、すべて初期化コマンドとみなされます。

プロシージャウィンドウに初期化コマンドを記述すると、Igor Pro はそれらを実行します。

この仕組みを利用して、エクスペリメントが最初に読み込まれた時に初期化を行ってください。

Igor Pro に精通したプログラマーは、Igor Pro がエクスペリメントを開くたびに実行される関数を定義することもできます。

ヘルプ User-Defined Hook Functions を参照してください。

エクスペリメントの読み込み中のエラー

エクスペリメントの読み込み処理が正常に完了しない場合があります。

これは、ファイルやフォルダーを移動または名前を変更した時に最も頻繁に発生します。

また、エクスペリメントを別のコンピュータに移動した時、参照されているファイルやフォルダーも一緒に移動し忘れた場合にも発生します。

詳細については、「ファイルとフォルダーの参照」のセクションを参照してください。

ファイルが見つからない場合、Igor Pro はいくつかの選択肢を表示するダイアログを表示します。

エクスペリメントの読み込みを中止することを選択した場合、Igor Pro はエクスペリメントが不整合な状態にあることを警告します。問題の把握に役立つ診断情報が表示され、エクスペリメントの名前が「Untitled」に変更されます。

読み込みが途中で中断されたエクスペリメントを完全に削除するには、ファイルメニューから New Experiment または Open Experiment を選択してください。

ウェーブファイルの読み込みをスキップすると、後で Igor Pro がそのウェーブデータをグラフやテーブルに表示しようとした時に、別のエラーが発生する可能性があります。

その場合、右のようなダイアログが表示されます。

この例では、Igor Pro はグラフを再作成するために、エクスペリメント再作成手順に含まれる Graph0 マクロを実行しています。wave0 の読み込みをスキップするように設定したため、Igor Pro はそれを表示できません。

この時点で、以下の表に示すように、3つの選択肢があります。

オプション

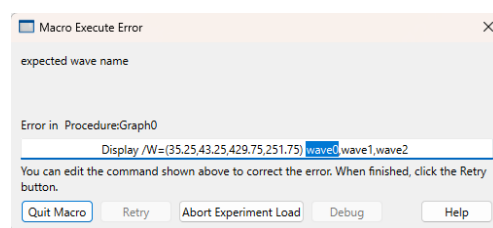
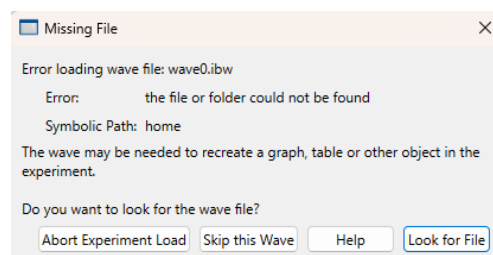
影響

Quit Macro	現在のマクロの実行を停止しますが、エクスペリメントデータの読み込みは継続されます。この例では、Graph0 は再作成されません。エクスペリメントデータの読み込みが完了すると、診断情報を表示します。
Abort Experiment Load	エクスペリメントの読み込みを直ちに中止し、診断情報を表示します。
Retry	この例では、「wave0,」を削除することでマクロを修正できます。その後、「Retry」ボタンをクリックします。Igor Pro は wave0 を含まない Graph0 を作成し、エクスペリメントデータの読み込みを続行します。

最初の2つのオプションでは、Igor Pro はエクスペリメントにタイトルを付けずにしておきます。そうすることで、保存操作を行う時に誤って元のエクスペリメントファイルを消去してしまうのを防ぎます。

Igor Pro がフォルダーの欠落に対処する方法

Igor Pro でエクスペリメントを保存すると、エクスペリメントファイルにコマンドが保存され、そのエクスペリメントを再度開いた時に、エクスペリメントのシンボリックパスが再現されます。



```
NewPath home ":Test Exp Folder:"  
NewPath/Z Data1 ">::Data Folder #1:"  
NewPath Data2 ">::Data Folder #2:"  
NewPath/Z Data3 "C:Test Runs:Data Folder #3:"
```

ホームフォルダーの場所は、エクスペリメントファイルが含まれるフォルダーを基準として指定されます。その他のすべてのフォルダーの場所は、エクスペリメントフォルダーを基準として指定するか、別のボリュームにある場合は絶対パスを使って指定します。

可能な限り相対パスを使うことで、エクスペリメントファイルとエクスペリメントフォルダーをまとめて別のディスクドライブや、同じディスクドライブ上の別の場所に移動しても、問題が発生しないようにすることができます。

/Z フラグは、このエクスペリメントにおいて Data1 と Data3 フォルダーからファイルをロードする必要がないことを示しています。

つまり、このエクスペリメントにはこれらのフォルダーへのシンボリックパスは存在しますが、エクスペリメントを再現するためにそれらからファイルを読み込む必要はありません。

エクスペリメントを再度開くと、Igor Pro はこれらの NewPath コマンドを実行します。フォルダーを移動または名前変更した場合、またはエクスペリメントファイルを移動した場合、NewPath コマンドではそのフォルダーが見つかりません。

エクスペリメントを再作成する時に、シンボリックパスに関連付けられたフォルダーが必要ない場合、NewPath コマンドには /Z フラグが含まれます。

この場合、Igor Pro はシンボリックパスの作成をスキップし、エラーを出力することなく、読み込みを続行します。

その結果、エクスペリメントには欠落したシンボリックパスが含まれないまま完了します。

欠落したフォルダーがウェブ、ノートブック、またはプロシージャファイルの読み込みに必要である場合、Igor Pro は Missing Folder ダイアログを表示し、そのフォルダーを検索するかどうかを確認します。

Look for Folder をクリックすると、Choose Folder ダイアログを表示し、そこで失われたフォルダーを指定することができます。

Skip This Path をクリックすると、シンボリックパスを作成しません。

その後、スキップされるシンボリックパスを使って読み込まれたウェブをスキップするかどうかを聞いてきます。

「はい」をクリックすると、存在しないフォルダーに関連付けられたシンボリックパスを使うすべての LoadWave コマンドをスキップします。

これにより、グラフ、テーブル、またはその他のウィンドウでウェブデータが必要になる場合、エクスペリメントの読み込みプロセスの後半でエラーが発生します。

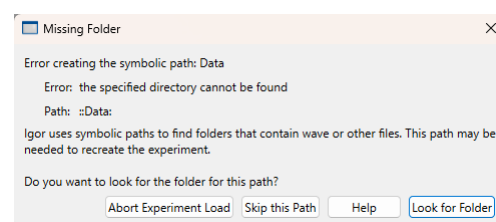
「いいえ」をクリックし、そのフォルダーにエクスペリメントで参照されているウェブファイルが含まれている場合、処理の途中で Missing File ダイアログが表示され、ウェブファイルの場所を指定する機会が与えられます。

