TK-850/JH3U-SP

グラフィック・ライブラリ

アプリケーション・ノート

（第 1.0 版）

テセラ･テクノロジー株式会社

目次

図表の目次

第1章　サンプル・プログラムの概要

この章では、V850 向けグラフィック・ライブラリを使用したサンプル・プログラムの概要を説明します。

1.1. V850 向けグラフィック・ライブラリとは

V850 向けグラフィック・ライブラリは、LCD コントローラを接続してグラフィックの表示が可能なV850マイコン搭載のボードに対して図形の描画やGUI等のグラフィック機能を提供する汎用ライブラリです。以下のグラフィック機能を実装しています。

表 1-1　グラフィック・ライブラリの実装機能一覧

種類	機能
図形描画	直線の描画
	長方形の描画
	長方形の描画と塗りつぶし
	円の描画
	円の描画と塗りつぶし
	楕円の描画
	楕円の描画と塗りつぶし
	円弧の描画
	円弧の描画と塗りつぶし
	領域の塗りつぶし
	文字の描画
	文字列の描画
	画像の描画
GUI	コマンド・ボタン
	リスト・ボックス
	チェック・ボックス
	オプション・ボタン
	ラベル
	フレーム

グラフィック・ライブラリについての詳細は、「グラフィック・ライブラリ　ユーザーズ・マニュアル」をご覧ください。

1.2. サンプル・プログラム

本サンプル・プログラムは「基本図形表示サンプル」を PM+のプロジェクト形式で提供しています。

表 1-2　サンプル・プログラム一覧

サンプル・プログラムの格納フォルダ	説明
TK850¥JH3U_graphic¥include	サンプル・プログラムの共用ヘッダ・ファイル
TK850¥ JH3U_graphic ¥src	サンプル・プログラムの共用ソース・ファイル
TK850¥ JH3U_graphic ¥graphic_sample	基本図形表示サンプル・プロジェクト
TK850¥ JH3U_graphic ¥bin	ビットマップ変換ツール
TK850¥ JH3U_graphic ¥lib	グラフィック・ライブラリ

第2章　サンプル・プログラム

サンプル・プログラムは、サンプルで使用するプログラムです。この章では、サンプル・プログラムについて解説します。

2.1. サンプル・プログラムのファイル

2.1.1. ヘッダ・ファイル

サンプル・プログラムのヘッダ・ファイルの格納フォルダは、“TK850¥ JH3U_graphic ¥include”です。

表 2-1　サンプル・プログラムのヘッダ・ファイル一覧

ファイル名	説明
graphic_driver.h	ドライバ関連のヘッダ・ファイルです。
font_config.h	フォント関連のヘッダ・ファイルです。
timer.h	タイマー処理機能のヘッダ・ファイルです。
sample_util.h	基本機能のヘッダ・ファイルです。
graphic_color.h	色のマクロ定義を格納したヘッダ・ファイルです。

2.1.2. ソース・ファイル

サンプル・プログラムのソース・ファイルの格納フォルダは、“TK850¥ JH3U_graphic ¥src”です。

表 2-2　サンプル・プログラムのソース・ファイル一覧

ファイル名	説明
start.s	スタートアップ・ファイルです。
font_config.c	フォント関連のソース・ファイルです。
graphic_driver.c	ドライバ関連のソース・ファイルです。
timer.c	タイマー処理機能のソース・ファイルです。
sample_util.c	基本機能のソース・ファイルです。

2.2. サンプル・プログラムのマクロ

サンプル・プログラムでは、各サンプルで共通に使用するためのマクロを定義しています。

表 2-3　サンプル・プログラムのオブジェクト形式マクロ一覧

マクロ	値	意味
LCD_WIDTH	320	LCD の幅です。graphic_driver.h で定義しています。
LCD_HEIGHT	240	LCD の高さです。graphic_driver.h で定義しています。
TOUCH_GET_CHECK_X	3	連続して得られるまで繰り返す x 座標の範囲を定義しています。
TOUCH_GET_CHECK_Y	3	連続して得られるまで繰り返す y 座標の範囲を定義しています。
COLOR_*	0x????	色です。代表的な 79 色を graphic_color.h で定義しています。

2.3. サンプル・プログラムの関数

サンプル・プログラムでは、以下の関数を定義しています。

表 2-4　サンプル・プログラムの関数一覧

関数名	説明
get_pixel	指定された座標の色を取得します。 グラフィック・ライブラリで使用します。
set_pixel	指定された座標に指定された色を描画します。 グラフィック・ライブラリで使用します。
get_touch	タッチした座標を取得します。 グラフィック・ライブラリで使用します。
clear_touch	タッチした座標の情報をクリアします。 グラフィック・ライブラリで使用します。
lcd_init	LCD を初期化します。
clear_framebuffer	LCD 全体の表示をクリアします。
get_timer	ミリ秒単位のタイマーの値を取得します。
elapsed_systime	指定されたタイマー値からの経過時間を取得します。
elapsed_systimer	指定されたタイマー値からの経過時間と現在の値を取得します。
msleep	指定された時間待ちます。
save_pixel	LCD 上の指定された範囲を保存します。
load_pixel	save_pixel で保存していたデータを LCD 上の指定された範囲に描画します。

2.3.1. get_pixel

get_pixel は、graphic_driver.c で定義しています。

【機能】

指定された座標の色を取得します。
この関数は、グラフィック・ライブラリで使用するドライバ関数です。

【C 言語形式】

unsigned int get_pixel(unsigned short x, unsigned short y);

【パラメータ】

I/O	パラメータ	説明
I	unsigned short x	色を取得する x 座標です。
I	unsigned short y	色を取得する y 座標です。

【返却値】

値	内容
ALL	取得した色です。

【定義】

```
～
#define LCD_BUFFER              0x680000
#define GET_PIXEL_MEMORY(x,y)   ((unsigned short *)LCD_BUFFER + ((x) + (y) * LCD_WIDTH))   ①
～
unsigned int
get_pixel(unsigned short x, unsigned short y)
{
    return (unsigned int)*GET_PIXEL_MEMORY(x, y);   ·····································   ②
}
```

① 指定された座標に対応するグラフィックスメモリ内のポインタを得るマクロ定義です。

② 指定された座標の値をグラフィックスメモリから取得し、unsigned int 型として返却します。

2.3.2. set_pixel

set_pixel は、graphic_driver.c で定義しています。

【機能】

指定された座標に指定された色を描画します。
この関数は、グラフィック・ライブラリで使用するドライバ関数です。

【C 言語形式】

void set_pixel(unsigned short x, unsigned short y, unsigned int color);

【パラメータ】

I/O	パラメータ	説明
I	unsigned short x	描画する x 座標です。
I	unsigned short y	描画する y 座標です。
I	unsigned int color	描画する色です。

【返却値】

なし

【定義】

```
～
#define LCD_BUFFER              0x680000
#define GET_PIXEL_MEMORY(x,y)   ((unsigned short *)LCD_BUFFER + ((x) + (y) * LCD_WIDTH))   ①
～
```

```
void
set_pixel(unsigned short x, unsigned short y, unsigned int color)
{
    *GET_PIXEL_MEMORY(x, y) = (unsigned short)color;   ·····   ②
}
```

① 指定された座標に対応するグラフィックスメモリ内のポインタを得るマクロ定義です。

② グラフィックスメモリの指定された座標に対応する位置に指定された色を設定します。

2.3.3. get_touch

get_touch は、graphic_driver.c で定義しています。

【機能】

タッチした座標を取得します。
この関数は、グラフィック・ライブラリで使用するドライバ関数です。

【C 言語形式】

unsigned char get_touch(unsigned short *x, unsigned short *y);

【パラメータ】

I/O	パラメータ	説明
O	unsigned short *x	タッチした x 座標を格納する領域です。
O	unsigned short *y	タッチした y 座標を格納する領域です。

【返却値】

値	内容
0	LCD にタッチしていません。
1	LCD にタッチしました。

【定義】

```
～
static short                    touch_pos_x = -1;   ·····   ①
static short                    touch_pos_y = -1;   ·····   ②
～
unsigned char
get_touch(unsigned short *x, unsigned short *y)
{
    int     diffx;
    int     diffy;
    int     i;
    short   posx[2] = {0, 0};
    short   posy[2] = {0, 0};

    if (touch_pos_x < 0 || touch_pos_y < 0) {       ·····   ③
        i = 0;
        touch_read(&posx[i], &posy[i]);
        do {
            i ^= 1;
```

```
            touch_read(&posx[i], &posy[i]);          ·····  ④
            if (posx[0] < posx[1]) {
                diffx = posx[1] - posx[0];
            }
            else {
                diffx = posx[0] - posx[1];
            }
            if (posy[0] < posy[1]) {
                diffy = posy[1] - posy[0];
            }
            else {
                diffy = posy[0] - posy[1];
            }
        } while (diffx > TOUCH_GET_CHECK_X || diffy > TOUCH_GET_CHECK_Y);      ·····  ⑤
        touch_pos_x = posx[i];                       ·····  ⑥
        touch_pos_y = posy[i];                       ·····  ⑦
    }
    if (touch_pos_x >= 0 && touch_pos_y >= 0) {      ·····  ⑧
        if (x != (unsigned short *)0) {
            *x = (unsigned short)touch_pos_x;        ·····  ⑨
        }
        if (y != (unsigned short *)0) {
            *y = (unsigned short)touch_pos_y;        ·····  ⑩
        }
        return 1;                                    ·····  ⑪
    }

    return 0;                                        ·····  ⑫
}
```

