グラフィックライブラリ RX230_LCD の使い方 Ver1.0 20250613

■概要

描画ライブラリはルネサスエレクトロニクス社のマイコンRX230と表示器2.8in ch 240×320ドットフルカラーTFT液晶※1で動作するグラフィック、タッチス イッチライブラリです。SPI接続 クロック6.5MHz※2 開発環境CS+ fo r CC です。

※1 MSP2807 (秋月電子殿)、KKHMF (アマゾン)等のコントロールIC ILI9341を使った液晶を想定。

※2 ILI9341の最高動作クロックは10MHzです。

目次-----

■動作環境

セットでご購入時の梱包物

- ●ハードウエア マイコン、液晶板製品外観
- ●寸法図

●回路図概要

- ●ソフトウエア CDの詳細
- ●CS+ for CCの設定
- ●ライブラリの使い方
- ●フォントデータの使い方、組込みマイコンの考え方
- ●フォントデータをRAMに配置するかROMに配置するか

■ライブラリ関数

- ■動作
- ●デモンストレーション1 RXD230_LCD_test1
- 画面ベタ塗り、図形表示、漢字表示、メーター表示、レベルメーター表示、タッチスイッ チで表示切替
- ●デモンストレーション2 RXD230 LCD AD1
- ADコンバータ値を数値、レベルメーター、縦棒グラフ、メーターで表示
- ●デモンストレーション3 RXD230 LCD AD2
- ADコンバータ値を数値、レベルメーター、メーターで表示

文字、背景の色を独立して16色ずつ設定できる関数、

●デモンストレーション4 RXD230_LCD_安全

安全、注意、危険の表示とタッチスイッチによる表示切替。安全、注意、危険を漢字や色 で表示でき、より人間に理解でき易い表示になります。

■備考

- ●CS+ for CC 初めからプログラムを作る設定
- ●フォントデータの作り方

●セットでご購入時の梱包物

ハード、ソフトのセットでご購入いただいた場合の箱の梱包物です。

グラフィックライブラリの使い方(このマニュアルです) RX230_240 × 320LCD ボードハードマニュアル CD(グラフィックライブラリ、サンプルソフト、 ドキュメント、フォントを作るソフト等) 2 RX230_240 × 320_LCD V1.0 ガラフィックライブラリ RX230 LCD の使い方 20280813 5 ●単語 構築・グラリロルキャスエレクトロニクス社のマイコン内と2002年8月8日 8 in ch 240×320ドッドフルカラードで可通知に定動作をグラフィック、ラッテス イッテライブラリてす、5 PI開展 クロック4.5MH1回2 開発現在34 io r CC で下、 5 P ※1 MSP2807 (数月電子数)、K (CHMF (アマゾン)等のコントロール10 1 し19341を使った読品を問言。 d L R0230_240×320L0D ポード 取扱設明書 N版 N255 00 13 有能会社 ビーリバーエレクトロニクス t t C I C I ●概要 ルネサスエレクトロニクス社のマイコンR スキャスエンクトロニタス社のマイコンR220 と混画集や着こ Binch 240×320ドクトフルカラーT FT基単語 とを理想にしたポートです。特別のグランインの、タッアスイッテライブラリて動作します。BFI 種類 クロック6 SANH-32 民間構成なド Fior CO です。 部1 MSP2807 (批賞単和)、KKHMF (アマゾン)等のコントロールIC 1L193412社プン社 1 1 ※11 MSF2807 (00) NTRU 、 (10) NTRU 、 10) ALTER 高を想定。 ※2 1し19341の最高動作クロックは10MHzです。 ●CPU時後 R5F52306ADFM R2230 まだぐうちれるのシフア紙 あた時間に数 Mee MA M 回答 CB 5ME CB ぐううMEC, B どう Materian Ser FM BEL だくう ARRESTING AD CDFM にあめ、 6月間 GB 20 のうでだがっ、 ARRENGA 5 M (プラインの DC /m CF/T-4757%) REServed ニュードをは Materian & インケングバップ回知者 ノゼソフ ALAF MPD HE 07751 4122 1870-1980 (1999-1299) 120 (1999) 120 (1997) 758782 (1997) 120 (1997) 1 ビン 10×10mm 0.5mmビッチ 10 1

液晶板+RX230CPU ボード

ド タッチキー用ペン デバック用25mmスペーサー E2lite やコネクタからブレットボード への線がストレスなく接続できます。

本製品開発には E1, E21 ite 等のエミュレータが必要です。

サンプルソフトは E2lite 用に書かれていますが、容易に E1 でも動作するように変更可能 です。お問い合わせください。

●ハードウエア マイコン、液晶板製品外観

表面は 2.8 inch 液晶が見えます。 裏に専用の RX230 マイコン基板が搭載されています。





CPU ボードのコネクタはメスコネクタが実装済みで、ブレッドボードに簡単に接続できます。 デバック時は E2lite からの 3.3V で液晶を駆動出来ます。デバック終了時は電源コネクタ に添付ケーブルで 3.3V を印加し使用してください。

●回路図概要

マイコン、液晶は+3.3V動作です。マイコンは5Vでも動作可能ですが、液晶が不可な ので、3.3Vで使用します。マイコンと液晶の接続はSPI接続です。CN22番R XD5がSDOに接続され、1番TXD5がSDIに接続されています。SCKは6.5 MHzのクロックです(最大10MHz)。液晶側CN6のT_xxがつくポートはタッチ スイッチのインターフェイス信号です。14番 T_IRQはタッチスイッチが押される とLow=0Vに下がります。T_DOからのデータで押された位置のX,Yポジション を認識します。



●ソフトウエア CDの詳細

CDには以下のホルダがあります。

| ドキュメント > RX230_240×3 | 320液晶制御 > RX230_240_320LCD_CD出 | 出荷用データ ≫ | |
|----------------------|--------------------------------|----------|------------------|
| ^ | 名前 | 状態 | 更新日時 |
| | Fonts_20250606 | \odot | 2025/06/10 17:11 |
| × | 🔄 サンプルソフト | \odot | 2025/06/10 17:10 |
| 1 | ドキュメント | \odot | 2025/06/10 17:01 |
| 1 | フォントを作る | 0 | 2025/06/10 16:53 |
| # | | \odot | 2025/06/10 17:23 |
| | | | |

Fontsはライブラリで使用出来る文字や図形データ集で、詳しい使い方は後述しますが、使いたいフォントをエディタで br_fonts. dat ファイルに移して使います。

| 名前 | 状態 | 更新日時 |
|------------------|----|------------------|
| アルファベット小文字 16×16 | C | 2025/06/12 14:22 |
| アルファベット大文字 16×16 | Ø | 2025/06/12 14:22 |
| アルファベット大文字 32×32 | Ø | 2025/06/12 14:22 |
| ひらがな 16×16 | C | 2025/06/12 14:22 |
| 🔜 フォントを作る | C | 2025/06/12 14:22 |
| | 2 | 2025/06/12 14:22 |
| メーター | 2 | 2025/06/12 14:22 |
| | C | 2025/06/12 14:22 |
| 英字_危険注意安全 | Q | 2025/06/12 14:22 |
| 漢字出現1-51 | 2 | 2025/06/12 14:22 |
| 漢字出現52-102 | 2 | 2025/06/12 14:22 |
| 数字16×16 | 2 | 2025/06/12 14:22 |
| 数字32×32 | C | 2025/06/12 14:22 |

サンプルソフトはCS+ for CC環境で動作し、動きはyoutubeでも確認できます。これをひな形として独自のプログラムを開発することが出来ます。

| C D 出荷用データ » RX230_240_320l | LCD_CD出荷用デ−タ > | サンプルソフト > |
|-----------------------------|----------------|------------------|
| 名前 | 状態 | 更新日時 |
| RX230_LCD_AD1 | 0 | 2025/06/11 17:31 |
| RX230_LCD_AD2 | \odot | 2025/06/11 17:31 |
| RX230_LCD_test1 | \odot | 2025/06/11 17:31 |
| RX230_LCD_安全 | 0 | 2025/06/11 17:31 |

ドキュメントは回路図、ハードの取扱説明書、ソフト グラフィックライブラリの使い方 があります。

| 5用データ → | ドキュメント |
|---------|------------------------------|
| 状態 | 更新日時 |
| 0 | 2025/06/10 16:59 |
| 0 | 2025/06/10 17:01 |
| 0 | 2025/06/10 16:55 |
| | 5用データ > 状態 ◎ ◎ ◎ |

フォントを作るは 16 × 16 のフォントを作る | fms16. exe と 32 × 32 のフォントを作る | fms32. exe の 2 つと、いくつかのフォントファイルが入っています。

C D 出荷用データ > RX230_240_320LCD_CD出荷用データ > フォントを作る

| 名前 | 状態 | 更新日時 |
|-----------------------|---------|------------------|
| Kanal Ifms32.exe | 0 | 2013/02/26 15:57 |
| 🍂 lfms16.exe | 0 | 2013/02/26 15:54 |
| 力.L32 | 0 | 2013/02/21 14:44 |
| ☐ ⊞.L32 | \odot | 2013/02/20 11:57 |
| 前.L32 | 0 | 2013/02/18 17:42 |
| 出.L32 | 0 | 2013/02/21 14:46 |
| 後.L32 | \odot | 2013/02/18 17:43 |
| yellow_panel.L32 | \odot | 2013/02/21 14:15 |
| 🗋 test_color_full.L32 | \odot | 2013/02/20 18:54 |
| test_color.L32 | \odot | 2013/02/20 11:31 |

ライブラリはグラフィックライブラリの本体で、新規にプログラムを作る場合、ここから コピーしてください。

C D 出荷用データ → RX230_240_320LCD_CD出荷用データ → ライブラリ_20250606

| 名前 | 状態 | 更新日時 |
|--------------|----|------------------|
| br_fonts.dat | 0 | 2025/05/30 16:22 |
| 230_lcd.c | 0 | 2025/06/06 14:34 |
| 230_lcd.h | 0 | 2025/06/06 14:34 |

