GR-CITRUS 搭載
Rubyファーム
TFT Touch Shield Capative用
TFTc Class ver1.01
説明資料 ver1.01

Wakayama.rb

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

画面を指定色で塗りつぶす: TFTc.fillScreen(color)

画面を color で指定した色で塗りつぶします。
  color: 16bit(RGB565)値

  戻り値
  なし

※16bit カラー値はMSBより 5bits red, 6bits green, 5bits blueの並び。
※下記の定義を使用可能。

黒色	C_BLACK	(値:0x0000)
茶色	C_BROWN	(値:0xcb43)
赤色	C_RED	(値:0xf800)
橙色	C_ORANGE	(値:0xfd20)
黄色	C_YELLOW	(値:0xffe0)
緑色	C_LIME	(値:0x07e0)
青色	C_BLUE	(値:0x001f)
紫色	C_VIOLET	(値:0x901a)
灰色	C_GRAY	(値:0xa554)
白色	C_WHITE	(値:0xffff)
水色	C_CYAN	(値:0x07ff)
マゼンタ	C_MAGENTA	(値:0xf81f)

※メソッドにて数値での色指定も可能（参照：TFTc.color ）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

16bitカラー値を得る: TFTc.color(color_no)

整数のカラーナンバーから、16bitカラー値を得ます。
  color_no: 0～11 の値（下表参照）

  戻り値
  16bitカラー値

※カラーナンバー定義
  カラーナンバー 0: 黒色　　(値:0x0000)
  カラーナンバー 1: 茶色　　(値:0xcb43)
  カラーナンバー 2: 赤色　　(値:0xf800)
  カラーナンバー 3: 橙色　　(値:0xfd20)
  カラーナンバー 4: 黄色　　(値:0xffe0)
  カラーナンバー 5: 緑色　　(値:0x07e0)
  カラーナンバー 6: 青色　　(値:0x001f)
  カラーナンバー 7: 紫色　　(値:0x901a)
  カラーナンバー 8: 灰色　　(値:0xa554)
  カラーナンバー 9: 白色　　(値:0xffff)
  カラーナンバー 10: 水色　　(値:0x07ff)
  カラーナンバー 11: マゼンタ　(値:0xf81f)

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

画面を回転する: TFTc.rotation(rotation)

画面を指定したrotation値に回転します。
rotation: 0:垂直 1:水平 2: 垂直反転 3:水平反転

戻り値
なし

※ 設定した値を呼び出すメソッドは実装していません。
※ 初期値は0です。

画面の横幅を取得する: TFTc.width()

画面の横幅を取得します。

戻り値は横幅のピクセル数

画面の縦幅を取得する: TFTc.height()

画面の縦幅を取得します。

戻り値は縦幅のピクセル数

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

ビットマップの表示: TFTc.drawBmp(filename,x,y)

SDカードにあるビットマップファイルを、指定の座標に表示します。
  filename:　ファイル名(8.3形式)
  x:　表示させるビットマップの左上のx座標
  y:　表示させるビットマップの左上のy座標

  戻り値
  0: 描画成功
  1: 失敗（ビットマップサイズが画面よりも大きい）
  2: 失敗（ファイルが存在しない）
  3: 失敗（サポート外のファイル形式）

ドットを描画する: TFTc.drawPixel(x,y,[,color])

指定座標に（指定した色で）ドットを描画します。
  x: ドットを描画するx座標
  y: ドットを描画するy座標
  color: 描画するドットの16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 描画終了点は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

線を描画する: TFTc.drawLine(x0,y0,x1,y1[,color])

２点の指定座標を結ぶ線を（指定した色で）描画します。
  x0: 直線の描画開始点のx座標
  y0: 直線の描画開始点のy座標
  x1: 直線の描画終了点のx座標
  y1: 直線の描画終了点のy座標
  color: 描画する直線の16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 描画終了点は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

連続して線を描画する: TFTc.drawLineto(x,y[,color])

