EPSON
EXCEED YOUR VISION

Arduino Due 接続用 TFT シールドボード

S5U13781R01C100
グラフィックスライブラリ
ユーザーズガイド

セイコーエプソン株式会社

Rev. 1.01

評価ボード・キット、開発ツールご使用上の注意事項

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないで下さい。本品は、完成品に対する設計品質に適合していません。

2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止して下さい。

3. 本評価ボード・キット、開発ツールに用いられる部品は、予告無く変更されることがあります。

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。

2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 製品および弊社が提供する技術を輸出等するにあたっては「外国為替および外国貿易法」を遵守し、当該法令の定める手続きが必要です。大量破壊兵器の開発等およびその他の軍事用途に使用する目的をもって製品および弊社が提供する技術を費消、再販売または輸出等しないでください。

5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

©SEIKO EPSON CORPORATION 2015, All rights reserved.

目次

1 はじめに

このドキュメントは S5U13781R01C100シールド TFT ボードのグラフィックスライブラリについて解説します。グラフィックスライブラリは、S5U13781R01C100シールド TFT ボードに接続されたパネル上へのグラフィックスやフォントの表示手順を簡略化するように設計されたソフトウェアライブラリです。

S5U13781R01C100シールド TFT ボードは、Arduino Due と接続して使用するように設計されています。Arduino Due の詳細については、Arduino web サイト www.arduino.cc/を参照して下さい。S1D13781および S5U13781R01C100シールド TFT ボードの詳細については、セイコーエプソンの表示コントローラの web サイト http://www.epson.jp/prod/semicon/products/lcd_controllers/を参照して下さい。

注:
S5U13781R01C100 シールド TFT ボードは、S1D13781 と同等の機能を持つシンプル LCD コントローラ S1D13L01 の搭載機能の評価にも適用可能です。S1D13L01 は S1D13781 の機能を受け継いでおり、両者の主な相違は下表のとおりとなります。その他の搭載機能については、vdc.epson.com の S1D13781 Hardware Functional Specification および S1D13L01 Hardware Functional Specification を参照下さい。

機能の違い	S1D13L01	S1D13781
表示インタフェース	TFT パネルのみ	STN パネルおよび TFT パネル
BitBLT サポート	SW BitBLT	Hardware BitBLT Engine with: - Move BitBLT - Solid Fill BitBLT
動作温度範囲	-40 to 85° C	-40 to 85° C or -40 to 105° C
パッケージ	QFP15 128-pin	QFP15 100-pin

製品の使用に当たってのデモ動画を Youtube チャンネル EEAVDC Productions に掲載していますのでご覧ください。ドキュメントに関するご意見は email にて下記アドレスまでお願いいたします。documentation@eea.epson.com.

ドキュメントは、英語版 S5U13781R01C100 Shield Graphics Library Users Guide が正規の資料であり、本書は正規英語版ユーザーマニュアルの補助的資料として、お客様のご理解を深めるために和訳したものです。製品のご検討および採用に当たりましては、必ず正規英語版の最新資料をご確認ください。なお、本書および正規英語版は適宜改訂されています。最新版は、
http://www.epson.jp/prod/semicon/products/lcd_controllers/

http://vdc.epson.com/

からダウンロードできます。

2 必要なもの

S5U13781R01C100 シールド TFT ボードのグラフィックスライブラリは、Arduino Due と接続した S5U13781R01C100 シールド TFT ボードに接続されたパネル上へのグラフィックスやフォントの表示手順を簡略化するように設計されています。

グラフィックスライブラリをインストールする前に、下記の準備ができていることを確認して下さい：

- S5U13781R01C100 シールド TFT ボード
- Arduino Due コントローラボード
- LCD パネル (デフォルトの設定は 480x272 @ 24bpp)
- Arduino Sketch IDE software v1.6.2 またはそれより新しいバージョン (注を参照)
- S5U13781R01C100 Graphics Library package
- マイクロ USB ケーブル (Arduino Due への電源供給およびオプションのシリアルモニターの使用)

注:
Arduino Sketch IDE software は Windows, Mac OS X,または Linux 上で動作します。詳細な必要事項とインストールの仕方については下記 Arduino の web サイトを参照下さい。
www.arduino.cc/en/Main/Software.

開発着手前に、最新版のグラフィックスライブラリを下記 web サイトでご確認下さい。

vdc.epson.com

3　インストール

グラフィックスライブラリのインストールには次の 2 つのステップが必要です：

- ハードウェアの接続
- ソフトウェアのインストール

3.1　ハードウェアの準備

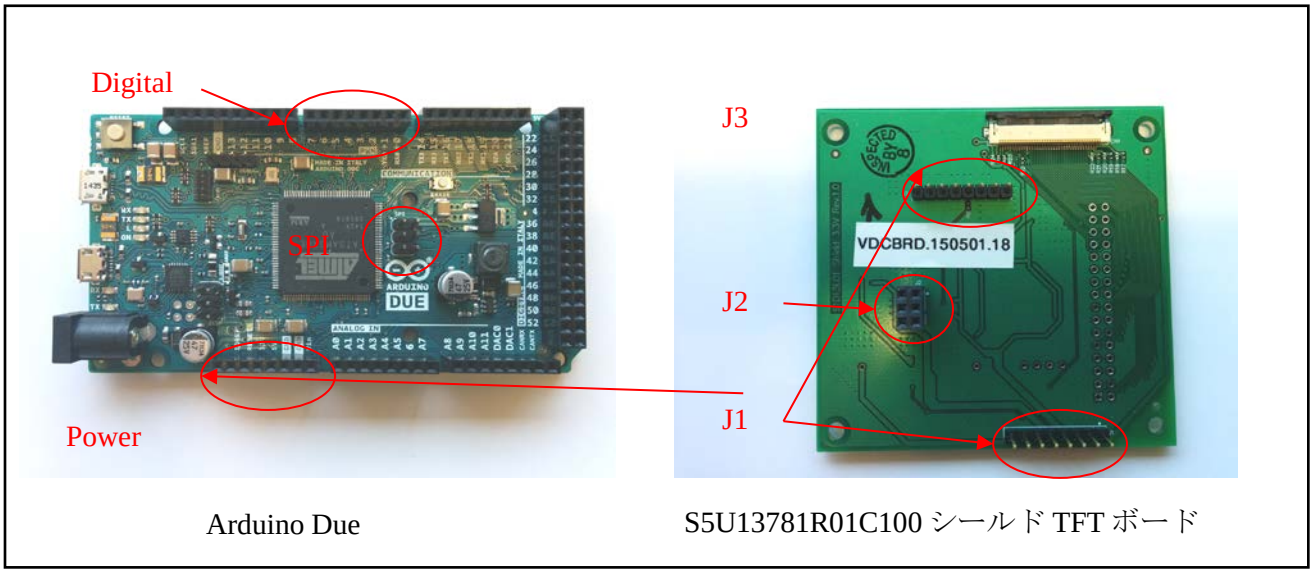
グラフィックスライブラリソフトウェアパッケージをインストールする前に、以下のインストラクションに従ってハードウェアの接続を行ってください。

3.1.1　S5U13781R01C100 シールド TFT ボードと Arduino Due の接続

S5U13781R01C100 シールド TFT ボードと Arduino Due の接続にあたり、両ボードの対応する下記ヘッダー を正しく接続する必要があります。

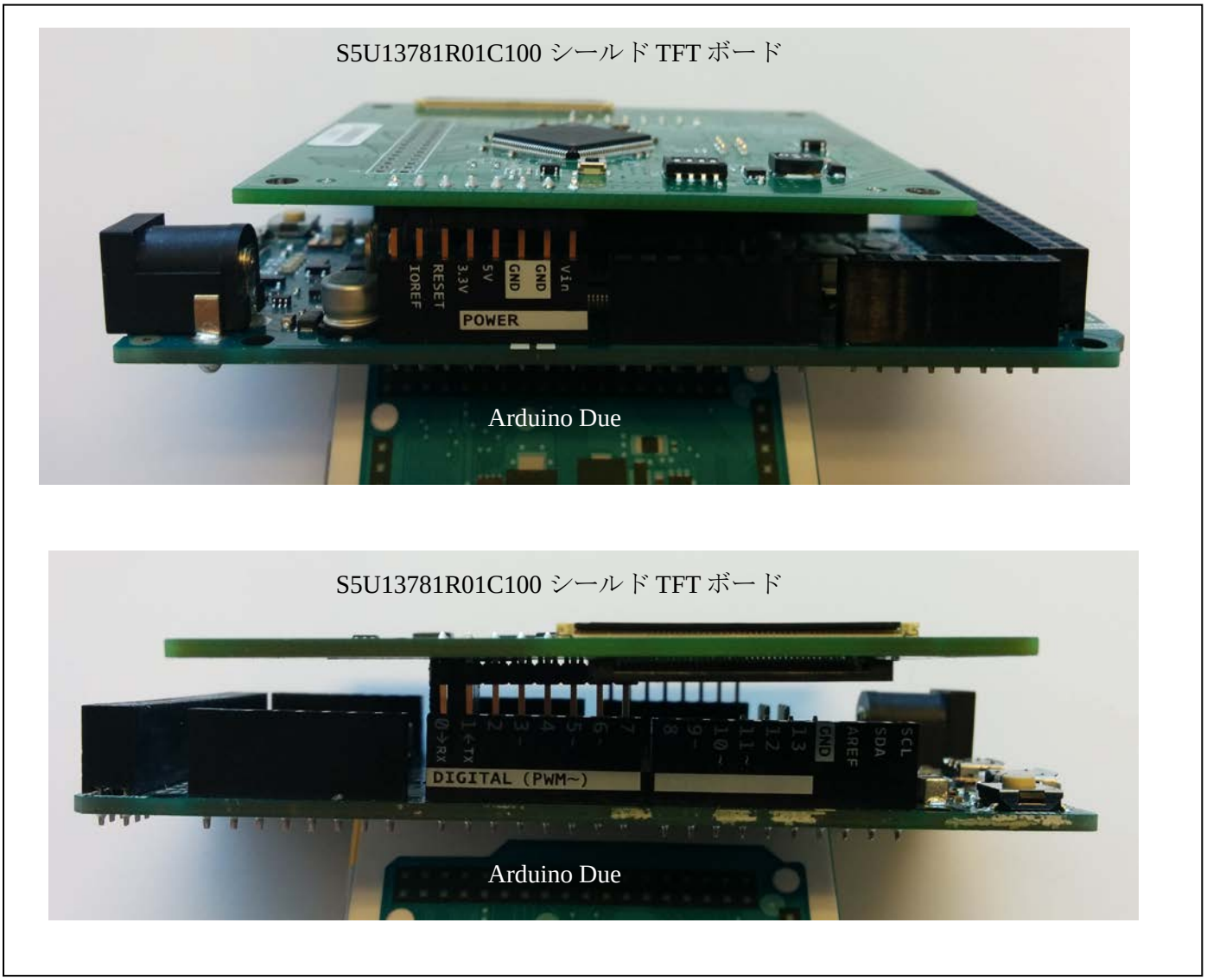
S5U13781R01C100 シールド TFT ボード		**Arduino Due**
J1 Header	←→	“Power” Socket
J2 Header	←→	SPI Socket
J3 Header	←→	“Digital” (PWML) Socket

両ボードのヘッダーとソケットは下図のとおり配置されています。



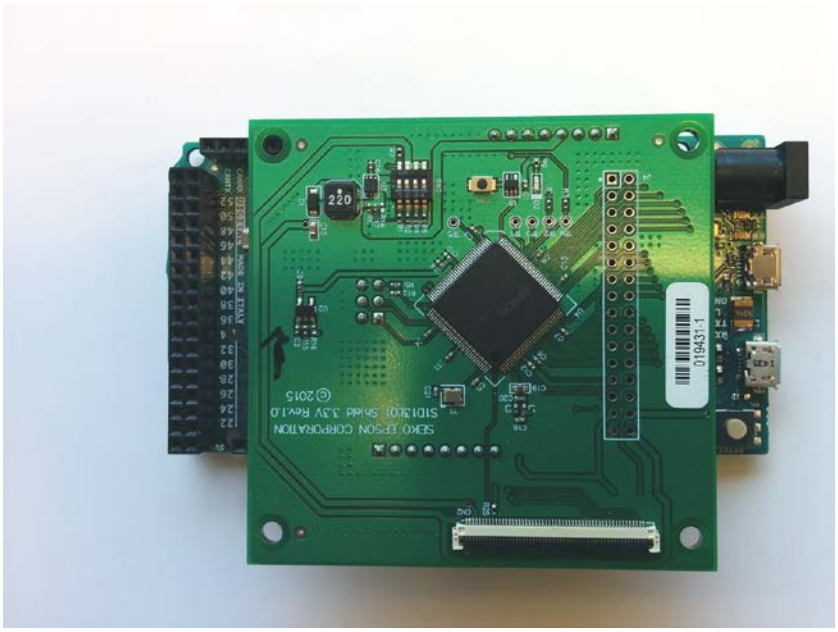
コネクターの位置

ボードを接続の際は、下図のように位置を合わせ、両ボードをゆっくりと押さえながら行ってください。両ボードの位置が正しければ、ヘッダーピンが完全にソケットに入るまでスムーズにスライドします。