●CS+ for CCの設定

液晶を駆動するにあたり、ポートの設定やタイマー割り込みを設定する必要があります。

新規にプログラムを作る場合の開発環境CS+ for CCのプロジェクト設定の仕 方は、簡易版と初めから設定する方法の2つがあります。初めからの設定は項目が多数あ り、ハードのことなので、特に液晶プログラムを開発、動作させるにあたり、必要な知識 とも言い難く、後の方に書いておきますので、必要な時に参考にしてみてください。ここ ではコピペで行う簡易版の方法を示します。

1. 既に動作しているホルダをコピーします。以下例ではRX230_LCD_test というホルダをコピーしています。

| - L | | | | |
|------------------------|---|------------------|------------|--|
| RX2301_LCD_test1 - コピー | C | 2025/05/16 14:24 | ファイル フォルダー | |
| RX2301_LCD_test1 | 0 | 2025/05/15 15:12 | ファイル フォルダー | |

2. ホルダの名前を付けます。例ではRX230_LCD_AD1に変更。

| RX2301_LCD_AD1 | S | 2025/05/16 14:24 | ファイル フォルダー | |
|------------------|---------|------------------|------------|--|
| RX2301_LCD_test1 | \odot | 2025/05/15 15:12 | ファイル フォルダー | |

3. ホルダの中の拡張子mtpjファイルの名前が前のままなので直します。

| mx230_lcd.c | \odot | 2025/05/15 15:12 | CJRTH | 21 KB |
|--------------------|---------|------------------|-----------|----------|
| 📸 rx230_lcd.h | \odot | 2025/05/14 10:35 | H ファイル | 3 KB |
| RX230_LCD_AD1.mtpj | \odot | 2025/05/02 17:57 | MTPJ ファイル | 8,807 KB |

以上で新しいプロジェクトが完成しました。ハード的な設定は継承され、ユーザーは液晶 表示を制御するプログラムを新たなプロジェクトで書くことが出来ます。

●ライブラリの使い方

ライブラリは br_fonts.dat フォントデータファイル rx230_lcd.h ライブラリ本体 ヘッダファイル rx230_lcd.c ライブラリ本体 で構成されます。新規作成の場合、下記例のように /* Start、、*/ /* End、、*/で挟 むように書いて下さい。 /* Start user code for include. Do not edit comment generated here */

#include "br_fonts.dat"
#include "rx230_lcd.h"
#include "rx230_lcd.c"

/* End user code. Do not edit comment generated here */



init__Icd() 液晶の初期化と

LCD_BRITE_ON; //LCD バックライトON 明るい 昼用 //LCD_DARK_ON; //LCD バックライトON ダーク 夜用

LCDバックライトの明るさを決めて関数を実行することにより液晶画面に文字、漢字、 図形、グラフ等々の表示を開始することが出来ます。

●フォントデータの使い方、組込みマイコンの考え方

フォントデータはbr_fonts.datの中にあります。

冒頭、コメントにありますが、表示できる色はO~fまで、16色、例えば黒の場合、O、 白は3で表しています。

/*

液晶制御プログラム 2013.2.13

フォント 16*16 32*32

| cIBIack | 0x000000 | //0 | 黒 |
|-----------|--|---|---|
| clGray | 0x808080 | //1 | グレー |
| clSilver | 0xc0c0c0 | //2 | 銀 |
| clWhite | Oxffffff | //3 | 白 |
| clFuchsia | 0xff00ff | //4 | ピンク |
| clPurple | 0x800080 | //5 | 紫 |
| clMaroon | 0x800000 | //6 | 栗 |
| clRed | 0xff0000 | //7 | 赤 |
| clYellow | 0xffff00 | //8 | 黄色 |
| clLime | 0x00ff00 | //9 | ライム |
| clGreen | 0x008000 | //a | 緑 |
| clTeal | 0x008080 | //b | 青緑 |
| clBlue | 0x0000ff | //c | 青 |
| clNavy | 0x000080 | //d | 紺 |
| clAqua | 0x00ffff | //e | アクア |
| clOlive | 0x808000 | //f | オリーブ |
| | clBlack clGray clSilver clWhite clFuchsia clPurple clMaroon clRed clYellow clLime clGreen clTeal clBlue clNavy clAqua clOlive | clBlack 0x000000 clGray 0x808080 clSilver 0xc0c0c0 clWhite 0xfffff clFuchsia 0xff00ff clPurple 0x800000 clRed 0xff0000 clYellow 0xffff00 clGreen 0x008000 clTeal 0x008000 clBlue 0x00000 clAqua 0x00000 clAqua 0x00ff00 | clBlack0x000000//0clGray0x808080//1clSilver0xc0c0c0//2clWhite0xffffff//3clFuchsia0xff00ff//4clPurple0x800080//5clMaroon0x800000//6clRed0xff0000//7clYellow0xffff00//8clLime0x00000//9clGreen0x008000//aclTeal0x008080//bclBlue0x0000ff//cclNavy0x000080//dclAqua0x00ffff//eclOlive0x808000//f |

2025.04.21 const ROM に配置 無し RAM に配置

*/

例えば数字のOは以下のような形で unsigned char データになっています。背景は3なので白、文字は0で黒と表現されます。

//数字0

 データは16×16、32×32ドットの2種類あります。

組込みマイコンでのフォントデータの考え方は、必要なデータをbr_fonts.da tファイルに書き加える、不要なものは削除する形になります。理由はフォントデータが 例えばJIS漢字第一、二水準のデータ容量合計は220Kバイトと言われます。本件の ハードウエアに使用されているマイコン R5F2306ADFM(RX230)でR0 M容量は256Kバイトですので、漢字データだけでROMを使ってしまう可能性があり ます。外部に漢字ROMを付けることも考えられますが、コスト高になります。

そこで、このライブラリではデータはある程度、弊社が用意しますので、その取捨選択は ユーザーにお任せする形になります。具体例は後述のサンプルプログラムによるデモンス トレーションを参照ください。

例えば漢字を追加する場合、ライブラリとは別に漢字_No1、No2というデータがあります。

| J御 → Fonts → 漢字_No1 → | | |
|-----------------------|-------------|---------|
| 名前 | ^ | 状態 |
| No1_16bit | | \odot |
| No1_32bit | | 0 |
| 🗐 漢字出現頻度表 | 順位対照表1.docx | 0 |

弊社の漢字フォントは漢字出現頻度表に従って作られています。出現頻度表とは統計的に 得られた世の中で使われる漢字の順位です。1つのナンバーに31個の16×16漢字フォン ト、32×32漢字フォントが収納されています。

| 凸版 (3) | 漢字 | 種類 | 備考 |
|-------------|----|----|----|
| 利 利 1 | Å | 常用 | |
| 2 | → | 常用 | |
| 3 | 日 | 常用 | |
| 4 | 大 | 常用 | |
| 5 | 年 | 常用 | |
| 6 | 出 | 常用 | |
| 7 | 本 | 常用 | |
| 8 | 中 | 常用 | |
| 9 | 子 | 常用 | |
| 10 | 見 | 常用 | |
| 11 | 国 | 常用 | |
| 12 | 言 | 常用 | |
| 13 | E | 常用 | |
| 14 | 分 | 常用 | |
| 15 | 生 | 常用 | |
| 16 | 手 | 常用 | |

晶制御 > Fonts > 漢字_No1 > No1_16bit



例えば 漢字出現頻度表1位は「人」という字ですが、No1_16bit の中に人.C16 と人.L16 と2つのファイルが収められています。人. C16がエディタで編集して実際に使うファ イルで、人.L16は後述するフォントの作り方に出てくるフォント作成ソフトが使うデー タ型式です。 人. C16をエディタで開けると16×16ドットのデータが出てきます。

unsigned char 人[]={ 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3,

これをコピーし、br_fonts. datファイルに加えて下さい。注意点は2つあり ます。

1. unsigned char 人[]={ 人はそのままではエラーになるので、アルファベットや数字 に変えて下さい。例 unsigned char hito[]={

2.

データをRAMに配置する場合はそのまま、ROMに配置する場合はconstを頭に付けて下さい。const unsigned char hito[]={

RX230はRAM容量は32KBとROM 256KBの1/8しかないので、フォン トをbr_fonts_datに登録していくとROMよりも早くオーバーフローするこ とになります。詳細は次項をご参照ください。

●フォントデータをRAMに配置するかROMに配置するか

フォントデータは表示したい内容が増えてくると、どんどん増加し、時にRAMやROM の領域を超えてしまうことがあります。組込みではこの管理が重要になります。

フォントデータの頭に const を書かないとデータはRAM上に展開され、書くとROMに 展開されます。以下例ではROM上に配置されます。

//数字 0

const unsigned char S0[]={

内容を調べるには表示→メモリマッピングプロファイラを開きます。

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウインドウ(W) ヘルプ(ト 💋 ソリューション一覧(S) スタート(S) | 「
 同 プロジェクト・ツリー(J) • 1 S S & S & S プロパティ(P) /ヨン一覧(S) 💷 プロジェクト・ツリー 🁒 スマート・ブラウザー(R) センブル1 🗹 r_ce_mainc 🗹 r_ce コード生成プレビュー(V) N. 2 0 8 ルされたメモリ・マッピング一覧 ì レシア P 端子配置(I) アイル(収集) 🔜 🗙 R5F 🐺 スマート・マニュアル(L) トマニュアル のプロファイル結果>(未収集) 표... 역 그-나 🔜 出力(0) - 🕀 プロ 🔢 メモリ・マッピング・プロファイラ(F) いるメモリ・マッピングの詳細 - 🗊 771 📷 関数変数アクセス表(V) - h_ i 「「「 デバッグ・フォージンタノワ)

プロファイル収集にして、コンパイルするとデータが収集されます。

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウインドウ(W) ヘルン -- 💎 🖓 🐙 🧖 🔚 🔲 💭 🖳 💭 🗗 💋 ソリューション一覧(S) 💷 プロジェクト・ツリー **μ** Χ 🌇 逆アセンブル1 🧹 r_cg_main.c 🧃 2 🕜 🤮 🗃 プロファイルされたメモリ・マッピング一覧 アター □ RX230 LCD test1 (プロジェクト)* 🚰 プロファイル(収集) 🛛 🔜 🗙 7 R5F52306AxFM (マイクロコントローラ) < 最新のプロファイル結果> マニュアル ☆ 雪 コード生成 (設計ツール) 🔨 CC-RX (ビルド・ツール) ----- 分 プログラム解析 (解析ツール)