直前に描画した線の最終座標と指定座標を結ぶ直線を（指定した色で）描画します。
  x: 直線の描画終了点のx座標
  y: 直線の描画終了点のy座標
  color: 描画する直線の16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 描画終了点は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

円を描画する: TFTc.drawCircle(x,y,r[,color])
塗りつぶしの円を描画する: TFTc.fillCircle(x,y,r[,color])

円を（指定した色で）描画します。
  x: 円の中心点のx座標
  y: 円の中心点のy座標
  r: 円の半径(dot)
  color: 描画する円の16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 描画した円の中心座標は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

四角形を描画する: TFTc.drawRect(x,y,w,h[,color])
塗りつぶしの四角形を描画する: TFTc.fillRect(x,y,w,h[,color])

四角形を（指定した色で）描画します。
  x: 四角形の左上のx座標
  y: 四角形の左上のy座標
  w: 四角形の幅(dot)
  h: 四角形の高さ(dot)
  color: 描画する四角形の16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 四角形の左上の座標は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

角の丸い四角形を描画する: TFTc.drawRoundRect(x,y,w,h,r[,color])
塗りつぶしの角の丸い四角形を描画する: TFTc.fillRoundRect(x,y,w,h,r[,color])

角の丸い四角形を（指定した色で）描画します。
  x: 四角形の左上のx座標
  y: 四角形の左上のy座標
  w: 四角形の幅(dot)
  h: 四角形の高さ(dot)
  r: 四角形の角の半径(dot)
  color: 描画する四角形の16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 四角形の左上の座標は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

三角形を描画する: TFTc.drawTriangle(x0,y0,x1,y1,x2,y2[,color])
塗りつぶしの三角形を描画する: TFTc.fillTriangle(x0,y0,x1,y1,x2,y2[,color])

三角形を（指定した色で）描画します。
  x0: 三角形の第1頂点のx座標
  y0: 三角形の第1頂点のy座標
  x1: 三角形の第2頂点のx座標
  y1: 三角形の第2頂点のy座標
  x2: 三角形の第3頂点のx座標
  y2: 三角形の第3頂点のy座標
  color: 描画する三角形の16bitカラー値

  戻り値
  なし

※ colorを省略した場合は、最後にグラフィックを描画した色で描画します。
  （ビットマップやテキストの描画色は除きます）　初期値は0x0000(黒色)です。
※ 四角形の第1頂点の座標は次のグラフィックが描画されるまで保持します。
  （ビットマップやテキストの描画色は除きます）

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

テキストを１文字描画する: TFTc.drawChar(x,y,value,color,bgcolor,size)

指定の位置に5x7サイズのアスキー文字を１文字、指定の文字色と背景色で描画します。
背景色の描画範囲は、文字の左上を基準にした6x8サイズとなります。

- x: 文字の左上のx座標
- y: 文字の左上のy座標
- value: キャラクタ値（アスキーコード）
- color: 描画する文字の16bitカラー値
- bgcolor: 文字の背景色の16bitカラー値
- size: 描画する文字の拡大率(1以上の整数)

戻り値
なし

文字列の描画位置を設定する: TFTc.textCursor(x,y)

文字列を描画するにあたり、カーソル位置の座標を設定します。

- x: あとで描画する文字列の左上のx座標
- y: あとで描画する文字列の左上のy座標

戻り値
なし

※ 文字列の描画は、TFTc.print, TFTc.println メソッドを用います。
※ 初期値は x=0、y=0 です。

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

文字列の描画色を設定する: TFTc.textColor(color[,bgcolor])

文字列を描画するにあたり、文字色（および背景色）を設定します。
  color: あとで描画する文字列の16bitカラー値
  bgcolor: あとで描画する文字列の背景色の16bitカラー値

  戻り値
  なし

※ 文字列の描画は、TFTc.print, TFTc.println メソッドを用います。
※ 初期値は color、bgcolorともに白(=0xFFFF)です。

