

グラフィカル・ユーザー・インターフェース  
構築ツール

開発ライブラリ  
プログラミング・マニュアル

Version 3.0

# はじめに

本マニュアルは、X Window System 上で動作する画面を作成する為の  
"開発ライブラリ" を使いこなす為のプログラミングマニュアルです。

本マニュアルでは、実際にプログラミングを行なう為の事例と説明を元に  
編集しています。各部品(オブジェクト)及び関数(サブルーチン)の詳細説明  
は記載されていない為『X-Mate開発ライブラリ リファレンスマニュアル』  
を参照して下さい。

本書をお読みになり、X-Mateのプログラミングにお役に立てれば幸いと存じます。

## 目次

<b>1 概要</b>	( 6 )
<b>2 開発手順</b>	( 7 )
<b>3 プログラミング</b>	( 9 )
1 main関数	( 10 )
2 コールバックルーチン	( 11 )
3 コンパイル	( 11 )
<b>4 ライブラリ関数</b>	( 12 )
1 基本関数	( 13 )
TKinit	
TKopen	
TKentry	
TKevent	
TKexit	
(TKclose)	
2 部品操作	( 14 )
TKentry	
TKdisplay	
TKerase	
TKdelete	
TKchgcell	
. 部品操作の種類と働き	( 14 )
. 部品制御手順1	( 15 )
. 部品制御手順2	( 15 )
3 キー入力操作	( 16 )
TKkeysw	
TKkeyoff	
TKimode	
TKpastlock	
TKgetbuff	
. キー入力位置の変更	( 16 )
. 入力文字の制御	( 16 )
. 入力モードの切り換え	( 17 )
4 マウスカーソル操作	( 18 )
TKwarp	
TKchgcursor	
TKchgbcursor	
. マウスカーソルの位置を変更す	( 18 )
. マウスカーソルの形状を変更する	( 18 )
5 フォント種別追加	( 19 )
TKaddfont	

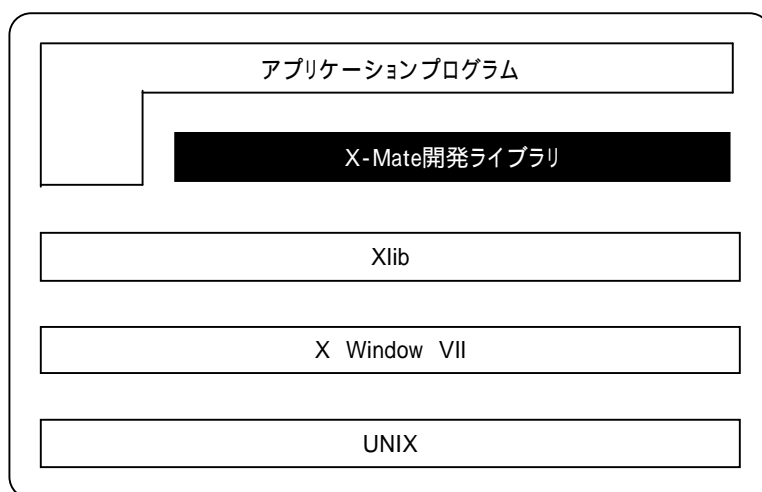
6	パネル操作	( 20 )
	TKopen	
	TKclose	
	TKuopen	
	TKiopen	
	TKiconning	
	TKmapping	
	TKchgpanel	
	TKnotify	
	TKevmask	
	TKclear	
	TKcanvcare	
	． パネルオープン/クローズ	( 20 )
	． アイコン化	( 20 )
	． パネル可視化/不可視化	( 21 )
	． パネル情報の変更	( 21 )
	． パネルイベント処理	( 22 )
	． パネルイベント制御	( 23 )
	． パネルクリア	( 23 )
	． キャンバスパネルを作る	( 24 )
7	子プロセス制御	( 27 )
	CWexec	
	CWgetarg	
	TKkill	
	TKchildkill	
8	プロセス間通信	( 29 )
	TKsend	
	TKnotify(KMSG)	
	アプリケーションインターフェイス	
5	イベント部品とのAPI	( 30 )
1	ボタン	( 31 )
	． モーメンタリモード	( 31 )
	． オルタネートモード	( 32 )
2	入力部品	( 33 )
3	スクロールバー	( 34 )
4	ポインタ	( 35 )
5	ムーブ	( 36 )
6	プルダウンメニュー	( 37 )
7	新メニュー	( 38 )

# 1 概要

本書ではアプリケーションプログラムを作成するための開発手順とプログラミングの解説をします。

アプリケーションプログラムのGUI部分はX-Mate開発ライブラリで実現している為、このライブラリの使用方法を解説します。

但し、下図のソフトウェア構成図で示す様にアプリケーションプログラムからXlibを直接使用する事は出来ませんが、これはXlibの知識を持っている方がXlibの便利な機能を利用出来る事を表しており、Xlibの知識がない方でもX-Mate開発ライブラリだけで十分アプリケーションを開発出来る様になっていますので、Xlibについての解説は行ないません。また、全てのXlib関数を使用出来る訳ではありません(パネルの状態などをライブラリが管理している為、使用されると弊害が発生する場合があります)。



ソフトウェア構成図

## 2 開発手段

この章ではX-Mateを使用したGUIアプリケーションを開発する為の作業手順を解説します。

### ステップ1 ソフトウェア設計

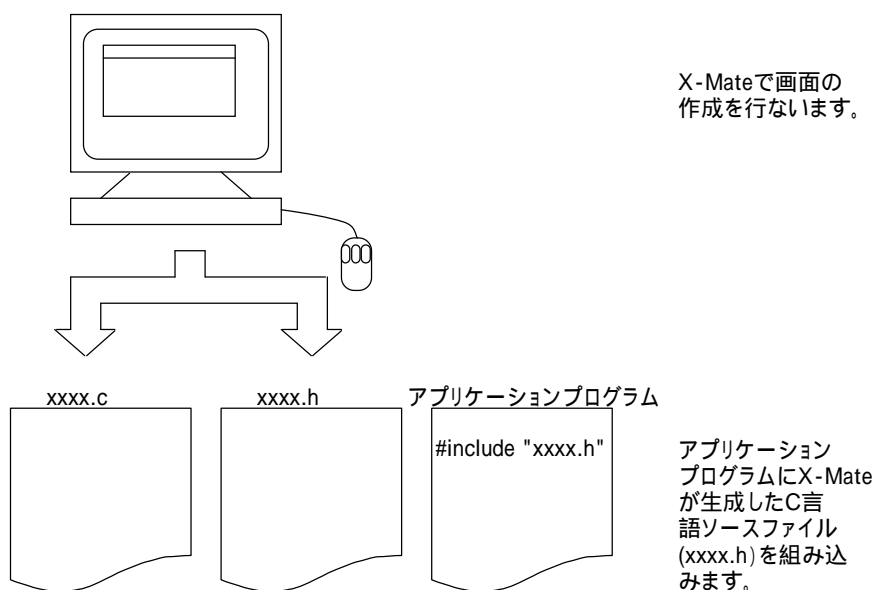
- ・ 要求仕様をより詳細に定義します。
- ・ 要求仕様を基にソフトウェアの機能仕様を作成します。  
この段階でアプリケーション(X-Mate)で実現出来る事を決定します。

