

CONTENTS

ビジュアルヘルプ – Igor Pro における HDF5	4
Igor Pro における HDF5.....	4
HDF5 ガイドツアー.....	4
HDF5 の概要.....	4
Igor Pro の HDF5 サポート.....	5
HDF5 Browser の使い方.....	5
プログラムによる HDF5 データの読み込み.....	9
プログラムによる HDF5 データの保存.....	13
今後の進め方.....	18
HDF5 についてさらに詳しく.....	18
HDF5 Browser.....	18
HDF5 Browser の使用方法.....	19
HDF5 Browser の基本的なコントロール.....	19
HDF5 Browser のコンテキストメニュー.....	21
HDF5 Browser の高度なコントロール.....	22
HDF5 Browser ダンプの技術的な詳細.....	23
HDF5 コマンドと関数.....	23
HDF5 プロシージャファイル.....	24
HDF5 の属性.....	24
HDF5 の数値属性の読み込み.....	25
HDF5 文字列属性の読み込み.....	26
HDF5 グループまたはデータセットのすべての属性の読み込み.....	27
HDF5 データセットのサブセット.....	28
Igor Pro と HDF5 の比較.....	30
大文字と小文字の区別.....	30
HDF5 のオブジェクト名と Igor Pro のオブジェクト名の比較.....	31
HDF5 のデータ型と Igor Pro のデータ型の比較.....	31
HDF5 の最大次元と Igor Pro の最大次元の比較.....	31
行優先と列優先のデータ順序.....	32
HDF5 画像と Igor Pro 画像の比較.....	32
Igor Pro データの保存と再読み込み.....	33

複素数ウェーブの取り扱い	33
Igor Pro 参照ウェーブの処理	34
HDF5 マルチタスキング	34
Igor Pro の HDF5 処理能力	34
Igor Pro の HDF5 の制限	35
HDF5 の高度なデータ型	35
HDF5 の可変長データ	35
HDF5 配列データ	36
HDF5 参照データの読み込み	37
HDF5 オブジェクト参照データの読み込み	37
HDF5 データセット領域参照データの読み込み	38
HDF5 オブジェクト参照データの保存	39
HDF5 データセットの領域参照データの保存	40
HDF5 Enum データの読み込み	40
HDF5 Enum データの保存	40
HDF5 の不透明データ	41
HDF5 ビットフィールドデータ	41
HDF5 のネストされたデータ型	42
HDF5 複合データ	42
HDF5 圧縮	43
HDF5 レイアウトチャンクサイズ	44
HDF5SaveGroup での圧縮	45
SaveExperiment での圧縮	45
HDF5 のデフォルトの圧縮	45
HDF5SaveDataHook の使用方法	46
HDF5 圧縮の参考情報	48
HDF5 の動的読み込みフィルター	49
HDF5 次元スケール	50
Igor Pro における HDF5 の次元スケールのサポート	50
HDF5 次元スケールの実装の詳細	51
HDF5 次元スケールの参考情報	51
その他の HDF5 の情報	52
HDF5 文字列フォーマット	52
HDF5 文字列変数のテキストエンコーディング	54

HDF5 ウェーブのテキストエンコーディング	54
Igor Pro の HDF5 との互換性	55
HDF5 の互換性に関する用語	55
古い HDF5 プログラムで Igor Pro の HDF5 ファイルを読み込む	55
古い HDF5 ファイルへの書き込み	56
HDF5 パックされたエクスペリメントファイル	56
HDF5 パックされたエクスペリメントファイルの構成	57
HDF5 パックされたエクスペリメントファイルの階層	58
HDF5 パックされたエクスペリメントファイル内のプレーンテキストの保存	58
HDF5 パックされたエクスペリメントの書き込み	59
HDF5 ファイル内のウェーブ	59
ウェーブ参照ウェーブとデータフォルダー参照ウェーブ	60
フリーウェーブとフリーデータフォルダー	60
HDF5 パックされたエクスペリメントの問題	61

ビジュアルヘルプ – Igor Pro における HDF5

Igor Pro における HDF5

HDF5 は、広く利用されている非常に柔軟性の高いデータファイル形式です。

HDF5 ファイル形式は、National Center for Supercomputing Applications (NCSA) によって開発されました。現在、HDF5 は HDF グループ (<<http://www.hdfgroup.org>>) によってサポートされています。

Igor Pro は HDF5 を幅広くサポートしており、その内容は以下の通りです。

- ・ HDF5 ファイルをブラウズする
- ・ HDF5 ファイルからデータを読み込む
- ・ HDF5 ファイルヘデータを保存する
- ・ 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
- ・ HDF5 ファイルから Igor Pro のエクスペリメントデータを読み込む

Igor Pro 9.0 以前のバージョンでは、HDF5 のサポートは、ユーザー自身が有効化する必要のある XOP によって提供されていました。

現在、HDF5 のサポートは Igor Pro に組み込まれており、有効化は不要になりました。

Igor Pro 9.0 では、Igor Pro のエクスペリメントデータを HDF5 ファイルとして保存、読み込む機能が追加されました。

詳細は、「HDF5 パック形式のエクスペリメントファイル」のセクションを参照してください。

HDF5 ガイドツアー

このセクションで、Igor Pro で HDF5 を活用するための第一歩を踏み出すことができます。

この資料には、実行すべき手順が番号付きで記載されています。

ガイドツアーからずれないように、記載通りに正確に実行してください。

このツアーは、Igor Pro の熟練ユーザーが Igor Pro から HDF5 ファイルにアクセスする方法を学ぶ手助けとなるだけでなく、Igor Pro をまだ使っていないユーザーへの説明も目的としています。

そのため、このツアーは Igor Pro に関する知識が一切ないことを前提に作成されています。

HDF5 の概要

HDF5 は、想像しうるほぼあらゆるデータセットを格納し、データセット間の関係をカプセル化できるように設計された、非常に強力ですが複雑なファイル形式です。

HDF5 ファイルには、ハードディスク上のディレクトリやファイルの階層構造と同様の階層構造を含めることができます。

HDF5 では、この階層構造は「グループ」と「データセット」で構成されています。

“/” という名前のルートグループがあります。

各グループには、データセットや他のグループを含めることができます。

HDF5 データセットとは、1次元または多次元の要素の集合のことです。

各要素は、「原子型」データ型（例：16ビットの符号付き整数や32ビットのIEEE浮動小数点数）である場合も

あれば、構造体や配列などの「複合型」データ型である場合もあります。
「複合」データ型とは、C 言語の構造体に類似した複合データ型のことです。
そのメンバーは、原子データ型または複合データ型であることができます。
ここでは、複合データ型については一旦忘れて、原子データ型のみを取り扱います。

各データセットには、任意の数の「属性」を関連付けることができます。
属性はデータセットに似ていますが、階層の一部となるのではなく、データセットやグループに紐付けられるものです。

Igor Pro の HDF5 サポート

Igor Pro HDF5 パッケージは、HDF5 の組み込みサポートと一連の Igor Pro プロシージャで構成されています。

Igor Pro の起動時に自動的に読み込まれるプロシージャは、Igor Pro 内に HDF5 Browser を実装するものです。
このブラウザーでは、以下の機能がサポートされています。

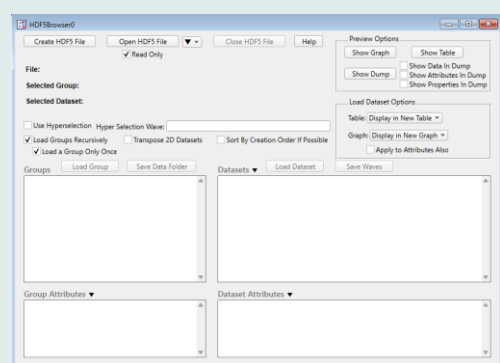
- ・ HDF5 ファイルのデータをプレビューする
- ・ HDF5 データセットとグループを読み込む
- ・ Igor Pro のウェーブとデータフォルダーを HDF5 ファイルとして保存する

Igor Pro 9 以降では、エクスペリメントデータを HDF5 形式のパックされたエクスペリメントファイルとして保存したり、そのファイルからエクスペリメントデータを読み込んだりすることができます。

HDF5 Browser の使い方

1. メニュー Data→Load Waves→New HDF5 Browser を選択します。

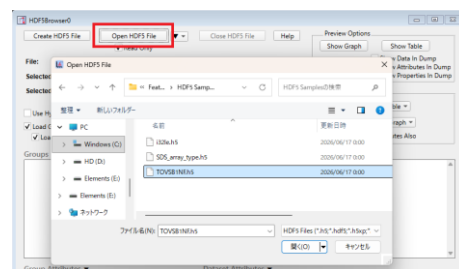
HDF5 Browser のコントロールパネルが表示されます。



2. Open HDF5 File ボタンをクリックし、次のファイルを開きます。

Igor Pro Folder\Examples\Feature Demos\HDF5
Samples\TOVSB1NF.h5

(Igor Pro Folder の場所がわからない場合は、メニュー Misc→Path Status を選択し、Igor のシンボリックパスをクリックして、Igor Pro Folder へのパスを確認してください。)



このサンプルは NCSA のウェブサイトから入手しました。

これで、右のような画面が表示されているはずですが。

このブラウザーには 4 つのリストが含まれています。

左上のリストは Groups リストであり、HDF5 ファイル内のグループが表示されています。

HDF5 ファイル内のグループは、ハードディスクの階層構造におけるディレクトリに相当します。

この例では、root (HDF5 の用語では「/」と呼ばれます) と HDF4_PALGROUP の 2 つのグループがあります。

HDF4_PALGROUP は root のサブグループです。

このファイルは、HDF4 ファイルが The HDF Group が提供するユーティリティを使って HDF5 形式に変換されたものであるため、名前が「HDF4」で始まるオブジェクトが多数含まれています。

Groups リストの下には、Group Attributes リストがあります。

図では、ルートグループが選択されているため、Group Attributes リストにはルートグループの属性が表示されています。

属性はデータセットに似ていますが、HDF5 ファイルの階層構造の一部ではなく、グループやデータセットに紐付けられています。

属性は通常、グループやデータセットに関連する小さな情報を保存するために使われます。

右上のリストは Datasets リストです。

ここには、選択されているグループ（この場合は「ルート」）内のデータセットが一覧表示されます。

このファイルのルートグループには 3 つのデータセットがあり、いずれも画像です。

Datasets リストの下には、Dataset Attributes リストがあります。

ここでは、選択されたデータセット（この場合は「Raster Image #0」）の属性が表示されています。

3 つのリストには、そのリスト内の項目に関する情報を表示する列があります。

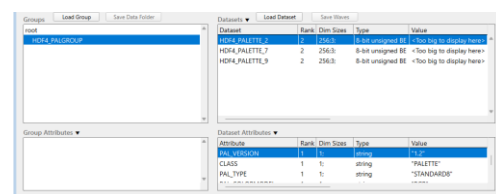
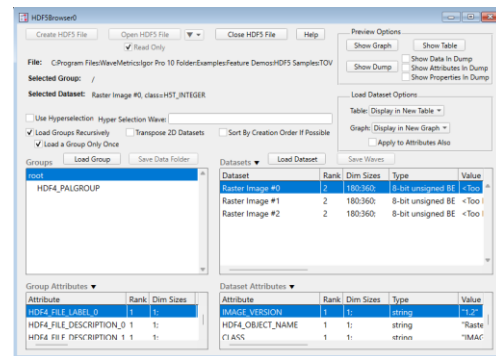
3. リストの各列に記載されている情報をよく確認してください。

すべての情報を確認するには、リストをスクロールするか、HDF5 Browser ウィンドウ全体のサイズを大きくする必要があります。

4. Groups リストでサブグループをクリックすると、他のリストに表示される情報が変化することに注目してください。

Groups リストで、ルートグループをもう一度クリックします。

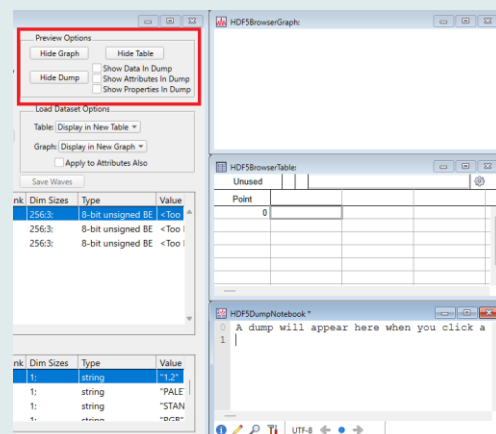
それでは、データセットを閲覧する方法を見ていきます。



5. Show Graph, Show Table, Show Dump の各ボタンをクリックし、表示される 3 つのウィンドウがすべて、少なくとも一部は見えるように配置してください。

これらのブラウザーのプレビューウィンドウは、あくまでプレビューを表示するためのものであるため、通常は比較的小さくしておくべきです。通常、HDF5 Browser の右側に配置しておくとう便利です。

3 つのウィンドウは現在、何も表示されていません。データセットや属性をクリックしたときにのみ、何かが表示されます。

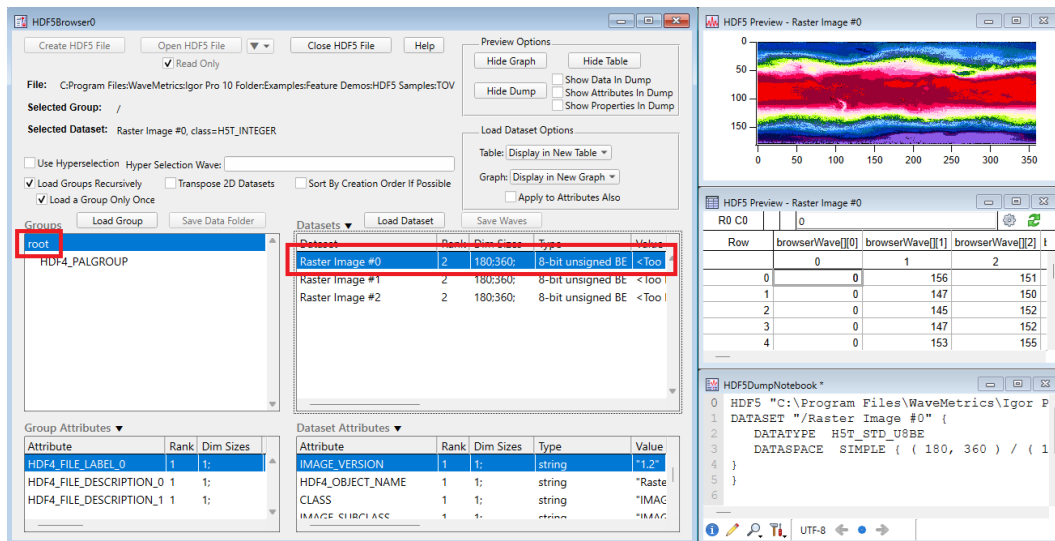


6. Datasets リストで、一番上のデータセット (Raster Image #0) をクリックします。

データセットは、3つのプレビューウィンドウそれぞれに表示されます。

ダンプウィンドウには、HDF5 ファイルの内容が「Data Description Language (データ記述言語)」（DDL）で表示されます。

これは、特定のグループ、データセット、または属性のフォーマットの詳細を確認したい専門家にとって有用です。ただし、日常的な使用においては、ダンプウィンドウを気にする必要はほとんどありません。



Show Data in Dump チェックボックスをオンにしてから、非常に大きなデータセットをクリックすると、データがダンプウィンドウに表示されるまでに非常に長い時間がかかります。

したがって、Show Data in Dump チェックボックスはオンにしないようにしてください。

プレビュー用のグラフとテーブルは、言うまでもなく、データセットをグラフやテーブルの形式でプレビューできるようにするものです。

このデータセットは特殊なケースです。

これは、HDF5 画像とパレット仕様に従ってフォーマットされた画像であり、この仕様では、画像を記述する特定の属性が必須とされています。

これらの属性は、Dataset Attributes リストで確認できます。

属性名は、CLASS、IMAGE_VERSION、IMAGE_SUBCLASS、PALETTE です。

HDF5 Browser は、これらの属性に含まれる情報を使って、見やすいプレビューグラフを作成します。

HDF5 ファイルには、HDF5 Image and Palette Specification に従ってフォーマットされていない 2D データセットを含めることができます。

実際、ほとんどの HDF5 ファイルはこの仕様に準拠していません。

ここでは、特定のデータセットが HDF5 Image and Palette Specification に従ってフォーマットされていることを明確にし、画像と見なされる可能性のある他の 2D データセットと区別するために、「正式な画像」という用語を使っています。

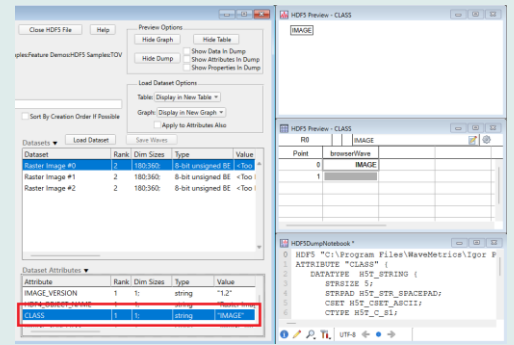
7. Dataset Attributes リストで、「CLASS」属性をクリックします。

選択した属性の値がプレビューウィンドウに表示されます。

他の画像属性、IMAGE_VERSION、IMAGE_SUBCLASS、PALETTE をクリックしてみてください。

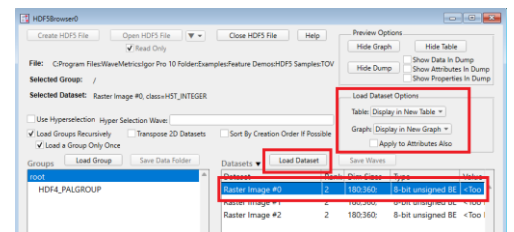
これまでのところ、データをプレビューしただけで、Igor Pro には読み込んでいません。(実際には、Igor Pro に読み込まれ、root:Packages:HDF5Browser データフォルダーに保存されていますが、それは HDF5 Browser の実装上の詳細です。)

それでは、実際に Igor Pro にデータを読み込んでみます。



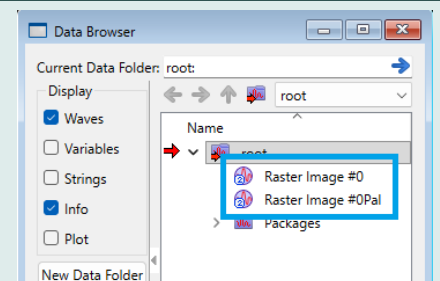
8. Raster Image #0 データセットが選択されていること、Table ポップアップメニューが「Display In New Table」に設定されていること、Graph ポップアップメニューが「Display In New Graph」に設定されていることを確認してください。その後、Load Dataset ボタンをクリックしてください。

HDF5 Browser は、データセット（と関連するパレット。これはパレットデータセットが関連付けられた正式な画像であるため）を現在のデータフォルダーに読み込み、新しいグラフと新しいテーブルを作成します。



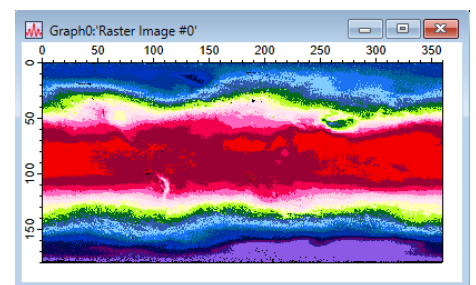
9. メニュー Data→Data Browser を選択し、ルートデータフォルダー内にある2つの「ウェーブ」のアイコンを確認してください。

「ウェーブ」は「ウェーブフォーム」の略で、Igor Pro ではデータセットを指す用語として使っています。この用語は、時系列信号処理という当社のルーツに由来しています。



10. Load Dataset ボタンをクリックすると、「Raster Image #0」と「Raster Image #0Pal」の2つのウェーブが読み込まれました。

グラフは、「Raster Image #0Pal」をパレットウェーブとして使用し、「Raster Image #0」を表示するように設定されています。

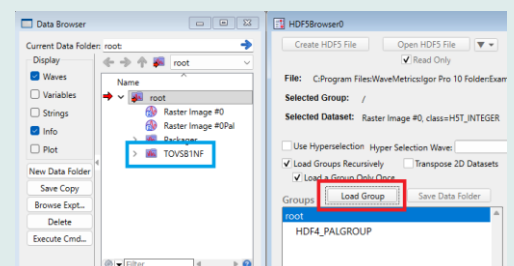


11. HDF5 Browser に戻り、ルートグループが選択されたままの状態、Load Group ボタンをクリックします。

HDF5 Browser は、「TOVSB1NF」という名前の新しいデータフォルダーを作成し、HDF5 のルートグループの内容をその新しいデータフォルダーに読み込みます。これはデータブラウザで確認できます。