文字列のサイズを設定する: TFTc.textSize(size)

文字列を描画するにあたり、文字のサイズ（拡大率）を設定します。
  size: 描画する文字列の拡大率(1以上の整数)

  戻り値
  なし

※ 文字列の描画は、TFTc.print, TFTc.println メソッドを用います。
※ 初期値は 1 です。

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

文字列を改行なしで描画する: TFTc.print([str])

予め指定したカーソル位置に、予め設定した文字色、文字サイズで、指定の文字列を改行なしで描画します。
[str]: アスキー文字列

戻り値
なし

文字列を描画し改行する: TFTc.println([str])

予め指定したカーソル位置に、予め設定した文字色、文字サイズで、指定の文字列を描画し、改行します。
[str]: アスキー文字列

戻り値
なし

文字列の折り返しを設定する: TFTc.textWrap(mode)

文字列を描画する際、画面の右端に差し掛かった場合に折り返すか折り返さないかを設定します。
mode: 0: 折り返さない 1:折り返す

戻り値
なし

※ 初期値は 1 です。

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

文字および文字列描画で使用するフォントを設定する: TFTc.textFont(fontno)

【オプション機能】使用するフォントを設定します。
fontno: フォントno 0～48

戻り値
なし

※ファームウェアサイズを圧縮するため、デフォルトでは標準フォント（fontno=0)以外は使えません。
必要に応じて sTFTc.cpp、およびsTFTc.hのコメントアウトを削除し、有効にしてください。

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

タッチを検出する: TFTc.panelTouched()

タッチパネルがタッチされているかどうかを検出します。タッチされている場合は内部変数にタッチされた座標が記録されます。

戻り値
0: タッチされていない
1: タッチされている

※画面回転の設定に応じて座標を自動変換します。

タッチされた座標のx値を得る: TFTc.touchedX()

TFTc.panelTouched()関数で1が返された場合、本メソッドを用いてタッチされた座標のx値を得ることが出来ます。

戻り値は、タッチされた座標のxの値

タッチされた座標のy値を得る: TFTc.touchedY()

TFTc.panelTouched()関数で1が返された場合、本メソッドを用いてタッチされた座標のy値を得ることが出来ます。

戻り値は、タッチされた座標のyの値

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

ボタンの最大数を設定する: TFTc.setButtonMax(buttonmax)

使用するボタンの最大数を設定します。（ボタン構造体のインスタンスを生成するためメモリを消費します）
プログラムで複数画面を使用する場合、画面毎にボタンの数が異なるため、一番多いボタン数のメモリをこのメソッドで確保します。
　buttonmax: プログラムで使用するボタン数の最大値(1～32)

　戻り値は、確保したボタン構造体の数

※ 最大のボタン数は 32個です。これ以上のボタン数を指定した場合は32個に制限されます。

設定したボタンの最大値をクリアする: TFTc.deleteButtons()

確保してあるボタン構造体のインスタンスを開放します。TFTc.setButtonMax で設定した最大値を変更したい場合はこの手続を実施したあとで設定し直します。

　戻り値は、開放したボタン構造体の数

※ 最大のボタン数は 32個です。これ以上のボタン数を指定した場合は32個に制限されます。

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用するボタンの数を設定する: TFTc.setDisplayButtonMax(buttonmax)

画面に表示し、制御するボタンの数を設定します。
  buttonmax: 画面に表示するボタンの数(1～32)

  戻り値は、設定したボタン数

※設定するボタンの数は、TFTc.setButtonMaxメソッドで設定した数を超えてはなりません。

ボタンを個別に初期化する: TFTc.initButton(no,x,y,w,h,edgec,backc,textc,[str],size)