サマリで現在のROM、RAMの使用状況が%で表示されます。残り%が少なくなり、ス タック領域などが維持できなくなると正常動作しなくなりますので注意が必要です。



シンボルで例えば数字の0(S0 16×16ドットデータ)がサイズ256バイトでR OM領域に配置されていることが確認できます。

| _rd | 4 | Data | 0×00000D38 | 0×00000D3B | Data | Global | R | DefaultBuild¥r_cg_m |
|-------------------|-------|----------|------------|------------|-------|--------|-----------|---------------------|
| brk | 4 | Data | 0×0000D3C | 0×00000D3F | Data | Local | R | DefaultBuild¥r_cg_s |
| _sinf_coeff | 16 | Data | 0×00000D40 | 0×00000D4F | Data | Local | R | sincosfsub |
| _cosf_coeff | 16 | Data | 0×00000D50 | 0×00000D5F | Data | Local | R | sincosfsub |
| _pi4ptr_table | 12 | Data | 0×00000D60 | 0×00000D6B | Data | Local | R | sincosfsub |
| _sub_table | 16 | Data | 0×00000D6C | 0×00000D7B | Data | Local | R | sincosfsub |
| _PowerON_Reset_PC | 62 | Program | 0×FFFC0000 | 0×FFFC003D | Entry | Global | PResetPRG | DefaultBuild¥r_cg_r |
| MY | 256 | Constant | 0×FFFC0100 | 0×FFFC01FF | Data | Global | C_1 | DefaultBuild¥r_cg_r |
| SO | 256 | Constant | 0×FFFC0200 | 0×FFFC02FF | Data | Global | C_1 | DefaultBuild¥r_cg_r |
| _S1 | 256 | Constant | 0×FFFC0300 | 0×FFFC03FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _\$2 | 256 | Constant | 0×FFFC0400 | 0×FFFC04FF | Data | Global | C_1 | DefaultBuild¥r_cg_u |
| _S3 | 256 | Constant | 0×FFFC0500 | 0×FFFC05FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| 84 | 256 | Constant | 0×FFFC0600 | 0×FFFC06FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| 85 | 256 | Constant | 0×FFFC0700 | 0×FFFC07FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _\$6 | 256 | Constant | 0×FFFC0800 | 0×FFFC08FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _87 | 256 | Constant | 0×FFFC0900 | 0×FFFC09FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _\$8 | 256 | Constant | 0×FFFC0A00 | 0×FFFC0AFF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _\$9 | 256 | Constant | 0×FFFC0B00 | 0×FFFC0BFF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _Sten | 256 | Constant | 0×FFFC0C00 | 0×FFFC0CFF | Data | Global | C_1 | DefaultBuild¥r_cs_ |
| _test2 | 256 | Constant | 0×FFFC0D00 | 0×FFFC0DFF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| test3 | 256 | Constant | 0×FFFC0E00 | 0×FFFC0EFF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _inu | 256 | Constant | 0×FFFC0F00 | 0×FFFC0FFF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _kanji_ie | 256 | Constant | 0×FFFC1000 | 0×FFFC10FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _kanji_kai | 256 | Constant | 0×FFFC1100 | 0×FFFC11FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _kanji_gaku | 256 | Constant | 0×FFFC1200 | 0×FFFC12FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _kanji_aida | 256 | Constant | 0×FFFC1300 | 0×FFFC13FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _kanji_ji | 256 | Constant | 0×FFFC1400 | 0×FFFC14FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| _S0 w | 1024 | Constant | 0×FFFC1500 | 0×FFFC18FF | Data | Global | C_1 | DefaultBuild¥r_cg_ |
| \$1w | 10.24 | Constant | 0×FFFC1900 | 0×FFFC1CFF | Data | Global | C 1 | DefaultBuild¥r.cci |

■ライブラリ関数

void lwait(long time) //短時間のウエイト

//ILI9341 液晶コントローラ制御関数
void CMD_Send(uint8_t command) //ILI9341 へのコマンド書き
void Data_Receive(rdata) //ILI9341 からのデータ受信
void Data_Send(uint16_t tdata, uint16_t rdata) //ILI9341 へのデータ書き込み、受信
void Data_Send16(uint16_t tdata) //ILI9341 への 16bit data 書き込み

//画面制御

| //RX230_ | lcd. | h | | |
|----------|------|-------|----|--|
| #dofino | | RRITE | ON | |

| #define LCD_B | RITE_ON | //LCD | バックライト | 明 | オン | (昼用) |
|---------------|----------|-------|--------|---|----|------|
| #define LCD_B | RITE_OFF | //LCD | バックライト | 明 | オフ | |
| #define LCD_D | ARK_ON | //LCD | バックライト | 暗 | オン | (夜用) |
| #define LCD_D | ARK_OFF | //LCD | バックライト | 暗 | オフ | |

//RX230_l cd. c

void gbeta(uint16_t color) //ベタ塗り 240×320ドット画面を color 色で塗ります。 //720msec 程度かかります。

void pset(uint16_t xposi,uint16_t yposi,uint16_t color) //ピクセルセット //xposi0-239 yposi0-319 の位置に color 色で 1 ドット //描画します。横軸が X、縦軸が Y

uint16_t color_sel(uint8_t color_code) //カラーコード0~Fに応じて実際に書き込 //むRGB データを返します。

void gwrite16(uint16_t xposi, uint16_t yposi, uint8_t *font_data) //16×16 データ //の書き込み。文字、漢字、図などを描きます。 //*font_data はフォントデータの開始アドレス

void gwrite32(uint16_t xposi, uint16_t yposi, uint8_t *font_data) //32×32 データ //の書き込み。文字、漢字、図などを描きます。 //*font_data はフォントデータの開始アドレス

void gwrite32_96 (uint16_t xposi, uint16_t yposi, uint8_t *font_data) //32×32 データ //を 96×96(3 倍) に拡張して書き込み。文字、漢字、図 //などを描きます。 //*font_data はフォントデータの開始アドレス

void gwrite32_64(uint16_t xposi,uint16_t yposi,uint8_t *font_data)//32×32 データ //を 64×64(2 倍) に拡張して書き込み。文字、漢字、図 //などを描きます。

//*font_data はフォントデータの開始アドレス

void yline(uint16_t xposi1,uint16_t yposi1,uint16_t xposi2,uint16_t yposi2,uint16_t color) //直線 Y 縦軸 line

//Y 縦軸に高速に直線を引きます。斜め線は引けません

//xposi1=xposi2 でないといけません

//yposi2 > yposi1 でないといけません

void gline(uint16_t xposi1,uint16_t yposi1,uint16_t xposi2,uint16_t yposi2,uint16_t color) //直線 XY line

//XY 横軸に直線を引きます。斜め線は引けません
//Y 軸の線は yposi2 > yposi1 でないといけません
//X 軸の線は xposi2 > xposi1 でないといけません
//X 軸描画は pset() 関数を使用(時間がかかる)

void gbox(uint16_t xposi1, uint16_t yposi1, uint16_t xposi2, uint16_t yposi2, uint16_t line_color, uint16_t fill_color)//BOX

//BOX 四角形を描画します。
//xposi1、yposi1起点 xposi2、yposi2 終点
//line_color 線の色 fill_color box の中の色
//xposi2 > xposi1でないといけません
//yposi2 > yposi1でないといけません
//描画に 150msec かかります (サイズで異なります)

void gcircle_fill(uint16_t xposi1, uint16_t yposi1, uint16_t radis, uint16_t color) //丸を描画します。 //xposi1、yposi1起点 radis 半径 color 円の中の色 //描画に 125msec かかります(サイズで異なります)

void gcircle_outline(uint16_t xposi1, uint16_t yposi1, uint16_t radis, uint16_t color) //丸を描画します。 //xposi1、yposi1 起点 radis 半径 color 外周の色 //描画に 30msec かかります

void gcircle_outline2(uint16_t xposi1, uint16_t yposi1, uint16_t radis, uint16_t color)

//丸を描画します。外周を半径を小さくして3回書いてます。
 //(外周強調)
 //xposi1、yposi1起点 radis 半径 color 外周の色
 //描画に 100msec かかります

void gcircle(uint16_t xposi1, uint16_t yposi1, uint16_t radis, uint16_t mode, uint16_t color)

//丸を描画します。 //mode 0 gcircle_outline(); //mode 1 gcircle_fill(); //mode 2 gcircle_outline2(); //xposi1、yposi1 起点 radis 半径 color 外周の色 //描画に~100msec かかります void Dsiplay ON(void) //液晶表示 ON void Dsiplay_OFF (void) //液晶表示 OFF //液晶初期設定、1msec 定周期タイマー ON void init_lcd(void) //タッチスイッチ void T_com_write(uint8_t comm) //タッチスイッチ コマンドライト void touch_switch_read(void) //タッチスイッチ データ読み込み //TX_data に X軸データ 8bit が入ります。 //TY_data に Y軸データ 8bit が入ります。 //58 µ sec かかります。1msec 定周期割り込みで呼んで //も10%以下の消費時間。 //テスト表示 void test_kanji(void) //16×16 漢字 学間家字 を全画面に書く void test_zukei(void) { gbox (50, 50, 200, 200, clBlue, clAqua); //四角 gcircle(100, 100, 50, 2, clGreen); //mode 2 円周強調 丸 gcircle(120, 140, 25, 1, clRed); //mode 1 内部ベタ塗 丸 gcircle(220, 220, 15, 0, clBlue); //mode 0 円周描画 丸 } void test_zukei2(void) ł gwrite32_64(90, 10, test_color); //32×32 16 色データを 64×64(2 倍) に拡大して表示 gwrite32_96(140,100,test); //32×32 データを96×96(3倍)に拡大して表示 gwrite32_96(140,220,test_color); //32×32 16 色データを 96×96(3 倍) に拡大して表示 gwrite32_64(10,220,test); //32×32 データを 64×64(2倍) に拡大して表示

```
}
void test_kanji3(void)
              gwrite32_64(90,10,kanji_mae);
                     //32×32 漢字 前を64×64(2倍)に拡大して表示
              gwrite32(200,10,kanji usiro);
                     //32×32 漢字 後を表示
              gwrite32_96(140,220,kanji_syutu);
                     //32×32 漢字 出を96×96(3倍)に拡大して表示
              gwrite32_96(140,100,kanji_riki);
                     //32×32 漢字 力を96×96(3倍)に拡大して表示
              gwrite32(10,280,kanji_mae);
                                          //32×32 漢字 前を表示
              gwrite32(42,280,kanji_usiro);
                                          //32×32 漢字 後を表示
              gwrite32(10,240,kanji_syutu);
                                         //32×32 漢字 出を表示
              gwrite32(42,240,kanji_riki);
                                          //32×32 漢字 力を表示
}
void test_char (void)
       for (yloop = 0; yloop < 320; yloop+=16) //画面いっぱいに書きます。
       {
          for (x \mid oop = 0; x \mid oop < 240; x \mid oop +=64)
          {
                                         //16×16 人の顔1
              gwrite16(xloop,yloop,kao1);
              gwrite16(xloop+16,yloop,inu2); //16×16 犬の顔2
              gwrite16(xloop+32,yloop,kao2); //16×16 人の顔2
              gwrite16(xloop+48,yloop,inu3); //16×16 犬の顔3
          }
       }
}
```

