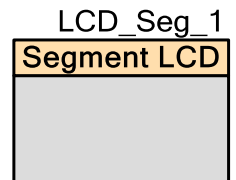


# セグメント LCD (LCD\_Seg)

## 3.0

## 特長

- 2 ～ 768 ピクセルまたはシンボル
- 1/3、1/4、1/5 バイアスをサポート
- 10～150Hz のリフレッシュレート
- 動的コントラスト制御用に最大 128 までのデジタルバイアス レベル制御による 2.0 V ～ 5.2 V のバイアス生成器を内蔵
- タイプ A (標準) およびタイプ B (ローパワー) の波形をサポート
- ディスプレイのピクセル状態によりネガティブ画像を反転
- 256 バイトの表示メモリ (フレーム バッファ)
- ユーザ定義可能なピクセルまたはシンボルマップとオプションの 7、14、または 16 セグメント文字、5x7 または 5x8 ドットマトリクス、およびバーグラフ計算ルーチン
- PSoC 3 ES3 シリコン リビジョン用以降を使用してください



## 概要説明

セグメント LCD (LCD\_Seg) コンポーネントは、最高 16 倍までの異なる電圧レベルの様々な LCD を直接駆動できます。このコンポーネントは、カスタムまたは標準の LCD を駆動するように PSoC デバイスを設定する容易な方法を提供します。

内部バイアス生成により、外部ハードウェアを必要としない、ソフトウェアベースのコントラスト調整が可能になります。ブースト コンバータを使うと、LCD バイアスを PSoC 電源電圧より高い電圧にすることができます。これにより、ポータブルアプリケーション用ディスプレイの柔軟性を向上できます。

各 LCD ピクセル/シンボルはオンまたはオフにできます。セグメント LCD コンポーネントはまた、次のタイプの LCD 表示構造を簡素化するための高度なサポートを提供します。

- 7 セグメント (数字)
- 14 セグメント (英数字)

- 16 セグメント (英数字)
- 5x7 および 5x8 ドットマトリクス 英数字 (5x7 と 5x8 で同じルックアップ テーブルを使用します。ルックアップ テーブル内のすべてのシンボルは 5x7 ピクセルサイズです。)
- 1~255 エLEMENTのバーグラフ

セグメント LCD コンポーネント使用の詳細については、アプリケーション ノート [AN52927: PSoC® 3: セグメント LCD 直接駆動の基本](#)。

## セグメント LCD はどのような場合に使うか

ダイレクト セグメント ドライブ LCD コンポーネントは、異なる電圧レベルで様々な LCD グラスを直接駆動する必要があるときに、最高 16x までの倍数比を使用できます。ダイレクト セグメント ドライブ LCD コンポーネントを使うには、対象となる PSoC デバイスが LCD ダイレクト ドライブをサポートする必要があります。

## 入出力接続

回路図上には、コンポーネントの接続は表示されませんが、デザインワイド リソース ピン エディタを使ってさまざまな信号を接続できます。

## コンポーネント・パラメータ

セグメント LCD コンポーネントを設計上にドラッグし、ダブルクリックして **Configure (設定)** ダイアログを開きます。**Configure (設定)** ダイアログには、異なるタイプのセグメント LCD コンポーネントの設定に対応したパラメータ用のタブがいくつかあります。

## Basic Configuration (基本設定) タブ

Configure 'SegLCD'

Name: LCD\_Seg\_1

Basic Configuration | Driver Power Settings | Display Helpers | Custom Characters

Number of common lines: 4

Number of segment lines: 8

☐ Enable Ganging Commons

Bias type: 1/3

Waveform type: Type A Standard

Frame rate, Hz: 60

Driver Power Mode: No Sleep

Bias voltage, V: 3.000 (3.0 V, 5.5 V)

Datasheet OK Apply Cancel

### Number of Common Lines (コモンライン数)

そのディスプレイに必要なコモン信号数を指定します (デフォルトは 4)。

### Number of Segment Lines (セグメントライン数)

そのディスプレイに必要なコモン信号数を指定します。指定できる値の範囲は 2～62 です。デフォルトは 8 です。

### Enable Ganging Commons (コモンの連結)

PSoC ピンを連結してコモン信号を駆動するには、このチェックボックスを選択します。各コモン信号に 2 つの PSoC ピンが割り当てられます。これは大きいディスプレイを駆動する場合に使用します。

### Bias Type (バイアス タイプ)

コモンとセグメント ラインのセットに適切なバイアス モードを設定します。

### Waveform Type (波形タイプ)

これは次の波形タイプを決定します: Type A - 単一フレームの平均 0 VDC (デフォルト) または Type B - 2 フレームの平均 0 VDC。

## フレーム レート、Hz

ディスプレイのリフレッシュ レートを設定します。No Sleep (スリープなし) モードでは 10Hz～150Hz の範囲で 10 の増分で選択可能です。デフォルトは 60 Hz です。

低消費電力モードでは、フレーム レートは制限されており、すべての設定に対して 1 つのみです。詳細については、このデータシートの「[関数の説明](#)」セクションの「[ドライバパワーモード](#)」を参照してください。