① 取得したｘ座標を格納しておく変数です。
タッチしたｘ座標を取得していない場合は-1，取得している場合は 0 以上の座標を格納します。

② 取得したｙ座標を格納しておく変数です。
タッチしたｙ座標を取得していない場合は-1，取得している場合は 0 以上の座標を格納します。

③ ｘ座標またはｙ座標を取得していない場合は、取得を試みます。

④ touch_read()は LCD 上でタッチされた情報を読み込み、座標の値に変換する関数です。
LCD にタッチしていない場合の座標は-1 になります。

⑤ イレギュラーな値を読み込む場合を考慮して、指定した範囲内の座標を連続して得られるまで座標の読み込みを繰り返します。

⑥ 得られたｘ座標を touch_pos_x 変数に格納します。

⑦ 得られたｙ座標を touch_pos_y 変数に格納します。

⑧ ｘ座標，ｙ座標ともに取得している場合は、パラメータで指定された領域に座標を格納して終了します。

⑨ ｘ座標を格納する領域が指定されている場合は、取得したｘ座標を指定された領域に格納

します。

⑩ y 座標を格納する領域が指定されている場合は、取得した y 座標を指定された領域に格納します。

⑪ タッチした座標を取得できたので 1 を返却します。

⑫ タッチした座標を取得できなかったので 0 を返却します。

2.3.4. clear_touch

clear_touch は、graphic_driver.c で定義しています。

【機能】

タッチした座標の情報をクリアします。
この関数は、グラフィック・ライブラリで使用するドライバ関数です。

【C 言語形式】

void clear_touch(void);

【パラメータ】

なし

【返却値】

なし

【定義】

```
～
static short                    touch_pos_x = -1;   ·····································   ①
static short                    touch_pos_y = -1;   ·····································   ②
～
void
clear_touch(void)
{
    touch_pos_x = -1;                               ·····································   ③
    touch_pos_y = -1;                               ·····································   ④
}
```

① 取得した x 座標を格納しておく変数です。
タッチした x 座標を取得していない場合は-1，取得している場合は 0 以上の座標を格納します。

② 取得した y 座標を格納しておく変数です。
タッチした y 座標を取得していない場合は-1，取得している場合は 0 以上の座標を格納します。

③ touch_pos_x 変数へ-1 を格納することにより、 x 座標を取得していない状態になります。

④ touch_pos_y 変数へ-1 を格納することにより、 y 座標を取得していない状態になります。

2.3.5. lcd_init

lcd_init は、graphic_driver.c で定義しています。

【機能】
LCD を初期化します。

【C 言語形式】
void lcd_init(void);

【パラメータ】
なし

【返却値】
なし

【定義】

```
void
lcd_init(void)
{
    *((volatile unsigned long *)(0x600004)) = 0x00000000; ·····························  ①
    *((volatile unsigned long *)(0x600008)) = 0x00000032;
～
    clear_framebuffer();                ·····························  ②
}
```

① レジスタの設定を行うことで、LCD を初期化します。

② 画面をクリアします。

2.3.6. clear_framebuffer

clear_framebuffer は、graphic_driver.c で定義しています。

【機能】
LCD 全体の表示をクリアします。

【C 言語形式】
void clear_framebuffer(void);

【パラメータ】
なし

【返却値】
なし

【定義】

```
～
#define LCD_BUFFER              0x680000                     ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ ①
～
void
clear_framebuffer(void)
{
    unsigned short  *p  = (unsigned short *)LCD_BUFFER;
    int             i;

    for (i = LCD_WIDTH * LCD_HEIGHT; i > 0; i--) {      ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ ②
        *p = 0;
        p++;
    }
}
```

① グラフィックスメモリのアドレスのマクロ定義です。

② LCD の幅×高さ分のグラフィックスメモリの領域をクリアすることにより、LCD の表示をクリアします。

2.3.7. get_timer

get_timer は、timer.c で定義しています。

【機能】
ミリ秒単位のタイマーの値を取得します。

【C 言語形式】
unsigned int get_timer(void);

【パラメータ】
なし

【返却値】

値	内容
ALL	タイマーの値です。

【定義】

```
～
static volatile unsigned int        p0_timer    = 0;   ·····  ①

/*---------------------------------------------------------------------------*/
#pragma interrupt   INTTPOCC0   ms_counter
__interrupt void
ms_counter(void)                                       ·····  ②
{
    p0_timer++;
}
/*---------------------------------------------------------------------------*/
unsigned int
get_timer(void)
{
    return p0_timer;                                   ·····  ③
}
```

① タイマーの値を格納しておく変数です。

② タイマーの割り込みハンドラです。
1 ミリ秒毎に割り込みが発生し、p0_timer 変数に 1 を加算します。

③ タイマーの値を返却します。

2.3.8. elapsed_systime

elapsed_systime は、timer.c で定義しています。

【機能】
指定されたタイマー値からの経過時間を取得します。

【C 言語形式】
unsigned int elapsed_systime(unsigned int past);

【パラメータ】

I/O	パラメータ	説明
I	unsigned int past	タイマーの値です。

【返却値】

値	内容
ALL	経過時間です。

【定義】

```
unsigned int
elapsed_systime(unsigned int past)
{
```

```
    unsigned int    work_cnt    = p0_timer;                ·····  ①

    if (work_cnt < past) {
        work_cnt += ((unsigned int)0xffffffff - past);     ·····  ②
        work_cnt++;
    }
    else {
        work_cnt -= past;                                  ·····  ②
    }

    return work_cnt;                                       ·····  ③
}
```

① 現在のタイマーの値を設定します。

② 指定されたタイマー値から現在までの経過時間を求めます。

③ 求めた経過時間を返却します。

2.3.9. elapsed_systimer

elapsed_systimer は、timer.c で定義しています。

【機能】

指定されたタイマー値からの経過時間と現在の値を取得します。

【C 言語形式】

unsigned int elapsed_systimer(unsigned int *time);

【パラメータ】

I/O	パラメータ	説明
I/O	unsigned int *time	タイマーの値を格納する領域です。

【返却値】

値	内容
ALL	経過時間です。

【定義】

```
unsigned int
elapsed_systimer(unsigned int *time)
{
    unsigned int    work_cnt;

    if (time == (unsigned int *)0) {
        return 0;                                      ·····  ①
    }
    work_cnt = *time;                                  ·····  ②
    *time = p0_timer;                                  ·····  ③

    if (*time < work_cnt) {
```

```
        work_cnt ^= (unsigned int)0xffffffff;    ····· ④
        work_cnt++;
        work_cnt += *time;
    }
    else {
        work_cnt = *time - work_cnt;             ····· ④
    }

    return work_cnt;                             ····· ⑤
}
```

① タイマーの値を格納する領域が指定されなかった場合は、0 を返却します。

② 指定されたタイマーの値を設定します。

③ 現在のタイマーの値を指定された領域へ格納します。

④ 指定されたタイマー値から現在までの経過時間を求めます。

⑤ 求めた経過時間を返却します。

2.3.10. msleep

msleep は、timer.c で定義しています。

【機能】

指定された時間待ちます。

【C 言語形式】

void msleep(unsigned int msec);

【パラメータ】

I/O	パラメータ	説明
I	unsigned int msec	ミリ秒単位の時間です。

【返却値】

なし

【定義】

```
void
msleep(unsigned int msec)
{
    unsigned int    start   = get_timer();       ····· ①

    if (msec <= 0) {
        return;                                  ····· ②
    }

    do {
```

```
        uwait(1);
    } while(msec > elapsed_systime(start));      ····· ③
}
```

① 現在のタイマーの値を設定します。

② 指定された時間が 0 の場合は終了します。

③ 指定された時間が経過するまで待ちます。

2.3.11. save_pixel

save_pixel は、sample_util.c で定義しています。

【機能】

LCD 上の指定された範囲を保存します。

【C 言語形式】

void save_pixel(unsigned short x, unsigned short y,
unsigned short width, unsigned short height);

【パラメータ】

I/O	パラメータ	説明
I	unsigned short x	保存する範囲の始点の x 座標です。
I	unsigned short y	保存する範囲の始点の y 座標です。
I	unsigned short width	保存する範囲の幅です。
I	unsigned short height	保存する範囲の高さです。

