
Open Platform of Transparent Analysis Tools for fNIRS

表示と Layout-Editor

国立研究開発法人 産業技術総合研究所

目次

1. 概要.....	3
1.1. POTAToにおける表示機能.....	3
1.2. 表示処理.....	3
1.3. 説明内容.....	4
2. POTAToの表示機能.....	5
2.1. POTAToと表示処理.....	5
2.2. 特徴.....	6
3. Layout Editorの使い方.....	7
3.1. LAYOUTの構成要素.....	7
3.2. LAYOUTの構造とデータの引継ぎ.....	8
3.3. Layout Editor概要.....	10
3.3.1. 概要.....	10
3.3.2. 起動.....	11
3.3.3. ファイルI/O.....	12
3.3.4. テスト表示.....	12
3.4. LAYOUTツリー.....	13
3.4.1. LAYOUTツリーの表示.....	13
3.4.2. ツリー操作.....	14
3.5. LAYOUT Overview.....	15
3.6. Figureの設定.....	16
3.7. Areaの設定.....	17
3.7.1. AreaのPrimary設定.....	17
3.7.2. AreaのVariables設定.....	18
3.7.3. AreaのOthers設定.....	19
3.8. CO: Control-Object.....	21
3.9. Axis-Areaの設定.....	21
3.10. AO: Axis-Object設定.....	22

4. 表示機能拡張	23
4.1. 説明内容	23
4.2. 表示機能とLAYOUT	23
4.3. データ構造	25
4.3.1. POTAToデータ	25
4.3.2. ObjectData	26
4.3.3. AO: Draw処理と関連データ	27
4.3.4. CO: Callback処理と関連データ	27
4.4. 補助関数	28
4.4.1. POTAToデータの取得	28
4.4.2. AO関連データのI/O	29
4.5. AO: Axis-Objectの作成	30
4.5.1. AO関数インタフェース	30
4.5.2. createBasicInfo	31
4.5.3. getArgument	31
4.5.4. drawstr, draw	31
4.6. CO: Control-Objectの作成	35
4.6.1. createBasicInfo	37
4.6.2. getArgument	37
4.6.3. drawstr, make	38
4.6.4. mycallback	40
付録: Script AO の使用方法	1
概要	1
設定	1
Axis用スクリプト	2
描画用スクリプト	3

1. 概要

1.1. POTAToにおける表示機能

本書は Platform for Optical topography Analysis Tools(POTATo)における表示について説明します。

POTATo では以下のような表示機能を備えています。

- 様々な表示方法を選択可能
- 表示された図をインタラクティブに変更可能
- 表示方法を変更可能
- 表示する内容を拡張可能

1.2. 表示処理

POTATo では表示内容が記述された LAYOUT ファイルに基づきデータを描画します。この LAYOUT ファイルは POTATo の補助ツールの一つである Layout Editor を用いて作成, 編集します。

POTATo は作成された LAYOUT と POTATo データを用いて表示を実行します。ここで POTATo データとは POTATo の解析結果として得られる連続データや区間データおよび要約統計量です。

POTATo における表示処理は以下の流れになります。

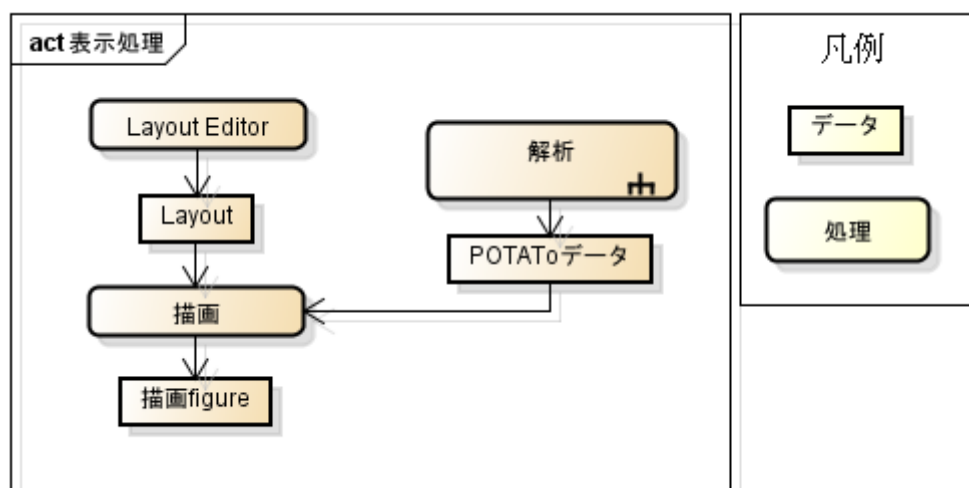


図 1.1 表示処理

1.3. 説明内容

以下、2章で POTATo の表示機能として、操作方法と LAYOUT で出来ることを説明します。

次に3章で LAYOUT の編集方法を説明します。編集に必要な知識として、LAYOUT の構成要素や変数(curdata)のスコープについても説明します。

最後に4章で LAYOUT の拡張方法を説明します。ここでは Figure の制御を行う CO や、実際のデータの描画を行う AO の作成方法について説明します。

2. POTATo の表示機能

2.1. POTATo と表示処理

POTATo メイン画と表示処理の対応を示します。

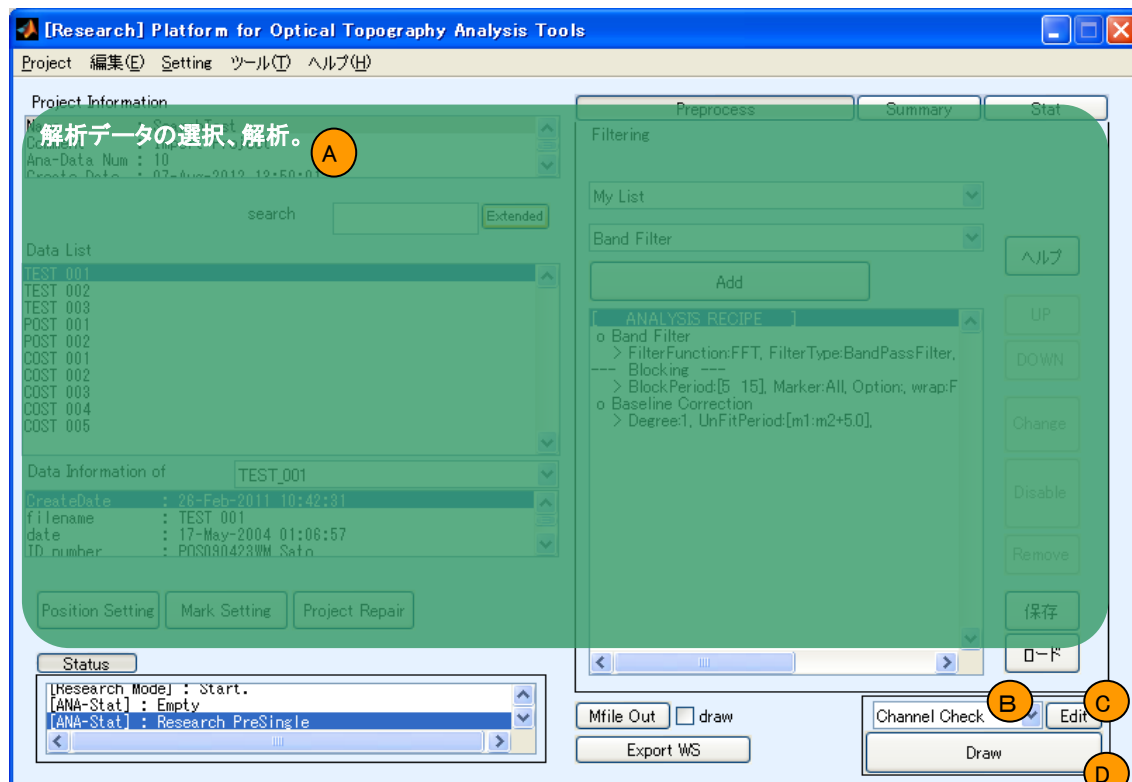


図 2.1 Normal モード メインウィンドウと領域

解析領域(A)で、表示する解析データを選択し、解析方法の編集を行います。解析方法は目的により異なります。

表示方法として LAYOUT をポップアップメニュー(B)から選びます。選択中の LAYOUT を編集したい場合は Edit ボタン(C)を押します。

最後に Draw ボタン(D)を実施すると、(A)で指定した解析に従い POTATo データが作成され、(B)の LAYOUT に従い描画処理が実行されます。

2.2. 特徴

POTATo ではデフォルトで数十個の LAYOUT を用意しています。

また、描画した図に対してチャンネルやデータの種類等を変更するための制御するものがあります。

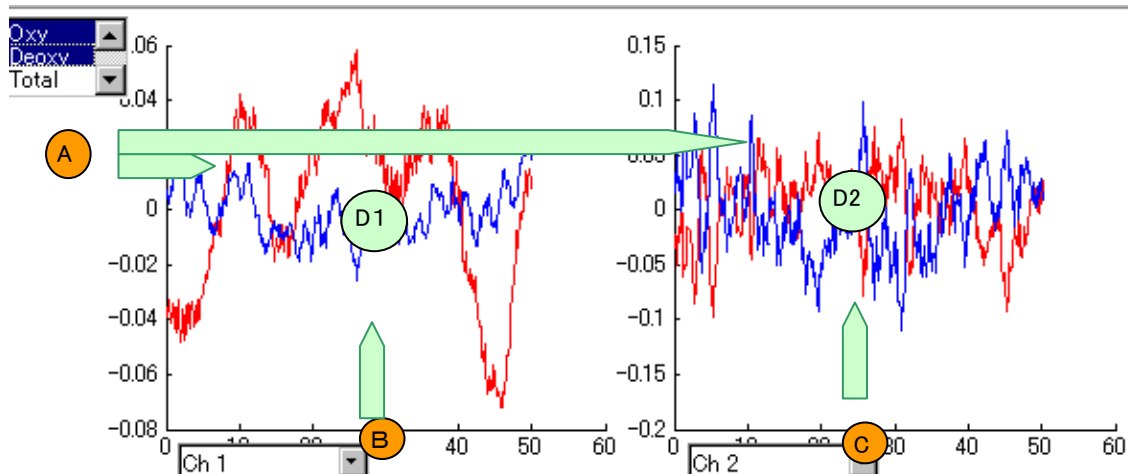


図 2.2 描画結果と制御の範囲

この時、制御には影響範囲があります。

例えば図 2.2 描画結果と制御の範囲を考えます。

図中のデータの種類の制御するリストボックス(A)は(D1)および(D1)を書き換えます。

また、チャンネル番号を制御するポップアップメニュー(B)は(D1) のみ書き換え、ポップアップメニュー(C)は(D2) のみ書き換えます。

LAYOUT は 3 章で説明する Layout Editor を使い、追加・編集可能です。

また、図の制御や表示物を追加し、機能を拡張することが出来ます。

これらの特徴を可能にするため、Layout は様々な構成物の組み合わせで示されます。構成物を組み合わせることにより、様々な Layout が作成でき、編集可能になります。