例えば、エクスペリメントが Data2 パスを使って2つのウェブを読み込む場合、エクスペリメントの再作成コマンドには次のような2行が含まれます：

```
LoadWave/C/P=Data2 "wave0.ibw"  
LoadWave/C/P=Data2 "wave1.ibw"
```

Data2 パスの作成をスキップすることを選択し、そのパスからのウェブ読み込みをスキップするかどうかを尋ねられた時に「はい」をクリックした場合、Igor Pro はこれらの LoadWave コマンドをすべてスキップします。

Data2 パスの作成をスキップすることを選択し、そのパスからのウェブをスキップするかどうかを尋ねられた時に「いいえ」をクリックした場合、これらの LoadWave コマンドを実行するたびに Missing File ダイアログが表示されます。



そのウェーブファイルが見つからず、かつそのウェーブファイルがグラフやテーブルで使われている場合、実験の再作成プロセスが進むにつれて、欠落しているウェーブファイルを使ってグラフやテーブルを再作成しようとした時に、さらにエラーが発生します。

実験の保存のされ方

実験を初めて保存する場合、Igor Pro は単に新しいファイルを作成し、そこに書き込んで閉じるという、ごく単純な保存処理を行います。

しかし、既存の実験を再保存して実験ファイルの以前のバージョンを上書きする場合、Igor Pro は「セーフセーブ」という方法を使います。

この方法は、新しいバージョンの書き込み中にエラーが発生した場合でも、元のファイルを保持できるように設計されています。

説明のために、ここでは「Experiment.pxp」という名の実験ファイルを再保存すると仮定します。セーフセーブでは以下の手順を行います。

1. 元のファイルの名前を「Experiment.pxp.T0.noindex」に変更します。この手順でエラーが発生した場合、保存操作は中止され、エラーメッセージが表示されます。
2. そのファイルの新しいバージョンを「Experiment.pxp」という名前で保存します。
3. ステップ2が成功したら、元のファイル（「Experiment.pxp.T0.noindex」）を削除します。
ステップ2が失敗した場合は、新しいバージョンのファイルを削除し、元のバージョンのファイル名を元の名前に戻します。

以下の3つのセクションは、ファイルの保存に関する問題のトラブルシューティングにのみ使います。
問題が発生していない場合は、これらをスキップして構いません。

実験の保存エラー

実験の保存中にエラーが発生する原因は数多くあります。

例えば、ディスク容量が不足している場合や、保存先のサーバーボリュームが切断されている場合、あるいはハードウェアの故障などが考えられますが、これらはあまり一般的ではありません。

保存エラーが発生する最も一般的な原因は、以下の理由によりファイルへの書き込みアクセス権限が得られないことです：

1. ファイルが読み取り専用設定されている。
2. そのファイルが含まれているフォルダーへの書き込み権限がない。
3. この特定のファイルへの書き込み権限がない。
4. このファイルが別のアプリケーションによって開かれている。ウイルススキャナーやスパイウェア対策プログラムなどが原因である可能性がある。

以下に、トラブルシューティングの手法をいくつか紹介します。

Windows ファイルのトラブルシューティング

ファイルの「プロパティ」ウィンドウを開き、「読み取り専用」のチェックボックスがオンになっている場合は、そのチェックを外してください。

そのファイルが含まれているフォルダーについても、同様の操作を行ってください。

次に、書き込み権限があることが確認済みのフォルダー（例えば「ドキュメント」フォルダーなど）に Save As を試してみてください。

これでうまくいけば、元々のフォルダーやファイルへの書き込みに必要な権限が不足していたことが原因である可能性があります。

例えば、フォルダーが「Program Files」フォルダー内にあり、かつ管理者として実行していない場合などに、このような現象が発生することがあります。

元のファイルの保存先に書き込みができるはずだと考えている場合は、アクセス権限を確認する必要があります。この問題は複雑になる可能性があり、Windows のバージョンによって動作が異なります。

ファイルを新しい場所に保存することはできるが、元のファイルを上書きする目的で再度保存しようとするエラーが発生する場合は、別のプログラムが不適切なタイミングでそのファイルを開いている可能性があります。

これは通常、前述の「セーフセーブ」の手順のステップ 3 で発生します。

ウイルス対策ソフトを一時的に無効にしてみてください。

特別なフォルダー

このセクションでは、ヘルプファイル、Igor Pro 拡張機能（XOP と呼ばれるプラグイン）、プロシージャファイルを検索する時に、Igor Pro が自動的に検索する特別なフォルダーについて説明します。

Igor Pro Folder

「Igor Pro Folder」とは、Igor Pro アプリケーションが格納されている「IgorBinaries」フォルダーを含むフォルダーのことです。

デフォルトでは、このフォルダー名には Igor Pro のメジャーバージョン番号が含まれますが（例えば：「Igor Pro 10 Folder」）、一般的には「Igor Pro Folder」と呼ばれています。

Igor Pro は、「Igor Pro Folder」内にある以下の特別なサブフォルダーを確認します：

Igor Help Files
Igor Extensions
Igor Extensions (64-bit)
Igor Procedures
User Procedures
WaveMetrics Procedures

Igor Pro のインストーラーは、ファイルを特定のフォルダーに配置します。

ヘルプファイル、拡張機能、プロシージャファイルを検索する時に、これらのフォルダーを調べます。

ごく稀な例外を除き、「Igor Pro Folder」には一切変更を加えないでください。

Igor Pro User Files

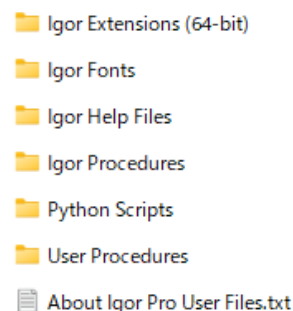
初回起動時に、Igor は「Igor Pro User Files」という名前の特別なフォルダーを作成します。デフォルトでは、このフォルダー名には Igor Pro のメジャーバージョン番号が含まれますが（例えば：「Igor Pro 9 User Files」）、一般的には「Igor Pro User Files」フォルダーと呼ばれます。