## Driver Power Mode (ドライバパワーモード)

Driver Power Mode (ドライバ パワー モード) パラメータは、コンポーネントのパワー モードを指定します。次のパワーモード設定が可能です。

- **No Sleep (スリープなし):** LCD DAC は常にオンになり、チップはスリープ モードに入りません。
- **Low Power using ILO (ILO を使用したローパワー):** LCD DAC はオンになりますが、チップは電圧遷移中にスリープ モードに入ります。コンポーネントは 1 kHz の内部 ILO をウェイクアップ ソースとして使用します。
- **Low Power using Ext 32 kHz crystal (外部 32kHz クリスタル使用のローパワー):** LCD DAC はオンになりますが、チップは電圧遷移中にスリープ モードに入ります。コンポーネントは OPPS タイマからの 8-K タップをウェイクアップ ソースとして使用します。

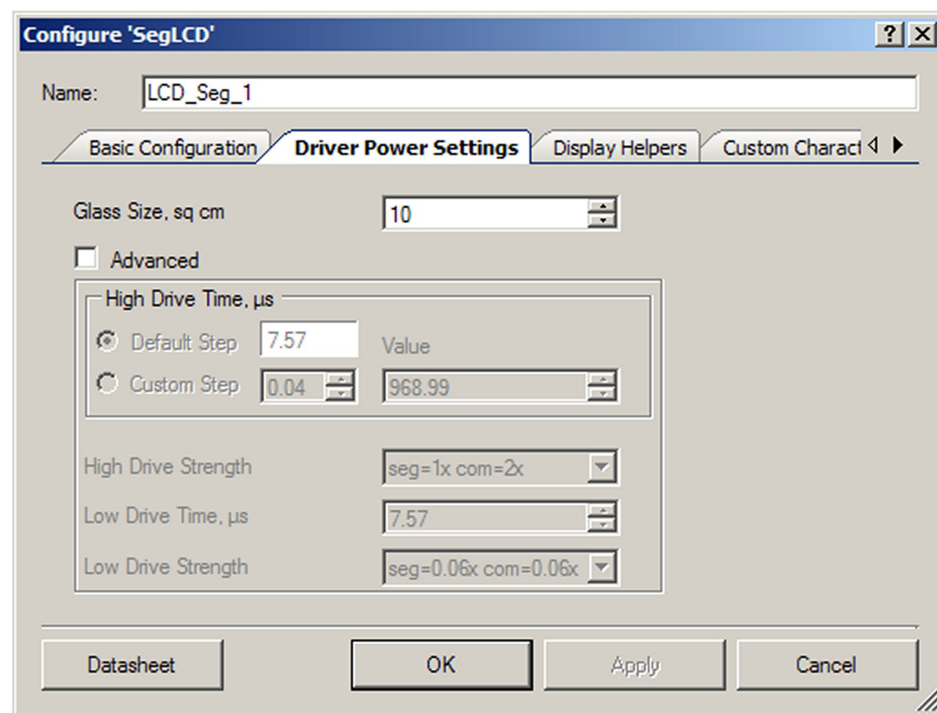
**注** 使用するローパワーモードに応じて、ILO を 1kHz に設定して使うか、外部 32 kHz クリスタルを接続し有効にして使用してください。32kHz クリスタルを有効にするもよし、ILO の周波数を Design-Wide Resources Clock Editor で設定することもできます。

このデータシートの「[関数](#)」セクションの「[ドライバパワーモード](#)」も参照してください。

## Bias voltage, V (バイアス電圧、V)

LCD DAC のバイアス電圧レベルを設定します。指定できる値は供給源によって異なり、2.35 V ～ 5.5 V または 2.017 V ～ 3 V です。

## Driver Power Settings (ドライバパワー設定) タブ



### Glass Size, sq cm (ガラスサイズ、平方 cm)

ガラス有効領域のおよそのサイズを平方センチメートルで入力します。この値は、**Number of Common Lines** (コモンライン数) パラメータと共におよその容量性負荷の計算に使用されます。

### 詳細設定

このチェックボックスを選択すると、LCD の **Driver Power Settings** (ドライバパワー設定) を手動で変更できます。このチェックボックスを選択しないと、ドライバパワー設定は **Basic Configuration** (基本設定) タブと **Glass Size** (ガラス サイズ) パラメータの選択に基づいて自動的に設定されます。

### High Drive Time, μs (ハイ ドライブ時間、μs)

1 回の電圧遷移内で High Drive (ハイ ドライブ) モードにする時間を指定します。

### Default Step (デフォルト増分値)

選択した設定の自動的に計算された High Drive (ハイドライブ) 増分値を指定します。これは、選択すると High Drive time (ハイドライブ時間) の値の増減値となります。

No Sleep (スリープなし) モードで選択できるのは、Default High Drive step (デフォルト ハイドライブ増分値) だけです。



## Custom Step (カスタム増分値)

Low Power (ローパワー) モードでのみ使用可能です。ユーザ選択可能な High Drive Step (ハイドライブ増分値) を指定します。カスタム High Drive (ハイドライブ) 増分値の可能な最低値は、マスタ クロックの周波数によって異なります。可能な最大値は、**Default Step** (デフォルト増分値) の値によって制限されます。

**注** **Frame Rate** (フレームレート)、**Number of Common Lines** (コモンライン数)、または **Waveform Type** (波形タイプ) パラメータを変更した場合、**High Drive Time** (ハイドライブ時間) パラメータは自動的にデフォルト値の半分に設定されます (**Advanced** (詳細設定) チェックボックスが選択されている場合のみ)。

$$\text{HighDriveTimeValdef} = 1.075 / \text{DefaultStep} \times 128 \quad (\text{Type A})$$

$$\text{HighDriveTimeValdef} = 1.075 / \text{DefaultStep} \times 128 \quad (\text{Type B})$$