//数字表示関数

//16×16、32×32 ドットの数字フォント開始アドレスをセットする void cv_suuji(uint8_t bangou, uint8_t fonts_size)

lcd_puts 関数などで、データが数字ではなく、小数点だった場合に自動的に使われる隣との表示間隔を狭め自然な表示にする関数群。32、64,96 ドット用があります。

// 点 専 用 縦 4 列 書 き 込 み void gwrite32_64_ten_color(uint16_t xposi,uint16_t yposi,uint8_t *font_data,uint16_t color1,uint16_t color2)

//点専用縦4列書き込み

void gwrite32_ten_color(uint16_t xposi, uint16_t yposi, uint8_t *font_data, uint16_t color1, uint16_t color2)

液晶に数字を表示する関数 色情報はフォントデータのまま表示される。フォントサイズ 16,32,64 ドットを選択できる。

void lcd_puts(uint16_t xposi, uint16_t yposi, uint8_t fonts_size, uint8_t *data)

液晶に数字を表示する関数 フォントデータの文字、背景の色を変えて表示できる。 void lcd_puts_color(uint16_t xposi,uint16_t yposi,uint8_t fonts_size,uint8_t *data,uint16_t color1,uint16_t color2)

//xposi、 yposi、 fontsize(16,32,64)、 *data の先頭アドレス、color1 黒データの変換 色、coloer2 白データの変換色

//各種表示コンポーネント 具体的なコンポーネンツの使い方は■動作 デモンストレー ションをご参照下さい。

回転するメーター表示 0-4095入力を0-9の10段階の位置に黄色、赤色の針で示します。 サイズは16,32,64,96ドット

//メーター黄色 0-9 まで動くメーター void meter_yellow(uint16_t xposi,uint16_t yposi,uint8_t m_size,uint16_t data)

//メーター赤色 0-9 まで動くメーター void meter_red(uint16_t xposi,uint16_t yposi,uint8_t m_size,uint16_t data)

レベルメーター 入力 0-4095 を 12 分割し、10 迄は緑、11 黄色、12 赤で表示します。サイズは 32, 64 ドット。

//レベルメーター初期化
void init_level_meter(uint16_t xposi,uint16_t yposi,uint8_t size)
//レベルメーター 入力を 12 分割し、10 迄は緑、11 黄色、12 赤で表示します。
void level_meter(uint16_t xposi,uint16_t yposi,uint8_t size,uint16_t data)

縦棒、点グラフ描画

//縦棒、点グラフ初期化 void init_Vbar_chart(uint16_t xposi,uint16_t yposi,uint8_t size)

縦棒 (X 軸)、点グラフ描画 開始、終了 X 座標指定 描画 64、96 ドット 色は選択 16 色、 入力データ 0-4095

void Vbar_chart(uint16_t xposi_start,uint16_t xposi_end,uint16_t yposi,uint8_t size,uint8_t mode,uint16_t color,uint16_t addata)

//タッチスイッチテスト タッチスイッチを押すたびに画像が変わります。 void touch_switch_test(void)

■動作 ●デモンストレーション1 RXD230_LCD_test1 画面ベタ塗り、図形表示、漢字表示、メーター表示、レベルメーター表示、タッチスイッチで表示切替 ●動作が見える youtube URL https://www.youtube.com/watch?v=0JpwdZ0Hc1E

void main(void)

{

R_MAIN_UserInit();

/* Start user code. Do not edit comment generated here */

//液晶初期化

init_lcd(); ①

//バックライト 明るい

| // | LCD_DARK_O | N; | //LCD バックライト ON | ダー | ク | 夜用 | |
|----|------------------|--------------|--------------------------------|------|------|---------|---------|
| | LCD_BRITE_O | N;2 | //LCD バックライト ON | 明る | い | 昼用 | |
| | gbeta(clWhite);(| 3 | //ベタ塗 | | | | |
| | test_kanji(); | 4 | //漢字表示 | | | | |
| | test_zukei(); | 5 | //図形表示 | | | | |
| | test_zukei2(); | 6 | //図形2表示 | | | | |
| | test_kanji3(); | \bigcirc | //漢字3表示 | | | | |
| | init_lm_flg = 0; | (8) | //レベルメーター初期化 | | | | |
| | while(1) (9) | | | | | | |
| | { | | | | | | |
| | for(le | 00p5 = 0;loc | pp5 < 10;loop5++) (10) | | | | |
| | { | • | • • • • | | | | |
| | | touch_swit | ch_test(); ① | | タ | ッチスイッチテ | スト |
| | | | | | | | |
| | | level_mete | r(10,80,fsize_32,loop5*455); | 12 | // V | ベルメーター | 140msec |
| | | meter_yell | ow(10,150,fsize_64,loop5*455); | (13) | //メ | ーター黄色 | |
| | | meter_red(| 10,10,fsize_32,loop5*455); | 14 | //メ | ーター赤色 | |
| | | if(loop5 == | = 9) | 15 | | | |
| | | { | | | | | |
| | | int_w | vait(500); | | | | |
| | | } | | | | | |
| | } | | | | | | |
| | } | | | | | | |

/* End user code. Do not edit comment generated here */

}

〇プログラム解説

//液晶初期化

init_lcd(); ①

①液晶の初期化です。関数はライブラリ RX230_lcd.cの中にあります。RX230 と液晶のやり 取りを行う SPI インターフェイスの初期化と定周期タイマー割り込み 1msec の初期化を行 っています。

//バックライト 明るい

 //
 LCD_DARK_ON;
 //LCD バックライトON
 ダーク
 夜用

 LCD_BRITE_ON;
 //LCD バックライトON
 明るい
 昼用

 ②液晶のバックライトの明るさは2段階に調整出来ます。
 区用、夜用を切り替え可能です。

gbeta(clWhite);③ //ベタ塗

③電源ON時にグラフィックRAMを白色で塗っています。

test_kanji(); ④ //漢字表示

④16×16の「学問家時」という漢字が画面いっぱいに表示されます。縦20文字、横15文字、合計300文字。

| 凈 | 問 | 灾 | 며キ | 受 | 問 | 灾 | 며북 | 学 | 問 | 家 | 미축 | 学 | 問 | 家 |
|---|----|---|-----|---|----|---|-----|---|---|---|-----|---|---|---|
| 五 | 日日 | 1 | nt. | T | 日日 | 1 | at. | 五 | | 1 | u j | 王 | | 1 |
| t | 同 | 3 | 时 | t | 旧 | 豕 | 时 | t | 日 | 涿 | вŦ | f | 目 | 豕 |
| 字 | 間 | 家 | 時 | 字 | 間 | 家 | 時 | 字 | 間 | 家 | 時 | 字 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 堂 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 堂 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 堂 | 間 | 家 | 時 | 堂 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |
| 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 | 時 | 学 | 間 | 家 |

関数の中身を見ると学問家時の漢字4文字を横(X)64ドット、縦(Y)16ドット毎に書いているのが分かります。

void test_kanji(void) //16×16 学間家字を全画面に書く

{

for(yloop = 0;yloop < 320;yloop+=16)
{
 for(xloop = 0;xloop < 240;xloop+=64)</pre>

{ gwrite16(xloop,yloop,kanji gaku); gwrite16(xloop+16,yloop,kanji_aida); gwrite16(xloop+32,yloop,kanji ie); gwrite16(xloop+48,yloop,kanji_ji); } } } test_zukei(); (5)//図形表示 Y:320 時学間 家時学 間 家時 家時学 間 家時 間 家時学 間 家時 学間 家時 学 学 家時 間 家時 家時 間 学 間 学 椙 間 家時学 学 家時 学間 間 家 間 間 家 間 問 家 間 問 家 間 間 間 間 家 間 間 家 家 間 家時学間家時学間家時学 家時学間家時学間家時学 間 間 間 間 家時学間家時学間家時学間 間 Y:0 X:0 X:240 図形を書いています (以下プログラム) ブルーの円 void test zukei(void) ł gbox(50,50,200,200,clBlue,clAqua); //四角 gcircle(100,100,50,2,clGreen); //mode 2 円周強調 丸 gcircle(120,140,25,1,clRed); //mode1 内部ベタ塗 丸 //mode 0 円周描画 丸 gcircle(220,220,15,0,clBlue); } 初めに gbox(50,50,200,200,clBlue,clAqua); //四角 起点が X:50、Y:50 終点が X:200、Y:200 の外が青、中がアクア(青緑)色の四角形を描く gcircle(100,100,50,2,clGreen); //mode 2 円周強調 丸 起点が X:100、Y:100、半径 50、描画 mode:円周強調、緑色の円を描く //mode1 内部ベタ塗 丸 gcircle(120,140,25,1,clRed); 起点が X:120、Y:140、半径 25、描画 mode: 内部赤で塗る //mode 0 円周描画 gcircle(220,220,15,0,clBlue); 丸 起点が X:220、Y:220、半径 15、描画 mode:通常、青色描画 見えにくいですがブルーの円です。