「Igor Pro User Files」は、デフォルトで以下の場所に作成されます：

C:\Users\<user>\Documents\WaveMetrics

Misc→Miscellaneous Settings から Igor Pro User Files フォルダーの場所を変更することは可能ですが、通常はその必要はほとんどありません。

「Igor Pro User Files」フォルダーは、右のような構成になっています。



Igor Pro のインストールに含まれるもの、自分で作成したもの、またはサードパーティから入手した追加のヘルプファイル、拡張機能、プロシージャファイルを有効にすることができます。

これを行うには、Igor Pro User Files フォルダー内の専用のサブフォルダーに、ファイルまたはファイルへのショートカットを追加してください。

追加の WaveMetrics ファイルの有効化に関する詳細については、「WaveMetrics プロシージャファイルの有効化」のセクションと「WaveMetrics XOP の有効化」のセクションを参照してください。

Help→Show Igor Pro User Files を選択すると、Igor Pro User Files フォルダーを表示できます。

WaveMetrics XOP やプロシージャファイルを有効にする時によく行うように、Igor Pro Folder と Igor Pro User Files フォルダーの両方を表示するには、Shift キーを押しながら Help→Show Igor Pro Folder and User Files を選択してください。

Igor Help Files フォルダー

Igor Pro を起動すると、「Igor Pro Folder/Igor Help Files」と「Igor Pro User Files/Igor Help Files」にある Igor Pro のヘルプファイルがすべて開かれます。

「Igor Help Files」内のショートカットやサブフォルダーについても、同様に扱われます。

WaveMetrics の標準ヘルプファイルは、「Igor Pro Folder/Igor Help Files」にプリインストールされています。

起動時に Igor Pro に自動的に開かせたいヘルプファイルが他にあれば、そのファイルまたはそのショートカットを「Igor Pro User Files/Igor Help Files」フォルダーに入れてください。

Igor Extensions フォルダー

Igor 拡張機能（XOP（「外部コマンド」）とも呼ばれる）は、Igor Pro に機能を追加するプラグインです。

WaveMetrics は、Igor Pro にいくつかの拡張機能を提供しています。

Igor Pro ユーザーやサードパーティも拡張機能を作成することができます。

詳細については、ヘルプ Igor Extensions を参照してください。

Igor Pro の拡張機能には、以前は 32 ビット版と 64 ビット版がありました。

32 ビット版の拡張機能は 32 ビット版の Igor Pro でのみ動作し、64 ビット版の拡張機能は 64 ビット版の Igor Pro でのみ動作します。

Igor Pro 10.0 以降、Igor Pro の 32 ビット版は提供されなくなり、すべての拡張機能は 64 ビット版である必要があります。

Igor Pro の起動時、プログラムは「Igor Pro Folder/Igor Extensions (64-bit)」と「Igor Pro User Files/Igor Extensions (64-bit)」フォルダー内を検索し、64 ビット版の Igor Pro 拡張ファイルを読み込みます。

読み込みが完了すると、これらの拡張機能を使用できるようになります。

「Igor Extensions (64-bit)」フォルダー内のショートカットやサブフォルダーについても、同様に扱います。

標準の WaveMetrics 拡張機能は、「Igor Pro Folder/Igor Extensions (64-bit)」にプリインストールされています。

その他の WaveMetrics 拡張機能については、「XOP Index」ヘルプファイルに記載されています。

このヘルプファイルは、Help→Help Windows メニューからアクセスでき、「Igor Pro Folder/More Extensions (64-bit)」内にあります。

なお、「Igor Pro Folder/More Extensions (64-bit)」にある拡張機能のヘルプファイルの一部は、「Igor Pro Folder/More Extensions」にも格納されています。

他にも使いたい拡張機能がある場合は、その拡張機能自体、またはそれを指すショートカットを「Igor Pro User Files/Igor Extensions (64-bit)」フォルダに入れてください。

Igor Procedures フォルダー

Igor Pro を起動すると、「Igor Pro Folder/Igor Procedures」と「Igor Pro User Files/Igor Procedures」にあるプロシージャファイルが自動的に開かれます。

「Igor Procedures」内のショートカットやサブフォルダーについても同様に扱われます。

このようなプロシージャファイルは「グローバル」プロシージャファイルと呼ばれ、すべてのエクスペリメントで使用可能です。

詳細については、ヘルプ Global Procedure Files を参照してください。

標準の WaveMetrics グローバルプロシージャファイルは、「Igor Pro Folder/Igor Procedures」にプリインストールされています。

その他の WaveMetrics プロシージャファイルについては、「WM Procedures Index」ヘルプファイルに記載されており、「Igor Pro Folder/WaveMetrics Procedures」内にあります。

また、独自のグローバルプロシージャファイルを作成したり、サードパーティから入手したりすることも可能です。

起動時に Igor Pro に自動的に開かせたい追加のプロシージャファイルがある場合は、そのファイル自体、またはそのファイルへのショートカットを「Igor Pro User Files/Igor Procedures」フォルダに配置してください。

User Procedures フォルダー

#include ステートメントを使うと、あるプロシージャファイルから別のプロシージャファイルを読み込むことができます。

この手法は、あるプロシージャファイルが別のプロシージャファイルを必要とする場合に使われます。

詳細については、ヘルプ Including a Procedure File を参照してください。

Igor Pro は #include 文に遭遇すると、「Igor Pro Folder/User Procedures」と「Igor Pro User Files/User Procedures」内の指定されたプロシージャファイルを検索します。

「User Procedures」内のショートカットやサブフォルダーも同様に扱われます。

プロシージャファイルの中から、追加で含めたいプロシージャファイルがある場合は、そのファイル自体、またはそのファイルへのショートカットを「Igor Pro User Files/User Procedures」フォルダに配置してください。