**High Drive Time** (ハイドライブ時間) の最小値の計算については、このデータシートで後述します。現在の設定での **High Drive Time** (ハイドライブ時間) の最大値は以下の式で計算されます。

$$\text{HighDriveTimeValmax} = 1.075 / \text{DefaultStep} \times 253 \quad (\text{Type A})$$

$$\text{HighDriveTimeValmax} = 1.075 / \text{DefaultStep} \times 253 \quad (\text{Type B})$$

上記の計算式は、**Custom Step** (カスタム増分値) 指定の **High Drive Time** (ハイドライブ時間) 値にも適用されます。

ローパワー モードのどれかを選択した場合、デフォルトの High Drive (ハイドライブ) 増分値を使って **High Drive Time** (ハイドライブ時間) の最大値を設定することは推奨されません。デバイスは短時間スリープモードになるだけか、最悪の場合は全くスリープにならないので、ローパワー モードの使用は効果的ではありません。Custom Step (カスタム増分) を使うと、多くのローパワー アプリケーションの要件である **High Drive Time** (ハイドライブ時間) 値をより正確に調整できます。

## High Drive Strength (ハイドライブ強度)

このパラメータを選択すると、High Drive (ハイドライブ) の駆動強度を選択できます。

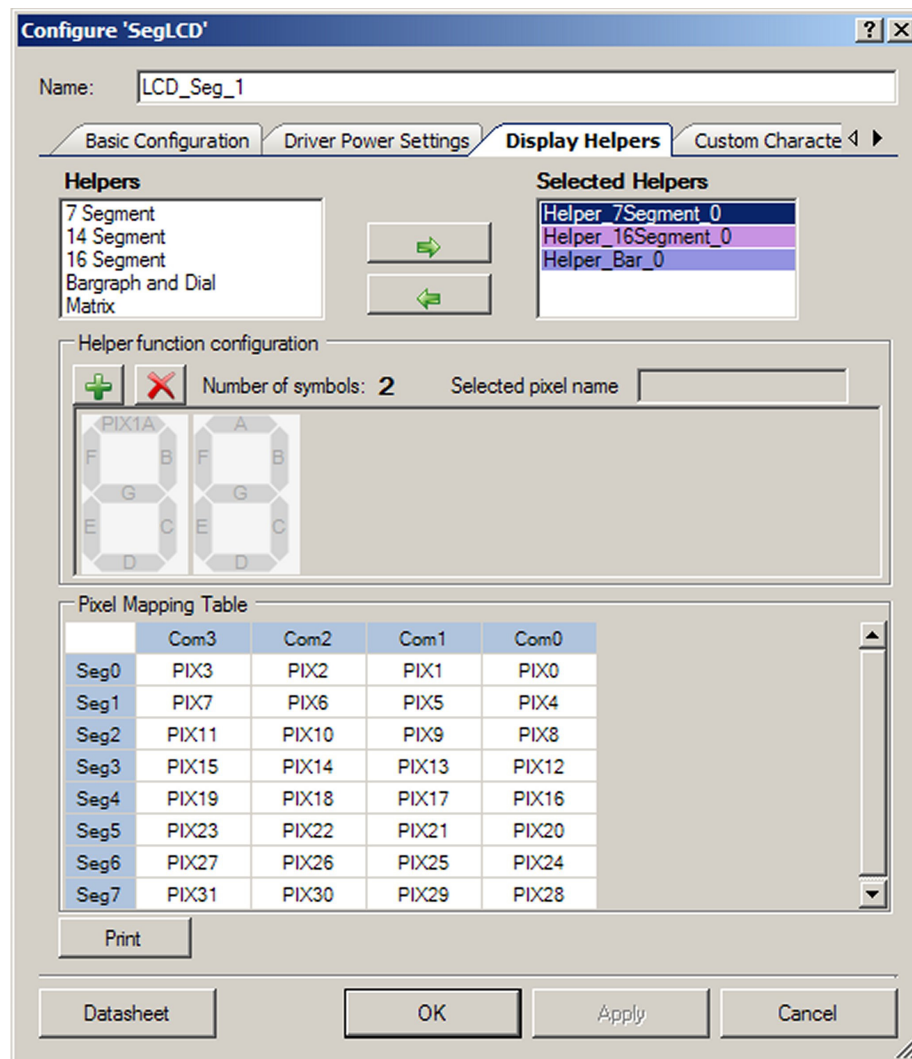
## Low Drive Time, $\mu\text{s}$ (ロー ドライブ時間、 $\mu\text{s}$ )

**Low Drive Time** (ロードライブ時間) パラメータは、1 回の電圧遷移内でロー ドライブ モードをアクティブにする時間を指定します。

## Low Drive Strength (ロードライブ強度)

このパラメータを選択すると、Low Drive (ロードライブ) の駆動強度を選択できます。**Low Drive Strength** (ロードライブ強度) は **High Drive Strength** (ハイドライブ強度) に比例しており、High Drive (ハイドライブ) に応じて自動的に設定されます。

## Display Helpers (ディスプレイ ヘルパー) タブ



Display Helpers (ディスプレイ ヘルパー) では、いくつかある、あらかじめ定義されたディスプレイ エLEMENT タイプの 1 つとして一緒に使用するディスプレイセグメントのグループを設定できます。