S5U13781R01C100 シールド TFT ボードと Arduino Due の接続

ボードを接続すると、下図のようになります。



S5U13781R01C100 シールドTFT ボードと Arduino Due の接続状態

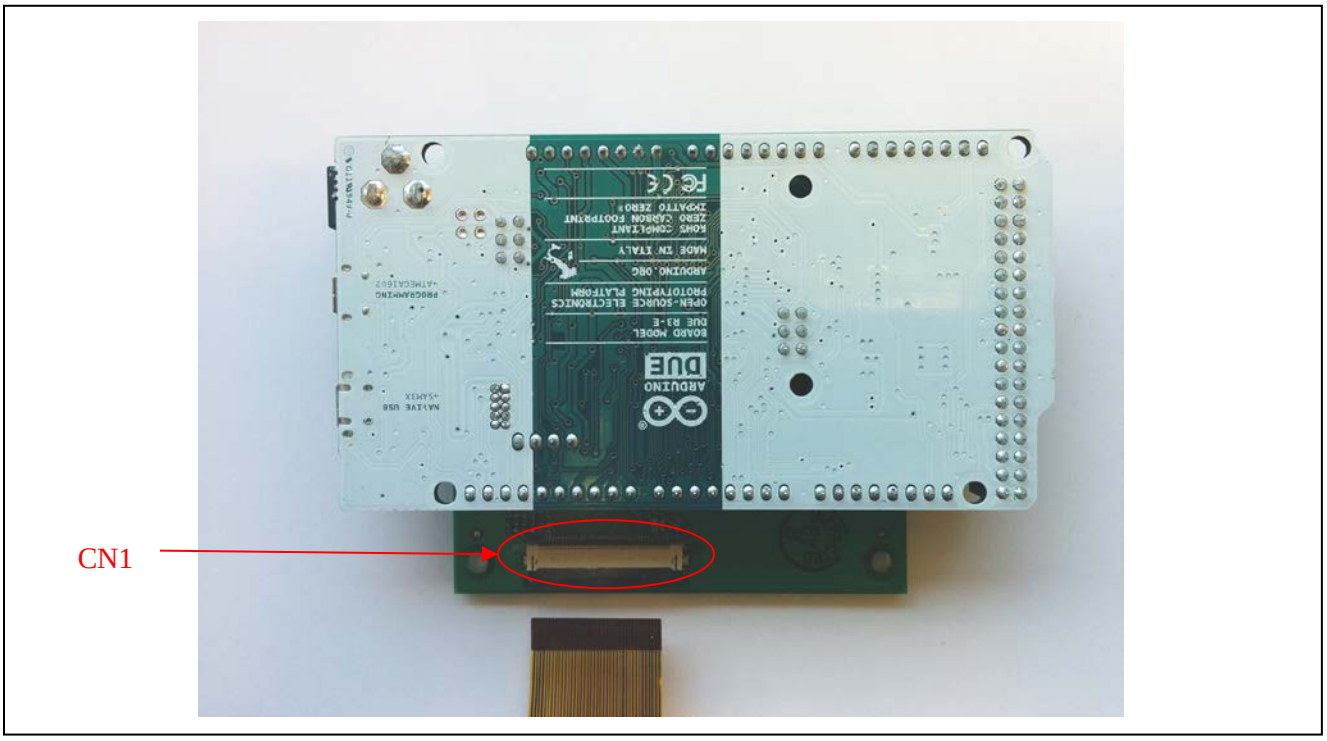
3.1.2 LCD パネルとの接続

両ボードを接続後、LCD パネルを S5U13781R01C100 シールド TFT ボードに接続します。グラフィックスライブラリの初期設定は Newhaven Display 製 480x272 LCD panel (NHD-4.3-480272EF-ATXL#)用になっています。

LCD パネルコネクタは、S5U13781R01C100 シールド TFT ボードの Arduino Due と接続する側に実装されている FPC コネクタの CN1 です。

LCD パネルを接続するには：

1. CN1 のこげ茶色のタブを、ボードの端に向かってゆっくりと引っ張って開きます。
2. フラットパネルケーブルを CN1 に差し込みます。
3. CN1 のこげ茶色のタブを押し込み、元の位置に戻します。



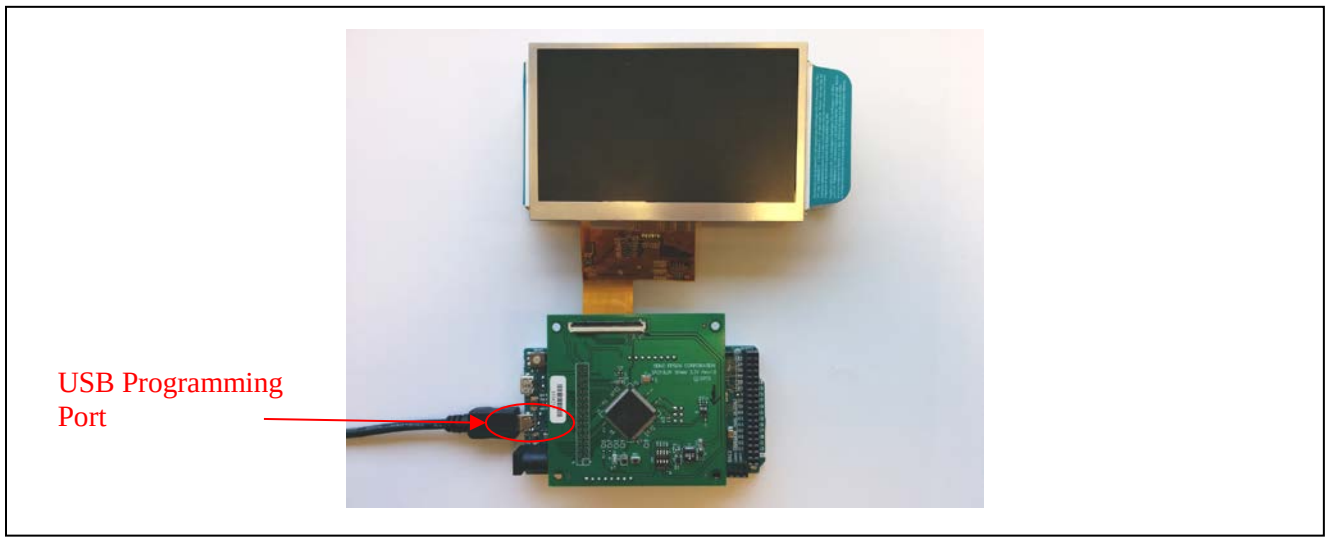
S5U13781R01C100 シールド TFT ボードの LCD パネルコネクタ

パネルケーブルを固定したら、慎重にボードをひっくり返し、パネルを上向きにします。

3.1.3 Arduino Due の開発環境への接続

ハードウェアの接続が完了したら、マイクロ USB ケーブルを Arduino Due のプログラミングポートと開発プラットホーム(パソコン)間に接続して下さい。

これにより、Arduino Due に電源が供給され、デバッグ用のシリアルモニタ機能が使えるようになります。



ハードウェア準備の完了状態

3.2 ソフトウェアインストール

グラフィックスライブラリを使うための開発環境の準備には、下記のステップが必要です。

1. Arduino Sketch IDE のインストール
2. Arduino SAM boards (Due を含む)のインストール
3. グラフィックスライブラリと example sketch のインストール
4. Sketch IDE を使って example sketch をコンパイル後 Arduino Due にアップロード

3.2.1 Arduino Sketch IDE のインストール

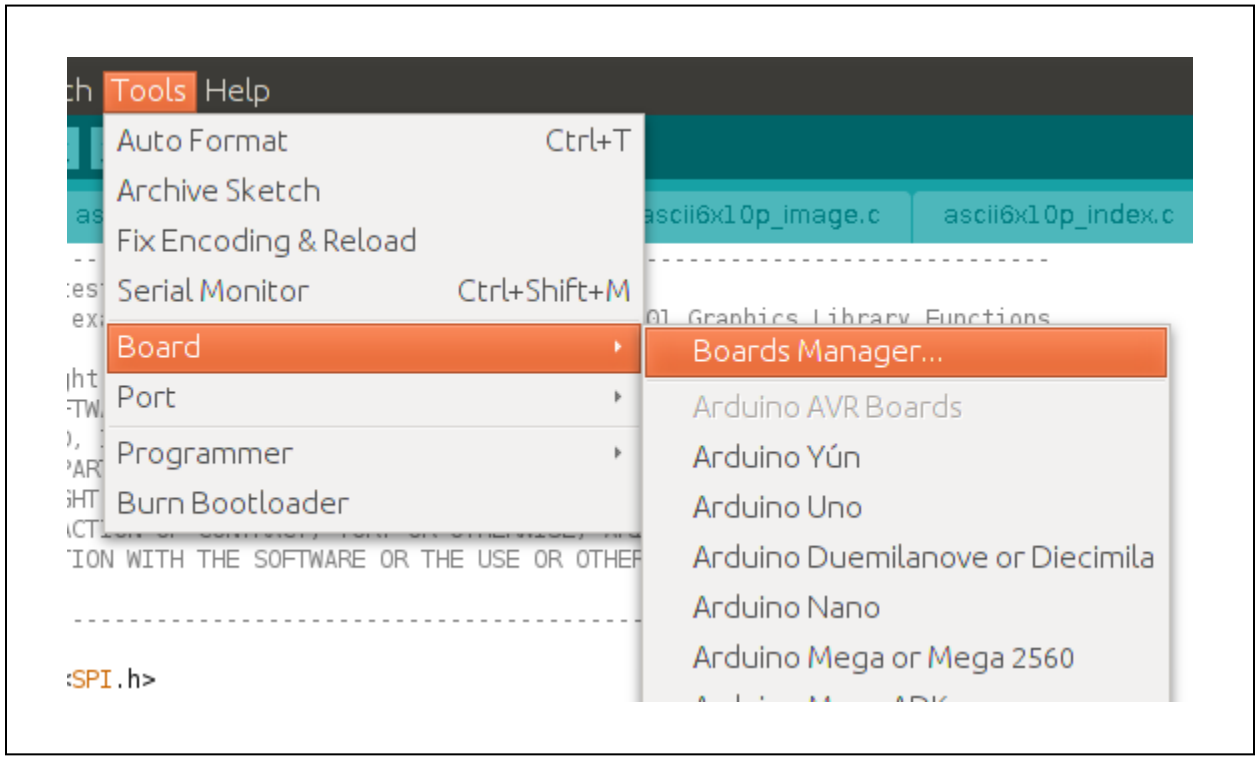
グラフィックスライブラリは、Arduino Sketch IDE 用に設計されており、Sketch v1.6.2 上で開発・検証されています。

Sketch は Windows, Mac OS X, または Linux のプラットホームが必要です。もし、Sketch がインストールされていない場合は、開発環境に合ったバージョンの Sketch を Arduino の web サイトからダウンロードしてインストールして下さい。開発環境に Sketch がインストールされたら次のステップに進みます。

詳細な要求事項やインストール方法については、Arduino の web サイトを参照して下さい。
www.arduino.cc/en/Main/Software.

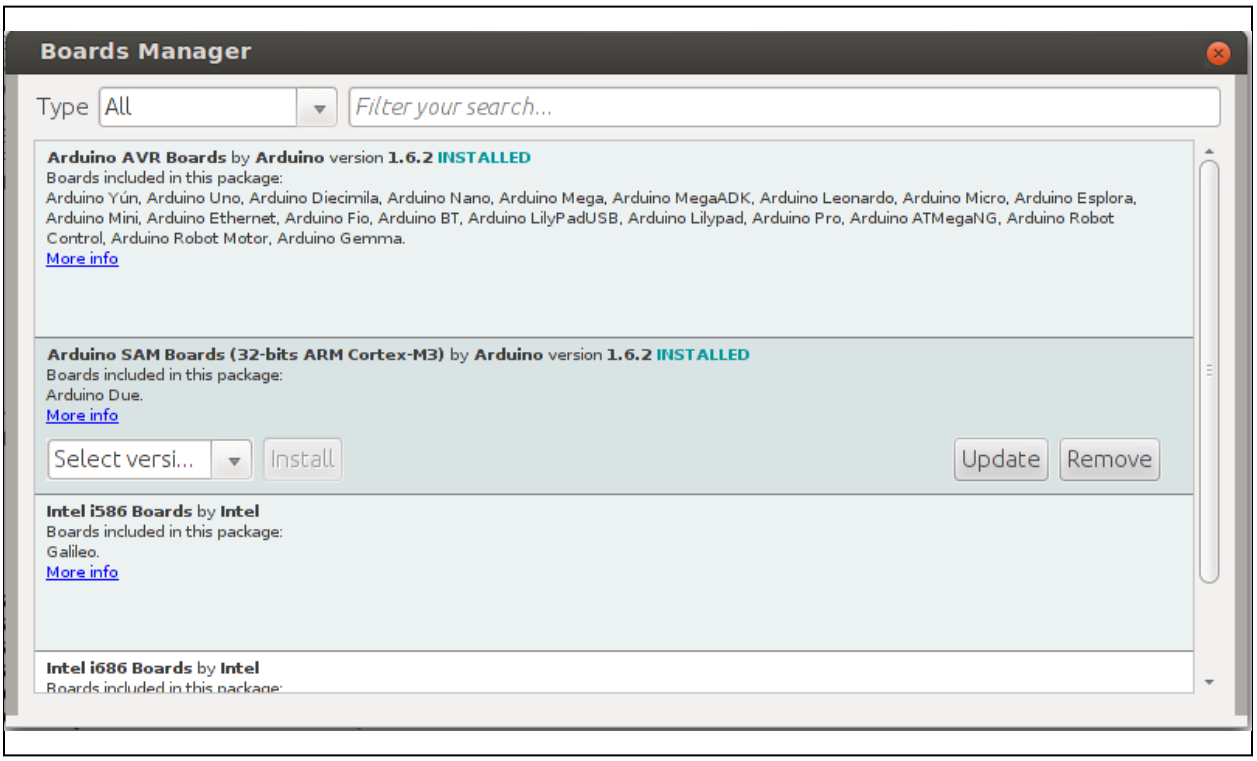
3.2.2 Arduino SAM Board のインストール

インストール直後の Arduino Sketch は、Arduino Due に対応していない可能性があります。Due に対応しているかどうかは、“Tools->Board->Boards Manager...”をクリックし、確認して下さい。



Arduino Sketch: Boards Manager のロード

これにより、下図の Boards Manager ウィンドウが表示されます。



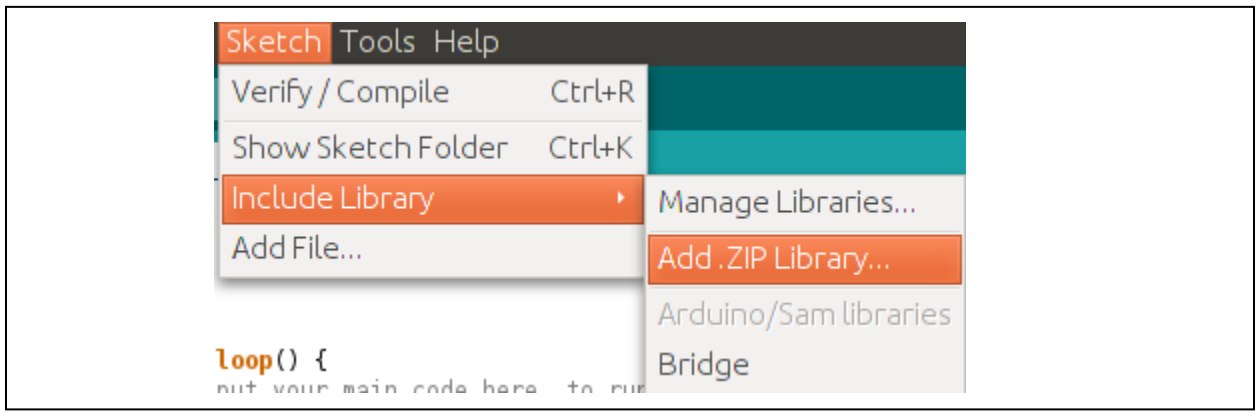
Arduino Sketch: SAM Boards Package のロード

“Arduino SAM Boards (32-bits ARM Cortex-M3)”ボードパッケージがインストールされていることを確認します。 もしインストールされていない場合、使用している Sketch IDE のバージョンに合ったものをインストールします。SAM Boards ボードパッケージがインストールされたら次のステップに進みます。