### ステップ2 プログラム設計&画面設計

- ・ ソフトウェア仕様書を基にプログラム構成からフローチャート等の作成を行ないます。
- ・ 画面イメージと動作の詳細を決定します。

### ステップ3 プログラム製作&画面製作

- ・ プログラム仕様書を基にプログラムのコーディング作業を行ないます。
- ・ 画面はX-Mateを使用して作成します。作成された画面情報(表示部品、イベント部品)はC言語のソースファイルに展開されます。



#### ステップ4 動作確認試験

- ・ アプリケーションソース及び画面情報ソースファイルのコンパイルを行ない、X-Mate開発ライブラリ等をリンクする事でアプリケーション実行モジュールを生成します。
- ・ 実行モジュールを起動し、要求仕様を満足しているか確認をして下さい。

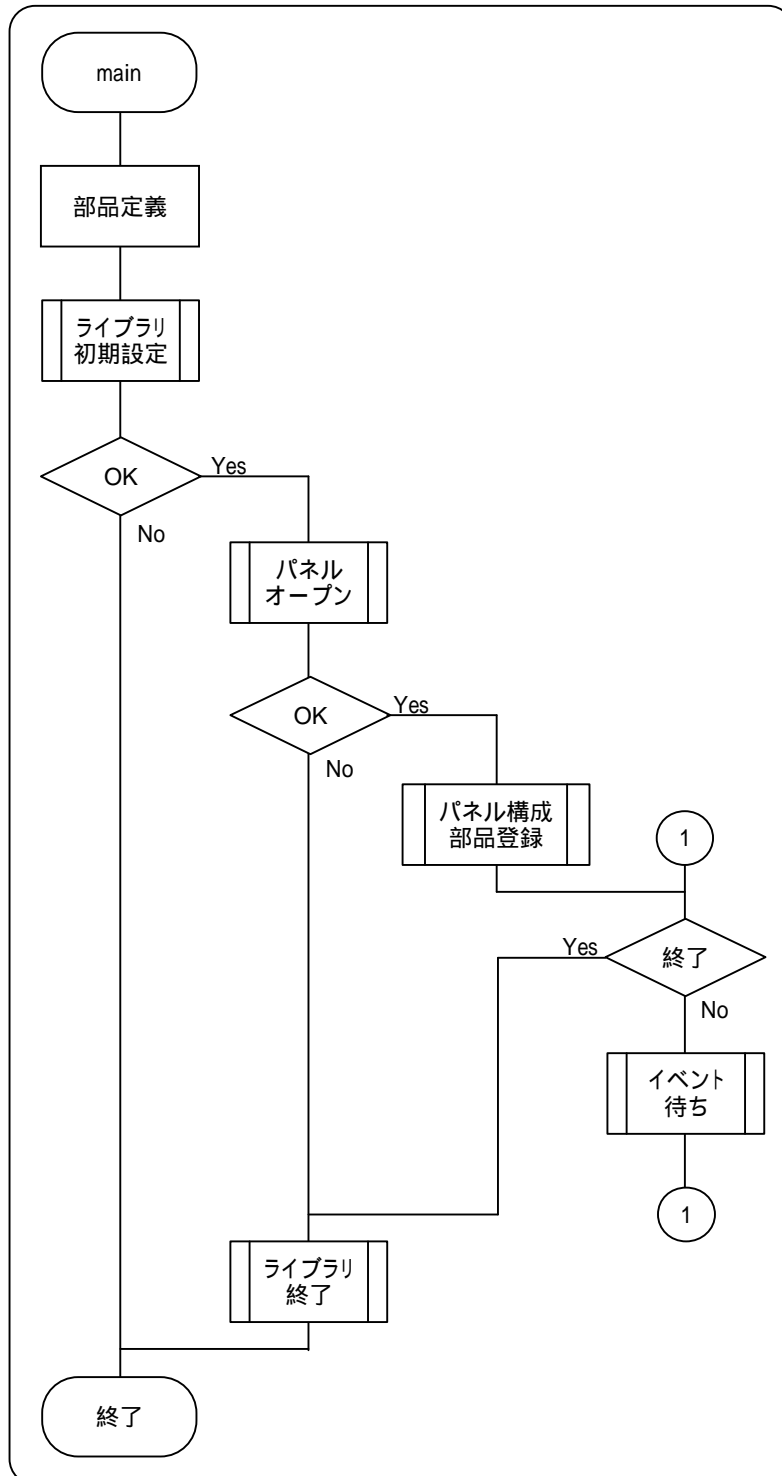
#### ステップ5 完成

- ・ 完成したアプリケーションは開発を行なったマシン(X-Mateが入っているマシン)以外では動作しません。他のマシンで動作させる為には、別途ランタイムライセンスが必要です。

### 3 プログラミング

X- main関数では、X-Mate開発ライブラリを使用してGUI環境の設定を行ないます。X-Mate開発ライブラリを使用する場合は、X-Mateが際にはコーディング手順が決まっています。

下図にmain関数のフローチャートを示します。



main関数フローチャート



## 1 main関数

main関数では、X-Mate開発ライブラリを使用してGUI環境の設定を行ないます。X-Mate開発ライブラリを使用する場合は、X-Mateが用意している次のファイルをインクルードして下さい。

<TK2/TK.h>	構造体定義、define定義
<TK2/TKusr.h>	表示色定義

X-Mateで作成したパネル部品情報を使用する場合も同様に、部品のヘッダーファイルをインクルードして下さい。

"xxx.h"	X-Mateが生成したファイル
---------	-----------------

下図に実際のコーディング例を示します。  
このコーディング例は最低限必要な手順を記述しているので完全に覚えておく様にして下さい。

また、リスト上で~~~~の部分にはパネル部品情報なので実際はアプリケーションに対応した名称が入ります。

```
#include <TK2/TK.h>
#include <TK2/TKusr.h>
#include "xxx.h"

Ktroot *root;
Ktcore *core;

void main()
{
    int sts = 0;

    root = (Ktroot *)TKinit();
    if( root != NULL )
    {
        core = (Ktcore *)TKopen(root,NULL,&panel);
        if( core != NULL )
        {
            TKentry(core,KARRAY,&panel_tree[0]);

            while(sts!=1)
            {
                sts = TKevent(root);
            }
        }
        TKexit(root);
    }
}
```

main関数コーディング例

## 2 コールバックルーチン

コールバックルーチンとはイベント部品に登録する関数でありX-Mate開発ライブラリによってパネルに発生したイベントが解析された後、対象となる関数が呼び出されます。

コールバックルーチンの記述は、それぞれが独立した関数で構成され、1回のイベントで処理が完結する仕組みになっています。

コールバックルーチンでは対応する処理をすみやかに実行し、制御をライブラリに渡す必要があります。もし、無限ループをしているとウィンドウシステムとしての制御が出来ない等の不具合が発生します。

