GOP チュートリアル

画面設計からホストプログラミングまで

改定履歴

改定番号	改定年月日	改定内容	担当	承認
－	2010 年 7 月 29 日	初版	島田	藤本
A	2012 年 8 月 29 日	GOP-5000 シリーズ対応	島田	藤本
備考				

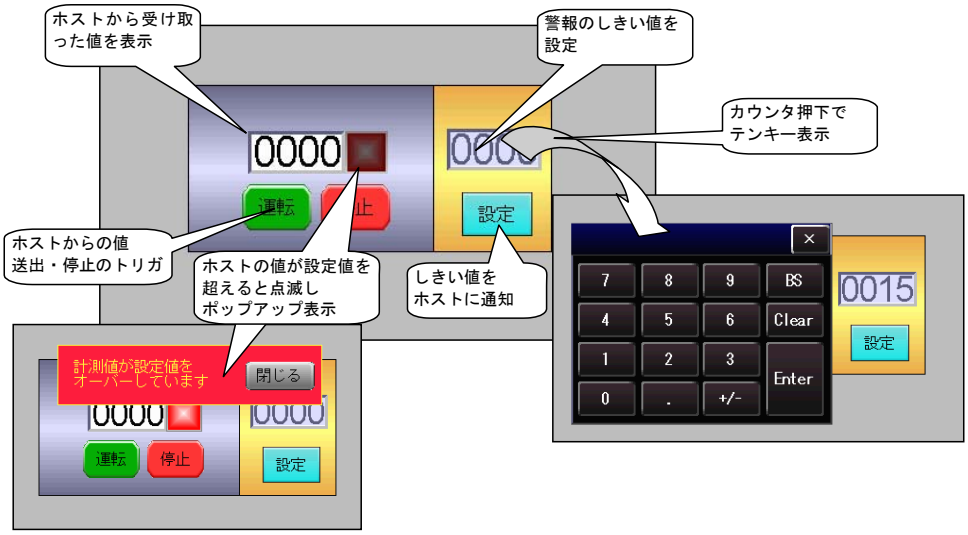
目次

1. 本書の概要

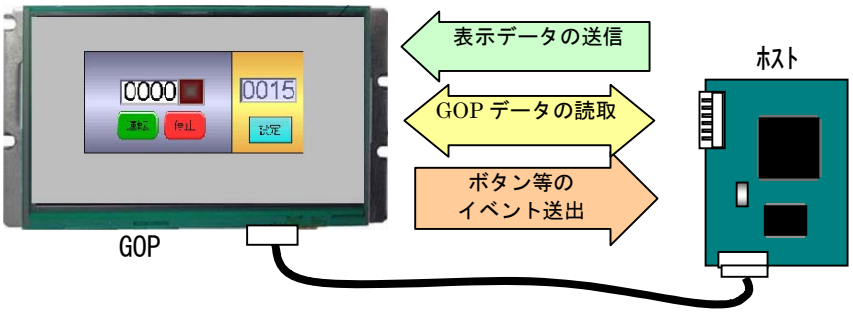
GOP を使って、簡単な画面設計から、ホストマイコンとの通信までの流れを説明いたします。本書を一読していただくことで GOP を使用したシステムの構築の仕方をご理解していただくことを目的としています。

(1) 本書で作成する画面のデザインと機能

本書の説明例として GOP 側は以下のような動作・画面デザインとします。



上記の画面設計にて
・基本的な作画手順
・ボタンなどの動作の設定方法
・テンキーの使用方法
などについて学習いたします。
また GOP とホストマイコンとの通信は以下のとおりです。



・ホストからのデータ書込み
・ホストから GOP にデータの要求(読み取り)
・ホストへ GOP のイベントの通知
システムを構築する上で最低限必要な三要素について学習いたします。

(2) 作業の進め方

以下の順番にて作業を進めます

①画面設計

TP デザイナーを使用した画面の作成方法について学習します。
本書の第2章にて説明します。

②画面データの転送

GOP へのデータ書込み方法を学習します。
本書の第3章にて説明します。

③GOP との通信、単体での動作

GOP と TP デザイナーの通信フォームを使用して通信し、コマンドの内容及びそれによる GOP の動作について学習します。
本書の第4章にて説明します。

④ホストのプログラミング

GOP ライブラリ(詳しくは GOP アプリケーションノートをご参照ください)を使用して GOP と通信し連係動作するホストアプリケーションの作成を行います。
本書の第5章にて説明します。
なお、本章ではホストボードの代わりとして、GOP アプリケーションノートで使用している仮想ホストを使用します。
仮想ホスト・GOP ライブラリ・サンプルアプリケーションは Microsoft 社の VisualStudio6 または VisualStudio2008 で動作するソースファイル一式を添付しています。
なお、上記サンプルは Microsoft 社が無償で提供している VisualC++2008ExpressEdition でも動作いたします。これを使用して、VisualStudio 製品をお持ちでないお客様も本チュートリアルを実行可能です。
以下の URL にてダウンロード可能です。
http://www.microsoft.com/japan/msdn/vstudio/2008/product/express/
※上記は Microsoft 社の製品であり、これについてのご質問などはお受付できません。
※ユーザー登録はお客様にて行ってください。弊社にてのサポートは行いません。
※上記 URL は 2010 年 7 月現在のものです。将来的には削除または移動される可能性があります。その場合お客様にて再検索してください。弊社にての再確認は行いません。

2. 画面設計

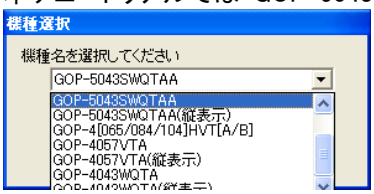
画面設計の流れとして
①使用するメモリの構成の決定
②オブジェクトの配置及び設定
③マルチアクション/マクロを使用した動作の設定
の順で作業を進めます。
実際の作業においてこの順で進める必要は特にありません。
(説明上、上記の順としています)

2.1 画面設計の準備

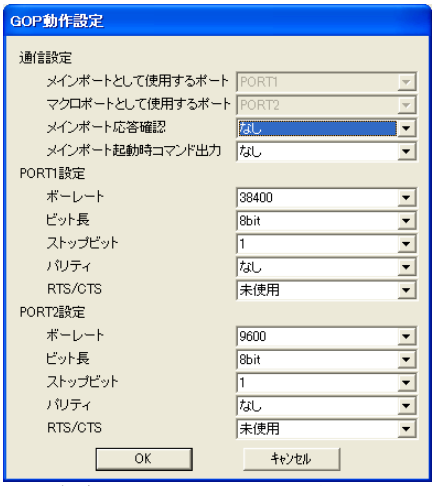
GOP の画面設計は専用の画面設計ソフト TP デザイナーを使用して行います。
TP デザイナーの起動は以下の手順で行います。

①TP デザイナーの起動
TP デザイナーがインストールされるとディスクトップ上に以下のアイコンが作成されています。
このアイコンをダブルクリックしてください。

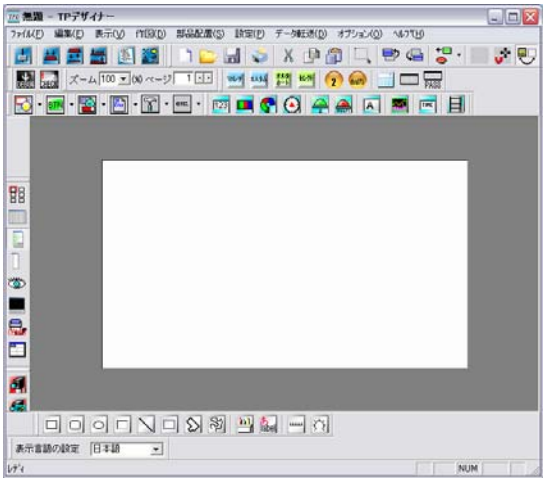
②GOP 機種の選択
リストボックスの中から設計する GOP のシリーズを選択します。
本チュートリアルでは"GOP-5043SWQTAA"を選択してください。



③GOP の初期設定画面が表示します。
本チュートリアルでは標準設定のまま進めますのでそのまま OK ボタンを押してください。

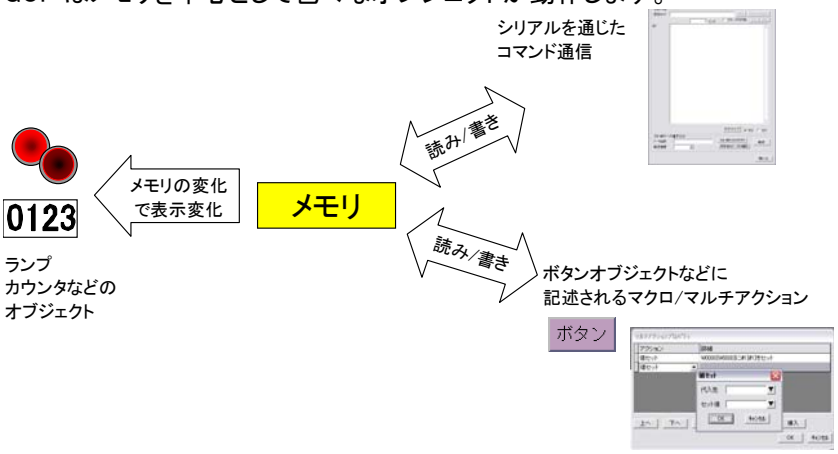


④TP デザイナーの編集画面が表示します。
これで画面設計の準備は終了です。



2.2 メモリの構成

GOP はメモリを中心として色々なオブジェクトが動作します。



カウンタの場合、カウンタにリンクするメモリを決め、そのメモリに対しホストから値を書き込むことで、表示を切替えることが出来ます。
また GOP のボタン操作などでメモリに値を書き込むと、ホストがその値をコマンドを使用して読むことも出来ます。

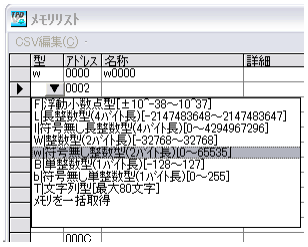
本チュートリアルのシステムにおいては
・ホストが書き込んだ値を格納するメモリ(左側のカウンタにリンクします)。
・GOP で設定しホストに読んでもらうためのメモリ(右側のカウンタにリンクします)。
上記のメモリが必要になります。
そのため TP デザイナーで 2 つのメモリを取得します。

メモリの取得手順は以下のとおりです。

①メモリリストを開きます。(以下のアイコンで開きます)

②一つ目のメモリを取得します。
アドレスを決め、型を指定します。
今回の例ではアドレスは 0000 番地にします。
型は 4 桁の整数のため w 型を指定します。

③2 つ目のメモリも②と同様に取得します。
アドレスは 0002 番地にします。



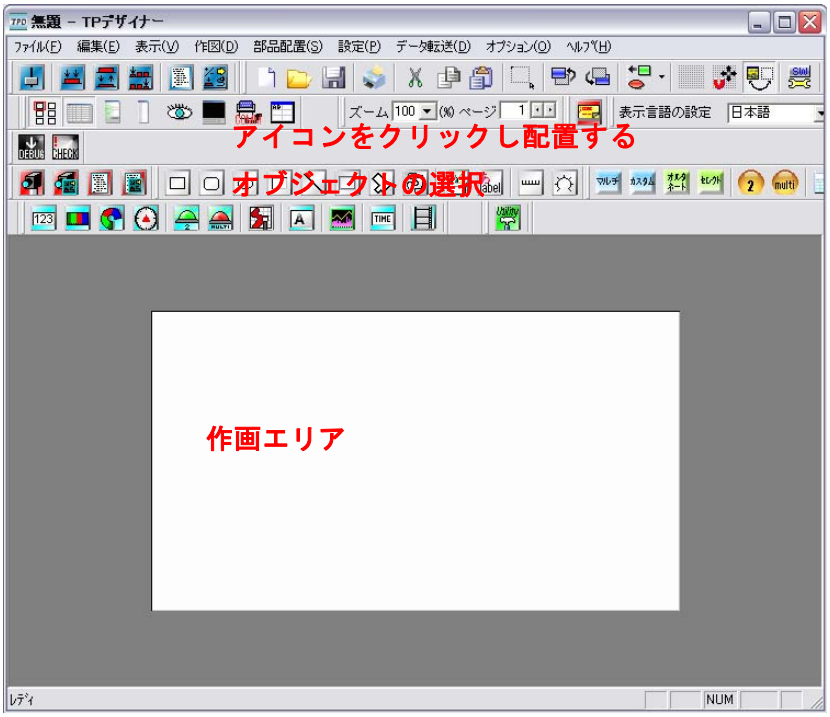
以上でメモリの取得は終了です。

メモリ名称はデフォルト(型+番地)のままでもかまいませんが、メモリの数が増えてくると用途がわかる名称をつけていたほうが、マクロなどで使用する場合にわかりやすいです。
※ホストからは型+アドレスでのやり取りになります。
本チュートリアルでは名称はデフォルトのままで進めます。