「TOVSB1NF」という名前は、先ほど読み込んだ HDF5 ファイルのルートグループ名に由来しています。

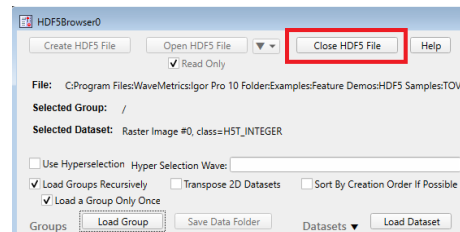
グループを読み込んだ時、HDF5 Browser には読み込まれたデータがグラフやテーブルとして表示されません。



これは、Load Dataset をクリックした場合にのみ行われ、また「Load Dataset Options」の設定にも依存します。

12. Close HDF5 File ボタンをクリックし、続いて HDF5 Browser のクローズボックスをクリックします。

Close HDF5 File ボタンをクリックせずに HDF5 Browser を閉じていた場合でも、ブラウザーは自動的にファイルを閉じます。また、File→New Experiment または File→Open Experiment を選択した場合、あるいは Igor Pro を終了した場合にも、ファイルは自動的に閉じられます。



次に、HDF5 ファイルにプログラムからアクセスするための Igor Pro プロシージャの書き方を学びます。その作業を始める前に、自分の HDF5 ファイルを使って HDF5 Browser を自由に試してみてください。まず、Data→Load Waves→New HDF5 Browser を選択します。

Igor Pro は、一般的に遭遇するほとんどの HDF5 ファイルに対応していますが、すべての HDF5 ファイルに対応しているわけではありません。

自分のファイルを検証中にエラーが発生した場合は、バグが原因であるか、あるいは Igor Pro が当該ファイルで使われている機能をサポートしていない可能性があります。

その場合は、問題の概要を簡潔に説明した上で、そのファイルを support@wavemetrics.com までお送りください。

弊社で、その問題がバグによるものか、単なる機能の制限によるものかを判断いたします。

プログラムによる HDF5 データの読み込み

Igor Pro には、HDF5 ファイルにアクセスするためのさまざまなコマンドや関数が用意されています。これらのコマンドの範囲や内容について把握するには、以下のリンクをクリックしてください。その後、ここに戻ってガイドツアーを続けてください。

このセクションでは、ユーザー定義関数を使って HDF5 データセットを読み込みます。

前のセクションの続きから説明を始めます。

1. メニュー File→New Experiment を選択します。

現在のエクスペリメントを保存するかどうかを尋ねられます。

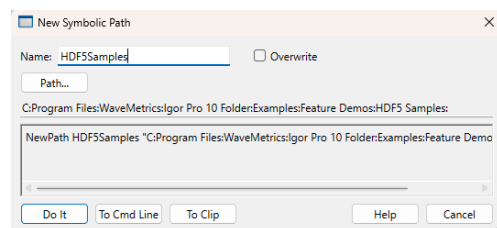
「いいえ」をクリックしてください（現在の作業環境を保存したい場合は「はい」をクリックしてください）。

これにより、現在のエクスペリメントが終了し、すべてのウェーブとウィンドウが破棄されます。

2. メニュー Misc→New Path を選択し、「HDF5Samples」という名前の Igor Pro シンボリックパスを作成します。

このパスは、前のセクションで使った TOVSB1NF.h5 ファイルが含まれている HDF5 Samples ディレクトリを指すように設定してください。

New Path ダイアログは、右のようになります。



Do It をクリックして、シンボリックパスを作成します。

Igor Pro のシンボリックパスとは、ディスク上の特定のディレクトリを参照する短縮名です。
このシンボリックパスを、HDF5 ファイルを開くコマンドで使います。

HDF5 では、まずファイルを開くか、新しいファイルを作成する必要があります。
HDF5 ライブラリはファイル ID を返し、この ID をその後のライブラリ呼び出しで使います。
処理が完了したら、ファイルを閉じる必要があります。
以下の例は、この3段階のプロセスを示しています。

3. メニュー Windows→Procedure Windows を選択し、以下の内容を Procedure ウィンドウに貼り付けてください。

```
Function TestLoadDataset(datasetName)
    String datasetName          // 読み込むデータセットの名前
    Variable fileID             // HDF5 ファイルの ID はここに保存
    Variable result = 0         // 0 はエラーなしを示す

    // HDF5 ファイルを開く
    HDF5OpenFile /P=HDF5Samples /R /Z fileID as "TOVSB1NF.h5"

    if (V flag != 0)
        Print "HDF5OpenFile failed"
        return -1
    endif

    // HDF5 データセットを読み込む
    HDF5LoadData /O /Z fileID, datasetName

    if (V flag != 0)
        Print "HDF5LoadData failed"
        result = -1
    endif

    // HDF5 ファイルを閉じる
    HDF5CloseFile fileID
    return result
End
```

プロシージャを読んでみてください。

HDF5OpenFile コマンドのパラメーターとして、先ほど作成したシンボリックパス「HDF5Samples」を使っている点に注目してください。

/R フラグは、ファイルが書き込み用に開けない場合にエラーが発生しないよう、ファイルを読み取り専用で開くために使われました。

/Z フラグは、エラーが発生した場合、HDF5OpenFile に対し、変数 V_flag を 0 以外の値に設定しつつ、エラーが発生しなかったかのように結果を返すよう指示します。
これにより、実行を中止してエラーダイアログを表示するのではなく、エラーを適切に処理することが可能になります。

HDF5OpenFile は、HDF5 ライブラリから返された値をローカル変数 fileID に代入します。
その値を HDF5LoadData コマンドに渡します。
また、読み込むそのファイル内のデータセット名も指定します。
ここでも、V_flag 変数をチェックして、エラーが発生したかどうかを確認します。

最後に、HDF5CloseFile を呼び出して、開いたファイルを閉じます。
開いたファイルは必ず閉じるようにすることが重要です。
プロシージャの開発中に、ファイルを閉じるのを忘れたり、エラーのせいでファイルを閉じられなかったりした場合は、次のコマンドを実行することで、開いているすべての HDF5 ファイルを閉じることができます。

HDF5CloseFile /A 0

また、File→New Experiment、File→Revert Experiment、File→Open Experiment を選択した場合、または IgorPro を終了した場合、開いているすべてのファイルを自動的に閉じます。

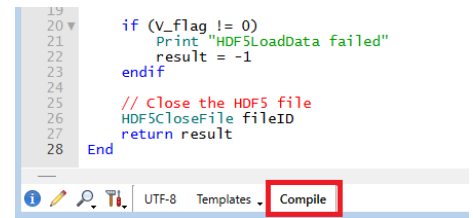
4. Procedure ウィンドウの下部にある Compile ボタンをクリックしてください。

Procedure ウィンドウの下部に Compile ボタンが表示されない場合は、Procedure ウィンドウを非表示にした時点で、プロシージャがすでに自動的にコンパイルされていたためです。

コンパイルエラーが発生した場合は、Procedure ウィンドウに正しいテキストを貼り付けていないか、あるいはその他の誤ったテキストを入力してしまった可能性があります。

前の手順に戻って、もう一度試してみてください。

これで、プロシージャを実行する準備が整いました。



```
19
20
21 if (V_Flag != 0)
22     Print "HDF5LoadData failed"
23     result = -1
24 endif
25
26 // Close the HDF5 file
27 HDF5CloseFile fileID
28 return result
29 End
```

5. Igor Pro のコマンドラインで、以下のコマンドを実行してください。

コマンドを入力して Enter キーを押す、コピー&ペーストして Enter キーを押す、またはコマンドを選択して Ctrl+Enter キーを押すのいずれかの方法で行ってください。

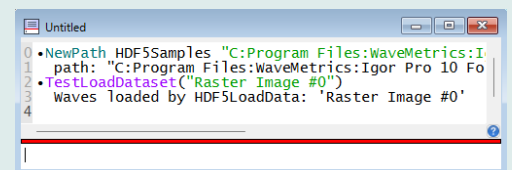
```
TestLoadDataset("Raster Image #0")
```

この時点で、プロシージャは正常に実行されているはずであり、履歴エリア（コマンドラインのすぐ上）にはエラーメッセージが表示されていないはずです。

コマンドウィンドウのこの履歴エリアには、以下の内容が表示されているはずです。

Waves loaded by HDF5LoadData: Raster Image #0

それでは、データが読み込まれたことを確認します。

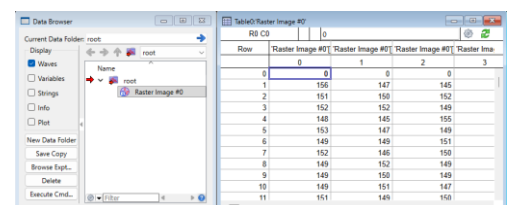


```
Untitled
•NewPath HDF5Samples "C:\Program Files\WaveMetrics\Igor Pro 10 Fo
path: "C:\Program Files\WaveMetrics\Igor Pro 10 Fo
•TestLoadDataset("Raster Image #0")
Waves loaded by HDF5LoadData: 'Raster Image #0'
```

6. メニュー Data→Data Browser を選択します。

次に、「Raster Image #0」という名前のウェーブをダブルクリックします。

これにより、読み込まれたばかりのデータを表示するテーブルが作成されます。



Row	Raster Image #0	Raster Image #0	Raster Image #0	Raster Image #0
0	0	0	0	0
1	155	147	145	
2	151	150	152	
3	152	152	149	
4	148	149	150	
5	153	147	149	
6	149	149	151	
7	152	146	150	
8	149	152	149	
9	149	150	149	
10	149	151	147	
11	151	149	150	

この時点では、ルートデータフォルダーには「Raster Image #0」という1つのウェーブファイルしか含まれておらず、対応するパレットウェーブファイルは含まれていません。

これは、データセットを正式な画像としてではなく、単なるデータセットとして読み込んだためです。

HDF5LoadData コマンドは、正式な画像について認識していません。

そのためには、HDF5LoadImage を使う必要があります。

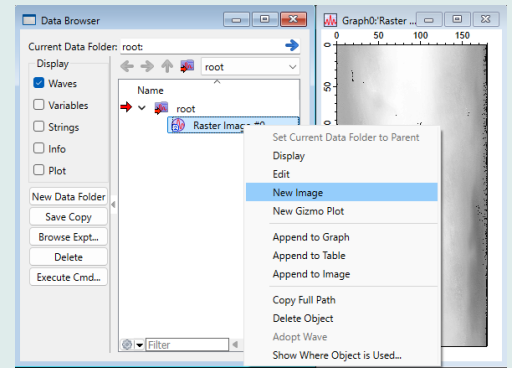
7. Data Browser で、「Raster Image #0」というウェーブを右クリックし、New Image を選択します。

Igor Pro は、そのウェーブの画像プロットを作成します。

「Raster Image #0」を正式な画像としてではなく、単なるデータセットとして読み込んだため、画像プロットは色付きではなく灰色になっています。

また、このツアーで先ほど見た画像プロットに比べて 90 度回転しています。

これは、Igor Pro が 2D データを他のほとんどのプログラムとは異なる方法でプロットするためであり、HDF5LoadData コマンドではデフォルトではこの補正が行われなかったためです。



このプロシージャは、特定のシンボリックパスと特定のファイル名を使うようにハードコーディングされているため、その有用性はかなり限定的です。

そこで、これをより汎用的なものにします。

その作業を行う前に、現在の作業環境を保存しておきます。

そうすれば、万が一ミスをしても、正常に動作していた状態に戻すことができます。

8. メニュー File→Save Experiment As を選択し、現在の作業環境を「HDF5 Tour.pxp」という名前の新しいエクスペリメントファイルとして、自分のファイルを保存しているディレクトリに保存してください。

これにより、すべてのデータとプロシージャが 1 つのファイルに保存されます。

必要に応じて、後で File→Revert Experiment を選択することで、保存された状態に戻すことができます。

9. Procedure ウィンドウ（メニュー Windows→Procedure Windows→Procedure Window）を開き、TestLoadDataset 関数の最初の 2 行を次のように置き換えてください。

```
Function TestLoadDataset(pathName, fileName, datasetName)
    String pathName           // Name of symbolic path
    String fileName           // Name of HDF5 file
    String datasetName        // Name of dataset to be loaded
```

10. HDF5OpenFile コマンドを次のように変更してください。

```
HDF5OpenFile /P=$pathName /R /Z fileID as filename
```

ここでは、HDF5Samples を \$pathName に置き換えました。

\$pathName により、文字列パラメーター pathName の内容を /P フラグのパラメーターとして使うように指示されます。

また、「TOVSB1NF.h5」を fileName に置き換えました。

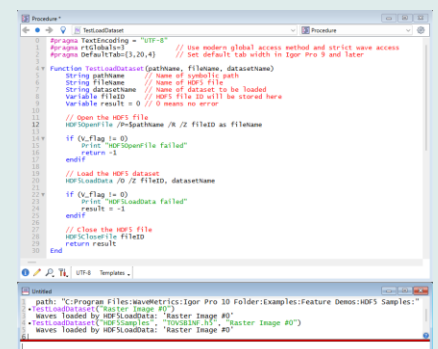
これにより、コード内で指定するのではなく、関数を呼び出す時に読み込むファイルを指定できるようになります。

11. Compile ボタンをクリックし、コマンドラインで次のコマンドを実行してください。

```
TestLoadDataset("HDF5Samples", "TOVSB1NF.h5",
"Raster Image #0")
```

これは、以前のバージョンの関数と同じ動作をしますが、今では、どのディレクトリにあるどのファイルに対しても使用できる、より汎用性の高い関数となっています。

上記のコマンドにより、「Raster Image #0」データセットが Igor に再読み込みされ、以前の内容が上書きされました。



新しい内容は以前の内容と同一であるため、グラフやテーブルに変化はありません。

12. File→Save Experiment を選択し、現在の実行環境を、先ほど作成した **HDF5 Tour.pxp** エクスperiment ファイルに保存します。

もしよければ、ここで一息ついて、Igor Pro を終了しても構いません。

プログラムによる HDF5 データの保存

HDF5 データをプログラムで読み込む方法を確認したので、次は Igor Pro データを HDF5 ファイルに保存する方法について見ていきます。

このセクションでは、Igor Pro のデータを作成し、ユーザー定義関数から HDF5 データセットとして保存します。

1. 以前に保存した「HDF5 Tour.pxp」のエクスperiment をまだ開いていない場合は、File→Recent Experiments→HDF5 Tour.pxp を選択して開いてください。

次に、HDF5 ファイルに保存できるデータを作成します。

他のデータと区別するため、新しいデータフォルダー内にこのデータを作成します。

2. Data Browser が開いていない場合、Data→Data Browser を選択します。

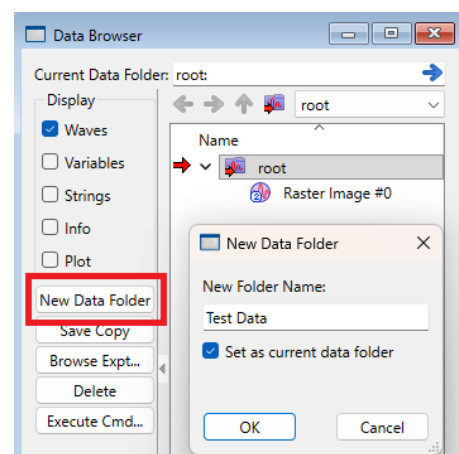
New Data Folder ボタンをクリックします。

「Test Data」という名前を入力し、**Set As Current Data Folder** をチェックしてから、**OK** をクリックします。

Data Browser には、新しいデータフォルダーと、それを指す赤い矢印が表示されます。

赤い矢印は、現在のデータフォルダーを示しています。

特定のデータフォルダーを明示的に指定しない操作は、現在のデータフォルダーに対して実行されます。



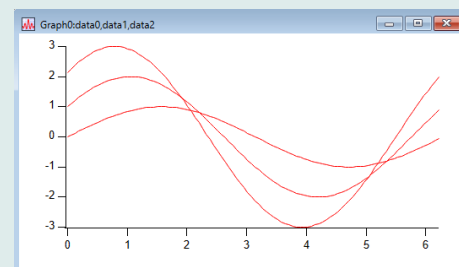
3. Igor Pro のコマンドラインで、以下のコマンドを実行します。

```
Make/N=100 data0, data1, data2
```

```
SetScale x 0, 2*PI, data0, data1, data2
```

```
data0=sin(x); data1=2*sin(x+PI/6);  
data2=3*sin(x+PI/4)
```

```
Display data0, data1, data2
```

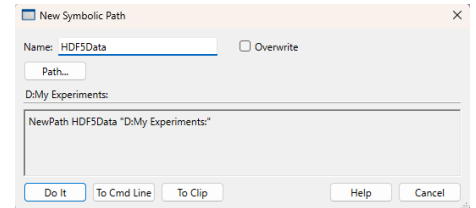


SetScale コマンドにより、各ウェーブの X 軸のスケーリング範囲が 0 から 2π になるように設定されます。ウェーブ代入文中の変数 x は、代入が実行される時に、宛先となるウェーブの各ポイントにおける x 値を取り込みます。

(まだ行っていない場合は、後で Help→Getting Started を選択して、Igor Pro のガイドツアーを実行してください。そこでは、X スケーリングやウェーブ代入文をはじめ、Igor Pro のその他の多くの機能について解説されています。)

4. Misc→New Path を選択し、データを保存するハードディスク上のディレクトリを指定して、「HDF5Data」という名前の Igor Pro シンボリックパスを作成してください。

Misc→Path Status を選択し、HDF5Data のシンボリックパスが存在し、目的のディレクトリを指していることを確認してください。



保存する価値のある作業をさらにいくつか行っただけで、必要に応じて正常な状態に復元できるように、現在のエクスペリメント内容を保存しておきます。

5. File→Save Experiment を選択し、現在の作業環境を、先ほど作成した HDF5 Tour.pxp エクスペリメントファイルに保存してください。

必要に応じて、後で File→Revert Experiment を選択することで、保存された状態に戻すことができます。

6. Windows→Procedure Windows→Procedure Window を選択し、TestLoadDataset 関数の下に以下を貼り付けてください。

```
Function TestSaveDataset(pathName, fileName, w)
    String pathname                // シンボリックパス名
    String filename                // HDF5 ファイル名
    Wave w                        // 保存するウェーブ

    Variable result = 0           // 0 はエラーなしを意味する
    Variable fileID

    // 新しい HDF5 ファイルを作成。同じ名前が既に存在する場合は上書き
    HDF5CreateFile/P=$pathname /O /Z fileID as fileName

    if (V_flag != 0)
        Print "HDF5CreateFile failed"
        return -1
    endif

    // ウェーブをデータセットとして保存
    HDF5SaveData /O /Z w, fileID

    if (V_flag != 0)
        Print "HDF5SaveData failed"
        result = -1
    endif

    // HDF5 ファイルを閉じる
    HDF5CloseFile fileID

    return result
End
```

プロシージャをよく読んでください。

以前作成した TestLoadDataset 関数と似ているため、見覚えがあるはずです。

ここでも、シンボリックパスとファイル名を指定するパラメーターがあります。

HDF5SaveData は、特に指定がない限りウェーブ名をデータセット名として使うため、データセット名を指定するパラメーターはありません。

このプロシージャにはウェーブパラメーターがあり、これを通じて HDF5 ファイルに書き込むデータのウェーブを指定します。

ここでは、HDF5OpenFile ではなく HDF5CreateFile を使って新しいファイルを作成します。
HDF5CreateFile は新しいファイルを作成して開き、ファイル ID を返します。
既存のファイルにデータセットを追加したい場合は、HDF5CreateFile ではなく HDF5OpenFile を使います。

HDF5CreateFile は、ローカル変数 fileID に HDF5 ライブラリから返された値を設定します。
その値を HDF5SaveData コマンドに渡します。
/O フラグを指定すると、同じ名前のデータセットがすでに存在する場合、それが上書きされます。

最後に、HDF5CreateFile で開いたファイルを閉じるために、HDF5CloseFile を呼び出します。

7. Procedure ウィンドウの下にある Compile ボタンをクリックします。

エラーが表示された場合は、Procedure ウィンドウに正しいテキストを貼り付けていないか、あるいはその他の誤ったテキストを入力した可能性があります。

エラーの原因が特定できない場合は、File→Revert Experiment を選択し、このセクションの手順 6 に戻ってください。

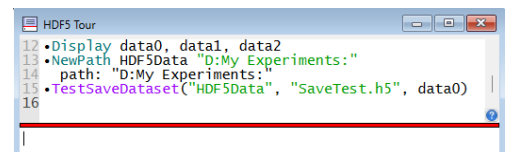
これで、プロシージャを実行する準備が整いました。

8. Igor Pro コマンドラインで、以下のコマンドを実行します。 コマンドを入力して Enter キーを押すか、コピー&ペーストして Enter キーを押してください。

```
TestSaveDataset("HDF5Data", "SaveTest.h5", data0)
```