Layout 構成物には、制御する影響範囲を示す描画領域(Area)が存在します。Area は親子関係を持ち、親が変更されると子も影響を受けます。例えば図 2.2 の場合、(B)、(C) が属する Area は(A) が属する Area の子になります。

また、構成物の制御を行う(CO)や描画を行う(AO)が存在します。これらは追加することが可能です。

3. Layout Editor の使い方

3.1. LAYOUT の構成要素

最初に編集する対象となる Layout の構造を示します。

Layout は Figure、Area、Axis-Area および CO、AO の 5 つの構成要素からなります。

Figure は MATLAB の figure のプロパティを書き換えます。Figure には Area や Axis-Area を含みます。

Area は表示物を記載する領域で、描画の実態は持ちません。制御を行うための Layout 構成物 CO や、Area や Axis-Area を含みます。

Axis-Area は MATLAB の axes を描画します。また Axes 内に描画するための Layout 構成物、AO を含みます。

CO は MATLAB の uicontrol や uimenu などを描画します。これらの GUI により、描画した図に対してチャンネルやデータの種類等を変更します。CO には設定・描画・制御のための処理が含まれます。

AO は Axis 内に Line や Image を描画します。AO はこれらの設定・描画・再描画のための処理が含まれます。

3.2. LAYOUT の構造とデータの引継ぎ

LAYOUT は Figure をルートとし、Area、Axis-Area をリーフとする木構造で出来ています。

具体的に Figure、Area、Axis-Area の LAYOUT の構成要素は親子関係をもち、1つの親に対して0個以上の子が存在します。最初の親は Figure で、子どもには Area/Axis-Area を持つことができます。Area は Area 自身および Axis-Area を子に持つことができ、Axis-Area は子を持ちません。

また、Area は構成物として CO を、Axis-Area は構成物として AO を持ちます。

この関係をクラス図で表すと以下のようになります。

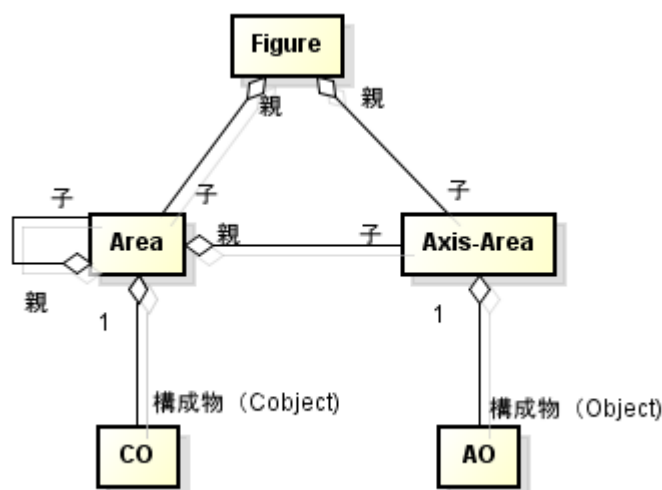


図 3.1 LAYOUT のデータ構造(編集時)

これらの親子関係は描画実施時のデータの引継ぎ方法に深く関係します。

描画時を考えると、描画するチャンネルやデータの種類、線の色など、描画のための様々なデータがあります。これらのデータは curdata (current-data) 構造体にまとめて保存・管理されます。

curdata は全ての構成要素が持っており、親から子へ引き継がれます。また子や兄弟から変更されません。

CO は所属する Area の curdata に従って初期設定されます。ただし CO に curdata の初期値が設定されている場合、その Area の curdata を書き換えます。

AO は所属する Area の curdata に従って描画されます。先祖に CO が居る場合、描画後、CO の Callback により curdata は書き換えられることがあります。

データの影響範囲について、図 3.2 Layout のオブジェクト図を例に説明します。

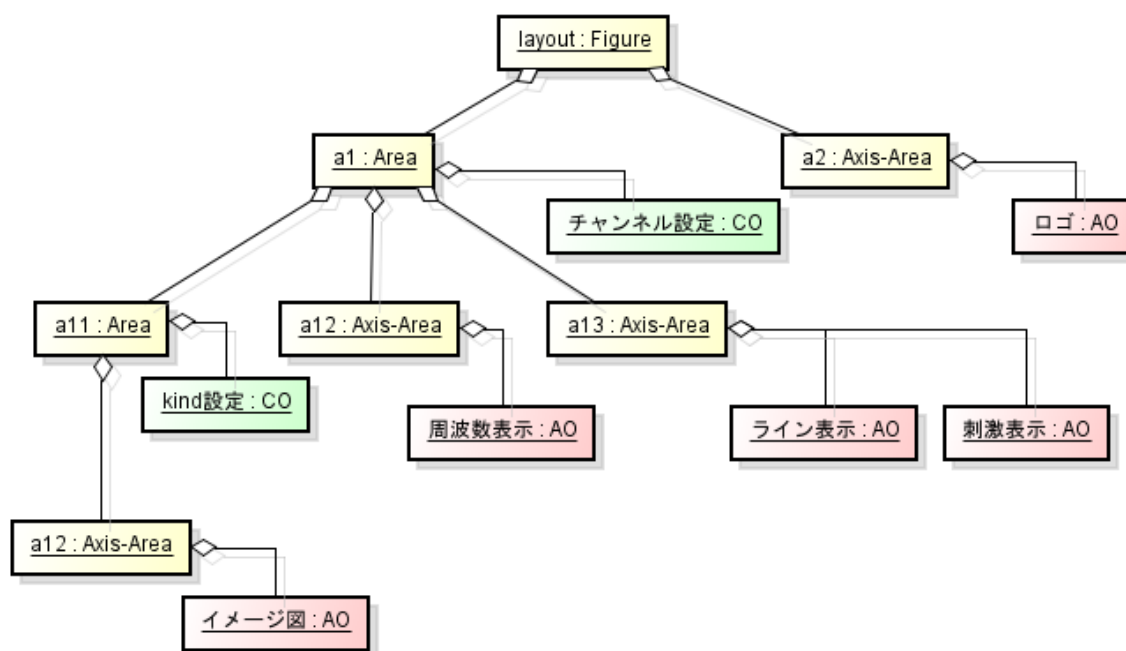


図 3.2 Layout のオブジェクト図

例にある Layout を描画するとき、layout:Figure で curdata が作成されます。layout の子、a1, a2 には同じ curdata が渡ります。

a1 ではチャンネル設定を行う co が存在し、curdata のチャンネル値を変更します。この時、a2 の curdata は変更されませんが、その子、a11, a12, a13 には変更されたチャンネル値が引き継がれます。

a11 には kind を設定する CO が存在し、curdata の kind を変更します。この結果はイメージ図:AO に引き継がれます。

描画完了後、a1 に属するチャンネル CO により再描画が実施される場合、a1 の子孫であるイメージ図、周波数表示、ライン表示、刺激表示が再描画されます。

イメージ図を再描画する際、イメージ図が持つ curdata のうち、チャンネル値のみが更新されイメージが再描画されます。

a12 に属する kind 設定 CO により再描画される場合は、イメージ図のみが再描画されます。この時、イメージ図が持つ curdata のうち kind のみが更新されイメージが再描画されます。

3.3. Layout Editor 概要

3.3.1. 概要

Layout Editor は 2 つの画面を用いて Layout を編集します。

最初の画面は Layout Editor メイン画面です。

ここで基本的なレイアウトの編集を行います。

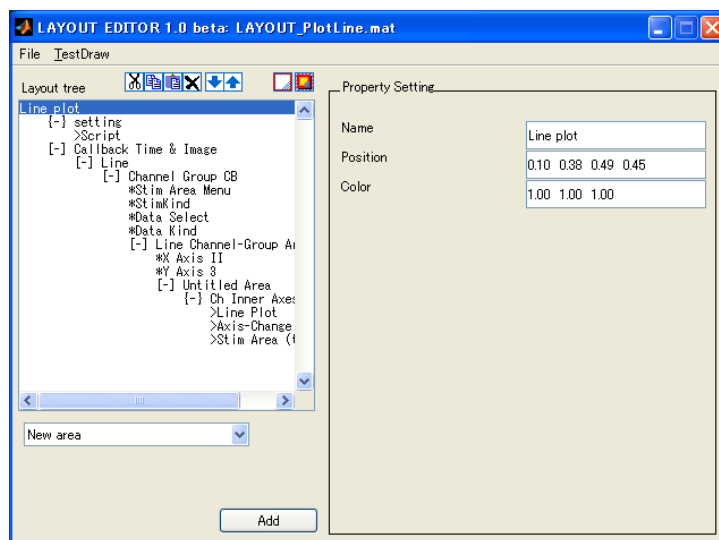


図 3.3 Layout Editor メイン画面

もうひとつの画面は Layout Overview 画面です。ここではメイン画面だけではイメージしにくい概略図の確認を行います。また、配置の調整を行うことも可能です。