- 7、14、または 16 セグメントディスプレイ
- ドット マトリクス ディスプレイ (5x7 または 5x8)
- 直線または円形(メーター状) バーグラフディスプレイ

キャラクタベースのディスプレイヘルパーは、複数のディスプレイ シンボルをまとめるのに使用され、マルチキャラクタディスプレイ エLEMENTを作成できます。



## Helpers (ヘルパー) / Selected Helpers (選択したヘルパー)

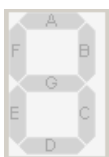
Helpers (ヘルパー) リストから使用するヘルパーのタイプを選択し、右向き矢印ボタンをクリックすることで、1 つまたは複数のヘルパーを **Selected Helpers** (選択したヘルパー) リストに追加できます。新しいヘルパーをサポートするのに十分なピンがない場合は、そのヘルパーは追加されません。ヘルパーを削除するには、**Selected Helpers** (選択したヘルパー) リストからそのヘルパーを選択して左向き矢印ボタンをクリックします。

**注:** ディスプレイ ヘルパーを指定する前に、コンポーネントのコモンライン数とセグメントライン数を設定することが重要です。コモンライン数やセグメントライン数を変更する場合はヘルパー設定情報が失われる可能性があるため、その前に指定されているディスプレイ ヘルパーを削除する必要があります。コモンライン数またはセグメントライン数を変更しようとすると、ヘルパーのピクセルマッピング設定が失われる可能性があるという警告が表示されます。

選択したヘルパーが **Selected Helpers** (選択したヘルパー) リストに表示される順序は重要です。デフォルトで、**Selected Helper** (選択したヘルパー) リストに加えられた各ヘルパータイプの最初のヘルパーには 0 が後付けされ、そのタイプの次のヘルパーには 1 が後付けされるというように順々に名前が付けられます。選択したヘルパーのどれかが **Selected Helper** (選択したヘルパー) リストから削除されても、残りのヘルパーの名前は変更されません。ヘルパーが追加されると、最後の番号の次の番号が後付けされます。

各ヘルパーに API が提供されます。詳細については、[Application Programming Interface](#) (アプリケーションプログラミング インターフェイス) のセクションを参照してください。

- **7 Segment Helper (7 セグメント ヘルパー)** – このヘルパーは長さが 1 ～ 5 桁で、16 進数の 0 ～ F または 16 ビットの符号なし 10 進整数 (uint16) で値を表示します。ヘルパー関数では小数点表示はサポートされていません。



- **14 Segment Helper (14 セグメント ヘルパー)** – このヘルパーの長さは最大 20 文字です。単一の ASCII 文字を表示することもヌル('\0')終端文字列を表示することもできます。指定できる値は標準 ASCII 印刷可能文字です (コード 0 ～ 127)。



- **16 Segment Helper (16 セグメント ヘルパー)** – このヘルパーの長さは最大 20 文字です。単一の ASCII 文字を表示することも完全なヌル終端文字列を表示することもできます。指定できる値は標準 ASCII 文字と拡張コード表 (コード 0 ～ 255) です。拡張コード表は提供されていません。





- **Bar Graph and Dial Helper (バーグラフとダイヤル ヘルパー)** – これらのヘルパーは 1 ～ 255 セグメントのバーグラフとダイヤル表示器に使用します。バーグラフは選択した単一ピクセルとしても、選択したピクセルとその右または左側のすべてのピクセルとしても指定できます。



- **Matrix Helper (マトリクス ヘルパー)** – このヘルパーは 8 つのキャラクタ エlementをサポートします。このコンポーネントは x5x7 または x5x8 行/列文字をサポートします。より長い文字列は、複数のドットマトリクス ヘルパーを設定し、ディスプレイの隣接ドットマトリクス セクションを定義することで作成できます。このヘルパーは単一の ASCII 文字を表示することもヌル終端文字列を表示することもできます。

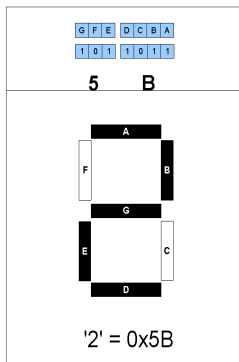


ドットマトリクス ヘルパーにはピンアウト制約があります。ドットマトリクス ヘルパーは、マトリクスの行用に 7 または 8 個のシーケンシャル コモン ドライバを持ち、マトリクスの列用に 5 ～ 40 個のシーケンシャル セグメント ドライバを持つ必要があります。このコンポーネントは標準 Hitachi HD44780 文字セットをサポートします。

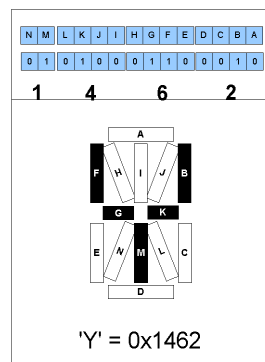
## 文字の符号化

高レベルのヘルパー API はすべて各々のルックアップ テーブルを持ちます。このテーブルには、指定した文字設定を形成する一連の符号化ピクセルの状態が格納されています。次の例は、特定の文字が符号化される方法を示しています (セグメント名は **Configure** (設定) ダイアログに示されるものとは異なる場合があります)。