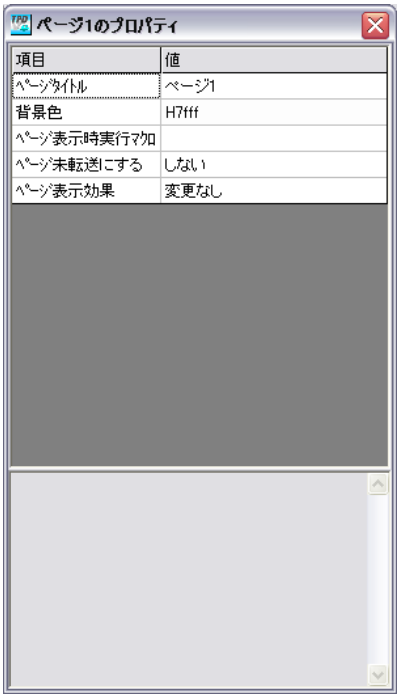
また、一度決めたメモリを変更する場合、型サイズが変る場合など作業量が多くなります。
※例えば w 型を I 型に変更する場合、アドレスが 2 個⇒4 個となるため後ろにメモリが詰まっている場合それらを全てずらす等の対応が必要。並びがばらばらになるが空いているアドレスに取り直してもよいです。

2.3 オブジェクトの配置

画面の設計は、配置するオブジェクトを選択し、作画エリアに配置するという手順で行います。



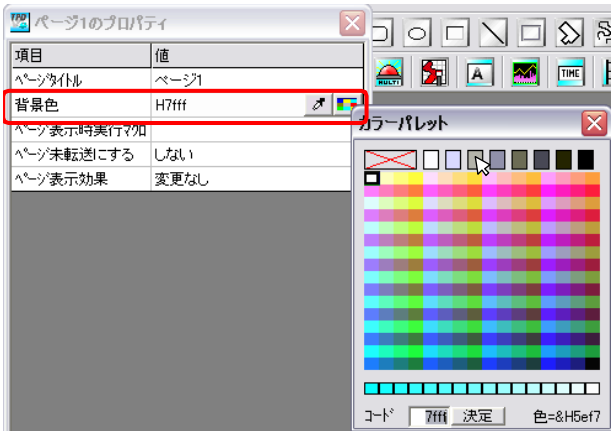
オブジェクトの設定はプロパティシートで行います。



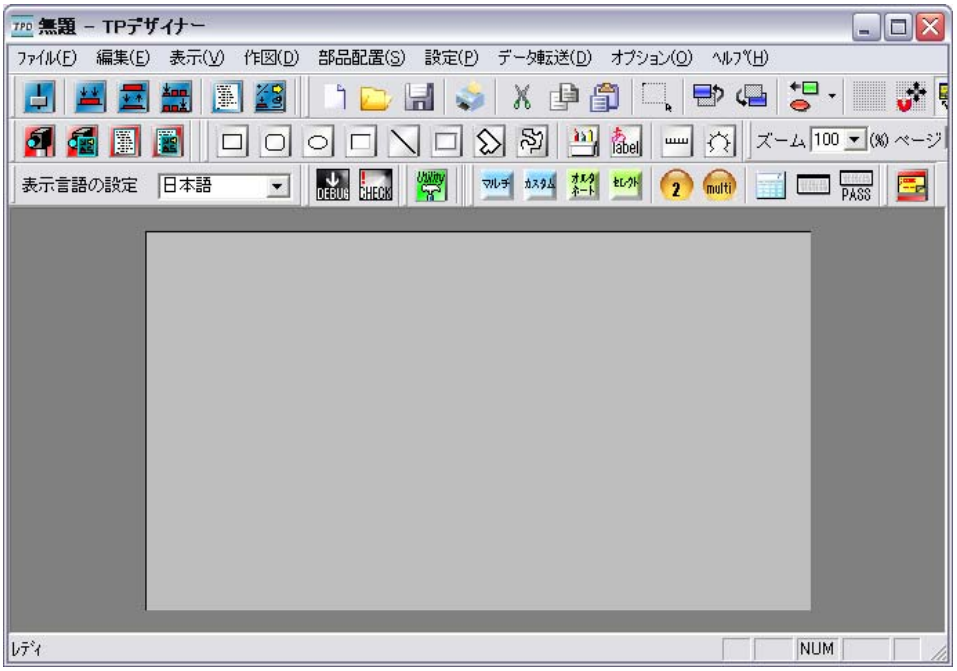
プロパティシートが表示されていない場合、以下のアイコンで表示されます。

(1) 背景色

背景色はページのプロパティで設定します。
ページのプロパティはオブジェクトを何も選択していないとき、プロパティシートで設定できます。

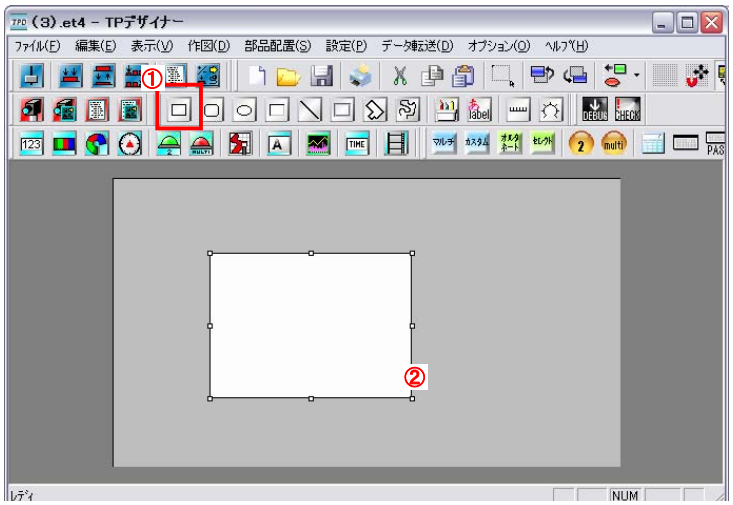


プロパティを変更すると作画エリアが変化します。



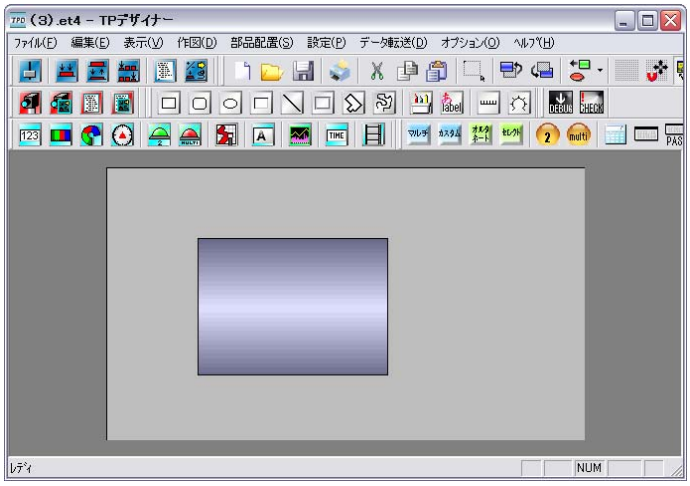
(2) デザイン用のボックスオブジェクト

ボックスなどのオブジェクトを使用して画面のデザインを行うことが出来ます。
簡単なデザインであれば、ボックス・角丸ボックスなどで行えます。凝ったデザインの場合
専用アプリケーションで画面デザインを行ないそれを BMP ファイル化し取り込むことも出来ます。
本チュートリアルではボックスを使用した例で説明します。
①ボックスオブジェクトの選択
②作画エリアに配置



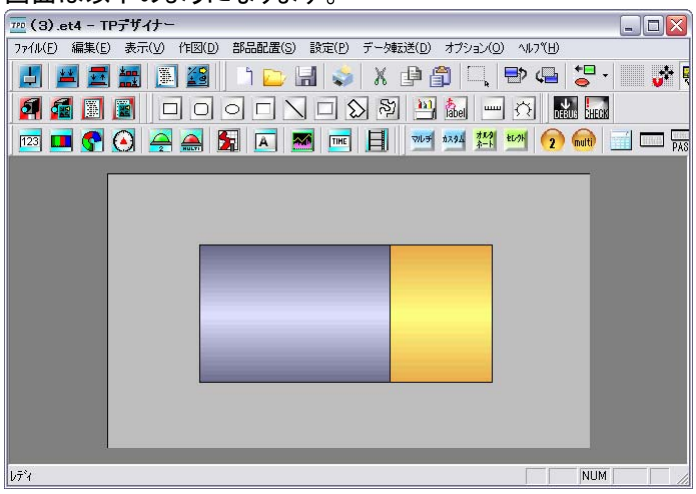
③プロパティを設定
オブジェクトを選択状態にすると、そのオブジェクトのプロパティを設定できます。
背景色を変更し、グラデーションを指定します。

プロパティ変更すると表示は下のようになります。



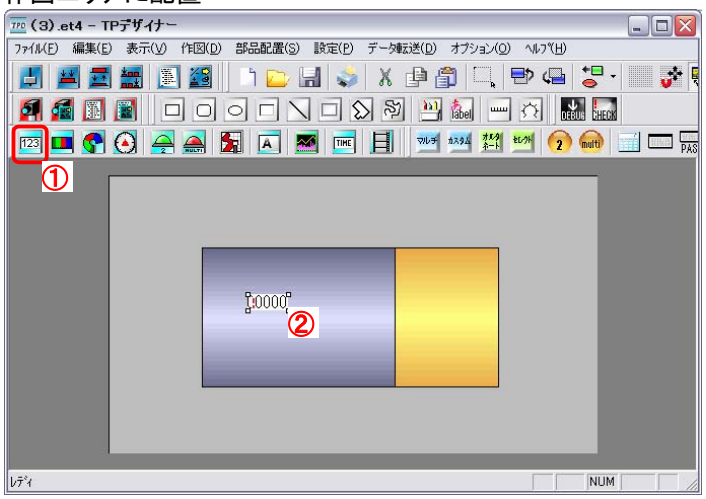
④①～③と同様の手順でもうひとつボックスを配置します。

画面は以下のようになります。



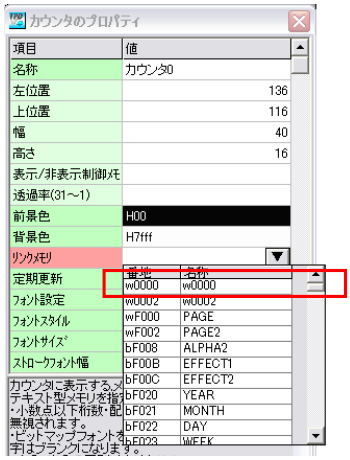
(3) カウンタ

数値表示のカウンタを配置します。
①カウンタオブジェクトの選択
②作画エリアに配置

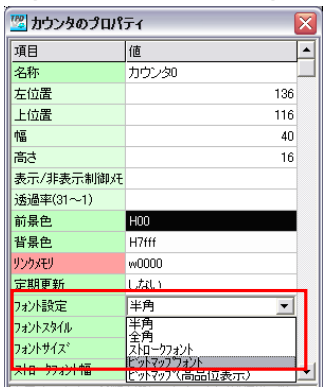


③プロパティを設定します。
プロパティは以下の内容を設定します
・リンクメモリの設定
・外観の設定(使用フォントなど)

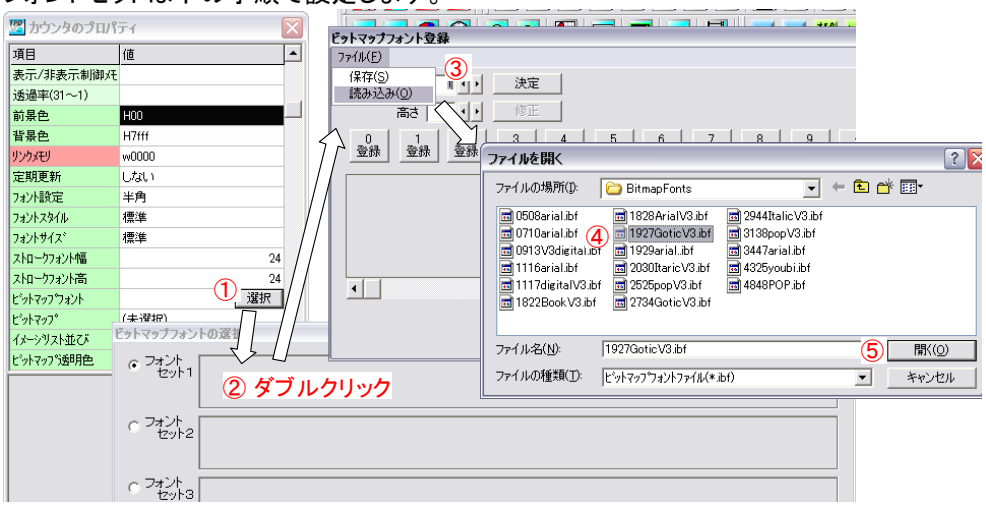
リンクメモリのドロップダウンを開くとメモリリストが表示されます。
ここから w0000 を選択します。



