Fujitsu Touch Panel (Serial)
デバイスドライバユーザーズマニュアル

for Windows95/98/Me/NT4.0/2000/XP
V1.0L27

本マニュアルは Serial タッチパネルドライバの設定方法についてのマニュアルです。インストール方法についてはドライバ内の readme.txt を参照してください。

1、設定画面の起動

本製品インストール後、コントロールパネルに下図の標準「タッチパネル」アイコンが登録されます。コントロールパネル上の「タッチパネル」アイコンをダブルクリックすると、設定ツールが起動します。

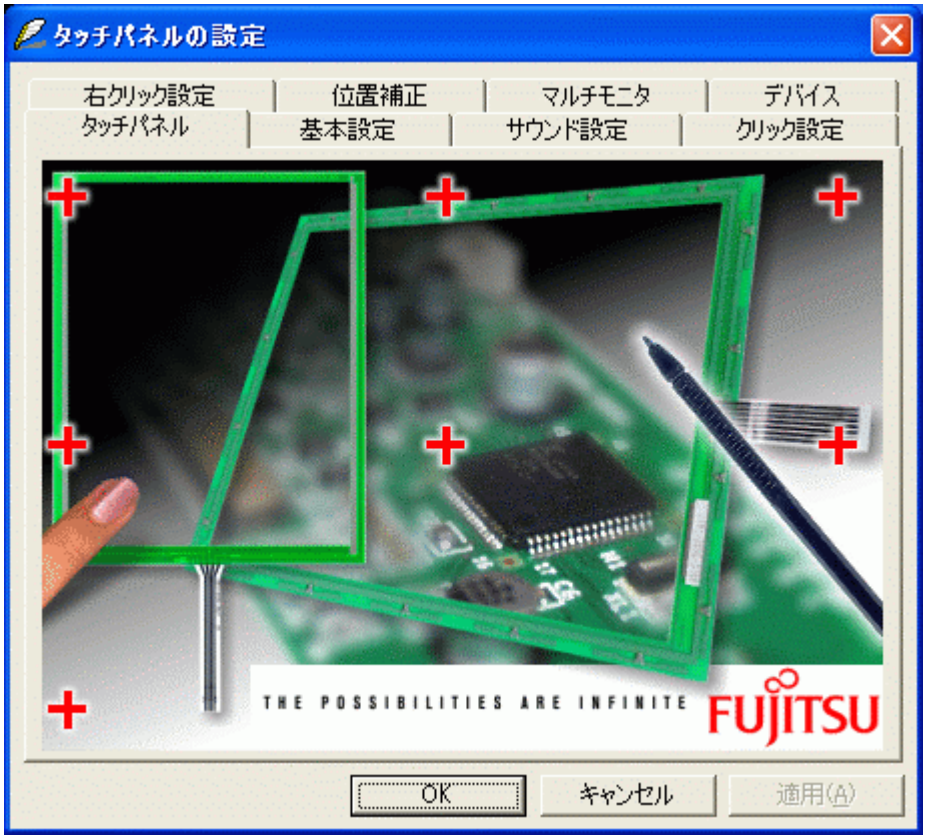
タッチパネル設定ツールの起動アイコン

2、設定画面

2.1 起動画面

設定ツールが起動すると下図のダイアログボックスが表示されます。Windows95/98/Me/NT4.0/2000/XP 用 NonPNP タッチパネルは 8 枚の設定ページが有ります。Windows95/98/Me/2000/XP 用 PNP タッチパネルの設定ツールはマルチモニタページを抜いた 7 枚の設定ページが有ります。

・タッチパネル画面



*1)WindowsNT4.0 の場合、PNP タッチパネルは NonPNP タッチパネルと同じドライバが使用されます。

2.2 基本設定画面

Windows95/98/Me PNP 用ドライバ、Windows95/98/Me/NT4.0 NonPNP 用デバイスドライバがインストールされている場合は、図 ①が表示されます。

Windows2000/XP PNP 用ドライバ、Windows2000/XP NonPNP 用ドライバ、Windows98/Me/NT4.0 NonPNP 用ユーザーモードドライバがインストールされている場合は、図②が表示されます。

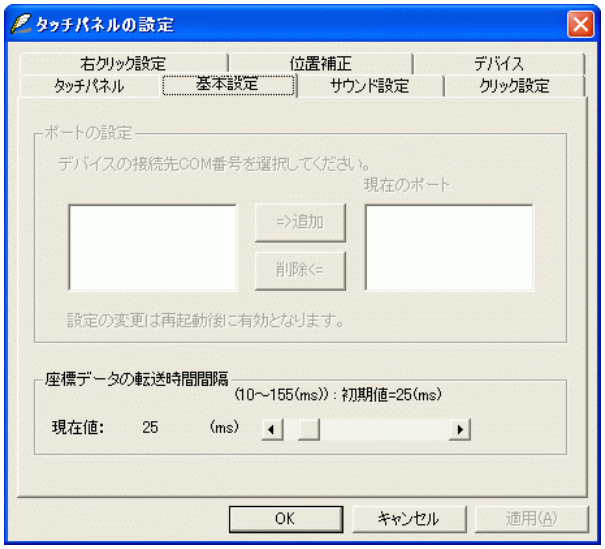


図 ①

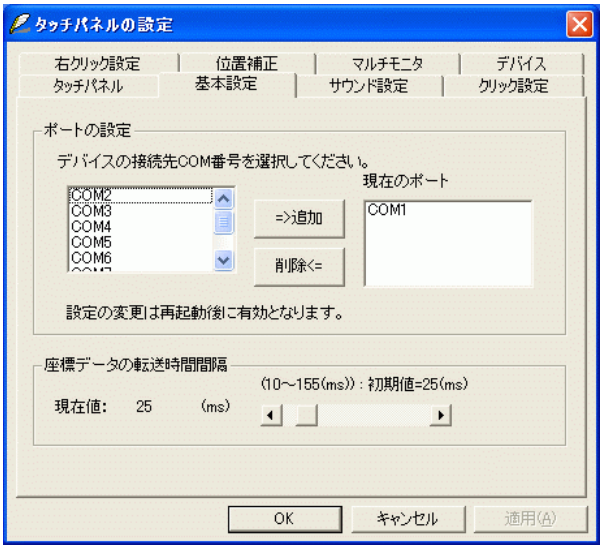


図 ②

タッチパネルの基本設定(ポート、転送間隔)に関する設定を行います。

＜ポートの設定＞

タッチパネルが接続されているポートの設定を行います。なお、設定できるポートは COM1～9までです。

・Windows95/98/Me/2000/XP 用 PNP タッチパネルの場合

COM が自動認識されるために設定を変更することはできません。そのため、Windows2000/XP ではポート設定部分、Windows95/98/Me では「現在のポート」以外のポート設定部分が入力不可状態になっています。そして、Windows95/98/Me では「現在のポート」の表示されている COM ポートにフォーカスを当てた場合、そのポートの I/O アドレスとIRQ 番号の設定値が「接続されているポート」の右側に表示されます。