3.2.3 グラフィックスライブラリのインストール

グラフィックスライブラリは.zip のアーカイブで用意されています。この.zip ファイルには example sketch のフォルダとグラフィックスライブラリの.zip ファイルが含まれています。開発プラットホームの適切な場所で解凍して下さい。

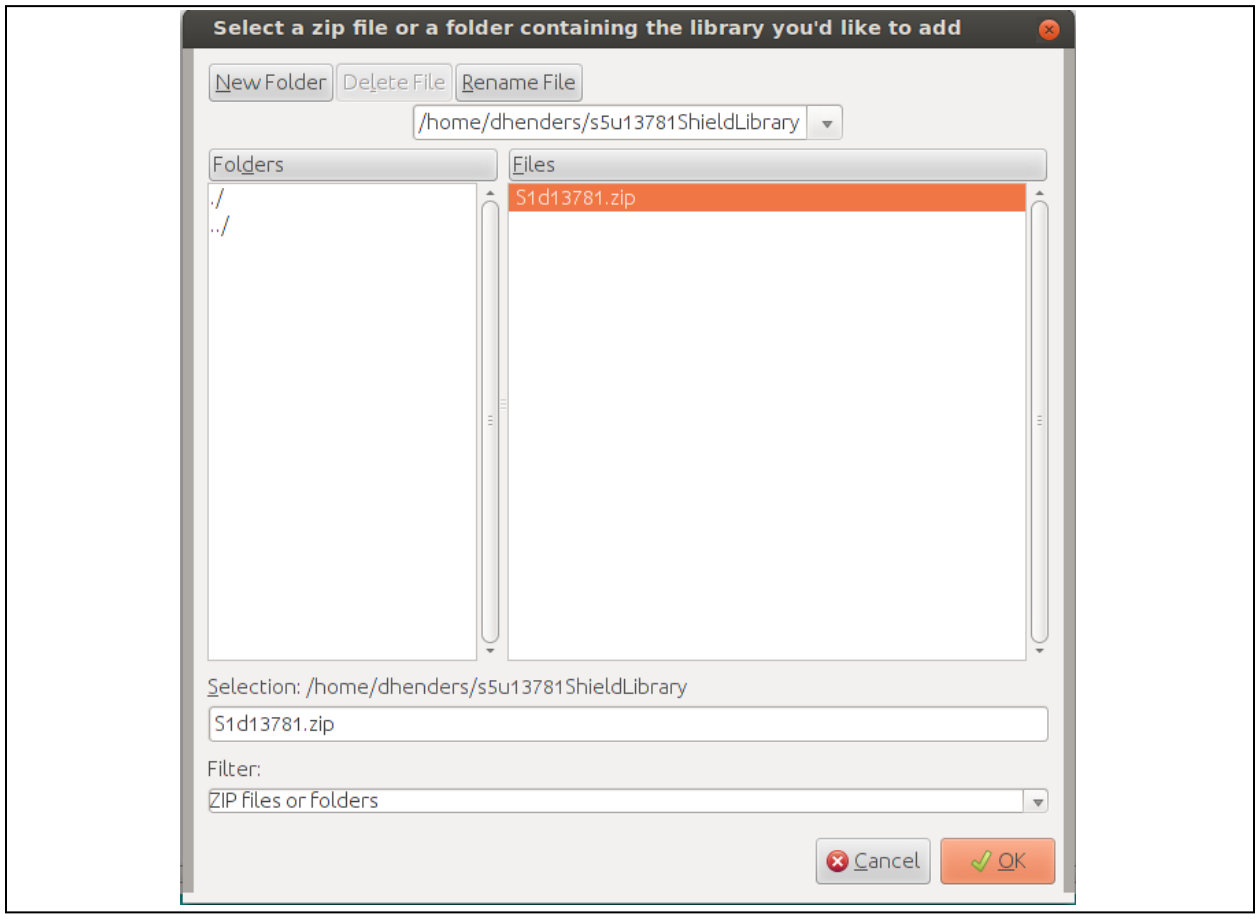
Sketch IDE には、グラフィックスライブラリを直接インポート可能です。Sketch メニューで “Sketch->Include Library->Add .ZIP Library...” をクリックして下さい。



Arduino Sketch: .ZIP Library の追加

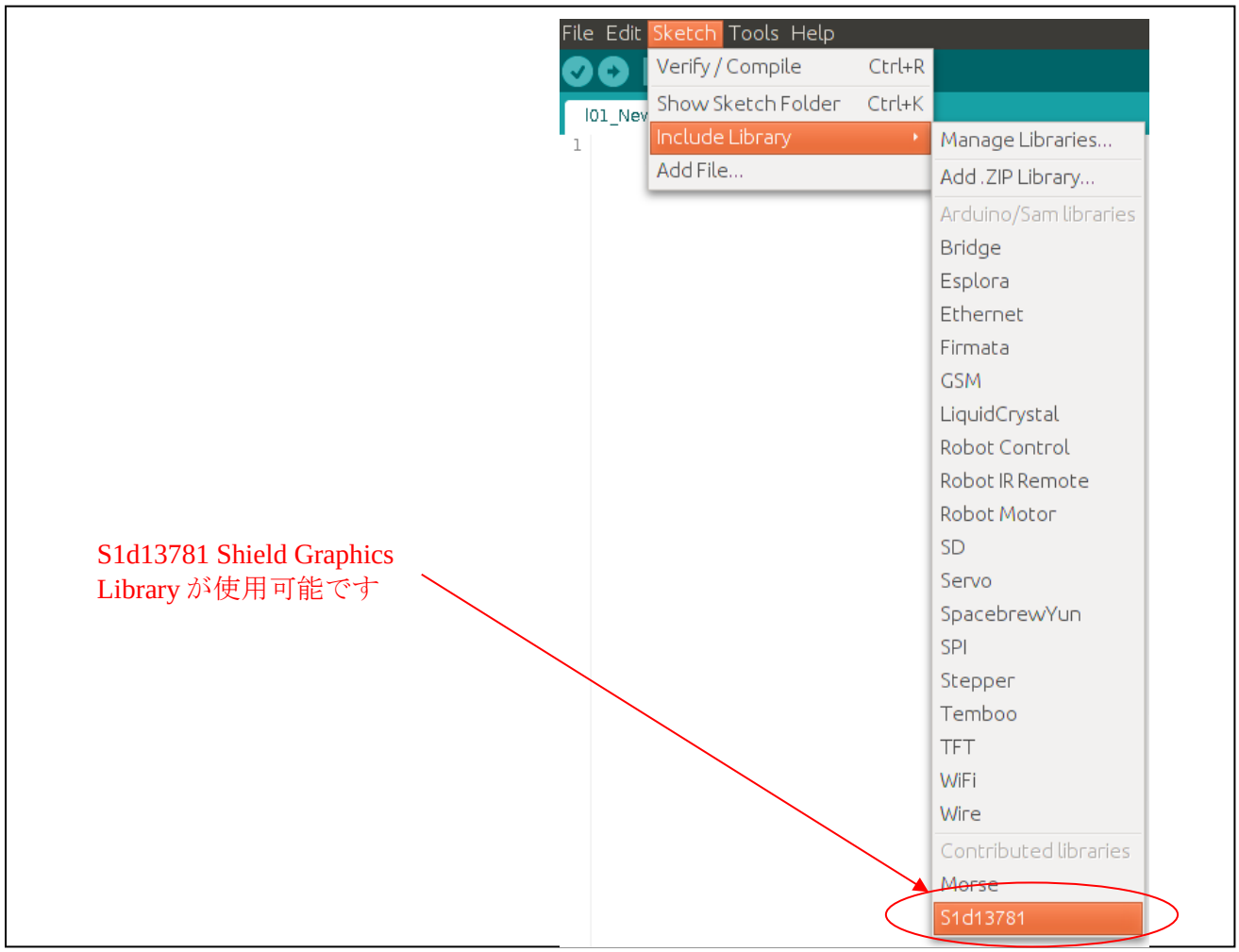
これにより、セクションウィンドウが開きます。“S1d13781.zip”ファイルを解凍した場所まで移動して選択して OK をクリックして下さい。これによってグラフィックスライブラリが Sketch IDE にインストールされます。

注:
既存ライブラリのアップデート版や変更版をインストールする際は、新ライブラリを再インストールする前に、既存のライブラリファイルを含むフォルダを“Arduino/libraries” フォルダから消去しておく必要があります。



Arduino Sketch: S5U13781R01C100 Shield Graphics Library のロード

グラフィックライブラリがインストールされたことを確認するため、”Sketch->Include Library”をクリックします。使用可能なライブラリリストの一番下に “S1d13781”のエントリがあればインストールされています。

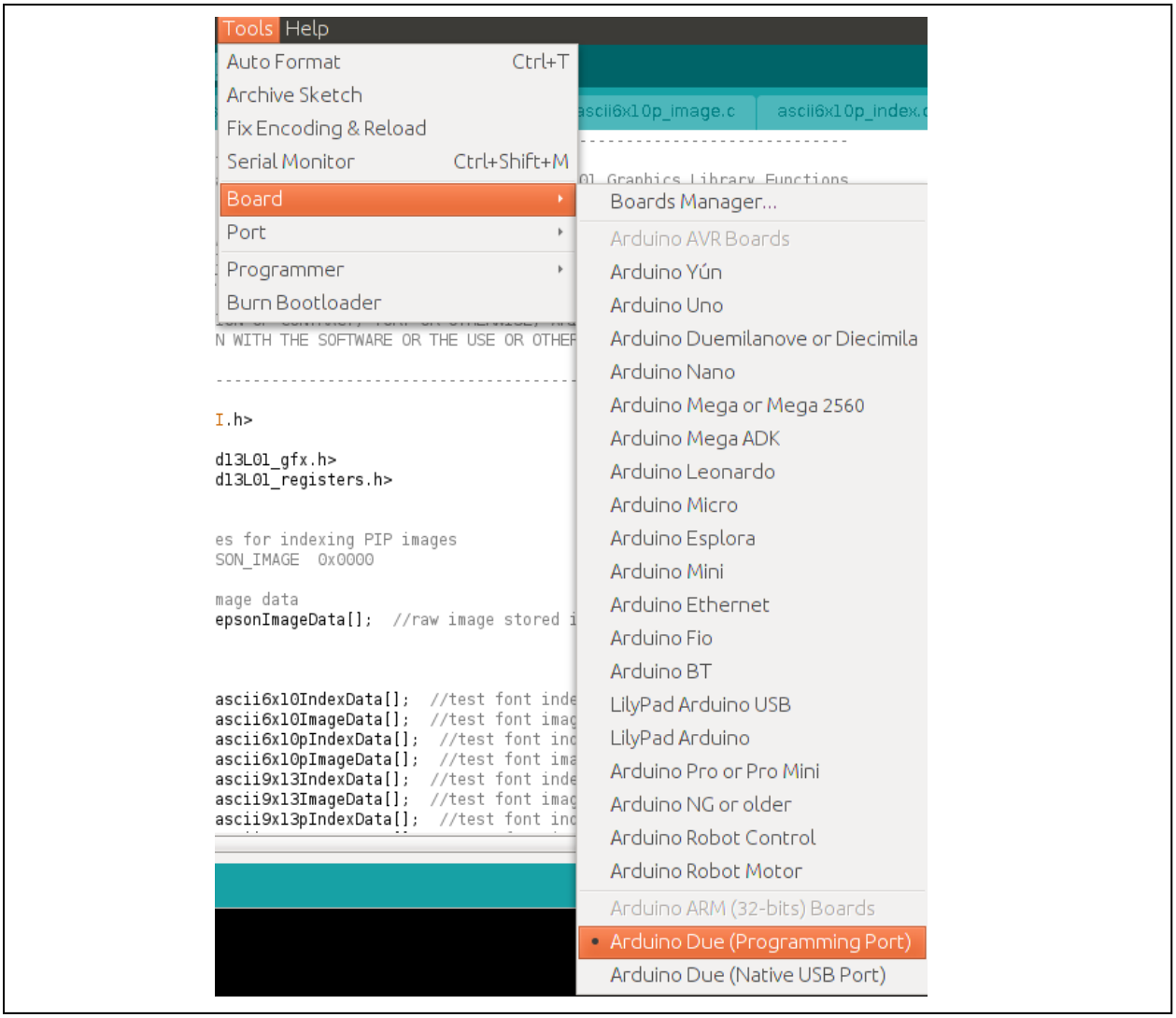


Arduino Sketch: S1d13781 Shield Graphics Library ロード状況の確認

3.2.4 Example Sketch のコンパイルと実行

Example sketch を実行する前に、Sketch IDE でボードとポートの設定が必要です。ボードとポートの設定では、Sketch IDE に、どの Arduino 製品が使われていて、どのように通信を行うかの設定を行います。

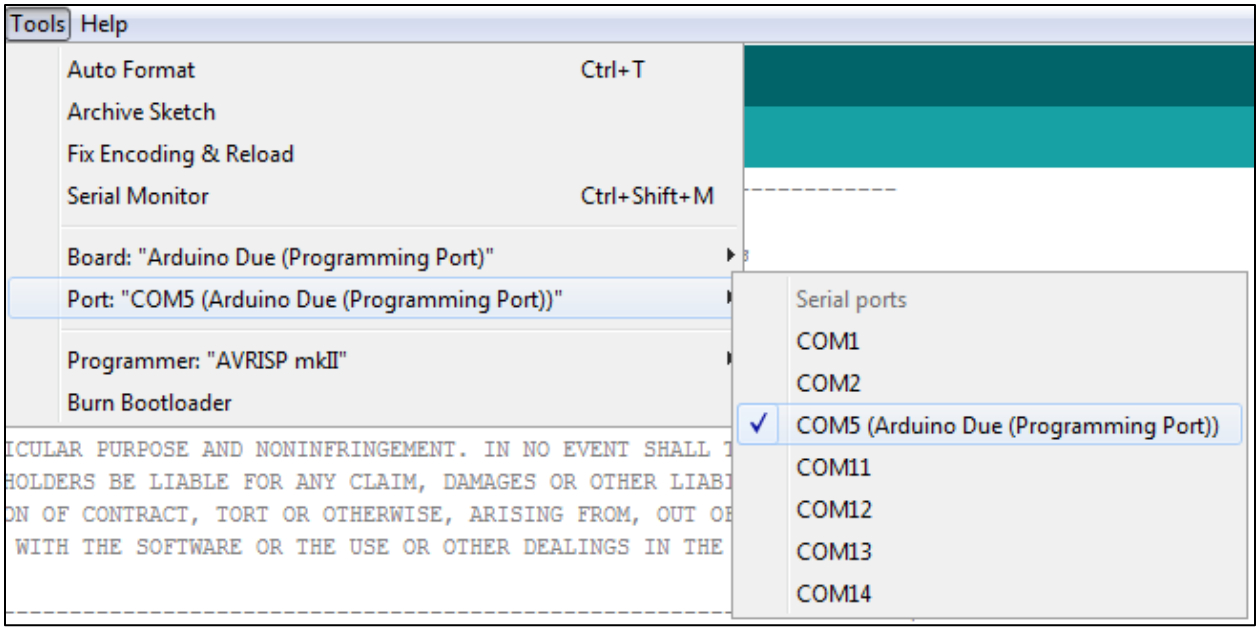
ボードの設定を Arduino Due にするには、下図のとおり、“Tools->Board->Arduino Due (Programming Port)”をクリックします。