【返却値】

なし

【定義】

```
#define SAVE_BUF(a)     *(((unsigned short *)0110000)+(a))      ····· ①
/*--------------------------------------------------------------------------*/
void
save_pixel(unsigned short x, unsigned short y, unsigned short width, unsigned short height)
{
    int             i   = 0;
    unsigned short  z;

    width += x;                                                 ····· ②
    height += y;                                                ····· ③
    while (y < height) {                                        ····· ④
        for (z = x; z < width; z++) {                           ····· ⑤
            SAVE_BUF(i) = (unsigned short)get_pixel(z, y);      ····· ⑥
            i++;
        }
        y++;
    }
```

```
}
```

① 保存領域の指定されたオフセットの値のマクロ定義です。
このサンプルでは、保存領域に外部 SRAM を使用しています。

② 終点+1 の x 座標を計算します。

③ 終点+1 の y 座標を計算します。

④ y 座標が始点から終点になるまで繰り返します。

⑤ x 座標が始点から終点になるまで繰り返します。

⑥ 1 ピクセルずつ保存領域へ格納していきます。

2.3.12. load_pixel

load_pixel は、sample_util.c で定義しています。

【機能】
save_pixel で保存していたデータを LCD 上の指定された範囲に描画します。

【C 言語形式】
void load_pixel(unsigned short x, unsigned short y,
unsigned short width, unsigned short height);

【パラメータ】

I/O	パラメータ	説明
I	unsigned short x	描画する範囲の始点の x 座標です。
I	unsigned short y	描画する範囲の始点の y 座標です。
I	unsigned short width	描画する範囲の幅です。
I	unsigned short height	描画する範囲の高さです。

【返却値】
なし

【定義】

```
～
#define SAVE_BUF(a)     *(((unsigned short *)0x110000)+(a))     ·····················   ①
～
void
load_pixel(unsigned short x, unsigned short y, unsigned short width, unsigned short height)
{
    int             i   = 0;
    unsigned short  z;
```

```
    width += x;                                                    ・・・・・・・・・・・・・・・・・・・・・・・・  ②
    height += y;                                                   ・・・・・・・・・・・・・・・・・・・・・・・・  ③
    while (y < height) {                                           ・・・・・・・・・・・・・・・・・・・・・・・・  ④
        for (z = x; z < width; z++) {                              ・・・・・・・・・・・・・・・・・・・・・・・・  ⑤
            set_pixel(z, y, SAVE_BUF(i));                          ・・・・・・・・・・・・・・・・・・・・・・・・  ⑥
            i++;
        }
        y++;
    }
}
```

① 保存領域の指定されたオフセットの値のマクロ定義です。
このサンプルでは、保存領域に外部 SRAM を使用しています。

② 終点+1 の x 座標を計算します。

③ 終点+1 の y 座標を計算します。

④ y 座標が始点から終点になるまで繰り返します。

⑤ x 座標が始点から終点になるまで繰り返します。

⑥ 保存領域のデータを 1 ピクセルずつ LCD 上に描いていきます。

第3章　基本図形表示サンプル

基本図形表示サンプルは、グラフィック・ライブラリを使用して基本的な図形の表示を行うサンプル・プログラムです。この章では、基本図形表示サンプルについて解説します。

3.1.　基本図形表示サンプルの概要

基本図形表示サンプルは、グラフィック・ライブラリの図形描画機能を使用して文字，直線，円，長方形，画像をそれぞれ順番に表示します。また、GUI 機能を使用して図形選択画面を作成しています。図形選択画面は、各図形の表示画面上のタイトル部分にタッチすると表示され、次にどの図形を表示するかを直接選択することができます。