}

gwrite32_64() 関数は 32×32 ドットデータを 2 倍の 64×64 に表示する関数です。 gwrite32_96() 関数は 3 倍の 96×96 ドットデータに拡大し表示します。使い方は後述し ますが、フォントメーカーで 32 ドットデータを作成すれば 32,64,96 ドット、3 種類の大き さの文字、図形が液晶画面に表示できます。



test_kanji3();

//漢字3表示



void test_kanji3(void)

{

| gwrite32_64(90,10,kanji_mae); //32×32 漢字 前を 64×64(2 倍) に拡大して表示 |
|---|
| gwrite32(200,10,kanji_usiro); //32×32 漢字 後を表示 |
| gwrite32_96(140,220,kanji_syutu);//32×32 漢字 出を 96×96(3 倍) に拡大して表示 |
| gwrite32_96(140,100,kanji_riki); //32×32 漢字 力を 96×96(3 倍) に拡大して表示 |
| |
| gwrite32(10,280,kanji_mae); //32×32 漢字 前を表示 |
| gwrite32(42,280,kanji_usiro); //32×32 漢字 後を表示 |
| gwrite32(10,240,kanji_syutu); //32×32 漢字 出を表示 |
| gwrite32(42,240,kanji_riki); //32×32 漢字 力を表示 |
| |
| |

}

本ライブラリは一般的なライブラリのようにpset()、line()などで表示させる

絵を作ることも可能ですが、固定された形であれば、あらかじめフォントメーカーで16× 16、32×32 ドットの表示したい文字や絵を作り、関数で16、32、64、96 ドット表示させる 方が断然、描画が高速です。

init_lm_flg=0; ⑧ //レベルメーター初期化

⑧はこの後、使うレベルメーターコンポーネントの初期化です。

while(1) (9)

{

⑨これ以降、無限ループ動作です。

for(loop5 = 0; loop5 < 10; loop5++) (10)

{ Ioop5は0から9まで変化します。

touch_switch_test(); ① //タッチスイッチテスト 液晶表面のタッチスイッチのデータを読み、各種描画を行います。タッチスイッチは押さ れた座標がそれぞれ8bitデータ で変数 TX_data, TY_dataに格納されま す。この関数では、押されるたびに異なる図形を描画していて、押された位置情報は参照 していません。



meter_red(10,10,fsize_32,loop5*455); ⑭//メーター赤色

(15)

{
int_wait(500);

}

Ioop5が9の時だけ、見た目の区切りのために500msecのウエイトを入れて止めています。このウエイト関数は1msec定周期割り込みで作成されていて、極めて高精度です。

●デモンストレーション2 RXD230_LCD_AD1 A Dコンバータ値を数値、レベルメーター、縦棒グラフ、メーターで表示

ADコンバータの値を数値で表示し、縦棒グラフ表示で時間経過によるデータの変化を示 すことが出来ます。

●動作が見える youtube URL https://www.youtube.com/watch?v=KV62mYcnDxM







//12bit AD スタート

//

R_S12AD_Start(); ①

//時間計測

gbeta(clWhite);

LED ON;

//ベタ塗

| init_lm_flg = 0; | |
|------------------------|---|
| $init_vc_flg = 0; (2)$ |) |

//レベルメーター初期化 //縦棒グラフ初期化

```
gwrite16(5,5,S0);
                      3
gwrite16(5,75,S3);
gwrite16(20,75,Sten);
gwrite16(35,75,S3);
gwrite16(50,75,V);
```

//縦棒グラフ目盛り付け

while(1) //AD 変換開始 S12AD.ADCSR.BIT.ADST = 1;4 while(S12AD.ADCSR.BIT.ADST) ; ad0 = S12AD.ADDR0; 5 //AD 変換終了 10µsec 6 //演算開始 fdata1 = ad0/(4095/3.3);sprintf(ad_buff,"%1.2f¥n",fdata1); \bigcirc 8 if(ad0 < lm yellow level) { lcd_puts_color(20,100,fsize_64,ad_buff,clGreen,clWhite);//緑文字 演算開始からここま

で 180msec

{

} else { 9 if(ad0 < lm_red_level) { lcd puts color(20,100,fsize 64,ad buff,clYellow,clWhite); //黄色文字 } else { lcd puts color(20,100,fsize 64,ad buff,clRed,clWhite); //赤文字 } } meter yellow(10,180,fsize 64,ad0); (10) //meter 46msec

level_meter(10,250,fsize_64,ad0); //level meter 140msec (11) Vbar chart(30,222,5,fsize 64,0,clYellow,ad0); 12 //x:開始、x:終了、y:y 座標、サイズ、描画モード、色 //縦棒グラフ 500µsec //1 ループ 360msec 程度 1秒間に約3回

}

〇プログラム解説

//12bit AD スタート

//

R_S12AD_Start(); ①

①コード生成機能で自動的に作られる AD コンバータ初期設定関数です。

//縦棒グラフ目盛り付け

| LED_ON; | //時間計測 | |
|----------------------|--------|--------------|
| gbeta(clWhite); | | //ベタ塗 |
| | | |
| init_lm_flg = 0; | | //レベルメーター初期化 |
| $init_vc_flg = 0; @$ | | //縦棒グラフ初期化 |

②縦棒グラフの初期設定フラグです。

gwrite16(5,5,S0); ③ gwrite16(5,75,S3); gwrite16(20,75,Sten); gwrite16(35,75,S3); gwrite16(50,75,V);

③縦棒グラフのスケール 0, 3.3V を表示しています。

while(1)

{

S12AD.ADCSR.BIT.ADST = 1; ④ //AD 変換開始 while(S12AD.ADCSR.BIT.ADST)

④AD 変換を開始し、データが揃うのを待ちます。

;

ad0 = S12AD.ADDR0; ⑤ //AD 変換終了 10μsec ⑤データが揃ったら、変数 a d O に入れます。

fdata1 = ad0/(4095/3.3); ⑥ //演算開始 ⑥AD コンバータデータは分解能 12bit ですので、O-4095 迄変わります。4095 で 3.3V と表 示するために演算を行います。 sprintf(ad_buff,"%1.2f¥n",fdata1);
⑦

⑦16 進数データを表示するためにASCII文字に変換し、ad_buffに整数1桁、
 小数点以下2桁に変換します。

| ウォッチ1 | | 20 | |
|-----------------|--------------------|------------------|------------|
| 🗃 🧐 👯 🧠 🏂 | 🚺 🗙 表記(N)- | Hes | |
| ウォッチ式 | 値 | 型情報(バイト数) | アドレス |
| 🔹 ad0 🔪 🔪 | 4000 (9-0ff8) | uint16_t(2) | 0×00000578 |
| 📷 S1240. ADD R0 | 0×0ff8 | Ton(2) | 0×00089020 |
| fdata1 | 3.294359E+000 | float(4) | 0×000009a4 |
| 😑 👽 ad_buf f | "3.29" | uint8_t 20 | 0×00000037 |
| 😜 [0] | '3' (0×33) | uint8_t(1) | 0×00000037 |
| 😜 [1] | '.' (0x2e) | uint8_t(1) | 0×00000038 |
| 😜 [2] | '2' (0x32) | uint8_t(1) | 0×00000039 |
| 💊 [3] | '9' (0×39) | uint8_t(1) | 0×0000003a |
| 🥥 [4] | '' (0x0a) | uint8 t(r) | 0×0000003b |
| e | '' (0x0 <u>0</u>) | uint8_t(1) | 0×0000003c |
| 6] | (0x00) | uint8_t(1) | 0×000003d |
| 😜 [7] | '' (0x00) | uint8_t(1) | 0×0000003e |

例えば入力が 3.29V の場合、上記のように 1 数字毎に a d _ b u f f [] に 3.29 と格納 されます。

if(ad0 < lm yellow level) (8) { lcd puts color(20,100,fsize 64,ad buff,clGreen,clWhite);//緑文字 演算開始からここま で 180msec } ⑧レベルメーターは通常、緑表示ですが、Im_yellow_levelを越て、Im_red_level以下の 場合、黄色、以上の場合、赤表示になります、ここでは数値表示も同期して色を変えてい ます。現在の設定はRX230_lcd.hの中になります、黄色が3597/4095×3.3V≒2.9V、赤が 3924/4095×3.3V=3.16V です。 #define lm_yellow_level 3597 #define lm red level 3924 else { (9) if(ad0 < lm red level) { lcd puts color(20,100,fsize 64,ad buff,clYellow,clWhite); //黄色文字 ⑨Im_yellow_level以上で、Im_red_level以下の場合、文字黄色、背景色白にしています。 例 1.90→3.12 Im_red_level 以上で 文字赤、背景色白にしています。 例 3.12→3.30 else {

lcd_puts_color(20,100,fsize_64,ad_buff,clRed,clWhite); //赤文字 } meter_yellow(10,180,fsize_64,ad0); ⑩ //meter 46msec

10回転するメーターです。ad 0の値 0-4095 を 0-9 の針の位置で表示します。

level_meter(10,250,fsize_64,ad0);

//level_meter 140msec



 (\mathbf{N})

⑪レベルメーターです。 a d O の値 0-4095 を 0-11 分割して位置で表示します。



①縦棒グラフです。Y軸高さ64(入力電圧レベル)、X軸64×3=192ドット(時間軸)の棒 グラフを作成します。黄色線で描画します。オシロスコープのように時間軸での電圧変化 が目視出来ます。

(11)

Vbar_chart(30,222,5,fsize_64,0,clYellow,ad0); 12

//x:開始、x:終了、y:y 座標、サイズ、描画モード、色

//縦棒グラフ 500µsec

//1 ループ 360msec 程度 1 秒間に約3回

}

●デモンストレーション3 RXD230_LCD_AD2
 A Dコンバータ値を数値、レベルメーター、メーターで表示
 文字、背景の色を独立して 16 色ずつ設定できる関数を使って文字表示。
 ●動作が見える youtube URL https://www.youtube.com/watch?v=3CuG_DaCR30



gwrite16_color(10,10,S3,clWhite,clGray); ① //文字表示 gwrite32_color(30,10,S4w,clSilver,clLime); ② gwrite32_64_color(70,10,S5w,clAqua,clTeal); ③ gwrite32_96_color(140,10,S6w,clRed,clYellow); ④