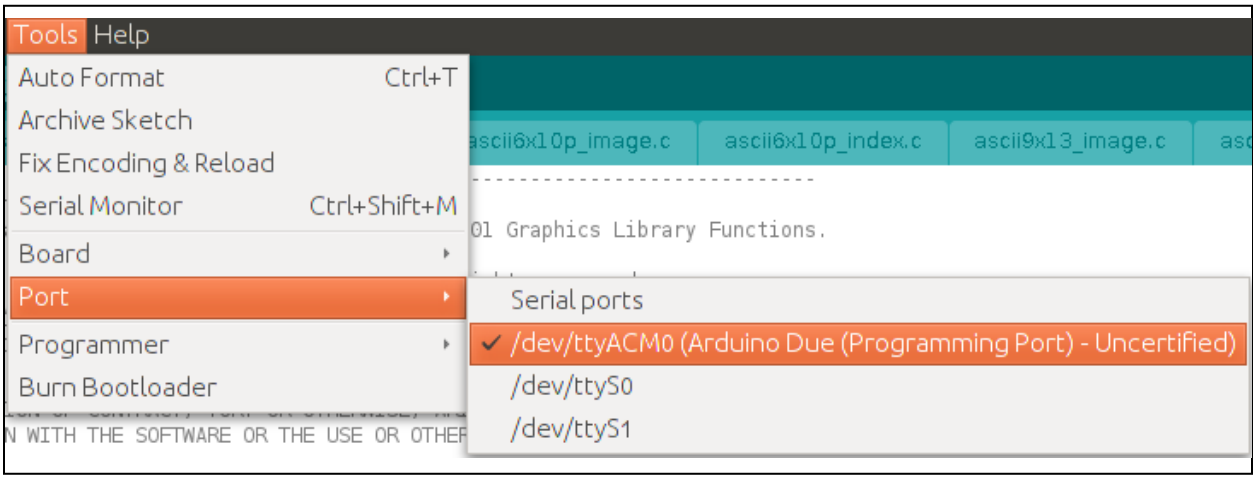
Arduino Sketch: ボードの設定

ポートを Arduino Due に設定するには、下図のとおり、 “Tools->Port->Arduino Due (Programming Port)” をクリックします。

Arduino Due のポートのロケーションは開発環境の OS によって異なる場合があります。USB ポート接続に関する情報については、Arduino の web サイト www.arduino.cc/を参照下さい。



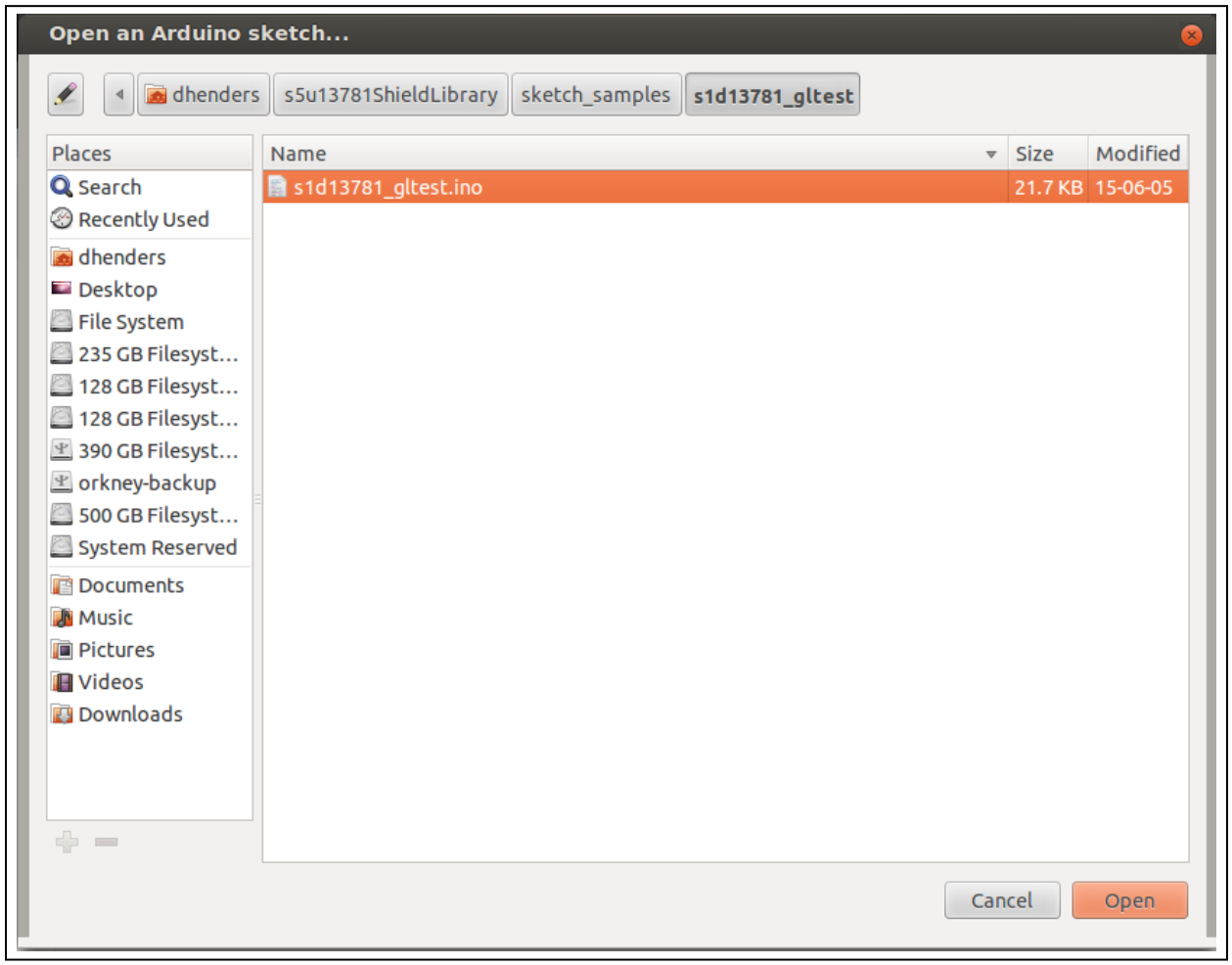
Arduino Sketch: ポートの設定(Windows の例)



Arduino Sketch: ポートの設定(Linux の例)

以上で、グラフィクスライブラリがインストールされ、example sketch の実行が可能です。“781_gltest” example sketch を開くためには、“File->Open…”をクリックし、example sketch が解凍されているフォルダを指定します。“781_gltest.ino”を選択し、Open をクリックします。これにより、LCD コントローラ S1D13781 とグラフィックスライブラリの複数の機能を実行する example sketch が開きます。

注:
デフォルトは S5U13781R01C100 シールド TFT ボードに Newhaven Display 社製 480x272 LCD パネルが接続された場合の設定になっています。

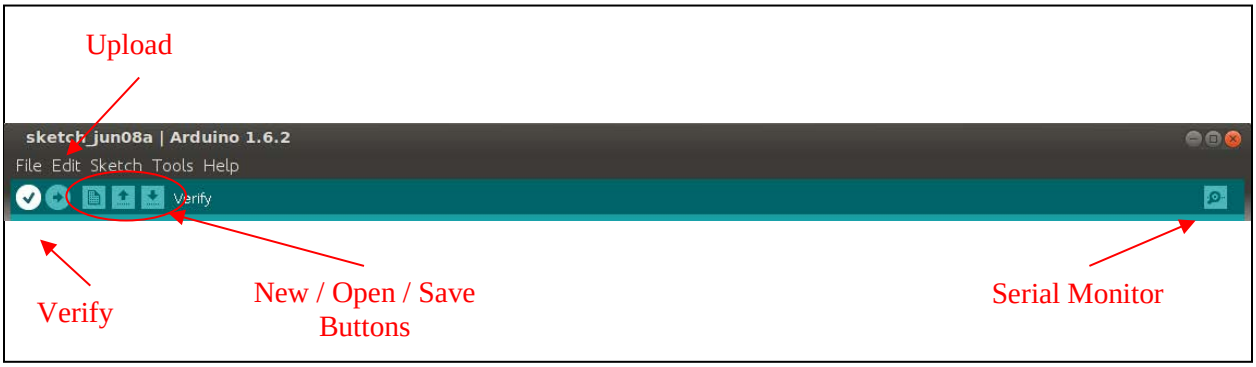


Arduino Sketch: Example Sketch のロード

Example sketch がロードされると、コンパイルと Arduino Due へのアップロードが可能です。もし、単に変更箇所の確認や新しいコードのエラーのチェックを行う場合は、Sketch Toolbar の “Verify”をクリックして下さい。Arduino Due にアプリケーションをアップロードし、実行する場合は、“Upload”ボタンをクリックして下さい。Upload はアプリケーションコードを検証して、必要なバイナリイメージを Arduino Due にアップロードします。

注：
Sketch の“Board”および “Port”設定が適切にされていない場合には、アップロードができません。これらの設定についてはこのセクションの最初の方の説明を参照下さい。

下図は Sketch toolbar のボタンの配置を示します。



Arduino Sketch: 重要な Toolbar 機能

“Verify” と “Upload”のオプションは、その結果やエラーメッセージを Sketch IDE ウィンドウに表示します。Arduino Due へのアップロードが問題なく完了すると、781_gltest デモを開始し、グラフィックスライブラリの複数の描画機能を実行します。

4 グラフィックスライブラリの Sketch での使用

4.1.1 既存の Sketch の変更

グラフィックスライブラリには、以下のような、いくつかの異なるコンセプトのデモを行うための example sketch が含まれています。

- 781_gltest.ino – グラフィクスライブラリで可能ないくつかの機能のデモ
- 781_RegisterAccessExample – S1D13781 のレジスタを直接 Read/Write するデモ
- 781_MemoryAccessExample – S1D13781 のメモリを直接 Read/Write するデモ

Sketch を変更し、新しいアイデアを試すためには、Sketch loop()ファンクションにグラフィックスライブラリの機能をコールする新しい行を追加します。例えば、LCD パネル上で緑色の線を 20,20 から 100,100 の座標まで引くときは、次の内容の行を追加します。

```
result = lcdc.drawLine( S1d13781_gfx::window_Main,20,20,100,100,0x0000FF00 );
sleep(2000);
```

上記の例で、

- 結果は drawLine() 関数からの戻り値です。多くのグラフィックスライブラリ関数は要求値あるいはエラーのチェックに使われる値を返します。
- S1d13781_gfx のインスタンス lcdc は example sketch で使われます。
- drawLine は呼び出されるメソッドです。
- drawLine に渡されるパラメータは、対象のウィンドウ、線分の終点、線分の色を決めます。
- 次の関数を呼び出す前に sleep(2000) を入れることで、関数の効果を確認することができます。

グラフィックスライブラリとして用意されているメソッドについては、ライブラリリファレンスの章をご確認下さい。

4.1.2 新規 Sketch の作成

新規 sketch を作成する際には、基本テンプレートに特定のエレメントを追加する必要があります。下記コードは以下に説明されている追加を示しています。

```
/*-----------------------------------------------------------------
 * new_sketch.ino
 * Example of a new sketch using the S5U13781R01C100 Graphics
 * Library Functions.
 *-----------------------------------------------------------------*/

#include <SPI.h>

#include <S1d13781_gfx.h>
#include <S1d13781_registers.h>


//create an instance of S1d13781 for us to work with
S1d13781_gfx lcdc;

void setup() {
  //start serial for serial monitor
```

```
  Serial.begin(9600);


  //start the S1d13781 library
  lcdc.begin();

  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

新規 sketch でグラフィックスライブラリを使用する場合、以下の追加が必要です。

- #include <SPI.h> - SPI ヘッダーは Arduino Due と S5U13781R01C100 シールド TFT ボードとのインタフェースが SPI のため必要です。
- #include <S1d13781_gfx.h> - このヘッダは、S5U13781R01C100 Shield のライブラリクラスをインクルードします。
- #include <S1d13781_registers.h> - このヘッダは S1D13781 レジスタの#define をインクルードします。S1D13781 のレジスタをアクセスする際、定義されていないレジスタに意図しないアクセスをしないよう、ここで定義された定数を使用します。
- “S1d13781_gfx lcdc;” – この行は、S1d13781_gfx クラスのインスタンスを作成します。lcdc が example sketch に使用されていますが、これはアプリケーションに合わせて自由に変更できます。
- “Serial.begin(9600);” – この行はオプションですが、アプリケーションからシリアルモニターにメッセージを送ることができ、デバッグ時に便利です(「シリアルモニタの使用」を参照下さい)。
- “lcdc.begin();” – この行は S5U13781R01C100 ライブラリの開始を示し、下記を実行します。
    1. S5U13781R01C100 シールド TFT ボードが使用する SPI インタフェースの設定
    2. S1d13781_init.h の設定に従ったレジスタの初期化
    3. S1D13781 をパワーオン状態(レディ状態)に留める

4.1.3 シリアルモニタの使用

Arduino Due がプログラミングポートを通して接続されている場合、シリアルモニター機能が利用可能です。シリアルモニターは Sketch toolbar の“Serial Monitor”ボタンを押すか、Sketch menu で“Tools->Serial Monitor”をクリックすることで開始可能です。

シリアルモニターは Arduino Due で実行しているアプリケーションからの情報やメッセージを Sketch IDE に送信したり表示する事が可能です。これは、新規または変更したアプリケーションコードをデバッグする場合に有用です。グラフィックスライブラリに含まれている example sketch はシリアルモニターに情報を出力する手段の例を示します。

シリアルクラスおよびシリアルモニターを使用する際の詳細な情報については下記サイトで Aruduino language reference を参照して下さい。

www.arduino.cc/en/Reference/

4.1.4 グラフィックスライブラリでのフォントの使用

グラフィックスライブラリはプログラマブルなフォントを使ったテキスト描画をサポートしています。フォントは下記 2 つのコンポーネントに依存します。

- .pbm グラフィックスファイル(バイナリ)として保存されている 1bpp 画像ファイル
- .pfi ファイル(バイナリ)として保存されているポータブルフォントインデックス

グラフィックスライブラリパッケージは複数のサンプルフォントを提供し、.pbm 画像ファイルと.pfi インデックスファイルをバイナリで同梱しています。

注:
.pfi ファイルはサブフォルダに参照用のテキストファイルとして保存されています。カスタムフォントの作成に有用な、フォントインデックスファイルの詳細情報については、サンプルフォルダの Readme.txt を参照下さい。

パッケージに含まれている全てのサンプルフォントは、Epson で作成したものですのでご自由に変更あるいはそのままご使用下さい。

- Ascii4x6
- Ascii4x6p (プロポーショナルフォント)
- Ascii6x10
- Ascii6x10p (プロポーショナルフォント)
- Ascii7x11
- Ascii7x11p (プロポーショナルフォント)
- Ascii9x13
- Ascii9x13p (プロポーショナルフォント)
- AsciiCaps4x6
- AsciiCaps4x6p (プロポーショナルフォント)
- Latin6x10
- Latin6x10p (プロポーショナルフォント)
- LineDraw6x10 (line art ボタンに適した line draw グラフィックスを含む)

Sketch アプリケーションでフォントを使用するとき、所望のフォントをアプリケーションに与える簡単な方法として、バイナリファイルの.pbm および.pfi ファイルをシンプルなバイト配列に変換する方法があります。

もし、配列が”C”ソースで保存されている場合、Sketch アプリケーションフォルダにコピーして、必要に応じて S1d13781_gfx::createFont() メソッドに渡すことで Sketch アプリケーションから参照可能です。

一例として、複数のサンプルフォントを含む“781_gltest.ino”があります。外部のバイト配列を使用する場合、以下の行を Sketch アプリケーションに追加して下さい。