・図①が表示される NonPNP タッチパネルの場合

最大 2 つの COM ポートを追加できます。また、「現在のポート」の表示されている COM ポートにフォーカスを当てた場合、そのポートの I/O アドレスとIRQ 番号の設定値が「現在のポート」の右側に表示されます。

ポートの追加方法：「=>追加」ボタンの左側の３つのフィールドを入力して、「=>追加」ボタンを押すことによって接続されているポートに追加されます。*2) *3) *4)

COM　　:デバイスが接続されている COM ポート番号を入力します。
I/O　　:デバイスが接続されている I/O アドレスを入力します。
IRQ　　:デバイスが接続されている IRQ 番号を入力します。
ポートの削除方法　：「現在のポート」から削除したいポートにフォーカスを当て、「削除<=」 ボタンを押すことによって削除されます。*2)

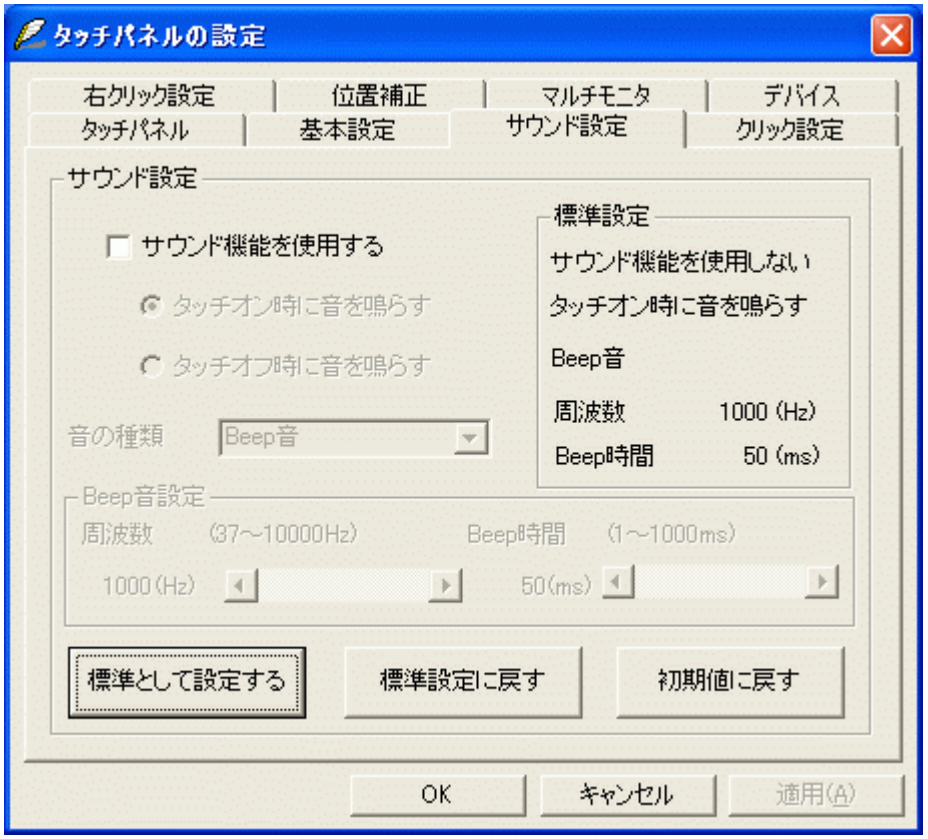
・図②が表示される NonPNP タッチパネルの場合
最大 9 つの COM ポートを追加できます。

ポートの追加方法：「=>追加」ボタンの左側のリストから追加したいポートにフォーカスを当て、「=>追加」ボタンを押すことによって「現在のポート」に選択された COM が追加されます。*2)
ポートの削除方法：「現在のポート」から削除したいポートにフォーカスを当て、「削除<=」ボタンを押すことによって選択された COM が削除されます。*2)

*2) ポートの設定を変更した場合、再起動後に設定が有効になります。その際、システムの再起動するまで「位置補正」ページの補正プログラム、「マルチモニタ」ページの「使用ポート」の選択が使用不可になります。
*3) Windows で設定されていないポートアドレス（I/O アドレス、IRQ 番号）を入力された場合、動作不良を起こすことがあります。
*4) タッチパネルが COM1～9 以外の特殊なポートに接続されている場合は、使用されていない COM ポートの番号を任意に入力します。

＜座標データの転送時間間隔＞
スクロールバーをスライドさせることにより、タッチパネルコントローラから送出される座標データの時間間隔を 10ms～155ms まで 5ms 間隔で設定できます。

2.3 サウンド設定画面



パネルタッチ/オフ時に発生させる音に関する設定を行います。

＜サウンド設定＞

サウンド機能を使用する　:音を出すか出さないかの設定をします。

タッチオン時に音を鳴らす　:パネルタッチ時に音を鳴らすように設定します。

タッチオフ時に音を鳴らす　:パネルオフ時に音を鳴らすように設定します。

音の種類　:Beep 音、メッセージ（情報）、メッセージ(警告)、メッセージ（問い合わせ）、システムエラー、一般の警告音、標準 Beep 音の中から選択することが出来ます。*5)

初期値に戻す　:サウンド設定を初期値に戻します。（サウンド機能は使用しない、タッチオン時に音を鳴らす、Beep 音、周波数：1000Hz、Beep 時間 :50ms）