〇プログラム解説

既説明の部分は省略します。

gwrite16_color(10,10,S3,clWhite,clGray); ① //文字表示 ① 16×16ドット、数字の3を白色、背景グレーでX:10、Y10:の位置に表示。

gwrite32_color(30,10,S4w,clSilver,clLime); ② ② 3 2 × 3 2 ドット、数字の 4 をシルバー、背景をライムで X:30, Y:10 の位置に表示。

gwrite32_64_color(70,10,S5w,clAqua,clTeal); ③

③32×32データ、数字の5を64ドットに拡張し、アクア、背景をテイルでX:70,Y10 の位置に表示

gwrite32_96_color(140,10,S6w,clRed,clYellow); ④

④32×32のデータ、数字の6を96ドットに拡張し、赤、背景を黄色で X:140, Y10の 位置に表示

●デモンストレーション4 RXD230_LCD_安全

安全、注意、危険の表示とタッチスイッチによる表示切替。安全、注意、危険を漢字や色 で表示でき、より人間に理解でき易い表示になります。

●動作が見えるyoutubeURL https://www.youtube.com/watch?v=D5WdA6a_UUE



●プログラム while(1)

{

 $cnt++;if(cnt > 12){cnt = 0;}$ ①

```
sprintf(ad buff,"%1.2f¥n",cnt/3.63); ②
lcd_puts_color(20,240,fsize_32,ad_buff,clWhite,clNavy); //xposi yposi mode(16,32,64,96) data[]
level meter(60,280,fsize 32,cnt*341);
                                           3
                                                //level meter 140msec
meter_yellow(20,280,fsize_32,cnt*341);
                                           ④ //meter 46msec
if(cnt < 11)
               5
{
                                                           //安
     gwrite32_96_color(20,100,an,clWhite,clLime);
     gwrite32_96_color(20+96,100,zen,clWhite,clLime);
                                                           //全
}
else
{
     if(cnt < 12)
     {
           for(cnt2 = 0; cnt2 < 5; cnt2++)
                                           6
           {
```

```
gwrite32_96_color(20,100,cyuu,clWhite,clYellow); //注
gwrite32_96_color(20+96,100,i,clWhite,clYellow); //注
int_wait(100);
gwrite32_96_color(20,100,cyuu,clRed,clYellow); //注
gwrite32_96_color(20+96,100,i,clRed,clYellow); //注
int_wait(100);
touch_switch_test();
}
else
{
gwrite32_96_color(20,100,ki,clWhite,clRed); //症
gwrite32_96_color(20+96,100,ken,clWhite,clRed); //症
gwrite32_96_color(20+96,100,ken,clWhite,clRed); //症
```

```
}
}
else
{
gwrite32_96_color(20,100,ki,clWhite,clRed); //
gwrite32_96_color(20+96,100,ken,clWhite,clRed); //
int_wait(2000);
}
touch_switch_test();
```

}

〇プログラム解説

cnt++;if(cnt > 12){cnt = 0;} ① ①このプログラムでは AD の値でなく、変数 c n t を0から12まで変えて動作を見ていま す。

```
gwrite32 96 color(20+96,100,zen,clWhite,clLime);
                                                         //全
            }
⑤ c n t が 11 以下では「安全」とライム色、背景白で表示します。96×96 ドットです。
            else
            {
                if(cnt < 12)
                {
                    for(cnt2 = 0;cnt2 < 5;cnt2++)
                                             6
                    {
                        gwrite32 96 color(20,100,cyuu,clWhite,clYellow); //注
                        gwrite32_96_color(20+96,100,i,clWhite,clYellow); //意
                        int wait(100);
                        gwrite32_96_color(20,100,cyuu,clRed,clYellow); //注
                        gwrite32 96 color(20+96,100,i,clRed,clYellow); //意
                        int_wait(100);
                        touch switch test();
                    }
                }
⑥ c n t が 11 以上、12 以下で「注意」が表示されますが、より注意を喚起するために白文
字、背景黄色、
赤文字、背景黄色を100msecの時間を挟んで5回繰り返しています。
タッチスイッチデータ読み込んで、タッチがあれば各種表示します。
                else
                {
                    gwrite32 96 color(20,100,ki,clWhite,clRed);
                                                             //危
                                                                     (7)
                    gwrite32_96_color(20+96,100,ken,clWhite,clRed);
                                                             //険
                    int wait(2000);
```

}

⑦12で「危険」と表示します。白文字、背景赤、96×96ドットサイズで2秒間表示します。

■備考

●CS+ for CC 初めからプログラムを作る設定

🕼 RX230_LCD_安全 - CS+ for CC - [r_cg_main.c]

| 新規作成(N) | • | E. | 新しいプロジェクトを作成(N) | |
|-----------------|--------|----|-----------------------|----|
| ファイルを開く(0) | Ctrl+O | 5 | 新しいマルチコア用プロジェクトを作成(W) | 00 |
| エンコードを指定して開く(W) | | D. | 新しいファイルを作成(F) Ctrl+N | |

64pinのR5F52305AxFを選択します。ファイル名はここではRX230_newにしました。

| リリシェクト作成 | | | > |
|---|---|--|---------------------|
| マイクロコントローラ(<u>1</u>): | RX | | ~ |
| 使用するマイクロコントローラ | 5(<u>M</u>): | | |
| 🏔 (マイクロコントローラを検 | 読索できます) | アップデート(<u>U</u>) | |
| ➡ ■ RX210 ➡ ■ RX21A ➡ ■ RX220 ➡ ■ RX230 ■ RSF52305A×F ■ RSF52305A×F ■ RSF52305A×F ■ RSF52305A×F ■ RSF52305A×F | L(48pin) M(64pin) P(100pin) A(100pin) | 品種名:R5F52305A×FM 内蔵ROMサイズ[K/ドイト]:128 内蔵RAMサイズ[/ドイト]:32768 追加情報:Package=PLQP0064KB-A | < |
| 1 101 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | | | |
| プロジェクトの種類(<u>K</u>): | アプリケーション(CC |)-RX) | ~ |
| プロジェクトの種類(<u>K</u>): プロジェクト名(<u>N</u>): | アプリケーション(CC RX230_new | C-RX) | ~ |
| プロジェクトの種類(<u>K</u>): プロジェクト名(<u>N</u>): 作成場所(<u>L</u>): | アプリケーション(CC RX230_new C:¥Users¥hirosi¥ | D-RX) OneDrive¥ドキュメント¥WorkSp IMマ | ✓ 参照(<u>R</u>) |
| プロジェクトの種類(<u>K</u>): プロジェクト名(<u>N</u>): 作成場所(<u>L</u>): | アプリケーション(OC RX230_new C:¥Users¥hirosi¥ ビ プロジェクト名の | C-RX) (OneDrive¥ドキュメント¥WorkSp IMI))フォルダを作成する(<u>A</u>) | ✓ 参照(<u>R</u>) |
| プロジェクトの種類(<u>K</u>): プロジェクト名(<u>N</u>): 作成場所(<u>L</u>): C:¥Users¥hirosi¥OneDrive | アプリケーション(OC RX230_new C:¥Users¥hirosi¥ ご プロジェクト名の W プロジェクト名の | ン-RX) (OneDrive¥ドキュメント¥WorkSp アランマン Dフォルダを作成する(<u>A</u>) ace¥RX230_new¥RX230_new.mtpj | ✓ 参照(<u>R</u>) |
| ブロジェクトの種類(<u>K</u>): ブロジェクト名(<u>N</u>): 作成場所(<u>L</u>): C:¥Users¥hirosi¥OneDrive] 既存のブロジェクトのファイ | アプリケーション(OC RX230_new C:¥Users¥hirosi¥ ☑ プロジェクト名の ¥ドキュメント¥WorkSpa (ル構成を流用する(<u>S</u>) | C−RX) {OneDrive¥ドキュメント¥WorkSp ⊮√ Dフォルダを作成する(<u>A</u>) ace¥RX230_new¥RX230_new.mtpj | ✓ 参照(<u>R</u>) |
| プロジェクトの種類(<u>K</u>): プロジェクト名(<u>N</u>): 作成場所(<u>L</u>): C:¥Users¥hirosi¥OneDrive] 既存のプロジェクトのファイ 流用元のプロジェクト(<u>P</u>): | アプリケーション(OC RX230_new C:¥Users¥hirosi¥ ⑦ プロジェクト名の *ドキュメント¥WorkSpa (ル構成を流用する(S) (流用元のプロジェク) | C-RX) OneDrive¥ドキュメント¥WorkSp IMマ Dフォルダを作成する(A) ace¥RX230_new¥RX230_new.mtpj ロクト・ファイルを入力してください) | ✓ 参照(R) |