表示および制御するボタンを個別に初期化します。
  no: ボタンno (1～32)
  x: ボタンの中央のx座標
  y: ボタンの中央のy座標
  w: ボタンの幅
  h: ボタンの高さ
  edgec: ボタンの縁の16bitカラー値
  backc: ボタンの背景の16bitカラー値
  textc: ボタンの中に表示する文字の16bitカラー値
  [str]: ボタンの中に表示するアスキー文字列
  size: ボタンの中に描画する文字の拡大率(1以上の整数)

  戻り値
  0: 正しく設定できなかった場合 1:正しく設定できた場合

※設定するボタンnoは、TFTc.setDisplayButtonMaxメソッドで設定した数を超えてはなりません。

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

全てのボタンを描画する: TFTc.drawButtons()

TFTc.setDisplayButtonMaxで設定したボタン数のボタンを、全て通常表示で描画します。
TFTc.initButtonで個別に全てのボタンを設定した後、このメソッドを用いて最初の描画を行います。

戻り値
なし

ボタンを個別に描画する: TFTc.drawButton(no,invert)

ボタンを個別に描画します。
no: ボタンno (1～32)
invert: 反転表示フラグ 0:反転しない(通常表示) 1:反転する(反転表示)

戻り値
なし

ボタンがタッチされたどうか調べる: TFTc.searchTouchedButton(x,y)

タッチが検出された場合に、その座標がボタンかどうかを調べます。
検索の結果は、各ボタンの状態変数に記録されます。
x: タッチ検出位置のx座標
y: タッチ検出位置のy座標

戻り値
なし

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

個別にボタンの変化を調べる: TFTc.searchButtonState(no)

指定したボタンがリリース状態から押下状態に変化したか、押下状態からリリース状態に変化したか、それとも変化がないか、判定を行う。
また、押下状態からリリース状態に変化した場合は、ボタンの描画を通常の描画に変更する。
リリース状態から押下状態に変化した場合は、ボタンの描画を反転させる。
どちらでもない場合（変化がない場合）は、描画を変更しない。
　no: ボタンno

　戻り値
　1:押下→リリースと変化 -1:リリース→押下と変化 0:変化なし

※同時押しや長押し検出など、細かな制御を行いたい場合に使用します。

押下されたボタンを検出する: TFTc.searchPressedButton()

TFTc.setDisplayButtonMaxメソッドで設定したボタン数のボタンについて、リリース状態から押下状態へ変化したボタンnoを検出する。
押下されたボタンは自動的に反転表示となる。また、押下されていないボタンは自動的に通常表示となる。

　戻り値
　0:どのボタンも押下されていない 1～32:数値のボタンがリリース状態から押下状態へ変化

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

リリースされたボタンを検出する: TFTc.searchReleasedButton

TFTc.setDisplayButtonMaxメソッドで設定したボタン数のボタンについて、押下状態からリリース状態へ変化したボタンnoを検出する。
押下されたボタンは自動的に反転表示となる。また、押下されていないボタンは自動的に通常表示となる。