```
//test fonts
extern byte ascii9x13IndexData[];  //test font index data
extern byte ascii9x13ImageData[];  //test font image data
```

そして、フォントを作るために、各々の配列のバイト数およびデータを送ります。

```
testfont = lcdc.createFont(ascii9x13ImageData, 1453, ascii9x13IndexData, 498);
```

一旦フォントが作成されると、drawText()や drawMultiLineText()などのメソッドで使うことができます。フォントが必要でなくなった場合は、freeFont() メソッドを呼び出して、フォントのリソースを開放して下さい。

フォントメソッドの詳細情報については、ライブラリリファレンスの章を参照下さい。

4.1.5 グラフィックスライブラリを使用した画像表示

Arduino ボードはメモリおよびフラッシュ容量が小さいため、グラフィックスライブラリはビットマップ画像を扱うための関数を持っていません。

しかし、小さなビットマップ画像を sketch プログラムの一部として持たせることができ、TFT パネルに表示することができます。十分なメモリが無いので、大きな画像や、多くの小さな画像を置くことはできません。一つの方法として、次のように外部のバイトストリームとして画像を取り込むことが挙げられます。example sketch の s1d13781_gltest.ino がこの方法のデモです。

1. 画像を raw データで保存可能な画像エディタを使って準備します。オープンソースの画像エディタ Gimp がその一例です。www.gimp.org を参照下さい。
2. 画像を “Raw Data” で保存します(Gimp では、“Export As…”コマンドで可能です)。
3. “Raw Data”ファイルを C-style バイトストリームに変換します。これは、グラフィックスライブラリの“extras”フォルダにある bin2c ツールで可能です。bin2c ツールのソースファイルを同梱していますので、OS に応じてコンパイル後、コマンドラインで下記コマンドを入力して下さい。

   bin2c file1 file2

   例えば、次のように入力して下さい。

   bin2c imagefile.data imagefile.c

4. テキストエディタを使い、.cファイルのバイトストリームに変数の型と名前を追加します。例えば、下記、赤文字の行を追加します。

```
unsigned char imageData[] = {
0xE6,0xE6,0xE9,0xE9,0xE9,0xFB,0xFB,0xFB,0xFB,0xFB,0xFB,0xFC,0xFC,0xFC,0xF5,0xF5,
    *
    *
    *
0xF5,0xFC,0xFC,0xFC,0xFC,0xFC,0xFC,0xF9,0xF9,0xF9,0xF5,0xF5,0xF5,0xFD,0xFD,0xFD,
};
```

5. Sketch が保存されているフォルダに、 imagefile.c ファイルをコピーし、sketch に外部バイトストリームとしてデータを追加します。例えば、次のようになります。

```
//external image data
extern byte imageData[];  //raw image stored in imagefile.c
```

6. S1D13781 にデータを送る関数を追加します。例えば、次のようになります。

```
void drawImageAtXy( int x, int y, int width, int height)
{
```

```
    unsigned int vramAddress = lcdc.lcdGetStartAddress();  //video memory address
    word stride = lcdc.lcdGetStride();  //number of bytes in line
    word bytesPerPixel = lcdc.lcdGetBytesPerPixel();
    word imageStride;
    unsigned int offset;
    unsigned int i,j;  //loop vars

    //calculate starting offset
    offset = (y*stride) + (x*bytesPerPixel);

    //calculate image stride
    imageStride = (width * bytesPerPixel);

    //adjust address with offset
    vramAddress = vramAddress + offset;

    //write image to Main window
      for (i=0; i<height; i++){
        for (j=0; j<(width*bytesPerPixel); j+=3){
          lcdc.memWriteByte(vramAddress+j, imageData[(imageStride*i)+j+2]);
          lcdc.memWriteByte(vramAddress+j+1, imageData [(imageStride*i)+j+1]);
          lcdc.memWriteByte(vramAddress+j+2, imageData [(imageStride*i)+j]);
        }
        vramAddress = vramAddress + stride;
      }
  }
```

7. Sketch の loop()ルーチンに画像を S1D13781 のメモリに送る drawImageAtXy() ファンクションコールを追加します。

詳細な例については、s1d13781_gltest.ino example sketch のソースコードを参照下さい。

5　グラフィックスライブラリの解説

グラフィックスライブラリは、S1d13781 と S1d13781_gfx の 2 クラスに分かれた C++のメソッドによって構成されています。これらが Arduino Sketch IDE にインテグレートされると、ハードウェアアクセスおよび S5U13781R01C100 シールド TFT ボードに接続された LCD パネル上にすぐに画像やテキストを描画可能するシンプルな描画ルーチンを提供します。

5.1.1　ライブラリ構成

グラフィックスライブラリは下記ファイルで構成されています。

- S1d13781.h – S1d13781 クラスのヘッダ。グラフィックスライブラリのハードウェア指向の関数についての全体像を把握できます。またこのファイルは、S5U13781R01C100 シールド TFT ボードと Arduino Due を接続する SPI インタフェースを設定する複数の定数を含んでいます。
- S1d13781.cpp – S1d13781 クラスのソースファイル。LCD コントローラ S1D13781 のハードウェアレベルの関数にアクセスができるようにするメソッドのソースを含んでいます。メモリやレジスタへの直接アクセス、S1D13781 の初期化、および S5U13781R01C100 シールド TFT ボードと Arduino Due を接続する SPI の設定をする関数を含みます。
- S1d13781_gfx.h – S1d13781_gfx クラスのヘッダファイルです。グラフィックスライブラリでインプリメントされている、描画やテキストのメソッドの全体像を 把握できます。
- S1d13781_gfx.cpp – S1d13781_gfx クラスのソースファイルです。グラフィックスライブラリに用意されている描画やテキストのメソッドのソースファイルを含んでいます。
- S1d13781_init.h –このヘッダファイルは、S1D13781 のハードウェアレジスタの初期化に使われる値を持つストラクチャを含んでいます。
- S1d13781_registers.h – このヘッダファイルは S1D13781 ハードウェアレジスタ用の #defines を含みます。S1D13781 のレジスタをアクセスする際、定義されていないレジスタに意図しないアクセスをしないよう、ここで定義された定数を使用します。
- keywords.txt - Arduino Sketch IDE でのライブラリサポートに必要なファイルです。どのような新規クラスまたはメソッドもこのファイルに追加する必要があります。keywords.txt の詳細については、www.arduino.cc/ の Library Tutorial を参照下さい。

5.1.2　グラフィックスライブラリの変更

ユーザーによるカスタマイズや変更ができるように、グラフィックスライブラリの全てのソースコードを提供しています。グラフィックスライブラリのソースは Sketch IDE 内で変更されることを想定していません。このため外部のコード編集ツールを使用する必要があります。

Sketch IDE にグラフィックスライブラリがインストールされると、“Arduino/libraries/S1d13781” フォルダの中で変更が直接可能です。または、“off-tree”で変更でき、.ZIP ライブラリとして再インストール可能です。しかし、再インストールの場合、アップデート後のグラフィックスライブラリを再インストール前に、S1d13781 ライブラリを“Arduino/libraries”フォルダから削除する必要があります。

Sketch IDE のフルサポートのため、新規クラスまたはメソッドがグラフィックスライブラリに追加される場合は、グラフィックスライブラリの keyword.txt ファイルにこれらを追加する必要

があります。keywords.txt の詳細については、www.arduino.cc/ にある Library Tutorial を参照下さい。

5.1.3 S1D13781 初期設定値のカスタマイズ

デフォルトの LCD パネル以外の物を使用する場合など、LCD コントローラ S1D13781 のレジスタ初期設定をカスタマイズする必要がある場合は、S1d13781_init.h ファイルのストラクチャをアップデートすることにより可能です。S1d13781_init.h ファイルはレジスタ値および S1D13781 を立ち上げる際のシーケンスの情報を含みます。

vdc.epson.com から入手できる、S1D13781 の Windows ユーティリティ“781cfg.exe”を使うことで、新規設定を作成することができます。“781cfg”で S1D13781 の所望の設定を選択し、“Export…”オプションを使って“C Header File for S1D13781 Generic Drivers”を選択します。これにより、S1d13781_init.h ファイルをアップデートするための値を含む S1D13781.h が出力されます。

“781cfg”で作成された S1D13781.h ファイルを開き、Index / Value の組み合わせを variable_name[] array からグラフィックスライブラリファイル S1d13781_init.h の regInitValues[] array にコピーします。

例えば、以下のようになります。

S1D13781.h generated by the “781cfg” から作成された S1D13781.h から、下記赤文字部分の値をコピーします。