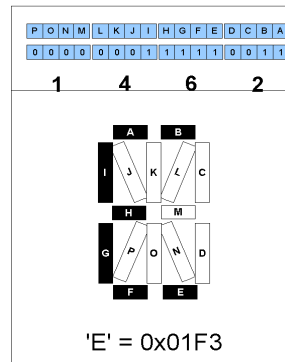
7 セグメント符号化



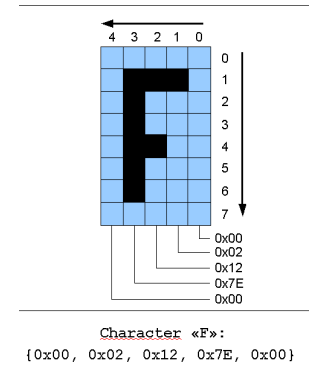
14 セグメント符号化



16 セグメント符号化



ドットマトリクス符号化



## ヘルパー関数の設定

ダイアログのこのセクションでは、ユーザはシンボルの追加と削除、ピクセルの命名など、ヘルパーの設定を行うことができます。

1. **Selected Helpers** (選択したヘルパー) リストから設定するヘルパーを選択します。
2. 選択したヘルパーのシンボルを追加するには **[+]** ボタンをクリックし、削除するには **[x]** ボタンをクリックします。  
追加できるシンボルの最大数はヘルパーのタイプとコンポーネントでサポートされているピクセルの数によって異なります。使用可能なピン数が新しいシンボルをサポートするのに十分でない場合、そのシンボルは追加されません。
3. ヘルパー機能の一部であるピクセルの名前を変更するには、ヘルパー関数設定画面上のそのシンボル画像でそのピクセルを選択します。現在の名前が選択したピクセル名フィールドに表示され、自由に変更できます。

## ピクセルの命名

デフォルトのピクセル名は「PIX#」の形式で指定されます。ここで、「#」はピクセル マッピング テーブルの右上端から始まり、1 ずつ増分されるピクセル番号です。

ヘルパー シンボルに関連付けられているピクセルのデフォルト名は異なる形式です。デフォルト名は、1 つのシンボル内のすべてのピクセルに共通のプリフィックス部分と各ピクセルで一意的なセグメント識別子から成ります。デフォルト名のプリフィックス部分はヘルパー タイプとシンボルインスタンスを示します。たとえば、7 セグメント ディスプレイ ヘルパー内の 1 つのシンボルの 1 つのピクセルのデフォルト名が「H7SEG4\_A」であるとする、

- H7          そのピクセルが 7 セグメント ヘルパーの一部であることを示します
- SEG4        そのピクセルがそのプロジェクト内の 4 番目の 7 セグメントシンボルとして指定されているシンボルの一部であることを示します
- A            7 セグメント シンボル内の一意なセグメントとして他と区別します

デフォルトのピクセル名では、ピクセル名の一意部分のみがシンボル画像に表示されます。ピクセル名を変更すると、共通プリフィックスを持つ場合にもシンボル画像上に名前全体が表示されます。

**注** すべてのピクセル名は一意でなければなりません。

ヘルパー関数シンボル エLEMENTが**ピクセルマッピング テーブル** (次項で説明) に割り当てられているとき、そのピクセルはそのヘルパー シンボル エLEMENTの名前を受け継ぎます。ヘルパー シンボル エLEMENT名はデフォルトピクセル名に優先しますが、上書きはしません。ヘルパー関数に関連付けられているピクセルのデフォルトピクセル名は再使用できません。

### ピクセルマッピング テーブル

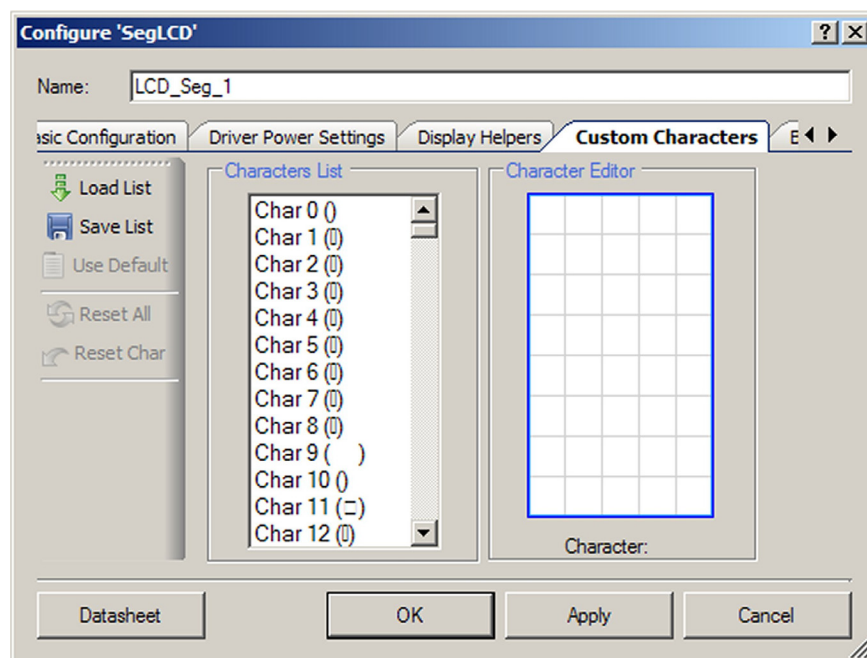
ピクセルマッピング テーブルはフレーム バッファを表示したものです。API 関数が正しく働くためには、**Helper Function Configuration** (ヘルパー関数設定) からの各ピクセルが**ピクセルマッピング テーブル**のピクセル位置に割り当てられている必要があります。正しい割り当てを行うために必要な情報は、LCD ガラスのデータシートを参照してください。