3.2.　基本図形表示サンプルの画面構成

基本図形表示サンプルは、以下の画面で構成されます。

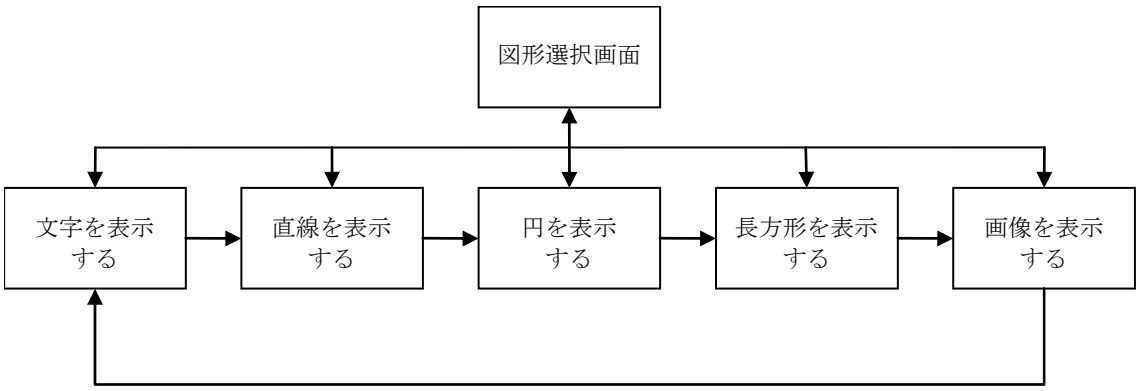

図 3-1　基本図形表示サンプルの画面構成

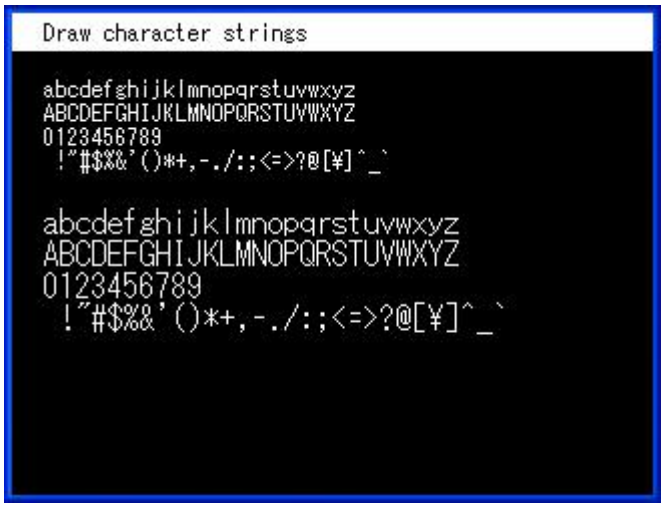

図 3-2　文字を表示する

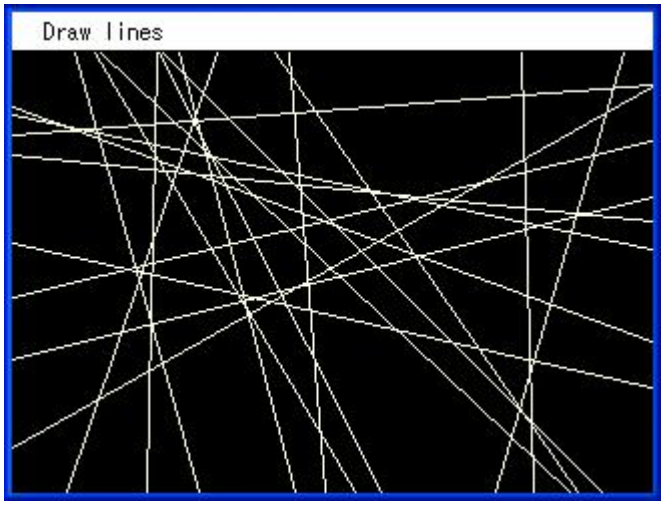

図 3-3　直線を表示する

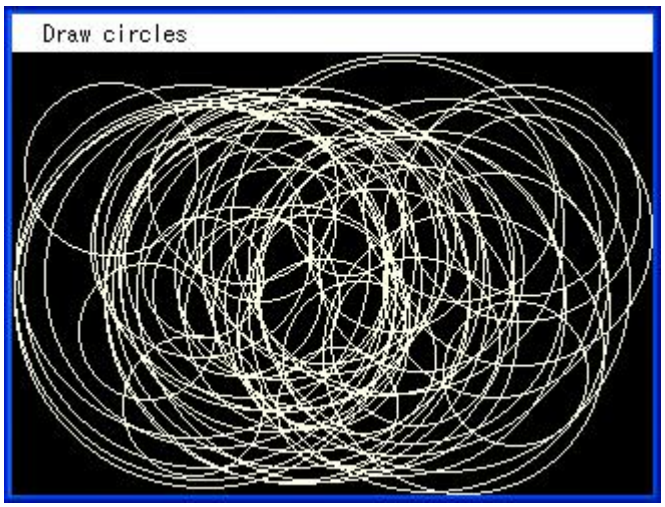

図 3-4　円を表示する

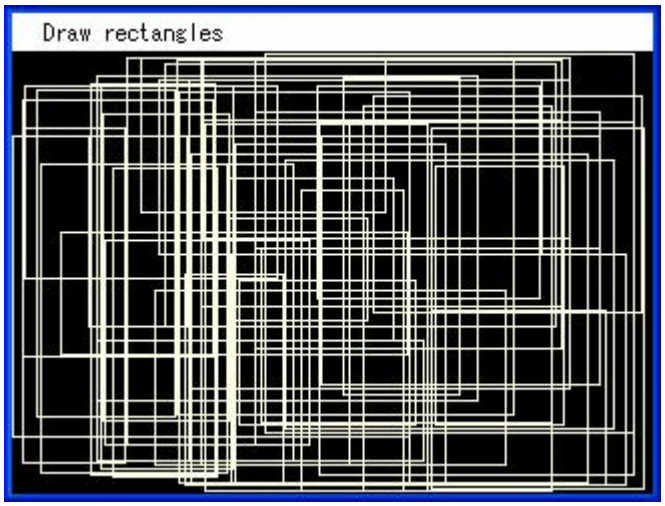

図 3-5　長方形を表示する

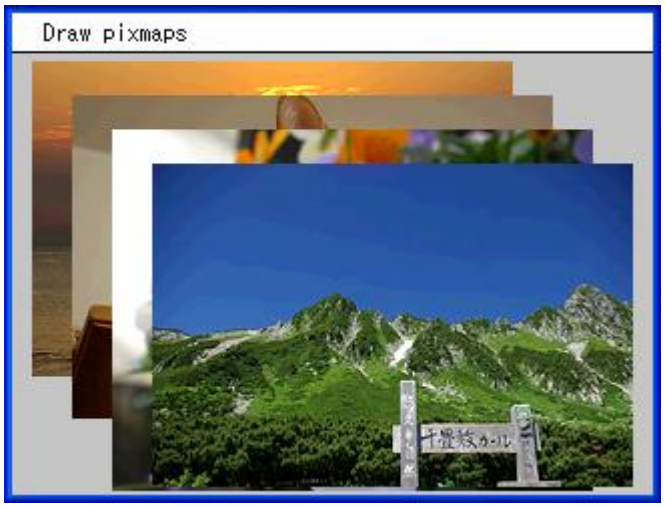

図 3-6　画像を表示する

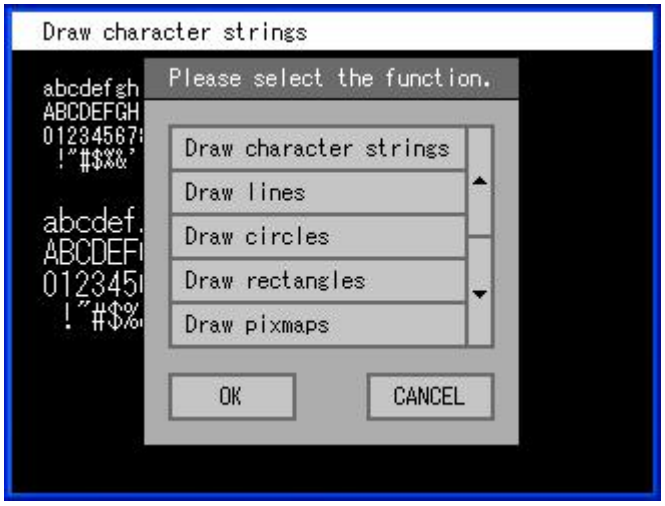

図 3-7　図形選択画面

3.3. 基本図形表示サンプルのファイル

基本図形表示サンプルの格納フォルダは、“TK850¥ JH3U_graphic ¥graphic_sample”です。基本図形表示サンプルには、以下のファイルが存在します。

表 3-1 基本図形表示サンプルのファイル一覧

ファイル名	説明
graphic_sample.prw	PM+ワークスペース・ファイルです。
graphic_sample.prj	PM+プロジェクト・ファイルです。
graphic_sample.pri	PM+プロジェクトの PRI ファイルです。
graphic_sample.cld	PM+プロジェクトの CLD ファイルです。
graphic_sample.dir	リンク・ディレクティブ・ファイルです。
graphic_sample.mak	メイクファイルです。
a.out	実行可能オブジェクト・ファイルです。
romp.out	ROM 化用実行可能オブジェクト・ファイルです。
romp.hex	ヘキサ･フォーマットの実行可能オブジェクト・ファイルです。
include¥graphic_sample.h	基本図形表示サンプルのヘッダ･ファイルです。
src¥main.c	メイン・ルーチンのソース・ファイルです。
src¥graphic_sample.c	図形描画機能のソース・ファイルです。
src¥gui_sample.c	GUI 機能のソース・ファイルです。
src¥pixmap¥*.c	基本図形表示サンプルで使用する画像データです。

3.4. 基本図形表示サンプルのマクロ

基本図形表示サンプルでは、以下のマクロを定義しています。

表 3-2 基本図形表示サンプルのオブジェクト形式マクロ一覧

マクロ	値	意味
TITLE_HEIGHT	20	タイトル部分の高さです。 graphic_sample.h で定義しています。
MENU_ITEM_COUNT	5	図形選択画面のリスト・ボックスに表示される項目の数です。 gui_sample.c で定義しています。

表 3-3 基本図形表示サンプルの関数形式マクロ一覧

マクロ	説明
RANDOM(v)	0 以上，指定された数未満の乱数を得ます。 graphic_sample.c で定義しています。

3.5. 基本図形表示サンプルの構造体

基本図形表示サンプルが独自に使用している構造体はありません。

3.6. 基本図形表示サンプルの定数

基本図形表示サンプルでは、以下の定数を定義しています。

3.6.1. sample_number 列挙型

図形の表示順序を制御するための定数です。sample_number 列挙型は、graphic_sample.h で宣言しています。

```
enum sample_number {
    SAMPLE_TOP,
    SAMPLE_STRING,
    SAMPLE_LINE,
    SAMPLE_CIRCLE,
    SAMPLE_RECTANGLE,
    SAMPLE_PIXMAP,
    SAMPLE_BOTTOM
};
```

① SAMPLE_TOP
先頭を表す番号です。

② SAMPLE_STRING
文字を表示する画面の番号です。

③ SAMPLE_LINE
直線を表示する画面の番号です。

④ SAMPLE_CIRCLE
円を表示する画面の番号です。

⑤ SAMPLE_RECTANGLE
長方形を表示する画面の番号です。

⑥ SAMPLE_PIXMAP
画像を表示する画面の番号です。

⑦ SAMPLE_BOTTOM
末尾を表す番号です。

3.6.2. config 定数

graphic_init()へ渡すための情報を定義した graphic_config 構造体の定数です。config 定数は、main.c で宣言しています。

```
static const graphic_config     config  = {
    get_pixel,
    set_pixel,
    get_touch,
    clear_touch,
    LCD_WIDTH,
    LCD_HEIGHT
};
```

3.6.3. color_list 配列

色をランダムに取得するために使用する色の配列です。color_list 配列は、graphic_sample.c で宣言しています。

```
static const unsigned short     color_list[]    = {
    COLOR_BLACK,
    COLOR_RED,
    COLOR_LIME,
    COLOR_BLUE,
    COLOR_YELLOW,
    ～
    COLOR_MOCCASIN
};
```

3.6.4. title_form_p 定数

図形選択画面を表示させるために、図形表示画面のタイトル部分に配置するコマンド・ボタンのためのフォームのプロパティです。title_form_p 定数は、gui_sample.c で宣言しています。

```
static const form_property      title_form_p    = {
    0,                                          /* x */
    0,                                          /* y */
    0,                                          /* width */
    0,                                          /* height */
    COLOR_BLACK,                                /* background */
```

```
    COLOR_BLACK,                                /* foreground */
    COLOR_BLACK,                                /* border_color */
    0                                           /* border_width */
};
```

3.6.5. title_commandb_p 定数

図形選択画面を表示させるために、図形表示画面のタイトル部分に配置するコマンド・ボタンのプロパティです。title_commandb_p 定数は、gui_sample.c で宣言しています。

```
static const commandb_property  title_commandb_p        = {
    0,                                          /* x */
    0,                                          /* y */
    LCD_WIDTH,                                  /* width */
    TITLE_HEIGHT,                               /* height */
    COLOR_BLACK,                                /* background */
    COLOR_BLACK,                                /* foreground */
    COLOR_BLACK,                                /* border_color */
    0,                                          /* border_width */
    GRAPHIC_FONTSIZE_12x6,                      /* font */
    0,                                          /* name */
    0,                                          /* function */
    0                                           /* argument */
};
```