```
#define S1D_INSTANTIATE_REGISTERS(scope_prefix,variable_name)  \
  scope_prefix S1D_REGS variable_name[] =  \
  {  \
    { 0x06,             0x0100 },   /* Software Reset Register */  \
    { S1D_REGDELAYON,   0x2710 },   /* LCD Panel Power On Delay (in ms) */  \
    { 0x04,             0x0000 },   /* Power Save Register  */  \
      *
      *
      *
    { 0xD2,             0x0001 },   /* GPIO Status / Control Register */  \
    { 0xD4,             0x0000 }    /* GPIO Pull-Down Control Register */  \
  }
```

コピー先は、グラフィックスライブラリファイル S1d13781_init.h の“regData regInitValues[] = {}” array です。これにより、Index / Value ペアは、新しい設定用にアップデートされたレジスタ初期設定値になります。

新規設定値が適切でない場合、S1D13781 は初期化されないまたはパネルの表示が正常に行われないことがあります。もしそういう状況になった場合は、“781cfg”を使って再度確認して下さい。S1D13781 レジスタ情報の詳細については、vdc.epson.com の S1D13781 Hardware Functional Specification (ドキュメント番号 X94A-A-001-xx)を参照下さい。

6　ライブラリリファレンス

グラフィックスライブラリは、下記 2 つのクラスで構成されています。

S1d13781 –S5U13781R01C100 シールド TFT ボードのハードウェアレベルをサポートし、Arduino Due の SPI インタフェースに接続するためのベースクラス

S1d13781_gfx – グラフィックス描画とテキスト表示の機能を提供するクラス

6.1　S1d13781 クラス

S1d13781 クラスは以下のパブリックメソッドを提供します。プライベートメソッドはこのドキュメントには記載していませんが、ソースコードに記載しています。

6.1.1　S1d13781()

これはクラスのコンストラクタです。

6.1.2　S1d13781::begin()

このメソッドは S5U13781R01C100 シールド TFT ボードで使われる SPI インタフェースをセットアップするのに一回実行する必要があり、これによりレジスタの設定を行います。

パラメータ -　なし。

戻り値:

- なし。

6.1.3　S1d13781::regWrite()

1 ワード (16-bit unsigned int)を S1D13781 のレジスタに書き込むメソッドです。引数 regIndex は、定義されていないレジスタ S1d13781_registers.h ファイルの中で事前に定義されたレジスタ名を使用する必要があり、これによってミスアライメントまたは無効な(存在しない)レジスタへのアクセスを防ぎます。

注:
全てのレジスタを常時書き込みできるわけではありません。レジスタによっては、S1D13781 が NMM(詳細はテクニカルマニュアルを参照)状態では書き込めません。

パラメータ -　regIndex　　書き込み先のレジスタインデックス。

パラメータ -　regValue　　レジスタに書き込むデータ。

戻り値:

- なし。

6.1.4 S1d13781::regRead()

1 ワード (16-bit unsigned int)を S1D13781 のレジスタから読み出すメソッドです。引数 regIndex は、S1d13781_registers.h ファイルの中で事前に定義されたレジスタ名を使用する必要があり、これによってミスアライメントまたは無効な(存在しない)レジスタへのアクセスを防ぎます。

注:
全てのレジスタを常時読み出しできるわけではありません。レジスタによっては、S1D13781 が PSM0(詳細はテクニカルマニュアルを参照)状態では書き込めません。

パラメータ - regIndex　　読み出すレジスタのインデックス(オフセット)。

戻り値:

- 指定した S1D13781 のレジスタの値を返します。

6.1.5 S1d13781::regModify()

ビットマスクを使って S1D13781 レジスタの内容を変更するメソッド。使い方は以下の通りです。

1. 現在のレジスタ値を読む。
2. ビットマスク clearBits を使って選択したビットをクリアする。
3. ビットマスク setBits を使って選択したビットをセットする。
4. レジスタに値を書き戻す。

この関数は、”1”を書くとステータスクリアするなど、読み出しと書き込みで異なる効果を持つレジスタのどのビットに対しても使用不可です。

通常、この関数は、レジスタが、対象のビットフィールドのみでなく、他のビットも含む場合に、そのビットフィールドに新しい値を設定するときに使います。もし、レジスタが 1 つのビットフィールドを含み、他のビットは使われていない場合は、より効率的な regRead/regWrite() を使用して下さい。

パラメータ - regIndex　　変更するレジスタインデックス。

パラメータ - clearBits　　クリア(‘0’に設定)されるレジスタビットのビットマスク。このマスクで’1’になっているビットは、レジスタ値がクリアされます。

パラメータ - setBits　　セット(‘1’に設定)されるレジスタビットのビットマスク。

戻り値:

- レジスタに書かれた値を返します。

6.1.6 S1d13781::regSetBits()

ビットマスクを使って S1D13781 の特定のビットをセットするメソッド。使い方は以下の通りです。

1. 現在のレジスタ値を読む。
2. ビットマスク setBits を使って選択したビットをセットする。
3. レジスタに値を書き戻す。

通常、この関数は、レジスタが、1 つのビットだけでなく、他の無関係なビットを含む場合に 1 つのビットをセットするときに使います。レジスタの全てのビットをセットする場合には regWrite()を使用して下さい。

パラメータ - regIndex　　対象のレジスタインデックス。

パラメータ - setBits　　セット('1'に設定)されるレジスタビットのビットマスク。

戻り値:

- アップデートされたレジスタ値を返します。

6.1.7 S1d13781::regClearBits()

ビットマスクを使用し、S1D13781 の特定のビットをクリアするメソッド。使い方は以下の通りです。

1. 現在のレジスタ値を読む。
2. ビットマスク clearBits を使って選択したビットをクリアする。
3. レジスタに値を書き戻す。

通常、この関数は、レジスタが、1 つのビットだけでなく、他の無関係なビットを含む場合に 1 つのビットをクリアするときに使います。レジスタの全てのビットをクリアする場合には regWrite()を使用して下さい。

パラメータ - regIndex　　対象のレジスタインデックス。

パラメータ - clearBits　　クリア('0'に設定)されるレジスタビットのビットマスク。

このマスクで'1'になっているビットは、レジスタ値がクリアされます。

戻り値:

- アップデートされたレジスタ値を返します。

6.1.8 S1d13781::memWriteByte()

S1D13781 のビデオメモリの特定のアドレスオフセットにバイト値(8-bit)を書くメソッド。

注:
メモリアドレスオフセット値は有効なメモリ空間内 (0x00000000~0x0005FFFF)である必要があります。

パラメータ - memAddress　アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

パラメータ - memValue　　ビデオメモリに書くバイト値 (8-bit)。

戻り値:

- なし。

6.1.9 S1d13781::memReadByte()

S1D13781 のビデオメモリの特定のアドレスオフセットからバイト値(8-bit)を読み出すメソッドです。

注:
メモリアドレスオフセットは有効なメモリ空間内にあることが必要です。
(0x00000000~0x0005FFFF)。

パラメータ - memAddress アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

戻り値:

- 戻り値は設定したアドレスからのバイト値 (8-bit)。

6.1.10 S1d13781::memWriteWord()

S1D13781 のビデオメモリの特定のアドレスオフセットにワード値(16-bit)を書くメソッドです。

注:
1. メモリアドレスオフセット値は有効なメモリ空間内 (0x00000000~0x0005FFFF)である必要があります。
2. ワード書き込みの際は、メモリアドレスオフセットは適切なメモリアライメントとなるように偶数でなければなりません。

パラメータ - memAddress アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

パラメータ - memValue ビデオメモリに書くワード値 (16-bit)。

戻り値:

- なし。

6.1.11 S1d13781::memReadWord()

S1D13781 のビデオメモリの特定のアドレスオフセットからワード値(16-bit)を読み出すメソッドです。

注:
1. メモリアドレスオフセットは有効なメモリ空間内にあることが必要です
(0x00000000~0x0005FFFF)。
2. ワード読み出しの際は、メモリアドレスオフセットは適切なメモリアライメントとなるように偶数でなければなりません。

パラメータ - memAddress アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

戻り値:

- 指定したアドレスからのワード値 (16-bit)。

6.1.12 S1d13781::memBurstWriteBytes()

指定した数のバイト値(8-bit)を指定した S1D13781 のビデオメモリのアドレスオフセットにバースト書き込みするメソッドです。

注:
メモリアドレスオフセットは有効なメモリ空間内にあることが必要です
(0x00000000~0x0005FFFF)。

パラメータ - memAddress　アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

パラメータ - memValues　ビデオメモリに書くバイト値(8-bit)を含むバッファへのポインタ。もしポインタが NULL の場合、書き込みは行われない

パラメータ - count　バースト書き込みの回数。この値が 0 の場合は、書き込みは行われない。

戻り値:

- なし。

6.1.13 S1d13781::memBurstReadBytes()

指定した数のバイト値(8-bit)を指定した S1D13781 のビデオメモリのアドレスオフセットからバースト読み出しするメソッドです。

注:
メモリアドレスオフセットは有効なメモリ空間内にあることが必要です (0x00000000~0x0005FFFF)。

パラメータ - memAddress　アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

パラメータ - memValues　ビデオメモリから読み出されたバイト値(8-bit)を置くバッファへのポインタ。もしポインタが NULL の場合、読み出しは行われない。

パラメータ - count　バースト読み出しをする回数。この値が 0 の場合は読み出しは行われない。

戻り値:

- なし。

6.1.14 S1d13781::memBurstWriteWords()

指定した数のワード値(16-bit)を指定した S1D13781 のビデオメモリのアドレスオフセットにバースト書き込みするメソッドです。

注:
1. メモリアドレスオフセットは有効なメモリ空間内にあることが必要です (0x00000000~0x0005FFFF)。
2. ワード書き込みの際は、メモリアドレスオフセットは適切なメモリアライメントとなるように偶数でなければなりません。

パラメータ - memAddress　アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

パラメータ - memValues　ビデオメモリに書くワード値(16-bit)を含むバッファへのポインタ。もしポインタが NULL の場合、書き込みは行われない。

パラメータ- count　バースト書き込みするワード数。この値が 0 の場合は、書き込みは行われない。

戻り値:

- なし。

6.1.15 S1d13781::memBurstReadWords()

指定した数のワード値(16-bit)を指定した S1D13781 のビデオメモリのアドレスオフセットからバースト読み出しするメソッドです。

注:
1. メモリアドレスオフセットは有効なメモリ空間内にあることが必要です (0x00000000~0x0005FFFF)。
2. ワード読み出しの際は、メモリアドレスオフセットは適切なメモリアライメントとなるように偶数でなければなりません。

パラメータ - memAddress　アドレス 0x00000000 から始まるビデオメモリ内のオフセット。

パラメータ - memValues　ビデオメモリから読み出されたワード値(16-bit)を置くバッファへのポインタ。もしポインタが NULL の場合、読み出しは行われない。

パラメータ - count　バースト読み出しをするワード数。この値が 0 の場合は読み出しは行われない。

戻り値:

- なし。

6.1.16 S1d13781::lcdSetRotation()

メインレイヤーの回転を設定するメソッドです。

パラメータ - rotationDegrees　メインレイヤーの反時計回りの角度。許容角度は (0, 90, 180, 270)。

戻り値:

- なし。

6.1.17 S1d13781::lcdGetRotation()

メインレイヤーの現在の回転設定を返すメソッドです。

パラメータ - なし。

戻り値:

- 現在の反時計回りの角度設定値 (0, 90, 180, 270)。.

6.1.18 S1d13781::lcdSetColorDepth()

メインレイヤーの色深度を設定するメソッドです。

パラメータ - colorDepth　列挙型 S1d13781::imageDataFormat の値に応じたメインレイヤーの色深度:

- format_RGB_888
- format_RGB_565
- format_RGB_888LUT
- format_RGB_565LUT
- format_RGB_332LUT

戻り値:

- なし。

6.1.19 S1d13781::lcdGetColorDepth()

メインレイヤーの現在の色深度を返すメソッドです。

パラメータ　- なし。

戻り値:

- 現在の色深度 (設定可能な値):
  - format_RGB_888
  - format_RGB_565
  - format_RGB_888LUT
  - format_RGB_565LUT
  - format_RGB_332LUT

6.1.20 S1d13781::lcdGetBytesPerPixel()

メインレイヤーの色深度に基づくピクセル当たりのバイト数を返すメソッドです。

パラメータ - なし。

戻り値:

- ピクセル当たりのバイト数 (設定可能な値: 1, 2, または 3)。

6.1.21 S1d13781::lcdSetStartAddress()

メインレイヤーのメモリスタートアドレスを設定するメソッドです。スタートアドレスはメインレイヤーの画像が始まるディスプレイメモリ内のオフセットです。

注:
スタートアドレスは 32-bit アラインでなければなりません (4 で割り切れること)。

パラメータ - lcdStartAddress　メインレイヤーの画像が始まるディスプレイメモリ内のバイト単位でのオフセット。

戻り値:

- なし。

6.1.22 S1d13781::lcdGetStartAddress()

メインレイヤーのスタートアドレスを返すメソッドです。スタートアドレスはメインレイヤーの画像が始まるディスプレイメモリ内のオフセットです。

注:
スタートアドレスは 32-bit アラインでなければなりません (4 で割り切れること)。

パラメータ - なし。

戻り値:

- メインレイヤーのスタートアドレス。

6.1.23 S1d13781::lcdSetWidth()

LCD パネルの表示幅ピクセル数を返すメソッドです。この値は、回転設定に応じてメインレイヤーの幅/高さを決めるために使用されます。

注:
LCD の表示幅ピクセル数は 8 の倍数でなければなりません。

パラメータ - lcdWidth LCD パネルの表示幅ピクセル数。

戻り値:

- なし。

6.1.24 S1d13781::lcdGetWidth()

(回転に基づいて)メインレイヤーの幅を返すメソッドです。

注:
1. 0°および 180°のローテーションでは、メインレイヤーの幅は、LCD パネルの表示幅ピクセル数に基づきます。
2. 90°と 270°のローテーションでは、メインレイヤーの幅は、LCD パネルの 表示高さピクセル数に基づきます。

パラメータ - なし。

戻り値:

- メインレイヤー幅ピクセル数。

6.1.25 S1d13781::lcdSetHeight()

LCD パネルの表示高さピクセル数を設定するメソッドです。この値は、回転設定に応じてメインレイヤーの幅/高さを決めるために使用されます。

パラメータ - lcdHeight LCD パネルの表示高さピクセル数。

戻り値:

- なし。

6.1.26 S1d13781::lcdGetHeight()

(回転に基づいて)メインレイヤーの高さピクセル数を返すメソッドです。

注:
1. 0°および 180°のローテーションでは、メインレイヤーの幅は、LCD パネルの表示高さピクセル数に基づきます。
2. 90°と 270°のローテーションでは、メインレイヤーの幅は、LCD パネルの 表示幅ピクセル数に基づきます

パラメータ - なし。

戻り値:

- メインレイヤーの高さピクセル数。

6.1.27 S1d13781::lcdGetStride()

メインレイヤーの Stride をバイト単位で返すメソッドです。

Stride は、画像の 1 ライン(または行)当たりのバイト数。この値をディスプレイメモリ内のピクセルのアドレスに加えることで、そのピクセルの下のピクセルのアドレスを得られます。

パラメータ - なし。

戻り値:

- メインレイヤーの Stride(バイト単位)。

6.1.28 S1d13781::pipSetDisplayMode()

PIP ウィンドウの効果(ブリンク/フェード)を設定するメソッドです。

パラメータ - newEffect 列挙型 S1d13781::pipEffect の値に応じた PIP の効果:

- pipDisabled PIP の表示を直ぐに停止する
- pipNormal PIP を直ぐに表示する。PIP は現在のアルファブレンドモードで表示される。PIP を表示中にアルファブレンディング比を変えた場合は、次のフレームから有効になる。