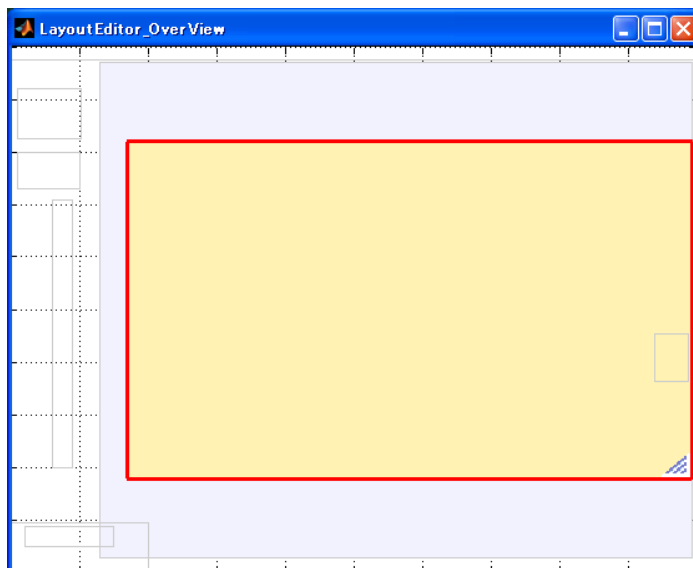


図 3.4 Layout Overview 画面

3.3.2. 起動

Layout Editor の起動方法を説明します。

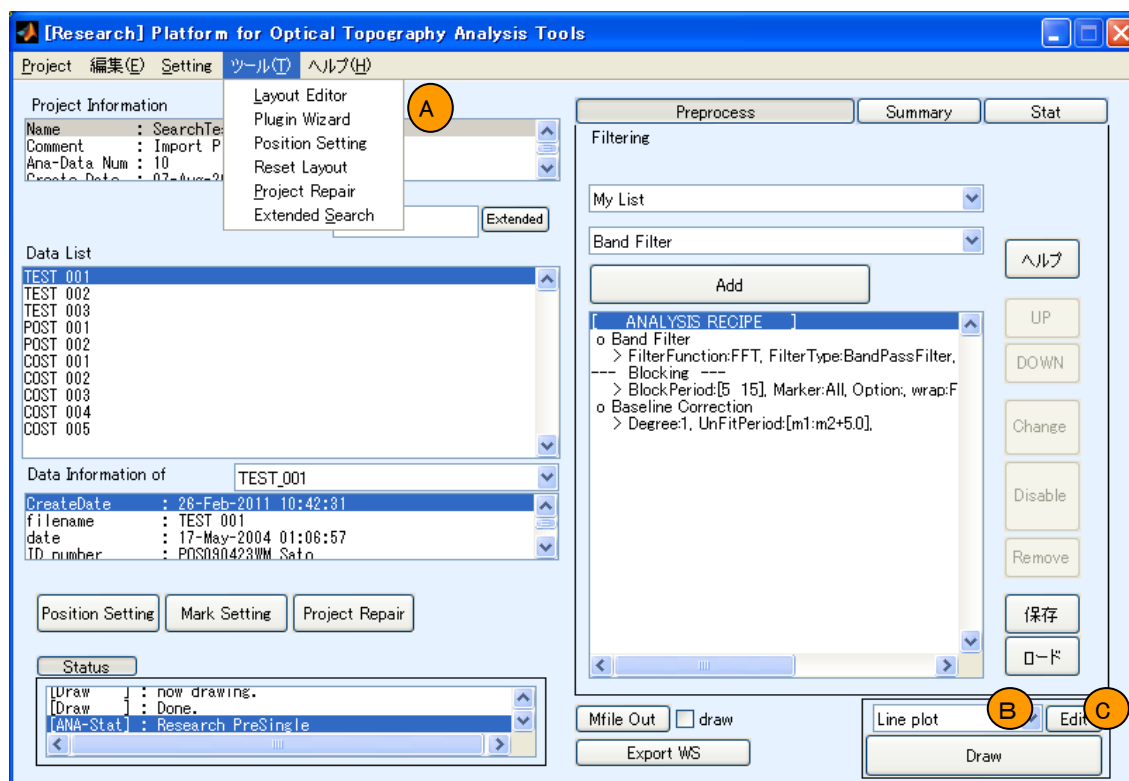


図 3.5 Normal モード メインウィンドウと領域

新に Layout を作成する場合や Layout ファイルを手動で選択する場合、メニューから Layout-Editor を起動します。この時、POTATo メインウィンドウのツールメニューから Layout Editor (A) を選択してください。

POTATo で管理している Layout を編集する場合、Layout Edit ボタン (C) を押下します。そうすると LAYOUT ポップアップメニュー (B) で選択されている Layout を編集します。

3.3.3. ファイル I/O

Layout Editor が起動するとメイン画面には File メニューが存在します。File メニューは、LAYOUT ファイルの操作として、New・Open 等の基本的な処理を提供します。File メニューを下表に示します。

表 3.1 ファイルメニュー一覧

ファイルメニュー	内容
New	新規に LAYOUT を作成する
Open	既存 LAYOUT のオープン
Save	編集 LAYOUT の保存
SaveAs	編集 LAYOUT を別名で保存
Close	LAYOUT のクローズ

3.3.4. テスト表示

作成した Layout の表示結果を手早くみたい場合はメイン画面にある TestDraw メニューを選択します。そうするとこの Layout を用いてサンプルデータの表示が行われます。

なお、サンプルデータは小さなデータで多くの情報が欠落しています。そのため、一部の AO では期待した値が表示されないことがあります。サンプルデータを変更したい場合は

LayoutEdit¥TestData¥data0.mat

を変更してください。

3.4. LAYOUT ツリー

3.4.1. LAYOUT ツリーの表示

Layout は Figure, Area, Axis-Area および, CO, AO の 5 つの構成要素からなります。また、LAYOUT は Figure をルートとし、Area, Axis-Area をリーフとする木構造で出来ています。

これをテキストで表現したものが、Layout ツリーリストボックス **(A)** に示されています。

Figure は先頭に1つのみ存在し、Layout 名を表示しています。右図では”Line plot”が Layout 名です。Figure の子である Area や Axis-Area はひとつ下の階層で示されます。

Area は [+], [-] または [] で始まる文字列で示されます。図では”[-] Callback Time & Image”や”[-] Channel Group C8”等が Area です。

Area を構成する CO は”*”で始まる文字列で示されます。図では”*Stim Area Menu”等が CO です。

Axis-Area は [+], [-] または [] で始まる文字列で示されます。図では”[-] setting”等が Axis-Area です。

Axis-Area を構成する AO は”>”で始まる文字列で示されます。図では”>Script”等が AO です。

Area や Axis-Area はダブルクリックすることによって展開、もしくは折りたたむことができます。Area の場合、展開表示されている場合は “[-]”、折りたたまれている場合は “[+]” と表示され、子がない場合は “[]” と表示されます。

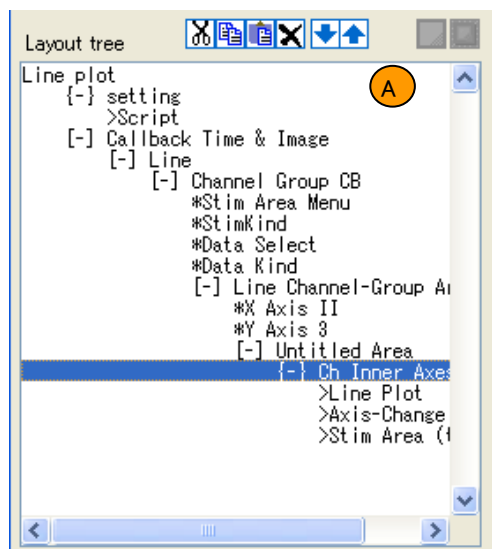





図 3.6 Layout ツリー


3.4.2. ツリー操作


LAYOUT ツリーは操作ボタン(A)もしくは、リストボックスを右クリックした際に開くコンテキストメニュー(B)を実行することで以下の操作が可能です。


カットボタン、もしくは切り取りメニューにより選択中の構成要素をカットできます。選択した構成要素に付随する構成要素（子や C0, A0 など）があれば併せてカットされます。

コピーボタン、もしくはコピーメニューにより選択中の構成要素をコピーできます。選択した構成要素に付随する構成要素（子や C0, A0 など）があれば併せてコピーされます。

ペーストボタンもしくは貼り付けメニューによりコピーもしくはカットにて保存されたデータを貼り付けます。

削除ボタン、もしくは Delete メニューにより選択中の構成要素を削除できます。選択した構成要素に付随する構成要素（子や C0, A0 など）があれば併せて削除されます。

UP ボタンもしくは UP メニューにより選択中の構成要素を上に移動します。移動は同一エリア内に限りますので、他のエリアへ移動する場合はカット＆ペーストを利用ください。

Down ボタンもしくは Down メニューにより選択中の構成要素を下に移動します。移動は同一エリア内に限りますので、他のエリアへ移動する場合はカット＆ペーストを利用ください。