ピクセルを割り当てるには、**Helper Function Configuration** (ヘルパー関数設定) パネルでそのピクセルを選択し、**ピクセルマッピング テーブル**内の正しい位置にドラッグします。

**ピクセル マッピング テーブル**にてピクセル名を変更するには、テーブル ディスプレイにてピクセルをダブルクリックし、新しい名前を入力してください。この方法は、使用可能なヘルパー タイプに関連付けられていないピクセル名に使用できます。

**Print** (印刷) ボタンをクリックすると、**ピクセルマッピンググテーブル**が印刷されます。

## Custom Characters (カスタム文字) タブ



このタブでは、5x8 ドットマトリクス表示用のカスタム文字を作成できます。カスタム文字のルックアップ テーブルを XML 文字列として保存するために使用することもできます。

**Character List** (文字リスト) フィールドにはデフォルトで、標準の Hitachi HD44780 文字セットに入っている 255 個の ASCII 文字があります。これらの文字はどれも **Character Editor** (文字エディタ) を使ってアクセスし、変更できます。

**Reset Char** (文字のリセット) オプションを使うと、未保存 (編集) 文字をデフォルト設定に戻すことができます。

**Reset All** (すべてリセット) オプションを使うと、未保存の文字をすべて標準設定に戻すことができます。

保存した自分の文字セットは、**Save List** (リストの保存) コマンドを用いて XML 文字列として保存できます。

**Load List** (リストのロード) コマンドを使うと、XML 文字列から自分の文字リストをロードできます。**Use Default** (デフォルトの使用) オプションを使って、標準文字セットに戻すこともできます。

## Clock Select (クロック選択)

LCD\_Seg コンポーネントは 1 つの内部クロックを使用するので、外部クロックは必要ありません。LCD コンポーネントが配置されると、クロックは自動的にそのコンポーネント専用になります。周波数は、コモンライン数、リフレッシュレート、および波形タイプに基づいて自動的に計算されます。

## 配置

LCD\_Seg コンポーネントの実装は 2 つの部分から成ります。LCDDAC は、このコンポーネントで使用される PSoC 内の固定機能ハードウェア ブロックです。UDB には、ドライブ信号用の追加のタイミング ロジックが実装されます。UDB リソースは、プロジェクト生成プロセス中に自動的に UCB アレイ内に配置されます。

**注:** 1 つのプロジェクトで使用できるコンポーネントのインスタンスは 1 つだけです。複数のインスタンスを使用すると、ビルド時に配置エラーが発生します。

デフォルト ピン割り当てはビルド プロセス中に行われ、PSoC Creator Design-Wide Resources ツール内のピン エディタを使って変更できます。

## リソース

下の表に、スタティック セグメント LCD コンポーネントに対して指定できるすべての設定を示します。設定名の意味は次の通りです。

**基本:** 高レベルのヘルパーAPI のない低レベルの API 関数

**基本、7 セグメント ヘルパー:** 低レベルの API 関数群 + 7 セグメント ヘルパー API

**基本、14 セグメント ヘルパー:** 低レベルの API 関数群 + 14 セグメント ヘルパー API

**基本、16 セグメント ヘルパー:** 低レベルの API 関数群 + 16 セグメント ヘルパー API

**基本、ドットマトリックス ヘルパー:** 低レベルの API 関数群 + ドットマトリックス ヘルパーの高レベル API。

**基本、バーグラフ ヘルパー:** 低レベルの API 関数群 + バーグラフ ヘルパーの高レベル API。

リソース	リソースのタイプ						API メモリ(バイト)		ピン(外部入出力ごと)
	データバスセル	PLD	LCD 固定ブロック	Control/Count7 セル	同期セル	割り込み	フラッシュ	RAM	
基本	1	3	1	1	0	1	2691	104	3 ~ 62
基本、7 セグメント ヘルパー	1	3	1	1	0	1	2870	129	3 ~ 62
基本、14 セグメント ヘルパー	1	3	1	1	0	1	3711	409	3 ~ 62
基本、16 セグメント ヘルパー	1	3	1	1	0	1	3812	409	3 ~ 62
基本、ドットマトリックス ヘルパー	1	3	1	1	0	1	4875	409	3 ~ 62



リソース	リソースのタイプ						API メモリ(バイト)		ピン(外部入出力ごと)
	データバスセル	PLD	LCD 固定ブロック	Control/Count7 セル	同期セル	割り込み	フラッシュ	RAM	
基本、バーグラフヘルパー	1	3	1	1	0	1	3590	409	3 ~ 62

## アプリケーション プログラミング インタフェース

アプリケーション プログラミング インターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインターフェースとその説明を示しています。後続のセクションでは、各関数について詳しく説明します。

デフォルトで、PSoC Creator は、インスタンス名「LCD\_Seg\_1」を、特定の設計における最初のコンポーネントインスタンスに割り当てます。そのインスタンス名は、PSoC Creator の識別子の構文ルールに従う任意の一意名に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。便宜上、次の表ではインスタンス名として「LCD\_Seg」を使っています。