この時点で、プロシージャは正常に実行されているはずであり、コマンドウィンドウの履歴エリア（コマンドラインのすぐ上）にはエラーメッセージが表示されていないはずです。

それでは、データが保存されたことを確認します。

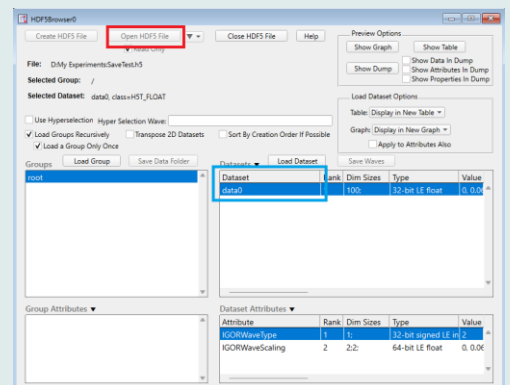


9. Data→Load Waves→New HDF5 Browser を選択します。 Open HDF5 File ボタンをクリックし、先ほど作成した SaveTest.h5 ファイルを開きます。

そのファイルに「data0」という名前のデータセットが含まれていることを確認してください。

データセット「data0」にはいくつかの属性があります。
これらの属性により、HDF5LoadData 関数を使ってこのデータを再度読み込んだ場合、ウェーブとそのすべてのプロパティを完全に再現することができます。

これらの属性を保存したくない場合は、HDF5SaveData を呼び出す時に /IGOR=0 フラグを使ってください。



10. Close HDF5 File ボタンをクリックします。

HDF5 Browser でファイルが開かれている間は、そのファイルにデータを追加で書き込むことはできません。

TestSaveDataset 関数は、1つのウェーブしか保存できないため、かなり制限があります。
そこで、任意の数のウェーブを保存できるように、この関数をより汎用的なものにします。
その前に、現在、正常な状態にあるため、まず作業内容を保存しておきます。

11. File→Save Experiment を選択し、現在の作業環境を、先ほど作成した HDF5 Tour.pxp エクスペリメントファイルに保存します。

必要に応じて、後で File→Revert Experiment を選択することで、保存された状態に戻すことができます。

次に、別の名前 (TestSaveDataset ではなく TestSaveDatasets) をつけた、より汎用性の高い新しいユーザー関数を作成します。

12. Windows→Procedure Windows→Procedure Window を選択し、以下のコードを「TestSaveDataset」関数の下に貼り付けてください。

```
Function TestSaveDatasets(pathName, fileName, listOfWaves)
    String pathname          // シンボリックパス名
    String filename          // HDF5 ファイル名
    String listOfWaves       // セミコロンで区切られたウェーブのリスト

    Variable result = 0      // 0 はエラーなしを意味する
    Variable fileID

    // 新しい HDF5 ファイルを作成。同じ名前が既に存在する場合は上書き
    HDF5CreateFile/P=$pathname /O /Z fileID as fileName

    if (V_flag != 0)
        Print "HDF5CreateFile failed"
        return -1
    endif

    String listItem
    Variable index = 0

    do
        listItem = StringFromList(index, listOfWaves)

        if (strlen(listItem) == 0)
            break          // 次のウェーブなし
        endif

        // ウェーブに対するローカル参照を作成
        Wave w = $listItem

        // ウェーブをデータセットとして保存
        HDF5SaveData /O /Z w, fileID

        if (V_flag != 0)
            Print "HDF5SaveData failed"
            result = -1
            break
        endif

        index += 1
    while(1)

    // HDF5 ファイルを閉じる
    HDF5CloseFile fileID

    return result
End
```

プロシージャをよくお読みください。これは TestSaveDataset 関数と似ています。

最初の違いは、ウェーブそのものを渡すのではなく、文字列パラメーターとしてウェーブ名のリストを渡すという点です。

このパラメーターはセミコロンで区切られたリストであり、これは Igor Pro でよく使われるプログラミング手法です。

次の違いは、リストから名前を取り出し、対応するウェーブを HDF5 ファイル内のデータセットとして保存する do-while ループがあるという点です。

```
Wave w = $ListItem
```

は、ローカルの「ウェーブ参照」を作成し、リストアイテム変数「listItem」の内容によってそのウェーブの識別子が決定されるウェーブを、w を使って参照できるようにします。

これも Igor Pro プログラミングにおける一般的な手法であり、プログラミングのヘルプファイルで詳しく説明しています。

13. Procedure ウィンドウの下部にある Compile ボタンをクリックします。

エラーが表示された場合は、Procedure ウィンドウに正しいテキストを貼り付けていないか、またはその他の誤ったテキストを入力している可能性があります。

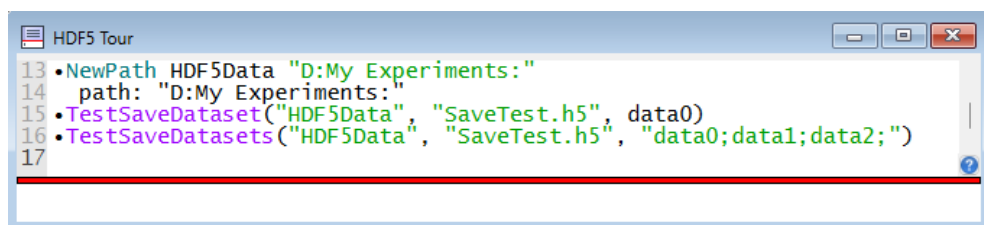
エラーの原因が特定できない場合は、File→Revert Experiment を選択し、このセクションのステップ 12 に戻ってください。

これで、プロシージャを実行する準備が整いました。

14. Igor Pro のコマンドラインで、以下のコマンドを入力してください。

入力して Enter キーを押す、コピー＆ペーストしてから Enter キーを押す、のいずれかの方法で実行できます。

```
TestSaveDatasets("HDF5Data", "SaveTest.h5", "data0;data1;data2;")
```



この関数では、3 番目のパラメーターは文字列です。

前の例では、これは名前、具体的にはウェーブの名前でした。

文字列は二重引用符で囲んで指定しますが、名前はそうではありません。

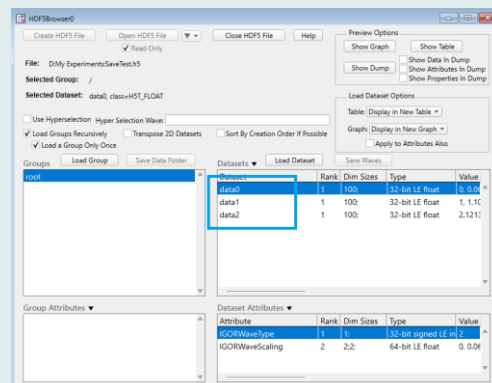
この時点で、プロシージャは正常に実行されているはずであり、コマンドウィンドウの履歴エリア（コマンドラインのすぐ上）にはエラーメッセージが表示されていないはずです。

それでは、データが保存されたことを確認します。

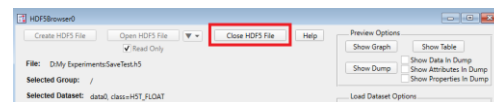
15. HDF5 Browser ウィンドウを起動します（Data→Load Waves→New HDF5 Browser）。

Open HDF5 File ボタンをクリックし、先ほど作成した SaveTest.h5 ファイルを開きます。

そのファイルに「data0」、「data1」、「data2」という名前のデータセットが含まれていることを確認してください。



16. Close HDF5 File ボタンをクリックします。



17. File→Save Experiment を選択し、現在の作業環境を HDF5 Tour.pxp エクスperimentファイルとして保存します。

これで HDF5 ガイドツアーは終了です。

今後の進め方

Igor Pro を初めて使う方や、これまで使ったことがない方は、ぜひ「Igor Pro ガイドツアー」を学習してください。

急いている場合は、その前半部分だけでも学習してください。

また、Igor Pro の基本についてより詳しく解説している以下のヘルプファイルも併せてお読みください。

Getting Help, Experiments, Files and Folders, Windows, Waves

Igor Pro のプログラミングを始めるには、以下のヘルプセクションをお読みください。

Working With Commands, Programming Overview, User-Defined Functions

HDF5 独自のプログラミングを行うには、HDF5 について少なくとも基本的な理解が必要です。

次のセクションのリンクを参照してください。

その後、「HDF5 のコマンドと関数」のセクションに記載している HDF5 関連のコマンドや関数について理解を深めておく必要があります。

HDF5 についてさらに詳しく

HDF5 の操作を行うには、HDF5 について少なくとも基本的な理解が必要です。

HDF5 のウェブサイトには、豊富な資料が掲載されています。

まずは、以下のウェブページをご覧ください。

https://support.hdfgroup.org/documentation/hdf5/latest/_learn_basics.html

HDF グループは、「HDFView」という Java ベースのプログラムを提供しています。

入門資料を読み進めながら HDF5 ファイルを簡単に閲覧できるように、HDFView をダウンロードしてインストールしておくとい良いでしょう。

あるいは、Igor Pro が提供する HDF5 Browser を使うこともできます。

HDF5 Browser

Igor Pro には、HDF5 ブラウザーを実装した、自動的に読み込まれるプロシージャファイル「HDF5 Browser.ipf」が含まれています。

このブラウザーを使うと、HDF5 ファイルを対話的に確認して、その内容を確認することができます。

また、HDF5 データセットやグループを読み込んだり、Igor Pro のウェーブやデータフォルダーを HDF5 ファイルとして保存したりすることも可能です。

このブラウザーでは属性の作成に対応していません。

属性を作成するには、HDF5SaveData コマンドを使う必要があります。

HDF5 Browser には、以下の情報を表示するリストが含まれています。

- ・ ファイル内のすべてのグループ
- ・ 選択されたグループのすべての属性
- ・ 選択されたグループのすべてのデータセット
- ・ 選択されたデータセットのすべての属性

さらに、HDF5 Browser では、オプションで以下の情報を表示します。

- ・ 選択されたデータセットまたは属性を表示するグラフ
- ・ 選択されたデータセットまたは属性を表示するテーブル
- ・ 選択されたデータセットまたは属性のダンプを含むノートブックウィンドウ

HDF5 Browser の使用方法

HDF5 ファイルを参照するには、メニュー Data→Load Waves→New HDF5 Browser を選択します。

これにより、HDF5 Browser のコントロールパネルが表示されます。

同じメニュー項目を再度選択することで、ブラウザーを追加で作成できます。

各ブラウザーのコントロールパネルでは、一度に1つの HDF5 ファイルを閲覧できます。

ほとんどのユーザーにとっては、ブラウザーが1つあれば十分でしょう。

新しいブラウザーを作成したら、Open HDF5 File ボタンをクリックして、参照するファイルを選択してください。

HDF5 Browser には、閲覧中のファイルに含まれるグループ、グループ属性、データセット、データセット属性を表示する4つのリストが含まれています。

HDF5 Browser の基本的なコントロール

ここでは、ほとんどのユーザーが使う HDF5 Browser の基本的な操作方法について説明します。

Create HDF5 File ボタン

新しい HDF5 ファイルを作成し、読み書き用に開きます。

Open HDF5 File ボタン

Read Only チェックボックスの状態に応じて、既存の HDF5 ファイルを読み取り専用または読み書き可能で開きます。

Close HDF5 File ボタン

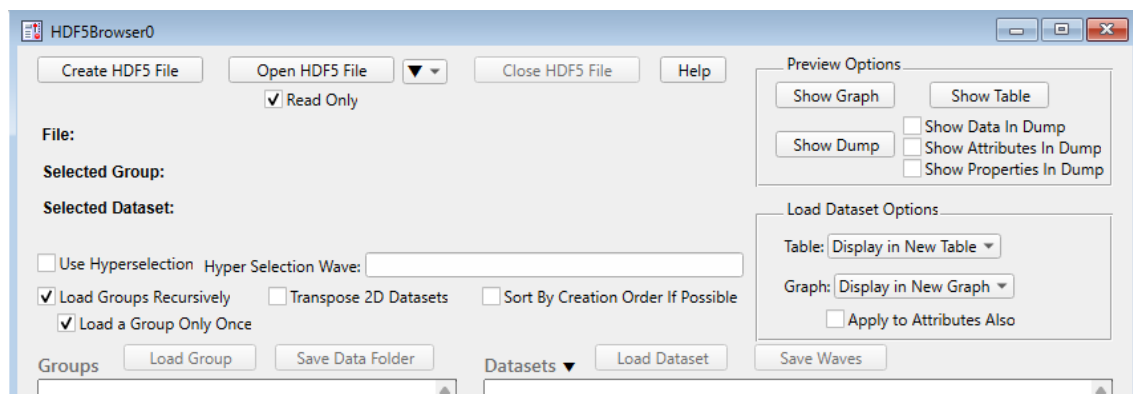
HDF5 ファイルを閉じます。

Show Graph ボタン

Show Graph ボタンをクリックすると、ブラウザーには、その後選択したデータセットや属性のプレビューグラフが表示されます。

Show Table ボタン

Show Table ボタンをクリックすると、ブラウザーには、その後選択するデータセットや属性のプレビューテーブルが表示されます。



Load Dataset ボタン

Load Dataset ボタンをクリックすると、現在選択されているデータセットが現在のデータフォルダーに読み込まれます。

Load Dataset Options エリア

ブラウザーのこのセクションには、Load Dataset ボタンをクリックして読み込んだデータを、テーブルで表示するかグラフで表示するかを決める2つのポップアップメニューがあります。

Table ポップアップメニューには、No Table、Display in New Table、Append To Top Table の3つの項目があります。

Append To Top Table を選択しても、テーブルが存在しない場合は、Display in New Table を選択した場合と同じように動作します。

Graph ポップアップメニューには、No Graph、Display in New Graph、Append To Top Graph の3つの項目があります。

Append To Top Graph を選択しても、グラフが存在しない場合は、Display in New Graph を選択した場合と同様に動作します。

1D データを読み込む時には追加機能が便利ですが、多次元データを追加する場合にはあまり役に立ちません。多次元データは画像プロットとして追加されるため、グラフにすでに表示されていた内容が隠れてしまいます。

Display in New Graph または Append To Top Graph を選択しても、テキストのウェーブはグラフに表示されません。

Save Waves ボタン

HDF5 ファイルが読み書き可能な状態で開かれている場合、そのファイル内のウェーブを選択して保存できるパネルが表示されます。

Transpose 2D Datasets チェックボックス

Transpose 2D Datasets チェックボックスがオンになっている場合、Igor Pro と他のプログラムでは画像プロットにおける行と列の扱いに違いがあるため、その違いを補正するために 2D データセットが転置されます。

詳細は、「HDF5 画像と Igor Pro 画像の比較」のセクションを参照してください。

これは、「正式な」画像（HDF5 画像とパレット仕様に従ってフォーマットされた画像）の読み込みには影響しません。

Sort By Creation Order If Possible チェックボックス

このチェックボックスがオンになっており、かつファイルが作成順での一覧表示に対応している場合、HDF5 Browser はグループとデータセットを作成順に表示・読み込みます。

ほとんどの HDF5 ファイルには作成順の情報が含まれていないため、このチェックボックスにチェックを入れていても、アルファベット順に一覧表示および読み込まれます。

ただし、Igor Pro 9 以降で作成された HDF5 ファイルには作成順の情報が含まれているため、作成順に一覧表示、読み込むことができます。

Load Group ボタン

Load Group ボタンをクリックすると、現在選択されているグループ内のすべてのデータセットが、現在のデータフォルダー内の新しいデータフォルダーに読み込まれます。

Load Group ボタンは、/IMAG フラグを使って HDF5LoadGroup コマンドを呼び出します。

つまり、グループに正式な画像が含まれている場合（「HDF5 画像 と Igor Pro 画像の比較」のセクションを参照）、それは正式な画像として読み込まれます。

Hyperselection コントロールは、Load Group ボタンの動作には適用されません。

Load Groups Recursively チェックボックス

Load Groups Recursively チェックボックスがオンになっている場合、Load Group ボタンを押すと、現在選択されているグループ内のサブグループがすべて、サブデータフォルダとして読み込まれます。

Load Group Only Once チェックボックス

HDF5 ファイルは厳密な階層構造ではなく「有向グラフ」であるため、HDF5 ファイル内の特定のグループは、そのファイルの階層構造内で複数の場所に現れることがあります。

Load Group Only Once チェックボックスがオンになっている場合、Load Group ボタンをクリックすると、指定されたグループは最初に検出されたときのみ読み込まれます。

オフになっている場合、Load Group ボタンをクリックするたびに、ファイルの階層内に指定されたグループが現れるたびに読み込まれるため、データが重複することになります。

迷った場合は、Load Group Only Once のチェックを入れたままにしておいてください。

Save Data Folder ボタン

ファイルが読み書き可能な状態で開かれている場合、1つのデータフォルダーを選択して HDF5 ファイルに保存できるパネルが表示されます。

HDF5 Browser のコンテキストメニュー

Datasets、Dataset Attributes、または Group Attributes の一覧で行を右クリックすると、ブラウザーにコンテキストメニューが表示され、以下の操作を行うことができます。

- ・ 選択したデータセットまたは属性の値をテキストとしてクリップボードにコピーする
- ・ 選択したデータセットまたは属性をウェーブとして読み込む
- ・ すべてのデータセットまたは属性をウェーブとして読み込む

リストの上にある ▼ アイコンをクリックしても、これらのポップアップメニューが表示されます。

これらは、前述の HDF5 Browser のコンテキストメニューと同じように機能します。

Datasets リストがアクティブな状態で、Load Selected Dataset as Wave を選択すると、Load Dataset ボタンをクリックした場合と同じ動作が行われます。

データをクリップボードにコピーする場合、データはテキストとしてコピーされるため、元のデータセットや属性の精度が完全に反映されない可能性があります。

すべてのデータ型がサポートされているわけではありません。

右クリックしたデータ型をブラウザーがサポートしている場合、コンテキストメニューに Copy to Clipboard as

Text と表示されます。

データ型がサポートされていない場合は、Can't Copy This Data Type と表示されます。

ウェーブとして読み込む時、同じ名前を持つ既存のウェーブは上書きされます。

データセットからウェーブを読み込む時は、ブラウザーの Load Dataset Options セクションにあるテーブルとグラフのオプションが適用されます。

Apply to Attributes にもチェックを入れると、属性からウェーブを読み込む時にもこれらのオプションが適用されます。

HDF5 Browser の高度なコントロール

ここでは、HDF5 Browser の高度な操作機能について説明します。

これらは、HDF5 ファイル形式に精通しているユーザー向けの機能です。

Show Dump ボタン

Show Dump ボタンをクリックすると、ブラウザーにノートブックが表示され、そこで選択したグループ、データセット、または属性に関する追加の詳細情報を確認できます。

いずれかのリストからグループ、データセット、または属性を選択するたびに、ダンプウィンドウが更新されます。

Show Data In Dump チェックボックス

このチェックボックスがオフの場合、ダンプにはデータセットのヘッダー情報は表示されますが、実際のデータは表示されません。

このチェックボックスがオンの場合、データセットのヘッダー情報に加えてデータも表示されます。

注意

Show Data In Dump チェックボックスをオンにして、非常に大規模なデータセットのダンプを実行すると、ダンプ処理に非常に長い時間がかかる場合があります。

ダンプ処理がいつまでも終わらないようであれば、Igor Pro のステータスバーにある Abort ボタンをクリックしてください。

Show Data In Dump チェックボックスがオンになっている場合でも、グループのダンプにはヘッダー情報のみが含まれ、データセットや属性の実際のデータは省略されます。

Show Attributes In Dump チェックボックス

Show Attributes In Dump チェックボックスを使うと、グループまたはデータセットを選択した時に、属性をダンプに出力するかどうかを指定できます。

このチェックボックスをオンにすると、データセットに関連付けられている属性に関する情報がダンプに含まれます。

このチェックボックスは、グループまたはデータセットの属性リストで項目を選択した時にダンプされる内容には影響しません。

Show Properties In Dump チェックボックス

Show Properties In Dump チェックボックスをオンにすると、ストレージレイアウトやフィルター（圧縮）などのプロパティを確認できます。

この情報は通常、あまり重要ではありませんが、圧縮の影響を調査する時には役立ちます。

Use Hyperselection チェックボックス

Use Hyperselection チェックボックスをオンにし、Hyper Selection Wave へのパスを入力すると、HDF5 Browser はそのウェーブ内のハイパーセクションを使って、ユーザーがクリックした後続のデータセットまたは属性のサブセットを読み込みます。

これは、HDF5 のハイパーセクションを理解しており、以下の「HDF5 データセットのサブセット」のセクションに関する説明を読んだことのある上級ユーザー向けの機能です。