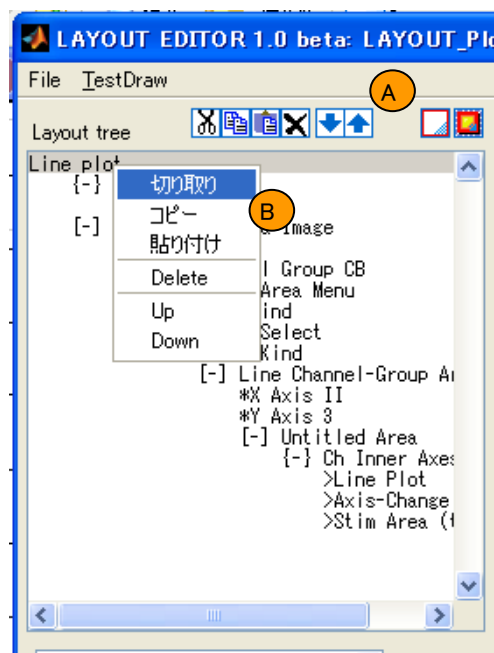


図 3.7 Layout ツリー

3.5. LAYOUT Overview

Layout Overview は描画後の Figure の確認のため、概略図を示します。また構成物の位置の変更を行います。

選択中の構成要素は赤い線で示されます。

選択中の構成要素の親となる Area が青い点線で示され、内部は薄い青色で塗られます。ただし、親が Figure の場合は白色になります。

親 Area 内の CO は緑色の点線で示され、内部は薄い緑色で示されます。

親 Area 内の Area、Axis-Area は青い点線で示されます。また AO は Axis-Area 内部をベージュ色にすることで示されます。

サンプルを図で示します

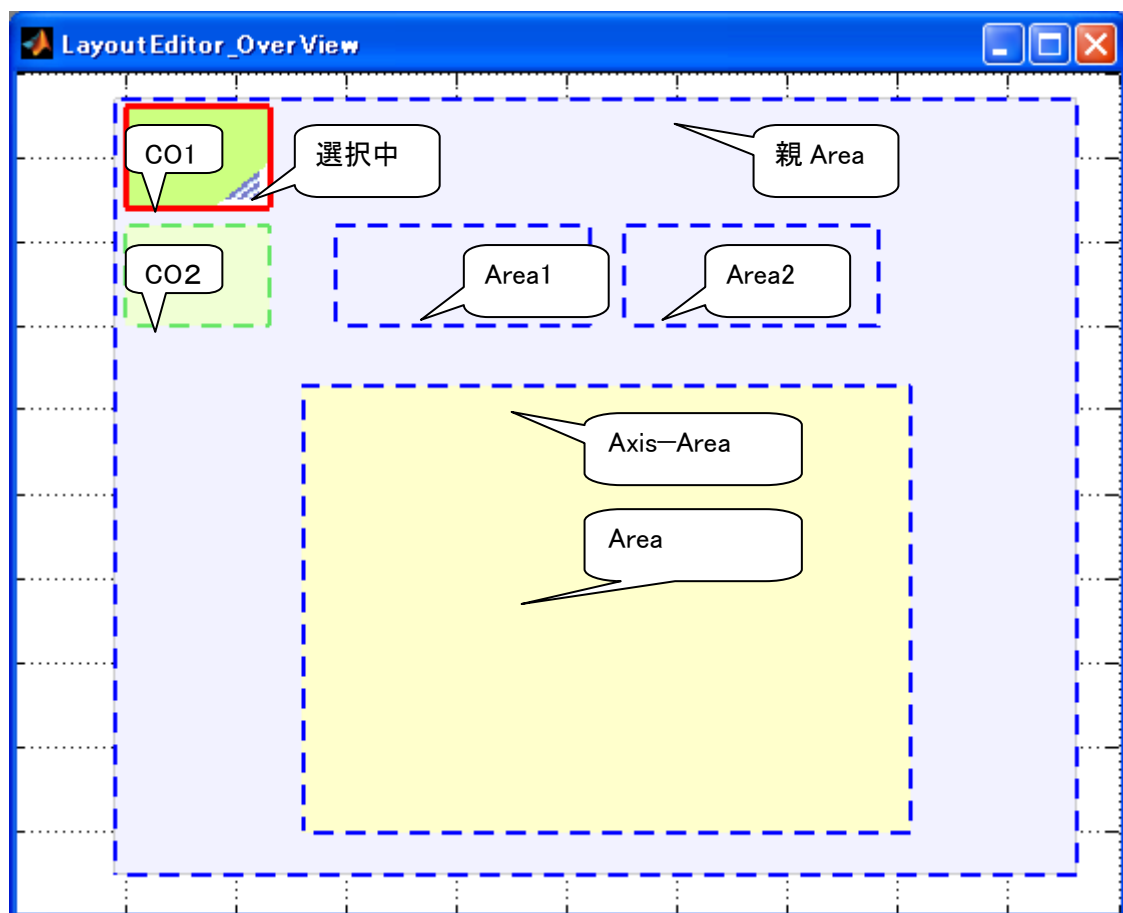


図 3.8 Layout Editor メイン画面

選択中の構成物は枠をドラッグすることにより移動できます。また、右隅をドラッグすることでサイズを変更できます。

3.6. Figureの設定

Layout の先頭には必ず Figure が入ります。Figureは子として Area および Axis-Area を持てます。

Figure では LAYOUT 名、Figure のサイズ、色を設定できます。

Name **(A)** はレイアウト選択ポップアップメニューのレイアウト名として使われます。また、Figure の Name プロパティに設定されます。

Position **(B)** は Figure の位置を示します。位置の単位系は 'Normalized' で、左、下、幅、高さの4つのデータで設定します。

Color **(C)** は Figure の背景色になります。値の設定には uisetcolor 関数が参考になります。

Figure の子の追加はポップアップメニュー **(D)** から Area もしくは Axis-Area を選択し、

Add ボタン **(E)** を押してください。

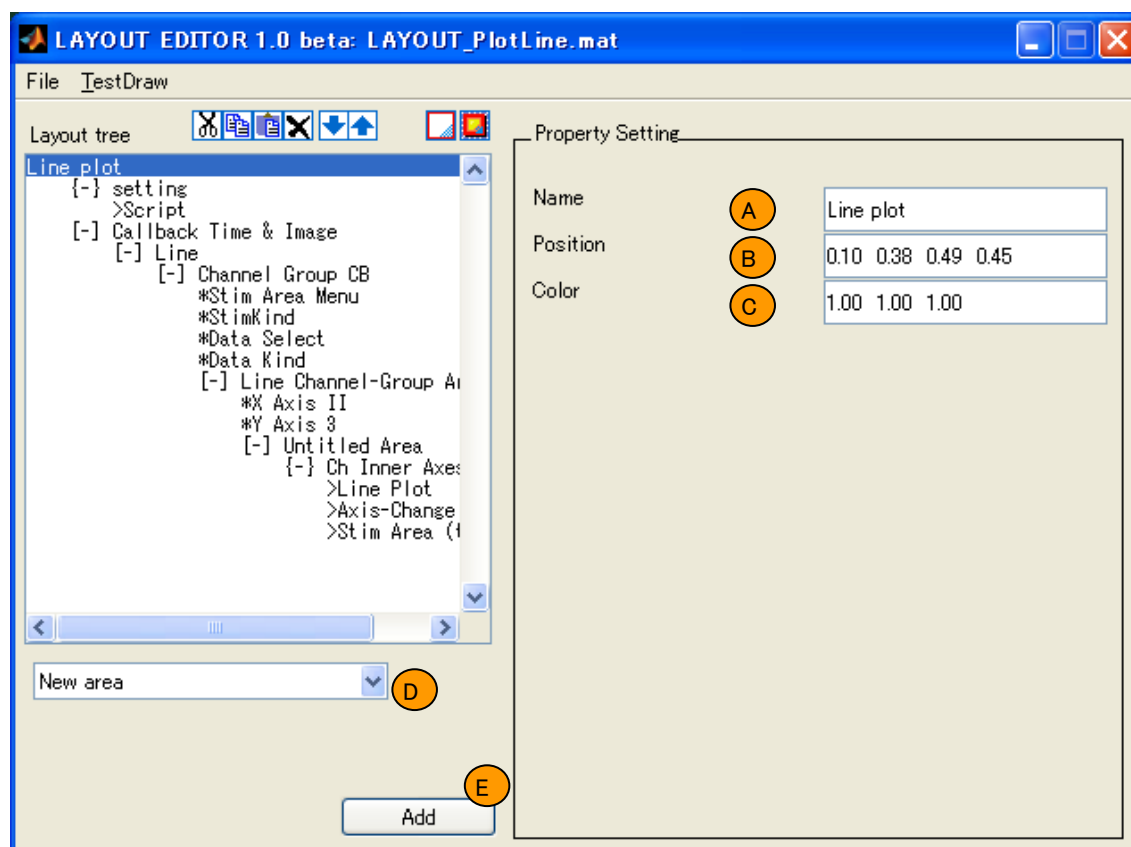


図 3.9 Layout Editor メイン画面

3.7. Area の設定

Area は Figure もしくは Area の子として作成され、子として Area および Axis-Area を持てます。また構成物として CO を持てます。

これらを追加するにはポップアップメニュー(A)から area, axis, special control を選択し、Add ボタン(B)を押してください。なお、special control は一部の CO を除く CO です。

Area の設定には Primary, Variables, Others の3つの設定があります。

Primary は Area の配置や位置を設定します。

また Variable では良く使われる CO として、データを選択、チャンネルの選択、データの種類の選択に関する CO の設定を行います。

Others ではその他の curdata の変更を行います。

3.7.1. Area の Primary 設定

Area の Primary 設定では Area の基本的な設定を行います。

最初に配置方法リストボックス(A)で通常の "Simple Area" か "Channel Order Area" を選びます。Simple Area は通常の Area で、"Channel Order Area" は Area の子の部分をチャンネル数分コピーし、作成・配置します。

このとき配置方法をリストボックス(B)から選びます。配置方法は通常 "Normal" を選びますが、"Array(Square)" により配列のように並べることも可能です。

また Area の名前を Name(C)に指定します。

位置は親からみた相対位置(D)もしくは Figure 上での絶対位置(E)を "normalized" 単位で指定します。

また Area 内の以降の線に関するプロパティ設定を行いたい場合は Line Property チェックボックス(F)を有効にします。なお、Line Property は curdata として渡されるため、上書きされる場合があります。

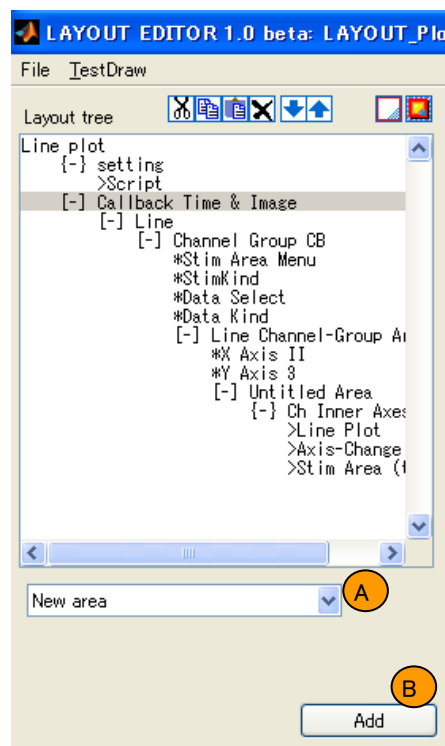


図 3.10 Area 設定

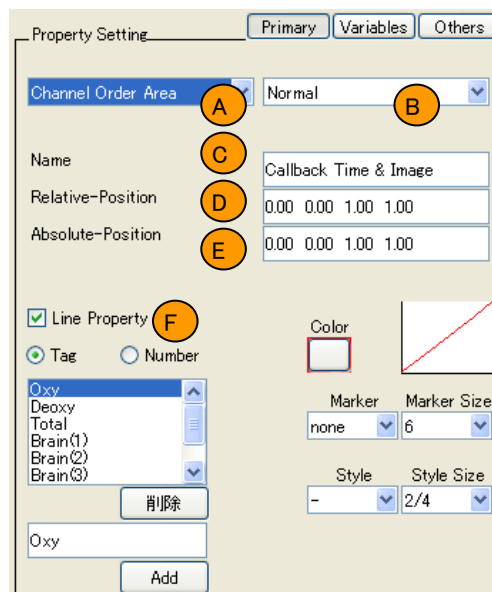


図 3.11 Area Primary 設定

3.7.2. Area の Variables 設定

Variable では良く使われる CO として、データの選択、チャンネルの選択、データの種類の選択に関する CO の設定を行います。

最初に Area の Variable トグルボタンを押下状態にします。

Variable 設定では各行に CO の種類が、列に設定値が表示されます。

CO の種類は左の文字列(A)に記載されています。文字列の右側にあるチェックボックス(B)で CO の有効/無効を設定します。

有効にしたあと、GUI の表示スタイル(C)を選択します。デフォルトは”None”です。”None”の場合は CO を作成せず、描画時に curdata を変更します。

”None”以外のスタイルが設定されると、CO が有効になり、LAYOUT ツリーに追加されます。

また、位置を持つスタイルが選択されると相対位置の入力ボックス(D)が表示されますので、所属する Area 内での相対位置を記載します。

最後に初期値を Set ボタン(E)で設定します。

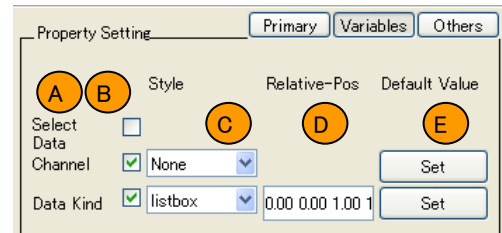


図 3.12 Area Variable 設定

3.7.3. Area の Others 設定

Area の Others トグルボタンを押下状態にするとスクリプトの設定画面が開きます。
ここでは、スクリプトを実行することにより、詳細な設定を行います。
スクリプトで利用/変更できる変数は curdata 構造体です。curdata 構造体のフィールドは任意に追加できますが、主要なフィールドは以下の通りです。