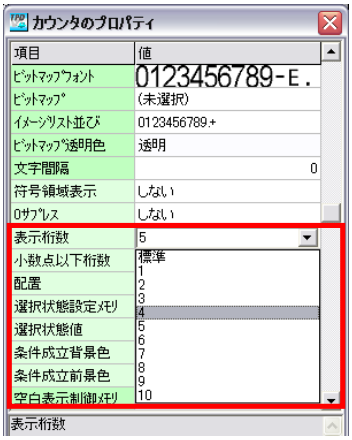
フォントはビットマップフォントを使用します。



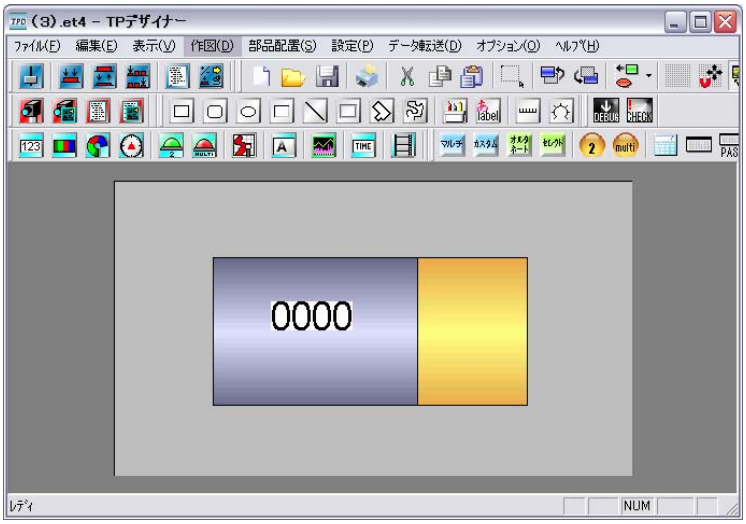
ビットマップフォントを使用する場合、使用するフォントセットを指定する必要があります。
フォントセットは下の手順で設定します。



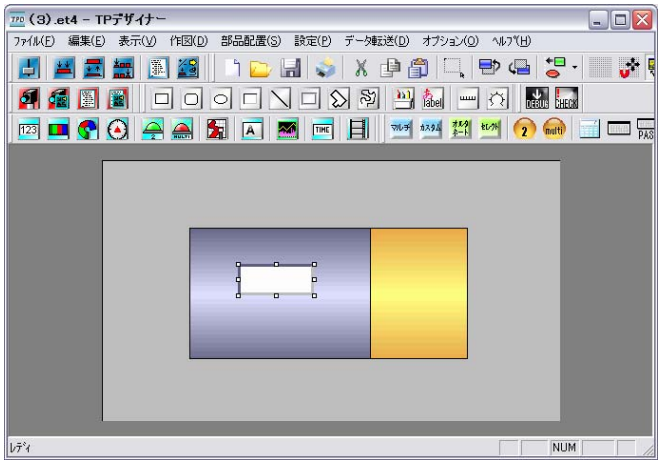
カウンタの表示桁を 4 桁に指定します。



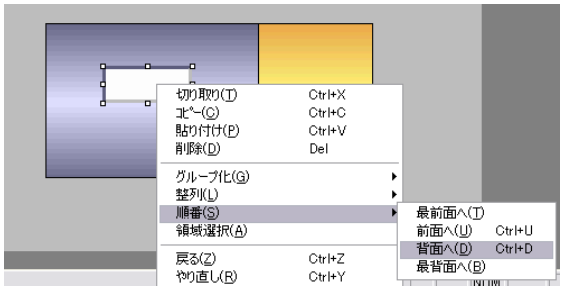
上記の設定で作画エリアは下のようになります。



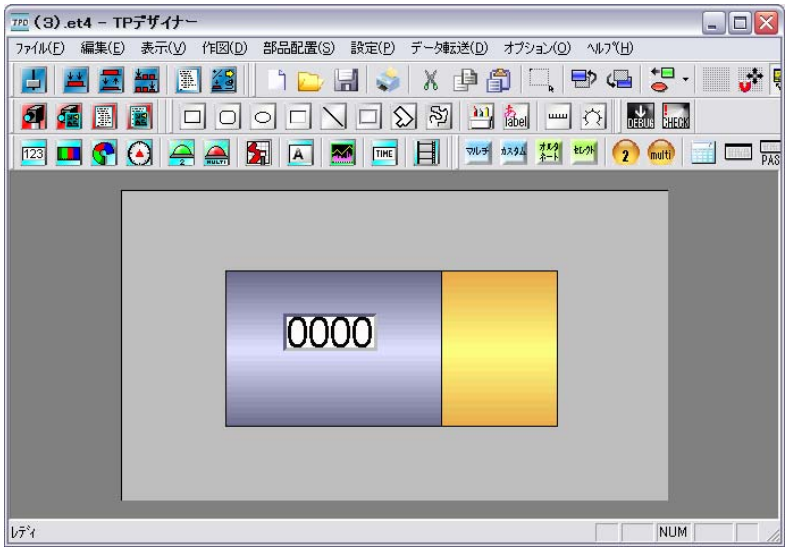
④カウンタの見栄えをよくするため、カウンタの外形に合わせて立体枠つきボックスを配置します



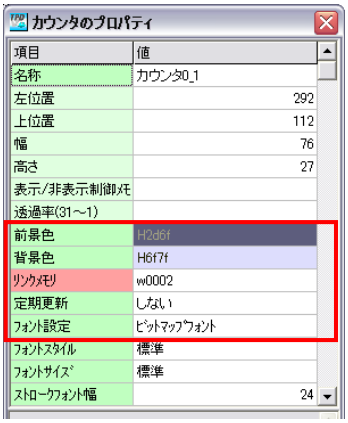
立体枠つきボックスの上で右クリックしポップアップメニューを出し、背面へを選択し並び順を変えます。



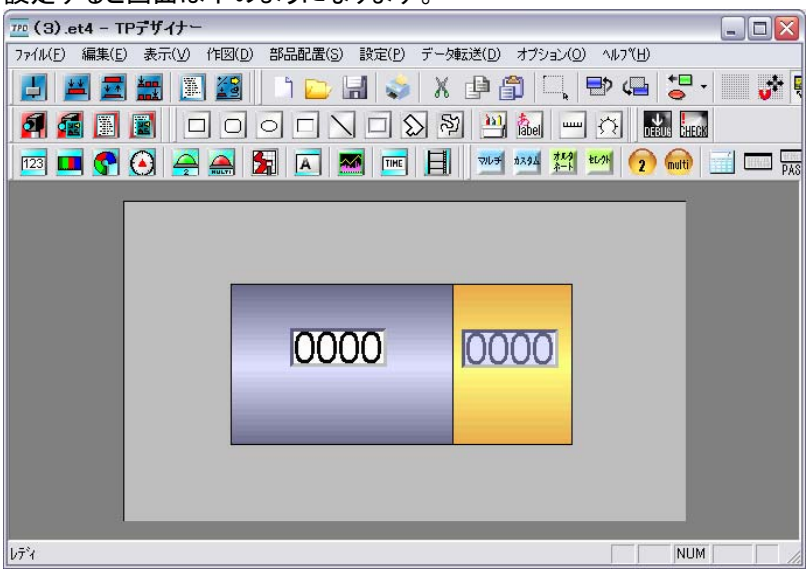
立体枠つきボックスの大きさ、位置を微調整します。



⑤同様の手順で右側のカウンタも作成します。
背景色、前景色を変更し、リンクメモリを w0002 に設定します。



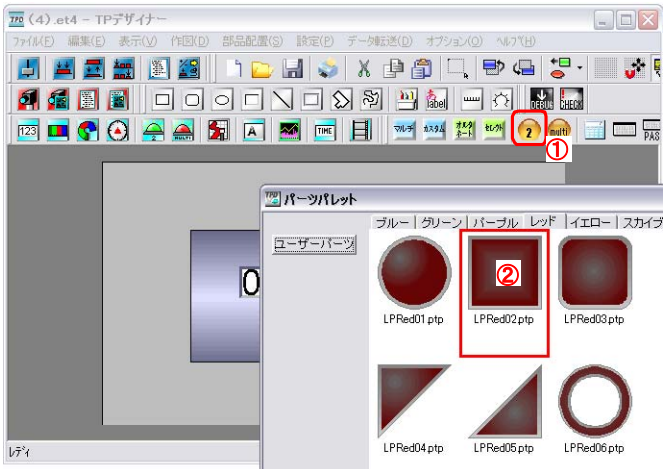
設定すると画面は下のようになります。



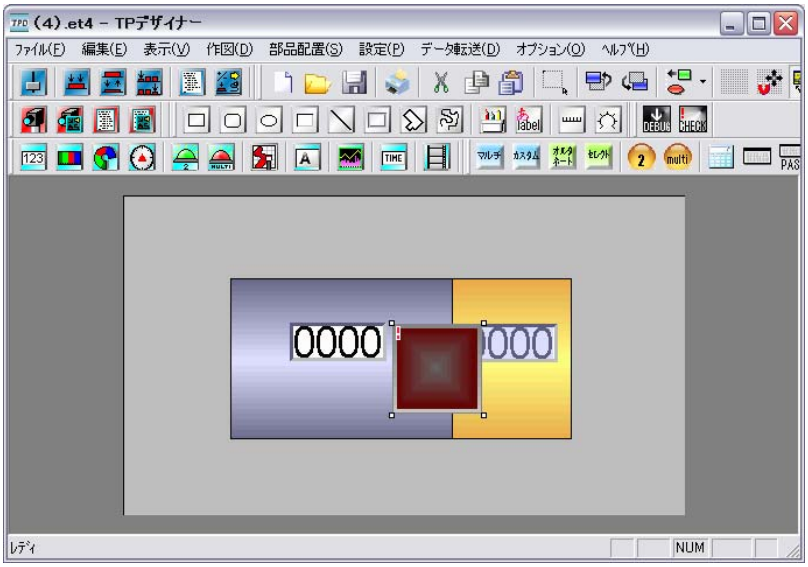
(4) ランプ

警報ランプを配置します。
ランプの動作としてはホストからの書込み値(w0000)が GOP の設定値(w0002)より大きい場合点灯、そうでない場合消灯とします。
ランプは、標準のものを配置して外観を設定できますが、ここではパーツパレット(あらかじめデザインされたビットマップが登録されたパーツ)からの選択で作成します。

パーツパレットのランプは以下の手順で配置します。
①パーツパレットを開きます
②配置したいランプをダブルクリックします。

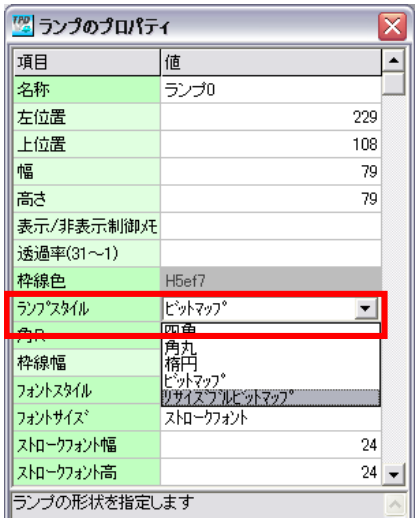


配置すると以下のようになります。

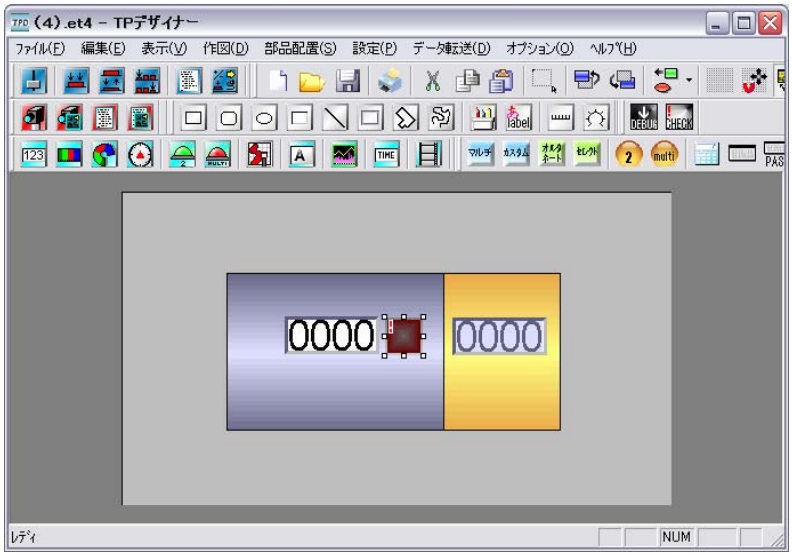


③ランプのプロパティを設定します。

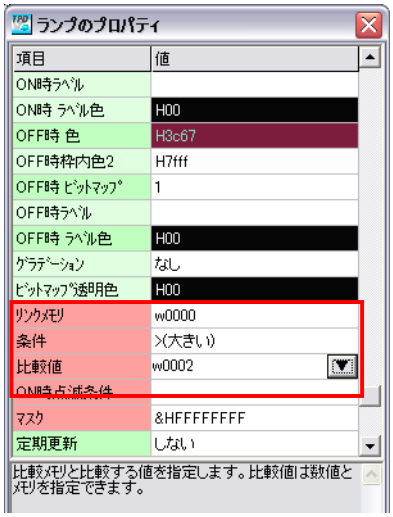
パーツは配置したままだと、サイズの調整が出来ないためランプスタイルをリサイズブルビットマップに変更します。