コールバックルーチンの書式は、イベント部品の種類により異なります。部品毎の詳細は『リファレンスマニュアル』を参照して下さい。

## 3 コンパイル

実行モジュールを作成する為にはコンパイル時にいくつかのオプションを指定する必要があります。また、ご使用のOSによって設定するオプションが異なりますので、デモプログラム等の Makefile を参考にして下さい。ここでは、各OSに共通する部分のみ記述しています。

### コンパイルオプション

ご使用のOSによってはオプションを別途追加する必要があります。

-DSVR2	(Solarisの場合)
-Y	(HP-UXの場合)

### リンクオプション (ライブラリ)

・ tk2	X-Mate開発ライブラリ
・ X11	Xlib
・ wnn	日本語ライブラリ
・ m	三角関数

### コンパイル例

cc xxx.c -o yyy -ltk2 -Xext -lX11 -lwnn -lm	(SunOSの場合)
cc xxx.c -o yyy -Y -ltk2 -lX11 -lwnn -lm	(HP-UXの場合)

xxxはソースファイル名 yyyは実行ファイル名です。

## 4 ライブラリ関数

この章では、X-Mate開発ライブラリを使用してプログラミングを行なう際に必要な知識と注意事項等を実際のプログラム例を交えて解説します。

参照するプログラムリストは、\$XMATEHOME/PROG に収めていますのでREADMEファイルと合わせて参照して下さい。

この解説では関数のパラメータ、コリングシケンス等、必要以上の説明は行なっていません。詳しくはリファレンスマニュアルを参照して下さい。

## 1 基本関数

ここここで解説する関数は、アプリケ - ションで必ず使用します。

### TKinit

- ・ ライブラリの初期化を行ないます。
- ・ アプリケ - ションの起動時に必ず行なって下さい。
- ・ 戻り値のroot情報がない場合 (NULL値) は、ウィンドウが生成出来ませんので必ず確認して下さい。

### TKopen

- ・ パネルをOPENします。
- ・ root情報が必要です。
- ・ パネルを複数OPENする場合、パネルに親子関係が定義出来ます。  
「4. [6] パネル制御」参照
- ・ 戻り値のcore情報がない場合 (NULL値) は、パネルが存在していない為、必ず確認して下さい。

### TKentry

- ・ パネルに部品を登録します。
- ・ 複数の部品を一度に登録出来ます。  
「4. [2] 部品操作」参照

### TKevent

- ・ イベント待ちをします。
- ・ この関数を呼ぶ事により、パネルに発生したイベントに対応するコ - ルバックル - チンを呼び出します。
- ・ この関数から戻ってくる条件を下記に示します。
  1. 親パネルのクローズボタンが押された時。(戻り値1)
  2. 全てのパネルがクローズされた時 (戻り値1)
  3. コ - ルバックル - チンの返り値が正の値の時。  
但し、値1はライブラリが使用しているので2以上を薦めます。

### TKexit

- ・ ライブラリを終了します。
- ・ TKinitが成功していなければいけません。

### 補足 : TKclose

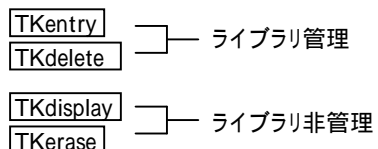
- ・ パネルのCLOSEをします。
- ・ 基本関数には含まれません。クロ - ズボタンを押してTKeventから抜ける時、ライブラリが自動的にCLOSE処理を行なっているからです。  
「4. [6] パネル制御」参照

## 2 部品操作

このパネルのOPENからCLOSEまでの間、パネルの表示部品が固定の場合は「4 1」基本関数」で説明した処理だけで満足出来ます。しかし、実際のアプリケーションでは、あるタイミングで画面の切り替えを行ったり、ある情報を読み込むと同時に、画面の情報も切り換えを行なう必要があります。ここでは部品の操作を行なう際に必要な知識と制御の手順を解説します。

### ・ 部品操作の種類と働き

部品を操作する関数には次の4つがあり、それぞれの機能はライブラリの管理と非管理の2種類に分けられます。



#### TKentry

- ・ 部品をライブラリに登録します。同時に画面表示も行ないます。
- ・ イベント部品は必ず登録しなければいけません。
- ・ 既に登録済の部品を指定した場合は何も行ないません。

#### TKdelete

- ・ ライブラリに登録している部品を管理から削除します。同時に画面から削除します。
- ・ 登録されていない部品を指定した場合は何も行ないません。

#### TKdisplay

- ・ 画面に指定した部品を表示します。
- ・ ライブラリに登録しないで表示のみ行ないます。

#### TKerase

- ・ 画面から指定した部品を消去します。
- ・ ライブラリに登録している場合は、表示はされていませんが画面には存在しています。

#### ・ 部品の制御手順1

『画面の切り替えを行なう』

この例ではボタンの状態がON又はOFFになった時、現在登録している部品のうち、切り替えの対象となる部品を削除し、新たに部品の登録を行なう事で画面を切り替えています。

ポイント1      部品を1つずつ削除/登録を行なうよりも、例の様に部品のARRAYテーブルを作成しておいた方が、処理が簡素になる上に、速度が早くなります。

ポイント2      画面の切り替えの場合は、前画面の部品は管理する必要が無いので、TKdelete/TKentryの繰り返しになります。

ポイント3      画面の部品を全て切り替える場合は、TKdeleteのパラメータでKALLを指定すると更に高速になります。

#### ・ 部品の制御手順2

『表示中の部品情報を変更する』

現在、表示されている部品情報の変更手順を示します。

1. TKeraseで現情報を画面から消去。
2. 表示情報を変更。
3. TKdisplayで画面の再描画。

ポイント1      TKeraseは背景色で塗りつぶすので、他の部品と重なっている場合は、その部品も再描画の対象です。

ポイント2      変更する部品がキー入力部品の場合、TKeraseを省略する事が出来ます。再描画時に背景色で塗りつぶしてから描画しているからです。

ポイント3      変更する部品が帳票でセル情報の場合、TKdisplayで帳票全体を再描画するよりも、TKchgcellで特定のセルだけを再描画した方が高速になります。

### ③ キー入力操作

この章では、X-Mate開発ライブラリを使用してプログラミングを行なう際に  
通常のキ - 入力の開始は、マウスカ - ソルを入力エリアでクリックする事  
により行ない、終了は[ESC]キ - の入力で行ないます。但し、キ - 入力を行  
なった後、次の入力エリアを入力状態にする場合はこの限りではありません。  
この場合に対応する為に、プログラムから入力状態を切り換える関数が用意  
されています。

#### ・ キー入力位置の変更

『キー入力を順次遷移させる』

この例では、ボタンを押す事でキー入力を開始し、リターンキーの入力で、  
次の入力エリアを入力状態にしています。また、ボタンを上げるとキー入力を  
終了します。

ポイント1      TKkeyswは特定の入力部品の入力状態をON/OFF出  
来ます。TKkeyoffは、キー入力中であるなしに関わらず  
強制的に入力を終了します。

#### ・ 入力文字の制御

『入力文字のコピー&ペースト制御』

この例は、キー入力部品に入力している文字をマウス操作で写し取る機能の  
禁止/解除の制御及び写し取っている文字列を取得します。

ポイント1      デフォルトはコピー&ペースト機能が有効となっています。  
うっかりマウス中ボタンを押してしまうと、写し取っている文字  
列が入ってしまいます。このような場合に、禁止状態にしておく事  
でキー入力以外からの入力を防ぐ事が出来ます。