標準設定として設定する　:現在のサウンド設定の状態を標準設定として保存します。また、「標準設定」の欄に保存された設定内容が表示されます。

標準設定に戻す　:標準設定として保存した設定にします。

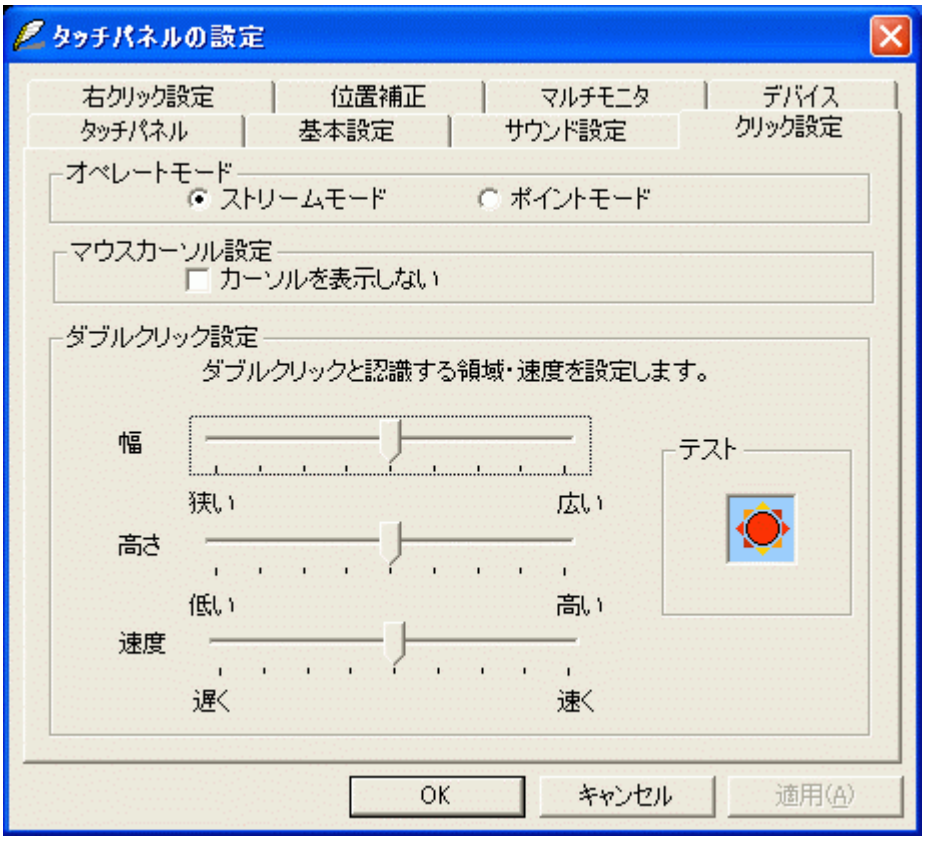
＜Beep 音設定＞

周波数　：「音の種類」を「Beep 音」とした場合の周波数の設定を行います。（37～10000Hz）

Beep 時間　：「音の種類」を「Beep 音」とした場合の Beep 時間の設定を行います。（1～1000Hz）

*5) メッセージ（情報）、メッセージ（警告）、メッセージ（問い合わせ）、システムエラー、一般の警告音で発生させる音の種類は、コントロールパネル内のサウンド設定で選択します。

2.4 クリック設定画面



タッチ動作とクリックに関する設定を行います。

＜オペレートモード＞

ストリームモード　　：タッチした時点からオフするまで継続して座標を出力します。

ポイントモード　　：点動作。タッチ時にON を通知し、即 OFF を通知します。以降タッチが継続しても、座標を出力しません。

＜マウスカーソル設定＞

「カーソルを表示しない」のチェックボックスの設定によりマウスカーソルの表示を消すことが出来ます。なお、本設定ではアプリケーションなどの独自カーソルの削除は行いません。

＜ダブルクリックの設定＞

ダブルクリックと認識される範囲・速度の変更が可能です。「OK」または「適用」ボタンを押した後に設定が変更されます。*6)

幅　　：ダブルクリックと認識する水平方向の範囲を設定します。2～32[pixel]

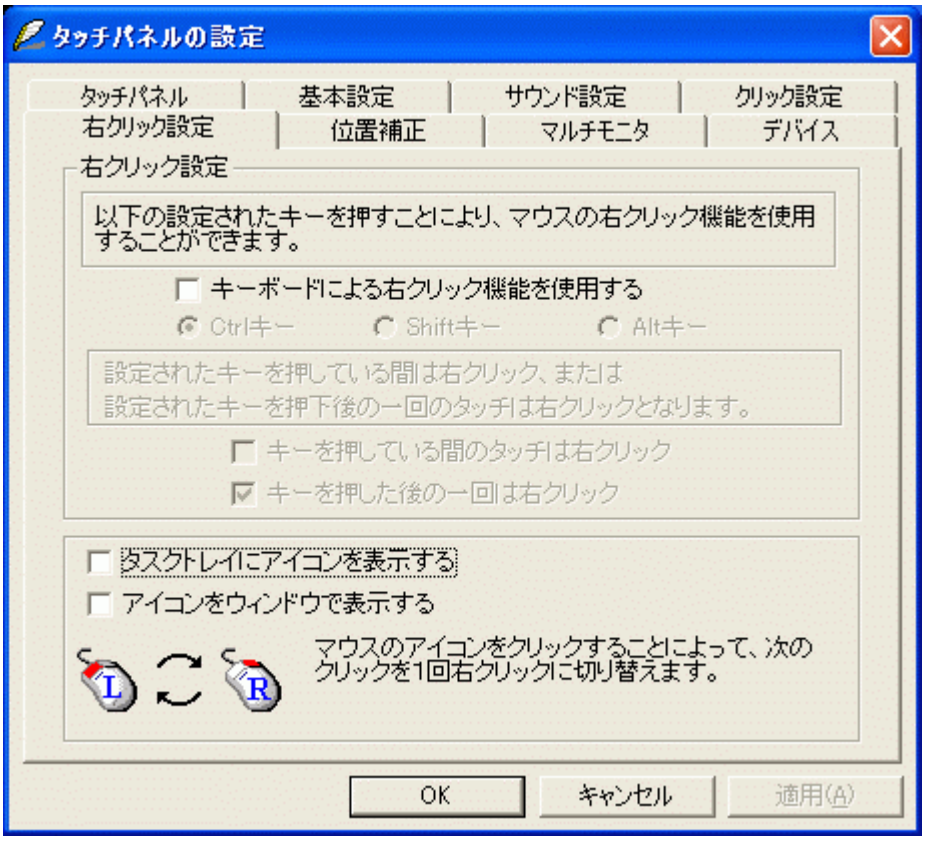
高さ　　：ダブルクリックと認識する垂直方向の範囲を設定します。2～32[pixel]

速度　　：ダブルクリックと認識するクリック間の時間を設定します。100～900[ms]

テスト　　：アイコン上を実際にダブルクリックしてテストします。ダブルクリックと認識されると表示されているアイコンが変化します。

*6)Windows システム上のマウス設定を更新する為に、システムに接続されている他のマウスについても上記の幅、高さ、速度の設定が適用されます。しかし、ダブルクリック設定の禁止の設定はタッチパネルのみ有効です。

2.5 右クリック設定画面



タッチパネル右クリック動作に関する設定を行います。

＜右クリック設定＞

キーボードによる右クリック機能を使用する。

:キーボードのキー同時操作で右クリック動作します。キーボードによる右クリックを使用する場合は、以下の項目が設定有効になります。

Ctrl キー、Shift キー、A ltキー

:どのキーに機能を与えるか設定します。

キーを押している間のタッチは右クリック

:設定したキーを押している間のタッチは右クリックとなります。

（この設定では Alt キーは選択できません ）

キーを押した後の 1 回は右クリック

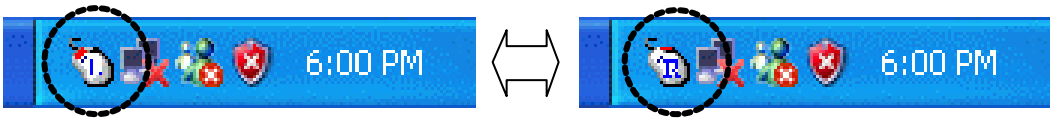
:設定したキーを押した後の 1 回のタッチを右クリックとします。

＜アイコンの表示＞

「タスクトレイにアイコンを表示させる」をチェックすると、タスクトレイに図③のようなマウス風のアイコンが登録されます。

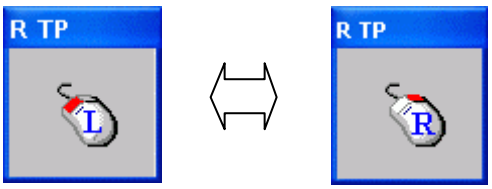
「アイコンでウィンドウを表示する」をチェックすると、デスクトップ上に図④のようなマウス風のアイコンが表示されたウィンドウが表示されます。