Load Dataset ボタンをクリックした場合はハイパーセクションが使われますが、Load Group ボタンをクリックした場合は使われません。

HDF5 Browser ダンプの技術的な詳細

ダンプノートブックには、選択したグループ、データセット、または属性のダンプが「データ記述言語」（DDL）形式で表示されます。

通常、このダンプウィンドウを使う必要はありません。

この機能は、問題のデバッグを行う専門家や、HDF5 の仕組みを深く理解しようとしているユーザーにとって有用です。

HDF5 ファイル内の文字列には、末尾に多数のヌル文字が含まれていることがあります。

これらはダンプウィンドウには表示されません。

HDF5 ファイル内の文字列には、キャリッジリターン、ラインフィード、タブを表すリテラル文字列

“\r”、“\n”、“\t” が含まれていることがあります。

可読性を高めるため、ダンプウィンドウでは、これらのリテラル文字列が実際のキャリッジリターン、ラインフィード、タブとして表示されます。

HDF5 コマンドと関数

このセクションでは、Igor Pro の HDF5 に関連するコマンドと関数を一覧で示します。

HDF5CreateFile	新しい HDF5 ファイルを作成するか、既存のファイルを上書きします。
HDF5OpenFile	HDF5 ファイルを開き、他のコマンドや関数に渡されるファイル ID を返します。
HDF5CloseFile	1 つの HDF5 ファイル、または開いているすべての HDF5 ファイルを閉じます。
HDF5FlushFile	1 つの HDF5 ファイル、または開いているすべての HDF5 ファイルをフラッシュします。
HDF5CreateGroup	HDF5 ファイル内にグループを作成し、他のコマンドや関数に渡すことができるグループ ID を返します。
HDF5OpenGroup	既存の HDF5 グループを開き、他のコマンドや関数に渡すことができるグループ ID を返します。
HDF5ListGroup	グループ内のすべてのオブジェクトを一覧表示します。
HDF5CloseGroup	HDF5 グループを閉じます。
HDF5LinkInfo	HDF5 リンクに関する情報を返します。
HDF5ListAttributes	グループ、データセット、またはデータ型に関連付けられているすべての属性を一覧表示します。
HDF5AttributeInfo	HDF5 属性に関する情報を返します。
HDF5DatasetInfo	HDF5 データセットに関する情報を返します。
HDF5LoadData	HDF5 データセットまたは属性からデータを Igor Pro に読み込みます。

HDF5LoadImage	HDF5 画像とパレット仕様バージョン 1.2 に基づいて記述された画像を読み込みます。
HDF5LoadGroup	HDF5 グループとそのデータセットを Igor Pro のデータフォルダーに読み込みます。
HDF5SaveData	Igor Pro のウェーブデータを HDF5 ファイルに保存します。
HDF5SaveImage	HDF5 画像とパレット仕様バージョン 1.2 で規定された形式で画像を保存します。
HDF5SaveGroup	Igor Pro データフォルダーを HDF5 グループに保存します。
HDF5TypeInfo	HDF5 のデータ型に関する情報を返します。
HDF5CreateLink	新しいハードリンク、ソフトリンク、または外部リンクを作成します。
HDF5UnlinkObject	HDF5 ファイルからオブジェクト（グループ、データセット、データ型、またはリンク）のリンクを解除します。これにより、ファイルの階層構造からそのオブジェクトが削除されますが、ファイル内でそのオブジェクトが使っていた領域は解放されません。
HDF5DimensionScale	HDF5 の次元スケールの作成および照会をサポートしています。
HDF5LibraryInfo	Igor Pro で使われている HDF5 ライブラリに関する情報を返します。これは、上級プログラマのみが関心を寄せる内容です。
HDF5Control	Igor Pro による HDF5 ファイル形式の使用に関する各種設定をコントロールします。
HDF5Dump	グループ、データセット、または属性の DDL 形式のダンプを返します。
HDF5DumpErrors	Igor Pro で発生した HDF5 関連のエラーに関する情報を返します。これは専門家向けの診断ツールであり、ごく稀な場合にのみ必要となります。

HDF5 プロシージャファイル

IgorPro には、HDF5 の使用とプログラミングをサポートするための 2 つのプロシージャファイルが付属しています。

これらのファイルは、Igor Pro の起動時に自動的に読み込まれるため、常に利用可能です。

HDF5 Browser.ipf は、HDF5 Browser を実装しています。

HDF5 Utilities.ipf は、HDF5 に関連する定数を定義し、HDF5 の機能を利用するプロシージャを作成する時に役立つ HDF5 関連のユーティリティーチェーンを提供する、パブリックプロシージャファイル（つまり、独立したモジュールではない）です。

独自のプロシージャファイルを作成する場合、HDF5 Utilities.ipf に含まれる定数やユーティリティーチェーンを、`#include` 命令を使わずに利用できます。

ただし、HDF5 プログラミング用の独自の独立モジュールを作成する場合は、その独立モジュールに HDF5 Utilities.ipf を `#include` する必要があります。

例えば、HDF5 Browser.ipf のプロシージャを参照してください。

HDF5 の属性

属性とは、HDF5 のグループ、データセット、または名前付きデータ型に紐付けられたデータのことです。

属性を読み込むには、HDF5LoadData コマンドを使い、`/A=attributeNameStr` フラグと `/TYPE=objectType` フラグを指定する必要があります。

H5T_COMPOUND 型の属性（化合物、すなわち構造）の読み込みはサポートされていません。

HDF5 の数値属性の読み込み

次の関数は、グループまたはデータセットの数値型属性を読み込む方法を示しています。

この関数の戻り値はエラーコードです。

属性の値は、参照渡しによる数値型パラメーター attributeValue を通じて返されます。

```
Function LoadHDF5NumericAttribute(pathName, filePath, groupPath, objectName, objectType,
attributeName, attributeValue)
    String pathname                // シンボリックパス名 - filePath がフルパスの場合は "" でも可
    String filePath                // ファイル名、またはシンボリックパスに対する部分パス（一部）、あるいは
                                // はファイルへのフルパス
    String groupPath              // グループへのパス、例えば "/"、"/my_group"
    String objectName             // グループ名またはデータセット名
    Variable objectType           // 1=group, 2=dataset
    String attributeName          // 属性名
    Variable& attributeValue      // 出力 - pass-by-reference パラメーター

    attributeValue = NaN

    Variable result = 0

    // HDF5 ファイルを開く
    Variable fileID               // ここに HDF5 ファイル ID を格納
    HDF5OpenFile /P=$pathName /R /Z fileID as filePath
    if (V_flag != 0)
        Print "HDF5OpenFile failed"
        return -1
    endif

    Variable groupID              // ここに HDF5 グループ ID を格納
    HDF5OpenGroup /Z fileID, groupPath, groupID
    if (V_flag != 0)
        Print "HDF5OpenGroup failed"
        HDF5CloseFile fileID
        return -1
    endif

    HDF5LoadData /O /A=attributeName /TYPE=(objectType) /N=tempAttributeWave /Q /Z groupID,
objectName
    result = V_flag               // OK またはゼロではないエラーコードの場合は 0

    if (result == 0)
        Wave tempAttributeWave
        if (WaveType(tempAttributeWave) == 0)
            attributeValue = NaN    // 属性は数値ではなく文字列
            result = -1
        else
            attributeValue = tempAttributeWave[0]
        endif
        KillWaves/Z tempAttributeWave
    endif

    // HDF5 グループを閉じる
    HDF5CloseGroup groupID

    // HDF5 ファイルを閉じる
    HDF5CloseFile fileID

    return result
End
```

End

HDF5 文字列属性の読み込み

次の関数は、グループまたはデータセットの文字列属性を読み込む方法を示しています。

この関数の戻り値はエラーコードです。

属性の値は、参照渡しによる文字列パラメーター attributeValue を通じて返されます。

```
Function LoadHDF5StringAttribute(pathName, filePath, groupPath, objectName, objectType,
attributeName, attributeValue)
    String pathname                // シンボリックパス名 - filePath がフルパスの場合は "" でも可
    String filePath                // ファイル名、またはシンボリックパスに対する部分パス（一部）、あるいは
                                // はファイルへのフルパス
    String groupPath              // グループへのパス、例えば "/", "/metadata_group"
    String objectName              // グループ名またはデータセット名
    Variable objectType            // 1=group, 2=dataset
    String attributeName          // 属性名
    String& attributeValue        // 出力 - pass-by-reference パラメーター

    attributeValue = ""

    Variable result = 0

    // HDF5 ファイルを開く
    Variable fileID                // ここに HDF5 ファイル ID を格納
    HDF5OpenFile /P=$pathName /R /Z fileID as filePath
    if (V_flag != 0)
        Print "HDF5OpenFile failed"
        return -1
    endif

    Variable groupID              // ここに HDF5 グループ ID を格納
    HDF5OpenGroup /Z fileID, groupPath, groupID
    if (V_flag != 0)
        Print "HDF5OpenGroup failed"
        HDF5CloseFile fileID
        return -1
    endif

    HDF5LoadData /O /A=attributeName /TYPE=(objectType) /N=tempAttributeWave /Q /Z groupID,
objectName
    result = V_flag                // OK またはゼロではないエラーコードの場合は 0

    if (result == 0)
        Wave/T tempAttributeWave
        if (WaveType(tempAttributeWave) != 0)
            attributeValue = ""    // 属性は数値ではなく文字列
            result = -1
        else
            attributeValue = tempAttributeWave[0]
        endif
        KillWaves/Z tempAttributeWave
    endif

    // HDF5 グループを閉じる
    HDF5CloseGroup groupID

    // HDF5 ファイルを閉じる
    HDF5CloseFile fileID

    return result
End
```

HDF5 グループまたはデータセットのすべての属性の読み込み

この関数は、指定されたグループまたはデータセットのすべての属性を読み込む方法を示しています。
属性は、現在のデータフォルダー内のウェーブに読み込まれます。

```
Function LoadHDF5Attributes(pathName, filePath, groupPath, objectName, objectType, verbose)
    String pathname                // シンボリックパス名 - filePath がフルパスの場合は "" でも可
    String filePath                // ファイル名、またはシンボリックパスに対する部分パス（一部）、あるいは
                                // はファイルへのフルパス
    String groupPath               // グループへのパス、例えば "/", "/metadata_group"
    String objectName              // 属性を取得したいオブジェクトの名前、または groupPath で指定され
                                // たグループの場合は "."
    Variable objectType            // objectPath が参照するオブジェクトの型：1=group, 2=dataset
    Variable verbose               // Bit 0: エラーを出力; Bit 1: 警告を出力; Bit 2: ルーチン情報を
                                // 出力

    Variable printErrors = verbose & 1
    Variable printWarnings = verbose & 2
    Variable printRoutine = verbose & 4

    Variable result = 0           // 0 はエラーなしを意味する

    // HDF5 ファイルを開く
    Variable fileID               // ここに HDF5 ファイル ID を格納
    HDF5OpenFile /P=$pathName /R /Z fileID as filePath
    if (V_flag != 0)
        if (printErrors)
            Print "HDF5OpenFile failed"
        endif
        return -1
    endif

    Variable groupID              // ここに HDF5 グループ ID を格納
    HDF5OpenGroup /Z fileID, groupPath, groupID
    if (V_flag != 0)
        if (printErrors)
            Print "HDF5OpenGroup failed"
        endif
        HDF5CloseFile fileID
        return -1
    endif

    HDF5ListAttributes /TYPE=(objectType) groupID, objectName
    if (V_Flag != 0)
        if (printErrors)
            Print "HDF5ListAttributes failed"
        endif
        HDF5CloseGroup groupID
        HDF5CloseFile fileID
        return -1
    endif

    Variable numAttributes = ItemsInList(S_HDF5ListAttributes)
    Variable i
    for(i=0; i<numAttributes; i+=1)
        String attributeNameStr = StringFromList(i, S_HDF5ListAttributes)

        STRUCT HDF5DataInfo di
        InitHDF5DataInfo(di)           // 構造体を初期化
        HDF5AttributeInfo(groupID, objectName, objectType, attributeNameStr, 0, di)
        Variable doLoad = 0
        switch(di.datatype_class)
            case H5T_INTEGER:
```

```

        case H5T_FLOAT:
        case H5T_TIME:           // まだテストしていない
        case H5T_STRING:
        case H5T_BITFIELD:       // まだテストしていない
        case H5T_OPAQUE:         // まだテストしていない
        case H5T_REFERENCE:
        case H5T_ENUM:           // まだテストしていない
        case H5T_ARRAY:          // まだテストしていない
            doLoad = 1
            break
        case H5T_COMPOUND:       // HDF5LoadData は複合属性を読み込むことができない
            doLoad = 0
            break
    endswitch
    if (!doLoad)
        if (printWarnings)
            Printf "Not loading attribute %s - class %s not supported\r",
attributeNameStr, di.datatype_class_str
        endif
        continue
    endif

    HDF5LoadData /O /A=attributeNameStr /TYPE=(objectType) /Q /Z groupID, objectName
    if (V_flag != 0)
        if (printErrors)
            Print "HDF5LoadData failed"
        endif
        result = -1
        break
    endif
    if (printRoutine)
        Printf "Loaded attribute %d, name=%s\r", i, attributeNameStr
    endif
endfor

// HDF5 グループを閉じる
HDF5CloseGroup groupID

// HDF5 ファイルを閉じる
HDF5CloseFile fileID

return result
End

```

HDF5 データセットのサブセット

通常は必要ありませんが、「ハイパースラブ」を使って HDF5 データセットの一部を読み込むことは可能です。この機能を使うには、HDF5 のドキュメントで説明されているハイパースラブについて十分に理解している必要があります。

以下の例では、そのドキュメントを読み、内容を理解していることを前提としています。

HDF5 では、属性のサブセットを読み込むことはサポートされていません。

データセットの一部を読み込むには、HDF5LoadData コマンドの /SLAB フラグを使います。

/SLAB フラグは、パラメーターとして「スラブウェーブ」を受け取ります。

これは、正確に 4 列を持ち、読み込むデータセットの次元数以上の行数を持つ 2 次元ウェーブです。

スラブウェーブの 4 つの列は、HDF5 ライブラリの HDF5 H5Sselect_hyperslab ルーチンにおける、start、stride、count、block の各パラメーターに対応しています。

以下の例は、ハイパースラブの使用方法を示しています。
これらの例では、NCSA の提供の Igor Pro の HDF5 サンプルディレクトリに保存されているサンプルファイルを使っています。

「HDF5Samples」という名前のシンボリックパス（メニュー Misc→New Path）を作成し、i32le.h5 ファイルが含まれるフォルダー（Igor Pro Folder : Examples : Feature Demos : HDF5 Samples）をそのパス先として指定します。

最初の例では、「TestArray」という名前の、行数 6、列数 5 の 2D データセットを読み込みます。
まず、ハイパースラブを使わずにデータセット全体を読み込みます。

```
Variable fileID
HDF5OpenFile/P=HDF5Samples /R fileID as "i32le.h5"
HDF5LoadData /N=TestWave fileID, "TestArray"
Edit TestWave
```

次に、スラブウェーブを作成し、どの列にどのような情報が格納されているかを覚えやすくするために、その次元ラベルを設定します。

Igor Pro の起動時に自動的に読み込まれる HDF5 Utilities.ipf プロシージャファイル内のユーティリティルーチンを使います。

```
// HDF5 Utilities.ipf にある関数
HDF5MakeHyperslabWave("root:slab", 2)
Edit root:slab.ld
```

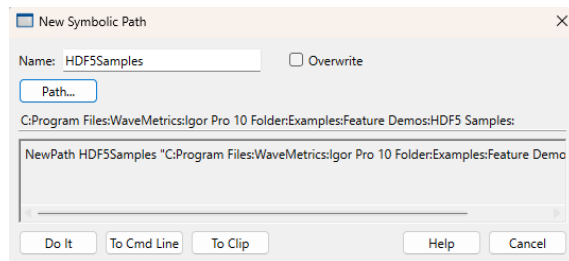
ここで、スラブウェーブの値を、先ほどと同じ結果、つまりデータセット全体を読み込むように設定し、その後、スラブを使ってデータを再度読み込みます。

```
slab[][%Start] = 0           // すべての次元について、0 から開始する
slab[][%Stride] = 1          // すべての次元でストライドを 1 にする
slab[][%Count] = 1           // すべての次元でカウントを 1 に設定する
slab[0][%Block] = 6          // 次元 0 から 6 までのブロックサイズを設定する
slab[1][%Block] = 5          // 次元 1~5 のブロックサイズを設定する
TestWave = -1                // 次のコマンドがそれを変更するのがわる
HDF5LoadData /N=TestWave /O /SLAB=slab fileID, "TestArray"
```

Row	slab.l	Dimension 0	slab[][%0] d	slab[][%1] d	slab[][%2] d	slab[][%3] d
	x y	Start	Stride	Count	Block	
0	Dimension 0	0	1	1	6	
1	Dimension 1	0	1	1	5	
2						

ここでは、ストライド、カウント、ブロックのパラメータを設定し、両方の次元から 1 つおきに要素を読み込むようにします。

```
slab[][%Start] = 0           // すべての次元について、0 から開始する
slab[][%Stride] = 2          // すべての次元でストライドを 2 にする
slab[0][%Count] = 3          // 次元 0 の 3 ブロックを読み込む
slab[1][%Count] = 2          // 次元 1 の 2 ブロックを読み込む
```



Row	TestWave[][%0]	TestWave[][%1]	TestWave[][%2]	TestWave[][%3]	TestWave[][%4]
0	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8
5	5	6	7	8	9

Row	slab.l	Dimension 0	slab[][%0] d	slab[][%1] d	slab[][%2] d	slab[][%3] d
	x y	Start	Stride	Count	Block	
0	Dimension 0	1	1	1	1	
1	Dimension 1	1	1	1	1	
2						

```
slab[0][%Block] = 1 // 次元 0 のブロックサイズを 1 に設定する
slab[1][%Block] = 1 // 次元 1 のブロックサイズを 1 に設定する
TestWave = -1 // 次のコマンドがそれを変更するのがわる
HDF5LoadData /N=TestWave /O /SLAB=slab fileID, "TestArray"
```

Row	slab.l	slab[[0].d	slab[[1].d	slab[[2].d	slab[[3].d
0	Dimension 0	2	3	1	
1	Dimension 1	2	2	1	
2	Dimension 2	2	2	1	

最後に、ファイルを閉じます。

```
HDF5CloseFile fileID
```

スラブウェーブの各行には、ファイル内のデータセットの対応する次元に関する一連のパラメーター（start、stride、count、block）が格納されています。

スラブウェーブの 0 行目にはデータセットの次元 0 に関するパラメーターが格納されており、以下同様です。

開始値は 0 以上でなければなりません。

その他の値はすべて 1 以上でなければなりません。

すべての値は 20 億未満でなければなりません。

HDF5LoadData は、ブロックサイズとして指定された値を、読み込まれるデータセットの対応するサイズに合わせて切り詰めます。

HDF5LoadData を使うには、スラブウェーブが正確に 4 列である必要があります。

自動的に読み込まれる HDF5 Utilities.ipf プロシージャファイルに含まれる HDF5MakeHyperslabWave 関数は、Start、Stride、Count、Block という列次元ラベルを持つ 4 列のウェーブを作成します。

HDF5LoadData では、列次元ラベルは必須ではありません。

Igor Pro 9.0 以降では、path パラメーターに "" を指定した場合、HDF5MakeHyperslabWave はフリーウェーブを返します。

また、ウェーブがフリーであるかどうかにかかわらず、関数の結果としてウェーブ参照を返します。

スラブウェーブには、データセットの次元数以上の行数が必要です。

それ以上の行は無視されます。

HDF5LoadData は、読み込まれたデータをすべて格納するのにちょうど十分な大きさのウェーブを作成します。したがって、最初の例では 6×5 のウェーブが作成され、最後の例では 3×2 のウェーブが作成されました。

Igor Pro と HDF5 の比較

このセクションでは、Igor Pro と HDF5 のデータ構成の違い、そして Igor Pro がそれらをどのように整合させるかについて説明します。

大文字と小文字の区別

Igor Pro は大文字と小文字を区別しませんが、HDF5 は大文字と小文字を区別します。

したがって、例えば HDF5 データセットの名前を指定する時は、大文字と小文字に注意が必要です。

"/Dataset1" と "/dataset1" は同じものではありません。

Igor Pro で、デフォルトのウェーブ名を使って “/Dataset1” と “/dataset1” を読み込むと、2 回目の読み込みによって 1 回目で作成されたウェーブが上書きされてしまいます。

HDF5 のオブジェクト名と Igor Pro のオブジェクト名の比較

HDF5 のオブジェクト名にはスラッシュ (/) を使うことはできません。

スラッシュを含む名前では Igor Pro のウェーブフォルダーまたはデータフォルダーを作成し、そのオブジェクトを HDF5 ファイルに保存しようとする、エラーが発生します。

ドット “.” で始まるオブジェクト名でも、エラーが発生する可能性があります。

本稿執筆時点において、『HDF5 ユーザーガイド』の 4.2.3 節「HDF5 のパス名」には次のように記載されています：

コンポーネントのリンク名は、スラッシュやドットを含まない任意の ASCII 文字列にすることができます (“/” と “.” は、前述の通り予約済みです)。ただし、句読点や非表示文字の使用は、他のソフトウェアで問題を引き起こす可能性があるため、避けることを推奨します。