- pipBlink1 PIP は、設定されたアルファブレンディングモードの表示と表示停止を繰り返す。
- pipBlink2 PIP はノーマルと反転を繰り返す。アルファブレンディング比は一定のまま維持する。
- pipFadeOut PIP を現在のアルファブレンド値から 0x0000(ブランク)までフェードアウトする。
- pipFadeIn PIP を 0x0000(ブランク)から設定したアルファブレンド値までフェードインする。
- pipContinous PIP の表示を、0x0000(ブランク)と設定したアルファブレンド値で繰り返す。

戻り値:

- なし。

6.1.29 S1d13781::pipGetDisplayMode()

PIP ウィンドウの現在の効果設定(ブリンク/フェード)を返すメソッドです。

パラメータ - なし。

戻り値:

- 下記 S1d13781::pipEffect 列挙型の値 による現在の効果設定。
  - pipDisabled 表示されている PIP を直ぐに停止する。
  - pipNormal PIP を直ぐに表示する。 現在設定されているアルファブレンドモードで PIP が表示される。PIP を表示中にアルファブレンディング比を変えた場合は、次のフレームから有効になる。
  - pipBlink1 PIP は、設定されたアルファブレンディングモードの表示と表示停止を繰り返す。
  - pipBlink2 PIP はノーマルと反転を繰り返す。アルファブレンディング比は一定のまま維持する。
  - pipFadeOut PIP を現在のアルファブレンド値から 0x0000(ブランク)までフェードアウトする。
  - pipFadeIn PIP を 0x0000(ブランク)から設定したアルファブレンド値までフェードインする。
  - pipContinous PIP の表示を、0x0000(ブランク)と設定したアルファブレンド値で繰り返す。

6.1.30 S1d13781::pipSetRotation()

PIP レイヤーの回転を設定するメソッドです。

パラメータ - rotationDegrees メインレイヤーの反時計回りの角度。許容角度は (0, 90, 180, 270)。

戻り値:

- なし。

6.1.31 S1d13781::pipGetRotation()

PIP レイヤーの回転設定を返すメソッドです。

パラメータ - なし。

戻り値:

- 現在の反時計回りの角度設定値 (0, 90, 180, 270)。

6.1.32 S1d13781::pipIsOrthogonal()

メインと PIP レイヤーが同じ回転設定であることを確認するメソッドです。

パラメータ - なし。

戻り値:

- 同じ回転設定のとき真。

- 回転設定が異なるとき偽。

6.1.33 S1d13781::pipSetColorDepth()

PIP レイヤーの色深度を設定するメソッドです。

パラメータ - colorDepth 下記 S1d13781::imageDataFormat 列挙型の値 による PIP レイヤーの色深度。

- format_RGB_888
- format_RGB_565
- format_RGB_888LUT
- format_RGB_565LUT
- format_RGB_332LUT

戻り値:

- なし。

6.1.34 S1d13781::pipGetColorDepth()

PIP レイヤーの現在の色深度設定を返すメソッドです。

パラメータ - なし。

戻り値:

- 現在の色深度 (設定可能な値):
  - format_RGB_888
  - format_RGB_565
  - format_RGB_888LUT
  - format_RGB_565LUT
  - format_RGB_332LUT

6.1.35 S1d13781::pipGetBytesPerPixel()

PIP レイヤーの色深度に基づいた、使用されるピクセル当たりのバイト数を返すメソッドです。

パラメータ - なし。

戻り値:

- ピクセル当たりのバイト数 (設定可能な値: 1, 2, または 3)。

6.1.36 S1d13781::pipSetStartAddress()

PIP レイヤーのメモリスタートアドレスをセットするメソッドです。スタートアドレスは PIP レイヤーの画像が始まるディスプレイメモリのオフセットです。

注:
スタートアドレスは 32-bit アラインでなければなりません (4 で割り切れること)。

パラメータ - pipStartAddress PIP レイヤー画像が始まるディスプレイメモリ内のオフセットバイト数。

戻り値:

- なし。

6.1.37 S1d13781::pipGetStartAddress()

PIP レイヤーのメモリスタートアドレスを返すメソッドです。スタートアドレスは PIP レイヤーの画像が始まるディスプレイメモリのオフセットです。

注:
スタートアドレスは 32-bit アラインでなければなりません (4 で割り切れること)。

パラメータ - なし。

戻り値:

- PIP レイヤー画像のスタートアドレス。

6.1.38 S1d13781::pipSetWidth()

PIP ウィンドウの幅ピクセル数を設定するメソッドです。

パラメータ - pipWidth PIP ウィンドウの幅ピクセル数。.

戻り値:

- なし。

6.1.39 S1d13781::pipGetWidth()

PIP ウィンドウの幅ピクセル数を返すメソッドです。

パラメータ - なし。

戻り値:

- PIP ウィンドウの幅ピクセル数。

6.1.40 S1d13781::pipSetHeight()

PIP ウィンドウの高さピクセル数を設定するメソッドです。

パラメータ - pipHeight PIP ウィンドウの高さピクセル数。

戻り値:

- なし。

6.1.41 S1d13781::pipGetHeight()

PIP ウィンドウの高さピクセル数を返すメソッドです。

パラメータ - なし。

戻り値:

- PIP ウィンドウの高さピクセル数。

6.1.42 S1d13781::pipGetStride()

PIP ウィンドウメインレイヤーの Stride をバイト単位で返すメソッドです。

Stride は、画像の 1 ライン(または行)当たりのバイト数です。この値をディスプレイメモリ内のピクセルのアドレスに加えることで、そのピクセルの下のピクセルのアドレスを得られます。

パラメータ - なし。

戻り値:

- PIP レイヤーの Stride(バイト単位)。

6.1.43 S1d13781::pipSetPosition()

LCD パネルの左上角を原点としたときの PIP ウィンドウの左上角の座標の位置を設定するメソッドです。

注:
1. PIP の x,y 座標はパネルの表示エリア内で設定する必要があります。
2. メインレイヤーの回転はチェックしないため、常にパネルの左上角に対しての PIP の位置となります。

パラメータ - xPos PIP ウィンドウの新しい x 座標(ピクセル単位)。

パラメータ - yPos PIP ウィンドウの新しい y 座標(ピクセル単位)。

戻り値:

- なし。

6.1.44 S1d13781::pipGetPosition()

LCD パネルの左上角(原点)に対する PIP ウィンドウの位置(x,y ピクセル単位)を得るためのメソッドです。

パラメータ - xPos x 開始位置へのポインタ(ピクセル数)。

パラメータ - yPos y 開始位置へのポインタ(ピクセル数)。

戻り値:

- なし。

6.1.45 S1d13781::pipSetFadeRate()

PIP ウィンドウのブリンク/フェード周期(フレーム単位)を設定するメソッドです。

PIP レイヤーがフェードイン、フェードアウトあるいはフェードイン/フェードアウト繰り返しに設定された場合、fadeRate 値は、次のステップのアルファブレンド比を設定する前にポーズするフレーム数を決めます。

パラメータ - fadeRate アルファブレンド比のステップ間で自動的にポーズするフレーム数。範囲は 1 から 64 まで。

戻り値:

- なし。

6.1.46 S1d13781::pipGetFadeRate()

PIP ウィンドウのブリンク/フェード周期(フレーム単位)の現在の設定を返すメソッドです。

PIP レイヤーがフェードイン、フェードアウトあるいはフェードイン/フェードアウト繰り返しに設定された場合、fadeRate 値は、次のステップのアルファブレンド比を設定する前にポーズするフレーム数を決めます。

パラメータ - なし。

戻り値:

- PIP ウィンドウのブリンク/フェード周期(フレーム単位)。

6.1.47 S1d13781::pipWaitForFade()

現在の PIP のブリンク/フェードが完了するまで待つメソッドです。

このメソッドは、通常、単発のフェードアウト、フェードイン効果で、フェードインまたはフェードアウトの完了を確認するのに使います。また、Blink1,Blink2 およびフェードイン/アウト効果のノーマルとブランクの繰り返しで、ブリンキングとフェーディングの完了を確認するのにも使います。

パラメータ - maxTime ウェイトの最大時間 (タイムアウト値)。

戻り値:

- フェードが完了の場合真、maxTime に到達すると偽。

6.1.48 S1d13781::pipSetAlphaBlendStep()

PIP ウィンドウのアルファブレンドステップを設定するメソッドです。

アルファブレンドステップは、フェードイン/フェードアウト効果でのアルファブレンド値のインクリメント/デクリメントステップを設定します。PIP ウィンドウのアルファブレンド比が "Full PIP" (100%)に設定されていない場合は、ステップ値はアルファブレンディング比の設定値を均等に割ることのできる値にする必要があります。

パラメータ - step アルファブレンドステップ (設定可能な値: 1, 2, 4, 8)。

戻り値:

- なし。

6.1.49 S1d13781::pipGetAlphaBlendStep()

PIP ウィンドウのアルファブレンドステップを設定するメソッドです。

アルファブレンドステップは、フェードイン/フェードアウト時のアルファブレンド値のインクリメント/デクリメントステップを設定します。もし、PIP ウィンドウのアルファブレンドレシ

オが"Full PIP" (100%)に設定されていない場合は、ステップ値はアルファブレンディング比の設定値を均等に割ることのできる値にする必要があります。

パラメータ - なし。

戻り値:

- アルファブレンドステップ値(設定可能な値: 1, 2, 4, 8)。

6.1.50 S1d13781::pipSetAlphaBlendRatio()

PIP ウィンドウのアルファブレンド比(%)を設定するメソッドです。

PIP レイヤーはメインレイヤー画像とアルファブレンドが可能です。S1D13781 はブレンディングとして 64 レベルをサポートしていますが、この関数は、%での設定で一番近い値を算出するように単純化されています。

パラメータ - ratio　　PIP レイヤの一のアルファブレンドの%で、0%(PIP が消失)から100%(PIP のみが見える)の間で設定。

戻り値:

- なし。

6.1.51 S1d13781::pipGetAlphaBlendRatio()

PIP ウィンドウのアルファブレンド比(%)を返すメソッドです。

PIP レイヤーはメインレイヤー画像とアルファブレンドが可能です。S1D13781 はブレンディングとして 64 レベルをサポートしていますが、この関数は、%での設定で一番近い値を算出するように単純化されています。

パラメータ - なし。

戻り値:

- PIP レイヤーのアルファブレンド比(%)に最も近い値(%)。

6.1.52 S1d13781::pipEnableTransparency()

PIP ウィンドウの透過関数をイネーブル/ディセーブルするメソッドです。

パラメータ - enable　　真で透過をイネーブル、偽で透過をディセーブル。

戻り値:

- なし。

6.1.53 S1d13781::pipGetTransparency()

PIP ウィンドウの現在の透過状態を返すメソッドです。

パラメータ - なし。

戻り値:

- 透過がイネーブルの場合真。
- 透過がディセーブルの場合偽。

6.1.54 S1d13781::pipSetTransColor()

PIP ウィンドウの透過キー色を設定するメソッドです。

PIP レイヤーの透過がイネーブルされていると、PIP レイヤーの透過色に合致する全てのピクセルは透過し、そこにはメインレイヤーのピクセルが代わりに表示されます。

透過色は 32-bit の RGB8888 で定義されます。

- x (bits 31-24) = 使用されない
- r (bits 23-16) = 8-bits の赤
- g (bits 15-8) = 8-bits の緑
- b (bits 7-0) = 8-bits の青

注:
RGB888 以外のモードについては、有効なビット位置をテクニカルマニュアルで確認して下さい。

パラメータ - xrgbColor 上記の透過キー色。

戻り値:

- なし。

6.1.55 S1d13781::pipGetTransColor()

PIP ウィンドウの透過キー色を返すメソッドです。

PIP レイヤーの透過がイネーブルされていると、PIP レイヤーの透過色に合致する全てのピクセルは透過し、そこにはメインレイヤーのピクセルが代わりに表示されます。

透過色は 32-bit の RGB8888 で定義される。

- x (bits 31-24) = 使用されない
- r (bits 23-16) = 8-bits の赤
- g (bits 15-8) = 8-bits の緑
- b (bits 7-0) = 8-bits の青

注:
RGB888 以外のモードについては、有効なビット位置をテクニカルマニュアルで確認して下さい。

パラメータ - なし。

戻り値:

- 上記の透過キー色。

6.1.56 S1d13781::pipSetupWindow()

シングルファンクションコールで PIP ウィンドウの設定を行うメソッドです。

この関数は PIP の x,y 座標、幅、高さおよび Stride を一つのファンクションコールで行えます。PIP の初期設定や複数のパラメータを同時にアップデートする必要がある場合に使います。

パラメータ - xPos　メインウィンドウの左上角に対する PIP ウィンドウの新しい x 座標(ピクセル単位)。

パラメータ - yPos　メインウィンドウの左上角の対する PIP ウィンドウの新しい y 座標(ピクセル単位)。

パラメータ - pipWidth　PIP ウィンドウの新しい幅(ピクセル単位)。

パラメータ - pipHeight　PIP ウィンドウの新しい高さ(ピクセル単位)。

戻り値:

- なし。

6.1.57 S1d13781::lcdSetLutEntry()

特定の値で LUT を設定するメソッドです。詳細は、テクニカルマニュアルの Look-Up Table Architecture を参照下さい。

LUT の xrgbData 色は以下のとおり、32-bit の RGB8888 で定義される。

- x (bits 31-24) = 使用されない
- r (bits 23-16) = 8-bits の赤
- g (bits 15-8) = 8-bits の緑
- b (bits 7-0) = 8-bits の青

注:
RGB888 以外のモードについては、有効なビット位置をテクニカルマニュアルで確認して下さい。

パラメータ - index　LUT データを書き込む LUT インデックス。

パラメータ - xrgbData　LUT インデックスに書き込むデータ。

パラメータ - window　列挙型 S1d13781::windowDestination の値に応じて指定したウィンドウに LUT がアクセスする。

- window_Main はアドレス 0x00060000 の LUT1 を使用。
- window_Pip は 0x00060400 の LUT2 を使用。

戻り値:

- なし。

6.1.58 S1d13781::lcdGetLutEntry()

特定の LUT の設定値を返すメソッドです。詳細は、テクニカルマニュアルの Look-Up Table Architecture を参照下さい。

LUT の xrgbData 色は以下のとおり、32-bit の RGB8888 で定義されます。

- x (bits 31-24) = 使用されない
- r (bits 23-16) = 8-bits の赤
- g (bits 15-8) = 8-bits の緑
- b (bits 7-0) = 8-bits の青

注:
RGB888 以外のモードについては、有効なビット位置をテクニカルマニュアルで確認して下さい。

パラメータ - index　LUT データを読み出す LUT インデックス。

パラメータ - window　列挙型 S1d13781::windowDestination の値に応じて指定したウィンドウに LUT がアクセスする。

- window_Main はアドレス 0x00060000 の LUT1 を使用。
- window_Pip は 0x00060400 の LUT2 を使用。

戻り値:

- 上記のとおり、特定の LUT の設定値のデータ。

6.1.59 S1d13781::lcdSetLutDefault()

LUT にデフォルトの値を設定するメソッドです。詳細は、テクニカルマニュアルの Look-Up Table Architecture を参照下さい。

パラメータ - window　列挙型 S1d13781::windowDestination の値に応じて指定したウィンドウに LUT がアクセスする。

- window_Main はアドレス 0x00060000 の LUT1 を使用。
- window_Pip は 0x00060400 の LUT2 を使用。

戻り値:

- なし。

6.2 S1d13781_gfx クラス

S1d13781_gfx クラスは、以下のパブリックメソッドを提供します。プライベートメソッドはこのドキュメントには記載していませんが、ソースコードに記載しています。

6.2.1 S1d13781_gfx()

これはクラスのコンストラクタです。

6.2.2 S1d13781_gfx::fillWindow()

特定の色で指定したウィンドウを塗りつぶすメソッドです。

色のパラメータは 32-bit の符号なし整数で以下のとおりです。

- RGB 8:8:8 モード (24 bpp)のとき:

  - bits 31-24 = 使用されない
  - bits 23-16 = 8-bits の赤
  - bits 15-8 = 8-bits の緑
  - bits 7-0 = 8-bits の青
- RGB 5:6:5 モード(16 bpp)のとき:
  - bits 31-16 = 使用されない
  - bits 15-11 = 5-bits の赤
  - bits 10-5 = 6-bits の緑
  - bits 4-0 = 5-bits の青
- for 8 bpp モードのとき:
  - bits 31-8 = 使用されない
  - bits 7-0 = 8-bit カラー値

注:

S1D13781 でサポートされるカラーフォーマットの詳細については、テクニカルマニュアルを参照下さい。

パラメータ - window　　塗りつぶされるウィンドウ。

パラメータ - color　　上記のように指定されるカラー値。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。