アイコンの動作　:アイコンの L、R の表示は現在の状態 （左クリックまたは右クリック ）を示します。アイコンをタッチするとL とR が切り替わります。R 状態の場合の一回のタッチは右クリックとなります。



L の表示　　アイコンタッチ　　**R の表示**

図③

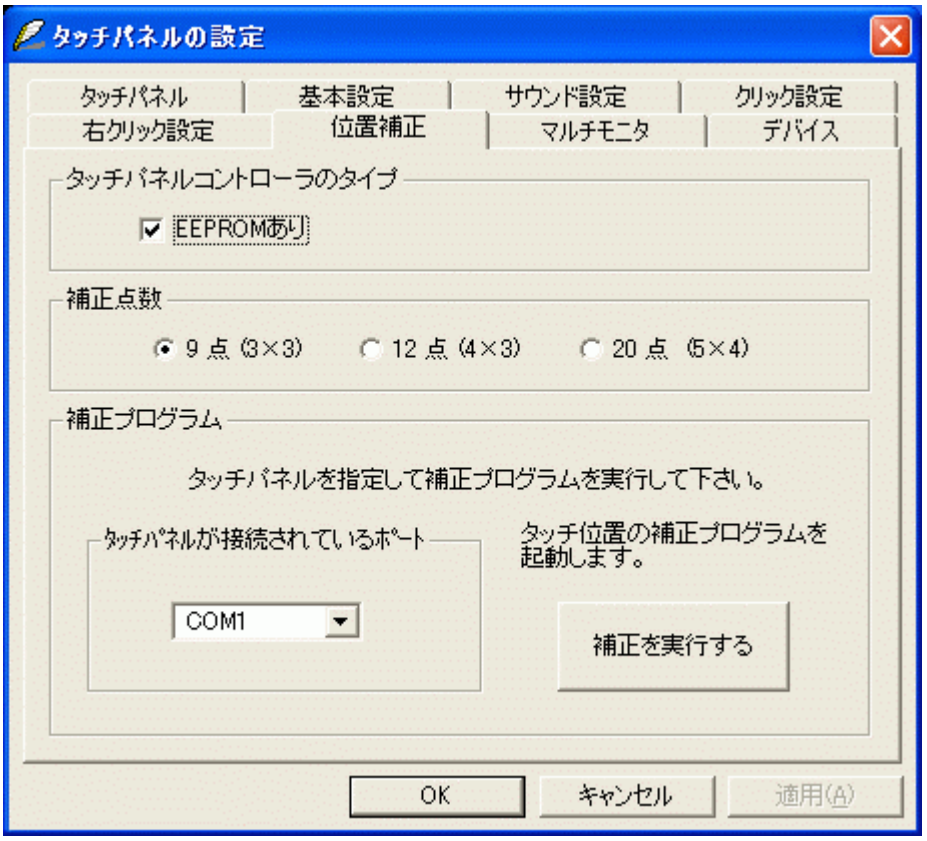


L の表示　　アイコンタッチ　　**R の表示**

図④

*7） マウスのプロバティで左利き用に設定している場合は L,R の表示は反対となります。

2.6 位置補正画面



タッチパネルの位置補正に関する設定、及び補正プログラムの実行を行います。

＜タッチパネルコントローラのタイプ＞

タッチパネルコントローラに EEPROM が搭載されている場合は「EEPROM あり」をチェックします。タッチパネルコントローラに EEPROM が搭載されていない場合は「EEPROM あり」のチェックを外します。

＜補正点数＞

9 点（横 3×縦 3）、12 点（横 4×縦 3）、20 点（横 5×縦 4）になります。

＜補正プログラム＞

接続されているタッチパネルの補正を行います。*8)

タッチパネルが接続されているポート

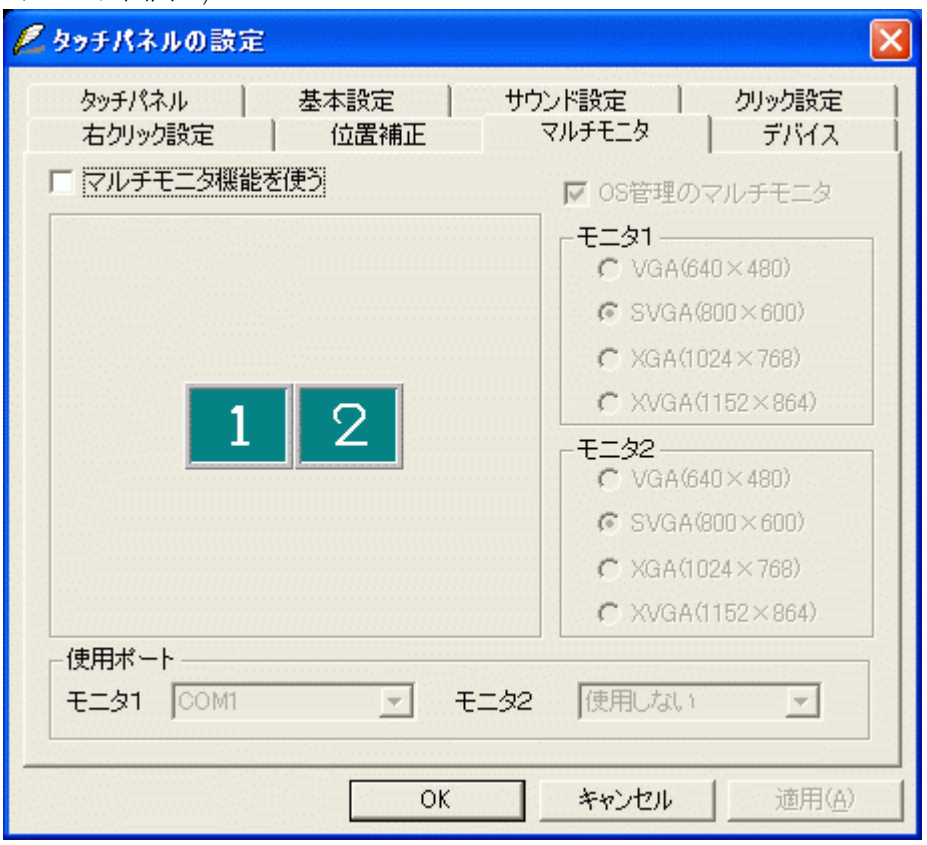
：補正を実行する COM ポートを選択します。（WindowsNT4.0 以外の PNP タッチパネルの場合は、自動で COM が認識されるために選択を行う必要はありません。）

補正を実行する

：補正プログラムを起動します。なお、タッチパネルのコントローラのタイプ、補正点数が変更されている場合、「OK」、「適用」ボタンを押して設定を更新するまで補正を実行することはできません。（補正プログラムの操作は３項参照 ）

*8) 「基本設定」ページでポートの設定を変更した場合、再起動するまで「位置補正」ページの補正プログラム、「マルチモニタ」ページの「使用ポート」の選択の機能が使用不可になります。