上記の設定を行うことでランプのサイズが変更可能になります。



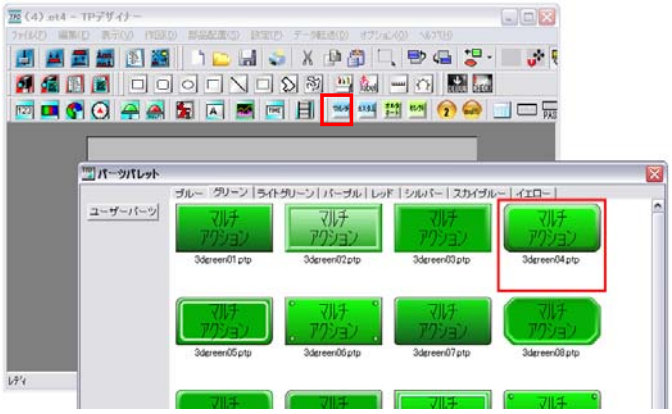
その他のプロパティを設定します
動作の設定として
リンクメモリを w0000
動作条件を>
比較値を w0002
に設定します。
これにより
w0000>w0002 の時点灯します。



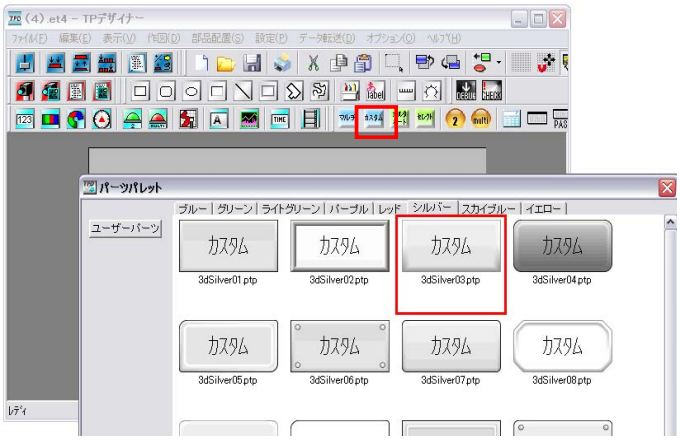
(5) ボタン

運転・停止ボタンと設定ボタンを配置します。
ランプと同様パーツパレットからの配置で行います。
後の説明のため、運転・停止ボタンはマルチアクションボタン
設定ボタンはカスタムボタンにします。
※マルチアクションは出来る動作が限られますが、選択式メニュー式で簡単に設定できます。カスタムボタンはマクロ文での動作の記述が必要ですが作り込めば複雑な動作も記述できます。

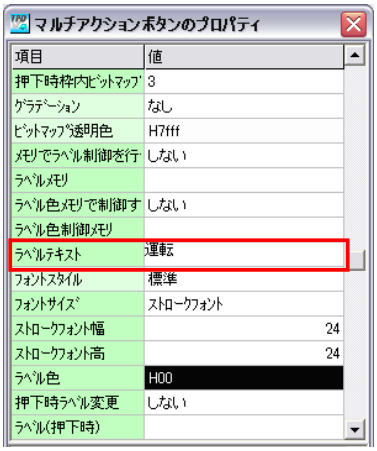
マルチアクションボタンのパーツパレット



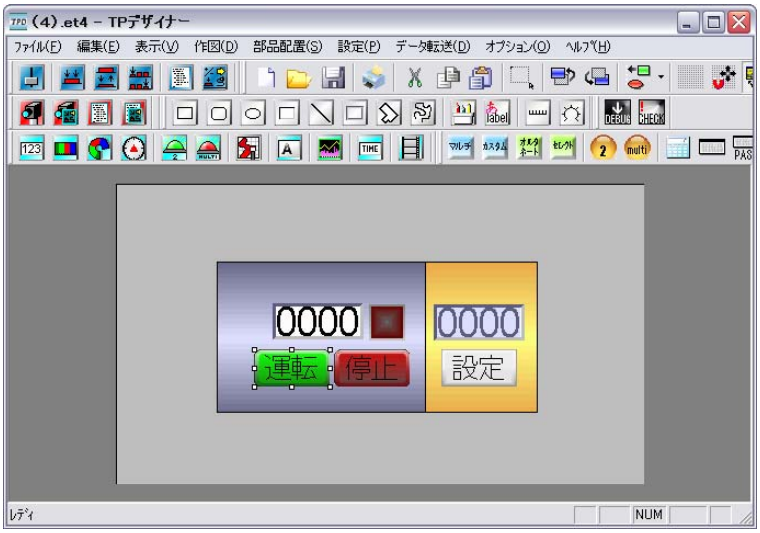
カスタムボタンのパーツパレット



ランプと同様ボタンスタイルをリサイズブルビットマップに変更し、サイズ調整しラベルテキストをそれぞれ、"運転"・"停止"・"設定"に変更します。

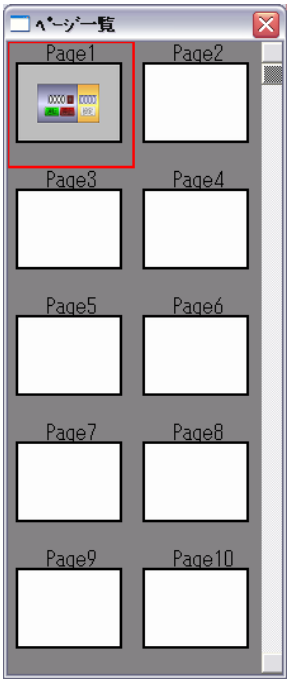


設定すると画面は下のようになります。



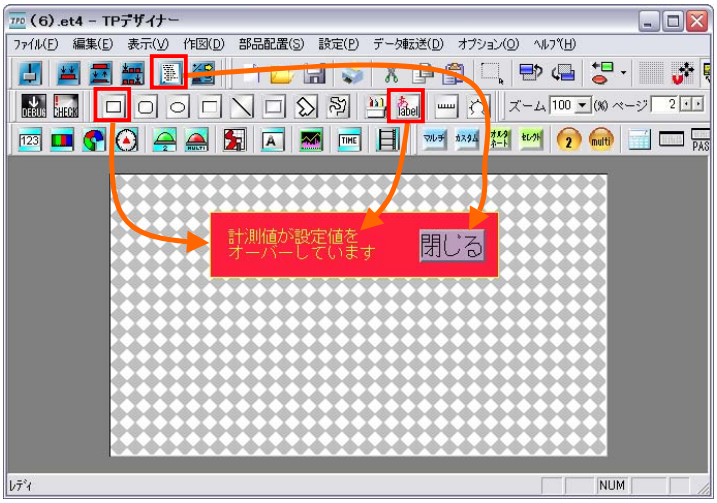
(6) ポップアップ

警告ポップアップは別ページとして作成し、レイヤー重ね合わせとして表示します。
新しく 2 ページ目にポップアップのページを作画します。
作画するページを変えるには、ページ一覧ウィンドウでページ 2 の位置をクリックしてください。



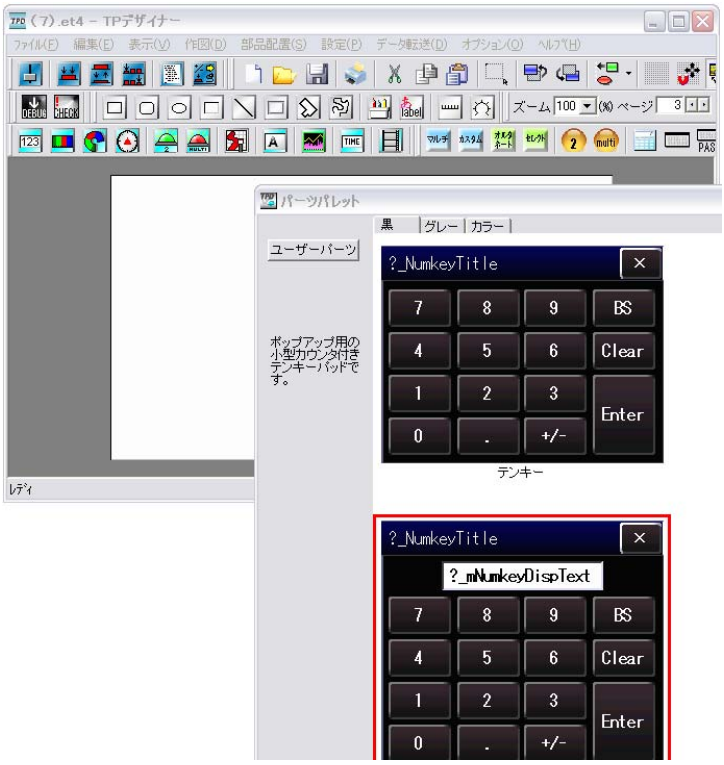
また、ページセレクタでもページ切替が出来ます。

ページ 2 では以下を配置します。
・ページ背景を透明
・ボックスを配置し色を変更
・文字オブジェクトを配置し、テキストを設定
・カスタムボタンを配置(閉じるボタン)



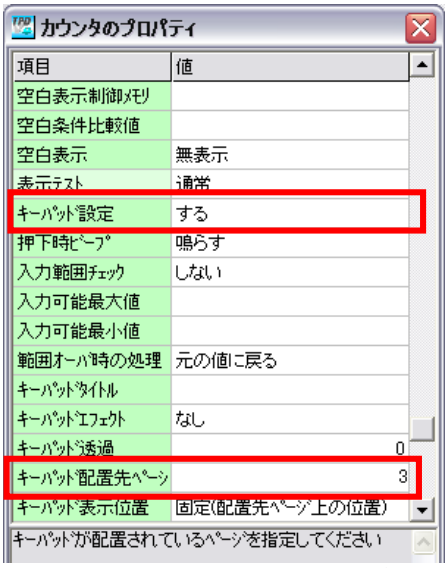
(7) テンキー

ページ1の右側のカウンタにテンキー入力を設定します。
テンキーの設定は以下の手順で行います。
①テンキーのパーツを任意の空のページに配置する。
今回は 3 ページにカウンタつきのテンキーを配置します。



テンキーを配置しページ背景を透明にします。

②1 ページの右側のカウンタにテンキーの呼び出し設定を行います。
プロパティシートで以下の設定を行います。
・キーパッド設定をするに変更
・キーパッド配置先ページを①で配置したページ(3 ページ)に設定



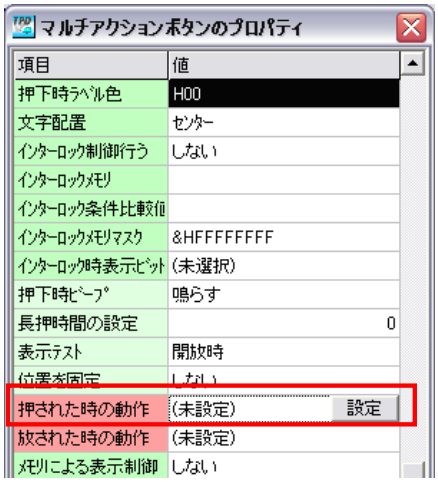
上記でカウンタを押すとテンキーが表示されるようになります。

2.4 ボタン動作の設定

ボタンの動作として以下の動作を設定します。
・運転ボタンを押すと、"RUN"という文字列をホストに送信
・停止ボタンを押すと、"STOP"という文字列をホストに送信
・設定ボタンを押すと、"SET"という文字列をホストに送信
・ページ 2 の閉じるボタンを押すと、ポップアップを閉じる

(1) 運転・停止ボタン(マルチアクションでの設定例)

運転・停止ボタンはマルチアクションでの記述で説明します。
①運転ボタンのプロパティの押下時の動作の設定を押します。

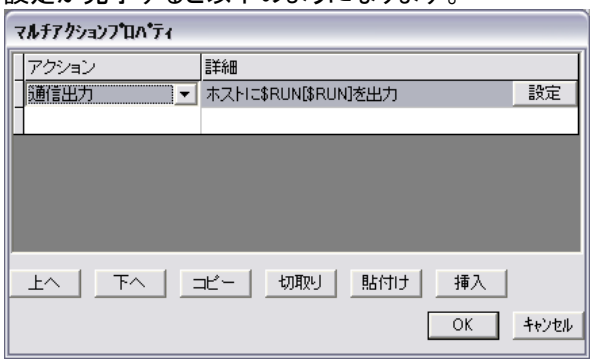


マルチアクションプロパティの設定画面を表示します。

②アクションで動作を選択します。

③詳細動作は次のように設定します。

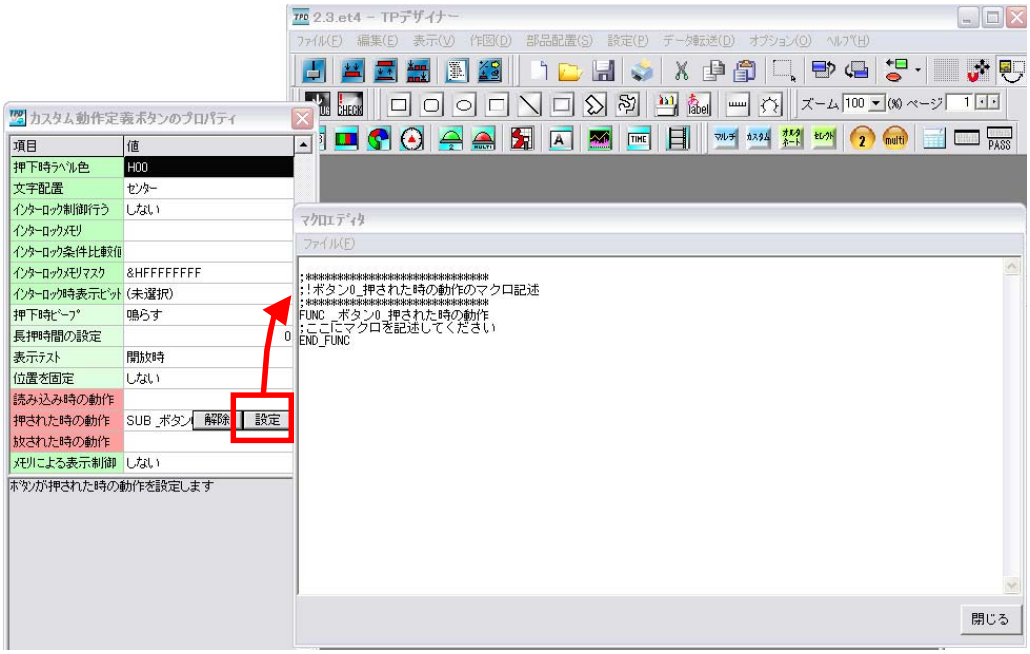
④設定が完了すると以下のようになります。



⑤同様に停止ボタンに STOP を出力するよう設定してください。

(2) 設定ボタン(マクロでの設定例)