WaveMetrics Procedures フォルダ

「Igor Pro Folder/WaveMetrics Procedures」フォルダには、WaveMetrics によって作成された、役に立つ可能性のある各種プロシージャファイルが含まれています。

これらのファイルについては、Help→Help Windows メニューからアクセスできる「WM Procedures Index」ヘルプファイルに説明があります。

#include ステートメントを使うと、別のプロシージャファイルから WaveMetrics プロシージャファイルを読み込むことができます。

詳細については、ヘルプ Including a Procedure File を参照してください。

「Igor Pro User Files」フォルダ内に「WaveMetrics Procedures」フォルダはありません。

追加の WaveMetrics ファイルの有効化

以下のセクションでは、Igor Pro の追加機能を有効にする方法について説明します。

WaveMetrics のプロシージャファイルの有効化

すべてのエクスペリメントで利用できるように、WaveMetrics が提供するプロシージャファイルを有効にするには：

1. Shift キーを押しながら、Help→Show Igor Pro Folder and User Files を選択します。これにより、「Igor Pro Folder」と「Igor Pro User Files」フォルダが表示されます。
2. Igor Pro フォルダ内の「WaveMetrics Procedures」フォルダを開き、有効にしたい WaveMetrics プロシージャファイルを探してください。
3. そのファイルのショートカットを作成し、「Igor Pro User Files/Igor Procedures」フォルダに配置してください。これにより、次回 Igor Pro を起動した時に、そのプロシージャファイルが自動的に読み込まれます。
4. Igor Pro を再起動します。

有効化されたプロシージャファイルは、Windows→Procedure Windows サブメニューに表示されます。

WaveMetrics XOP の有効化

WaveMetrics 提供の XOP（「拡張機能」とも呼ばれるプラグイン）を有効にするには：

1. Shift キーを押しながら、Help→Show Igor Pro Folder and User Files を選択します。これにより、「Igor Pro Folder」と「Igor Pro User Files」フォルダが表示されます。
2. Igor Pro Folder 内の「More Extensions (64-bit)」フォルダを開き、有効にしたい XOP を特定してください（IGOR32 を使っている場合は、「More Extensions」フォルダを開いてください）。
3. XOP ファイルのショートカットを作成し、「Igor Pro User Files/Igor Extensions」フォルダに配置してください。これにより、次回 Igor Pro を起動した時に、XOP が自動的に読み込まれます。
4. Igor Pro を再起動します。

有効化された XOP は、そのコマンドや関数を Igor Help Browser に追加し、プログラムから使用できるようになります。

一部の XOP では、メニュー項目も追加されます。

その他のファイルの有効化

Igor パッケージを作成したり、サードパーティからパッケージを受け取ったりする場合があります。

各パッケージは、「Igor Pro User Files」フォルダー内の個別のフォルダーに、あるいは自身の判断で別の場所に保存してください。

バックアップや更新が複雑になるため、Igor Pro Folder 内にこれらのファイルを保存しないでください。

パッケージ内のファイルを有効にするには、パッケージファイルへのショートカットを作成し、「Igor Pro User Files」フォルダー内の適切なサブフォルダーに配置してください。

有効にしたいプロシージャファイルや Igor Pro 拡張ファイルが 1 つだけの場合、それらを「Igor Pro User Files」フォルダー内の適切なサブフォルダーに直接配置することを推奨します。

複数ユーザーのシナリオでのファイルの有効化

Igor Pro の「Igor Pro User Files」フォルダー内の専用サブフォルダーを使ってファイルを有効化することを推奨します。

「Igor Pro Folder」内では行わないでください。

ただし、サーバーから複数のユーザーが同じ Igor Pro コピーを実行しているマルチユーザー環境の場合は例外です。

この場合、すべてのユーザーに対してファイルを有効化したい場合は、そのファイルまたはそのショートカットを「Igor Pro Folder」内の適切なサブフォルダーに配置してください。

変更を有効にするには、ユーザーは Igor Pro を再起動する必要があります。

Igor Pro のファイル処理

Igor Pro には、ファイルを開いたり読み込んだりするのためのさまざまな方法があります。

以下のセクションでは、Igor Pro が要求に応じてさまざまなファイルをどのように扱うかについて解説します。

Open or Load File ダイアログ

File→Open File→Notebook などのメニュー項目を使ってファイルを開く場合、Igor Pro がそのファイルをどのように扱うべきかについて、迷うことはありません。

しかし、ファイルを Igor Pro のアイコンにドラッグ&ドロップしたり、デスクトップ上のファイルをダブルクリックしたりする場合は、必ずしもそうとは限りません。

多くの場合、Igor Pro はファイルの開き方や読み込み方を自動的に判断し、ユーザーに確認することなくその操作を実行します。

また、あるファイル（プレーンテキストファイルや書式設定済みの Igor Pro ノートブックなど）が複数の方法で適切に開けることを認識した場合、Open or Load File ダイアログを表示して、どのように処理するかをユーザーに尋ねることがあります。

このダイアログには、ファイルを開く（通常はウィンドウに表示する）方法や、データとして読み込む方法の一覧が表示されます。

注記： Recent Files または Recent Experiments メニューからファイルを開く時、またはファイルを Igor Pro のアイコンにドラッグ&ドロップする時に Shift キーを押すと、このダイアログを強制的に表示させることができます。

これは、Igor Pro のヘルプファイルを編集用のノートブックファイルとして開く場合や、ノートブックをヘルプファイルとして開いて Igor Pro にコンパイルさせる場合に特に便利です。

下のリストは、ファイル进行处理するための3つの方法を示しています。

1. ファイルをドキュメントウィンドウまたはエクスペリメントとして開く。
2. 別のダイアログを開かずに、ファイルをデータとして読み込む。
3. 別のダイアログを使って、ファイルをデータとして読み込む。

アスタリスクが付いたリスト項目を選択すると、Igor Pro は、あたかも Data メニューの Load Waves サブメニューから対応する項目を選択したかのように、選択されたダイアログを表示します。

リストの右側には、ファイルに関する情報や、そのファイルが最後に開かれた時の状況が表示されます。リストの下には、ファイルへの完全なパスが表示されます。

Rename File 編集ボックスのテキストを変更すると、ファイルを開くか読み込む前にファイル名を変更します。

Recent Files と Recent Experiments

ダイアログを使ってエクスペリメントやファイルを開いたり保存したりすると、Igor Pro はその項目を File メニュー内の Recent Experiments または Recent Files サブメニューに追加します。