3.6.6. menu_form_p 定数

図形選択画面のフォームのプロパティです。menu_form_p 定数は、gui_sample.c で宣言しています。

```
static const form_property      menu_form_p     = {
    65,                                         /* x */
    43,                                         /* y */
    189,                                        /* width */
    174,                                        /* height */
    COLOR_DARKGRAY,                             /* background */
    COLOR_DARKGRAY,                             /* foreground */
    COLOR_BLACK,                                /* border_color */
    1                                           /* border_width */
};
```

3.6.7. menu_label_p 定数

図形選択画面のタイトル部分のラベルのプロパティです。menu_label_p 定数は、gui_sample.c で宣言しています。

```
static const label_property     menu_label_p    = {
    65,                                         /* x */
    22,                                         /* y */
    189,                                        /* width */
    22,                                         /* height */
    COLOR_DIMGRAY,                              /* background */
    COLOR_WHITE,                                /* foreground */
    COLOR_BLACK,                                /* border_color */
    1,                                          /* border_width */
    GRAPHIC_FONTSIZE_12x6,                      /* font */
    "  Please select the function."             /* name */
};
```

3.6.8. list_items 配列

図形選択画面のリスト・ボックスに表示される項目名の配列です。list_items 配列は、gui_sample.c で宣言しています。

```
static const char       *list_items[]   = {
    " Draw character strings",
    " Draw lines",
    " Draw circles",
    " Draw rectangles",
    " Draw pixmaps"
};
```

3.6.9. menu_listb_p 定数

図形選択画面のリスト・ボックスのプロパティです。menu_listb_p 定数は、gui_sample.c で宣言しています。

```
static const listb_property     menu_listb_p    = {
    78,                                         /* x */
    56,                                         /* y */
    146,                                        /* item_width */
    20,                                         /* item_height */
    COLOR_SILVER,                               /* background */
```

```
        COLOR_BLACK,                            /* foreground */
        COLOR_DIMGRAY,                          /* border_color */
        2,                                      /* border_width */
        GRAPHIC_FONTSIZE_12x6,                  /* font */
        0xff,                                   /* value */
        5,                                      /* visible_count */
        list_items,                             /* items */
        MENU_ITEM_COUNT,                        /* item_count */
        0                                       /* top_index */
    };
```

3.6.10. menu_commandb_p 配列

図形選択画面のコマンド・ボタンのプロパティです。menu_commandb_p 配列は、gui_sample.c で宣言しています。

```
    static const commandb_property  menu_commandb_p[]       = {
        {
            78,                                 /* x */
            180,                                /* y */
            64,                                 /* width */
            24,                                 /* height */
            COLOR_SILVER,                       /* background */
            COLOR_BLACK,                        /* foreground */
            COLOR_DIMGRAY,                      /* border_color */
            2,                                  /* border_width */
            GRAPHIC_FONTSIZE_12x6,              /* font */
            "OK",                               /* name */
            ok_cb,                              /* function */
            0                                   /* argument */
        },
        {
            177,                                /* x */
            180,                                /* y */
            64,                                 /* width */
            24,                                 /* height */
            COLOR_SILVER,                       /* background */
            COLOR_BLACK,                        /* foreground */
            COLOR_DIMGRAY,                      /* border_color */
            2,                                  /* border_width */
            GRAPHIC_FONTSIZE_12x6,              /* font */
            "CANCEL",                           /* name */
            cancel_cb,                          /* function */
```

```
            0                                   /* argument */
        }
    };
```

3.7. 基本図形表示サンプルの変数

基本図形表示サンプルでは、以下の変数を定義しています。

3.7.1. g_context 変数

グラフィック・ライブラリで使用する graphic_context 構造体の領域です。g_context 変数は、main.c で宣言しています。

```
graphic_context g_context;
```

3.7.2. g_number 変数

図形選択画面で選択された図形の画面番号を格納する変数です。g_number 変数は、gui_sample.c で宣言しています。

```
static int      g_number;
```

3.7.3. title_form 変数

図形選択画面を表示させるために、図形表示画面のタイトル部分に配置するコマンド・ボタンのためのフォーム・オブジェクトの領域です。title_form 変数は、gui_sample.c で宣言しています。

```
static form     title_form;
```

3.7.4. menu_form 変数

図形選択画面のフォーム・オブジェクトの領域です。menu_form 変数は、gui_sample.c で宣言しています。

```
static form     menu_form;
```

3.7.5. menu_label 変数

図形選択画面のタイトル部分のラベル・オブジェクトの領域です。menu_label 変数は、

gui_sample.c で宣言しています。

```
static gui_object       menu_label;
```

3.7.6. menu_listb 変数

図形選択画面のリスト・ボックス・オブジェクトの領域です。menu_listb 変数は、gui_sample.c で宣言しています。

```
static gui_object       menu_listb;
```

3.7.7. title_commandb 変数

図形選択画面を表示させるために、図形表示画面のタイトル部分に配置するコマンド・ボタン・オブジェクトの領域です。title_commandb 変数は、gui_sample.c で宣言しています。

```
static gui_object       title_commandb;
```

3.7.8. menu_commandb 配列

図形選択画面のコマンド・ボタン・オブジェクトの領域です。menu_commandb 配列は、gui_sample.c で宣言しています。

```
static gui_object       menu_commandb[2];
```

3.8. 基本図形表示サンプルの関数

基本図形表示サンプルでは、以下の関数を定義しています。

表 3-4 基本図形表示サンプルの関数一覧

関数名	説明
main	メイン･ルーチンです。
graphic_sample	各図形を順番に表示します。
get_random_color	color_list に設定されている色のうち、黒を除く 78 色のなかから 1 色をランダムに取得します。
display_title	図形表示画面のタイトル部分を表示します。
sample_string	文字を表示します。
sample_line	直線を表示します。
sample_paint	領域を塗りつぶします。
sample_circle	円を表示します。
sample_rectangle	長方形を表示します。
sample_pixmap	画像を表示します。
select_function_init	図形選択画面の表示を受け付けるコマンド・ボタンを作成します。
select_function	図形選択画面の表示，動作を行います。
ok_cb	図形選択画面の OK ボタンを選択したときの処理を行います。
cancel_cb	図形選択画面の CANCEL ボタンを選択したときの処理を行います。
touch_wait	指定された時間だけ、タッチスクリーンの入力を受け付けます。

3.8.1. main

main は、main.c で定義しています。

【機能】

メイン･ルーチンです。グラフィック機能の初期化を行った後に、graphic_sample 関数を呼び出して図形の表示を開始します。

【C 言語形式】

```
int main(int argc, char **argv);
```

【パラメータ】

I/O	パラメータ	説明
I	int argc	main へ渡されるパラメータ数です。
I	char **argv	main へ渡されるパラメータのリストです。

【返却値】

0 を返却します。

【定義】

```
int
main(int argc, char **argv)
{
    int             ret;

    __EI();                                 ..........................................  ①

    /* Initialize */
    lcd_init();                             ..........................................  ②
    ret = graphic_init(&config);            ..........................................  ③
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end;
    }
    ret = g_context_init(&g_context);       ..........................................  ④
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end;
    }

    /* Sample */
    graphic_sample();                       ..........................................  ⑤

end:
    return 0;                               ..........................................  ⑥
}
```

① 割り込みを有効にします。

② LCD を初期化します。

③ グラフィック・ライブラリを初期化します。

④ 使用するコンテキストを初期化します。

⑤ 図形の表示を開始します。

⑥ 0 を返却して終了します。

3.8.2. graphic_sample

graphic_sample は、graphic_sample.c で定義しています。

【機能】

各図形を順番に表示します。

【C 言語形式】

void graphic_sample(void);

【パラメータ】
　なし

【返却値】
　なし

【定義】

```
void
graphic_sample(void)
{
    int             ret;
    int             num = SAMPLE_TOP;        ·····  ①

    /* Initialize */
    ret = select_function_init();            ·····  ②
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end;
    }

    while (1) {
        switch (num) {
            case SAMPLE_STRING:
                ret = sample_string();       ·····  ③
                break;
            case SAMPLE_LINE:
                ret = sample_line();         ·····  ④
                break;
            case SAMPLE_CIRCLE:
                ret = sample_circle();       ·····  ⑤
                break;
            case SAMPLE_RECTANGLE:
                ret = sample_rectangle();    ·····  ⑥
                break;
            case SAMPLE_PIXMAP:
                ret = sample_pixmap();       ·····  ⑦
                break;
            case SAMPLE_TOP:
                ret = 0;                     ·····  ⑧
                break;
            case SAMPLE_BOTTOM:
            default:
                num = SAMPLE_TOP;            ·····  ⑨
                ret = 0;
        }
        num++;                               ·····  ⑩
        if (ret > SAMPLE_TOP && ret < SAMPLE_BOTTOM) {
            num = ret;                       ·····  ⑪
        }
    }

end:
    return;
}
```