関数	説明
LCD_Seg_Start()	initVar 変数を設定し、LCD_Seg_Init() 関数を呼び出してから、LCD_Seg_Enable() 関数を呼び出します。
LCD_Seg_Stop()	LCD コンポーネント、それに関連した割り込み、DMA チャンネルを無効にします。
LCD_Seg_EnableInt()	LCD の割り込みを有効にします。LCD_Seg_Start() が呼び出される場合は必要ありません。
LCD_Seg_DisableInt()	LCD の割り込みを無効にします。LCD_Seg_Stop() が呼び出される場合は必要ありません。
LCD_Seg_SetBias()	LCD ガラスのバイアス レベルを 128 ある値のどれかに設定します。
LCD_Seg_WriteInvertState()	入力パラメータに基づいて表示を反転します。
LCD_Seg_ReadInvertState()	標準か反転かという表示反転状態の現在値を返します
LCD_Seg_ClearDisplay()	表示とその関連フレームバッファ RAM をクリアします。
LCD_Seg_WritePixel()	PixelState の値に基づいてピクセルをセットまたはクリアします
LCD_Seg_ReadPixel()	フレーム バッファ内のピクセルの状態を読み取ります。
LCD_Seg_Sleep()	LCD を停止し、ユーザ設定を保存します。
LCD_Seg_Wakeup()	ユーザ設定を復元し、イネーブルにします。
LCD_Seg_SaveConfig()	LCD 設定を保存します。
LCD_Seg_RestoreConfig()	LCD 設定を復元します。

関数	説明
LCD_Seg_Init()	Configure (設定) ダイアログの設定に従ってLCDを初期化または復元します。
LCD_Seg_Enable()	LCD を有効にします。

## グローバル変数

変数	説明
LCD_Seg_initVar	LCD_Seg_initVar は、セグメントLCD が初期化されているかどうかを示します。変数は 0 に初期化され、LCD_Seg_Start() が最初に呼び出されたときに 1 に設定されます。これにより、LCD_Seg_Start() ルーチンを最初に呼び出した後、再初期化を行うことなく、コンポーネントを再起動できます。  コンポーネントの再初期化が必要な場合には、LCD_Seg_Start() または LCD_Seg_Enable() 関数の前に LCD_Seg_Init() 関数を呼び出すことができます。

## uint8 LCD\_Seg\_Start(void)

**説明:** この関数は、LCD コンポーネントを起動して、必要な割り込み、DMA チャンネル、フレーム バッファ、ハードウェアを有効にします。フレーム バッファ RAM はクリアされません。

**パラメータ:** なし

**戻り値:** uint8 cstatus: 標準API戻り値。

戻り値	説明
CYRET_LOCKED	一部のDMA TDまたはチャンネルが既に使用中。
CYRET_SUCCESS	関数は正しく完了

**副作用:** このAPIは、Timewheel Configuration Register 2 からの 1 pulse-per-second ビットを有効にします。これは ローパワー32kHz外部Xtalコンポーネントモードでのみ有効です。

## void LCD\_Seg\_Stop(void)

**説明:** この関数は、LCDコンポーネント、それに関連した割り込みおよびDMAチャンネルを無効にします。自動的に表示をブランクにし、DCオフセットからの損傷を避けます。フレーム バッファはクリアされません。

**パラメータ:** なし

**戻り値:** なし

**副作用:** この API は、LCD\_Seg\_Init() API で前に有効にされた Timewheel Configuration Register 2 からの 1 pulse-per-second ビットをクリアしません。これは ローパワー32kHz外部Xtalコンポーネント モードでのみ有効です。





**void LCD\_Seg\_EnableInt(void)**

説明:	この関数は、LCD割り込みを有効にします。LCD_Seg_Start() が呼び出された場合は必要ありません。LCD の更新 (TD の完了) ごとに割り込みが発生します。
パラメータ:	なし
戻り値:	なし
副作用:	なし

**void LCD\_Seg\_DisableInt(void)**

説明:	この関数は LCD割り込みを無効にします。LCD_Seg_Stop() が呼び出される場合は必要ありません。
パラメータ:	なし
戻り値:	なし
副作用:	なし

**void LCD\_Seg\_SetBias(uint8 biasLevel)**

説明:	この関数は、LCD グラスのバイアスレベルを 128 ある値のどれかに設定します。実際の値の数はアナログ電源電圧 $V_{DDA}$ によって制限されます。バイアス電圧は $V_{DDA}$ を超えることはできません。バイアスレベルを変更すると LCD のコントラストに影響します。
パラメータ:	uint8 biasLevel: ディスプレイのバイアスレベル
戻り値:	なし
副作用:	なし

**uint8 LCD\_Seg\_WriteInvertState(uint8 invertState)**

説明:	この関数は、入力パラメータに基づいて表示を反転します。反転はハードウェアで行われ、フレーム バッファ内のディスプレイ RAM の変更は必要ありません。
パラメータ:	uint8 invertState: 表示の反転状態を設定します
戻り値:	uint8 cstatus: 標準 API 戻り値
副作用:	なし