これらのサブメニューから項目を選択すると、Igor Pro は前回開いたときや保存したときと同じ方法で、そのエクスペリメントやファイルを開きます。

例えば、前回テキストファイルをフォーマットなしのノートブックとして開いた場合、Recent Files からそのファイルを選択すると、再びフォーマットなしのノートブックとして開かれます。一般テキストデータファイルとして読み込んだ場合は再びデータとして読み込みます。

ただし、Igor Pro はデータファイルを最初に読み込んだ時の詳細をすべて記憶しているわけではありません。フォーマットに関して細かい調整を施したテキストデータファイルを読み込んだ場合、それらの調整をそのまま適用して読み込むことはありません。

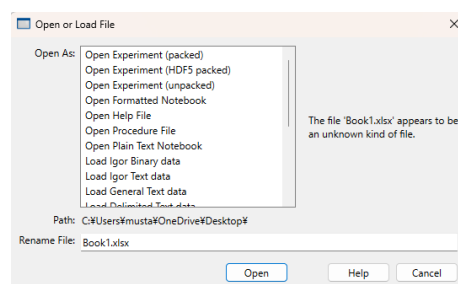
Igor Pro が確実にデータを正しく読み込むようにするには、適切な Load Data ダイアログを使ってください。

Shift キーを押しながらエクスペリメントまたはファイルを選択すると、Open or Load File ダイアログが表示され、そのファイルをどのように開くか、または読み込むかを選択できます。

デスクトップでのドラッグ&ドロップ

ファイルを Igor Pro のウィンドウにドラッグ&ドロップできます。この機能の用途の1つは、複数のデータファイルを一度に読み込むことです。

ドロップしたファイルが最近開かれたり読み込まれたりした場合は、Recent Files または Recent Experiments メニューに表示されます。その場合、Igor Pro は同じ方法でそのファイルを再度開いたり読み込んだりします。



ファイルを Igor Pro にドラッグ&ドロップする時に Shift キーを押すと、Open or Load File ダイアログを表示し、そこでファイルの処理方法を選択できます。

上級プログラマーは、Igor Pro をカスタマイズして、特定の種類のファイルをさまざまな方法で処理できるようにすることができます。

例えば、XOP を含むファイルを自動的に読み込むように設定することも可能です。

ヘルプ User-Defined Hook Functions を参照してください。

IGOR64 エクスperimentファイル

Igor Pro の 64 ビット版では、32 ビットの制限を超えるサイズのウェーブデータを、パックされたエクスperimentファイルには保存できますが、パックされていないエクスperimentファイルには保存できません。

巨大なウェーブデータの保存には、64 ビット対応のウェーブレコードが使われ、数値ウェーブのみがサポートされています。

バージョン 6.2 以降の Igor Pro の 32 ビット版は、64 ビット対応のウェーブレコードをサポートしています。デフォルトでは、IGOR32 はこれらのレコードを読み込もうとし、読み込めない場合はメモリ不足の警告を表示します。

このセクションの残りの部分では、上級ユーザー向けの機能について説明します。

SetIgorOption を使うと、Igor Pro が 64 ビット対応のウェーブレコードを処理する方法の一部をコントロールできます。

SetIgorOption の設定は、Igor Pro を終了するまでしか有効ではないことに注意してください。

```
SetIgorOption UseNewDiskHeaderBytes = bytes
```

bytes が 0 以外の場合、ウェーブデータのサイズがこの値を超えると、64 ビット対応のウェーブレコード型が使われます。

IGOR32 では、bytes のデフォルト値は 0（オフ）です。

IGOR64 では、bytes のデフォルト値は 100E6 です。

64 ビット対応のウェーブレコード型をどうしても必要な場合のみ使いたい場合は、2E9 を指定してください。

```
SetIgorOption UseNewDiskHeaderCompress = bytes
```

bytes が 0 以外の場合、ウェーブデータのサイズがこの値を超え、かつ 64 ビット対応のウェーブレコード型が使われているときは、データは圧縮されて書き込まれます。

デフォルト値は 0（オフ）です。

これは、一部のデータ種別ではファイルサイズを大幅に削減できますが、エクスperimentを保存する時の処理速度が大幅に低下するためです。

多くの場合、圧縮によって実際には元のデータよりもサイズが大きくなってしまいます。

このような場合、Igor Pro は圧縮データではなく元のデータを書き込みます。

```
SetIgorOption MaxBytesReadingWave = bytes
```

```
SetIgorOption BigWaveLoadMode = mode
```

これら2つのオプションは、64 ビット対応のウェーブレコードタイプを含むエクスperimentデータを読み込む時の IGOR32 の動作をコントロールします。

モードは 0（デフォルト）に設定すると、特別な処理は行われません。

Igor Pro はすべてのレコードを読み込もうとします。

mode が 1 の場合、データサイズが bytes を超えるウェーブは、警告なしにスキップされます。

mode が 2 の場合、データサイズが bytes を超えるウェーブはすべてより小さいサイズに縮小され、そのデータの一部分のみが読み込まれます。

このような部分的なウェーブは「ロック済み」としてマークされ、ロックフラグのビット 1 も設定されます。さらに、ウェーブの注記には「PARTIAL LOAD」というテキストが含まれ、エクスペリメントの読み込み後に警告ダイアログが表示されます。

IGOR32 では、bytes のデフォルト値は 500E6 です。

Autosave (自動セーブ)

Igor Pro は、作業中にファイルを自動保存することができます。

これは、保存する前にクラッシュしたり、その他の予期しない事態が発生したりした場合に役立ちます。

もし、保存しておきたいような重要な変更を加えるたびに保存する習慣があるなら、自動保存機能は必要ありません。

もし、保存せずに大量の作業を行う習慣があるなら（これは本質的にリスクを伴います）、自動保存機能があれば、クラッシュが発生した時に作業内容が失われるのを防ぐことができます。