ポイント2      写し取った文字列を取得 (TKgetbuff) した場合の文字列は  
mallocされていますので、不要になった際には、freeして下さい。

ポイント3      写し取った文字列を変更する事は出来ません。  
マウス操作による写し取りのみです。

・ 入力モードの切り換え

『入力モード(上書き/押入)の切り換え』

この例は、入力エリアを入力状態にする際に、指定の入力モードで入力開始します。

- ポイント1      デフォルトは押入モードとなっています。  
アプリケーション起動時にモード変換をしておく事で、次回入力開始時は指定したモードになります。
- ポイント2      この機能はキーボードの `ins` キーと同等の制御をアプリケーションで行なえる様にしたものであり、入力中に `ins` キーでモードを変更された場合は、こちらが優先されます。



#### 4 マウスカーソル操作

このマウスカーソルの制御はマウスを動かす事によりカーソル位置を移動させる事が出来ます。また、カーソルの形状はパネルオープン時に、パネル情報に設定しているXカーソルの形状に自動的に変更されます。

実際のアプリケーションでは、操作の状態により、マウスの形状を変更したり、オープンしたパネルの中央にマウスを移動させたい場合があります。

##### ・ マウスカーソルの位置を変更する

『オープンしたパネル上にマウス移動させる』

この例は、子パネルのオープンと同時にオープンしたパネル上にマウスカーソルを移動させています。また、子パネルをクロズするとマウスカーソルを親パネルに移動させます。

ポイント1      移動先の座標はCRTの絶対座標でなく、移動先の基準となるパネルの座標になります。

ポイント2      移動先のパネルのウィンドウIDが分かっているならば、どこにでも移動させる事が出来ます。

##### ・ マウスカーソルの形状を変更する

『各ボタンを押す事で、所定のカーソルの形状に変更する』

この例は、3種類のボタンを押す事で、カーソルの形状を「パネルオープン時のカーソル」、「ビットマップカーソル」、「Xカーソル」のそれぞれに変更させます。

ポイント1      パネルオープン時に指定出来るカーソルはXの標準カーソルのみです。ビットマップカーソルを指定するには、TKchgbcursor を使用して下さい。

## 5 フォント種別追加

X-Mateでは、日本語表示を意識している為、半角と全角に対応する  
いはX-Mate開発ライブラリが行なっています。  
し、この様にマウス操作によるパネルの制御は、X-Mate開発ライブラリ  
本フォントとして「スモ - ル」、「ミドル」、「ラ - ジ」の3種類の大きさを  
選択出来るようになっていきます。

この3種類以外のフォントを使用したい場合にプログラムからフォントの種  
別を追加する事が出来ます。

『フォント種別を追加』

この例は、X-Mateで作成したテキスト部品のフォント種別を新たに  
追加したフォント種別に変更した後に表示します。

ポイント1      表示文字列に漢字が含まれていない場合は、特に日本語フ  
ォント名称を指定する必要はありません。

ポイント2      ANKフォント名称と日本語フォント名称を指定する場合  
に各フォントのサイズを合わせてください。  
(例)

正   : ANK(9x18)と日本語(18x18)

誤   : ANK(10x20)と日本語(14x14)

## 6 パネル制御

アプリケ - ションを実行し、実際のパネル操作 (マウスで行なう操作) についてはX-Mate開発ライブラリが行なっています。

この様にマウス操作によるパネルの制御は、X-Mate開発ライブラリが対応しておりアプリケ - ションは意識する必要がありません。

これとは逆に、パネルの状態を意識的に変更したり、パネル自身に発生したイベントに対応した処理を行なう事が可能です。

### ・ パネルオープン/クローズ

『各種パネルのオ - プン/クロ - ズと親子関係』

この例は、複数のパネルをオ - プンする為の手順と、パネル間で親子関係を定義しています。

- ポイント1      アプリケ - ションダイアログとシステムダイアログパネルに親子関係の定義は無意味です。CRT上のパネルは自分自身以外の操作しか出来ない為です。
- ポイント2      親子関係を定義しておくと、親のパネルがクロ - ズされると自動的に子パネルをクロ - ズします。
- ポイント3      パネル1つに対して1つのcore情報が必要です。その為、パネルの2重オ - プンは出来ません。パネルのオ - プン/クロ - ズの状態を認識する必要があります。

### ・ アイコン化

通常のパネルをアイコンにするには、右上にあるアイコンボタンをマウスで押しますが、この制御をプログラムから行なう事が出来ます。

『パネルのアイコン制御』

この例は、子パネルをアイコンとしてオ - プンし、更に親パネルから子パネルをパネル化/アイコン化の制御を行なっています。

- ポイント1      bitmapファイルが存在しない場合、またはパネル情報に、アイコンファイル名が設定されていない場合は、無意味です。
- ポイント2      アイコンの位置をマウスで移動させると、2回目以降のTKiconningのアイコン座標は無意味になります。

## ・ パネル可視化/不可視化

複数のパネルを使用する場合に複雑な画面のオ - プン/クロ - ズを繰り返すと画面を描き上げるまでに時間がかかります。この様な場合は、一度オ - プンしたパネルはクロ - ズを行わずに、不可視状態にしておき、次にオ - プンする時に可視状態にする事で、最短時間でオ - プンする(様に見せかける)事が出来ます。

『パネルの可視化/不可視化』

上記の実例を示しています。

ポイント1      不可視状態にして画面から見えなくしてもパネルの情報を全て保持しているの、あまりたくさんのパネルをオ - プン状態にしているとメモリの無駄です。

## ・ パネル情報の変更

現在オープンされているパネルの位置や大きさ等を変更する事が出来ます。

『パネル情報を変更』

この例は、変更項目のボタンを押す事でパネルの情報を変更します。

ポイント1      パネルタイプの変更は出来ません。  
同一パネルタイプの情報変更を行ないます。

ポイント2      タイトルバ - をマウスで操作するとサイズが変わりますが、これを強制的に元に戻す事も可能です。  
「4.6」 . パネルイベント」参照

## パネルイベント処理

パネルに発生するイベントのほとんどは、X-Mate開発ライブラリが処理します。

アプリケーションは、下記のイベントに対応する処理を行なう事が出来ます。

- (1) クロ - ズボタンを押した場合
- (2) パネルが移動/変形等した場合
- (3) キ - 入力が発生した場合
- (4) メッセージを受信した場合 「4. [8](#) プロセス間通信」参照
- (5) Xのイベントの全て

### 『パネルイベント処理』

この例は、各イベントに対応する処理を登録し、それぞれの処理がどのように動作しているか確認出来ます。

上記(1)の場合、終了確認のパネルをオ - プンし、終了する場合は、パネルをクロ - ズし、アプリケーションを終了する。そうでない場合は、アプリケーションを継続します。

- (2)の場合、パネルが移動するとパネル情報を更新し、変形すると強制的に元の大きさに戻します。
- (3)の場合、キ - のコードを表示します。
- (4)の場合、他の関数と関連するので、ここでは除外します。
- (5)の場合、特に実例は紹介しません。Xを熟知する必要があります。