クロックは外部の12.5MHzを使います。CPUは50MHzで動かします。

| 2 0 2 2 | 🔞 コードを生成する 🍰 🧯 🥳 💓 🗱 | 100001 |] & & 🛎 J 🕮 J 🖷 |
|--|--|----------------------------------|-----------------------|
| □ | クロック設定 ブロック図 | | |
| | VCCBD | | |
| | ● 27 (\0 < \/CC < 55 (\0 | \bigcirc 24.0.0 < VCC < 27.0.0 | 0.1800 < 100 < 2400 |
| | 0 2.1 (0/ 3 000 3 0.0 (0/ | 0 24(0) 3 000 (24(0) | 0 10 (0/3 000 (24 (0) |
| | - メインクロック発振器、SSIクロック(SSISCK)設定 | | |
| 中心 索压输出回路 | ☑ 動作 | | |
| ● クロック周波数精度測定回路 | メインクロック発振源 | 発振子 | ~ |
| 📦 消費電力低減機能 | | 12.5 | (MHz) |
| | Contraction of Contra | | |
| ₩_ /\͡ス | 单 | 2サイクル ~ 0.5 | (µs) |
| <u>⊕</u> | 発振停止検出 | 無効 | ~ |
| | | | |
| | PLL回路設定 | | |
| | | | |
| ■ ポートアウトプットイネーブル2 | 入力分周比 | × 1/2 v | |
| | | × 8 ~ | |
| □ 22ペアマッチタイマ | | | |
| ⊕… 👽 16ビットタイマパルスユニット | | 50 (MHz) | |
| | - BTC サブクロック発振器設定 | | |
| ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ | | | |
| | サブカロック発振器ドライブ能力 | ドライブ能力 低 | |
| | | | |
| ● CRC 演算器 | 后以废蚕灰 | 32./68 | (KHz) |
| 中 2 シリアルコミュニケーションインタフ | -高速オンチップオシレータ(HOCO)設定 | | |
| | | | |
| | | 54 | (MHz) |
| | | | |
| ⊕ ● コンパレータB | - 11(J)速オンチップオシレータ(LOCO)設定 | | |
| | | 102 | |
| | 1吉);废蚕荚 | 4 | (MHz) |
| | | | |

| 2 0 2 2 | 🐻 コードを生成する 🛃 🗋 🚜 达 💕 👪 🕄 | 9 # # # 2 0 0 0 (| 친 🗆 윤 윤 🕾 문 🎜 문 🖷 🔍 🛝 🕂 🕸 볼 |
|--|--------------------------------|--------------------------|----------------------------------|
| □ RX230 LCD test1 (プロジェクト) ^ | - BTC サブクロック発振器設定 | | |
| R5F52306AxFM (マイクロコントローラ) | | | |
| □ *型 コード生成 (設計ツール) | Hづhnwh祭振器ドラノづ始力 | ドライブ 能力 任 | |
| Ⅲ / 端子図 | 5750559Enrer(5154E) | 1.51.5 8653 184 | × |
| | 周波数 | 32.768 | (kHz) |
| 0090発生回路 | | | |
| | - 向速オノナッノオンレーダ(HOGO)設定 | | |
| → 消毒電力任減機能 | | F 4 | 0.003 |
| → 割り込みコントローラ | 1012度至2 | 04 | (MH2) |
| | - 低速オンチップオシレータ(LOCO)設定 | | |
| | □ 動作 | | |
| ש סארכאש סארבי ש | [四])成贵女 | 4 | (MHz) |
| ● データトランスファコントローラ | シュラティンロック語会 | | |
| イベントリンクコントローラ | hnwhy-7 | PIIOS | |
| ■ マルチファンクションタイマパルスニ | | | |
| ····・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ | システムクロック(ICLK) | × 1 ~ 50 | (MHz) (CPU, DMAC, DTC, ROM, RAM) |
| ■ ■ 8ビットダイマ == 2 = 2 = 2 = 2 = 2 = 2 = 2 = 2 = 2 = | 周辺モジュールクロック(PCLKA) | × 1 v 50 | (MHz) (MTU2) |
| | 周辺モジュールクロック(PCLKB) | × 1/2 v 25 | (MHz) (その他の周辺機能) |
| ♥ リアルタイムクロック ♥ ウォッチドッグタイマ | 周辺モジュールクロック(PCLKD) | × 1 ~ 50 | (MHz) (S12AD) |
| | 外部バスクロック選択(BCLK) | × 1/2 v 25 | (MHz) (外部バスコントロールと外部端子出力) |
| ▼ ローバワータイマ | FlashIFクロック(FCLK) | × 1/2 v 25 | (MHz) (FlashIF) |
| | - IWDT専用オンチップオシレータ(IWDTLOCO)設定 | | |
| | 周波数 | 15 | (kHz) |
| 12L9FA/DJ//-9 | | | |
| | -LP1クロック(LP1CLK)設定 | | |
| ↓ データ演算回路 | □ 動作 | | |
| | クロックソース | サブクロック発振器 | ~ |
| - 🔨 CC-RX (ビルド・ツール) | 国际中学校 | 1/0 | (C) () |
| | halfile 21X | x 1/2 V | UKHZJ |

電圧検出回路はリセット電圧を決めます。液晶板が3.3V動作なので、RX230も3.3 Vで動かし、3.3Vより下の2.82Vにします。

| ファ | イル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) う | デバッグ(D) ツール(T) ウインドウ(W) ヘルプ(H) | |
|------|--|--|-----------------------------|
| | | | |
| 8 | 🖏 Z9-H(S) 🛛 🛃 🛃 🎒 🕺 🐚 🛍 🔊 🤇 | × A ♣ ♣ · 100% | 🝸 ன ன DefaultBuild 💽 🔨 |
| 9 | 두 🖓 🖓 🦓 🗆 🗩 다 이 🏷 | ソリューション一覧(S) | |
| | プロジェクト・ツリー | ☆フロパティ ⊴r_cg_mainc ⊴r_cg_sciuserc | ✓r_cg_scic |
| マーーン | 2 ② 2 ② RX230 LCD test1 (プロジェクト) RSF52306AxFM (マイクロコントローラ) | ③ コードを生成する 点 □ ぷ と w 編 LVD0 LVD1 LVD2 | ₩ # # # © © © © @ @ @ # # # |
| ニュアル | 白… ⁻ ⁻ | - 電圧検出回路0動作設定 ()使用しない - 電圧検出設定 | ● 使用する |
| | □ | 電圧検出レベル - 雷圧検出の回路起動設定 | [2.82 V) |
| | | ○ リセット後、電圧監視のセット有効 ● リセット後、電圧監視のセット無効 | |



RX230マイコンと液晶表示器の接続はSPIバス、タッチスイッチ、__CS(チップ セレクト)、__RESET、バックライト選択などの線が接続されています。

I/0 ポート

PA ポートは PA1 が SCL、PA3 が RXD5、PA4 が TXD5 と SPI インターフェイスに使われるため、 使用しない設定にする必要があります。

| | 🐻 コードを生成する | i 🚣 🖨 💩 | 🗠 💕 🖩 | 제 문 및 2 |) 🦚 🔿 🔿 🗇 🔳 🔗 ८ | 8 📧 7 18 7 🖷 🏊 🛧 🕯 |
|--|--|-------------|-------------|-------------------|-----------------|--------------------|
| ■ RSF52306AxFM (マイクロコントローラ) | Port0 Port1 Por | t2 Port3 Po | rt4 Port5 [| PortA PortB PortC | PortE PortH | |
| □- · · · · · · · · · · · · · · · · · · · | ○ 使用しない PA1 | 〇入力 | ◉ 出力 | □ 内蔵ブルアップ | CMOS出力 ~ | 🗌 1を出力 📋 高駆動出力 |
| ✓ クロック発生回路 図- 電圧検出回路 グロック周波数糖度測定回路 | 使用しない PA3 | O እታ 😗 | 〇 出力 🤨 | □ 内蔵ブルアップ | СМОЅ出力 | 🗌 1を出力 🔛 高駆動出力 |
| | ● 使用しない PA4 | 🔿 ኢታ 😲 | 〇 出力 👎 | □ 内蔵ブルアップ | CMOS出力 | 🗌 1を出力 📋 高駆動出力 |
| | 使用しない PA6 | 0 дл 😗 | 〇 出力 😲 | □ 内蔵ブルアップ | CMOS出力 🗸 | 🗌 1を出力 📋 高駆動出力 |
| | ○ 使用しない | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 ~ | 🗌 1を出力 📋 高駆動出力 |

PB ポートは PBO がタッチスイッチの T_IRQ、PB1 が液晶バックライト:通常点灯(昼用)、 PB3 が夜間点灯(夜用)を制御します。PB7 は基板に実装済の LD1 に抵抗を介して接続され ていて、各種インジケート、関数時間計測に使用します。

| | 🕲 コードを生成する | 1 🔬 🗋 | a 🗠 💕 | u 1 🗛 🛤 🗱 🕼 | 0 0 0 0 0 | 1 & & | 🐮 🍠 12° . | 罗罐龟头包 |
|---|--|----------|-------------|--------------------|-------------|-------|-----------|---------|
| ■ KX230 LCD test1(フロシェクト) へ ■ R5F52306AxFM (マイクロコントローラ) □ □ □ ドチ ボ (型目・ソール) | Port0 Port1 Por - PB0 - | t2 Port3 | Port4 Port5 | PortA PortB PortC | PortE PortH | | | |
| | ○ 使用しない | ③ 入力 | 〇 出力 | 🗹 内蔵ブルアップ | CMOS出力 | ~ | □ 1を出力 | □ 高駆動出力 |
| | -PBI | О ЛЛ | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| | - PB5 | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| | ● 使用しない PB6 | 〇入力 | 〇出力 | □ 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| ジ データトランスファコントローラ ジ イベントリンクコントローラ ジ マルチファンクションタイマパルスニ | ● 使用しない PB7 | О ЛЛ | 〇出力 | □ 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| ポートアウトブットイネーブル2 ⊕ 8ビットタイマ | ○ 使用しない | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |

PE ポートはタッチスイッチインターフェイス、液晶の_CS や_RESET に使用します。

| | | | | | * -144 | | | |
|--|--|----------|-------------|-------------------|-------------|----|---------|-----------|
| | 🐻 コードを生成する | l 🔬 🗋 | a 🗠 💕 | 8 🕫 🖧 😂 (| ð 🧟 🔕 🚳 🖻 🗖 | 88 | 🐮 J 🔀 . | ም 🖷 🏊 👞 ብ |
| | Port0 Port1 Por _PE0 | t2 Port3 | Port4 Port5 | PortA PortB PortC | PortE PortH | | | |
| 日 | ○ 使用しない PE1 | ◉ 入力 | ○ 出力 | 🗹 内蔵ブルアップ | CMOS出力 | × | 🗌 1を出力 | □ 高駆動出力 |
| | ○ 使用しない | О ЛЛ | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| | - rez 〇 使用しない | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| | | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| | ○ 使用しない PF5 | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |
| [©] ポートアウトプットイネーブル2 ⊕ [©] 8ビットタイマ | ○ 使用しない | 〇入力 | ◉ 出力 | 🗌 内蔵ブルアップ | CMOS出力 | ~ | 🗌 1を出力 | 🗌 高駆動出力 |