自動保存は絶対的なものではなく、複数のユーザーが同じファイルを編集した場合や、Igor Pro の複数のインスタンスで同じファイルを編集した場合、混乱を招くような予期しない動作を引き起こす可能性があります。こうした理由やその他の理由から、自動保存はあくまで最終手段と捉え、作業が進んだら必ず手動で保存し、失うのが痛手となるようなデータは常にバックアップを取るようにしてください。

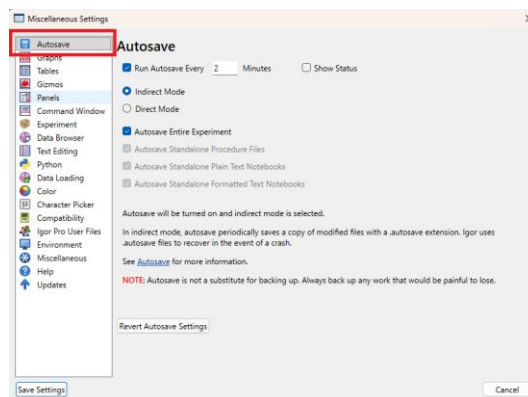
自動保存はデフォルトでオフになっています。

これをオンにしたり、動作を詳細に設定したりするには、Misc→Miscellaneous Settings を選択し、左側のリストにある Autosave アイコンをクリックして、Miscellaneous Settings ダイアログの Autosave パネルを表示してください。

Run Autosave チェックボックスをオンにすると、自動保存が有効になります。

有効にすると、指定した間隔（分単位）で自動保存が実行されます。

自動保存は、Igor Pro がアイドル状態のときに実行され、プロシージャの実行中やモーダルダイアログが表示されている間は実行されません。



自動保存の代わりに利用できる方法には、次のようなものがあります：

- ・ エクスペリメントが良好な状態であると判断した場合は、File→Save Experiment を選択して手動で保存する。
- ・ エクスペリメントが良好な状態であると判断した場合は、File→Run Autosave Now を選択して、手動で自動保存を実行する。
- ・ File→Save All Standalone Procedure Files のみを使って、変更されたプロシージャファイルを手動で保存する。

Igor Pro は、ステータスバーに自動保存の状態を示す情報を表示することができます。

自動保存情報の表示・非表示は、Miscellaneous Settings ダイアログの Autosave ペインにある Show Status チェックボックスを使うか、ステータスバーを右クリックすることで切り替えることができます。

自動保存が有効になっている場合、ステータスバーには次回の自動保存の実行時刻が表示されます。

「Run」とは、「自動保存すべきファイルがあるかどうかを確認し、ある場合は自動保存を行う」ことを意味します。

自動保存が有効な状態で、Igor Pro の異なるインスタンスで同じファイルを編集すると、混乱を招き、予期せぬ動作を引き起こす可能性があります。

Igor Pro のプログラマーは、IgorInfo(13) を使って自動保存の設定を取得できます。

Autosave の対象

選択した設定に応じて、自動保存では以下の種類のファイルを保存できます：

- ・ エクスペリメントファイル全体
- ・ スタンドアロンのプロシージャファイル
- ・ スタンドアロンのノートブックファイル

以下のセクションでは、自動保存の対象として選択したファイルの種類を「自動保存可能なファイル」と呼びます。

以下の場合には自動保存が機能しません：

- ・ まだディスクに保存していないファイル
これには、作成済みだがまだディスクに保存されていないエクスペリメントファイル、プロシージャファイル、ノートブックファイルが含まれます。
- ・ 間接モードのバックされていないエクスペリメントファイル (.uxp)
- ・ 独立したモジュールに `#include` されたプロシージャファイルのコピー
これらのファイルは変更しないでください。詳細については、ヘルプ Independent Modules and `#include` を参照してください。

自動保存機能は、バックされたエクスペリメント (.pxp および .h5xp) に対してのみ、かつ Autosave Entire Experiment チェックボックスがオンになっている場合に限り、以下の項目を保存します：

- ・ ビルトインプロシージャウィンドウ
- ・ バックされたプロシージャファイル
- ・ バックされたノートブックファイル
- ・ グラフ、テーブル、ページレイアウト、Gizmo プロット、コントロールパネル、その他のウィンドウ
- ・ データフォルダーとウェーブ

自動保存は、手動での保存やバックアップの代わりではない

自動保存の目的は、Igor Pro やコンピューターがクラッシュしたり、停電が発生したりした場合でも、作業内容を復元できる可能性を高めることにあります。

頻繁な保存やバックアップの代わりとして自動保存だけに頼るのは、災いを招く行為です。

選択した自動保存の設定によっては、自動保存が不適切なタイミング（例えば、ミスをした直後など）で行われたり、作業内容のすべてが保存されない場合があります（「Autosave の対象」のセクションを参照）。

重要な変更を加えた後、自動保存が実行される前に、システムクラッシュやその他の障害が発生する可能性があります。

今日保存したファイルに誤りがあることに気づき、昨日のバージョンに戻す必要もあるかもしれません。

- ・ コンピューターが致命的な障害を起こす可能性があります。
- ・ コンピューターがマルウェアに感染している可能性があります。

- ・ コンピューターが盗まれる可能性があります。
- ・ 火災や自然災害、あるいはその他の不測の事態により、コンピューターが損傷を受ける可能性があります。

こうした理由やその他の理由から、作業が進んだら必ず保存し、失うのが痛手となるようなデータはバックアップしておくべきです。

コンピューターが破損したり盗難に遭ったりした場合に備え、ローカルとリモートの両方にバックアップをとっておくべきです。

Autosave のモード

Miscellaneous Settings ダイアログの Autosave パネルで、自動保存の動作モードを設定できます。

自動保存には3つのモードがあります。

それぞれに長所と短所があります。

Off

Run Autosave チェックボックスがオフになっている場合、自動保存は無効になります。

このモードでは、Igor Pro の自動保存ルーチンは自動的に実行されません。

長所

- 理解しやすい。
- 時間がかからない。
- ファイルの保存タイミングは、完全にコントロールできる。

短所

- 重要な変更を加えた時、保存するのはユーザーの責任。

Indirect Mode

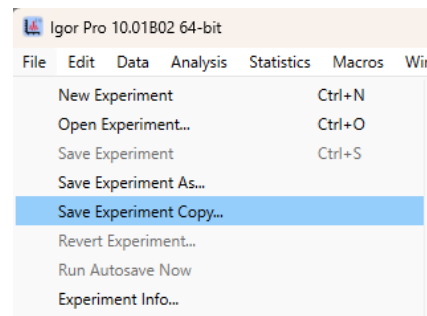
現在のエクスペリメント（バックされている場合。バックされていない場合は除く）またはスタンドアロンプロシージャやスタンドアロンノートブックファイルを、元のファイルと同じ場所に .autosave ファイルとして保存します。