また、X-MateではXの知識が一切ない方でもアプリケーションを作る事が出来る様になっていますので特にこの機能を使う必要はないと考えています。これは、すでにXの知識があり、Xの持っている機能を使いたい方の為に用意しました。

ポイント1 一度設定したイベント処理を無効にするには、TKnotifyのサブルーチンをNULLにします。

ポイント2 修飾キ - だけのキ - 入力イベントは発生しません。

## ・ パネルイベント制御

『長い処理実行中のパネルイベントを無視する』

この例は、ボタンのコールバックルーチン内で長い処理を行なう際に、パネルに対して発生するイベントを無視する場合としない場合の違いを見せます。

- ポイント1      パネルイベントを禁止した場合は、returnする前に必ず解除しなければなりません。
- ポイント2      コールバックルーチンが呼ばれてから禁止するまでに発行されたイベントを無視する事は出来ません。
- ポイント3      ボタンやポインタ部品等でマウスイベントのButtonPress, ButtonReleaseの2つのイベントがある場合に、pressで禁止した場合、Releaseのイベントが無視される事があります。  
                 極力、Releaseイベントで行なう様にしてください。

## ・ パネルクリア

パネル画面の切り換えを行なう場合に、登録してある部品が多く、部品を消去する様気が気になる時は、画面全体を背景色で塗りつぶす事が出来ます。  
この機能は単純に塗りつぶしをしているだけで、部品が画面から削除された訳ではありません。

『パネルをクリアし、10秒後に再表示』

これは、実用的な例ではありませんが、パネルクリアの参考にして下さい。

・ キャンバスパネルを作る

これは、パネルの中に窓があり、ここに描いた図形等がはみ出して描画されない機能があり、さらに、スクロ - ルバ - があれば、これに追従して図形が移動する機能です。

- 1. キャンバスパネルのしくみ [1] (スクロ - ルバ - なし)

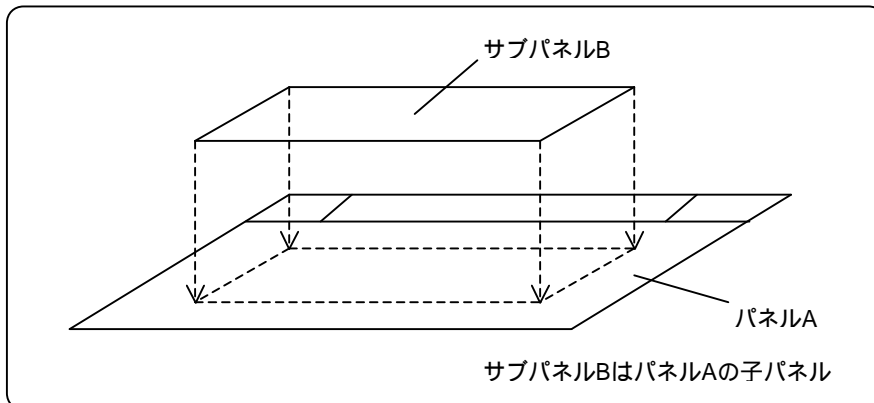
これは、パネルとサブパネルの組み合わせで実現します。サブパネルがキャンバスになります。下図参照。

『キャンバスパネルを作る (スクロ - ルバ - なし)』

この例は、前項図のサブパネルBにボタンやテキスト等の部品をわざとサブパネルからはみ出る様に登録しています。

ポイント1      サブパネルは、単独で存在する事は出来ません。必ず何らかのパネルの子パネルになります。

ポイント2      親パネルから、はみ出る事は出来ません。  
はみ出た所は画面には表示されません。  
この機能を応用した例を次に示します。



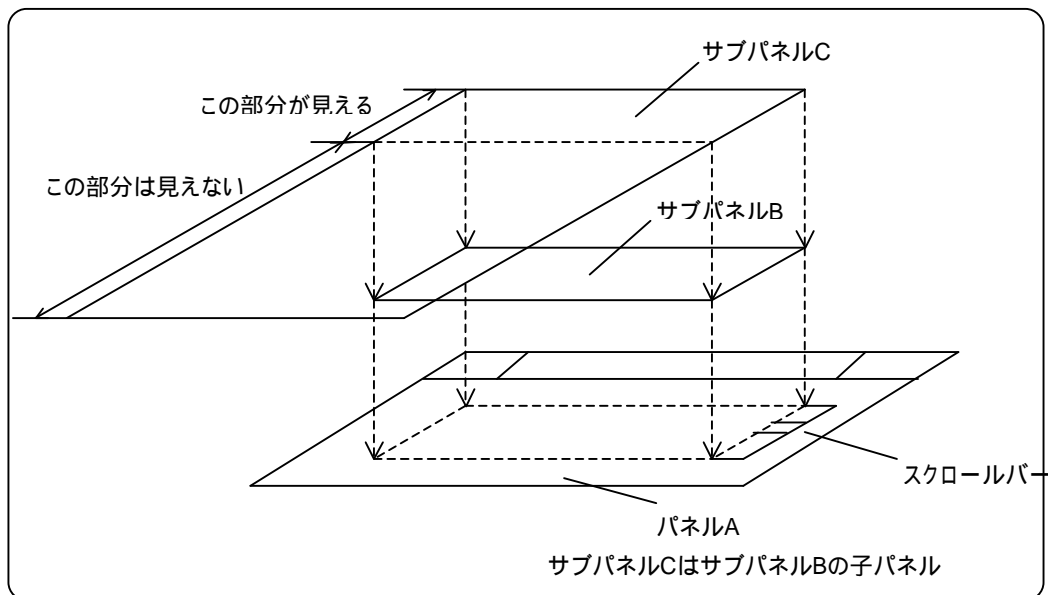
## -2. キャンバスパネルのしくみ[1](スクロ - ルバ - あり)

前記のサブパネルに対して更にこれより大きいサブパネルをオープンする事で実現しています。下図参照。

『キャンバスパネルを作る(スクロ - ルバ - あり)』

この例は、前記のサブパネルBに対して、更にサブパネルCをオープンし、このパネルに部品を登録します。

- ポイント1      サブパネルCはサブパネルBよりも大きくないと意味がありません。
- ポイント2      スクロールバーを動かすとサブパネルCの位置を動かすしくみになります。
- ポイント3      スクロールバーのサイズはサブパネルCのサイズを100%とした場合にサブパネルBのサイズが何%になるかを算出する必要があります。  
また、スクロールバーが動いた場合1%が何ドットに対応するかも、あらかじめ算出しておく必要があります。





### -3. キャンバスパネルのしくみ[2]

前記では、サブパネルを応用した場合でしたが、ここではキャンバス情報を利用した場合の説明をします。

『キャンバスパネルを作る2(キャンバス部品を使う)』

この例は、前記の例題に対してキャンバス情報を使用しています。

ポイント1      前述までの手法よりも便利かつ手軽になります。

ポイント2      スクロールバーなどのパネルの制御は必要ありません。

## 7 子プロセス制御

前項までは、1 プロセスを前提に解説してきました。  
いてはX-Mate開発ライブラリが行なっています。  
をこの様にマウス操作によるパネルの制御は、X-Mate開発ライブラリ