① 先頭から開始するために、画面番号に SAMPLE_TOP を設定します。

② 図形選択を初期化します。

③ 画面番号が SAMPLE_STRING の場合は、文字を表示します。

④ 画面番号が SAMPLE_LINE の場合は、直線を表示します。

⑤ 画面番号が SAMPLE_CIRCLE の場合は、円を表示します。

⑥ 画面番号が SAMPLE_RECTANGLE の場合は、長方形を表示します。

⑦ 画面番号が SAMPLE_PIXMAP の場合は、画像を表示します。

⑧ 画面番号 SAMPLE_TOP は、先頭を示す制御上の番号なので、何も表示は行わず結果に 0 を設定します。

⑨ 画面番号が末尾まで到達した場合は、画面番号に SAMPLE_TOP を設定して先頭に戻ります。

⑩ 画面番号に 1 加算して次の画面番号に設定します。

⑪ 図形が選択された場合は、選択された図形の画面番号を設定します。

3.8.3. get_random_color

get_random_color は、graphic_sample.c で定義しています。

【機能】

color_list に設定されている色のうち､黒を除く 78 色のなかから 1 色をランダムに取得します。

【C 言語形式】

static unsigned short get_random_color(void);

【パラメータ】

なし

【返却値】

値	内容
ALL	取得した色です。

【定義】

```
static unsigned short
get_random_color(void)
{
    int          i;

    i = RANDOM(78);              ........................................................  ①
    i += 1;                      ........................................................  ②
```

```
    return color_list[i];    ·····  ③
}
```

① 0～77 の乱数を取得します。

② 1 を加算して 1～78 の乱数にします。
これが、黒を除いた color_list 配列のインデックスになります。

③ color_list 配列の求めたインデックスの値を返却します。

3.8.4. display_title

display_title は、graphic_sample.c で定義しています。

【機能】
図形表示画面のタイトル部分を表示します。

【C 言語形式】
static void display_title(const char *title);

【パラメータ】

I/O	パラメータ	説明
I	const char *title	タイトルとして表示する文字列です。

【返却値】
なし

【定義】

```
static void
display_title(const char *title)
{
    g_rectanglef(&g_context, 0, 0, LCD_WIDTH, TITLE_HEIGHT - 1, COLOR_WHITE);  ·····  ①
    g_string(&g_context, 16, 4, title, GRAPHIC_FONTSIZE_12x6, COLOR_BLACK);    ·····  ②
}
```

① タイトル部分を白で表示します。

② 指定された文字列を表示します。

3.8.5. sample_string

sample_string は、graphic_sample.c で定義しています。

【機能】

文字を表示します。文字の表示中にタイトル部分を選択すると図形選択画面が表示され、表示させたい図形を選択することができます。

【C 言語形式】

static int sample_string(void);

【パラメータ】

なし

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
sample_string(void)
{
    int                 ret;
    int                 y;
    int                 i;
    static const char   *characters[]   = {
        "abcdefghijklmnopqrstuvwxyz",
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "0123456789",
        " !¥"#$%&'()*+,-./:;<=>?@[¥¥]^_`",
        0
    };                                                              .....  ①

    /* Initialize */
    clear_framebuffer();                                            .....  ②

    /* Title */
    display_title("Draw character strings");                        .....  ③

    /* Draw character strings */
    y = TITLE_HEIGHT + 12;
    for (i = 0; characters[i] != (const char *)0; i++) {
        g_string(&g_context, 16, y,
            characters[i], GRAPHIC_FONTSIZE_12x6, COLOR_WHITE);     .....  ④
        y += 12;                                                    .....  ⑤
    }
    for (i = 0; characters[i] != (const char *)0; i++) {
        y += 16;                                                    .....  ⑥
        g_string(&g_context, 16, y,
            characters[i], GRAPHIC_FONTSIZE_16x8, COLOR_WHITE);     .....  ⑦
    }

    /* Wait */
    ret = touch_wait(5000);                                         .....  ⑧

    return ret;                                                     .....  ⑨
}
```

① 表示する文字列のリストです。

② 最初に画面をクリアします。

③ タイトルを表示します。

④ spec_string[]の文字列を 12x6 のフォントで表示していきます。

⑤ 改行するために Y 座標に 12 を加算します。

⑥ 改行するために Y 座標に 16 を加算します。

⑦ spec_string[]の文字列を 16x8 のフォントで表示していきます。

⑧ 5 秒待ちます。この間、図形選択を受け付けます。

⑨ 終了時は、touch_wait()の返却値をそのまま返却します。

3.8.6. sample_line

sample_line は、graphic_sample.c で定義しています。

【機能】

直線を表示します。直線の表示中にタイトル部分を選択すると図形選択画面が表示され、表示させたい図形を選択することができます。

【C 言語形式】

static int sample_line(void);

【パラメータ】

なし

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
sample_line(void)
{
    int             ret;
    int             xy1;
    int             xy2;
    unsigned int    times;
    unsigned int    timeout;
    unsigned int    elapsed;
    unsigned short  num;
```

```
    unsigned short  color1;

    /* Initialize */
    clear_framebuffer();                                                    ·····  ①

    /* Title */
    display_title("Draw lines");                                            ·····  ②

    /* Wait */
    ret = touch_wait(1000);                                                 ·····  ③
    if (ret > 0) {
        goto end;
    }

    /* Draw lines */
    color1 = get_random_color();                                            ·····  ④
    times = get_timer();                                                    ·····  ⑤
    timeout = 5000;
    elapsed = 0;
    while (elapsed < timeout) {                                             ·····  ⑥
        if (elapsed >= 100) {
            ret = touch_wait(0);                                            ·····  ⑦
            if (ret > 0) {
                goto end;
            }
            timeout -= elapsed;                                             ·····  ⑧
            times = get_timer();
        }
        num = RANDOM(2);                                                    ·····  ⑨
        if (num == 0) {
            xy1 = RANDOM(LCD_HEIGHT - TITLE_HEIGHT);
            xy2 = RANDOM(LCD_HEIGHT - TITLE_HEIGHT);
            xy1 += TITLE_HEIGHT;
            xy2 += TITLE_HEIGHT;
            g_line(&g_context, 0, xy1, LCD_WIDTH - 1, xy2, color1);         ·····  ⑩
        }
        else {
            xy1 = RANDOM(LCD_WIDTH);
            xy2 = RANDOM(LCD_WIDTH);
            g_line(&g_context, xy1, TITLE_HEIGHT, xy2, LCD_HEIGHT - 1, color1); ····· ⑪
        }
        elapsed = elapsed_systime(times);                                   ·····  ⑫
    }

end:
    return ret;                                                             ·····  ⑬
}
```

① 最初に画面をクリアします。

② タイトルを表示します。

③ 直線を表示する前に 1 秒待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

④ 表示する直線の色を取得します。

⑤ 経過時間を得るために現在のタイマー値を取得しておきます。

⑥ 5 秒経過するまで繰り返します。

⑦ 100 ミリ秒毎に図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑧ touch_wait()の処理時間が 5 秒の繰り返し時間に含まれないようにするために、touch_wait()の前までの経過時間を減算後、現在のタイマー値を取得し直します。

⑨ 0 か 1 の乱数を取得します。

⑩ 取得した乱数が 0 の場合は、LCD の左端から右端までで Y 座標がランダムな直線を表示します。

⑪ 取得した乱数が 1 の場合は、LCD の上端から下端までで X 座標がランダムな直線を表示します。

⑫ 経過時間を取得します。

⑬ 終了時は、touch_wait()の返却値をそのまま返却します。

3.8.7. sample_paint

sample_paint は、graphic_sample.c で定義しています。

【機能】

領域を塗りつぶします。領域の塗りつぶし中にタイトル部分を選択すると図形選択画面が表示され、表示させたい図形を選択することができます。

【C 言語形式】

static int sample_paint(unsigned short color, unsigned int timeout);

【パラメータ】

I/O	パラメータ	説明
I	unsigned short color	表示済みの線の色です。
I	unsigned int timeout	領域の塗りつぶしを繰り返す時間です。

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
sample_paint(unsigned short color, unsigned int timeout)
{
```

```
    int             ret;
    int             x;
    int             y;
    int             num;
    unsigned int    times;
    unsigned int    elapsed;
    unsigned short  paint_color;

    /* Title */
    display_title("Paint areas");                                          ·····  ①

    /* Wait */
    ret = touch_wait(1000);                                                ·····  ②
    if (ret > 0) {
        goto end;
    }