例えば、「Experiment.pxp」は「Experiment.pxp.autosave」として自動保存され、「Proc0.ipf」は「Proc0.ipf.autosave」として自動保存されます。

現在のエクスペリメントを保存するのは、File→Save Experiment Copy を選択するのと同じです。

スタンドアロンのプロシージャファイルを保存する操作は、File→Save Procedure Copy を選択するのと同じです。

スタンドアロンのノートブックを保存するには、File→Save Notebook Copy を選択します。



現在のエクスペリメント、スタンドアロンのプロシージャファイル、スタンドアロンのノートブックを自動保存するかどうかを選択できます。

バックされていないエクスペリメントデータの Indirect Autosave はサポートされていません。

長所

- 例えば誤操作直後など、不適切なタイミングで元のファイルが上書きされてしまうリスクがない。

短所

- エクスペリメントの規模が非常に大きい場合、エクスペリメント全体を保存するには時間がかかることがある。
- エクスペリメント全体を保存しない場合、組み込みプロシージャウィンドウ、パックされたプロシージャファイルやノートブック、グラフ、レイアウト、その他のウィンドウ、データフォルダー、ウェーブデータなど、多くの項目は自動保存されない。
詳細については、「Autosave の対象」のセクションを参照。
- 複数のユーザー間でのファイル共有、または1人のユーザーが起動した Igor Pro の複数のインスタンス間でのファイル共有に支障をきたす可能性がある。
- パックされていないエクスペリメントでは動作しない。
- .autosave ファイルは、ダイレクトモードに比べてファイルが散らかりやすくなる。
- クラッシュ後の復旧は、ダイレクトモードに比べて複雑になる。Igor Pro は、.autosave ファイルを使うか、元のファイルを使うかを聞いてくる。

Direct Mode

現在のエクスペリメントまたはスタンドアロンのプロシージャ、スタンドアロンのノートブックファイルを直接保存します。

現在のエクスペリメントを保存するには、File→Save Experiment を選択します。

スタンドアロンのプロシージャファイルを保存する方法は、File→Save Procedure を選択するのと同じです。

スタンドアロンのノートブックを保存するには、File→Save Notebook を選択します。

現在のエクスペリメント、スタンドアロンのプロシージャファイル、スタンドアロンのノートブックを自動保存するかどうかを選択できます。

長所

- 単純さ。

短所

- エクスペリメントの規模が非常に大きい場合、エクスペリメント全体を保存するには時間がかかることがある。
- 自動保存は、例えばミスをした直後など、不適切なタイミングでファイルを保存してしまうことがある。
- エクスペリメント全体を保存しない場合、組み込みプロシージャウィンドウ、パックされたプロシージャファイルやノートブック、グラフ、レイアウト、その他のウィンドウ、データフォルダー、ウェーブデータなど、多くの項目は自動保存されない。詳細については、「Autosave の対象」のセクションを参照。
- 複数のユーザー間でのファイル共有、または1人のユーザーが起動した Igor Pro の複数のインスタンス間でのファイル共有に支障をきたす可能性がある。

エクスペリメント全体の自動保存を設定する変更点

Autosave Entire Experiment にチェックを入れ、内容に大幅な変更を加えた場合、Igor Pro は次回の自動保存実行時にエクスペリメント全体を自動保存します。

実質的な変更には、ウェーブやデータフォルダーの作成、削除、変更、グラフ、テーブル、レイアウト、パネル、または Gizmo プロットの内容の変更、およびプロシージャファイルやノートブックの編集などが含まれます。

エクスペリメント全体を自動保存するには時間がかかる場合があります。

エクスペリメントが頻繁に自動保存されるのを防ぐため、比較的軽微な変更については、エクスペリメントの自動保存が行われません。

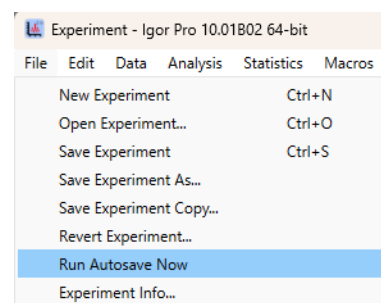
これには、ウィンドウの移動やサイズ変更、他のオブジェクトに実質的な変化をもたらさない履歴領域の変更、グローバル変数の変更などが含まれます。

File→Run Autosave Now メニュー項目が無効になっている場合、前回のエクスペリメント保存または前回の自動保存以降、実質的な変更が行われていないことを示しています。

自動保存の実行を強制する

File→Run Autosave Now を選択すると、オフになっているときや、通常の実行時間より前に、自動保存ルーチンを強制的に実行させることができます。

File→Run Autosave Now メニュー項目が無効になっている場合、現在、自動保存可能なファイルが存在しないことを意味します。これは通常、前回の自動保存処理以降、自動保存可能なファイルが変更されていないために発生します。



Run Autosave Now メニューコマンドは、Miscellaneous Settings ダイアログの Autosave ペインにある Run Autosave チェックボックスの状態にかかわらず、自動保存プロシージャを実行します。ただし、そのペイン内のその他の設定は反映されます。

自動保存なしでスタンドアロンファイルを保存する

自動保存がオンかオフに関わらず、変更を加えたすべてのプロシージャファイル、またはすべてのスタンドアロンのノートブックファイルを一度に保存できます。

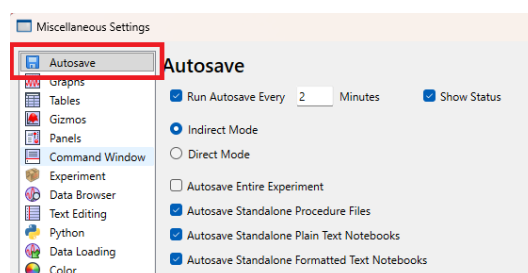
詳細については、「すべてのスタンドアロンファイルの保存」のセクションを参照してください。