設定ボタンはマクロ記述で設定します。
※動作上の必要性でなく説明のためです。
設定ボタンの押された時の動作の設定ボタンを押下すると、下のようにマクロエディタが開きます。



マクロエディタの中を下のように編集します。

```
;*****************************
;!ボタン 0_押された時の動作のマクロ記述
;*****************************
FUNC _ボタン 0_押された時の動作
;ここにマクロを記述してください
      OUTPUT $SET
END_FUNC
```

ここで OUTPUT とはホストに対して出力を行うマクロコマンドです。
マクロコマンドについては詳しくはマクロプログラミングマニュアルを参照してください。
上記のように編集し、マクロエディタを閉じるとマクロが登録されます。

※マクロエディタについてですが、外部エディタを使用することができます。
メニューのオプション-外部エディタの設定で、任意のエディタを設定できます。
外部エディタを使用することで、検索･置換などエディタのもつ高度な編集機能が使用できます。設定方法等詳しくは TP デザイナー取り扱い説明書を参照下さい。
※SET の前にある$についてですが、TP デザイナーのマクロでは直書きされた数字･文字を識別するために先頭に識別子が必要です。
#は整数、@は実数、$は文字列という意味になります。
マクロコマンドによっては識別子をつけてはいけない場合もありますので、詳しくはマクロプログラミングマニュアルを参照してください。

(3)閉じるボタン(GOP単体でのページ遷移の仕方)

閉じるボタンの押された時の動作を次のように記述します。

```
;*****************************
;!ボタン 2_押された時の動作のマクロ記述
;*****************************
FUNC _ボタン 2_押された時の動作
;ここにマクロを記述してください
      MOV PAGE2 #0
END_FUNC
```

ページの切替えはシステムメモリの PAGE(wF000)または PAGE2(wF002)に表示したいページ番号を書込むことで行います。
PAGE はベース画面の表示ページ番号です。
PAGE2 はレイヤー重ね合わせされるページのページ番号です。
PAGE2 に 0 を書込むとレイヤー重ね合わせページは消えます。
ポップアップはレイヤー重ね合わせページとして表示します。(後述しますが警報時にホストから PAGE2 に 2 を書込みます)。ポップアップを消すにはレイヤー重ね合わせページを消すことになるため PAGE2 にマクロを使用して 0 を書込みます。

以上で GOP 側の画面の動作についての設計は一通り完了です。

3. GOPへの画面書込み

GOP への画面データの書込みですが、以下の方法があります。

・シミュレータへの書込み

PC 上で動作するシミュレータにデータを書込ます。シミュレータは実機ではありませんが PC 上で動作確認が出来、通信なども行えるためデバッグに便利です。

・実機への書込み デバッグダウンロード

GOP 実機に書込む方法で、ROM にデータを保存しないため高速に書き込み出来ます。シミュレータでは確認できない実機上での動作速度などの確認が行えます。ROMに保存されないため GOP の電源を切ると書込んだデータは失われます。

・実機への書込み ROM 化

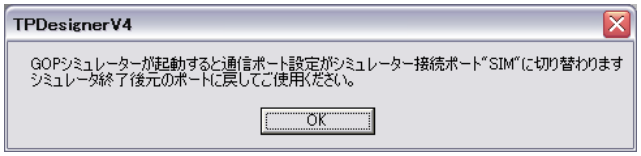
GOP の ROM にデータを書込みます。システムレベルでの動作確認や最終量産時に ROM 書込みを行います。

3.1 シミュレータへ書込む

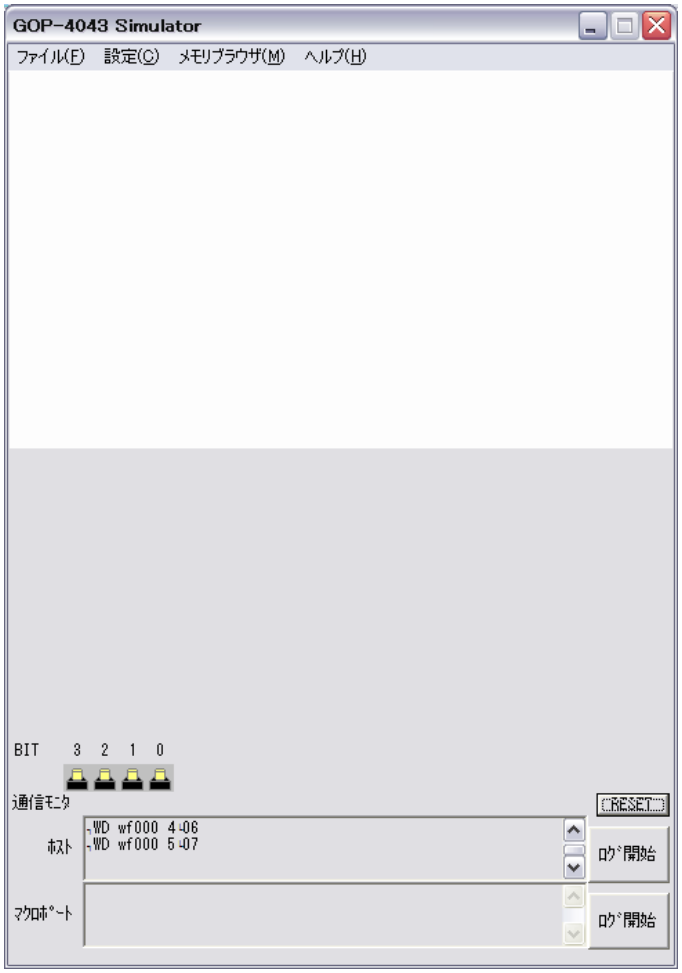
①シミュレーターを起動します

起動するにはしたのアイコンをクリックするか、メニューの「データ転送」⇒「シミュレーターの起動」を選択します。

②確認ダイアログが出て OK を押すとシミュレータが起動します。

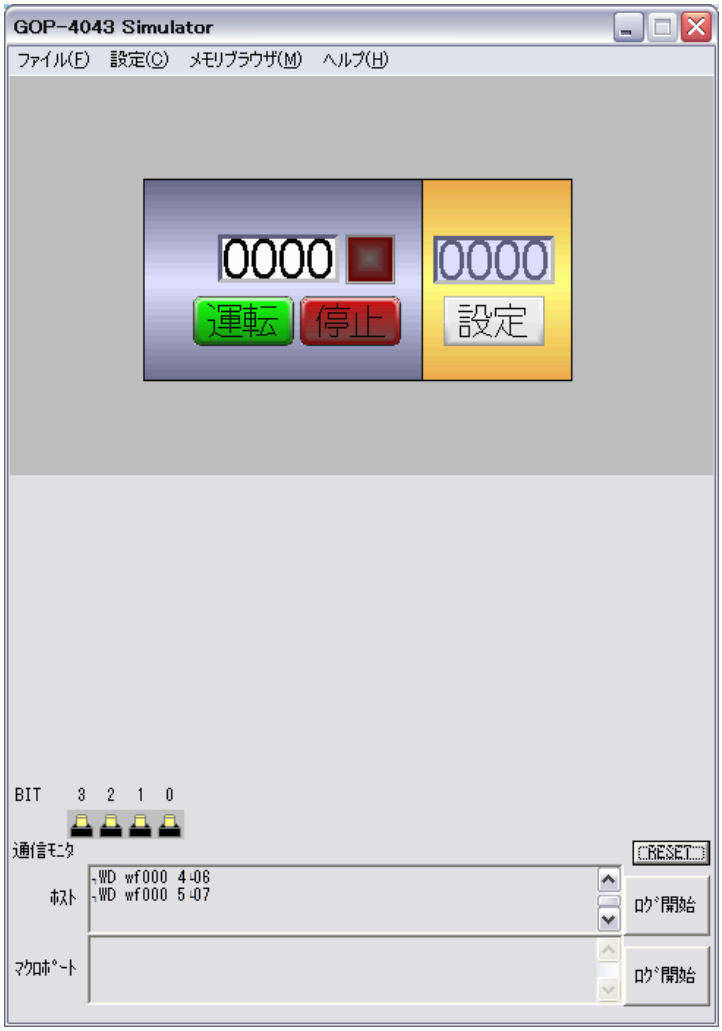


シミュレータ



③デバッグ転送ボタンを押します。

④シミュレータにデータが転送されます。



上記でシミュレータへの書込みは終了です。

3. 2 実機に書込む

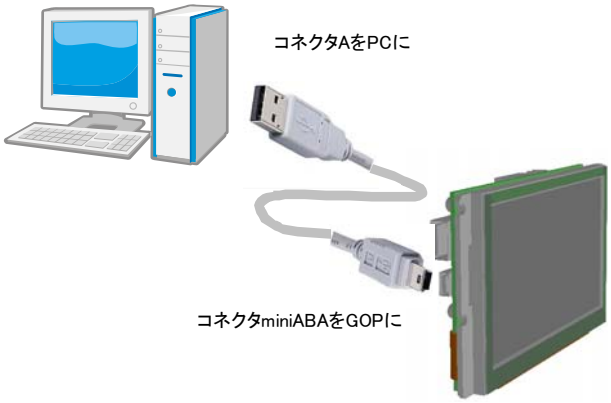
実機への書込みは GOP-4000 シリーズ、5000 シリーズで異なります。
以下それぞれのシリーズ毎に説明いたします。

3. 2. 1GOP-4000 シリーズの場合

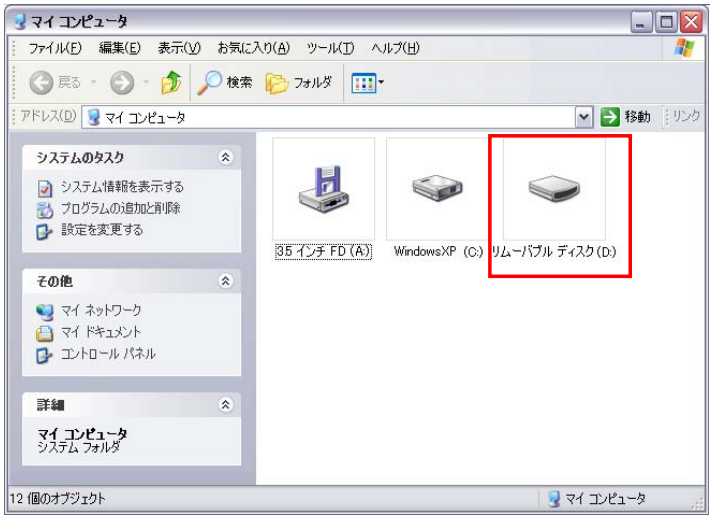
(1) デバッグダウンロード

※シミュレータを起動後にデバッグダウンロードを行う場合、シミュレーターを閉じ、メニューの「データ転送」⇒「シリアルポート」の設定を選択し、ポート番号を COM1(または GOP と接続に使用するポート)に変更しておいてください。