戻り値
0:どのボタンもリリースされていない 1～32:数値のボタンが押下状態からリリース状態へ変化

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

※ 現状のファームウェアに、SDカードアクセス時に10pinよりノイズが出力される不具合があります。
そのため、このpinをCS信号として使用しているTFTタッチパネル液晶にゴミが描画されてしまう場合があります。
バグが取り切れていない現状では、下記のコードをrubyコードの最初に挿入することで回避できます。

```
# 初期化開始 (SDカードとTFTcapacitiveタッチパネルの共存)
# 10pinからのノイズ出力の不具合対策
# 予めSDカードにsample.txtを作り、カードを挿入しておく
pinMode(10,1)       # pin10 output
digitalWrite(10,1)  # pin10をHighに
USB.println("Checking SD")
if (0 == System.useSD()) then
    USB.println("SD card is not inserted.")
else
    USB.println("SD card is ready.")
    if (0==SD.open(0,"sample.txt",0)) then
        while(true) do
            c = SD.read(0)
            if(c < 0) then
                break
            end
            USB.print(c.chr)
        end
        SD.close(0)
    end
end
tone(10,1000,100)
digitalWrite(10,1)  # pin10をHighに (tone off)

delay(100)
 if(1==System.useTFTc()) then
    TFTc.drawPixel(1,1,TFTc.color(1))
end
USB.println("Checking SD") #2回目
if (0 == System.useSD()) then
    USB.println("SD card is not inserted.")
else
    USB.println("SD card is ready.")
    if (0==SD.open(0,"sample.txt",0)) then
        while(true) do
            c = SD.read(0)
            if(c < 0) then
                break
            end
            USB.print(c.chr)
        end
        SD.close(0)
    end
end
tone(10,1000,100)
digitalWrite(10,1)  # pin10をHighに (tone off)
USB.println("Clear screen")
if(1==System.useTFTc()) then
    TFTc.fillScreen(TFTc.color(0))
end #if
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
# 基本的なグラフィック描画メソッドの動作

if(1==System.useTFTc()) then

    #drawCircle & fillCircle test
    x=50;y=50;
    TFTc.drawCircle( x ,y    ,15,TFTc.color(6))
    TFTc.drawCircle( x ,y+5 ,12,TFTc.color(7))
    TFTc.fillCircle( x ,y+10,10,C_YELLOW)
    TFTc.fillCircle( x ,y    ,5 ,C_GRAY)

    #drawPixel test
    x=100;y=50;a=20;
    TFTc.drawPixel(x   ,y   ,TFTc.color(1))
    TFTc.drawPixel(x   ,y+a)
    TFTc.drawPixel(x+a,y+a)
    TFTc.drawPixel(x+a,y    ,0xcb43)
    TFTc.drawPixel(x+a/2 ,y+a/2 ,TFTc.color(4))

    #drawLine ,drawLineto test
    x=150;y=40;
    10.step(35,5){ |a|
        TFTc.drawLine(x ,y ,x,  y+a,TFTc.color(5))
        TFTc.drawLineto(x+a,y+a)
        TFTc.drawLineto(x+a,y    )
        TFTc.drawLineto(x   ,y    )
        TFTc.drawLineto(x+a/2 ,y+a/2 ,TFTc.color(7))
    }

    #drawRect & fillRect test
    x=50;y=100;
    TFTc.drawRect(x    ,y    ,10,4,TFTc.color(8))
    TFTc.drawRect(x+5 ,y+5 ,20,8)
    TFTc.fillRect(x+10,y+10,15,20)
    TFTc.fillRect(x+20,y+15,5 ,10,TFTc.color(9))

    #drawRoundRect & fillRoundRect test
    x=100;y=100;
    TFTc.drawRoundRect(x    ,y    ,20,14,5,TFTc.color(8))
    TFTc.drawRoundRect(x+5 ,y+5 ,25,20,4)
    TFTc.fillRoundRect(x+10,y+10,15,30,3)
    TFTc.fillRoundRect(x+20,y+15,5,10,2,TFTc.color(9))

    #drawTriangle & fillTriangle test
    x=180;y=110;
    TFTc.fillTriangle(x    ,y    ,x+5 ,y+10,x-5 ,y+10,TFTc.color(2))
    x=150;y=100;
    TFTc.drawTriangle(x+10,y+10,x+15,y+30,x+5 ,y+30,C_ORANGE)
    TFTc.fillTriangle(x+15,y+20,x+23,y+35,x+7 ,y+35,C_BROWN)
    TFTc.drawTriangle(x+25,y+30,x+35,y+45,x+15,y+45,C_YELLOW)

    TFTc.drawRoundRect(20,20,200,150,5,TFTc.color(8))
    x=120;y=240;
    for i in 1..5 do
        for j in 1..5 do
            TFTc.drawCircle(x+j*2,y+j*2,i*10,TFTc.color(6)-j*3-i*2)
        end
    end

end #if
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
#Screen rotation test

if(1==System.useTFTc()) then
    for @Rot in 0..4 do
        TFTc.fillScreen(TFTc.color(11))
        TFTc.fillScreen(0x1111)
        TFTc.textCursor(0,0)
        TFTc.print("abcd")
        w=TFTc.width()
        h=TFTc.height()
        Usb.print("w=");Usb.print(w.to_s);Usb.print("h=");Usb.println(h.to_s);
        @Rot += 1
        if(4==@Rot) then
            @Rot = 0
        end
        TFTc.rotation(@Rot)
        Usb.print("rotation");Usb.println(@Rot.to_s);