2.7 マルチモニタ画面*9)



マルチモニタに関する設定を行います。

＜マルチモニタ機能を使う＞

マルチモニタ機能を使用するかどうかの設定を行います。マルチモニタ機能を使用する場合は、以下の項目で設定されたとおりに座標データを変換し、Windows に通知します。

＜OS 管理マルチモニタ＞

OS の機能を用いたマルチモニタ（ 仮想デスクトップ ）を使用する場合はチェックを入れます。そうでない場合はビデオドライバでの単純画面分割によるマルチモニタとなります。

＜モニタ位置設定＞

２つのモニタの位置関係を設定します。モニタ 1,2 の画をドラッグし、実際のモニタの位置関係と同じとなるようにします。

＜モニタ 1＞

モニタ 1 の解像度を設定します。

＜モニタ 2＞

モニタ 2 の解像度を設定します。

＜使用ポート＞*10)

モニタ 1 :モニタ 1 に設置されているタッチパネルのポートを選択します。

モニタ 2 :モニタ 2 に設置されているタッチパネルのポートを選択します。

*9) Widnows95/98/Me/2000/XP 用 PNP タッチパネルではマルチモニタ設定画面は表示されません。

*10) 基本設定」ページでポートの設定を変更した場合、再起動するまで 位置補正」ページの補正プログラム、 「マルチモニタ」ページの 使用ポート」の選択の機能が使用不可になります。

2.8 デバイス画面

タッチパネルドライバに関する情報を表示します。

＜ドライババージョン＞

タッチパネルデバイスドライバ	：ドライバ名
Windows 2000	：ドライバの対応OS
V1.0L27	:使用しているドライバのバージョン
ファイルバージョン	:個々のファイルバージョン

3、補正プログラム

補正は、タッチパネルの押下位置と、表示位置とを一致させるために行います。タッチパネルドライバセットアップ時、または、タッチパネルデバイスを交換した場合は、補正を必ず行う必要があります。

3.1 補正プログラムの起動

タッチパネル設定ツールを起動します。『位置補正」ページを開きます。「タッチパネルが接続されているポート」で補正を行いたいタッチパネルが接続されているポートを選択し、『補正を実行する」ボタンをクリックします。

3.2 補正プログラムの構成

補正プログラムは、『補正作業を行う補正画面」と、『補正後にタッチパネルデバイスの押下位置と表示位置との位置を目視確認する描画テスト画面」とで構成されています。

3.3 補正画面

ⅰ)補正プログラムを実行します。

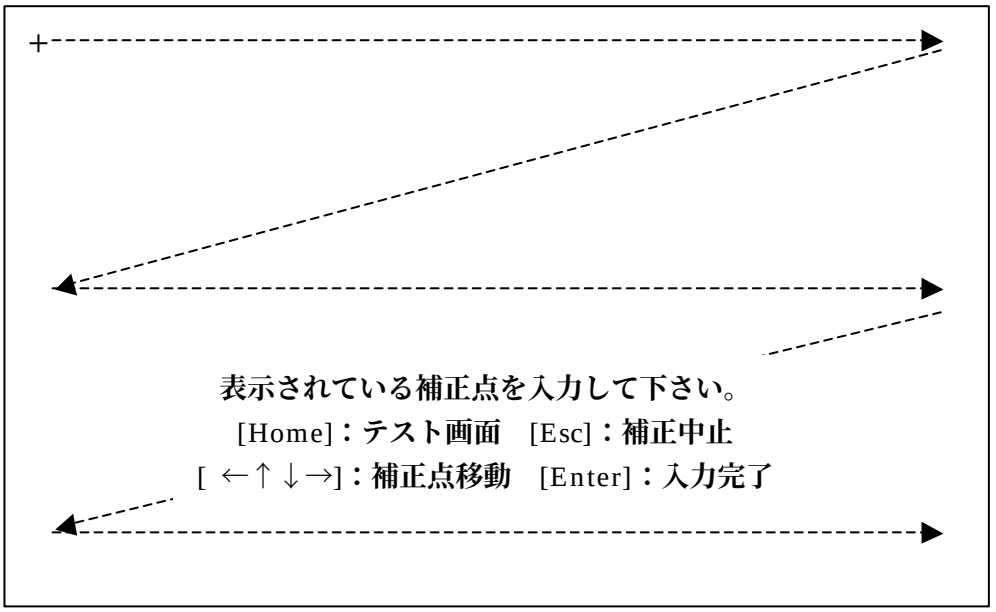
ⅱ)補正画面が表示されます。(+マークが左上に表示されます)

ⅲ)左上から右下へ順番に表示される+マークを押下します。押下点が有効だった場合、次のマークが表示されます。

ⅳ)+マークを全て押下します(9 点, 12 点, 20 点)。途中でEnter キーを押すとそれまでの補正点入力が有効になります。

ⅴ)補正が正常終了した場合、描画テスト画面に移ります。

＜補正画面＞



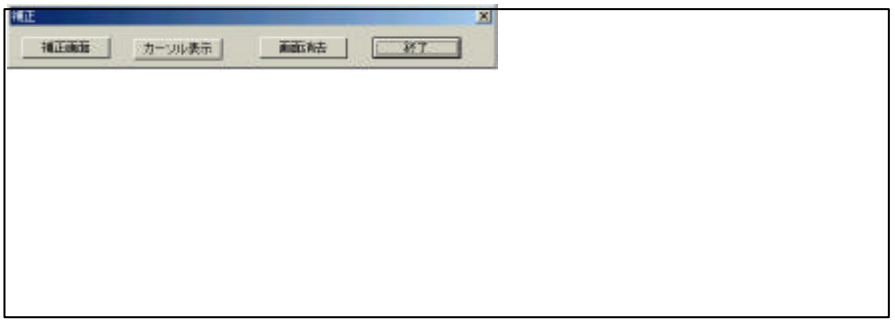
補正画面のキー入力

[Home]： テスト画面　・・・・・・・・・・補正を中止して、描画テスト画面に移ります。
[Esc]： 補正中止　・・・・・・・・・・・・・補正を中止して、補正プログラムを終了します。
[←↑↓→]： 補正点の移動　・・・・補正する点 (+マーク)が移動します。
[Enter]： 入力終了　・・・・・・・・・・・補正データを作成します。

*11)「EEPROM あり」のチェックを入れた場合、補正データおよび補正点は EEPROM に保存されます。初回の補正では、全ての補正点を入力する必要がありますが、2 回目以降は部分入力も可能です。「EEPROM あり」のチェックをはずした場合、補正データおよび補正点はレジストリに保存されます。補正では、毎回全ての補正点を入力する必要があります。

3.4 描画テスト画面

描画テスト画面は、タッチパネルデバイスによる自由描画が行えるテスト画面になっています。タッチパネルの押下位置と、表示位置との一致を目視確認することにより、補正の結果を判定できます。補正画面において、[Home]、[Enter]キー入力された場合、または、補正点を全て入力した場合に描画テスト画面に移ります。