①GOP と PC を USB ケーブルで接続します。

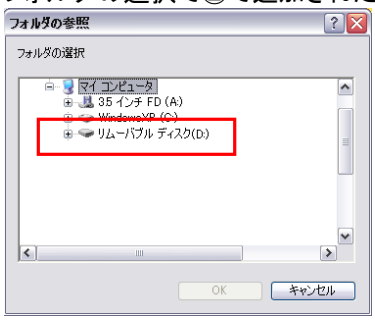


②マイコンピュータにリムーバブルディスクが追加されます。
追加されたドライブは覚えておいて下さい。

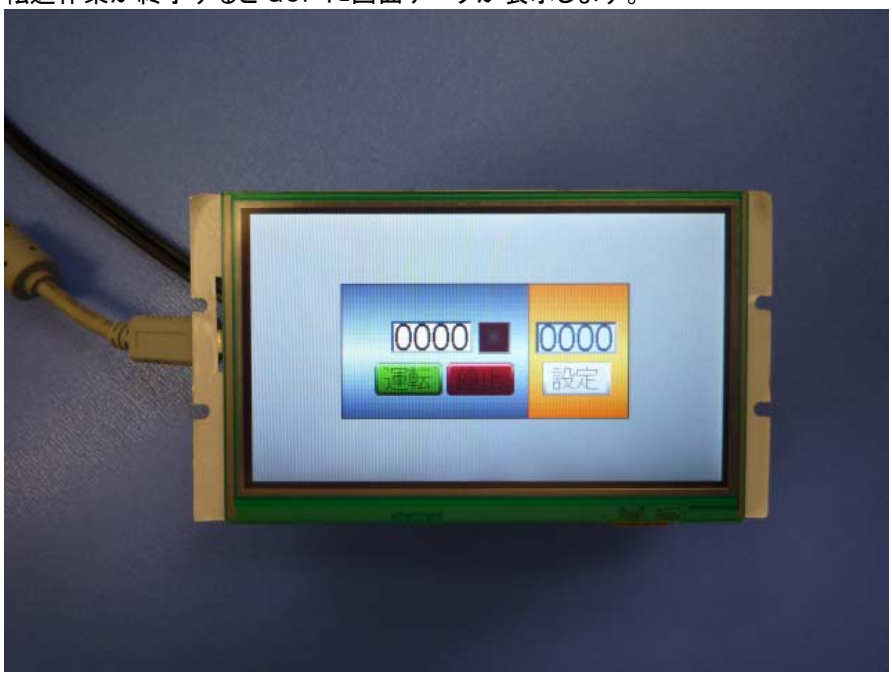


③デバッグ転送ボタンを押します。

④フォルダの選択で②で追加されたリムーバブルディスクを選択します。



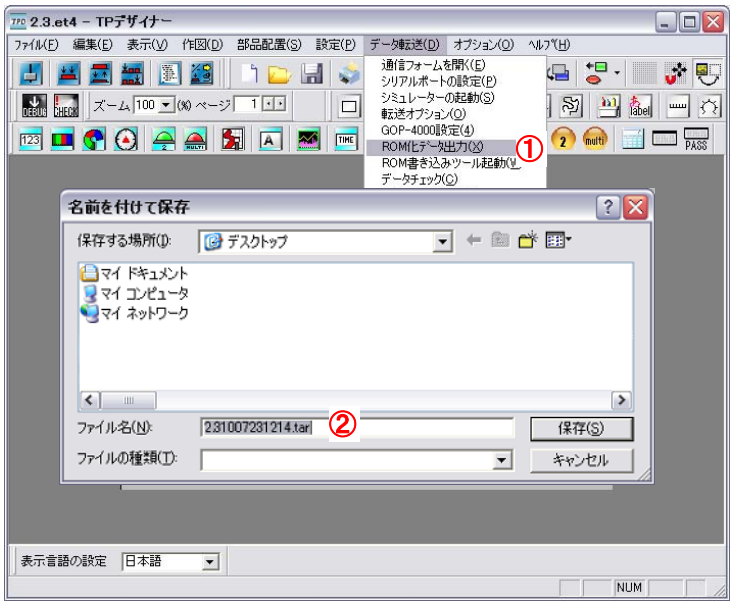
⑤転送作業が終了すると GOP に画面データが表示します。



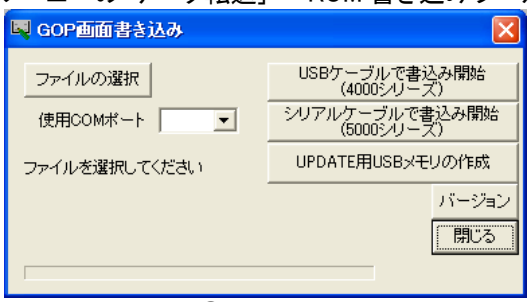
(2) ROM書込み

ROM 書込みは、書き込み用のデータを生成後、書き込みツールで書込む手順となります。
書込みデータは、GOP の動作に必要なファイル一式を Tar 形式にてアーカイブしたものになります。
書き込み手順は以下のとおりです。

①メニューの「データ転送」⇒「ROM 化データ出力」で、ROM 化データを作成します。
②保存ファイル名を指定します。



③メニューの「データ転送」⇒「ROM 書き込みツールを起動」で、書込みツールを起動します。

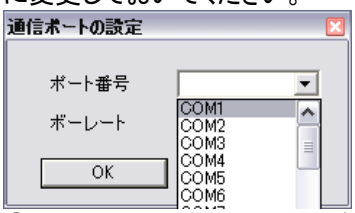


④ファイルの選択で②で出力したTarファイルを選択します。
⑤"USB ケーブルで書込みを開始(4000 シリーズ)"を選択するとフォルダの選択画面になります。
GOP のリムーバブルディスクを選択してください。
⑥書込みが開始します。GOP が下の表示になったら書込み終了です。
画面をタッチすると再起動します。

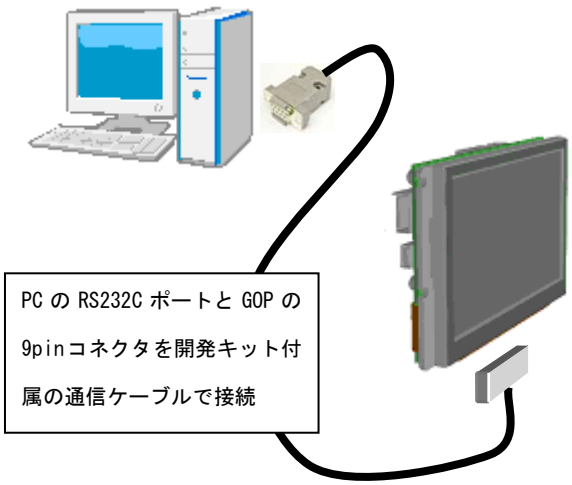
3.2.1GOP-5000 シリーズの場合

(1) シリアルケーブルを使ったデバッグダウンロード

※シミュレータを起動後にデバッグダウンロードを行う場合、シミュレーターを閉じ、メニューの「データ転送」⇒「シリアルポート」の設定を選択し、ポート番号を COM1(または GOP と接続に使用するポート)に変更しておいてください。

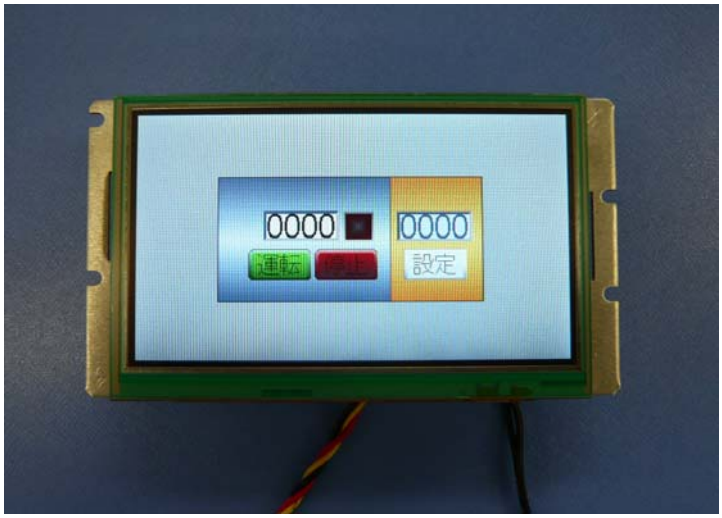


①GOP と PC をシリアルケーブルで接続します。



②デバッグ転送ボタンを押します。

③転送作業が終了すると GOP に画面データが表示します。

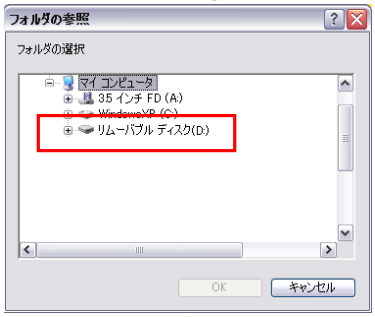


(2)USBメモリを使ったデバッグダウンロード

①PC に USB メモリを差し込みます

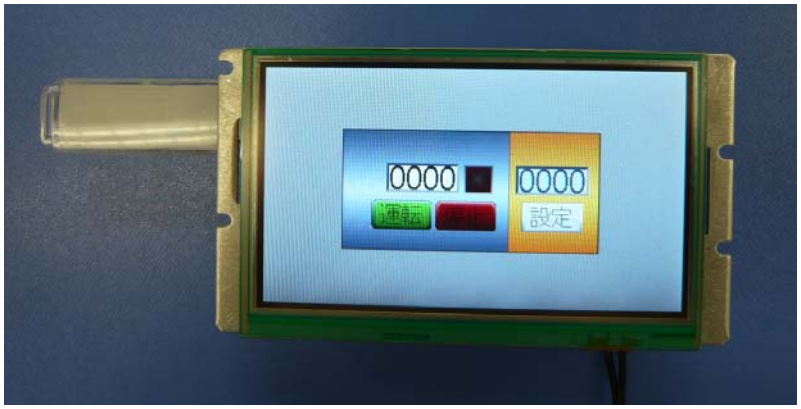
②TP デザイナーの"デバッグダウンロード用 USB メモリの書込み"を押します

③フォルダの選択で①で差し込んだ USB メモリを選択します。



④下のメッセージが出たら USB メモリを PC から抜き、GOP に差し込みます。

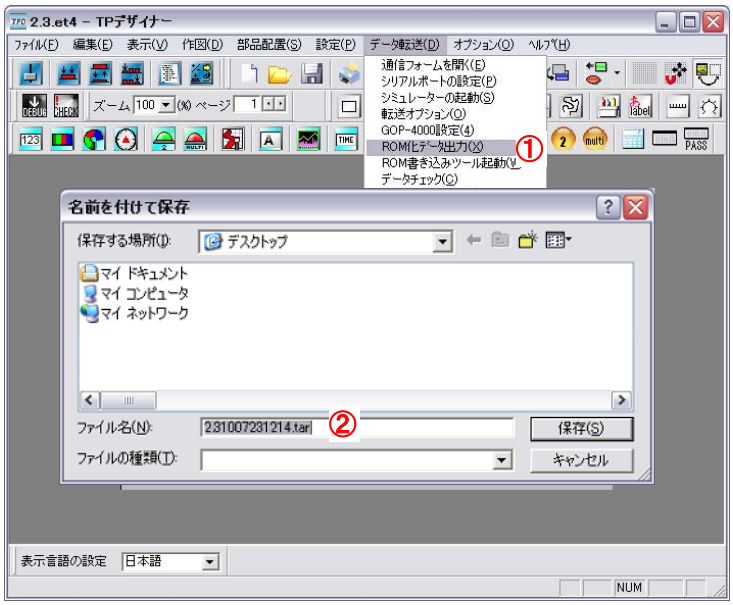
⑤GOP の電源を入れなおすと、画面データの読込が行われ書き込んだデータを表示します。



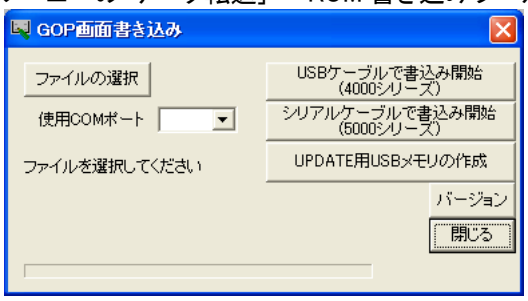
(3) ROM書込み

ROM 書込みは、書き込み用のデータを生成後、書き込みツールで書込む手順となります。
書込みデータは、GOP の動作に必要なファイル一式を Tar 形式にてアーカイブしたものになります。
書き込み手順は以下のとおりです。

①メニューの「データ転送」⇒「ROM 化データ出力」で、ROM 化データを作成します。
②保存ファイル名を指定します。



③メニューの「データ転送」⇒「ROM 書き込みツールを起動」で、書込みツールを起動します。



④ファイルの選択で②で出力したTarファイルを選択します。
⑤GOP と接続しているポートを,"使用している COM ポート"で指定します。
⑥"シリアルケーブルで書き込み開始(5000 シリーズ)"を選択すると書込みが開始します。GOP が下の表示になったら書込み終了です。

画面をタッチすると再起動します。

4. GOPとの通信と動作確認

TP デザイナーの通信フォームを使用して GOP の動作確認を行います。
本章以降の内容では、GOP と PC で通信いたします。開発キット付属の通信ケーブルで PC と GOP を接続してください。(USB ケーブルでは通信できません)
※確認をシミュレーターで行う場合は、接続は不要です。

4.1 通信フォームを開く

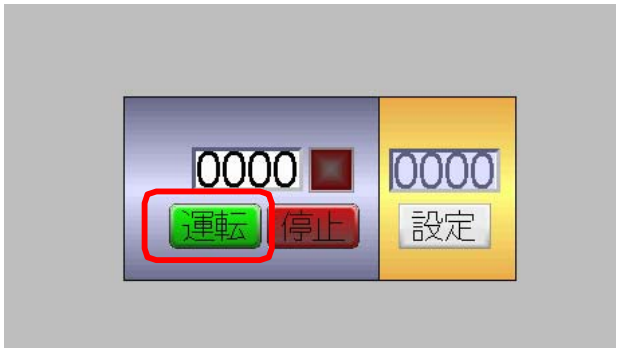
①使用するシリアルポートの設定
シミュレータで確認する場合は、シミュレータを起動することで通信先が自動でシミュレータに切り替わります。
実機と通信する場合、GOP と接続している COM ポートをメニューの「データ転送」⇒「シリアルポートの設定」で設定します。