表 3.2 curdata 構造体の主要フィールド

フィールド名	内容	例
region	解析データの種類 (Continuous/Block/Summary)	‘Continuous’
cidmax	連続データの数	1
bidmax	ブロック数	2
time	時刻範囲	[-Inf Inf]
ch	チャンネル番号	1
kind	データの種類	[1 2]
gcf	描画中, figure ハンドル	2
menu_current	メニューハンドル	10.01
path	Curdata 所有構成物の LAYOUT 内の構成要素位置	[1 2 2]
LineProperty	線のプロパティ (通常 GUI で設定)	-
CommonCallback-	共通 CO 参照用データ	-
Data		

次のようなレイアウトを例にスクリプト作成例を示します。

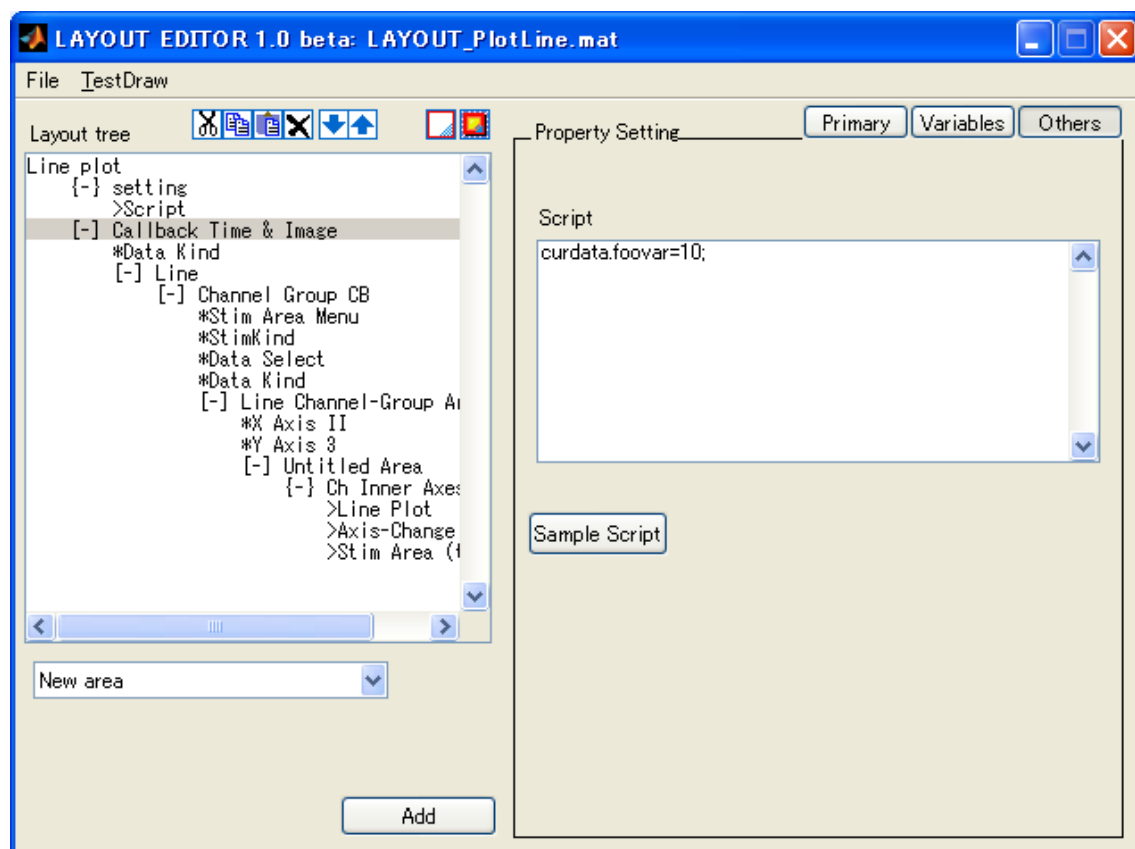


図 3.13 スクリプト例

Area, A1 のスクリプトに”curdata.foovar=10”と記述します。

このとき、A1 の子孫である AA11 とその Script、AA12 とその Script に curdata.foovar の値が引き継がれます。

また A1 の子ではない A2 では curdata.foovar は存在しません。

3.8. CO: Control-Object

CO は Area の構成物として作成され、子を持ちません。

CO を Layout ツリーで選択すると編集画面が表示されます。

編集画面では、CO の位置と固有のパラメータを設定します。

位置は親からみた相対位置(A)もしくは Figure 上での絶対位置(B)を”normalized”単位で指定します。

設定した CO 固有パラメータはリストボックス(C)に表示されますが、変更したい場合は Modify ボタン(D)を押します。

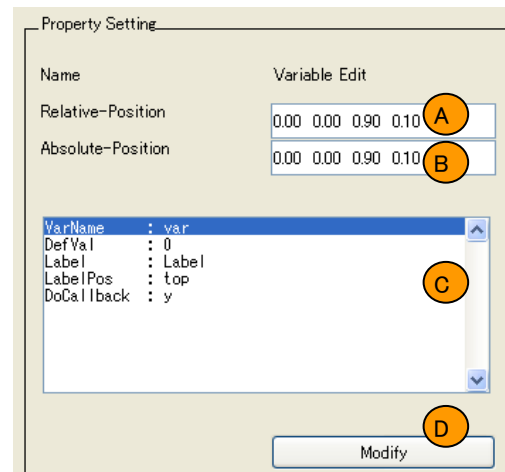


図 3.14 CO 設定

3.9. Axis-Area の設定

Axis-Areaは Figure もしくは Area の子として作成されます。子は持ちませんが、構成物として AO を持ちます。

Axis-Area の設定は名称と位置のみです。

Axis-Area の名称はエディットテキスト(A)に記入します。

位置は親からみた相対位置(B)もしくは Figure 上での絶対位置(C)を”normalized”単位で指定します。

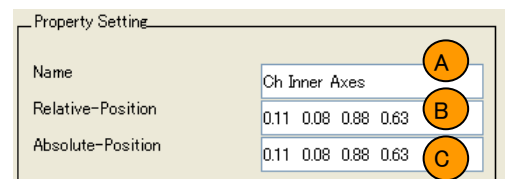


図 3.15 Axis-Area 設定

Axis 内に AO を追加するには、画面左下の AO ポップアップメニュー(A)から AO を選択し、Add ボタン(B)を押してください。

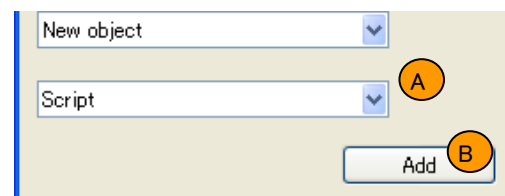


図 3.16 AO の追加

3.10. AO: Axis-Object 設定

AO は Area の構成物として作成され、子を持ちません。

AO を Layout ツリーで選択すると編集画面が表示されます。

編集画面では、AO の固有のパラメータを設定します。

設定済みの AO 固有パラメータがリストボックス(A)に表示されます。この値を、変更したい場合、Modify ボタン(B)を押します。

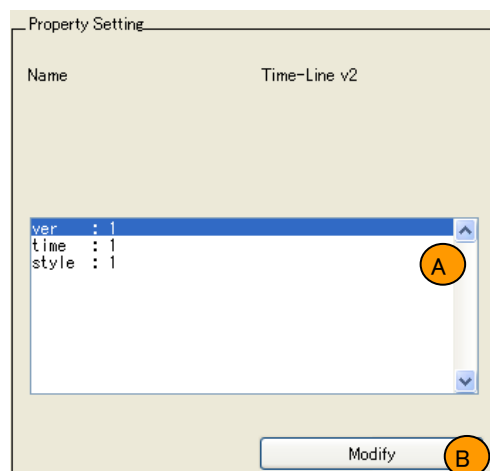


図 3.17 CO 設定

4. 表示機能拡張

4.1. 説明内容

Layout の構成要素である AO, CO を新に作成することにより、POTATo の表示機能を拡張する方法を説明します。ここではプログラムコードの作成を前提としていますので、プログラムサイドからの説明になります。

作成のための予備知識として、POTATo における表示処理について説明します。ここで、3.2 LAYOUT の構造とデータの引継ぎに関しては再度説明しません。

ここでは典型的な AO, CO について説明します。一部の AO, CO の動作と異なる場合があります。

4.2. 表示機能と LAYOUT

表示処理において、LAYOUT の構成物の CO, AO の相互作用に注目して説明します。ここで相互作用の説明にはUMLにおけるシーケンス図を用います。

最初に簡単に今回の表記で用いているシーケンス図について説明します。

シーケンス図の例を右図に記載します。

図の縦軸は時間で、上から下に流れます。

また考慮するオブジェクト(A)を四角と点線で示しています。点線は存在する期間を示しており、削除される時点を×で示します。ここで言うオブジェクトは各種ウィンドウや CO や AO です。CO, AO は MATLAB 上では単なる特定の構造を持つデータで、そのひとつひとつをオブジェクトと言います。オブジェクトは、特定の関数(およびそのサブ関数)を用いて操作します。

図ではこれら操作をメッセージ(B)として記載します。特に注目している引数がある場合のみ括弧内に引数を記載しています。メッセージを受け、オブジェクト内部に正しく初期化されたデータを持つようになった時、状態不変式(C)にそのデータを示しています。