＜キーボックス有り描画テスト画面＞

補正テスト画面のキーボックス入力

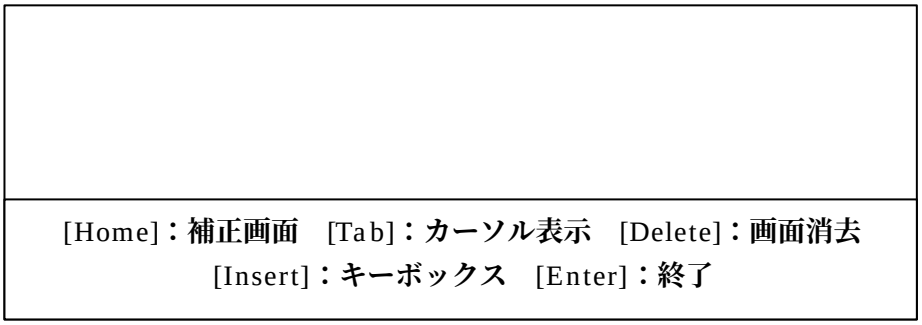
補正画面 ・・・・・・・・・・・・・・・・・・補正画面に戻り補正をやり直します。

カーソル表示・・・・・・・・・・・・・・・・画面上のカーソルの表示をon/off します。

画面消去 ・・・・・・・・・・・・・・・・・・画面上に描画した線を消去します。

終了 ・・・・・・・・・・・・・・・・・・・・・補正プログラムを終了します。

閉じるボタン・・・・・・・・・・・・・・・・ キーボックスを終了し、キーボックス無し描画テスト画面に移行します。



＜キーボックス無し描画テスト画面＞

描画テスト画面のキー入力

[Home] :補正画面 ・・・・・・・・・・・ 補正画面に戻り補正をやり直します。

[Tab] :カーソル表示・・・・・・・・・・画面上のカーソルの表示をon/off します。

[Delete] :画面消去 ・・・・・・・・・・・画面上に描画した線を消去します。キーガイダンス部が上下します。

[Insert] :キーボックス・・・・・・・・・キーボックスを表示させ、キーボックス有り描画テスト画面に切り替わります。

[Enter] :終了 ・・・・・・・・・・・・・・ 補正プログラムを終了します。

4、座標回転設定

4.1 レジストリ変更方法

座標回転設定のレジストリ変更方法は、レジストリ内の「SpinTop」の値を変更することで座標原点をローテーションさせることができます。その手順を以下に示します。

① 「スタート」ボタンをクリックし「ファイル名を指定して実行」を開きます。
② 名前の欄に「regedit」と入力します。
③ 「OK」ボタンを押して「レジストリエディタ」を開きます。
④ 「HKEY_LOCAL_MACHINE\Software\Fujitsu Takamisawa\Serial」の「SpinTop」の値を以下のように変更します。*12)

・座標回転の設定値

プライマリ画面の設定

0x00000000 :座標回転を行わない。
0x00000001 :右に 90 度回転する。
0x00000002 :右に 180 度回転する。
0x00000003 :右に 270 度回転する。

セカンダリ画面の設定

0x00000000 :座標回転を行わない。
0x00000010 :右に 90 度回転する。
0x00000020 :右に 180 度回転する。
0x00000030 :右に 270 度回転する。

＊PNP タッチパネルドライバの場合、セカンダリ画面用の設定はできません。

⑤ システムを再起動します。

*12）Windows のバージョンによって以下のレジストリ内の「SpinTop」を変更します。

・WindowsNT 用 NonPNP タッチパネルの場合：
「HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\FidmousP\Parameters」
「HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\FidmousS\Parameters」
の両方の「SpinTop」を同じ値に変更します。

・Windows95/98/Me 用 NonPNP タッチパネルの場合：
「HKEY_LOCAL_MACHINE\Software\Fujitsu Takamisawa\Serial」

・Windows95/98/Me 用 PNP タッチパネルの場合：
「HKEY_LOCAL_MACHINE\Software\Fujitsu Takamisawa\SerialPnP」

・Windows2000/XP 用 NonPNP タッチパネルの場合：
「HKEY_LOCAL_MACHINE\Software\Fujitsu Takamisawa\FIDTSERV」

・Windows2000/XP 用 PNP タッチパネルの場合：
「HKEY_LOCAL_MACHINE\ System\CurrentControlSet\Services\FIDMOUS\Parameters」

4.2 外部アプリケーションから座標原点をローテーションさせる方法

レジストリ設定のように外部アプリケーションが通信することにより、座標原点をローテーションさせることができます。本設定は電源再投入後にクリアされます。外部アプリケーション作成方法を以下に示します。

4.2.1 Windows95/98/Me NonPNP 用デバイスドライバ、Windows95/98/Me PNP 用ドライバ、Windows2000/XP PNP 用ドライバの場合

① CreateFile 関数を使用しドライバをオープンしてハンドルを取得します。それぞれのドライバのシンボリック名は以下のように定義されています。

・シンボリック名

Windows95/98/Me PNP 用ドライバ　：「IDMOUR」
Windows2000/XP PNP 用ドライバ　：「IDMOUS」
Windows95/98/Me NonPNP 用デバイスドライバ　（プライマリ）：「IDMOUA」
Windows95/98/Me NonPNP 用デバイスドライバ　（セカンダリ）：「IDMOUSR」

② DeviceIoControl 関数を使用して、ドライバとの通信を行い座標回転の設定、現在の座標回転情報を取得します。それぞれのコントロールコード、及び設定値は以下のように定義されています。この関数を使用する場合は「winioctl.h」が必要です。

・コントロールコード

＜Windows95/98/Me PNP 用ドライバ、Windows95/98/Me NonPNP 用デバイスドライバの場合＞

座標回転の設定：
0x7000000c
現在の座標回転情報の取得：
0x7000000f

＜Windows2000/XP PNP 用ドライバの場合＞

座標回転の設定：
CTL_CODE(FILE_DEVICE_UNKNOWN, 0x0841, METHOD_BUFFERED, FILE_ANY_ACCESS)
現在の座標回転情報の取得：
CTL_CODE(FILE_DEVICE_UNKNOWN, 0x0840, METHOD_BUFFERED, FILE_ANY_ACCESS)

・座標回転の設定値
プライマリ画面の設定
0x00000000 :座標回転を行わない。
0x00000001 :右に 90 度回転する。
0x00000002 :右に 180 度回転する。
0x00000003 :右に 270 度回転する。

セカンダリ画面の設定
0x00000000 :座標回転を行わない。
0x00000010 :右に 90 度回転する。
0x00000020 :右に 180 度回転する。
0x00000030 :右に 270 度回転する。

＊PNP タッチパネルドライバの場合、セカンダリ画面用の設定はできません。

③ CloseHandale 関数を使用してオープンしているドライバのハンドルをクローズします。

4.2.2 WindowsNT4.0 用デバイスドライバの場合

① CreateFile 関数を使用しドライバをオープンしてハンドルを取得します。ドライバのシンボリック名は以下のように定義されています。

・シンボリック名
「NTMOU」

② DeviceIoControl 関数を使用して、ドライバとの通信を行いドライバ ID の判別を行うことによってドライバのユニット ID を取得します。コントロールコード及びプライマリとセカンダリのドライバ ID は以下のように定義されています。