### 『複数プロセスの制御』

この例は、子プロセス(パネル)の起動から終了までに必要な手続きを記述しています。また、ここで行なっている処理は一例ではなく最低限必要なルールですので、よく理解して下さい。

#### 親プロセス

- ポイント1 親プロセスは子プロセスを起動する際にウィンドウIDを渡す必要があります。もし、渡さなければパネル間の親子関係が確立出来ないからです。
- ポイント2 子プロセスを起動すると親パネルはプロセスIDを得る事が出来ます。これはPSコマンドで表示されるPIDと同値であり、ウィンドウIDとは異なります。
- ポイント3 親プロセスはsignalのSIGCHLD処理を登録する必要があります。この処理は子プロセスが終了した時に呼ばれます。この処理内で実行しているwait関数は最も重要な働きがあり、これを行わなければ子プロセスが完全に終了出来なくなります。
- ポイント4 signalのSIGCHLD処理では終了した子プロセスのプロセスIDを知る事が出来るので、子プロセスの動作状態を管理する事が可能になります。
- ポイント5 CWexecでプロセスIDを取得しただけでは子プロセスが完全に起動した訳ではありません。子プロセスからウィンドウIDを通知するメッセージの受信で確立する様にして下さい。  
「4.8 プロセス間通信」参照
- ポイント6 CWexecでは子プロセスに対して最大6個までの数値パラメータを引き渡す事が出来ます。ウィンドウIDもパラメータの1つです。

## . 子プロセス

ポイント1 子プロセスで行なう処理は親プロセスから渡されたパラメータを取り出す処理 (CWgetarg) とパネルオープンに成功した時にウィンドウIDを通知する処理 (TKsend) を記述するだけです。

ポイント2 子プロセスの単体試験を行ないたい場合はコマンドラインから子プロセスを直接起動します。この時、6つの数値パラメータを空白で区切って入力します。親ウィンドウIDの値は 0値にしておいて下さい。

(例) % child 0 0 0 0 0 0

子プロセスの処理では親ウィンドウIDが0の場合はTKsendを行なわない様にする処理を追加しておいて下さい。

## 8 プロセス間通信

複数のプロセスを起動する場合それぞれが単独で動作する事はほとんどあり  
いてはX-Mate開発ライブラリが行なっています。

この様にマウス操作によるパネルの制御は、X-Mate開発ライブラリ  
TKnotifyによるメッセージ受信処理の登録です。

『他プロセスから通知されたデータをグラフ化する』

この例は、定周期に親プロセスから数値データを受け取り、子プロセス  
はこれを座標値に変換しグラフ化して表示します。

- ポイント1      メッセージの送受信を行なう場合メッセージの種別を認識  
するためにIDを定義します。この時、双方のプロセスで直  
接数値を使用したり、別々に#define定義を行なってはいけ  
ません。共通のヘッダーファイルを作成し双方でこれをincl  
udeする様にして下さい。
- ポイント2      今回は他のプロセスに対して通信を行ないましたが自分自  
身にメッセージを送信する事も可能です。送り先のウィン  
ドウIDを自分を示す様にだけです。(core->kit)
- ポイント3      送信するデータは何でも良いが受信側はデータのポインタ  
を受け取るので、これの型変換が必要です。また、このポイ  
ンタはライブラリが一時的に確保している領域なので受信し  
たデータを保存しておきたい場合はアプリケーションの領域  
にコピーする必要があります。
- ポイント4      文字列のデータを送信する場合はNUL文字も転送する事  
を忘れないで下さい。  
(例) strlen("文字列")+1
- ポイント5      TKsendを行なう場合は受信側の処理能力を考慮する必要が  
あります。大量に送信すると受信出来ない場合があります。

## 5 イベント部品とのAPI

この章ではイベント部品のコールバックルーチンを作成する際に必要な知識を解説します。

イベント部品の動作とコールバックルーチンが呼び出されるタイミング等を詳しく解説していますので参考にして下さい。

### コールバックルーチンの登録

コールバックルーチンはイベントが発生した時に呼び出される関数でGUIとアプリケーションとのI/Fになります。

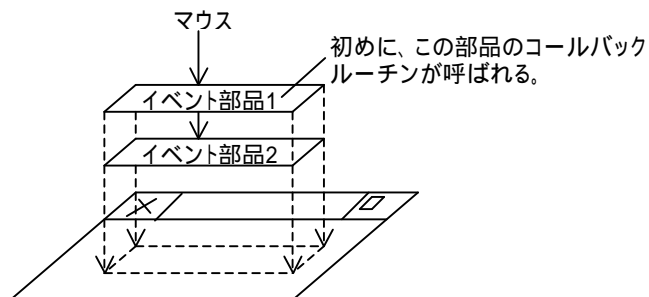
このコールバックルーチンをイベント部品の情報に登録しておく事でX-Mate開発ライブラリから自動的に呼び出されます。登録の方法はX-Mateを起動し、部品の詳細設定画面の「処理名称」の欄に関数名を設定するだけです。

### コールバックルーチンの起動

コールバックルーチンはイベント部品の種類により起動の方法とタイミングが異なります。イベントが発生する種類を次に示します。

- ・ マウスボタンのプレス
- ・ マウスボタンのリリース
- ・ マウスのムーブ
- ・ キー入力の終了 ( ☐ ☐ ☐ ☐ ...etc. 入力部品により異なる)
- ・ メッセージの受信 (部品のイベントではありません。  
4.ライブラリ関数参照 )

パネルの同一領域に複数のイベント部点を登録する事が出来ます。この時、パネルの一番上にあるコールバックルーチンが最初に呼び出されます。また、一番上のコールバックルーチンの戻り値の設定で2番目のイベント部品に対応するコールバックルーチンを呼び出す事も可能です。下図参照。詳細は『リファレンスマニュアル』参照。



**注意 1**

重複したイベント部品の制御はマウスボタンのプレスにのみ対応しており、リリースのコールバックルーチンは最後に呼び出された部品にのみ対応します。

**注意 2**

プルダウンメニュー部品はマウスボタンのプレスではコールバックルーチンは起動しませんので、この部品の下にイベント部品を登録する事は無意味です。

X-Mate開発ライブラリから自動的に呼び出されます。登録の方法

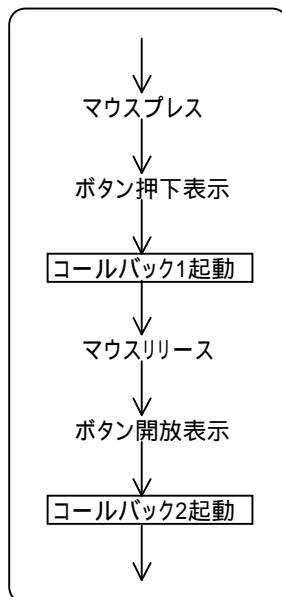
**1 ボタン**

ボタン部品はマウスの操作よりイベントが発生します。コールバックルーチンはマウスボタンのプレスとリリースで起動されます。また、ボタンのモードにより動作が若干異なりますので各モード別に解説します。