HDF5 のデータ型と Igor Pro のデータ型の比較

HDF5 は、Igor Pro がサポートしていない多くのデータ型をサポートしています。

このようなデータ型を読み込む時、可能であれば、サポートしているデータ型に変換しようとします。

その過程で、精度が失われる可能性があります。

デフォルトでは、下位互換性を確保するため、HDF5 の 64 ビット整数データを倍精度浮動小数点数として保存、読み込みします。

この変換により、精度が失われる場合があります。

64 ビット整数データを 64 ビット整数として保存、読み込むには、HDF5SaveData、HDF5SaveGroup、HDF5LoadData、HDF5LoadGroup で /OPTS=1 を指定してください。

Igor Pro ではほとんどの演算が倍精度浮動小数点で行われるため、64 ビット整数を読み込む時は、53 ビット以内に収まる場合は倍精度浮動小数点として、53 ビットを超える可能性がある場合は 64 ビット整数として読み込むことを推奨します。

Igor Pro は、128 ビット浮動小数点データ (long double) をサポートしていないため、HDF5 形式の 128 ビット浮動小数点データを倍精度浮動小数点データとして読み込みます。

この変換により、精度が低下します。

HDF5 の最大次元と Igor Pro の最大次元の比較

HDF5 ライブラリは、最大 32 次元のデータをサポートしています。

Igor Pro は 4 次元のみをサポートしています。

次元数が 4 を超える HDF5 データを読み込む場合、HDF5LoadData コマンドは、すべてのデータを格納するのに十分なチャンクを持つ 4 次元のウェーブを作成します。

(「チャンク」とは、Igor Pro において「行」、「列」、「レイヤー」に続く 4 番目の次元の名前です。)

例えば、次元が $7 \times 6 \times 5 \times 4 \times 3 \times 2$ のデータセットを読み込むと、HDF5LoadData は次元が $7 \times 6 \times 5 \times 24$ のウェーブを作成します。

このウェーブには 24 個のチャンクが含まれます。

24 という数は、HDF5 データセット内のチャンク数 (この場合は 4) に、各高次元のサイズ (この場合は 3 と 2) を乗じたものになります。

行優先と列優先のデータ順序

HDF5 と Igor Pro では、多次元データのメモリへの格納方法が異なります。

ほとんどの場合、この違いは問題になりません。

ごくまれにこの違いが重要になる場合について、以下に説明します。

HDF5 は、多次元データを「行優先」の順序で格納します。

これは、メモリ上である要素から次の要素へと移動するにつれて、最も高い次元に関連するインデックスが最も速く変化することを意味します。

例えば、5 行 4 列の 2 次元データセットがある場合、メモリ内を順次移動していくと、列インデックスが最も速く変化し、行インデックスが最も遅く変化します。

Igor Pro はデータを「列優先」の順序で格納します。

これは、メモリ内のある要素から次の要素へと移動するにつれて、最も低い次元に関連付けられたインデックスが最も速く変化することを意味します。

例えば、5 行 4 列の 2 次元データセットがある場合、メモリ内を順次移動していくと、行インデックスが最も速く変化し、列インデックスが最も遅く変化します。

この違いを回避するために、HDF5LoadData または HDF5LoadGroup コマンドで 2 次元以上のデータが最初に読み込まれた後、メモリ内でデータの配置が再編成されます。

その結果、テーブルでデータを表示した時、メモリ上のレイアウトは異なりますが、HDF Group の HDFView プログラムで表示した場合とまったく同じように表示されます。

HDF5SaveData または HDF5SaveGroup を使って Igor Pro データを保存する場合も、同様の処理が行われます。

HDF5 画像と Igor Pro 画像の比較

HDF5 画像とパレット仕様は、HDF5 ファイルに画像とそれに関連する情報（パレット、色モデルなど）を格納するための詳細なガイドラインを定めています。

しかし、多くの HDF5 ユーザーはこの仕様に従わず、単に画像データを通常の 2D データセットとして書き込んでいます。

これら 2 つの画像保存方法を区別するために、ここでは、仕様に従って書き込まれた画像を「正式な画像 (formal image)」と呼び、ユーザーが画像として認識している通常の 2 次元データセットを「通常の画像 (regular image)」と呼ぶことにします。

Igor Pro の画像プロットでは、ウェーブの列データが水平方向にプロットされますが、HDFView やその他のほとんどのプログラムでは、行データが水平方向にプロットされます。

したがって、特別な処理を行わない限り、通常の画像は、Igor Pro では他のほとんどのプログラムに比べて回転して表示されてしまいます。

HDF5LoadImage と HDF5SaveImage コマンドは、正式な画像の読み込みと保存を処理します。

これらのコマンドでは、画像の向きが異なる場合の補正が自動的に行われます。

通常の画像を扱う場合は、HDF5LoadData と HDF5SaveData コマンド、あるいは HDF5LoadGroup と HDF5SaveGroup を使います。

これらのコマンドには /TRAN フラグがあり、これを指定すると 2D データが転置されます。

HDF5LoadData で /TRAN を使うと、Igor Pro や HDFView などのプログラムで表示された画像の向きは同じですが、テーブルで表示した場合は転置されたように見えます。

/TRAN フラグは、2D またはそれ以上の次元のデータで機能します。

高次元データ (3D または 4D) で使う場合、データの各レイヤーは個別の画像として扱われ、転置されます。

つまり、/TRAN は高次元データを画像の積み重ねとして扱います。

Igor Pro データの保存と再読み込み

HDF5SaveData と HDF5SaveGroup コマンドを使うと、Igor Pro のウェーブ、数値変数、文字列変数を HDF5 ファイルに保存できます。

これらの Igor Pro オブジェクトはすべて、HDF5 データセットとして書き込まれます。

ウェーブから保存されたデータセットには、デフォルトで、ウェーブのデータ型、ウェーブのスケーリング、ウェーブのメモなどのウェーブの特性を格納する属性が付与されます。

これらの属性には、IgorWaveType や IgorWaveScaling といった名前が付けられています。

これにより、後で HDF5 ファイルから Igor Pro へ読み戻す時に、HDF5LoadData と HDF5LoadGroup を使ってウェーブを完全に再現することが可能になります。

HDF5SaveData または HDF5SaveGroup を呼び出す時に /IGOR=0 フラグを使うことで、これらの属性の作成を抑制することができます。

ウェーブテキストは、常に UTF-8 テキストエンコーディングを使って記述されます。

詳細は、「HDF5 ウェーブのテキストエンコーディング」のセクションを参照してください。

HDF5 パックされたエクスペリメントデータを読み込む時、ウェーブ参照ウェーブとデータフォルダー参照ウェーブはそのままの形式で読み込まれますが、HDF5LoadData と HDF5LoadGroup では、これらのウェーブが倍精度数値として読み込まれます。

その理由は、これらのウェーブが正しいウェーブまたはデータフォルダーを指すように復元できるのは、エクスペリメントデータ全体を読み込んだ場合に限られるためです。

HDF5SaveGroup によって Igor Pro 変数から保存されたデータセットには、「IGORVariable」属性が付与されます。

これにより、ファイルを再読み込みする時、HDF5LoadData と HDF5LoadGroup がこれらのデータセットを Igor Pro 変数を表すものとして認識できるようになります。

この属性がない場合、これらのコマンドではすべてのデータセットがウェーブとして読み込まれます。

IGORVariable 属性の値は、Igor Pro 変数のデータ型コードです。

この値は、以下のいずれかになります。

- 0: Igor Pro 文字列変数
- 4: Igor Pro 実数変数
- 5: Igor Pro 複素数変数

「HDF5 文字列変数テキストエンコーディング」のセクションも参照してください。

複素数ウェーブの取り扱い

Igor Pro は複素数ウェーブをサポートしていますが、HDF5 は複素データセットをサポートしていません。

そのため、複素数ウェーブを保存する時、HDF5SaveData はそのウェーブの行数が2倍になったかのようにデータを書き込みます。

例えば、HDF5SaveData は、以下のウェーブについて、HDF5 ファイルに同じデータを書き込みます。

```
Make wave0 = {1,-1,2,-2,3,-3} // 6 スカラーポイント
Make/C cwave0 = {cmplx(1,-1),cmplx(2,-2),cmplx(3,-3)} // 3 複素数ポイント
```

HDF5SaveData によって書き込まれた HDF5 ファイルを読み込む時、HDF5SaveData が複素数ウェーブの場合にデータセットに付加する IgorComplexWave 属性の有無を確認することで、元のウェーブが複素数ウェーブであったかどうかを判断できます。

適切な /IGOR フラグを使えば、HDF5LoadData と HDF5LoadGroup がこれを自動的に行います。

Igor Pro 参照ウェーブの処理

Igor Pro は、ウェーブ参照とデータフォルダー参照ウェーブをサポートしています。

ウェーブ参照の各要素は、別のウェーブへの参照です。

データフォルダー参照ウェーブの各要素は、データフォルダーへの参照です。

エクスペリメントを HDF5 パケット形式で保存する場合、Igor Pro ウェーブ参照を正しく書き出しますが、HDF5SaveData と HDF5SaveGroup は Igor Pro ウェーブ参照の保存に対応していません。

ウェーブ参照を保存しようとする、HDF5SaveData コマンドはエラーを返します。

参照ウェーブを保存するよう要求された場合の HDF5SaveGroup コマンドの挙動は、/CONT フラグの設定によって異なります。

デフォルト (/CONT=1) では、ウェーブを保存できないことを示すメッセージを履歴に出力した後、データフォルダー内の残りのオブジェクトの保存を続行します。

/CONT=0 を指定した場合、HDF5SaveGroup は参照ウェーブの保存を要求されるとエラーを返します。

HDF5 マルチタスキング

Igor Pro のプリエンティブスレッドから、HDF5 のコマンドや関数を呼び出すことができます。

HDF5 ライブラリでは、一度に1つの HDF5 ファイルにしかアクセスできません。

このため、特定の Igor Pro インスタンス内で複数の HDF5 ファイルを同時に読み込むことはできませんが、Igor Pro のメインスレッドで他の処理を行っている間に、プリエンティブスレッドで HDF5 ファイルを読み込むことは可能です。

HDF5 ファイルへのアクセスを試みるスレッドを複数作成した場合、そのうちの1つのスレッドが HDF5 ライブラリへのアクセス権を取得します。

HDF5 ファイルにアクセスしようとする他のスレッドは、最初のスレッドが処理を完了するまで待機し、解放された時点で別のスレッドが HDF5 ライブラリへのアクセス権を取得します。

Igor Pro の HDF5 処理能力

Igor Pro は、HDF5 の全機能のうち、一部の機能のみをサポートしています。

以下は、Igor Pro がサポートしている HDF5 の機能の一部です。

- ・ すべての原子型データの読み込み
- ・ 文字列の読み込み
- ・ 配列データ型の読み込み
- ・ 基本型が整数型または浮動小数点型である可変長データセットの読み込み
- ・ 化合物データセット (C 言語風の構造を持つデータセット) の読み込み。これには、配列である要素を含む複素数データセットも含まれる
- ・ ハイパースラブを使ってのデータセットの一部の読み込み
- ・ オブジェクト参照の読み込みと保存
- ・ データセットの領域参照の読み込み

Igor Pro の HDF5 の制限

以下は、Igor Pro がサポートしていない HDF5 の機能の一部です。

- ・ VLEN データセット（不規則配列）の作成または追加
- ・ 深くネストされたデータ型の読み込み。「HDF5 のネストされたデータ型」のセクションを参照してください
- ・ データセットの領域参照の保存

HDF5 の高度なデータ型

このセクションは、主に HDF5 の上級ユーザー向けのものです。

HDF5 の可変長データ

ほとんどの HDF5 ファイルは可変長データ型を使っていないため、ほとんどのユーザーはこの情報を知る必要はありません。

可変長データは、各要素が「ベース」データ型と呼ばれる別のデータ型の要素からなる 1D セットである配列で構成されます。

各セットに含まれるベースデータ型の要素数は、それぞれ異なる場合があります。

例えば、ベース型が H5T_STD_I32LE である 5 要素の可変長データセットには、32 ビットのリトルエンディアン整数からなる 1D セットが 5 つ含まれており、各セットの長さはそれぞれ独立しています。

HDF5LoadData は、基本型が整数型または浮動小数点型の可変長データセットのみを読み込みます。

各要素のデータは、個別のウェーブとして読み込まれます。

ほとんどの種類のデータを読み込む場合、HDF5LoadData は 1 つのウェーブのみを作成します。

可変長のデータセットや属性を読み込む場合、読み込まれた要素ごとに 1 つのウェーブが作成されます。

複数の要素を読み込む場合、ウェーブに提案された名前（読み込まれるデータセットまたは属性の名前、あるいは /N=name で指定された名前）は、ベース名として扱われます。

例えば、データセットまたは属性に 3 つの要素があり、名前が test で、かつ /O フラグが使われた場合、test0、test1、test2 という名前のウェーブが作成されます。

/O フラグが使われない場合、test<n> という形式の名前が作成されます。

ここで、<n> はウェーブ名が一意になるように選ばれた数値です。

HDF5LoadData コマンドでは、可変長データセットの一部を読み込むことができます。

これを行うには、HDF5LoadData の /SLAB フラグを使ってスラブウェーブを指定します。

前の段落の例で、要素を 1 つだけ読み込んだ場合、その名前は test0 ではなく test になります。

要素を 2 つ読み込んだ場合、どの 2 つの要素を読み込んだかに関係なく、それらは test0 と test1 という名前になります。

次の関数は、可変長データセットの要素を 1 つ読み込む方法を示しています。

「Data」という名前のシンボリックパスと、「Vlen.h5」という名前のファイルが存在し、そのファイルには少なくとも 2 つの要素を含む「TestVlen」という名前の 1D 可変長データセットが含まれているものと仮定します。

この関数は、2 番目の可変長要素を「TestWave」という名前のウェーブに読み込みます。

```
Function DemoVlenLoad()  
    Variable fileID  
    HDF5OpenFile /P=Data /R fileID as "Vlen.h5"
```

```

HDF5MakeHyperslabWave("root:slab", 1)           // HDF5 Utilities.ipf にある
Wave slab = root:slab
slab[0][%Start] = 1                               // vlen の2番目の要素から開始する
slab[0][%Stride] = 1                             // 1 のストライドを使う
slab[0][%Count] = 1                              // 1 ブロックを読み込む
slab[0][%Block] = 1                              // ブロックサイズを 1 に設定
HDF5LoadData /N=TestWave /O /SLAB=slab fileID, "TestVlen"

HDF5CloseFile fileID
End

```

HDF5 配列データ

ほとんどの HDF5 ファイルでは配列データ型が使われていないため、ほとんどのユーザーはこの情報を知る必要はありません。

HDF5 データセット（または属性）は、1 つ以上の次元で構成された要素から成ります。

各要素は、unsigned short や倍精度 float などの原子データ型である場合もあれば、構造体や配列などの複合データ型である場合もあります。

したがって、HDF5 データセットは、unsigned short の配列、倍精度 float の配列、構造体の配列、あるいは配列の配列となる可能性があります。

このセクションでは、最後のタイプである「配列の配列」の読み込みについて説明します。

この場合、データセットのクラスは H5T_ARRAY です。

データセットの型は「符号付き long の 5×4 配列」のようなものであり、「符号付き long」は配列データ型の「基本型」とであると言われています。

データセット自体が 10 行の 1D である場合、10 行のデータセットとなり、その各要素は 1 つの符号付き long の 5×4 配列で構成されます。

Igor Pro では、これは 5 行、4 列、10 レイヤーからなる 3D ウェーブとして扱うことができます。

しかし、Igor Pro は 4 次元までしかサポートしていないのに対し、HDF5 は 32 次元までサポートしているため、Igor Pro では次元が不足してしまう可能性があります。

そのため、H5T_ARRAY 型のデータセットを読み込む時、HDF5LoadData は、基になるデータセットの次元数ではなく、配列型の次元数を持つウェーブを作成します。

ただし、すべてのデータを格納するためのスペースを確保するために、最高次元は増加されます。

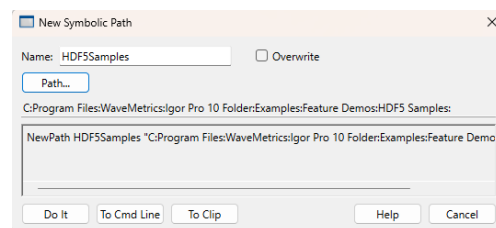
これにより、次元が不足する可能性が低くなります。

HDF5LoadData は、基本型が compound または array である配列データを読み込むことはできません（詳細は、「HDF5 のネストされたデータ型」のセクションを参照してください）。

以下の例は、配列データ型を読み込む方法を示しています。

この例では、Igor Pro の HDF5 Samples ディレクトリに保存されている NCSA のサンプルファイルを使っています。

「HDF5Samples」という名前の Igor シンボリックパス（メニュー Misc→New Path）を作成し、SDS_array_type.h5 ファイルが含まれるフォルダー（Igor Pro Folder:Examples:Feature Demos:HDF5 Samples）を指定します。



サンプルファイルには、10 行からなる 1D データセットが含まれており、各要素は 5 行×4 列の行列で、32 ビットの符号付きビッグエンディアン整数（別名：ビッグエンディアン符号付き long 型、または I32BE）で構成されています。

データセットを読み込むと、5 行 40 列のデータが得られます。

```
Variable fileID
HDF5OpenFile/P=HDF5Samples /R fileID as "SDS_array_type.h5"
HDF5LoadData fileID, "IntArray"
HDF5CloseFile fileID
Edit IntArray
```

最初の4列には、データセットの0行目からの 5×4 の行列が含まれています。

次の4列には、データセットの1行目からの 5×4 の行列が含まれています。

以下同様です。

Row	IntArray[0]	IntArray[1]	IntArray[2]	IntArray[3]	IntArray[4]
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

この場合、Igor Pro には十分な次元数があるため、必要に応じて、5×4 の行列からなる 10 層で構成される 3D ウェーブとして再構成することができます。

その時は、次のコマンドを使います。

```
Redimension /N=(5,4,10) /E=1 IntArray
```

/E=1 フラグを指定すると、Redimension は保存されたデータを実際に変更することなく、ウェーブの次元を変更するよう指示されます。

5x4x10 のウェーブのメモリ上のレイアウトは、5x40 のウェーブのレイアウトと同じです。

Redimension は、ウェーブの表示方法を、5 行×40 列から 5 行×4 列×10 層に変更するだけです。

配列データ型を持つデータセットを読み込むことはできますが、Igor Pro では、配列データ型を持つデータ型を書き出す方法はありません。

HDF5 参照データの読み込み

ほとんどの HDF5 ファイルは参照データ型を使わないため、ほとんどのユーザーはこの情報を知る必要はありません。

HDF5 のデータセットや属性には、他のデータセット、グループ、名前付きデータ型への参照が含まれることがあります。

参照には、「オブジェクト参照」と「データセット領域参照」の 2 種類があります。

HDF5LoadData は、これら 2 種類の参照を両方ともテキストウェーブに読み込みます。

オブジェクト参照データセットの要素は、同じファイル内または別のファイルにあるデータセットを参照することができます。

データセット領域参照データセットの要素は、同じファイル内にあるデータセットのみを参照することができます。

HDF5 オブジェクト参照データの読み込み

データセットへの各オブジェクト参照について、HDF5LoadData は「D:」に HDF5 ファイル内のデータセットのフルパスを追加した形式（例えば：「D:/GroupA/DatasetB」）を返します。

グループおよび名前付きデータ型への参照については、HDF5LoadData はそれぞれ「G:」および「T:」を返し、その後オブジェクトへのパスが続きます。

<list of blocks> は、以下の形式になっています：

```
<coordinates>-<coordinates>/<coordinates>/...
```

ここで、<coordinates> は、カンマ区切りの座標のリストです。

<coordinates> のペアの間にはダッシュが表示されます。

ペアの最初の座標はブロックの開始座標を指定し、2番目の座標はブロックの終了座標を指定します。

3つのブロックが選択された 2D データセットの場合、次のような感じになります：

```
3,4-6,7/11,13-15,17/21,30-37,38
```

これには、以下の3つのブロックが指定されています：

```
row 3, column 4 to row 6, column 7  
row 11, column 13 to row 15, column 17  
row 21, column 30 to row 37, column 38
```

以下は、赤で示された追加情報を含む、完全なポイントデータセットの地域情報文字列の例です：

```
D:/Group1/Dataset3.REGIONTYPE=POINT;NUMDIMS=2;NUMELEMENTS=3;COORDS=3,4/11,13/21,30;
```