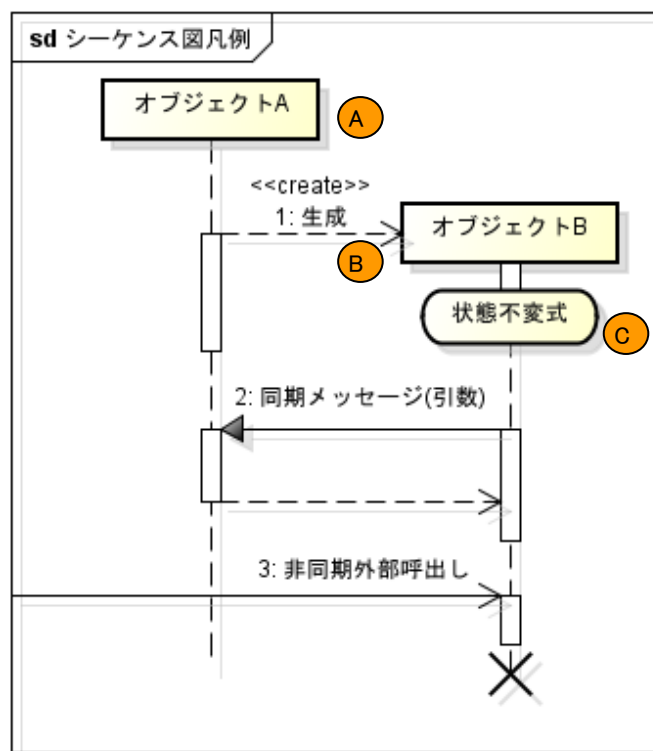


図 4.1 CO 設定

描画時の CO, AO の主要なシーケンスを示します。

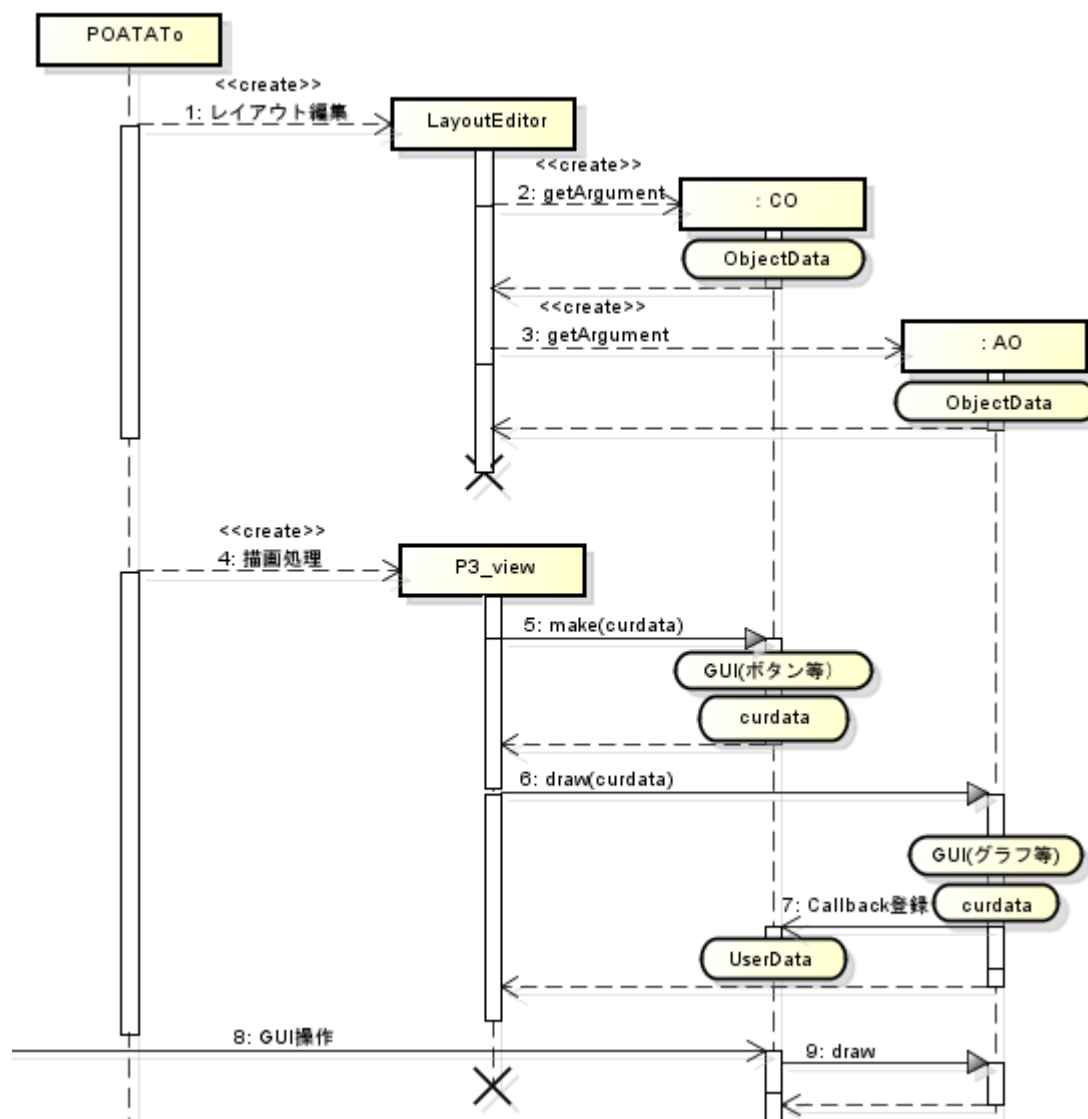


図 4.2 描画シーケンス

描画する際、POTATo からみると3つの状態があります。1つは Layout Editor で Layout を作成している状態、もうひとつは P3_view 関数で描画を実行している状態、そして最後は描画後、コントロールにより図の再描画等を行っている状態です。

最初に、POTATo により 1.Layout Editor が起動され、Layout の作成状態になります。

Layout Editor から CO の 2.getArgument サブ関数が呼び出され CO が生成されます。このとき CO は ObjectData として作成され Layout に保存されます。

同様に Layout Editor から AO の 3.getArgument サブ関数が呼び出され AO が生成されます。

ここで実際の編集作業では CO, AO の生成順序、生成数に制限はありません。また、削除や変更などのメッセージは省略しています。

Layout が作成されると、POTATo により 4 描画処理が実行されます。この時、P3_View により処理が行われます。描画処理中、LAYOUT 構成要素である Area, Axis-Area はそれぞれ自分の curdata を保持し、curdata を利用・変更し、親から子へ curdata を引き継いでいきます。

CO 描画時、描画処理(P3_View)は CO に対して CO の属する Area の curdata を引数とし 5. make を実施します。この時 CO はボタン等の GUI を作成し、curdata を内部に保持します。また、上位 Area の curdata を更新します。

AO 描画時、描画処理(P3_View)は AO に対して AO の属する Axis-Area の curdata を引数とし 6. draw を実施します。この時、AO はグラフを描画などにより GUI を作成し、curdata を保持します。また、先祖に Callback を受けたい CO が居る場合、その CO に 7.Callback してもらうようデータを登録します。登録されたデータは CO 内に UserData として保存されます。

最後に CO が作成した GUI がユーザ等により 8 操作された際、CO は登録されている AO の持つ curdata を書き換え 9. draw 関数を実行します。

なお、ここでの説明は CO, AO の相互作用を主眼にしたため、POTATo 内部で行う描画管理処理に関しては大幅に省略しています。省略しているものは Area, Axis-Area に関する処理や、CO, AO の基本情報の参照メソッド createBasicInfo や、描画処理を行うための処理を drawstr メソッドです。また、ObjectData や curdata 引継ぎ、保存方法に関しても省略しています。

4.3. データ構造

CO, AO を拡張する上で必要なデータ構造を説明します。LAYOUT の構造と curdata に関しては 3.2 LAYOUT の構造とデータの引継ぎも参照ください。

なおデータ構造は、LAYOUT 編集時、描画中、GUI による Callback 中の 3つの状態の影響を受けます。

4.3.1. POTATo データ

描画中の POTATo の解析結果として渡された、連続データや区間データおよび要約統計量は、Figure の Application Data として保存されます。なお、存在しないデータは空 ()になります。

表 4.1 POTATo データ保存

データ名	内容	関連 curdata
CHDATA	連続データ(ヘッダ)	curdata.region='Continuous'
CDATA	連続データ	curdata.cid0
BHDATA	区間データ(ヘッダ)	curdata.region='Block'
BDATA	区間データ	curdata.stimkind curdata.flag.MarkAveraging
SSHDATA	要約統計量(ヘッダ)	curdata.region='Summary'

SSDATA

統計量

現在、注目している描画対象のデータは curdata により記載されます。

データの種別は curdata の.region で指定されます。

また、連続データが複数ある場合、cid0 に連続データの通し番号が、区間データの場合は stimkind に注目中の刺激の種別が指定されます。

データの取得には p3_LayoutViewTool の 'getCurrentData' が利用できます。

4.3.2. ObjectData

ObjectData は AO, CO が持つサブ関数 getArgument により作成されるデータで、利用関数名や引数などが格納されます。

AO の ObjectData は LAYOUT 編集時に getArgument により作成され LAYOUT 内の AxisArea の構成要素(Object)に保存されます。

描画中、AxisArea 内では obj[idx]として参照可能です。通常、AO のサブ関数 drawstr を通して参照され、AO のサブ関数 draw の引数 objdata として渡ってきます。また、draw 中に描画中の AO に対応する Application Data として保存されます。

GUI による Callback 中、ObjectData は CO 内で Application Data から呼び出され、AO のサブ関数 draw に渡されます。

なお、Application Data との I/O は p3_ViewCommonCallback 関数が利用できます。AO:描画処理終了時に、"checkin"で保存し、CO は"getData"で呼び出します。また、AO 再描画時は"update"を用いてデータを更新します。詳細は補助関数を参考ください。

ObjectData の例を示します。なお AO のサブ関数 getArgument 関数に依存します。

表 4.2 AO: ObjectData 例

フィールド名	内容	例
str	表示名	'3D_BrainSurf'
fnc	AO 関数名	'LAYOUT_AO_3DBrainSurf'
ver	Version:内部管理用	1
(Arg1)	引数など	4

CO の ObjectData は LAYOUT 編集時に getArgument により作成され、LAYOUT 内の Area の構成要素(CObject)として保存されます。

描画中、Area 内で cobj[idx]として参照可能です。通常、CO のサブ関数 drawstr を通して参照され、AO のサブ関数 make の引数 obj として渡ってきます。obj データは通常 make 後に破棄されますが、CO によっては作成 uicontrol 内の 'UserData' に保存されることがあります。

GUI による Callback 中、ObjectData は通常利用しません。

ObjectData の例を示します。なお CO のサブ関数 getArgument 関数に依存します。

表 4.3 CO: ObjectData 例

フィールド名	内容	例
name	表示名	'Time Point'
fnc	AO 関数名	'LAYOUT_CCO_TimePoint',
(pos)	相対位置 [x, y, width, height] (x, y)は左下位置。	[0.6 0.1 0.3 0.08]
(Arg1)	引数など	4

4.3.3. AO:Draw 処理と関連データ

AO のサブ関数 draw を実施する上で必要なその他のデータについて説明します。

関連するハンドルとして、AO を描画する Axes ハンドルが入力として渡ってきます。また、Draw により作成されたハンドル群を出力します。ハンドル群は再描画前に削除するために利用されます。

また、AO は ObjectID を持ちます。AO の Draw 処理に必要な情報の多くは Application Data として保存しますが、このデータを取得する際に利用します。なお、ObjectID は通常 p3_ViewCommonCallback 関数を用いて発行します。

4.3.4. CO:Callback 処理と関連データ

一般的な CO が GUI よりコールバックされた時に必要なデータについて説明します。

GUI からコールされた時の引数は対象 GUI のハンドルです。実行に必要なデータはそのハンドル内の UserData プロパティに格納されています。

UserData はセル配列で定義されており、Callback 登録されている AO の一覧が入っています。ただし、描画時に ObjectData を保存している場合、UserData が 1 番目のセルに格納されている場合があります。

CO の UserData 内に Callback 登録されているデータの変数名は多くの場合 ud と表現されています。ud は以下のデータを持ちます。

表 4.4 CO: Callback 登録データ(UserData 内)

フィールド名	内容	例
axes	対象 Axes のハンドル	347.0016
ObjectID	AO の ObjectID	1
name	AO の名称	'Layout_AO_TimeLine_ObjectData'
str	Callback するための文字列	'Layout_AO_TimeLine_ObjectData('draw')', ...