②通信フォームを開く
メニューの「データ転送」⇒「通信フォームを開く」で通信フォームを開きます。

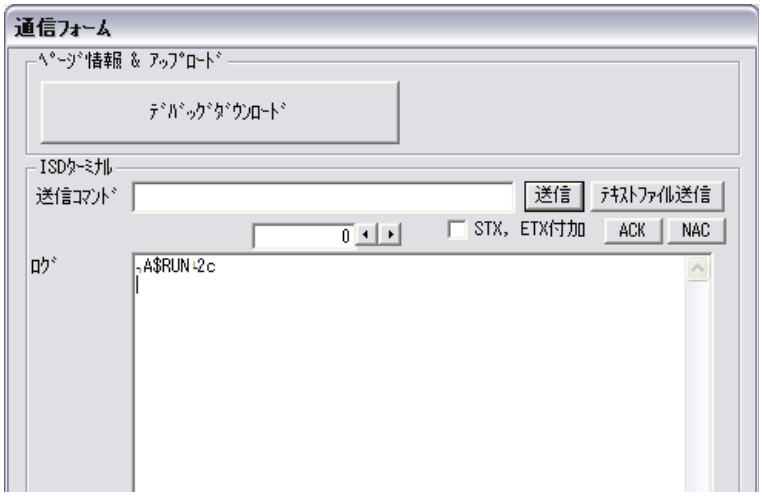
4. 2GOPの動作確認

(1) GOPからのデータの受信

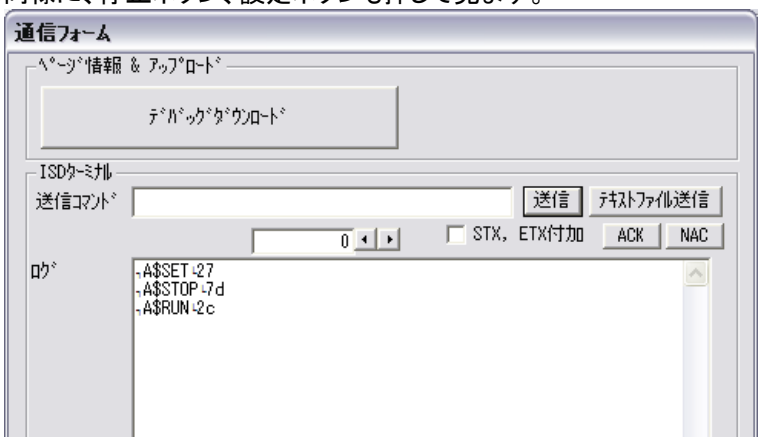
GOP のボタン動作に設定した、通信出力の動作を確認します。
GOP が出力するコマンドを通信フォームで受信します。
①GOP(またはシミュレータ)の運転ボタンを押してみます。



通信フォームのログ表示欄に送られてきたデータが表示します。

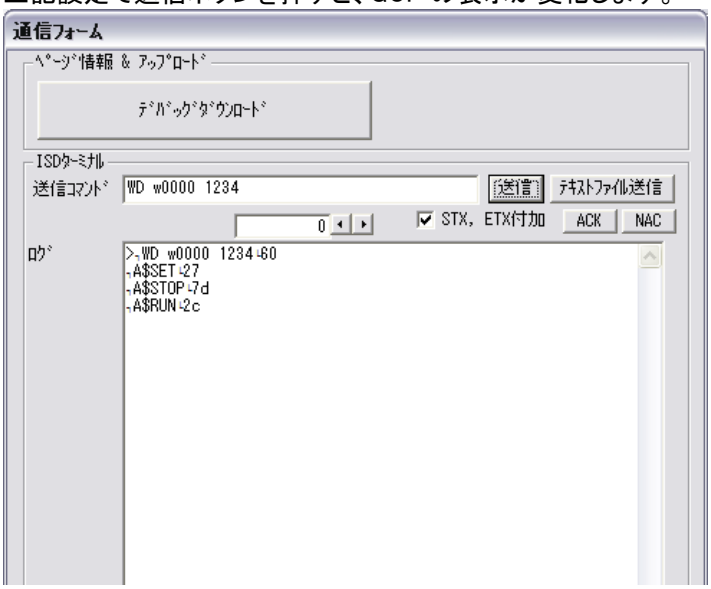


②同様に、停止ボタン、設定ボタンも押して見ます。

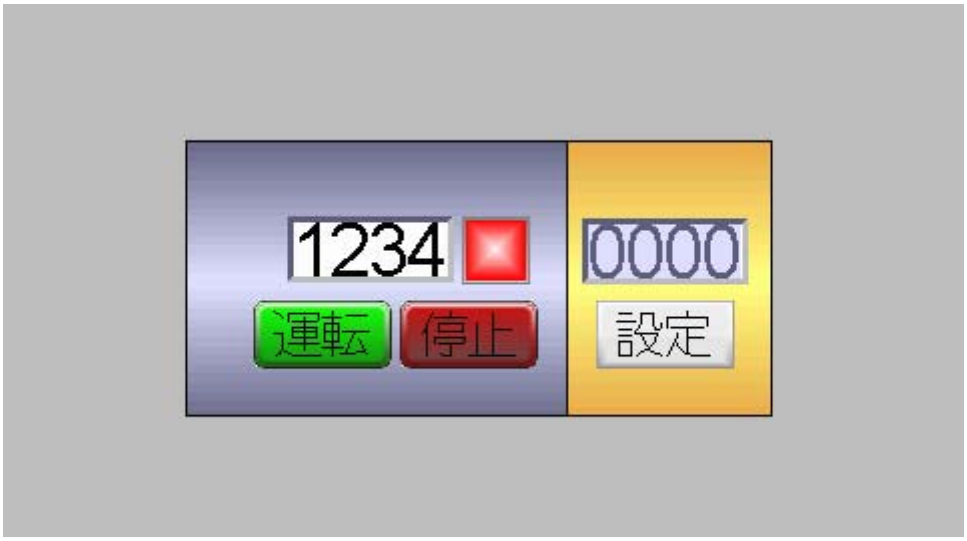


(2) GOPへのデータ書込み

GOP のメモリに通信フォームからデータを書込みます。
データを書込むことによる GOP の変化を確認します。
①送信コマンド欄に以下のコマンドを入力します。
″WD w0000 1234″
このコマンドはメモリの w0000 に 1234 を書込むという動作を行います。
STX,ETX 付加のチェックボックスにチェックを入れてください。これをチェックすると GOP のプロトコルに応じたパケット化、チェックサムの付加を通信フォームで自動で行います。
コマンドの詳細については、機能仕様書をご参照ください。
上記設定で送信ボタンを押すと、GOP の表示が変化します。



②GOP の表示が以下のようになります。
カウンタが w0000 にリンクしているため表示が変化します。
ランプが w0000 にリンクし、条件が w0000>w0002 のため点灯状態になります。



(3) GOPでテンキーで値を入力。その値を読込む

GOP 側でテンキーを使って値を設定します。
その後通信フォームから設定した値を読み込みます。
①GOP の右側のカウンタをタッチ。

②テンキーが表示します。

③テンキーで 4567 と入力し、Enter を押します。
右側のカウンターの表示が変化します。
※ランプの動作条件は w0000>w0002 となり、この入力動作により左記条件を満たさなくなりますが、ランプの表示変化の変化条件はリンクメモリの変化となります。w0002 はリンクメモリではない(比較値)のため、この動作でランプに表示変化は発生しません。

④通信フォームから w0002 の値を読み取ります。
送信コマンド欄に以下のコマンドを入力します。
″RD w0002″
このコマンドは w0002 の値を読み取る動作を行います。
このコマンドを送ると応答として w0002 の値が返ってきます。
″Rw0002=4567″

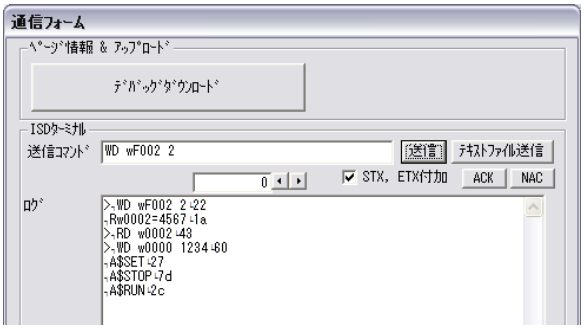
(4) ポップアップの表示と消去

ポップアップの表示します。ポップアップはレイヤー重ね合わせページとして表示するため PAGE2 のメモリに、ホストから 2 を書込みます。

①通信フォームから PAGE2 メモリに 2 を書込みます。

PAGE2 はメモリアドレス wF002 なので、以下のコマンドを送信します。

″WD wF002 2″



GOP の表示は次のようになります。

②閉じるボタンを押しポップアップが消えることを確認します。

③再度①の手順でポップアップを表示し、次は通信フォームから以下のコマンドでもポップアップが消えることを確認します。

(5) その他システムメモリの操作

GOP の動作はページ移動のように、ホストからシステムメモリ領域を読み書きすることで制御できます。

①ブザーを鳴らす

システムメモリの BUZER(wF090)に鳴らす長さを書込むことでブザーが鳴ります。

″WD wF090 10″

②タッチ座標

システムメモリの TOUCH_X(WF080),TOUCH_Y(WF082)でタッチされている押下位置がわかります。

GOP の任意の場所をタッチしたまま、通信フォームで

″RD WF080″

とすると押下点の X 座標がわかります。

その他のシステムメモリの動作については機能仕様書を参照ください。

5. ホストプログラミング

GOP と連携して動作するホスト側のプログラムを作成します。
プログラミングは C 言語を使用します。
また、GOP との通信に関する処理に付いては、TP デザイナーに添付の GOPLIB を使用します。
GOPLIB を使用することでコマンドの解析などの処理を作成する必要なく、簡単にホストシステムを作成できます。
※本章につきましては想定読者を C 言語を理解されていることを前提としています。

5.1 サンプルの実行環境について

本章で使用する、コードのサンプルを添付しています。
サンプルは MicroSoft VisualStudio 6 または 2008(以下 VS6 または VS2008)での動作を確認しています。
Windows 上で動作する仮想ホストと GOP の間で通信します。

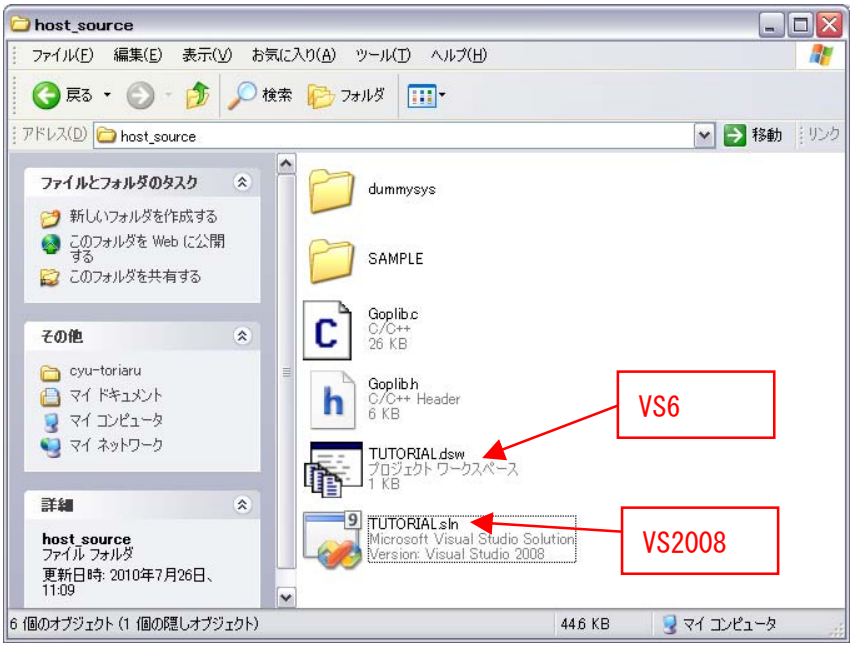
なおサンプルは GOP シミュレータを使用して動作するようにしています。
実機と接続する場合、dummyhard.h の下記の箇所を修正してください。

```
#define COM1 0
#define COM2 1
#define COM3 2
#define COM4 3
#define COM5 4
#define SIM 100
#define USEPORT SIM        <- ここを使用する COM ポートに変更します。
//例 COM1 にする場合
//#define USEPORT COM1
#define BAUDRATE 38400
```