    /* Paint area */
    do {
        paint_color = get_random_color();                                  ·····  ③
    } while (color == paint_color);
    num = LCD_HEIGHT - TITLE_HEIGHT;
    times = get_timer();                                                   ·····  ④
    while (1) {
        do {
            x = RANDOM(LCD_WIDTH);
            y = RANDOM(num);
            y += TITLE_HEIGHT;
            color = (unsigned short)g_context.config->get_pixel(x, y);     ·····  ⑤
            elapsed = elapsed_systime(times);                              ·····  ⑥
            if (elapsed >= timeout) {
                goto end;
            }
            if (elapsed >= 100) {
                ret = touch_wait(0);                                       ·····  ⑦
                if (ret > 0) {
                    goto end;
                }
                timeout -= elapsed;                                        ·····  ⑧
                times = get_timer();
            }
        } while (color != COLOR_BLACK);                                    ·····  ⑨
        g_paint(&g_context, x, y, paint_color);                            ·····  ⑩
    }

end:
    return ret;                                                            ·····  ⑪
}
```

① タイトルを表示します。

② 塗りつぶしを開始する前に 1 秒待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

③ 塗りつぶす色を取得します。
取得した色が表示済みの線の色と同じ場合は、違う色になるまで取得し直します。

④ 経過時間を得るために現在のタイマー値を取得しておきます。

⑤ LCD 上のランダムな座標の色を取得します。この座標を含む領域が塗りつぶす候補になります。

⑥ 経過時間を取得します。塗りつぶしを繰り返す時間が経過した場合は終了します。

⑦ 100 ミリ秒毎に図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑧ touch_wait()の処理時間が塗りつぶしの繰り返し時間に含まれないようにするために、
touch_wait()の前までの経過時間を減算後、現在のタイマー値を取得し直します。

⑨ ⑤で取得した色が黒以外の場合は、領域を分ける線、または既に塗りつぶした領域の座標
なので、ランダムな座標を取得するところからやり直します。

⑩ ランダムな座標を含む領域を塗りつぶします。

⑪ 終了時は、touch_wait()の返却値をそのまま返却します。

3.8.8. sample_circle

sample_circle は、graphic_sample.c で定義しています。

【機能】
円を表示します。円の表示中にタイトル部分を選択すると図形選択画面が表示され、表示させたい図形を選択することができます。

【C 言語形式】
static int sample_circle(void);

【パラメータ】
なし

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
sample_circle(void)
{
    int             ret;
    int             x;
    int             y;
    int             r;
    int             num;
    int             i;
    unsigned short  color;
```

```
    /* Initialize */
    clear_framebuffer();                                    ..... ①

    /* Draw circles */
    display_title("Draw circles");                          ..... ②

    /* Wait */
    ret = touch_wait(1000);                                 ..... ③
    if (ret > 0) {
        goto end;
    }

    /* Draw circles */
    color = get_random_color();                             ..... ④
    for (i = 50; i > 0; i--) {                              ..... ⑤
        r = RANDOM(76);
        r += 25;                                            ..... ⑥
        num = LCD_WIDTH - (r * 2);
        x = RANDOM(num);
        x += r;
        num = LCD_HEIGHT - TITLE_HEIGHT - (r * 2);
        y = RANDOM(num);
        y += r;
        y += TITLE_HEIGHT;                                  ..... ⑦
        g_circle(&g_context, x, y, r, color);               ..... ⑧
    }

    /* Wait */
    ret = touch_wait(1000);                                 ..... ⑨
    if (ret > 0) {
        goto end;
    }

    /* Paint area */
    ret = sample_paint(color, 5000);                        ..... ⑩

end:
    return ret;                                             ..... ⑪
}
```

① 最初に画面をクリアします。

② タイトルを表示します。

③ 円を表示する前に 1 秒待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

④ 表示する円の色を取得します。

⑤ 円は 50 個表示します。

⑥ 表示する円の半径は、25～100 のランダムな値にします。

⑦ 円の中心点の座標は、円が LCD 内に表示できる範囲でランダムな値にします。

⑧ 求めた座標，半径の円を表示します。

⑨ 塗りつぶしを開始する前に 1 秒待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑩ 表示した円どうしで囲まれた領域をランダムに 5 秒間塗りつぶし続けます。

⑪ 終了時は、touch_wait()または sample_paint()の返却値をそのまま返却します。

3.8.9. sample_rectangle

sample_rectangle は、graphic_sample.c で定義しています。

【機能】
長方形を表示します。長方形の表示中にタイトル部分を選択すると図形選択画面が表示され、表示させたい図形を選択することができます。

【C 言語形式】
static int sample_rectangle(void);

【パラメータ】
なし

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
sample_rectangle(void)
{
    int             ret;
    int             x;
    int             y;
    int             w;
    int             h;
    int             num;
    int             i;
    unsigned short  color;

    /* Initialize */
    clear_framebuffer();                                    ·····  ①

    /* Title */
    display_title("Draw rectangles");                       ·····  ②

    /* Wait */
    ret = touch_wait(1000);                                 ·····  ③
    if (ret > 0) {
        goto end;
    }
```

```
    /* Draw rectangles */
    color = get_random_color();                                    ④
    for (i = 50; i > 0; i--) {                                     ⑤
        w = RANDOM(151);
        w += 50;
        h = RANDOM(151);
        h += 50;                                                   ⑥
        num = LCD_WIDTH - w;
        x = RANDOM(num);
        num = LCD_HEIGHT - TITLE_HEIGHT - h;
        y = RANDOM(num);
        y += TITLE_HEIGHT;                                         ⑦
        g_rectangle(&g_context, x, y, w, h, color);                ⑧
    }

    /* Wait */
    ret = touch_wait(1000);                                        ⑨
    if (ret > 0) {
        goto end;
    }

    /* Paint area */
    ret = sample_paint(color, 5000);                               ⑩

end:
    return ret;                                                    ⑪
}
```

① 最初に画面をクリアします。

② タイトルを表示します。

③ 長方形を表示する前に 1 秒待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

④ 表示する長方形の色を取得します。

⑤ 長方形は 50 個表示します。

⑥ 表示する長方形の幅，高さは、50～100 のランダムな値にします。

⑦ 長方形の座標は、長方形が LCD 内に表示できる範囲でランダムな値にします。

⑧ 求めた座標，大きさの長方形を表示します。

⑨ 塗りつぶしを開始する前に 1 秒待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑩ 表示した長方形どうしで囲まれた領域をランダムに 5 秒間塗りつぶし続けます。

⑪ 終了時は、touch_wait()または sample_paint()の返却値をそのまま返却します。

3.8.10. sample_pixmap

sample_pixmap は、graphic_sample.c で定義しています。

【機能】

画像を表示します。画像の表示中にタイトル部分を選択すると図形選択画面が表示され、表示させたい図形を選択することができます。

【C 言語形式】

static int sample_pixmap(void);

【パラメータ】

なし

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
sample_pixmap(void)
{
    int             ret;

    /* Initialize */
    clear_framebuffer();                                                    ·····  ①

    /* Title */
    display_title("Draw pixmaps");                                          ·····  ②

    /* Background */
    g_rectanglef(&g_context, 0, TITLE_HEIGHT,
        LCD_WIDTH, LCD_HEIGHT - TITLE_HEIGHT, COLOR_SILVER);                ·····  ③

    /* Draw pixmaps */
    g_pixmap(&g_context, 10, 24,
        pixmap_sunset.width, pixmap_sunset.height, &pixmap_sunset);         ·····  ④
    ret = touch_wait(1000);                                                 ·····  ⑤
    if (ret > 0) {
        goto end;
    }
    g_pixmap(&g_context, 30, 41,
        pixmap_cat.width, pixmap_cat.height, &pixmap_cat);                  ·····  ⑥
    ret = touch_wait(1000);                                                 ·····  ⑦
    if (ret > 0) {
        goto end;
    }
    g_pixmap(&g_context, 50, 58,
        pixmap_flower.width, pixmap_flower.height, &pixmap_flower);         ·····  ⑧
    ret = touch_wait(1000);                                                 ·····  ⑨
    if (ret > 0) {
        goto end;
    }
    g_pixmap(&g_context, 70, 75,
        pixmap_mountain.width, pixmap_mountain.height, &pixmap_mountain);   ·····  ⑩
    ret = touch_wait(3000);                                                 ·····  ⑪
```

```
end:
    return ret;                                                    ・・・・・・・・・・・・  ⑫
}
```

① 最初に画面をクリアします。

② タイトルを表示します。

③ 背景の銀色を表示します。

④ 4 つ表示する画像のうちの 1 つ目を表示します。

⑤ 1 秒間待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑥ 4 つ表示する画像のうちの 2 つ目を表示します。

⑦ 1 秒間待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑧ 4 つ表示する画像のうちの 3 つ目を表示します。

⑨ 1 秒間待ちます。この間、図形選択を受け付けます。
touch_wait()の返却値が 0 より大きい場合は、図形が選択されたので終了します。

⑩ 4 つ表示する画像のうちの 4 つ目を表示します。

⑪ 3 秒間待ちます。この間、図形選択を受け付けます。

⑫ 終了時は、touch_wait()の返却値をそのまま返却します。

3.8.11. select_function_init

select_function_init は、gui_sample.c で定義しています。

【機能】
図形選択画面の表示を受け付けるコマンド・ボタンを作成します。

【C 言語形式】
int select_function_init(void);