        w=TFTc.width()
        h=TFTc.height()
        Usb.print("w=");Usb.print(w.to_s);Usb.print("h=");Usb.println(h.to_s);
        delay(1000)
    end #for
end #if
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
#Font test
if(1==System.useTFTc()) then
    TFTc.fillScreen(TFTc.color(0))
    TFTc.textFont(0)     # デフォルトフォントに設定
    TFTc.textSize(0)     # 倍率を1に設定
    TFTc.textCursor(10,0)
    TFTc.textColor(TFTc.color(2))
    TFTc.print("Size:0,RED ")
    TFTc.textColor(TFTc.color(4))
    TFTc.println("YELLOW")
    TFTc.textColor(TFTc.color(3))
    TFTc.textWrap(0)
    TFTc.println("Text Wrap 0,ORANGE:")
    TFTc.println("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefg
hijklmnopqrstuvwxyz_0123456789.!?#")
    TFTc.textWrap(1)
    TFTc.textColor(TFTc.color(5))
    TFTc.println("Text Wrap 1,GREEN :");
    TFTc.println("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefg
hijklmnopqrstuvwxyz_0123456789.!?#")
    TFTc.textSize(1)
    TFTc.textColor(TFTc.color(6))
    TFTc.println("Size:1,BLUE: ABC")
    TFTc.textSize(2)
    TFTc.textColor(TFTc.color(7))
    TFTc.println("Size:2,PURPLE: ABC")
    TFTc.textSize(3)
    TFTc.textColor(TFTc.color(8))
    TFTc.println("Size:3,GRAY: ABC")
    TFTc.textSize(1)
    TFTc.textCursor(50,130)
    TFTc.println("Size:1,cursor(50,100)[CR]")
    TFTc.textColor(TFTc.color(9))
    TFTc.print("White:ABCD")
    TFTc.textColor(TFTc.color(10))
    TFTc.println("CYAN:EFGHI")
    TFTc.textSize(2)
    TFTc.textColor(TFTc.color(0),TFTc.color(2))
    TFTc.print("                ")
    TFTc.print("     BLACK/RED  ")
    TFTc.print("                ")
    TFTc.print("                ")
    TFTc.drawChar(100,200,"*",TFTc.color(4),0,1)
    TFTc.drawChar(108,192,"*",TFTc.color(4),0,1)
    TFTc.drawChar(116,200,"*",TFTc.color(4),0,1)
    TFTc.textCursor(0,210)
    TFTc.textSize(1)
    TFTc.textColor(11)
    for i in 1..8 do
        TFTc.print("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefg
hijklmnopqrstuvwxyz_0123456789.!?#")
    end
    TFTc.drawChar(100,250,"A",TFTc.color(4),0,5)
    # 以下はフリーフォントを組み込んでいる場合に実行可能
    # TFTc.textCursor(0,100)
    # TFTc.textFont(5)
    # TFTc.print("ab")
    # TFTc.textFont(6)
    # TFTc.print("cd")
    # TFTc.textFont(7)
    # TFTc.print("ef")
    # TFTc.textFont(8)
    # TFTc.println("g")
    TFTc.textFont(0)
    TFTc.textSize(0)
end #if
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
# Draw Bitmap test (Only 24bit color BMP file)

Usb.println("Checking SD")
if (0 == System.useSD()) then
    Usb.println("SD card is not inserted.")
else
    Usb.println("SD card is ready.")
    if(1==System.useTFTc()) then
        Usb.println("TFT is ready.")
        delay(100)
        Usb.println("Draw 24bit color Bitmap.")
        a=TFTc.drawBmp("purple.bmp",0,0)
        Usb.print("drawBmp return code is :"); Usb.println(a.to_s)
    else
        Usb.println("TFT is not ready.")
    end #if
    SD.close(0)
end
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
# Button test 1    ボタン使用例（簡易）押下あるいはリリースのどちらかを検出する場合
if(1==System.useTFTc()) then
    TFTc.fillScreen(TFTc.color(0))
    #button初期化
    TFTc.setButtonMax(5)
    TFTc.setDisplayButtonMax(3)
    TFTc.initButton(1,50,100,50,50,TFTc.color(9),TFTc.color(0),C_YELLOW,"On",2)
    TFTc.initButton(2,120,100,50,50,TFTc.color(9),TFTc.color(0),C_RED,"Off",2)
    TFTc.initButton(3,195,100,60,50,TFTc.color(9),TFTc.color(0),C_BLUE,"Exit",2)
    TFTc.drawButtons()
    TFTc.searchTouchedButton(-1,-1);    # buttonのタッチ履歴をクリア
    while(true) do
        x=(-1); y=(-1); # release トラッキング用に座標をクリアする。
        if (TFTc. panelTouched() == 1) then
            x=TFTc.touchedX;
            y=TFTc.touchedY;
        end #if
        TFTc.searchTouchedButton(x,y)
        bno = TFTc.searchReleasedButton()  # リリース時検出の場合
        if(bno!=0) then
            Usb.print("Released Btn no:");Usb.println(bno.to_s)