この ud を用いると Application Data に保存している AO の Draw 処理に関する情報(AO 関連データ)が取得できます。取得は以下の様に補助関数 p3_ViewCommonCallback を利用します。

```
data=p3_ViewCommonCallback( 'getData',ud.axes, ud.name, ud.ObjectID);
```

関数を実行すると ud.axes をカレント Axes に変更し、Application-Data から name, ObjectID に対応した data を取り出します。ここで data の構造は以下のようになっています。

表 4.5 CO: AO 関連データ(data)

フィールド名	内容	主な用途
handles	AO が作成したハンドル	描画前に削除する
axes	AO の属する Axes	描画 (ud.str)
curdata	AO 内部の curdata	更新および描画
obj	AO の ObjectData	描画

4.4. 補助関数

AO, CO を作成中に利用する補助関数を説明します。

表 4.6 補助関数リスト

関数名	サブ関数名	内容
p3_LayoutViewerTool	getCurrentData	描画対象の POTATo データを取得する
p3_ViewCommonCalck	checkin	AO 関連データを保存し、 Common CO に登録する
	update	AO 関連データを更新する
	getdata	AO 関連データを取得する

4.4.1. POTATo データの取得

curdata に対応する POTATo データの取得のためには以下の関数を利用します。

シンタックス	[hdata, data] = p3_LayoutViewerTool('getCurrentData', fh, curdata);	
機能	描画中の curdata に対応する POTATo データを取得する。	
入力	fh	Figure ハンドル(描画中のもの)
	curdata	curdata
出力	hdata	POTATo データ(ヘッダ)
	data	POTATo データ

関数は curdata.region をみて、対象となる POTATo データを取得します。region が "Block" の場合は通常、curdata.flag.MarkAveraging = false の時を除き data はグランドアベレージ後の data になり 3次元になります。また、対象データが無いときは空行列が帰ってきます。

Application Data から直接 POTATo データを取得する場合やその他の関連する curdata に関しては 4.2 POTATo データをご参照ください。

4.4.2. AO 関連データの I/O

AO 関連データを保存し、Common-CO に Callback 登録するには以下の関数を使います。

シンタックス ID= p3_ViewCommonCalck (‘checkin’ ,
h,name, ah, curdata, obj);

機能	AO 関連データを保存し、Common-CO に Callback 登録する	
入力	h	AO が作成したハンドルの配列 (再描画時に書き換えるもののみ)
	name	Application Data に登録する名前 (AO 固有の名前であること)
	ah	AO の親となる axes
	curdata	curdata
	obj	AO の ObjectData
出力	ID	AO の ObjectID

checkin されたデータは AO 関連データ(表 4.5 CO: AO 関連データ(data))に変更されます。

この AO 関連データは AO 固有の名前”name”の Application データ内にあるセル配列の末尾に追加されます。

このとき、セル配列を参照するための番号が ObjectID として返されます。

checkin された AO 関連データを更新します。

シンタックス p3_ViewCommonCalck (‘update’ ,
h,name, ah, curdata, obj,
ID);

機能	指定した name, ID に対応する AO 関連データを更新する	
入力	h	AO が作成したハンドルの配列
	name	Application Data に登録する名前
	ah	AO の親となる axes
	curdata	curdata
	obj	AO の ObjectData
	ID	AO の ObjectID

保存した AO 関連データを読み出すには以下の関数を使います。

シンタックス	data=p3_ViewCommonCallback('getData' ,ah, name, ID)	
機能	AO 関連データを保存し、Common-CO に Callback 登録する	
入力	ah	AO の親となる axes
	name	Application Data に登録する名前
	ID	AO の ObjectID
出力	data	AO 関連データ

通常 CO から呼び出されますが、実行に必要なデータは UserData(表 4.4 CO: Callback 登録データ(UserData 内))に保存されています。

4.5. AO: Axis-Object の作成

4.5.1. AO 関数インタフェース

AO の処理は POTATo 内の LAYOUT/AxisObjectフォルダ内に作成された LAYOUT_AO_*.m に記述します。この関数は以下のようなインタフェースを持ちます。

LAYOUT_AO_*('subfname' ,[arg1, arg2,...])

ここで subfname にサブ関数名が入り、arg1, arg2,...はサブ関数の引数です。

用意すべきサブ関数は以下になります。

表 4.7 AO:サブ関数

サブ関数名	内容
createBasicInfo	基本情報設定
getArgument	ObjectData を設定する
drawstr	描画時の実行方法を提供する
(draw)	描画処理

それぞれのサブ関数の引数や用途は決まっており、ここでは各サブ関数について説明します。
なお、これらの関数の骨格となるコードは、他の AO をコピーするか、“P3_wizard_plugin“の Viewer Axis-Object にて作成できます。

4.5.2. createBasicInfo

編集時や描画中の基本情報を設定します。

シンタックス	info=createBasicInfo	
機能	指定した name, ID に対応する AO 関連データを更新する	
出力	info	基本情報(構造体)

ここで、基本情報構造体は以下のフォーマットです。

表 4.8 AO:基本情報

フィールド名	内容	例
MODENAME	AO の名前	'2D Image (2.0)';
fnc	AO の関数名	mfilename
ver	バージョン	2.0
ccb	Callback 登録する Common-CO のリスト	{'Data', 'DataKind', 'stimkind'};

ccb は p3_ViewCommonCalck /checkin で Common-CO に対して Callback 登録を実施するかどうかの判定をするさいに使います。ccb が 'all' の場合は全て Callback 登録します。

'all' 設定は容易ですが、描画に時間のかかる AO は不必要なタイミングで再描画されないよう、CO を限定すべきです。

4.5.3. getArgument

描画処理を実施するための引数設定を行います。

シンタックス	obj=getArgument(obj)	
機能	AO: 描画のための引数を設定する	
入出力	obj	AO の ObjectData(表 4.2 AO: ObjectData 例)

Layout Editor から呼び出されます。

新規作成時、obj は空白になり、更新時、obj は以前の obj が設定されます。

キャンセルする場合は return 前に obj=[];と設定します。

4.5.4. drawstr, draw

描画処理を実施するための文字列を渡します。文字列は Axis-Area 内の AO 描画処理で実行されますので、変数のスコープには注意が必要です。

シンタックス	str = drawstr(varargin)	
機能	AO: 描画処理のための文字列作成	
入力	varargin	POTATo 3.7 以降, varargin[1] に ObjectData が入ります
出力	str	Axis-Area 内の AO 描画処理で実行する文字列

通常、以下のように記載します。

```
function str = drawstr(varargin) %#ok
str=[mfilename, ' (' 'draw' ', h.axes, curdata, obj{idx})'];
```

上記の記載を前提に draw 処理を説明します。

シンタックス	hout=draw(gca0, curdata, objdata, ObjectID)	
機能	AO: 描画処理	
入力	gca0	AO の親となる axes
	curdata	curdata
	objdata	AO の ObjectData
	ObjectID	AO の ObjectID (ただし、再描画の時のみ)
出力	Hout	AO が作成したハンドルとタグを記載した構造体

ここで hout は hout.h にハンドルの配列、hout.tag にタグのセル配列を作成します。

ここで draw 処理の一般的な内容について説明します。

POTATo データを利用する場合は以下の一文を追加し POTATo データを取得します。

```
[hdata, data]=osp_LayoutViewerTool(...
    'getCurrentData', curdata.gcf, curdata);
```

次に、グラフを描画します。例えば以下の様なコードになります。

```
h.h = surf(peaks(10)); % ハンドルを設定します(必須)
h.tag = { 'test' }; % タグを設定します
```

実際にはこのグラフ描画が AO のコア部分です。

また、Application Data I/O および Common CO に Callback を呼び出すよう登録します。

```
%=====
%=      Common-CO への Callback 登録      =
%=====
myName=' LAYOUT_AO_hoge' ;                % Application Dataに登録する名前
if exist(' ObjectID', ' var'),
    % 再描画時
    p3_ViewCommCallback(' Update', ...
        h.h, myName, ...
        gca0, curdata, obj, ObjectID);
    return;    % Update して終了
else
    % Application Data の登録、Callbak 登録
    ObjectID = p3_ViewCommCallback(' CheckIn', ...
        h.h, myName, ...
        gca0, curdata, obj); % ObjectID 取得
end
```

最後に、Common CO 以外の CO に Callback 登録します。以下は Callback の必要がある場合のみ行います。なお、Callback の登録の方法は CO により異なりますが、典型的な CO は同じです。

最初に登録用データとして UserData(表 4.4 CO: Callback 登録データ(UserData 内))を作成します。

次に、curdata 内にある対象の CO に UserData を追記します。

```
% =====
% =                                Callback の登録                                =
% =====
%   Callback 登録用データの作成
udadd.axes      = gca0;
udadd.ObjectID = ObjectID; % Common CO の登録で発行された ID
udadd.name      = myName;   % Common CO の登録時と同じ
udadd.str       = [objdata.fnc,...
    ' ('draw',data.axes, data.curdata, data.obj, ud.ObjectID);'];

%-----
%   Callback 登録 ( XX)
%-----
% CO_XX が存在するかどうかのチェック
if isfield(curdata,'Callback_XX') && ...
    isfield(curdata.Callback_XX,'handles') && ...
    ishandle(curdata.Callback_XX.handles),
    % See also LAYOUT_CO_XX
    % ハンドル取得
    h = curdata.Callback_XX.handles;
    % UD の追加
    ud=get(h,'UserData'); % 現状の ud 取得
    if isempty(ud)
        ud = {udadd};
    else
        ud{end+1}=udadd;
    end
    set(h,'UserData',ud); % UD 更新
end
```

4.6. CO: Control-Object の作成

CO の処理は POTATo 内の LAYOUT/ ControlObject フォルダ内に作成された LAYOUT_CO_*.m
もしくは LAYOUT_CCO_*.m に記述します。

ここで、CO は通常の CO, CCO は Common -CO です。

この関数は以下のようなインタフェースを持ちます。

LAYOUT_[C]CO_*('subfname' ,[arg1, arg2,...])