以下は、赤色で示された追加情報を含む、完全なブロックデータセットのリージョン情報文字列の例です：

```
D:/Group1/Dataset4.REGIONTYPE=BLOCK;NUMDIMS=2;NUMELEMENTS=3;COORDS=3,4-6,7/11,13-15/17/21,30-37/38;
```

2行のデータセット領域参照データセットに対して HDF5LoadData を呼び出した後、ウェーブが戻ってきます。このデータセットには、先ほど示した例のようなテキストが2行含まれています。

データセット領域参照データセットの各行は、1つのデータセットと、そのデータセット内のポイントまたはブロックのセットを指しています。

「HDF5 Dataset Region Demo.pxp」のエクスペリメントでは、データセット領域の参照を扱うための例やユーティリティなど、さらなる情報が提供されています。

HDF5 オブジェクト参照データの保存

ほとんどの HDF5 ファイルは参照データ型を使っていないため、ほとんどのユーザーはこの情報を知る必要はありません。

HDF5 のデータセットや属性には、他のデータセット、グループ、名前付きデータ型への参照を含めることができます。

これらは「オブジェクト参照」と呼ばれます。

/REF フラグを使うことで、HDF5SaveData にテキストウェーブをオブジェクト参照として保存するよう指示することができます。

/REF フラグを使うには、Igor Pro 8.03 以降が必要です。

参照用として保存するテキストは、参照対象のオブジェクトの種類を識別する接頭辞文字と、その後続くオブジェクトへのフルパスまたは部分パスで構成されている必要があります。

グループ、データ型、データセットには、それぞれ「G:」、「D:」、または「T:」を使います。

例えば：

```
Function DemoSaveReferences(pathName, fileName)  
    String pathname          // シンボリックパス名  
    String filename          // HDF5 ファイル名  
  
    Variable fileID  
    HDF5CreateFile/P=$pathname /O fileID as fileName
```

```

// 参照を使って、ターゲットとするグループを作成
Variable groupID
HDF5CreateGroup fileID, "GroupA", groupID

// 参照を使って、ターゲットとするデータセットを作成
Make/O/T textWave0 = {"One", "Two", "Three"}
HDF5SaveData /O /REF=(0) /IGOR=0 textWave0, groupID

// フルパスを指定して、参照データセットをルートに書き込む
Make/O/T refWaveFull = {"G:/GroupA", "D:/GroupA/textWave0"}
HDF5SaveData /O /REF=(1) /IGOR=0 refWaveFull, fileID

HDF5CloseGroup groupID
HDF5CloseFile fileID

End

```

部分パスは、HDF5SaveData に渡されたファイル ID またはグループ ID を基準としています。

HDF5 データセットの領域参照データの保存

Igor Pro では、データセットの領域参照を保存する機能はサポートされていません。

HDF5 Enum データの読み込み

ほとんどの HDF5 ファイルでは列挙型データ型が使われていないため、ほとんどのユーザーはこの情報を知る必要はありません。

列挙型データの値は、HDF5 ファイルに整数として格納されます。

列挙型データセットまたは属性に関連付けられたデータ型は、整数値から名前への対応関係を定義します。

HDF5LoadData は、列挙型データの各要素について、整数値または名前のいずれかを読み込むことができます。

これは、/ENUM=enumMode フラグを使ってコントロールします。

/ENUM が省略された場合、または enumMode が 0 の場合、HDF5LoadData はデータ型が signed long の数値ウェーブを作成し、整数の列挙値をそこに読み込みます。

これは、8 ビット、16 ビット、32 ビットの整数列挙データで機能します。

64 ビット整数に基づく列挙データの読み込みはサポートされていません。

enumMode が 1 の場合、HDF5LoadData はテキストウェーブを作成し、各列挙型値に関連付けられた名前をそこに読み込みます。

これは、整数の列挙型値を読み込むよりも処理に時間がかかりますが、処理速度の低下が顕著になるのは、非常に大規模な列挙型データセット、あるいは非常に多くの列挙型データセットを読み込む場合に限られます。

HDF5 Enum データの保存

ほとんどの HDF5 ファイルでは enum データ型が使われていないため、ほとんどのユーザーはこの情報を知る必要はありません。

Igor Pro 9.01 では、HDF5 の列挙型データ型を使って Igor Pro の整数データを保存する機能が追加されました。

この操作は、HDF5SaveData コマンドの /ENUM フラグを使って行います。

HDF5 Browser と HDF5SaveGroup では、列挙型データの保存はサポートされていません。

例を挙げます：

```

Function DemoSaveWaveAsEnum()
    Variable fileID
    HDF5CreateFile/P=IgorUserFiles /O fileID as "DemoSaveWaveAsEnum.h5"
    Make/FREE/Y=(0x48) enumWave = {1,2,3}           // 0x48 は符号なしバイトデータを示す
    String enumList = "One=1;Two=2;Three=3;"         // HDF5 enum datatype データ型を指定
    HDF5SaveData /O /ENUM=enumList /IGOR=0 enumWave, fileID, "enumDataset"
    HDF5CloseFile fileID
End

```

この例では、基本型が unsigned byte (/Y=(0x48)) の「enumDataset」という名前のデータセットを記述しています。

この列挙型データ型は、3つの整数値である 1、2、3 に関連付けられた、One、Two、Three という3つの名前を定義しています。

enumList の構文は非常に厳格です。

各列挙型メンバーについて、名前、等号、整数、セミコロンの順で指定する必要があります。

空白やその他の文字は一切使用できません。

保存されるウェーブは、整数型（8 ビット、16 ビット、32 ビット、または 64 ビットの符号付きまたは符号なし整数）でなければなりません。

HDF5LoadData は 64 ビット整数の列挙型データを読み込むことができないため、データを Igor Pro に読み戻す必要がある場合は、64 ビットを使わないでください。

HDF5 の不透明データ

ほとんどの HDF5 ファイルでは不透明なデータ型が使われていないため、ほとんどのユーザーはこの情報を知る必要はありません。

不透明データは、指定された長さのバイト列として扱われる要素で構成されます。

HDF5LoadData は、不透明データを符号なしバイトのウェーブに読み込みます。

不透明データの要素が n バイトの場合、各要素はウェーブ内で n 行の連続した行を占めます。

HDF5 ビットフィールドデータ

ほとんどの HDF5 ファイルではビットフィールドのデータ型は使われていないため、ほとんどのユーザーはこの情報を知る必要はありません。

ビットフィールドデータは、個別の値として扱われるビットの列からなる要素で構成されます。

HDF5LoadData は、任意の長さのビットフィールドデータを、要素ごとに必要なバイト数だけ割り当てられた符号なしバイトウェーブに読み込みます。

例えば、ビットフィールドが 2 バイトの場合、各要素に対して符号なしバイトウェーブの 2 行が使われます。

データは、ファイルに保存されているバイト順で読み込まれます。

データをバイトのストリームと捉える場合には、この方法が適切です。

データを short（2 バイト）または long（4 バイト）とみなす場合は、Redimension コマンドを使用できます。

例えば、1 行あたり 2 バイトのビットフィールドが 5 つ、列が 3 つの 2D データセットを読み込んだ場合、結果は 10×3 の符号なしバイトのウェーブとなります。

次のようにして、これを 5×3 の short のウェーブに変更できます。

```
Redimension /N=(5,3) /W /E=1 bitfieldWave
```

バイト順を変更する必要がある場合は、/E=1 の代わりに /E=2 を使ってください。

HDF5 のネストされたデータ型

ほとんどの HDF5 ファイルではネストされたデータ型が使われていないため、ほとんどのユーザーは以下の情報を知る必要はありません。

HDF5 は、整数（HDF5 ライブラリではクラス H5T_INTEGER と呼ばれる）や浮動小数点数（クラス H5T_FLOAT）などの「原子型」データクラス、 また、構造体（クラス H5T_COMPOUND）、配列（クラス H5T_ARRAY）、可変長（クラス H5T_VLEN）型などの「複合」データクラスもサポートしています。

データ型が「原子型」であるデータセットでは、そのデータセットの各要素は単一の値です。
データ型が「複合」であるデータセットでは、各要素は、1 つ以上の他のデータ型の要素からなる集合です。

例えば、型 H5T_STD_I16BE の 5×3 のデータ配列を考えてみます。
H5T_STD_I16BE は、クラス H5T_INTEGER の組み込み原子型データ型です。
この配列の各要素は、16 ビットのビッグエンディアン整数値です。
このデータセットは原子型データ型を持っています。

クラス H5T_ARRAY の 4 行からなるデータセットについて考えてみます。
このデータセットの各要素は、型 H5T_STD_I16BE のデータからなる 5×3 の配列です。
このデータセットは複合データ型を持っています。
データセットの 4 つの要素はそれぞれ配列であり、この場合は 16 ビットのビッグエンディアン整数値の配列です。
H5T_STD_I16BE は、これら 4 つの配列それぞれの「基本データ型」です。

クラス H5T_COMPOUND の 4 行からなるデータセットについて考えてみます。
このデータセットには、整数型のメンバーが 1 つ、浮動小数点型のメンバーが 1 つ、文字列型のメンバーが 1 つ含まれています。
このデータセットは複合データ型を持っています。
このデータセットの 4 つの要素はそれぞれ、3 つの異なるデータ型のメンバーを持つ構造体です。

HDF5 では、データ型を任意の深さまでネストさせることができます。
例えば、各要素が H5T_COMPOUND データセットであり、そのメンバーが H5T_STD_I16BE、H5T_STD_I32BE、その他のデータ型の H5T_ARRAY データセットであるような H5T_ARRAY データセットを作成することができます。

Igor Pro は、無限にネストされたデータセットには対応していません。
対応しているのは以下のもののみです。

- ・ ほぼあらゆる種類の原子型データセット
- ・ 基本型が整数、浮動小数点数、文字列、ビットフィールド、不透明型、または列挙型の配列データセット
- ・ メンバーの基本型が整数、浮動小数点数、文字列、ビットフィールド、不透明型、列挙型、または参照である複合データセット
- ・ 配列の基本型が整数、浮動小数点数、文字列、ビットフィールド、不透明型、列挙型、または参照である配列の要素を含む複合データセット
- ・ 基本型が整数型または浮動小数点型の可変長データセット

HDF5 複合データ

これは、ほとんどのユーザーには必要のない高度な機能です。

HDF5 の複合データセットでは、データセットの各要素は、C 言語の構造体のインスタンスのように、名前付きフィールドのセットで構成されています。
複合データセットの読み込みは、その構造が任意に複雑になり得るため、問題が生じやすいものです。

複合データセットには、異なるデータ型の集合、異なるデータ型の配列、およびサブ複合構造が含まれる場合があります。

HDF5LoadData は、複合データセットから 1 つのメンバーを 1 つのウェーブに読み込むことも、複合データセットのすべてのメンバを一連のウェーブに読み込むこともできます。

ただし、メンバーが複雑すぎる場合、HDF5LoadData はそのメンバーを読み込むことができず、エラーを返します。

概して、「複雑すぎる」とは、そのメンバ自体が複合体（サブ構造）であることを意味します。

/COMP フラグを使い、mode パラメーターを 1 に設定し、読み込むメンバー名を指定することで、HDF5LoadData コマンドに対し、複合データセットの各要素から 1 つのメンバーを読み込むよう指示します。メンバーは、原子型データ型または配列データ型である必要があり、別の複合データ型であってはなりません（詳細については、「HDF5 のネストされたデータ型」のセクションを参照してください）。

HDF5LoadData は、HDF5 データセットのデータ型と互換性のあるデータ型を持つ Igor Pro ウェーブを作成します。

ウェーブの名前は、読み込まれるデータセットまたは属性の名前、あるいは /N フラグで指定された名前に基づいて決定されます。

/COMP フラグを省略するか、/COMP={0,「」} を指定することで、HDF5LoadData にすべてのメンバーを個別のウェーブに読み込むよう指示します。

作成されるウェーブの名前は、ベース名と、必要に応じて整えられたメンバー名が連結されたもので構成されます。ベース名は、読み込まれるデータセットまたは属性の名前、あるいは /N フラグで指定された名前に基づいています。

複合データ型を含む HDF5 データセットを読み込むことは可能ですが、複合データ型を持つデータ型を書き出す方法は提供されていません。

HDF5 圧縮

Igor Pro は、データセットを HDF5 ファイルに書き込む時に圧縮機能を利用できます。

圧縮されたデータセットは通常、ディスク上の占有容量は少なくなります。書き込みや読み取りに時間がかかるようになります。

圧縮によって節約できるディスク容量は、データのサイズや性質によって異なります。

データにノイズが極めて多い場合、圧縮によって節約できるディスク容量はごくわずかになるか、あるいは必要なディスク容量が増加することさえあります。

データがほぼすべてゼロ、あるいはその他の定数値で構成されている場合、圧縮によってディスク容量の 99 パーセントを節約できる可能性があります。

一般的な実世界のデータの場合、圧縮によってディスク容量の 10~50 パーセントを節約できる可能性があります。

圧縮されたデータセットの保存にかかる時間は、通常、非圧縮のデータセットを保存するのにかかる時間の 2 倍から 10 倍程度です。

圧縮されたデータセットの読み込みにかかる時間は、通常、非圧縮のデータセットを読み込むのにかかる時間の 2 倍程度です。

これらの所要時間は、データの内容によって大きく異なる場合があります。

以下の場合には、圧縮処置を行うことがあります。

- ・ HDF5 パックされたエクスペリメントファイルを保存する
- ・ HDF5 Browser を使ってデータセットまたはグループを保存する
- ・ HDF5SaveData コマンドを呼び出す

- ・ HDF5SaveGroup コマンドを呼び出す

圧縮を有効にするには、HDF5 ライブラリに対して特定のパラメーターを指定する必要があります。

- ・ GZIP の圧縮レベル (0~9)
- ・ シャッフルを行うべきかどうか
- ・ 保存するウェーブの各次元で使うチャンクサイズ

特定のデータセットの圧縮パラメーターは、データセットの作成時に設定され、既存のデータセットにデータを追加する時には変更できません。

これらのパラメーターを取得する方法はいくつかあります。

- ・ HDF5SaveData コマンドの /GZIP と /LAYO フラグから
- ・ HDF5SaveGroup、SaveExperiment、SaveData コマンドの /COMP フラグから
- ・ SaveGraphCopy、SaveTableCopy、SaveGizmoCopy コマンドの /COMP フラグから
- ・ HDF5SaveDataHook から (詳細は「HDF5SaveDataHook の使用」のセクションを参照)
- ・ 手動保存時の HDF5 のデフォルト圧縮を介して (詳細は「HDF5 のデフォルト圧縮」のセクションを参照)

HDF5 は、3つのカテゴリに分類されるフィルターを通じて、データセットの圧縮をサポートしています：

- ・ 内部フィルター：Shuffle、Fletcher32、ScaleOffset、NBit
HDF5 ライブラリのソースコードには、内部フィルターが実装されています。
- ・ 外部フィルター：GZIP、SZIP
外部フィルターは、HDF5 ライブラリがコンパイルされる時に、そのライブラリとリンクされます。
- ・ サードパーティのフィルター
サードパーティ製のフィルターは、実行時に HDF5 ライブラリによって検出され、読み込まれます。

Igor Pro は現在、以下の HDF5 フィルターに対応しています。

- ・ GZIP
- ・ SZIP (読み取り専用。書き込みはサポートしていない)
- ・ シャッフル

シャッフルは、マルチバイトデータ要素のバイト順序を入れ替える処理であり、これにより圧縮率を高めることができます。

HDF5 レイアウトチャンクサイズ

レイアウトとは、特定のデータセットのデータがディスク上にどのように配置されるかを指します。

HDF5 ライブラリは、連続 (contiguous)、コンパクト (compact)、チャンク (chunked) の3種類のレイアウトをサポートしています。

HDF5SaveData と HDF5SaveGroup コマンドは、デフォルトで連続レイアウトが設定されており、これは非圧縮のデータセットに適しています。

HDF5 ライブラリでは、圧縮されたデータセットはチャンク化されたレイアウトを使う必要があります。

これは、データセットのデータが1つの連続したブロックとしてではなく、いくつかの独立したチャンクに分けて書き込まれることを意味します。

圧縮の観点から、チャンク化は主にデータセットのサブセットへのアクセス速度を向上させるために有用です。

チャンク化を行わない場合、サブセットを読み込むにはデータセット全体を読み込んで解凍する必要があります。
HDF5 パックされたエクスペリメントファイルを読み込む場合など、データセットが常にその全体として読み込まれる場合は、チャンク化を行っても処理速度の向上にはつながりません。

チャンク化された保存を行うには、データセットの各次元について、HDF5 ライブラリにチャンクのサイズを指定する必要があります。

適切なチャンクサイズの選び方に普遍的な方法はありません。

これは、データの次元や性質、さらにはユーザーの判断を要するトレードオフに依存するためです。

Igor Pro では、チャンクサイズを指定することができます。

- ・ HDF5SaveData コマンドの /LAYO フラグを使う（詳細は HDF5SaveData コマンドのヘルプを参照）
- ・ HDF5SaveDataHook を使う（詳細は「HDF5SaveDataHook の使用」のセクションを参照）

HDF5 の SaveGroup、SaveExperiment、SaveData、SaveGraphCopy、SaveTableCopy、SaveGizmoCopy コマンド、さらに HDF5 のデフォルトの圧縮機能では、後述するセクションで説明するように、圧縮されたデータセットは常に 1 つのチャンクとして保存されます。

特定のデータセットの圧縮パラメーターは、そのデータセットの作成時に設定され、既存のデータセットにデータを追加する時には変更できません。

HDF5SaveGroup での圧縮

Igor Pro 9.0 以降では、/COMP フラグを使うことで、HDF5SaveGroup コマンドに数値データセットの圧縮を指示することができます。

HDF5SaveGroup は、数値ウェーブに対してのみ圧縮を適用します。

テキストウェーブやその他の非数値ウェーブ、/COMP フラグで指定された要素数未満の数値ウェーブに対しては圧縮を適用しません。

HDF5SaveGroup 圧縮では、ウェーブの各次元のサイズと等しいチャンクサイズが使われます。

このようなチャンクサイズでは、ウェーブ全体がチャンクレイアウトを使って 1 つのチャンクとして書き込まれます。

これは、データセットを一括で読み込む場合には問題ありません。

より詳細なコントロールを行うには、HDF5SaveDataHook 関数を使って /COMP で指定された圧縮設定を上書きすることもできますが、通常はこれを行う必要はありません。

SaveExperiment での圧縮

Igor Pro 9.0 以降では、/COMP フラグを使うことで、SaveExperiment コマンドに対して数値データセットの圧縮を指示することができます。

この動作は、前のセクションで説明した HDF5SaveGroup での圧縮と同じです。

SaveData、SaveGraphCopy、SaveTableCopy、SaveGizmoCopy の各コマンドについても同様です。

手順の詳細は、ヘルプ HDF5 Compression for Saving Experiment Files を参照してください。

HDF5 のデフォルトの圧縮

Igor Pro では、インターフェイスから HDF5 ファイルを保存する時に、デフォルトのデータセット圧縮を実行できます。

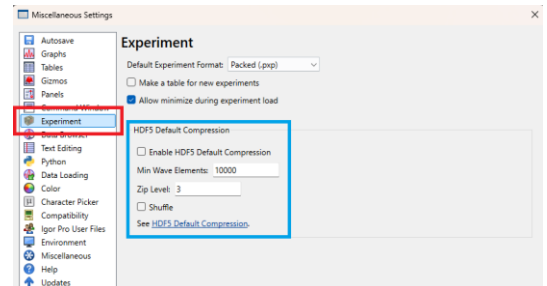
HDF5SaveData、HDF5SaveGroup、SaveExperiment、SaveData、SaveGraphCopy、SaveTableCopy、または SaveGizmoCopy コマンドを使って HDF5 ファイルを保存する場合は、デフォルトの圧縮は適用されません。HDF5 のデフォルト圧縮機能は、Igor Pro 9.0 で追加されました。

圧縮を行うと保存にかかる時間が大幅に長くなる可能性があるため、デフォルトでは圧縮は無効になっています。圧縮を行う価値があるかどうかは、データのサイズや性質、および時間とディスク容量のバランスをどう取るかによって異なります。

Misc→Miscellaneous Settings を選択し、Experiment メニューをクリックすると、デフォルトの圧縮機能を有効にできます。

これにより、以下の設定を行うことができます。

- ・ デフォルトの圧縮が有効か無効か
- ・ 圧縮が適用されるためのウェーブの最小サイズ
- ・ 0（圧縮なし）から 9（最大）までの圧縮レベル
- ・ 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



現在適用されているデフォルトの圧縮設定は、次のようにプログラムで判別できます。

```
Print IgorInfo(14) // デフォルトの圧縮設定を表示する
```