        end #if
        bno = TFTc.searchPressedButton() # 押下時検出の場合
        if(bno!=0) then
            Usb.print("Pressed Btn no:");Usb.println(bno.to_s)
        end #if
        if (bno==3) then # 押下/リリースにかかわらずモードを抜ける。
            TFTc.deleteButtons()   # 生成したボタンのインスタンスを全て消去(ここでは5個分)
            TFTc.fillScreen(TFTc.color(0))
            break   #exit this mode
        end
    end #while
end #if
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
# Button test 2   ボタン使用例（詳細）：押下とリリースを区別する場合。同時押し検出をする場合。
if(1==System.useTFTc()) then
    TFTc.fillScreen(TFTc.color(0))
    TFTc.setButtonMax(5)
    TFTc.setDisplayButtonMax(3)
    TFTc.initButton(1,50,100,50,50,TFTc.color(9),TFTc.color(0),TFTc.color(4),"On",2)
    TFTc.initButton(2,120,100,50,50,TFTc.color(9),TFTc.color(0),TFTc.color(2),"Off",2)
    TFTc.initButton(3,195,100,60,50,TFTc.color(9),TFTc.color(0),TFTc.color(6),"Exit",2)
    TFTc.drawButtons()
    TFTc.searchTouchedButton(-1,-1);    # buttonのタッチ履歴をクリア
    exit_flag=0;
    while(true) do
        x=(-1); y=(-1); # release トラッキング用に座標をクリアする
        if (TFTc. panelTouched() == 1) then
            x=TFTc.touchedX;  y=TFTc.touchedY;
        end #if
        TFTc.searchTouchedButton(x,y)
        for i in 1..3 do
            ret = TFTc.searchButtonState(i)
            case ret
            when -1 then
                Usb.print("Pressed Btn no:");Usb.println(i.to_s)
            when 1 then
                Usb.print("Released Btn no:");Usb.println(i.to_s)
                if (3==i) then
                    exit_flag=1
                end
            end #case
        end #for
        if exit_flag == 1 then
            TFTc.deleteButtons()   # 生成したボタンのインスタンスを全て消去(ここでは5個分)
            TFTc.fillScreen(TFTc.color(0))
            break   #exit this mode
        end #if
    end #while
end #if
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
# Draw Screen test    Rubyにてボタン生成、タッチ検出、グラフィック描画を制御
if(1==System.useTFTc()) then
    TFTc.fillScreen(TFTc.color(0))
    w=TFTc.width()  #ディスプレイの幅を取得
    h=TFTc.height() #ディスプレイの高さを取得
    #button初期化
    TFTc.setButtonMax(10) #ボタンのインスタンスを生成
    TFTc.setDisplayButtonMax(8) # この画面で使用するボタン数を指定
    TFTc.initButton(1,20,20,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(2),"RED",1)
    TFTc.initButton(2,60,20,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(4),"YELLOW",1)
    TFTc.initButton(3,100,20,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(5),"GREEN",1)
    TFTc.initButton(4,140,20,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(10),"CYAN",1)
    TFTc.initButton(5,180,20,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(6),"BLUE",1)
    TFTc.initButton(6,220,20,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(11),"MAGEN",1)
    TFTc.initButton(7,220,60,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(1),"EXIT",1)
    TFTc.initButton(8,20,60,40,40,TFTc.color(9),TFTc.color(0),TFTc.color(7),"CLEAR",1)
    TFTc.drawButtons()
    TFTc.searchTouchedButton(-1,-1);    # buttonのタッチ履歴をクリア
    currentcolor = 2   # RED
    old_bno = 1 # ハイライトのための処理:前回タッチしてハイライトされているボタンno
    bno = 1     # ハイライトのための処理:現在のタッチ検出ボタンno
    TFTc.drawButton(1,1);    # button 1 を反転させる
    exit_flag = 0
    while(true) do
        x=(-1); y=(-1); # releaseトラッキング用に座標をクリアする。
        if (TFTc. panelTouched() == 1) then # パネル全体のタッチ検出
            x=TFTc.touchedX;
            y=TFTc.touchedY;
        end #if
        if(x<w or y<h) then
            TFTc.searchTouchedButton(x,y) # 全てのボタンについてタッチされたか調査
            bno = TFTc.searchReleasedButton()  # リリースされたボタンを検出
            if(bno!=0) then # 何れかのボタンがタッチされていた場合
                Usb.print("Released Btn no:");Usb.println(bno.to_s)
                if(bno!=8) then     #no8のボタンは画面クリア用なので、反転させる必要なし
                    TFTc.drawButton(old_bno,0) # 前回タッチされたボタンの反転表示をクリア
                    TFTc.drawButton(bno,1)     # 今回タッチされたボタンを反転表示する
```