ここで subfname にサブ関数名が入り、arg1, arg2,...はサブ関数の引数です。

用意すべきサブ関数は以下になります。

表 4.9 AO:サブ関数

サブ関数名	内容
createBasicInfo	基本情報設定
getArgument	ObjectData を設定する
drawstr	CO 作成方法を提供する
(make)	CO 作成
(mycallback)	Callback 実施

サブ関数 getDefaultCObject は Layout Editor の Area 内 Variable 設定をする一部の CO には
必須ですが、今回は説明しません。

以下、それぞれのサブ関数の引数や用途は決まっており、ここでは各サブ関数について説明します。

なお、これらの関数の骨格となるコードは、他の CO をコピーするか、以下のコードを利用ください。

```
function varargout=LAYOUT_CO_XX (fcn, varargin)
% ヘルプを記載します
if nargin==0, fcn='help';end
%=====
% Switch by Function
%=====
switch fcn
case {'help','Help','HELP'},
    POTATO_Help(mfilename);
case {'createBasicInfo','drawstr','getArgument'},
    % Basic Information
    varargout{1} = feval(fcn, varargin{:});
case 'make',
    varargout{1} = make(varargin{:});
otherwise
    % Default
    if nargin,
        [varargout{1:nargout}] = feval(fcn, varargin{:});
    else
        feval(fcn, varargin{:});
    end
end
end
%=====
return;
```

4.6.1. createBasicInfo

編集時や描画中の基本情報を設定します。

シンタックス	info=createBasicInfo
機能	指定した name, ID に対応する CO 関連データを更新する
出力	info 基本情報 (構造体)

ここで、基本情報構造体は以下のフォーマットです。

表 4.10 AO:基本情報

フィールド名	内容	例
name	CO の名前	'XX';
fnc	CO の関数名	mfilename
rver	リビジョン	' Revision: 1.1'
date	最終更新日	'Date: 2012/09/01'

rver, date は使用していませんが記載を推奨します。

uicontrol フィールドは Layout Editor の Area 内 Variable 設定をする一部の CO には必須ですが今回は説明しません。

4.6.2. getArgument

描画処理を実施するための引数設定を行います。

シンタックス	obj=getArgument(obj)
機能	CO 作成のための引数を設定します
入出力	obj CO の ObjectData(表 4.3 CO: ObjectData 例)

Layout Editor から呼び出されます。

新規作成時、obj は空白になり、更新時、obj は以前の obj が設定されます。

キャンセルする場合は return 前に obj=[];と設定します。

4.6.3. drawstr, make

CO 描画処理を実施するための文字列を渡します。文字列は Axis-Area 内の CO 描画処理で実行されますので、変数のスコープには注意が必要です。

シンタックス	str = drawstr(varargin)	
機能	CO 作成のための文字列作成	
入力	varargin	POTATo 3.7 以降, varargin{1} に ObjectData が入ります
出力	str	Axis-Area 内の AO 描画処理で実行する文字列

通常、以下のように記載します。

```
function str = drawstr(varargin)
% Execute on ViewGroupCallback 'exe' Function
str=['curdata=' mfilename '('make',handles, abspos,' ...
    'curdata, cobj{idx});'];
return;
```

上記の記載を前提に make 処理を説明します。

シンタックス	curdata=make(hs, apos, curdata,obj)	
機能	AO: 描画処理	
入力	hs	上位ハンドル
	abspos	上位 Area の絶対位置 (Normalized Units)
	curdata	curdata
	objdata	CO の ObjectData
出力	curdata	curdata

ここで make 処理の一般的な内容について説明します。

Common CO の場合は Common CO 用の構造体の基本設定を行う。

```
%=====
% Common-Callback-Data
%=====
CCD.Name          = 'XX';           % CO の名前
CCD.CurDataValue = {'XX','xx'};    % AO 基本情報 ccb の指定名
CCD.handle        = [];            % Callback 登録するハンドル
```

この例では、AO の基本情報(表 4.8 AO:基本情報)内の ccb に、'XX' もしくは 'xx' をもつ AO に限り Callback 登録するよう設定しています。(ただし ccb が all の場合は Callback 登録します。

次に GUI を作成します。

```
h0 = uicontrol;
```

実際に h0 とハンドルを作る必要はないですが、これ以降便宜上 h0 として話しをします。

GUI の位置は相対位置で着ていますので以下のように変更が必要です。

```
pos=getPosabs(obj,pos,apos);

function lpos=getPosabs(lpos,pos)
% Get Absolute position from local-Position
lpos([1,3]) = lpos([1,3])*pos(3);
lpos([2,4]) = lpos([2,4])*pos(4);
lpos(1:2)   = lpos(1:2)+pos(1:2);
```

なお、単位は Normalized ですので位置設定を行う前に uicontrol などのプロパティ 'Units' を 'Normalized' に設定する必要があります。

Callback 設定は次のようにします。

```
set(h0, 'Callback', ...
      [filename ' (' mycallback', gobo)']);
```

ここでは Callback 関数をサブ関数の mycallback と設定しています。サブ関数名は自由ですが、以降 mycallback と設定したものとします。

最後に、make で作成した data を curdata に反映させます。

デフォルト値などで値を変えた場合、

```
curdata.foovar = obj.defaultfoovar; % この CO が foovar を変更
```

また、CO の場合はハンドルを引き継ぎます。

```
curdata.Callback_XX.handles =h0;
```

Common CO の場合は次のようにハンドルを引き継ぎます。

```
CCD.handle =h0;
if isfield(curdata,'CommonCallbackData')
    curdata.CommonCallbackData{end+1}=CCD;
else
    curdata.CommonCallbackData={CCD};
end
```

4.6.4. mycallback

上記の make 処理を前提に mycallback 関数を説明します。

シンタックス	mycallback(h)
機能	CO のコールバック処理
入力	h CO ハンドル

ここで mycallback 処理の一般的な内容について説明します。

最初に再描画する際に変更す値を用意します。

```
foovar = get(h,'Value');
```

この例では、CO は curdata.foovar を変更するものとし、foovar は h の 'Value' で指定されているものとします。

次に Callback 登録されている AO についてループ処理を行い終了します。

```
ud=get(h,'UserData');

for idx=1:length(ud), % for のループ変数名は必ず idx とします。
    <<< 以降はこの内部処理の説明になります>>>
end
return;
```

ここで、ud の内部は表 4.4 CO: Callback 登録データ(UserData 内)に記載しています。
ループ内では最初に AO 関連データ(表 4.5 CO: AO 関連データ(data))を取得します。


```
% Get Data
data = osp_ViewCommCallback('getData', ...
    ud{idx}.axes, ...
    ud{idx}.name, ud{idx}.ObjectID);
```

次に対象 AO の curdata を更新します。

```
% curdata の更新
data.curdata.foovar =foovar;
```

なお、この場合 foovar を更新しています。

再描画の前に以前の AO が書いた図を削除します。

```
% Delete handle
for idxh = 1:length(data.handle),
    try
        if ishandle(data.handle(idxh)),
            delete(data.handle(idxh));
        end
    catch
        warning(lasterr);
    end % Try - Catch
end
```

最後に再描画を実行します。

```
% Evaluate (Draw)
try
    eval(ud{idx}.str);
catch
    warning(lasterr);
end % Try - Catch
```

付録：Script AO の使用方法

概要

ここで、AO のひとつ Script AO の使用例を説明します。

Script AO はスクリプトにより比較的自由に描画を行うための AO です。

AO は入力に2つのスクリプトを持ちます。

ひとつは Axis および同一 Axis 内にある後続の AO に影響を与える curdata を編集するためのスクリプトです。これを Axis 用スクリプトと呼びます。このスクリプトは描画時に1度のみ実行されます。

もうひとつは draw 実施のスクリプトです。これを描画用スクリプトと呼びます。このスクリプトは描画時および再描画時に実行されます。

設定

Script AO を Layout に追加するには他の AO と同じく Layout Editor 上で行います。

Layout を作成し、Script を設定したい Axis Area に移動し、AO ポップアップメニューから”Script”を選択します。

そうすると Script 設定画面が開かれます。

Script 設定時、右図のようなダイアログが表示されます。ここで上のエディットボックス(A)に Axis 用スクリプトを、下のエディットボックス(B)に描画用スクリプトを記述します。

内容がよければ OK(C) ボタンを押して確定します。

以下、Axis 用スクリプト、描画用スクリプトの詳細を説明します。

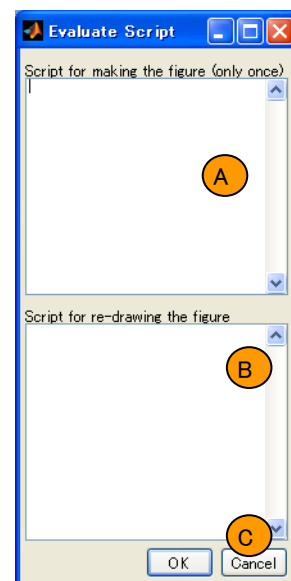


図 0.1 CO 設定

Axis 用スクリプト

Axis 用スクリプトは、Axis および同一 Axis 内にある後続の AO に影響を与える curdata を編集するためのスクリプトです。このスクリプトは描画時に1度のみ実行されます。

また、AO は全ての Common-CO から Callback されるため、1度きりで良い処理かつ時間の掛かる処理も Axis 用スクリプトに入れます。

Axis 用スクリプトは Axis-Area の描画処理内で実行されます。

そのため変更は親 Axis と後続する AO に影響を及ぼします。Axis 内で利用できる主なデータは以下になります。

表 0.1 Axis 用スクリプト利用可能変数

フィールド名	内容
h.axes	Axes ハンドル
curdata	Axis-Area 内の curdata
obj[idx]	ScriptAO の ObjectData

ここで例を示します。

たとえば、対象 Axis ではデータの種類(kind)として全ヘモグロビンデータのみ表示する場合、Axis に以下の様な設定を付加します。

```
curdata.kind=3;  
title('Kind =3');  
xlabel('time [sec]');  
ylabel('Total HB data');
```

curdata を設定
(以降の Axis 全体に影響)

一度だけ行われるべき処理

ある情報(foovar)を取得する関数(foo)に時間がかかるとし、この値が Callback に影響を受けない場合、以下のように事前に計算します。

```
[hdata, data]=p3_LayoutViewerTool('getCurrentData', curdata.gcf, curdata);  
curdata.foovar=foo(hdata, data);
```

描画用スクリプト

描画用スクリプトは draw 実施のスクリプトです。このスクリプトは描画時および再描画時に実行されます。

Script AO の draw サブ関数処理内で実行されます。

そのため変更は curdata の変更は再描画時にのみ引き継がれます。AO: draw サブ関数内で利用できる主なデータは以下になります。

表 0.2 Axis 用スクリプト利用可能変数

フィールド名	内容
gca0	親 Axes ハンドル
curdata	AO 内の curdata
objdata	ScriptAO の ObjectData
ObjectID	再描画時のみ存在する。ObjectID。
hout	出力ハンドル

ここで例を示します。

たとえば、Line-Property を無視し、HB データを線で表示します。

```
[hdata, data]=p3_LayoutViewerTool('getCurrentData', curdata.gcf, curdata);  
  
unit = 1000/hdata.samplingperiod;  
t0=1:size(data, 1);  
t=(t0 -1)/unit;  
  
kind=curdata.kind;  
  
hout.h(end+1)=line(t, data(:, 1, kind));  
hout.tag{end+1}=['XX' hdata.TAGs.DataTag{kind}];
```

POTATO データ取得

時間軸計算

表示, 出力ハンドルの設定

ここで、hout.h が設定されていない場合、Redraw 時に削除されませんのでご注意ください。