・コントロールコード
CTL_CODE(FILE_DEVICE_SERIAL_MOUSE_PORT, 0x0800, METHOD_BUFFERED, FILE_ANY_ACCESS)

・ドライバ ID
プライマリ ID :7580
セカンダリ ID :6800

③ DeviceIoControl 関数を使用して、タッチパネルドライバとの通信を行い座標回転の設定、現在の座標回転情報を取得します。コントロールコード、及び設定値は以下のように定義されています。この関数を使用する場合は「winioctl.h」が必要です。

・コントロールコード

座標回転の設定：

CTL_CODE(FILE_DEVICE_SERIAL_MOUSE_PORT, 0x080c, METHOD_BUFFERED, FILE_ANY_ACCESS)

現在の座標回転情報の取得：

CTL_CODE(FILE_DEVICE_SERIAL_MOUSE_PORT, 0x080f, METHOD_BUFFERED, FILE_ANY_ACCESS)

・座標回転の設定値

プライマリ画面の設定

0x00000000 :座標回転を行わない。
0x00000001 :右に 90 度回転する。
0x00000002 :右に 180 度回転する。
0x00000003 :右に 270 度回転する。

セカンダリ画面の設定

0x00000000 :座標回転を行わない。
0x00000010 :右に 90 度回転する。
0x00000020 :右に 180 度回転する。
0x00000030 :右に 270 度回転する。

④ CloseHandale 関数を使用してオープンしているドライバのハンドルをクローズします。

4.2.3 Windows98/Me/NT4.0 NonPNP 用ユーザーモードドライバ、Windows2000/XP NonPNP 用ドライバの場合

① LoadLibrary 関数を使用し、座標回転用 DLL（SpinDll.dll）のハンドルを取得します。
② GetProcAddress 関数を使用して、座標回転用 DLL がエクスポートしている座標回転設定するための関数（FPHOOK_SetSPINTOP）、現在の座標回転情報の取得するための関数（FPHOOK_GetSPINTOP）のアドレスを取得します。
③ 取得した関数のアドレスを用いて、座標回転の設定、現在の座標回転の情報を取得します。設定値は以下のように定義されています。

・座標回転の設定値

プライマリ画面の設定

0x00000000 :座標回転を行わない
0x00000001 :右に 90 度回転する
0x00000002 :右に 180 度回転する
0x00000003 :右に 270 度回転する

セカンダリ画面の設定

0x00000000 :座標回転を行わない
0x00000010 :右に 90 度回転する
0x00000020 :右に 180 度回転する
0x00000030 :右に 270 度回転する

④ FreeLibrary 関数を使用して座標回転用 DLL のハンドルを無効にします。

4.3 プログラム例

● Windows95/98/Me 用 PNP シリアルタッチパネルドライバ

```
//必要なヘッダファイル
#include <winioctl.h>

//コントロールコード：
#define             FTlOCTL_9X_SetSpinTop                   0x7000000c
#define             FTlOCTL_9X_GetSpinTop                   0x7000000f

//変数
HANDLE hKeD;                                        //取得するタッチパネルドライバのハンドル
ULONG cbReturned;
typedef struct      tagDANGLE_98
{
        ULONG     Degree;                           //送受信する座標回転用の変数
}DANGLE_98, *PDANGLE_98;
DANGLE_98                   DAngle_98;

// Windows95/98/Me 用 PNP シリアルタッチパネルドライバでシンボリック名は FIDMOUR と定義されています。
  hKeD= CreateFile("\\\\.\\FIDMOUR",0,0,NULL,0,0,NULL);

//座標回転の設定
DAngle_98.Degree = 0x1;                             //0 :0 °   1 :90 °   2 :180 °   3 :270 °
(DeviceIoControl(hKeD   ,FTlOCTL_9X_SetSpinTop
                ,&DAngle_98   ,sizeof(DANGLE_98)
                ,NULL ,0    ,&cbReturned,  NULL));

//現在の座標回転情報の取得
(DeviceIoControl(hKeD,  FTlOCTL_9X_GetSpinTop
                ,NULL   ,0     ,&DAngle_98
                  ,sizeof(DANGLE_98)
                ,&cbReturned,NULL));

//ドライバのハンドルを閉じる
CloseHandle (hKeD);
```

● Windows2000/XP 用 PNP シリアルタッチパネルドライバ

```
//必要なヘッダファイル
#include <winioctl.h>

//コントロールコード：
#define FTlOCTL_2K_GetSpinTop
CTL_CODE(FILE_DEVICE_UNKNOWN, 0x0840, METHOD_BUFFERED, FILE_ANY_ACCESS)

#define FTlOCTL_2K_SetSpinTop
CTL_CODE(FILE_DEVICE_UNKNOWN, 0x0841, METHOD_BUFFERED, FILE_ANY_ACCESS)

//変数
HANDLE hKeD;                                    //取得するタッチパネルドライバのハンドル
char completeDeviceName[] = "\\\\.\\FIDMOUS";
ULONG cbReturned;
ULONG SpinTop ;                                 //送信する座標回転用の変数
ULONG SpinData;                                 //受信する座標回転用の変数

// Windows2000/XP 用 PNP シリアルタッチパネルドライバでシンボリック名はFIDMOUS と定義されています。
hKeD = CreateFile(completeDeviceName,
                  GENERIC_READ ¦ GENERIC_WRITE,
                  FILE_SHARE_READ ¦ FILE_SHARE_WRITE,
                  NULL,
                  OPEN_EXISTING, 0, NULL);

//座標回転の設定
SpinTop  =  0x1;                                //0 :0 °  1 :90 °  2 :180 °  3 :270 °
DeviceIoControl(hKeD, FTlOCTL_2K_SetSpinTop,
                &SpinTop, sizeof(ULONG),
                NULL, 0,
                &cbReturned, NULL);

//現在の座標回転情報の取得
DeviceIoControl(hKeD, FTlOCTL_2K_GetSpinTop,
                NULL, 0,
                &SpinData, sizeof(ULONG),
                &cbReturned, NULL);

//ドライバのハンドルを閉じる
CloseHandle(hKeD);
```