コンペアマッチタイマーで 1msec (1000 μ sec)の定周期割り込みを動作させます。 int_wait()関数はこの時間を計測しているので、極めて正確(クリスタル精度)です。 AD 変換やタッチキースキャンにも使用出来ます。

| 72- | 2 3 2 8 | 🐻 コードを生成する 🖕 | 🖞 🗋 🚜 🗠 💕 🕯 | 8 50 🖧 🚑 🚳 🤇 | 1000E | 1 & & t | 5 <i>7</i> | 123 🎝 🍽 |
|-----|---|---------------------------|-------------|---------------------|------------|----------|------------|---------|
| ÷ | R5F52306AxFM (マイクロコントローラ) | CMT0 CMT1 CMT2 | CMT3 | | | | | |
| | □『□] コード生成 (設計ツール) | -コンペアマッチタイマ動作 | 設定 | | | | | |
| FIL | → → → 端子図 | ○ 使用しない | | ● 使用する | | | | |
| | 白 🧱 周辺機能 | | | | | | | |
| | | POLK/8 | | ○ PCLK/128 | O POLK/512 | | | |
| | | | 0100002 | 010200120 | 01000012 | | | |
| | | -インターバル時間設定 - インターバル時間 | | 1000 | μs 🗸 | (実際の値:10 | 00) | |
| | | -割り込み設定 | | | | | | |
| | | □ コンペアマッチ割り | 込みを許可(CMIO) | | | | | |
| | | 優先順位 | | レベル15 | ~ | | | |
| | | | | | | | | |
| | □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ | | | | | | | |
| | ホートアウトノットイネーノル2 | | | | | | | |
| | H | | | | | | | |
| | ● · · · · · · · · · · · · · · · · · · · | | | | | | | |
| | | | | | | | | |

RX230 と ILI9341 (液晶コントローラ)の接続は SPI で行われています。SCI5 を簡易 SPI バ スモードに設定。

| R5F52306AxFM (マイクロコントローラ) | SCI1 SCI5 SCI6 SCI8 SCI9 | SCI12 | |
|---|--|-----------------------|--------------------------------|
| □ 📲 コード生成 (設計ツール) | <u>一般設定</u> 設定 | | |
| 由… 🔏 端子図 | - 機能設定 | | |
| ⋳ | ○ 使用しない | | |
| | | ì关/言 | |
| Ⅰ···································· | | A=18 | |
| ● 消費雷力低減機能 | ○ マルチプロセッサモード | 透信 | ~ |
| ● 割り込みコントローラ | ○ クロック同期式 | 送信 | ~ |
| - | ○ スマートカードインタフェース | 送信 | ~ |
| ±€ VOポート | | | |
| | | | |
| | ● 簡易SPI/バス | マ人タ 达信/受信 | ~ |
| | | | |
| 💗 ポートアウトプットイネーブル2 | TYPE DO1 | DVDE DOB | |
| | PG3 | RXU6 PG2 | ~ |
| | SSDA5 PC3 | SSOL5 PC2 | <u> </u> |
| 国… ジョリアルタイトクロック | SMOSI5 PA4 | SMISO5 PA3 | ~ |
| ····· ジョナッチドッグタイマ | | | |
| 🔍 独立ウォッチドッグタイマ | | | |
| | | | |
| ● シリアルペリフェラルインタフェース | | | |
| | | | |
| | | | |
| 転送クロックを早十の60 | | | |
| | JMIT ZIL しまり。 | | |
| R5F52306AxFM (マイクロコントローラ) | SCI1 SCI5 SCI6 SCI8 SCI9 SCI12 | | |
| ☆… 9월 コード生成(設計ツール) | | | |
| □…2 端丁凶 | データ転送方向設定 | @ MOD= _ 7.1 | |
| - 2 クロック発生回路 | ○LSBノアースト | ● M3B/7~×ト | |
| | ● 標準 | ○ 反転 | |
| | 転送速度設定 | | |
| | 転送クロック | 内部クロック ~ | PA1 ~ |
| | ビットレート | 6250000 ~ | (bps) (実際の値: 6250000, エラー: 0%) |
| | □ ビットレートモジュレーション機能有効 | | |
| | | カロック出力 | |
| | | | |
| ···································· | ·クロック設定 | |) |
| | クロックが遅延 | クロック遅れなし ~ | |
| | | | |
| ⊕ ■ 16ビットダイマハルスユニット □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ | データ処理設定 | 中心のコンサービュル・マン スが 用 ナス | |
| | 近信ナーダ処理 | 書切込みサービスルーチンで処理する | |
| | 受信データ処理 | 割り込みサービスルーチンで処理する | ~ |
| | 割り込み設定 | | |
| | ☑ エラー割り込み許可(ERI5) | L | |
| ● ● シリアルコミュニケーションインタフ | ノ TXI5, RXI5, TEI5, ERI5 優先順位 | | |
| | コールバック機能設定 | | |
| | ☑ 送信完了 | ☑ 受信完了 | ☑ 受信IF>- |
| 設定をセーブ! コード生 | 成をします | | |
| | | | |
| ファイル(E) 毎年(E) キテハハ プロジェクト(D) ビルド | (B) $\exists f(wf(D) w = IL(D) d A V K d(MO A II)$ | -T(H) | |



各種ファイルが作成されます。グラフィックライブラリの3つのファイルをホルダにもっ てきて、r_cg_main. cファイルに以下の#includeを追加してください。
 #include "br_fonts.dat"
 //フォントファイル

 #include "rx230_lcd.h"
 //ヘッダファイル
 #include "rx230_lcd.c" //グラフィックライブラリ本体

sprintf を使う場合、以下も追加する。

#include "stdio.h" //無いと sprintf が動かない

| | • <u></u> | | | |
|--|--|---|--|-------|
| 行 📅 アドレス | 1 | | | |
| 75 76 77 78 80 81 82 第3 83 84 85 86 87 87 88 89 90 | | ************************************** | ************************************** | |
| 90 | # | include ‴stdio₊h″ | //無いとsprintfが動かない | / |
| 92 93 94 95 96 97 98 98 99 99 | /> ₩ □// G (#0 // // | * End user code. Do no metwice <u>"r cg</u> userdefi ************************************ | edit comment generated here */ .h~ *********************************** | ***** |

以上、ゼロからプロジェクトを作る場合の説明ですが、前に書いたようにコピペで名前だ け変える方が簡単で落ちが無いと思います。

●フォントデータの作り方



〇フォントメーカーの使い方

「フォントを作る」ホルダに16×16ピクセル用フォントメーカー | fms16. ex eと32×32ピクセル用フォントメーカー | fms32が用意されています。これらフ ォントメーカーは以降で説明する漢字、数字、絵などを液晶画面に表示するためにC言語 で扱える形のデータを製作するソフトウエアです。

| fms16. exeで使い方を説明します。| fms32も使い方は同じです。



データファイルを読み込みます

編集中の描画が他のフォーム等で消えた場合に再描画させます

新規にデータを製作する場合、初めに「BaseColor」をクリックし、色指定しま す。このプログラムは16色の色を指定できます。それ以外の色指定を行うと正常に表示 されませんので、ご注意下さい。

■設定できる16色

それ以外を選択すると液晶表示が白色になります。



| Contraction (Contraction) | | | |
|---------------------------|-----------|---------|-----------|
| D Font M | aker I | | 16×16 Dot |
| | | | 15:58:18 |
| | | | Load |
| | | | Save |
| | | | Flash |
| | | | Memo |
| | | | |
| Color | BaseColor | ReWrite | |

例えば白を選択し、OKをクリックすると16×16ピクセルが全て白になります。

次に、描画色を決めます。「Color」をクリックし、BaseColor同様、設定して下さい。

マウスで変えたい色をクリックします。



描画してOKであればSaveをクリック。拡張子はL16です。付けなくて「保存」で きます。

(Ifms32dL32)

| 保存する場所(I): | 🌗 フォントを作る | - | - € 📸 🐨 | |
|------------|-------------|---------------------------|------------------|----------|
| œ. | 名前 | | 更新日時 | 種類 |
| まい 表示した場所 | 📓 MV.L16 | | 2013/02/20 16:09 | L16 ファイル |
| | Sten.L16 | | 2013/02/20 16:09 | L16 ファイル |
| | 📓 S9.L16 | | 2013/02/20 16:09 | L16 ファイル |
| テスクトップ | 📓 S8.L16 | | 2013/02/20 16:09 | L16 ファイル |
| | 📓 S7.L16 | | 2013/02/20 16:09 | L16 ファイル |
| ライブラリ | 📓 S6.L16 | | 2013/02/20 16:09 | L16 ファイル |
| | 📓 S5.L16 | | 2013/02/20 16:09 | L16 ファイル |
| コンピューター | 📓 S4.L16 | | 2013/02/20 16:09 | L16 ファイル |
| | 📓 S3.L16 | | 2013/02/20 16:08 | L16 ファイル |
| | 📓 S2.L16 | | 2013/02/20 16:08 | L16 ファイル |
| イットワーク | < [| III | | + |
| | ファイル名(N): | inul | • | 保存(S) |
| | ファイルの種類(工): | Font File 16 × 16(* 16) | | キャンヤル |

画像データを読み込む場合はLoadをクリックしてください。

ここでフォントメーカーは2種類のファイルをセーブしました。拡張子L16はデータ専用のバイナリーファイル、C16という拡張子のファイルはエディタで開くためのアスキ ーファイルです。

inu. c16ファイルをメモ帳で開いてみます。char型で先ほど製作した情報がセ ーブされています。このファイルは16×16ピクセルの位置と色情報が入っています。 このデータをそのままbr_fonts_datに追加コピペしてください。



下記例は色の見栄えをテストするサンプル test_colorです。

20250606

製品のお問い合わせは以下にお願いします。

〒350-1213 埼玉県日高市高萩 1141-1 有限会社ビーリバーエレクトロニクス http://www.beriver.co.jp/ info@beriver.co.jp TEL:042-985-6982 Copyright beyond the river Inc. 20250606