【パラメータ】
なし

【返却値】

値	内容
0	正常終了しました。
-1	パラメータエラーです。
-2	フォーム関連のエラーです。
-4	プロパティのエラーです。

【定義】

```
int
select_function_init(void)
{
    int             ret;

    ret = gui_create_form(&g_context, &title_form, &title_form_p);          ·····  ①
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end1;
    }
    ret = gui_commandb(&title_form, &title_commandb, &title_commandb_p);    ·····  ②
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end1;
    }
    ret = gui_draw_form(&title_form);                                       ·····  ③
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end1;
    }

    clear_touch();                                                          ·····  ④

end1:
    return ret;                                                             ·····  ⑤
}
```

① 図形選択画面の表示を受け付けるコマンド・ボタンのフォームを作成します。
gui_create_form()の返却値がエラーの場合は終了します。

② 図形選択画面の表示を受け付けるコマンド・ボタンを作成します。
gui_commandb()の返却値がエラーの場合は終了します。

③ 図形選択画面の表示を受け付けるコマンド・ボタンを表示します。
gui_draw_form()の返却値がエラーの場合は終了します。

④ タッチスクリーンの情報をクリアします。

⑤ 終了時は、API の返却値をそのまま返却します。

3.8.12. select_function

select_function は、gui_sample.c で定義しています。

【機能】

図形選択画面の表示，動作を行います。

【C 言語形式】

static int select_function(void);

【パラメータ】

なし

【返却値】

値	内容
0	図形が選択されませんでした。
1 以上	選択された図形の画面番号です。

【定義】

```
static int
select_function(void)
{
    gui_object      *touch_obj;
    int             ret;
    int             i;

    g_number = 0;                                                                   ·····  ①
    save_pixel(menu_label_p.x, menu_label_p.y, menu_form_p.width,
        menu_label_p.height + menu_form_p.height - menu_form_p.border_width);       ·····  ②

    ret = gui_create_form(&g_context, &menu_form, &menu_form_p);                    ·····  ③
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end3;
    }
    ret = gui_label(&menu_form, &menu_label, &menu_label_p);                        ·····  ④
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end2;
    }
    ret = gui_listb(&menu_form, &menu_listb, &menu_listb_p);                        ·····  ⑤
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end2;
    }
    for (i = 0; i < 2; i++) {
        ret = gui_commandb(&menu_form, &menu_commandb[i], &menu_commandb_p[i]);     ·····  ⑥
        if (ret != GRAPHIC_ERROR_NORMAL) {
            goto end2;
        }
    }
    ret = gui_draw_form(&menu_form);                                                ·····  ⑦
    if (ret != GRAPHIC_ERROR_NORMAL) {
        goto end1;
    }

    msleep(200);                                                                    ·····  ⑧
    clear_touch();                                                                  ·····  ⑨
    g_number = -999;
    do {
        touch_obj = gui_touch_check();                                              ·····  ⑩
    } while (g_number == -999);
    if (g_number > 0) {
        msleep(200);
        clear_touch();
        goto end2;                                                                  ·····  ⑪
    }
```

```
end1:
    load_pixel(menu_label_p.x, menu_label_p.y, menu_form_p.width,
        menu_label_p.height + menu_form_p.height - menu_form_p.border_width);  ·····  ⑫

end2:
    gui_delete_form(&menu_form);                                               ·····  ⑬

end3:
    return g_number;                                                           ·····  ⑭
}
```

① 選択された図形の画面番号の情報をクリアします。

② 図形選択画面が表示される部分の現在の表示を保存します。

③ 図形選択画面のフォームを作成します。
gui_create_form()の返却値がエラーの場合は終了します。

④ 図形選択画面のタイトル部分のラベルを作成します。
gui_label()の返却値がエラーの場合は終了します。

⑤ 図形選択画面のリスト・ボックスを作成します。
gui_listb()の返却値がエラーの場合は終了します。

⑥ 図形選択画面の OK ボタン，CANCEL ボタンを作成します。
gui_commandb()の返却値がエラーの場合は終了します。

⑦ 図形選択画面を表示します。
gui_draw_form()の返却値がエラーの場合は終了します。

⑧ 動作タイミングの調整のために少し待ちます。

⑨ タッチスクリーンの情報をクリアします。

⑩ 選択された図形の画面番号が、繰り返しの直前に設定した-999 でなくなるまでタッチスクリーンのチェックを行います。選択された図形の画面番号が変更されるのは、OK ボタンまたは CANCEL ボタンが選択されたときです。

⑪ 図形が選択された場合は、図形選択画面の表示前の状態に戻る必要がないため、表示の復元をせずに終了します。

⑫ 表示を図形選択画面の表示前の状態に戻すため、②で保存しておいた表示を復元します。

⑬ 図形選択画面を削除します。

⑭ 終了時は、選択された図形の画面番号を返却します。処理の途中でエラーになった場合や

図形選択画面で CANCEL ボタンが選択された場合は、選択された図形の画面番号の情報は 0 になります。

3.8.13. ok_cb

ok_cb は、gui_sample.c で定義しています。

【機能】

図形選択画面の OK ボタンを選択したときの処理を行います。

【C 言語形式】

static void ok_cb(void *arg);

【パラメータ】

I/O	パラメータ	説明
I	void *arg	OK ボタンのプロパティの argument メンバに設定されている値が渡されますが、この関数では使用しません。

【返却値】

なし

【定義】

```
static void
ok_cb(void *arg)
{
    int     idx;

    idx = gui_get_value(&menu_listb);                ..... ①
    if (idx >= 0 && idx < MENU_ITEM_COUNT) {
        g_number = idx + 1;                          ..... ②
    }
}
```

① 図形選択画面のリスト・ボックスで選択されている項目のインデックスを取得します。
項目が選択されていない場合は、255 を取得します。

② 図形選択画面のリスト・ボックスで項目が選択されている場合は、選択された図形の画面番号の情報をリスト･ボックスで選択された項目に対応する図形の画面番号に変更します。
リスト・ボックスの項目のインデックスに 1 を加算した値が、その項目に対応した図形の画面番号になります。
リスト･ボックスで項目が選択されていない場合は､情報が変更されないことになります。

3.8.14. cancel_cb

cancel_cb は、gui_sample.c で定義しています。

【機能】

図形選択画面の CANCEL ボタンを選択したときの処理を行います。

【C 言語形式】

static void cancel_cb(void *arg);

【パラメータ】

I/O	パラメータ	説明
I	void *arg	CANCEL ボタンのプロパティの argument メンバに設定されている値が渡されますが、この関数では使用しません。

【返却値】

なし

【定義】

```
static void
cancel_cb(void *arg)
{
    g_number = 0;    ·····  ①
}
```

① 選択された図形の画面番号の情報を 0 に変更します。
画面番号 0 は、図形が選択されていないことを示します。

3.8.15. touch_wait

touch_wait は、gui_sample.c で定義しています。

【機能】

指定された時間だけ、タッチスクリーンの入力を受け付けます。

【C 言語形式】

int touch_wait(unsigned int msec);

【パラメータ】

I/O	パラメータ	説明
I	unsigned int msec	待ち時間です。ここで指定された時間だけ、タッチ

		スクリーンの入力を受け付けます。

【返却値】

値	内容
0	図形が選択されませんでした。
1以上	選択された図形の画面番号です。

【定義】

```
int
touch_wait(unsigned int msec)
{
    unsigned int    timer;
    unsigned int    elapsed;
    gui_object      *touch_obj;

    timer = get_timer();                        ..........  ①
    do {
        touch_obj = gui_touch_check();          ..........  ②
        elapsed = elapsed_systime(timer);       ..........  ③
        if (touch_obj == &title_commandb) {
            select_function();                  ..........  ④
            if (g_number > 0) {
                return g_number;
            }
            if (elapsed > 0) {
                if (elapsed < msec) {           ..........  ⑤
                    msec -= elapsed;
                }
                else {
                    msec = 0;
                }
                elapsed = 0;
                timer = get_timer();
            }
        }
    } while (msec > elapsed);                   ..........  ⑥

    return 0;                                   ..........  ⑦
}
```

① 経過時間を得るために現在のタイマー値を取得しておきます。

② タッチスクリーンのチェックを行います。

③ 経過時間を取得します。

④ タッチスクリーンのチェックで図形選択画面の表示を受け付けるコマンド・ボタンが選択された場合は、図形選択画面の表示，動作を行います。図形が選択された場合は、選択された図形の画面番号を返却して終了します。

⑤ 図形選択の処理時間が待ち時間に含まれないようにするために、select_function()の前までの経過時間を減算後、現在のタイマー値を取得し直します。

⑥ 指定された時間が経過するまで繰り返します。

⑦ 図形が選択されずに指定された時間が経過した場合は、0 を返却します。