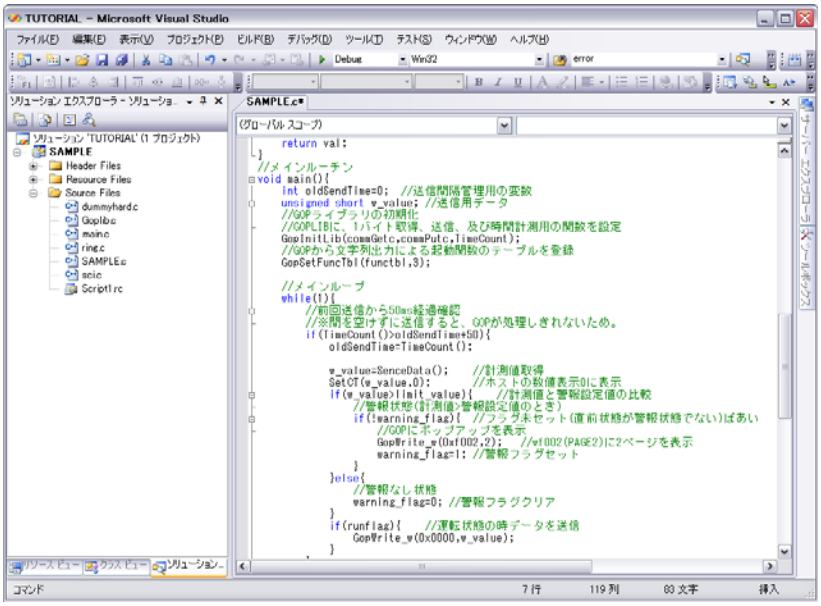
GOPLIB 及び仮想ホストに関する詳細はアプリケーションノートを参照してください。

5.2 プログラム説明

サンプルプロジェクトは host_source のホルダにある TUTORIAL.dsw(VS6 の場合)または TUTORIAL.sln(VS2008)のファイルをダブルクリックすることで開きます。



サンプルのプロジェクトを開くと以下のようになります。(以下は VS2008 で開いています)



色々ファイルが含まれていますが、本チュートリアルで使用するのは SAMPLE.C のみです。
他のファイルは仮想ホストや GOPLIB のもので、内部について本チュートリアルでは説明いたしません。
GOPLIB を使用して GOP と通信する方法について以下に説明いたします。

(1)GOPLIBの初期化

GOPLIB はシリアルポートを使った 1 バイト送受信関数と、ms 単位の時間計測が出来る関数を外部から指定する必要があります。
上記の関数は、ホストボード側で用意しておく必要があります。
仮想ホストではこれら関数は
commGetc,commPutc,TimeCount
という名前で用意しています。

これら関数を GopInitLib 関数で GOPLIB に登録します。

```
GopInitLib(commGetc, commPutc, TimeCount);
```

(2)GOPにデータを送信する

GOP にデータを書込む場合以下の関数を使用します。
GopWrite_?(アドレス,書込み値)
※?は書込むメモリの型が入ります。w 型メモリに書く場合 GopWrite_w となります。

本関数は以下のように使用します。

```
GopWrite_w(0x0000, 123);
```

これで GOP の w0000 番地に 123 が書込まれます。
※GOPLIB 内で GOP の通信コマンドのフォーマットにあわせてコマンドを生成し出力します。

※なお GOP への書き込みについてですが、常時送信するデータの場合、GOP はメモリに書込まれることによりオブジェクトの再描画等の動作が発生するため、ある程度の間隔を空けてデータを送る必要があります。(GOP の動作により受信データが処理しきれない場合、受信バッファオーバフローが発生する)
この間隔は画面データにより異なりますが目安として最低 10ms 以上あけるようにしてください。

(3)GOPのデータを読込む

GOP のメモリデータを読込むためには以下の関数を使用します。
GopRead_?(アドレス,読込み値を格納する変数のアドレス,タイムアウト値)
※?は読込むメモリの型が入ります。w 型メモリを読む場合 GopRead_w となります。
なお読込み値を格納する変数は、読込みメモリに合わせた型を試用する必要があります。

※	b unsigned char	B signed char
	w unsigned short	W signed short
	l unsigned long	L signed long
	F float	
	T char	

本関数は以下のように使用します。

```
unsigned short read_val;
GopRead_w(0x0002, &read_val, 500);
```

これで GOP の w0002 番地の内容が read_val に書込まれます。
GOP との通信に異常があり 500ms を過ぎても応答がない場合、エラーとして処理されます。
※GOPLIB 内で GOP の通信コマンドのフォーマットにあわせてコマンドを生成し出力します。
※エラー時の対応はアプリケーションノート参照してください。

(4) GOPのイベントを受け取りそれに対応する動作を行う

GOP が出力する文字列に応じ、任意の関数を起動させることが出来ます。
手順としては以下のようになります。
①文字列と関数の対応テーブルを作る。
②対応テーブルを GOPLIB に登録する。
③メインループ中で GOP からの送信コマンドをチェックする
上記について詳細を以下に説明します。

①文字列と関数の対応テーブルを作る。

```
GOP_FUNCTBL functbl[]={
        {"GOP から送られる文字列",起動する関数のポインタ},
          :
          :
};
```

上記で GOP_FUNCTBL は goplib.h にて定義される構造体です。
また起動する関数は以下の型となります。

```
void func(void);
```

今回の動作の場合 GOP からは"RUN","STOP","SET"の 3 種類の文字列が送られますのでそれ応じたテーブルを作成します。

```
GOP_FUNCTBL functbl[]={
    {"RUN",fnRUN},
    {"STOP",fnSTOP},
    {"SET",fnSET},
};
```

起動関数はコマンドの動作にあわせ作成します。

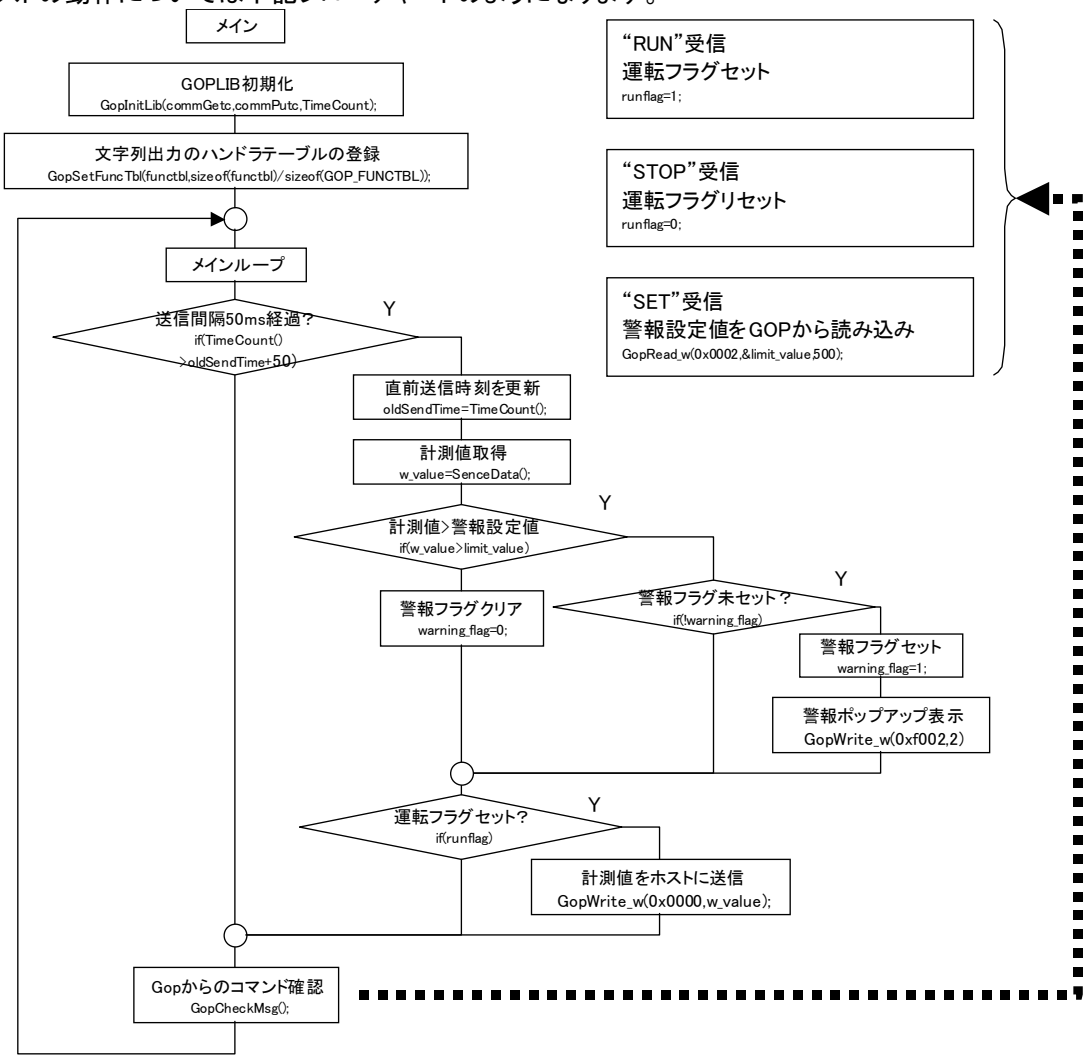
②対応テーブルを GOPLIB に登録する。
GOPLIB の初期化後、①のテーブルを GOPLIB に登録します。
GOPLIB への登録は GopSetFuncTbl 関数を使用します。
以下のように使用します。

```
GopSetFuncTbl(functbl,sizeof(functbl)/sizeof(GOP_FUNCTBL));
```

③メインループ中で GOP からの送信コマンドをチェックする
ホストのメインループ中で GopCheckMsg 関数を実行することで、
・ホストからのコマンドの確認
・テーブルとの照合
・一致した関数をコールバックで起動
という一連の作業を行うことが出来ます。

5.3 ホストプログラム

ホストの動作については下記フローチャートのようになります。



ホスト側のプログラムのソースは以下のようになります。
動作についてはコメントを参照ください。

```
//仮想ホストの定義ファイル
#include "../dummysys/dummyhard.h"
//GopLib の定義ファイル
#include "../goplib.h"
static int runflag=0;      //運転中フラグ
static int warning_flag=0;//警報状態フラグ
static unsigned short limit_value=10000;        //警報設定値 初回セットされるまで警報が出ないように
                                                //10000 を初期値とします。
//GOP からコマンドを受け取ったときの動作を設定します。
//文字列"RUN"の取得
void fnRUN(){
      //runflag をセット
      runflag=1;
      //ホスト側のランプ(ラジオボタン)をセット
      SetLamp(GetLamp()|1);
}
//文字列"STOP"の取得
void fnSTOP(){
      //runflag をリセット
      runflag=0;
      //ホスト側のランプ(ラジオボタン)をクリア
      SetLamp(GetLamp()&~1);
}
//文字列"SET"の取得
void fnSET(){
      //GOP の w0002 をホスト内の変数に読込み
      GopRead_w(0x0002,&limit_value,500);
      //ホスト側の数値表示 1 に読込み値を表示
      SetCT(limit_value,1);
}
//GOP の出力文字列と、それによる起動関数のテーブル
//関数は以下の様式
// void funcname(void)
GOP_FUNCTBL functbl[]={
      {"RUN",fnRUN},
      {"STOP",fnSTOP},
      {"SET",fnSET},
};
//ホストがデータを計測するルーチン
//サンプルなので、／＼／＼の波形を送る(0-9999)
unsigned short SenceData(){
      static int val=0;    //計測値
      static int delta=13; //増分値
      val+=delta;
      if(val>9999){
                  val=9999;
                  delta=-13;
      }
      if(val<0){
                  val=0;
                  delta=13;
      }
      return val;
}
//メインルーチン
void main(){
      int oldSendTime=0;              //送信間隔管理用の変数
      unsigned short w_value;         //送信用データ
      //GOP ライブラリの初期化
      //GOPLIB に、1 バイト取得、送信、及び時間計測用の関数を設定
      GopInitLib(commGetc,commPutc,TimeCount);
      //GOP から文字列出力による起動関数のテーブルを登録
      GopSetFuncTbl(functbl,sizeof(functbl)/sizeof(GOP_FUNCTBL));
      //メインループ
      while(1){
                  //前回送信から 50ms 経過確認
                  if(TimeCount()>oldSendTime+50){
                              oldSendTime=TimeCount();
                              w_value=SenceData();            //計測値取得
                              SetCT(w_value,0);               //ホストの数値表示 0 に表示
                              if(w_value>limit_value){        //計測値と警報設定値の比較
                                          //警報状態(計測値>警報設定値のとき)
                                          if(!warning_flag){  //フラグ未セット
                                                              //(直前状態が警報状態でない)ばあい
                                                      //GOP にポップアップを表示
                                                      GopWrite_w(0xf002,2);//wf002(PAGE2)に 2 ページを表示
                                                      warning_flag=1;       //警報フラグセット
                                          }
                              }else{                          //警報なし状態
                                          warning_flag=0;     //警報フラグクリア
                              }
```

```
                        if(runflag){          //運転状態の時データを送信
                                GopWrite_w(0x0000,w_value);
                        }
                }
                //GOP からのコマンドを確認
                //この処理は GOPLIB 内で行われます。
                GopCheckMsg();
        }
}
```

5.4 実行イメージ

サンプルを以下の手順で実行します。
①GOP シミュレータを起動する。
②画面データが入っていない場合、画面データを書込む
③サンプルをビルド・実行する
④正しくビルドされると仮想ホストが起動し、GOP シミュレーターと接続され通信が開始します。
⑤GOP の運転ボタンを押すと GOP のカウンタが変化します。
⑥右側のカウンタをテンキーで値を設定し、設置ボタンを押すと警報動作が動作します。
設定値を超えるとポップアップが表示します。