● Windows95/98/Me 用 NonPNP シリアルタッチパネルドライバ (デバイスドライバ)

```
//必要なヘッダファイル
#include <winioctl.h>

//コントロールコード：
#define         FTIOCTL_9X_SetSpinTop                   0x7000000c
#define         FTIOCTL_9X_GetSpinTop                   0x7000000f
//使用する変数
HANDLE hKeD;                            //取得するドライバのハンドル
ULONG cbReturned;

typedef struct      tagDANGLE_98
{
        ULONG   Degree;                 //送受信する座標回転用の変数
}DANGLE_98, *PDANGLE_98;

DANGLE_98                   DAngle_98;

// Windows95/98/Me 用 NonPNP シリアルタッチパネルプライマリドライバでシンボリック名は FIDMOUA と
//定義されています。
  hKeD= CreateFile("\\\\.\\FIDMOUA",0,0,NULL,0,0,NULL);

//座標回転の設定
DAngle_98.Degree = 0x01;                //下 1 桁 (0byte)目が画面 1 の設定値
                                        //下 2 桁 (1byte)目が画面 2 の設定値
                                        //0 :0 °  1 :90 °  2 :180 °  3 :270 °

(DeviceIoControl(hKeD  ,FTIOCTL_9X_SetSpinTop
                ,&DAngle_98  ,sizeof(DANGLE_98)
                ,NULL ,0    ,&cbReturned,  NULL));

//現在の座標回転情報の取得
(DeviceIoControl(hKeD,  FTIOCTL_9X_GetSpinTop
                ,NULL  ,0     ,&DAngle_98
                 ,sizeof(DANGLE_98)
                ,&cbReturned,NULL));

//ドライバのハンドルを閉じる
CloseHandle (hKeD);
```

＊Windows95/98/Me 用 PNP シリアルタッチパネルセカンダリドライバではシンボリック名を FIDMOUSR と定義してあります。そのため、CreateFile を以下のように実行します。座標回転の設定、現在の座標回転情報の取得方法はプライマリの場合と同じです。

```
hKeD= CreateFile("\\\\.\\FIDMOUSR",0,0,NULL,0,0,NULL);
```

●WindowsNT 用 シリアルタッチパネルドライバ (デバイスドライバ)

```
//必要なヘッダファイル
#include <winioctl.h>

//コントロールコード：
#define FTIOCTL_NT_GETID
CTL_CODE(FILE_DEVICE_SERIAL_MOUSE_PORT, 0x0800, METHOD_BUFFERED, FILE_ANY_ACCESS)

#define FTIOCTL_NT_ANGLECHANGE
CTL_CODE(FILE_DEVICE_SERIAL_MOUSE_PORT, 0x080c, METHOD_BUFFERED, FILE_ANY_ACCESS)

#define FTIOCTL_NT_GETSPINTOP
CTL_CODE(FILE_DEVICE_SERIAL_MOUSE_PORT, 0x080f, METHOD_BUFFERED, FILE_ANY_ACCESS)

//NT ドライバ ID
#define PrimaryID           7580                    //プライマリドライバ ID
#define SecondaryID         6800                    //セカンダリドライバ ID

//変数
HANDLE hKeD;                                        //取得するタッチパネルドライバのハンドル
ULONG cbReturned;

typedef struct   tagMOUSE_PRIVATE_IN {
        USHORT unitId;
        UCHAR    code[20];
} MOUSE_PRIVATE_IN, *PMOUSE_PRIVATE_IN;

typedef struct   tagMOUSE_PRIVATE_OUT {
        USHORT unitId;
        UCHAR    code[20];
} MOUSE_PRIVATE_OUT, *PMOUSE_PRIVATE_OUT;

typedef struct      tagDANGLE{
        USHORT   unitId;
        ULONG    Degree;                            //送受信する座標回転用の変数
}DANGLE, *PDANGLE;

DANGLE                  DAngle;
MOUSE_PRIVATE_IN        privatein;
MOUSE_PRIVATE_OUT       privateout;

USHORT          Primary_unitId;                     //プライマリユニットID 用変数
USHORT          Secondary_unitId;                   //セカンダリユニットID 用変数
char            completeDeviceNameNT[] = "\\\\.\\NTMOU";
```

```
// WindowsNT 用シリアルタッチパネルドライバでシンボリック名はNTMOU と定義されています。
hKeD = CreateFile ( completeDeviceNameNT,
                    GENERIC_READ | GENERIC_WRITE,
                    0, NULL,
                    OPEN_EXISTING ,0, NULL);

//UnitID の取得
for (privatein.unitId=0; privatein.unitId <=3; ++(privatein.unitId))
{
        if (DeviceIoControl (hKeD, (DWORD)FTIOCTL_NT_GETID,
                             &privatein, sizeof(MOUSE_PRIVATE_IN),
                             &privateout, sizeof(MOUSE_PRIVATE_OUT),
                             &cbReturned, NULL))
        {
                if(privateout.unitId == PrimaryID)          //ドライバ ID と比較を行い、
                        Primary_unitId = privatein.unitId;  //それぞれの、ユニットID を
                else if (privateout.unitId == SecondaryID)  //保存します。
                        Secondary_unitId = privatein.unitId;
        }
}

//座標回転の設定
DAngle.Degree = 0x01;                                   //下1桁 (0byte)目が画面1の設定値
                                                        //下2桁 (1byte)目が画面2の設定値
                                                        //0:0°  1:90°  2:180°  3:270°
DAngle.unitId = Primary_unitId;
(DeviceIoControl (hKeD, (DWORD)FTIOCTL_NT_ANGLECHANGE,
                    &DAngle,   sizeof(DANGLE),
                    NULL, 0,
                    &cbReturned, NULL));

//現在の座標回転情報の取得
 (DeviceIoControl (hKeD, (DWORD)FTIOCTL_NT_GETSPINTOP,
                    NULL,0,
                    &DAngle, sizeof(DANGLE),
                    &cbReturned, NULL));

//ドライバのハンドルを閉じる
CloseHandle (hKeD);
```

＊WindowsNT 用シリアルタッチパネルセカンダリドライバでは Secondary_unitId を使用して座標回転の設定、現在の座標回転情報の取得を行ってください。

● Windows98/Me/NT4.0/200/XP 用 NonPNP シリアルタッチパネルドライバ (ユーザーモードドライバ)

```
//変数
HINSTANCE       hFPSpin;
ULONG           SpinTop ;                       //送信する座標回転用の変数
ULONG           SpinData;                       //受信する座標回転用の変数

typedef void (*pFPSPIN_SetSPINTOP)(DWORD);
typedef DWORD (*pFPSPIN_GetSPINTOP)();

pFPSPIN_SetSPINTOP      FPSPIN_SetSPINTOP;
pFPSPIN_GetSPINTOP      FPSPIN_GetSPINTOP;

//SpinDll.DLL をロードする。
hFPSpin= LoadLibrary("SPINDLL.DLL");

// SpinDll.DLL 内の関数のアドレスを取得する。
FPSPIN_SetSPINTOP = (pFPSPIN_SetSPINTOP)GetProcAddress (hFPSpin, "FPHOOK_SetSPINTOP");
FPSPIN_GetSPINTOP = (pFPSPIN_GetSPINTOP)GetProcAddress (hFPSpin, "FPHOOK_GetSPINTOP");

//座標回転の設定
SpinTop  =  0x01;                               //下 1 桁 (0byte)目が画面 1 の設定値
                                                //下 2 桁 (1byte)目が画面 2 の設定値
                                                //0:0° 1:90° 2:180° 3:270°
(*FPSPIN_SetSPINTOP)(SpinTop);

//現在の座標回転情報の取得
SpinData = ((*FPSPIN_GetSPINTOP)());

//SpinDll.DLL をクローズする。
FreeLibrary(hFPSpin);
```