例えば、次のように出力されます。

```
ENABLED=0;MINWAVEELEMENTS=10000;GZIPLEVEL=3;SHUFFLE=0;
```

デフォルトの圧縮機能を有効にすると、ユーザーインターフェイスを介して HDF5 形式で圧縮されたエクスペリメントファイルを保存する時に、この設定が適用されます。

具体的には、Data Browser の Save Copy ボタン、HDF5 Browser の Save Waves と Save Data Folder ボタン、File メニューの以下の項目（Save Experiment、Save Experiment As、Save Experiment Copy、Save Graph Copy、Save Table Copy、Save Gizmo Copy）を通じて保存する場合は該当します。

HDF5SaveDataHook 関数が存在する場合、その関数によってデフォルトの圧縮設定を上書きすることができます。

Igor Pro は、デフォルトの圧縮を数値ウェーブにのみ適用し、テキストウェーブやその他の非数値ウェーブ、設定で指定された要素数未満の数値ウェーブには適用しません。

デフォルトの圧縮では、ウェーブの各次元のサイズと等しいチャンクサイズが与えられます。

このようなチャンクサイズでは、ウェーブ全体がチャンクレイアウトを使って 1 つのチャンクとして書き込まれます。

これは、一括で読み込まれるデータセットの場合は問題ありません。

HDF5 のデフォルトの圧縮設定は、ほとんどの用途において十分です。

必要に応じて（例えば、データセット全体を一度に読み込むのではなく、その一部を読み込む場合など）、HDF5SaveData コマンドまたは HDF5SaveDataHook 関数を使って、この設定を上書きすることができます。

HDF5SaveDataHook の使用方法

HDF5SaveDataHook は、特定のウェーブデータをデータセットとして保存する時に、どのような圧縮を適用すべきか（あるいは圧縮を適用しないか）を決定するために、Igor Pro が呼び出すユーザー定義関数です。

Igor Pro 9.0 では、HDF5SaveDataHook のサポートが追加されました。

注記

HDF5SaveDataHook は、上級ユーザーのみを対象とした実験的な機能です。この機能は変更または削除される可能性があります。

フック関数は、入力フィールドと出力フィールドを含む 1 つの HDF5SaveDataHookStruct パラメーターを受け取ります。

version、operation、waveToSave フィールドが入力です。

gzipLevel、shuffle、chunkSizes フィールドが出力です。

HDF5SaveDataHook の主な用途は、ウェーブの次元サイズや、ディスク容量と保存時間のトレードオフを考慮しつつ、ウェーブごとにチャンクサイズを指定できる手段を提供することです。

HDF5SaveData コマンドでは、/LAYO フラグを使ってチャンクサイズを指定できます。

しかし、/COMP フラグを通じて HDF5 圧縮をサポートするその他のコマンド（HDF5SaveGroup、SaveExperiment、SaveData、SaveGraphCopy、SaveTableCopy、SaveGizmoCopy）は、HDF5 のデフォルトの圧縮と同様に、常にデータセットを 1 つのチャンクとして保存します。

HDF5SaveDataHook を使うと、この動作を変更できますが、ほとんどの場合、その必要はありません。

コマンドフィールドがゼロ以外の場合、フック関数は HDF5SaveData または HDF5SaveGroup コマンドから呼び出されています。

これらのコマンドに指定された圧縮設定を、上書きする正当な理由がない限り、遵守する必要があります。

指定されたウェーブに対する圧縮パラメーターを設定するには、出力フィールドを設定して 1 を返します。

圧縮パラメーターを変更せずに保存を続行するには、0 を返します。

例を挙げます。

```
ThreadSafe Function HDF5SaveDataHook(s)
    STRUCT HDF5SaveDataHookStruct& s

    // Print "Global ThreadSafe HDF5SaveDataHook called" // デバッグ用
    // return 0 // このフックを NOP にするには、このコメントを外す

    if (s.version >= 2000) // 本稿執筆時点での構造体のバージョンは 1000
        return 0 // 構造が互換性を損なう形で変更された
    endif

    if (s.operation != 0) // HDF5SaveData または HDF5SaveGroup コマンドから呼び出されたものか?
        return 0 // 変更する正当な理由がない限り、設定を尊重する
    endif

    // 数値ウェーブに対してのみ圧縮パラメーターを設定する
    int isNumeric = WaveType(s.waveToSave) == 1
    if (!isNumeric)
        return 0
    endif

    // ウェーブが特定の閾値を超えた場合、要素の総数に応じたパラメーターを設定する
    int numElements = numpts(s.waveToSave)
    if (numElements < 10000)
        return 0
    endif

    // 圧縮パラメーターを設定する
    s.gzipLevel = 5 // 0（非圧縮）から 9（最大圧縮）まで
    s.shuffle = 0 // 0=シャッフル・オフ, 1=シャッフル・オン
    s.chunkSizes[0] = 200 // これらの値は任意。Igor Pro がこれらをクリップする
    s.chunkSizes[1] = 200 // 実際の関数では、これらを選ぶために、
    s.chunkSizes[2] = 50 // もっと体系的な方法を用いる必要がある
    s.chunkSizes[3] = 50
```

```
// このウェーブに対して圧縮パラメーターを設定したいことを示す
```

```
return 1
```

```
End
```

HDF5SaveDataHook 関数はスレッドセーフでなければなりません。
スレッドセーフでない場合、この関数を呼び出しません。

HDF5SaveDataHook 関数は、データセットが作成されたときに呼び出され、既存のデータセットにデータを追加する時には呼び出されません。

HDF5SaveDataHook 関数は、Igor Pro の状態を変更してはいけません。
つまり、ウェーブや変数の作成、ウェーブや変数の変更、ウェーブや変数の削除といった副作用があってはなりません。

この関数は、HDF5SaveData が呼び出されたとき、HDF5SaveGroup が呼び出されたとき、また HDF5 パック形式のエクスペリメントファイルを保存しているときに呼び出されます。

HDF5SaveDataHook 関数内で Igor Pro の状態を変更すると、ファイルの破損やクラッシュを引き起こす可能性があります。

HDF5SaveDataHook 関数を複数定義することは可能ですが、推奨されません。

例えば、ProcGlobal コンテキストに 1 つ、通常のコモジュールに 1 つ、独立したコモジュールに 1 つ、それぞれ定義することができます。

この場合、いずれかの関数が 1 を返すまで、各関数が順次呼び出されます。

関数の呼び出し順序は定義されていません。

HDF5SaveDataHook 関数は、通常のプロシージャが未コンパイルの状態でも呼び出せるように、独立したコモジュール内で定義することを推奨します。

そうしないと、例えばエラーなどの理由でプロシージャがコンパイルされていない状態で HDF5 圧縮エクスペリメントデータを保存した場合、HDF5SaveDataHook 関数が呼び出されず、すべてのデータセットがデフォルトの圧縮設定で保存されるか、あるいは非圧縮のまま保存されてしまいます。

次のコマンドを実行することで、Igor が HDF5SaveDataHook 関数を一切呼び出さないようにすることができます。

```
SetIgorOption HDF5SaveDataHook=0
```

これにより、HDF5SaveData と HDF5SaveGroup コマンド、あるいは HDF5 パックされた形式のエクスペリメントファイルを保存する時に、フック関数を呼び出すことが防止されます。

次のコマンドを実行することで、フック関数の呼び出しを再度有効にすることができます：

```
SetIgorOption HDF5SaveDataHook=1
```

HDF5 圧縮の参考情報

このセクションでは、HDF5 のフィルタリングや圧縮について解説している、上級 HDF5 ユーザーにとって参考になる可能性のある資料を一覧で紹介します。

HDF5 内の圧縮の使用方法

https://support.hdfgroup.org/documentation/hdf5-docs/hdf5_topics/UsingCompressionInHDF5.html

HDF5 内のチャンク化

https://support.hdfgroup.org/documentation/hdf5-docs/advanced_topics/chunking_in_hdf5.html

HDF5 の高度なトピック：HDF5 内のチャンク化

https://support.hdfgroup.org/documentation/hdf5-docs/advanced_topics/Chunking_Tutorial_EOS13_2009.pdf

データセットのチャンク化の問題

https://support.hdfgroup.org/documentation/hdf5/latest/hdf5_chunk_issues.html

HDF5 圧縮の詳細解説 #1

<https://www.hdfgroup.org/2015/04/hdf5-data-compression-demystified-1/>

HDF5 圧縮の詳細解説 #2

<https://www.hdfgroup.org/2017/05/hdf5-data-compression-demystified-2-performance-tuning/>

HDF5 圧縮データセットを扱う時の I/O パフォーマンスの向上

https://support.hdfgroup.org/documentation/hdf5-docs/hdf5_topics/HDF5ImprovingIOPerformanceCompressedDatasets.pdf

HDF5 圧縮のトラブルシューティング

https://support.hdfgroup.org/documentation/hdf5-docs/hdf5_topics/HDF5CompressionTroubleshooting.pdf

HDF5 の動的読み込みフィルター

HDF5 ライブラリでは、ライブラリ自体に組み込まれていない圧縮形式に対応するために、サードパーティ製の動的読み込み型フィルタープラグインを使うことができます。

「動的読み込み」とは、フィルタープラグインが HDF5 ライブラリにコンパイルされておらず、別のライブラリファイルに格納されていることを意味します。

HDF5 ライブラリはこれらのプラグインを検索し、見つかった場合はそれらを読み込み、その機能が利用可能になります。

(HDF5 の専門家向けに、動的に読み込まれるフィルタープラグインについては <https://support.hdfgroup.org/releases/hdf5/documentation/rfc/HDF5DynamicallyLoadedFilters.pdf> で説明されています。)

HDF5 ライブラリがフィルタープラグインを検索するデフォルトの場所は、次のとおりです。

```
%ALLUSERSPROFILE%/hdf5/lib/plugin (%ALLUSERSPROFILE% is C:\ProgramData on most systems)
```

ユーザーは、Igor Pro を起動する前に、環境変数 HDF5_PLUGIN_PATH に自身のプラグインのパスを設定することで、デフォルトの保存先を上書きすることができます。

Igor Pro には、The HDF Group (<https://www.hdfgroup.org/downloads/hdf5>) が提供するプラグインが同梱されています。

これらのプラグインには、以下のものが含まれます：

BLOSC, BLOSC2, BSHUF, BZ2, JPEG, LZ4, LZF, ZFP, ZSTD

プラグイン「BLOSC2」と「ZSTD」は、Igor Pro 10 で新たに追加されたものです。

Igor Pro では、これらのフィルターを使って作成されたデータセットのデコードに対応しています。

ただし、これらのフィルターを使ったエンコードにはまだ対応していません。

フィルターは「IgorBinaries_x64/hdf5plugins」に同梱されています。

Igor Pro が起動すると、HDF5 ライブラリに対し、デフォルトのフィルター保存場所や、HDF5_PLUGIN_PATH 環境変数で指定された場所（存在する場合）を検索する前に、まずこれらの場所からプラグインを探すよう指示します。

つまり、HDF5 ライブラリは、他の場所にあるフィルタープラグインではなく、Igor Pro に同梱されているフィルタープラグインを検出することになります。

上記に挙げたもの以外にも、「登録済みのフィルタープラグイン」は存在します。

包括的なリストは、

https://github.com/HDFGroup/hdf5_plugins/blob/master/docs/RegisteredFilterPlugins.md で見ることができます。

他のフィルタープラグインが必要になることはまずないでしょう。

もし必要になった場合は、デフォルトの配置場所、または HDF5_PLUGIN_PATH 環境変数で指定された場所に配置することができます。

HDF5 ライブラリは、Igor Pro の hdf5plugins フォルダー内でそれらが見つからなかった場合、その場所を検索します。

HDF5 次元スケール

HDF5 ライブラリは、H5DS API を通じて次元スケールをサポートしています。

次元スケールとは、別のデータセットの座標を提供するデータセットのことです。

データセットの次元 0 に対する次元スケールは、Igor の XY ペアにおける X ウェーブに例えられます。

この例えは、より高い次元にも当てはまります。

次元スケールは、主に HDF5 をベースとした netCDF-4 形式に関連して使われます。

ほとんどの Igor Pro ユーザーは、次元スケールについて知る必要はありません。

netCDF ファイル形式では、「変数 (variable)」という用語は、HDF5 における「データセット (dataset)」という用語と同様に使われます。

次元 n を持つ各変数には、n 個の名前付き次元が関連付けられています。

「座標変数 (coordinate variable)」と呼ばれる別の変数は、Igor Pro における X ウェーブが XY ペアの値を供給するように、次元のインデックスに対する値を供給することができます。

変数とその座標変数との関連付けは、その変数が作成される時に確立されます。

通常、特定の座標変数は、複数の変数の特定の次元に対する値を提供します。

例えば、netCDF ファイルでは、「latitude」や「longitude」という名前の座標変数と、その X 軸と Y 軸の次元が「latitude」と「longitude」に関連付けられた複数の 2D 変数 (画像) が定義される場合があります。

netCDF-4 ファイル形式は、HDF5 を用いて実装されています。

netCDF ライブラリは、HDF5 の次元スケール機能を利用して、座標変数およびそれらが変数と関連付けられる仕組みを実装しています。

Igor Pro における HDF5 の次元スケールのサポート

HDF5DimensionScale コマンドは、HDF5 次元スケールの作成と照会をサポートしています。

これは Igor Pro 9.0 で追加されました。

このコマンドは、パラメーターの設定には「キーワード=値」という構文を、アクションの呼び出しには値を持たない単純なキーワードを使って実装されています。

例えば、以下のコマンドは、特定のデータセットを次元スケールに変換し、その次元スケールを別のデータセットに紐付けます。

```
// XScale という名前のデータセットを、名前が "x" の次元スケールに変換する
```

```
HDF5DimensionScale dataset={fileID,"XScale"}, dimName="X", setScale
```

```
// データセット Data0 の次元 0 に、次元スケール XScale を関連付ける
```

```
HDF5DimensionScale scale={fileID,"XScale"}, dataset={fileID,"Data0"}, dimIndex=0, attachScale
```

詳細は、HDF5DimensionScale コマンドのヘルプを参照してください。

HDF5 次元スケールの実装の詳細

Igor Pro の HDF5 Browser を使って netCDF-4 ファイルを表示する場合、このセクションで、ブラウザーに表示される内容について理解を深めることができます。

HDF5 の「高レベル」ライブラリの一部である H5DS (次元スケール) API は、属性を使って、データセットを次元スケールとして指定したり、次元スケールを他のデータセットに関連付けたり、その関連付けを追跡したりします。

次元スケールの最も重要な特性は以下の通りです。

CLASS	データセットが次元スケールであることを示すには、“DIMENSION_SCALE” に設定します。
NAME	“X” のような次元の名前
REFERENCE_LIST	次元スケールが紐付けられているデータセットと次元を追跡するために使われる構造体の配列です。この構造体の各要素には、データセットへの参照と次元のインデックスが含まれています。

次元スケールが関連付けられたデータセットの最も重要な属性は、以下の通りです。

DIMENSION_LIST	データセットの各次元で使われる次元を追跡するために使われる、参照の可変長配列です。この配列には、データセットの次元ごとに 1 つの列があります。各列には、対応するデータセットの次元に紐付けられた次元スケールごとに 1 つの行があります。特定のデータセットの次元には、複数の次元スケールが紐付けられている場合があります。
DIMENSION_LABELS	データセットの各次元ラベルを含む 1D 配列です。

HDF5 次元スケールの参考情報

専門家の方向けに、HDF5 の次元スケールと関連する netCDF-4 の機能に関する追加情報源を以下に示します。

H5DS API 参考情報

https://support.hdfgroup.org/documentation/hdf5/latest/_h5_d_s__u_g.html

HDF5 次元スケール仕様 (2005)

https://support.hdfgroup.org/documentation/hdf5-docs/hdf5_topics/H5DS_Spec.pdf

NetCDF の HDF5 との互換性

https://docs.unidata.ucar.edu/netcdf-c/current/interoperability_hdf5.html

https://www.unidata.ucar.edu/blogs/developer/en/entry/dimensions_scales

NetCDF FAQ

<https://docs.unidata.ucar.edu/netcdf-c/current/faq.html>

NetCDF のドキュメント

<https://docs.unidata.ucar.edu/netcdf-c/current/index.html>

NetCDF データモデル

https://docs.unidata.ucar.edu/netcdf-c/current/netcdf_data_model.html

HDF5/netCDF-4 の規約

<http://stcorp.github.io/harp/doc/html/conventions/hdf5.html>

その他の HDF5 の情報

このセクションは、主に HDF5 の上級ユーザーにとって参考になる内容です。

HDF5 文字列フォーマット

文字列は、いくつかの形式を使って、HDF5 ファイルにデータセットとして、あるいは属性として書き込むことができます。

- ・ 変数の長さ
- ・ ヌル終端付き固定長
- ・ ヌルパディング付き固定長
- ・ 空白文字で埋めた固定長

さらに、文字列には「ASCII」または「UTF-8」のいずれかを指定することができます。

通常、文字列の書き込みに使われるフォーマットについて知る必要も、気にする必要もありません。

ただし、一部のプログラムでは、すべてのフォーマットがサポートされているわけではありません。

Igor Pro で作成された HDF5 ファイルをそのようなプログラムのいずれかに読み込もうとすると、エラーが発生する可能性があります。

その場合は、ファイルの書き込みに使われる文字列フォーマットを調整することで、問題を解決できる可能性があります。

このセクションの残りの部分では、そのような場合に役立つ情報を紹介します。

可変長文字列形式は、長さが異なる複数の文字列を含むデータセットや属性を書き込む時に最も有用です。

例えば、“ABC” と “DEFGHIJKLMNOP” という 2 つの文字列を含むデータセットを可変長で書き込む場合、各文字列とその長さが HDF5 ファイルに格納されます。

これにより、“ABC” には 3 バイト、“DEFGHIJKLMNOP” には 13 バイトが必要となり、さらに各文字列の長さを格納するために必要なスペースも加わります。

これらの文字列を固定長で書き込んだ場合、データセットまたは属性の固定長は少なくとも 13 バイト必要となり、“ABC” を書き込む時にパディングによって少なくとも 10 バイトが無駄になってしまいます。

固定長文字列形式は、単一の文字列で構成されるデータセットや属性を記述する時に最も有用です。

例えば、“ABC” は、パディングバイトを 0 とした 3 バイトの固定長データ型を使って記述することができます。

複数の文字列を固定長で書き出す場合、パディングモードによって余分なバイトがどのように埋められるかが決まります。

ヌル終端モードでは、余分なバイトはヌル (0) で埋められ、最初のヌルが文字列の終わりを示します。

ヌルパディングモードでは、余分なバイトはヌル (0) で埋められ、文字列の末尾に連続するヌルがすべて、文字列の終わりを示します。

空白文字パディングモードでは、余分なバイトはスペースで埋められ、文字列の末尾に連続するスペースがすべて、文字列の終わりを示します。

可変長と固定長の問題に加え、HDF5 にはテキストエンコーディングの問題もあります。

文字列データセットや属性は、それを書き込んだソフトウェアに応じて、ASCII または UTF-8 のいずれかとしてマークされます。

HDF5 ライブラリは、書き込まれたテキストがマーク通りに有効であるかどうかを確認したり、テキストエンコーディングの変換を行ったりしないため、このマークによってテキストが有効な ASCII または有効な UTF-8 であることが保証されるわけではありません。

このマークは、単に意図されたテキストエンコーディングを示すものに過ぎません。

HDF5 ライブラリは、読み取りプログラムが互換性のあるデータ型を使うことを必須としているため、ASCII また

は UTF-8 としてマークされたデータセットや属性を読み込む時に、一部のソフトウェアでエラーが発生する可能性があります。

後述するいくつかの例外を除き、HDF5SaveData と HDF5SaveGroup コマンドで /STRF={fixedLength,paddingMode,charset} フラグを使うことで、データセットや属性を保存する時に使う文字列形式をコントロールできます。
/STRF フラグは Igor Pro 9.0 で追加されました。

fixedLength が 0 の場合、HDF5SaveData は可変長の HDF5 文字列データ型を使って文字列を書き込みます。
fixedLength が 0 より大きい場合、HDF5SaveData は、指定された長さの固定長 HDF5 文字列データ型を使い、パディングモードで指定されたパディングを適用して文字列を書き込みます。
fixedLength が -1 の場合、HDF5SaveData は、指定されたデータセットまたは属性に対して書き込まれる文字列のうち最長のものの長さを判定し、その長さを持つ固定長の HDF5 文字列データ型を使って、paddingMode で指定されたパディングを適用して文字列を書き込みます。

paddingMode が 0 の場合、HDF5SaveData は固定長文字列をヌル終端文字列として書き込みます。
paddingMode が 1 の場合、HDF5SaveData は固定長文字列をヌルで埋められた文字列として書き込みます。
paddingMode が 2 の場合、HDF5SaveData は固定長文字列を空白文字で埋められた文字列として書き込みます。
可変長の文字列 (fixedLength=0) を書き込む場合、paddingMode は無視されます。

charset が 0 の場合、HDF5SaveData は文字列を ASCII とマークして書き込みます。
charset が 1 の場合、HDF5SaveData は文字列を UTF-8 とマークして書き込みます。