- 2 は無効な画像フォーマットを示すエラー。

6.2.3 S1d13781_gfx::clearWindow()

指定したウィンドウをクリアし、黒にするメソッドです。この関数は本質的に fillWindow()と同じです。

パラメータ - window　　塗りつぶされるウィンドウ。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。

6.2.4 S1d13781_gfx::drawPixel()

指定された x,y 座標に指定された色でピクセルを描画するメソッドです。

色パラメータは 32-bit の符号なし整数で以下のとおりです。

- RGB 8:8:8 モード (24 bpp):
  - bits 31-24 = 使用されない
  - bits 23-16 = 8-bits の赤
  - bits 15-8 = 8-bits の緑
  - bits 7-0 = 8-bits の青

- RGB 5:6:5 モード (16 bpp):
  - bits 31-16 =使用されない
  - bits 15-11 = 5-bits の赤
  - bits 10-5 = 6-bits の緑
  - bits 4-0 = 5-bits の青
- 8 bpp モード:
  - bits 31-8 = 使用されない
  - bits 7-0 = 8-bit カラー値

注:

S1D13781 でサポートされるカラーフォーマットの詳細については、テクニカルマニュアルを参照下さい。

パラメータ - window　　ピクセルの指定ウィンドウ。

パラメータ - x　　ウィンドウの座標 0,0 に対する X 座標。

パラメータ - y　　ウィンドウの座標 0,0 に対する Y 座標。

パラメータ - color　　上記のとおり指定されたカラー値。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。
- 2 はウィンドウ外の座標を示すエラー。

6.2.5 S1d13781_gfx::getPixel()

指定した x,y 座標のピクセルのカラー値を返すメソッドです。

カラーパラメータは 32-bit の符号なし整数で以下のとおりです。

- RGB 8:8:8 モード (24 bpp):
  - bits 31-24 = 使用されない
  - bits 23-16 = 8-bits の赤
  - bits 15-8 = 8-bits の緑
  - bits 7-0 = 8-bits の青
- RGB 5:6:5 モード (16 bpp):
  - bits 31-16 =使用されない
  - bits 15-11 = 5-bits の赤
  - bits 10-5 = 6-bits の緑
  - bits 4-0 = 5-bits の青
- 8 bpp モード:
  - bits 31-8 = 使用されない
  - bits 7-0 = 8-bit カラー値

注:

S1D13781 でサポートされるカラーフォーマットの詳細については、テクニカルマニュアルを参照下さい。

パラメータ - window　　ピクセルの指定ウィンドウ。

パラメータ - x　　ウィンドウの座標 0,0 に対する X 座標。

パラメータ - y　　ウィンドウの座標 0,0 に対する Y 座標。

戻り値:

- 0x00000000 to 0x00FFFFFF - カラーフォーマットに応じたピクセルのカラー値。
- 0xFF000000 - 無効なウィンドウを示すエラー 。
- 0xFE000000 -ウィンドウ外の座標を示すエラー。
- 0xFD000000 –無効なウィンドウ画像フォーマットを示すエラー。

6.2.6 S1d13781_gfx::drawLine()

指定した色で指定した 2 つの x,y 座標間に線を描画するメソッドです。

カラーパラメータは 32-bit の符号なし整数で以下のとおりです。

- RGB 8:8:8 モード (24 bpp):
  - bits 31-24 = 使用されない
  - bits 23-16 = 8-bits の赤
  - bits 15-8 = 8-bits の緑
  - bits 7-0 = 8-bits の青
- RGB 5:6:5 モード (16 bpp):
  - bits 31-16 =使用されない
  - bits 15-11 = 5-bits の赤
  - bits 10-5 = 6-bits の緑
  - bits 4-0 = 5-bits の青
- 8 bpp モード:
  - bits 31-8 = 使用されない
  - bits 7-0 = 8-bit カラー値

注:

S1D13781 でサポートされるカラーフォーマットの詳細については、テクニカルマニュアルを参照下さい。

パラメータ - window　　線を描画する指定ウィンドウ。

パラメータ - x1　　線分の始点を示すウィンドウの 0,0 を基準にした座標 X1。

パラメータ - y1　　線分の始点を示すウィンドウの 0,0 を基準にした座標 Y1。

パラメータ - x2　　線分の始点を示すウィンドウの 0,0 を基準にした座標 X2。

パラメータ - y2　　線分の始点を示すウィンドウの 0,0 を基準にした座標 Y2。

パラメータ- color　　上記のとおり指定されたカラー値。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。
- 2 はウィンドウ外の座標を示すエラー。

6.2.7 S1d13781_gfx::drawRect()

指定した色で指定した座標 x,y に指定した幅,高さで四角形を描画するメソッドです。

カラーパラメータは 32-bit の符号なし整数で以下のとおりです。

- RGB 8:8:8 モード (24 bpp):
  - bits 31-24 = 使用されない
  - bits 23-16 = 8-bits の赤
  - bits 15-8 = 8-bits の緑
  - bits 7-0 = 8-bits の青
- RGB 5:6:5 モード (16 bpp):
  - bits 31-16 =使用されない
  - bits 15-11 = 5-bits の赤
  - bits 10-5 = 6-bits の緑
  - bits 4-0 = 5-bits の青
- 8 bpp モード:
  - bits 31-8 = 使用されない
  - bits 7-0 = 8-bit カラー値

注:

S1D13781 でサポートされるカラーフォーマットの詳細については、テクニカルマニュアルを参照下さい。

パラメータ - window　四角形を描画するウィンドウ。

パラメータ - xStart　ウィンドウの 0,0 を基準にした四角形の描画開始位置座標 X。

パラメータ - yStart　ウィンドウの 0,0 を基準にした四角形の描画開始位置座標 Y。

パラメータ - width　四角形の幅。

パラメータ - height　四角形の高さ。

パラメータ - color　上記のとおり指定されたカラー値。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。
- 2 はウィンドウ外の座標を示すエラー。

6.2.8 S1d13781_gfx::drawFilledRect()

指定した色で指定した座標 x,y に指定した幅,高さで塗りつぶされた四角形を描画するメソッドです。

カラーパラメータは 32-bit の符号なし整数で以下のとおりです。

- RGB 8:8:8 モード (24 bpp):
  - bits 31-24 = 使用されない
  - bits 23-16 = 8-bits の赤
  - bits 15-8 = 8-bits の緑
  - bits 7-0 = 8-bits の青
- RGB 5:6:5 モード (16 bpp):
  - bits 31-16 =使用されない
  - bits 15-11 = 5-bits の赤
  - bits 10-5 = 6-bits の緑
  - bits 4-0 = 5-bits の青
- 8 bpp モード:
  - bits 31-8 = 使用されない
  - bits 7-0 = 8-bit カラー値

注:

S1D13781 でサポートされるカラーフォーマットの詳細については、テクニカルマニュアルを参照下さい。

パラメータ - window　　四角形を描画するウィンドウ。

パラメータ - xStart　　ウィンドウの 0,0 を基準にした四角形の描画開始位置座標 X。

パラメータ - yStart　　ウィンドウの 0,0 を基準にした四角形の描画開始位置座標 Y。

パラメータ - width　　四角形の幅。

パラメータ - height　　四角形の高さ。

パラメータ - color　　上記のとおり指定されたカラー値。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。
- 2 はウィンドウ外の座標を示すエラー。

6.2.9 S1d13781_gfx::drawPattern()

指定したウィンドウにテスト用パターンを描画するメソッドです。

パラメータ - window　　四角形を描画するウィンドウ。

パラメータ - pattern　　テストパターンの種類:

- S1d13781_gfx::patternRgbHorizBars　　(同一色の水平方向の RGB のバー)。

- S1d13781_gfx::patternRgbHorizGradient (グラデーション色の水平方向の RGB のバー)。
- S1d13781_gfx::patternVertBars (垂直方向の TV テストパターンタイプのカラーバー)。

パラメータ - colorStrength patternRgbHorizBars と patternVertBars に使用する色の強さのパーセンテージ(0=黒,100=最大強度)。 注: この設定は patternRgbHorizGradient には使用不可。

戻り値:

- ゼロ (0)はノーエラーを示す。
- 1 は無効なウィンドウを示すエラー。
- 2 は無効な画像フォーマットを示すエラー。
- 3 は無効なパターンを示すエラー。

6.2.10 S1d13781_gfx::createFont()

画像ファイルおよびインデックスフォントファイルの内容を元にシンプルなラスターフォントを描画するのに必要なストラクチャを作るためのメソッドです。

メモリのフットプリントを削減するために、フォントのルーチンは、テキストを描画する際に、ImageData バッファを参照します。このルーチンは、その後に使うために配置したりコピーしたりはしません。このため、ImageData を渡したバッファは開放しないで下さい。ここには、フォントインデックス情報用に作られた配列がありますので、アプリケーションを終了する際には、リソースを開放するために freeFont()をコールして下さい。

パラメータ - imageData フォントのイメージファイル(.pbm)のコンテンツ。下記の注を参照。

パラメータ - imageDataSize イメージファイルのコンテンツのサイズ。

パラメータ - indexData フォントインデックスファイル (.pfi)のコンテンツ。下記の注を参照。

パラメータ - indexDataSize インデックスファイルのコンテンツのサイズ。

戻り値:

- 作成されたフォントへのハンドルを返す。

注:

ImageData は、必ず、アスキー型でない”P4”画像ファイルを示す必要があります。フォントルーチンは直接、作成された ImageData バッファを参照するため P4 バイナリフォーマットでなければなりません。IndexData はアスキー型 (F1)およびバイナリ型 (F4)のどちらでも良いです。F1 および F2 フォーマットの詳細については、README.txt ファイルを参照下さい。

6.2.11 S1d13781_gfx::freeFont()

特定のフォントを無効化し、これに関連したシステムリソースを開放するメソッドです。

パラメータ - font 開放するフォント。

戻り値:

- NULL。

6.2.12 S1d13781_gfx::drawText() S1d13781_gfx::drawTextW()

指定したウィンドウに、与えられたフォントを使用して"Chars"または“Wchars”を含むテキストを描画するメソッドです。

パラメータ - window テキスト描画するウィンドウ。

パラメータ - font テキストに使用されるフォント。フォント情報は createFont()で予め作成されている必要がある。

パラメータ - text 描画されるテキスト。

パラメータ - X,Y テキストが描画される X と Y の位置。

パラメータ- width クロップ領域の幅(ピクセル単位)。0 はウィンドウ幅でクロップすることを意味する。

パラメータ - fgColor テキストの色。

パラメータ - bgColor テキストの背景色。NULL 値は背景色に変更なしを指定。

パラメータ - wordCrop ワードバウンダリでクロップする場合は真。

パラメータ - cropped テキストがクロップされた場合真(NULL 値を設定可能)。

戻り値:

- 描画された文字数 (クロップ後)。

6.2.13 S1d13781_gfx::drawMultiLineText() S1d13781_gfx::drawMultiLineTextW()

複指定したウィンドウに、与えられたフォントを使用して"Chars"または“Wchars”を含むテキストを含む複数のテキスト行を描画するメソッドです。

パラメータ - window テキスト描画するウィンドウ。

パラメータ - font テキストに使用されるフォント。フォント情報は createFont()で予め作成されている必要がある。

パラメータ - text 描画されるテキスト。

パラメータ - X,Y テキストが描画される X と Y の位置。

パラメータ - width クロップ領域の幅(ピクセル単位)。0 はウィンドウ幅でクロップすることを意味する。

パラメータ - fgColor テキストの色。

パラメータ - bgColor　テキストの背景色。NULL 値は背景色に変更なしを指定。

パラメータ - wordCrop　ワードバウンダリでクロップする場合は真。

パラメータ - cropped　テキストがクロップされた場合真(NULL 値を設定可能)。

パラメータ - linesDrawn　テキストを描画するのに要した行数。

戻り値:

- 描画された文字数(クロップ後)。

6.2.14 S1d13781_gfx::measureText() S1d13781_gfx::measureTextW()

"Chars"または“Wchars”を含むテキストのサイズをピクセル単位で求め、与えられた幅の中に表示可能な文字数を返すメソッドです。

パラメータ - font　テキストに使用されるフォント。フォントの情報は、予め createFont()で作成されている必要がある。

パラメータ - text　サイズを測るテキスト。

パラメータ - width　クロップ領域の幅(ピクセル単位)。

パラメータ - wordCrop　ワードバウンダリでクロップする場合は真。

パラメータ - cropped　テキストがクロップできるなら真を返す(NULL 値を設定可能)。

戻り値:

- 描画できる文字数。

6.2.15 S1d13781_gfx::getFontName()

与えられたフォント名を返すメソッドです。

パラメータ - font　テキストに使用されるフォント。フォント情報は予め createFont() で作成されている必要がある。

戻り値:

- フォント名。

6.2.16 S1d13781_gfx::getFontHeight(

与えられたフォントの高さをピクセル単位で返すメソッドです。

パラメータ - font　テキストに使用されるフォント。フォント情報は予め createFont() で作成されている必要がある。

戻り値:

- フォントの高さ(ピクセル単位)。

6.2.17 S1d13781_gfx::getCharWidth() S1d13781_gfx::getCharWidthW()

与えられたフォントで指定された"Char"または“Wchar”文字の幅をピクセル単位で返すメソッドです。

注: プロポーショナルフォントでは、文字幅は全て同じでない場合があります。

パラメータ - font　テキストに使用されるフォント。フォント情報は予め createFont() で作成されている必要がある。

パラメータ - character　サイズを測る "Char"文字。

戻り値:

- 文字の幅(ピクセル単位).。

6.2.18 S1d13781_gfx::getTextWidth() S1d13781_gfx::getTextWidthW()

与えられたフォントで、"Char"または“Wchar”文字で構成された指定されたテキストの幅をピクセル単位で返すメソッドです。

注: プロポーショナルフォントでは、文字幅は全て同じでない場合があります。

パラメータ - font　テキストに使用されるフォント。フォント情報は予め createFont() で作成されている必要がある。

パラメータ - text　"Char"文字で構成されるテキスト。

パラメータ - textLen　サイズを測るテキスト文字列の文字数。

戻り値:

- テキストの幅(ピクセル単位)。

6.2.19 S1d13781_gfx::captureFontIndexFile()

与えられたフォントからフォントインデックスファイルのコンテンツを生成するメソッドです。このメソッドは、アスキー版よりもサイズの小さいバイナリ版のインデックスファイルを生成するのに便利です。

パラメータ - font　テキストに使用されるフォント。フォント情報は予め createFont() で作成されている必要がある。

パラメータ - dFileBuffer　フォントインデックスファイルのコンテンツを受け取るバッファ。

パラメータ - dFileBufferSize　デスティネーションバッファの配列のサイズ。

パラメータ - binaryData　インデックスデータをバイナリまたはアスキーのどちらで書くかを決める。

戻り値:

- dBuffer に書くバイト数(処理がフェイルした場合は 0 を返す)。

6.2.20 S1d13781_gfx::copyArea()

矩形エリアを他の領域にコピーするメソッドです。

パラメータ - srcWindow コピー元のエリアがあるウィンドウ。

パラメータ - destWindow エリアをコピーする先のウィンドウ。

パラメータ - area コピーされるソース矩形エリア (x,y,w,h)。

パラメータ - destX コピー先 X 座標。

パラメータ - destY コピー先 Y 座標。

戻り値:

- 成功した場合 0 を返す。
- 無効なウィンドウの場合 1 を返す。
- ラインバッファメモリのアロケーションエラーの場合 2 を返す。
- drawPixel エラーの場合 3 を返す。

改訂履歴表

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev 1.01	2015/07/14	全ページ	新規	新規制定

セイコーエプソン株式会社
マイクロデバイス事業部　デバイス営業部

東京　〒191-8501　東京都日野市日野 421-8
TEL（042）587-5313 (直通)　　FAX（042）587-5116

大阪　〒541-0059　大阪市中央区博労町 3-5-1　エプソン大阪ビル 15F
TEL（06）6120-6000 (代表)　　FAX（06）6120-6100

ドキュメントコード：413109800
2015 年 7 月 作成