・ モーメンタリモード

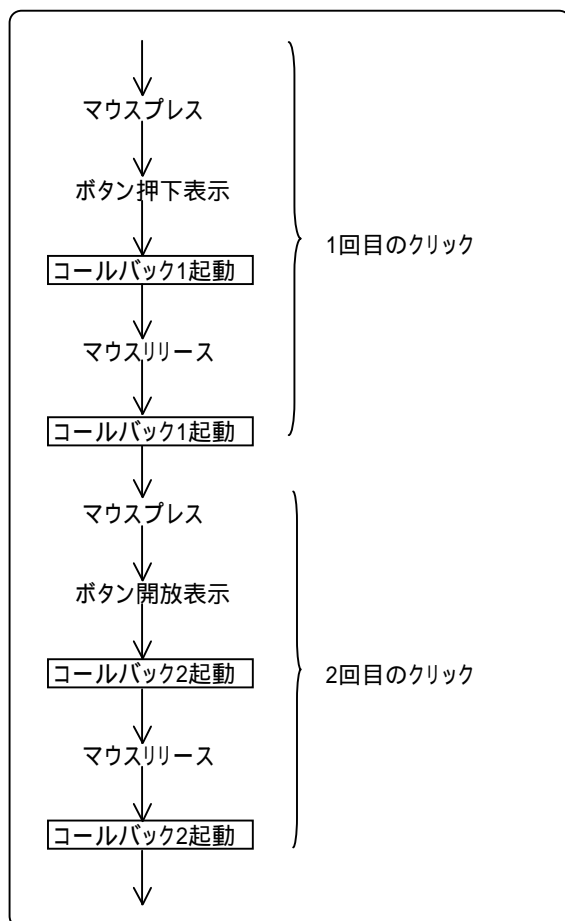
このモードのボタンではマウスボタンを押している最中は表示が押下状態になり、マウスボタンを離すと同時に表示は開放状態になります。

コールバックルーチンはボタンに登録してあるコールバック1がマウス操作に対応して2回呼び出されます。



## オルタネートモード

このモードのボタンはマウスボタンの1回のクリックで表示が押下状態になり、コールバック1が呼び出されます。次のクリックで表示は開放状態になり、コールバック2が呼び出されます。各コールバックルーチンはマウスのプレス/リリースで2回ずつ起動します。



## ② 入力部品(データ入力、テキスト入力、新テキスト入力、帳票)

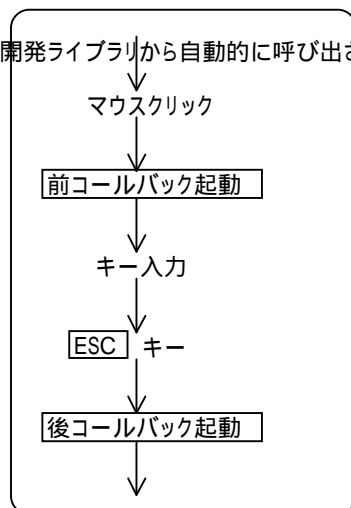
キー入力に対応する部品には数種類あり、それぞれは独特の機能を持っておりアプリケーションでは用途に合わせて使い分ける事が出来ます。

入力部品のコールバックルーチンは、入力が開始する時と終了する時の2回(2種類)を起動します。

コールバックルーチンが呼び出されるタイミングは部品により若干異なる所がありますが基本的な動作は同じです。

### ・ 共通動作

X-Mate開発ライブラリから自動的に呼び出されます。登録の方法



### ・ 異なる動作

後コールバックルーチンが呼び出されるタイミングが部品により異なります。これは、入力の終了となるキーが異なっているからです。詳細は『リファレンスマニュアル』参照。

### 各入力部品の特徴

- ・ データ入力 : 入力文字の制御を行ない、数値変換等を行なう。
- ・ テキスト入力 : 全ての文字入力を行なう。
- ・ 新テキスト : テキスト入力と同じ。更にスクロール表示が可能。
- ・ 帳票 : 複数のデータを表形式で管理。



### ③ スクロールバー

スクロールバー部品はバー部分をマウスで移動させた時にイベントが発生します。

コールバックルーチンには現在のバー位置がパラメータで渡されるのでアプリケーションでは、これに対応する処理を行いません。コールバックルーチンが呼び出されるタイミングは次の通りです。

(モードにより異なります)

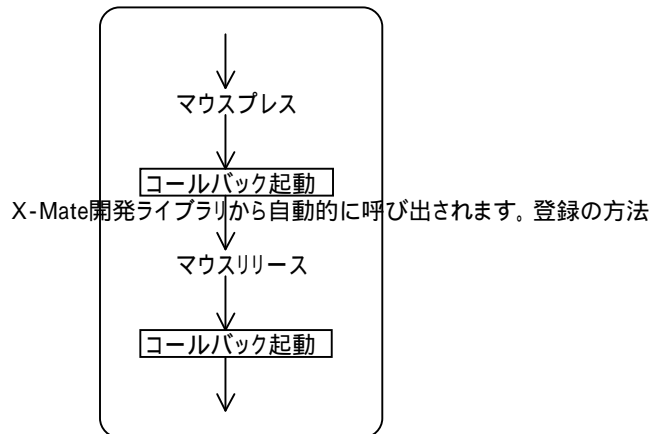
1. マウスボタンを押したままバーを動かす、この時リアルタイムにコールバックルーチンが連続して起動されます。
- X-Ma 2. マウス中ボタンを動かしている最中は起動されずに、マウスボタンをリリースした時に起動される。
3. 各マウスボタンをクリックした時に対応する位置にバーが移動し、起動される。

#### 補足

スクロールバーの長さや位置は登録(TKentry)を行なう前にあらかじめ設定する様にして下さい。

#### 4 ポインタ

ポインタ部品はマウスボタンのイベントを検出します。  
コールバックルーチンはマウスのプレスとリリースで2回起動されます。  
また、特定のマウスボタンのみを有効にする事も可能です。



この部品は画面には何も表示されない事とマウスボタンの指定を出来る事を除けばボタン部品のモーメンタリモードと同じ動きをしている事が分かります。

マウスボタンの指定がないと、どのボタンが押されていても反応しませんので、部品の存在が無意味になってしまいます。

この部品は単体で使われるよりも他の部品と組み合わせて使用する場合が多い様です。但し、組み合わせて使えない部品があります。下記参照。

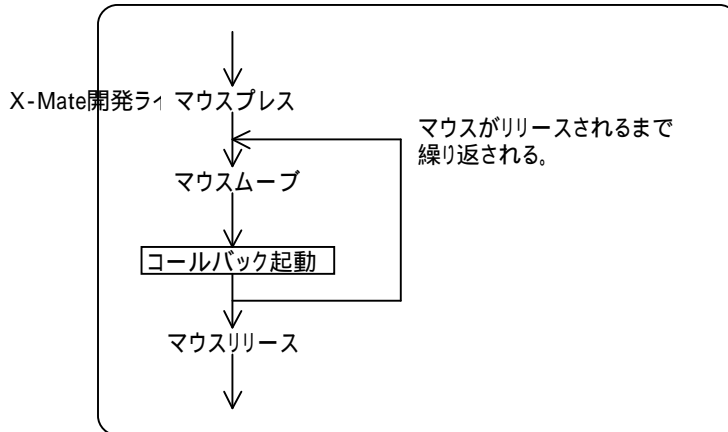
- ・ 帳票(スクロールバーが付く場合)
- ・ スクロールテキスト
- ・ 新テキスト