メソッドの説明(V2ライブラリ)

TFTcクラス

System.useTFTc()を呼んでおく必要があります。

使用例

```
                    old_bno = bno               # 「前回タッチされたボタンno」を記憶する
                end #if
            end #if
            case bno
            when 1 then
                currentcolor = 2;
            when 2 then
                currentcolor = 4;
            when 3 then
                currentcolor = 5;
            when 4 then
                currentcolor = 10;
            when 5 then
                currentcolor = 6;
            when 6 then
                currentcolor = 11;
            when 7 then # EXIT ボタンでこのモードを終了する
                TFTc.deleteButtons()    # 生成したボタンのインスタンスを全て消去(ここでは10個分)
                TFTc.fillScreen(TFTc.color(0)) # 画面を暗転する
                exit_flag=1
            when 8 then # 画面をクリアする
                TFTc.fillScreen(TFTc.color(0))
                TFTc.drawButtons()
                TFTc.drawButton(old_bno,1)      # 前回タッチされたボタンを反転表示する
            when 0 then
                if x!=(-1) then #画面がタッチされており、かつボタンがタッチされていない場合
                    TFTc.fillCircle(x,y,3,TFTc.color(currentcolor))
                end # then
            end #when
        end #if
        if exit_flag==1 then
            break   #exit while(true)
        end #if
    end #while
end #if
```