例外として、長さがゼロのデータセットや属性は、常に可変長 UTF-8 として書き込まれます。
もう 1 つの例外は、HDF5SaveGroup によって書き込まれる文字列変数で、これらは常に固定長でヌル文字で埋められた UTF-8 として書き込まれます。

次の表は、/STRF を省略した場合にさまざまな状況で使われるデフォルトの文字列形式と、HDF5SaveData と HDF5SaveGroup がそれぞれの状況において /STRF フラグを反映するかどうかを示しています。

	HDF5SaveData のデフォルト	HDF5SaveData の動き
Text Wave Zero Element as Dataset	可変, NULLPAD, UTF-8	/STRF を無視
Text Wave Single Element as Dataset	可変, NULLPAD, UTF-8	/STRF に従う
Text Wave Multiple Elements as Dataset	可変, NULLPAD, UTF-8	/STRF に従う
String Variable	N/A	HDF5SaveData は文字列変数を保存できない
Text Wave Zero Element as Attribute	可変, NULLPAD, UTF-8	/STRF を無視
Text Wave Single Element as Attribute	固定, NULLPAD, UTF-8	/STRF に従う
Text Wave Multiple Elements as Attribute	可変, NULLPAD, UTF-8	/STRF に従う
	HDF5SaveGroup のデフォルト	HDF5SaveGroup の動き
Text Wave Zero Element as Dataset	可変, NULLPAD, UTF-8	/STRF を無視
Text Wave Single Element as Dataset	可変, NULLPAD, UTF-8	/STRF に従う
Text Wave Multiple Elements as Dataset	可変, NULLPAD, UTF-8	/STRF に従う
Zero-Length String Variable	可変, NULLPAD, UTF-8	/STRF を無視
String Variable > Zero-Length	固定, NULLPAD, UTF-8	/STRF を無視

Text Wave Zero Element as Attribute	N/A	HDF5SaveGroup は属性を保存できない
Text Wave Single Element as Attribute	N/A	HDF5SaveGroup は属性を保存できない
Text Wave Multiple Elements as Attribute	N/A	HDF5SaveGroup は属性を保存できない

HDF5 文字列変数のテキストエンコーディング

Igor Pro は、文字列変数を UTF-8 の固定長文字列データセットとして記述します。

文字列変数には、ヌルバイトや、UTF-8 として無効なテキストが含まれている場合があります。これは、例えば、変数がバイナリデータを格納するために使われた場合などに発生します。このような文字列変数も、UTF-8 固定長文字列データセットとして書き込まれます。

HDF5 ウェーブのテキストエンコーディング

ウェーブテキストエンコーディングに関する背景情報は、ヘルプ Wave Text Encodings を参照してください。

Igor Pro のテキストウェーブデータの内容は、UTF-8 テキストエンコーディングを使った可変長文字列データセットとして記述されます。その他のウェーブ要素（ユニット、ノート、次元ラベル）は、UTF-8 固定長文字列属性として記述されます。

テキストウェーブの内容には、ヌルバイトや、UTF-8 として無効なテキストが含まれている場合があります。これは、例えば、テキストウェーブにバイナリデータを格納した場合などに発生します。そのような内容も、UTF-8 可変長文字列データセットとして書き込まれます。

Igor Pro では、ウェーブのテキスト要素に、サポートされている任意のテキストエンコーディングを指定することができます。ウェーブのテキスト要素にどのようなエンコーディングが指定されていても、HDF5 には UTF-8 文字列として書き込まれます。したがって、UTF-8 以外のテキストエンコーディングを使っているウェーブを HDF5 ファイルとして保存し、それを Igor Pro に読み戻すと、テキストエンコーディングは変更されますが、テキストで表される文字自体は変わりません。

バイナリとしてマークされたウェーブ要素には例外が適用されます（背景情報は、ヘルプ Text Waves Containing Binary Data を参照してください）。

1 つ以上のバイナリ要素を含むウェーブを HDF5 データセットとして保存すると、そのデータセットに IGORWaveBinaryFlags 属性を追加します。

この属性は、WaveTextEncoding 関数で定義されたビットを使って、バイナリとしてマークされたウェーブ要素を識別します。

HDF5 ファイルからデータセットを読み込む時、この属性に設定されたビットに対応するウェーブ要素のバイナリマークを復元します。

IGORWaveBinaryFlags 属性は、Igor Pro 9.0 で追加されました。

Igor Pro 9.0 以降では、HDF5LoadData と HDF5LoadGroup コマンドにより、文字列データセットからテキストウェーブに読み込まれたバイナリデータを検出し、そのようなテキストウェーブをバイナリを含むものとしてマークすることができます。

文字列データセットからバイナリデータを読み込む場合は、詳細について、これらのコマンドの /OPTS フラグのビット 1 の説明を参照してください。

Igor Pro の HDF5 との互換性

このセクションでは、HDF5 ライブラリのさまざまなバージョンに関連する事項について解説します。

HDF5 ライブラリのバージョン 1.8.0 より前のものを基にした、非常に古いソフトウェアを使っている場合を除き、この情報を知る必要はありません。

以下の表は、Igor Pro と HDF5 の各バージョンの一覧です。

Igor Pro バージョン	HDF5 ライブラリバージョン
Igor Pro 6.03	1.6.3 (released on 2004-09-22)
Igor Pro 6.10 to 6.37	1.8.2 (released on 2008-11-10)
Igor Pro 7.00 to 7.08	1.8.15 (released on 2015-05-04)
Igor Pro 8.00 to 8.04	1.10.1 (released on 2017-04-27)
Igor Pro 9.00 to 9.0x	1.10.7 (released on 2020-09-15)
Igor Pro 10.00	1.14.5 (released on 2024-09-30)

HDF5 の互換性に関する用語

以降において、「旧バージョンの HDF5」とは、HDF5 ライブラリのバージョン 1.8.0 より前のバージョンを指します。

「旧バージョンの HDF5 プログラム」とは、旧バージョンの HDF5 を使うプログラムを指します。

「旧バージョンの HDF5 ファイル」とは、旧バージョンの HDF5 プログラムによって作成された HDF5 ファイルを指します。

古い HDF5 プログラムで Igor Pro の HDF5 ファイルを読み込む

デフォルトでは、Igor Pro 10 以降では、HDF5 ライブラリバージョン 1.8.0 以降でコンパイルされたプログラムと互換性のある HDF5 ファイルが生成されます。

HDF5 1.8.0 は 2008 年 2 月にリリースされました。

ほとんどの用途において、このデフォルトの互換性設定で問題なく動作します。

このセクションの残りの部分は、HDF5 ライブラリのバージョン 1.8.0 より前のものを使う非常に古い HDF5 プログラムで Igor HDF5 ファイルを読み込む必要がある場合にのみ参考になります。

<https://docs.hdfgroup.org/archive/support/HDF5/doc/ADGuide/CompatFormat180.html> で説明されているように、HDF5 1.6.2 (2004 年 2 月 12 日リリース) またはそれ以前のバージョンでコンパイルされたソフトウェアは、HDF5 1.8.0 以降で生成された HDF5 ファイルと互換性がありません。

Igor Pro 10 以降で書き込まれたファイルを、HDF5 1.6.3 (2004 年 9 月 22 日リリース) から 1.6.10 (2009 年 11 月 10 日リリース) までのバージョンを使うソフトウェアで読み込む必要がある場合は、Igor Pro に互換性のある HDF5 フォーマットを使うよう指定する必要があります。

これを行うには、次のコマンドを実行します。

```
SetIgorOption HDF5LibVerLowBound=0
```

これを行うと、以下の機能が利用できなくなります：

- ・ 65,535 バイトを超える属性（「大容量属性」）を保存する機能
グループやデータセットに大きな属性値を書き込もうとした場合、エラーが発生します。
これには、HDF5SaveData や HDF5SaveGroup を使う場合、あるいは HDF5 パケット形式のエクスペリメントファイルを保存する場合に、非常に大きなウェーブノートや非常に多くの次元ラベルを含むウェーブを保存することが含まれます。
また、HDF5SaveData/A を使って大きな属性を追加することも含まれます。
- ・ グループやデータセットを作成順に並べ替える機能
この機能は HDF5 Browser によって提供されていますが、ファイル作成時にこの機能が有効に設定されていたファイルでのみサポートされています。
- ・ 属性を作成順に読み取れるように記述する機能

Igor Pro を正常な動作状態に戻すには、次のコマンドを実行します。

```
SetIgorOption HDF5LibVerLowBound=1
```

SetIgorOption の効果は、Igor Pro を再起動するまでしか持続しません。

通常、HDF5LibVerLowBound を設定する必要がある場合は、起動時に一度だけ行うようにします。
Igor Pro のプリエンティブスレッドが HDF5 呼び出しを行っている間は、SetIgorOption を呼び出して HDF5LibVerLowBound を設定しないでください。

古い HDF5 ファイルへの書き込み

HDF5 のドキュメントには、さまざまな HDF5 ライブラリのバージョン間で生じる数多くの互換性問題のすべてが明記されているわけではありません。

このセクションの情報は、当社による実証テストに基づいています。

このセクションで使われている用語の定義については、「HDF5 互換性に関する用語」のセクションを参照してください。

Igor Pro を使って古い HDF5 ファイルを開き、そこにデータセットを書き込むと、その新しいデータセットは古い HDF5 プログラムでは読み取ることができません。

また、新しいデータセットを含むグループは、古い HDF5 プログラムでは一覧表示できません。

つまり、古いファイルに新しい形式で書き込むと、その古いファイルは、少なくとも部分的に、古いソフトウェアでは読み取れなくなってしまうます。

古い HDF5 ファイルがあり、それらを読み込むために古い HDF5 プログラムを使っている場合は、そのようなファイルを開く時は「読み取り専用」モードで開き、「読み取り/書き込み」モードでは開かないでください。

また、ファイルが確実にバックアップされていることを確認してください。

HDF5 パックされたエクスペリメントファイル

Igor Pro 9 以降では、エクスペリメントを 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 をサポートする幅広いプログラムから、そのデータにすぐにアクセスできることです。

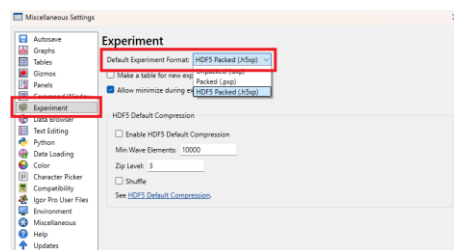
主な欠点は、Igor Pro でそのファイルを開くには Igor Pro 9 以降が必要になることです。

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、メニュー File→Save Experiment As を選択します。
表示される Save File 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」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 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格納しているかの詳細を知る必要はありませんが、もし興味があれば、HDF5 Browser（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、その仕組みを大まかに把握することができます。

このトピックの残りの部分は、他のプログラムから HDF5 パケット形式のエクスペリメントファイルを読み込んだり書き込んだりしたいプログラマーのために用意されています。
ここでは、読者が Igor Pro と HDF5 の熟練ユーザーであることを前提としています。

HDF5 パックされたエクスペリメントファイルの構成

HDF5 形式でパックされたエクスペリメントファイルは、一般的に次のような構成になっています。

```
/
  History
    History // 履歴エリアウィンドウの内容
  Packed Data
    <Waves, variables and data folders here>
  Shared Data
    <Paths to shared wave files here>
  Free Waves
    <Free waves here>
  Free Data Folders
    <Free data folders here>
  Packed Procedure Files
    Procedure // ビルトインプロシージャウィンドウの内容
    <Zero or more packed procedure files>
  Shared Procedure Files
    <Paths to shared procedure files>
  Packed Notebooks
    <Packed notebook files>
  Shared Notebooks
    <Paths to shared notebook files>
  Symbolic Paths
    <Paths to folders associated with symbolic paths>
  Recreation
    Recreation Procedures // エクスペリメント再作成プロシージャ
  Miscellaneous
    Dash Settings (dataset)
    Recent Windows Settings (dataset)
    Running XOPs (dataset)
    Pictures
      <Picture datasets here>
    Page Setups
      Page Setup for All Graphs (dataset)
      Page Setup for All Tables (dataset)
      Page Setup for Built-in Procedure Window (dataset)
      <Datasets for page layout page setups here>
```

必要がない場合は、最上位のグループを省略しても構いません。

例えば、エクスペリメントに「フリーウェーブ」が含まれていない場合、Igor Pro は Free Waves グループを記述しません。

Shared Data グループには、エクスペリメントで参照される Igor Pro バイナリウェーブ (.ibw) ファイルのフルパスを含むデータセットが格納されています。

Windows 環境では、これらのパスは Windows 形式のパスで表記されます。

Shared Procedure Files グループには、エクスペリメントで参照されるプロシージャ (.ipf) ファイルへのフルパスを含むデータセットが含まれています。

Windows 環境では、これらのパスは Windows 形式のパスで表記されます。

グローバルプロシージャファイルと #include で読み込まれるプロシージャファイルは、エクスペリメントの一部ではないため、含まれていません。

Shared Notebooks グループには、エクスペリメントで参照されているノートブックファイル (.ifn または任意のプレーンテキストファイル形式) へのフルパスが含まれるデータセットが格納されています。

Windows 環境では、これらのパスは Windows 形式のパスで表記されます。

Symbolic Paths グループには、ユーザーが作成したシンボリックパスのみが含まれます。

定義上、パックされたエクスペリメントファイルを含むフォルダーを指す組み込みのシンボリックパス Igor、IgorUserFiles、home は含まれません。

Windows 環境では、これらのパスは Windows 形式のパスで表記されます。

ユーザーが作成したシンボリックパスがない場合、Igor Pro は Symbolic Paths グループを記述しません。

Recreation Procedures データセットには、エクスペリメントを再現するために Igor Pro が作成したエクスペリメント再作成プロシージャが含まれています。

Miscellaneous グループには、データセットとサブグループが含まれています。

これらのオブジェクトは、必要がない場合は省略可能です。

Miscellaneous グループ内のデータの形式は変更される可能性があり、文書化されていません。

プログラムで HDF5 パケット形式のエクスペリメントファイルを読み書きする場合は、これらのオブジェクトの読み書きを行わないことを推奨します。

HDF5 パックされたエクスペリメントファイルの階層

HDF 5 ファイルは、厳密な階層構造というよりは「有向グラフ」です。

つまり、グループが自分自身の子になる、あるいはデータセットが複数グループの子になるなど、オブジェクト間に奇妙な関係を作り出すことが可能です。

HDF5 パックされたエクスペリメントファイルでは、このような奇妙な関係は許可されていません。

これらは厳密な階層構造を構成していなければなりません。

HDF5 パックされたエクスペリメントファイル内のプレーンテキストの保存

エクスペリメントファイルには、以下のようないくつかの種類のプレーンテキスト項目が保存されています。

- ・ ビルトインのプロシージャウィンドウの内容
- ・ ビルトインの履歴ウィンドウの内容
- ・ パックされたプロシージャファイルの内容

- ・ パックされたプレーンテキスト・ノートブックファイルの内容
- ・ テキストウェーブのテキストデータの内容
- ・ 数値ウェーブのテキストプロパティ（例：単位、ウェーブメモ、次元ラベル）
- ・ 文字列変数

Igor Pro は、HDF5 パックされたエクスペリメントファイルを書き込む時、そのようなテキストを UTF-8 テキストエンコーディング（H5T_CSET_UTF8）を使って文字列（HDF5 クラス H5T_STRING）として格納します。

有効な UTF-8 であると想定される項目であっても、UTF-8 では無効なバイトシーケンスが含まれていたり、ヌルバイトが含まれていたりする場合があります。

例えば、テキストデータではなくバイナリデータを格納するために使われているテキストウェーブや文字列変数などが挙げられます。

このようなデータも、依然として UTF-8 文字列データセットとして書き込まれます。

プレーンテキストファイル

この文脈における「プレーンテキストファイル」とは、履歴テキスト、再作成プロシージャ、プロシージャファイル、プレーンテキストのノートブックを指します。

Igor Pro は、プレーンテキストファイルを UTF-8 文字列データセットとして書き込みます。

ウェーブ

Igor Pro は、テキストウェーブを UTF-8 文字列データセットとして書き込みます。

詳細については、「HDF5 Wave Text Encoding」を参照してください。

文字列変数

Igor Pro は、文字列変数を UTF-8 の固定長文字列データセットとして記述します。

詳細は、「HDF5 文字列変数のテキストエンコーディング」のセクションを参照してください。

HDF5 パックされたエクスペリメントの書き込み

このセクションは、Igor Pro 以外のプログラムから HDF5 パックされたエクスペリメントファイルを作成したいプログラマーを対象としています。

このセクションでは、「HDF5 パケット形式のエクスペリメントファイル」の前のセクションで説明された内容を理解していることを前提としています。

HDF5 Browser を使って HDF5 パックされたエクスペリメントファイルを開くと、そのファイル（HDF5 ファイル内で「root」として表示される最上位グループ）にはいくつかの属性があることがわかります。

必須の属性は IGORRequiredVersion 属性のみであり、これはエクスペリメントファイルを開くために必要な Igor Pro の最低バージョンを指定するものです。

この属性には「9.00」と入力します。

HDF5 パックされたエクスペリメントファイルがそれより新しいバージョンの Igor Pro を必要とする場合は、それより大きい値を入力します。

HDF5 ファイル内のウェーブ

Igor Pro は、ウェーブの特性を表す属性を伴うデータセットとして、ウェーブを HDF5 ファイルに書き込みます。

例えば、IGORWaveType 属性はウェーブのデータ型を表します。

データ型のリストについては、WaveType コマンドのヘルプを参照してください。

HDF5 のウェーブ属性のリストについては、HDF5SaveData コマンドの /IGOR フラグに関する説明を参照してください。

ウェーブ参照ウェーブとデータフォルダー参照ウェーブ

ウェーブ参照ウェーブ（ヘルプ Wave Reference Waves を参照）、または略して「ウェーブウェーブ」とは、その要素が他のウェーブへの参照であるウェーブのことです。

ウェーブ参照ウェーブによって参照される各ウェーブには、参照先のウェーブのウェーブ ID を指定する IGORWaveID 属性があります。

ウェーブ参照ウェーブの各要素の内容は、参照先のウェーブのウェーブ ID です。

ウェーブ ID が 0 の場合は、ウェーブ参照がヌルであることを示します。

エクスペリメントの読み込み時に、各「ウェーブウェーブ」の各要素を復元し、適切なウェーブを指すようにします。

データフォルダー参照ウェーブ（「データフォルダー参照ウェーブ」を参照）、略して「DFREF ウェーブ」とは、その要素がデータフォルダーへの参照であるウェーブのことです。

DFREF ウェーブによって参照される各データフォルダーには、参照先のデータフォルダーの ID を指定する IGORDataFolderID 属性があります。

DFREF ウェーブの各要素の内容は、参照先のデータフォルダーの ID です。

データフォルダー ID が 0 の場合は、データフォルダー参照がヌルであることを示します。

エクスペリメントの読み込み時に、各 DFREF ウェーブの各要素を復元し、適切なデータフォルダーを指すようにします。

ウェーブと DFREF ウェーブの書き込みと復元は複雑で、扱いが難しいものです。

プログラムで HDF5 パックされたエクスペリメントファイルを書き込む場合は、これらのオブジェクトの書き込みを試みないことを推奨します。

プログラムで HDF5 パックされたエクスペリメントファイルを読み込む場合は、ウェーブ

（IGORWaveType=16384）と DFREF ウェーブ（IGORWaveType=256）を無視することを推奨します。

フリーウェーブとフリーデータフォルダー

Igor Pro は、フリーウェーブとフリーデータフォルダーの表現を HDF5 パックされたエクスペリメントファイルに書き込み、エクスペリメントが読み込まれる時にフリーウェーブとフリーデータフォルダーを復元します。

各フリーウェーブは、Free Waves グループ内のデータセットとして記述されます。

フリーウェーブが存在するためには、エクスペリメント内のウェーブによって参照されている必要があります。

各フリールートデータフォルダーとその子フォルダーは、Free Data Folders グループ内のグループとサブグループとして記述されます。

フリールートデータフォルダーが存在するためには、エクスペリメント内の DFREF ウェーブによって参照されている必要があります。

フリーウェーブとデータフォルダーへの書き込みや復元は、複雑で扱いが難しい作業です。

使っているプログラムが HDF5 パックされたエクスペリメントファイルを書き込む場合は、これらのオブジェクトへの書き込みを試みないことを推奨します。

使っているプログラムが HDF5 パックされたエクスペリメントファイルを読み込む場合は、Free Waves と Data Folders のグループを無視することを推奨します。

HDF5 パックされたエクスペリメントの問題

ごくまれに、名前の競合により、Igor Pro のエクスペリメントデータを HDF5 パックされた形式で記述できない場合があります。

詳細は、ヘルプ [Object Name Conflicts and HDF5 Files](#) を参照してください。