これらの部品は全てスクロールバーが付加される部品で、表示情報を上下左右にスクロールして見る事が出来ます。その為、表示部分がサブパネル化されているので、現在表示中のパネルに対して登録しているポインタ部品の制御が出来なくなってしまう為です。

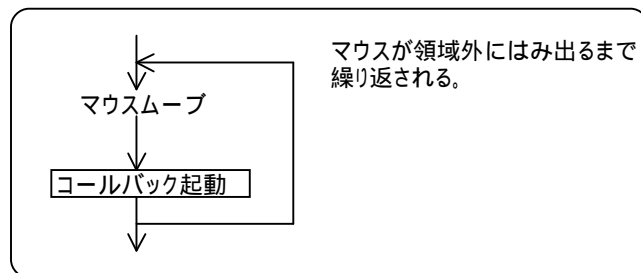
## 5 ムーブ

ムーブ部品はマウスのムーブ(移動)を検出します。コールバックルーチンはマウスの移動毎に起動されます。この時、特定のマウスボタンのみを有効にする事も可能です。また、どのマウスボタンも指定していない場合は領域内でマウスが移動ただけでコールバックルーチンが呼び出されます。

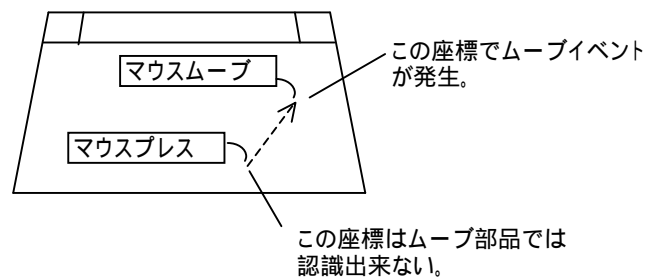
・ マウスボタンを指定した場合



・ マウスボタンを指定していない場合

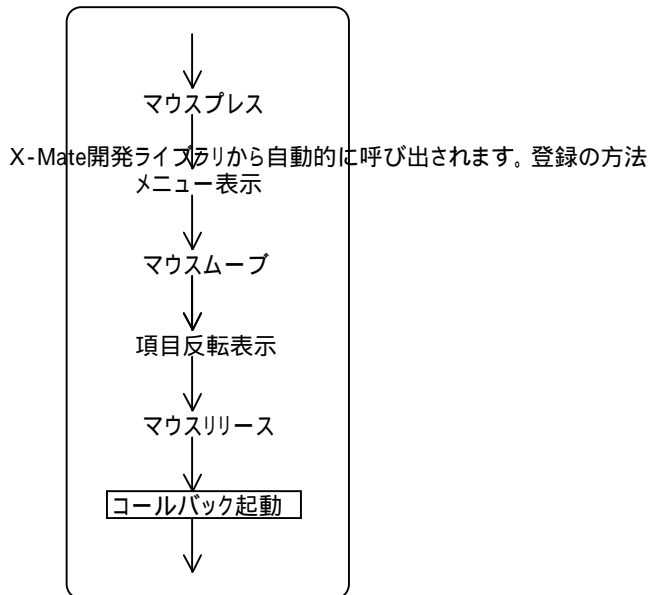


この部品は移動が発生した後の座標しか分からないのでマウスをプレス/リリースされた位置は認識出来ません。その為、ポインタ部品と組み合わせる必要があります。



## ⑥ プルダウンメニュー

プルダウンメニュー部品は通常は画面に表示されておらず、マウスのボタンをプレスする事によりメニューが画面に表示されます。この状態でマウスを上下に移動してメニューの項目を選択し、マウスのボタンをリリースする事で項目が決定されコールバックルーチンを起動します。



### 注意

マウスボタンをプレスした時、他のイベント部品の様にコールバックルーチンが呼ばれませんので、この部品の下にイベント部品を登録する事は無意味です。

マウスボタンをリリースした時に、何も項目が選択されていない場合は、コールバックルーチンは起動されません。

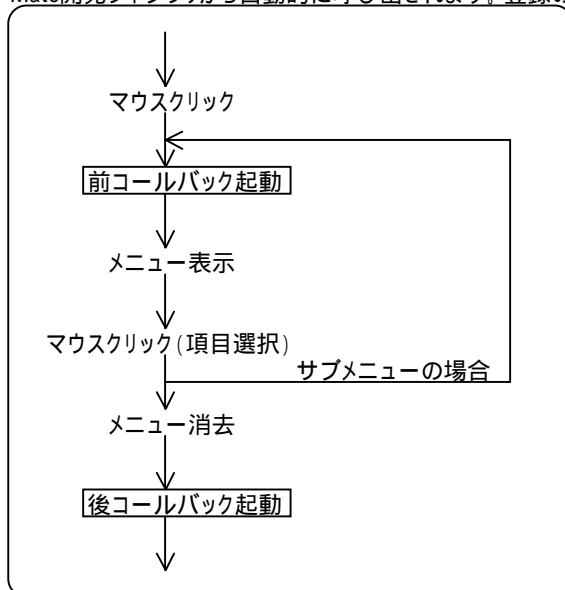
## 7 新メニュー

新メニュー部品はブルダウンメニュー部品を大幅に機能アップした部品です。

コールバックルーチンはメニューが画面に表示される時と、メニューが終了する時(項目が選択されなかった場合も同様)に呼び出されます。また、このメニューは特別な項目を選択すると更にメニューを表示するサブメニュー機能があります。この時も同様にサブメニューが表示される時にコールバックルーチンが呼び出されます。

- ・ ポップアップモード以外の場合。

X-Mate開発ライブラリから自動的に呼び出されます。登録の方法



- ・ ポップアップモードの場合。

ポップアップモードではアプリケーションプログラムから意図的にメニューを表示出来ます。メニューが表示されてからの操作は上図と同じです。

この機能を有効に利用する為にはメニュー状態フラグを活用します。

1. メニュー状態フラグをONで登録(TKentry)した場合。

登録と同時にメニューを表示します。  
2回目以降にメニューを表示したい時はフラグをONに設定しTKdisplayを行なうとメニューが表示されます。

### 注意

既に登録済みの場合は無効になります。

2. メニュー状態フラグをOFFで登録(TKentry)した場合。

登録時にはメニューは表示されません。  
2回目以降は上記と同じです。

- ・ 本書及びプログラムは、著作権上、当社に無断で使用、複製する事は出来ません。
- ・ 本書及びプログラムの運用上のトラブルについては責任を負いかねますのでご了承願います。
- ・ 本書または本製品の内容にご不審な点がございましたら御連絡下さい。
- ・ 本書及びプログラムは予告なしに変更する事があります。

初版発行 1991年 1月  
第八版 2010年 4月

Copyright 1991 FUJI Data System Co.,Ltd.