Indirect Autosave モード

間接自動保存モードは、直接モードよりも複雑です。

ここでは、Miscellaneous Settings ダイアログの Autosave ペインで右の設定が行われていることを前提に、間接自動保存の仕組みについて説明します。

ここで、以前に「Proc0.ipf」というファイル名で保存したスタンドアロン・プロシージャウィンドウを編集していると仮定します。



Igor Pro は 2 分ごとに、前回の自動保存以降に変更が加えられた自動保存の対象ファイルがあるかどうかを確認します。

この確認は、Igor Pro がアイドル状態のときに実行され、プロシージャの実行中やモーダルダイアログが表示されている間は実行されません。

前回の自動保存以降に「Proc0.ipf」に変更を加えた場合、その現在の内容を「Proc0.ipf.autosave」という名前のファイルに書き込むことで、ファイルを自動保存します。

「Proc0.ipf」を保存したとき、保存せずに閉じたとき、または元に戻したとき、対応する「Proc0.ipf.autosave」ファイルが存在する場合、それを削除します。

.autosave ファイルからファイルを復旧する

Igor Pro がクラッシュしたり、その他の重大な障害が発生し、Igor Pro が終了する前に Indirect Autosave ルーチンが実行された場合、元のファイル「Proc0.ipf」と、.autosave ファイル「Proc0.ipf.autosave」が残されます。手動またはプログラムで元のファイルを開こうとすると、Igor Pro は確認のダイアログを表示します。

注意： 作業を進める前に、元のファイルと自動保存されたファイルのバックアップをとっておくことをお勧めします。

Cancel をクリックした場合

Open 操作はキャンセルされ、元のファイルと自動保存されたファイルは変更されません。

Open Original File をクリックした場合

Igor Pro は元のファイル（例：「Proc0.ipf」）を開きます。

その後、Igor Pro は自動保存ファイル（例：「Proc0.ipf.autosave」）をゴミ箱に移動します。

ただし、その自動保存ファイルが Igor Pro で開かれている場合は、ゴミ箱には移動されず、その後の自動保存によって上書きされる可能性があります。

Open AutoSave File をクリックした場合

Igor Pro は、元のファイル名に「.original」という拡張子を付加してファイル名を変更します（例：「Proc0.ipf」は「Proc0.ipf.original」に変更されます）。

その後、Igor は自動保存ファイルの名前を元のファイル名に変更します（例：「Proc0.ipf.autosave」が「Proc0.ipf」に変更されます）。

その後、自動保存ファイルを開きます。

このファイルには、元のファイル名（例：「Proc0.ipf」）が付けられます。

その後、拡張子が「.original」に変更された元のファイル（例：「Proc0.ipf.original」）をゴミ箱に移動します。

Indirect Autosave モードでの問題

間接モードでは、複数のユーザー間でファイルを共有する場合や、1人のユーザーが起動した複数の Igor Pro インスタンス間でファイルを共有する場合、混乱を招く状況が生じることがあります。

例えば、間接自動保存モードがオンになっている状態でユーザーが共有プロシージャファイルを編集している場合、対応する .autosave ファイルを作成します。

別のユーザーが直接、または間接的に（たとえば #include ステートメントを介して）元のファイルを開くと、.autosave ファイルが存在するため、前のセクションで説明した Open Autosave File ダイアログを表示します。

これにより、2人目のユーザーは混乱し、1人目のユーザーの編集作業に支障をきたす可能性があります。

同様の状況は、共有ノートブックやエクスペリメントファイルでも発生する可能性があります。

このような問題は、ソースコード管理システムを利用したり、共有ファイルに直接編集を加えないようにしたり、自動保存機能をオフにしたりすることで軽減できます。

.autosave ファイルを明示的に開く

通常、.autosave ファイルを開く必要はありませんが、内容を確認したい場合は開くこともあるでしょう。

Igor Pro は、プロシージャとノートブックの .autosave ファイルを読み取り専用で開きます。

.autosave ファイルも開いている状態で、そのプロシージャまたはノートブックファイルを閉じると、.autosave ファイルを自動的に閉じてから削除します。

Igor Pro では、.autosave という拡張子を持つエクスペリメントファイルを開くことはできません。

そのようなファイルを開きたい場合は、Igor Pro で開く前に、拡張子「.autosave」を削除してファイル名を変更してください。

Indirect Autosave とバックされていないエクスペリメント

バックされていないエクスペリメントデータの Indirect Autosave はサポートされていません。

バックされていないエクスペリメントが開かれている場合でも、Miscellaneous Settings ダイアログの Autosave ペインにある対応するチェックボックスが有効になっている限り、スタンドアロンプロシージャファイルとノートブックの自動保存は実行されます。

バックされていないエクスペリメントに関する背景情報については、「バックされていないエクスペリメントファイルとして保存」のセクションを参照してください。

Indirect Autosave の復元機能を試す

Indirect Autosave の仕組みを理解したい場合は、この機能を使って試してみてください。

この機能は、Igor Pro User Files フォルダーに「Test Indirect Autosave.txt」と「Test Indirect Autosave.txt.autosave」というファイルを書き出し、その後「Test Indirect Autosave.txt」をノートブックとして開きます。

Igor Pro は「Open Autosave File?」というダイアログを表示し、そこで開きたいファイルを選択できます。選択されなかったファイルはごみ箱に移動されます。

```
Function TestOpenWithIndirectAutoSaveFilePresent()  
    KillWindow/Z TestAutosaveNotebook  
  
    String pathName = "IgorUserFiles"  
  
    // オリジナルファイルを作成  
    String fileName = "Test Indirect Autosave.txt"  
    Variable refNum  
    Open/P=$pathName refNum as filename  
    fprintf refNum, "This is a test\r\n"  
    Close refNum  
  
    // .autosave ファイルを作成  
    CopyFile /O /P=$pathName fileName as fileName + ".autosave"  
  
    // "Open Autosave File?" ダイアログを表示  
    OpenNotebook/P=$pathName/N=TestAutosaveNotebook filename  
    int err = GetRTError(1) // ユーザーがキャンセルしたときはゼロ以外を返す
```

```
Print err  
End
```