**uint8 LCD\_Seg\_ReadInvertState(void)**

**説明:** この関数は、標準か反転の表示反転状態の現在値を返します。

**パラメータ:** なし

**戻り値:** (uint8) invertState: 表示の反転状態

**副作用:** なし

**void LCD\_Seg\_ClearDisplay(void)**

**説明:** この関数は、表示とその関連フレーム バッファ RAM をクリアします。

**パラメータ:** なし

**戻り値:** なし

**副作用:** なし

**uint8 LCD\_Seg\_WritePixel(uint16 pixelNumber, uint8 pixelState)**

**説明:** 入力パラメータ PixelState の値に基づいてピクセルをセットまたはクリアします。ピクセルは、パック数値で指定されます。

**パラメータ:** uint16 pixelNumber: フレームバッファ内のピクセル位置をポイントするパック数値。LSB の下位ニブル内の最下位 3 ビットがそのバイトのビット位置で、LSB の上位ニブル (4 ビット) はマルチプレクスアドレス行のバイトアドレス、MSB の下位ニブル (4 ビット) はマルチプレクスアドレス行番号です。生成された component .h ファイルには、各ピクセルの #defines がこの形式で格納されています。

uint8 pixelState: 指定された pixelNumber がこのピクセル状態に設定されます。

**戻り値:** uint8 status: バイトアドレスおよびマルチプレクスアドレスの行番号に対する、範囲チェックの戻り値 (pass または fail) です。ビット位置のチェックは行われません。

**副作用:** なし

**uint8 LCD\_Seg\_ReadPixel(uint16 pixelNumber)**

**説明:** この関数は、フレーム バッファ内のピクセルの状態を読み取ります。ピクセルは、パック数値で指定されます。

**パラメータ:** uint16: pixelNumber: フレームバッファ内のピクセル位置をポイントするパック数値。LSB の下位ニブル内の最下位 3 ビットがそのバイトのビット位置で、LSB の上位ニブル (4 ビット) はマルチプレクス行のバイトアドレス、MSB の下位ニブル (4 ビット) はマルチプレクス行番号です。生成された component .h ファイルには、各ピクセルの #defines がこの形式で格納されています。

**戻り値:** uint8 pixelState: 指定された PixelNumber の現在の状態を返します。

値	説明
0x00	ピクセルはオフです。
0x01	ピクセルはオンです。
0xFF	ピクセルは割り当てられません。

**副作用:** なし

**void LCD\_Seg\_Sleep(void)**

**説明:** これは、コンポーネントのスリープを準備するのに推奨されるルーチンです。LCD\_Seg\_Sleep() ルーチンは、現在のコンポーネントの状態を保存します。その後、LCD\_Seg\_Stop() 関数と LCD\_Seg\_SaveConfig() 関数を呼び出し、ハードウェア設定を保存します。

CyPmSleep() または CyPmHibernate() 関数を呼び出す前に LCD\_Seg\_Sleep() 関数を呼び出します。電源管理関数については、PSoC Creator *System Reference Guide* を参照してください。

**パラメータ:** なし

**戻り値:** なし

**副作用:** コンポーネントのピンのドライブモードは変更しません。

**void LCD\_Seg\_Wakeup(void)**

**説明:** これは、コンポーネントを LCD\_Seg\_Sleep() が呼び出された時の状態に復元するための、推奨されるルーチンです。LCD\_Seg\_Wakeup() 関数は LCD\_Seg\_RestoreConfig() 関数を呼び出して設定を復元します。LCD\_Seg\_Sleep() 関数を呼び出す前にコンポーネントを有効にすると、LCD\_Seg\_Wakeup() 関数もコンポーネントを再度有効化します。

**パラメータ:** なし

**戻り値:** uint8 cstatus: 標準 API 戻り値

戻り値	説明
CYRET_LOCKED	一部のDMA TDまたはチャンネルは既に使用中
CYRET_SUCCESS	関数は正常終了

**副作用:** 最初に LCD\_Seg\_Sleep() または LCD\_Seg\_SaveConfig() 関数を呼び出すことなく LCD\_Seg\_Wakeup() 関数を呼び出すと、予期せぬ挙動を引き起こすことがあります。

**void LCD\_Seg\_SaveConfig(void)**

**説明:** この関数は、コンポーネントの設定と一時保持レジスタ内容を保存します。この関数は、[Configure] ダイアログで定義されている、または該当する API で変更される現在のコンポーネント・パラメータ値も保存します。この関数は LCD\_Seg\_Sleep() 関数で呼び出します。

**パラメータ:** なし

**戻り値:** なし

**副作用:** なし

**void LCD\_Seg\_RestoreConfig(void)**

**説明:** この関数は、コンポーネントの設定と一時保持レジスタを復元します。この関数はまた、コンポーネントのパラメータ値を LCD\_Seg\_Sleep() 関数を呼び出す前の状態に復旧します。

**パラメータ:** なし

**戻り値:** なし

**副作用:** 最初に LCD\_Seg\_Sleep() または LCD\_Seg\_SaveConfig() 関数を呼び出すことなくこの関数を呼び出すと、予期せぬ挙動を引き起こすことがあります。

**void LCD\_Seg\_Init(void)**

**説明:** [Configure] (設定) ダイアログの設定に従って、コンポーネントのパラメータを初期化または復元します。LCD\_Seg\_Start() ルーチンが LCD\_Seg\_Init() 関数を呼び出すので、この関数を呼び出す必要はありません。これはコンポーネントの動作を開始する際に推奨される方法です。必要なハードウェアブロックをすべて設定し、有効にして、フレーム バッファをクリアします。

**パラメータ:** なし

**戻り値:** なし

**副作用:** 全レジスタは、Configure (設定) ダイアログの設定に従って、値が設定されます。この API は、Timewheel Configuration Register 2 の 1 pulse-per-second ビットを有効にします。これは ローパワ-32kHz 外部 Xtal コンポーネント モードでのみ有効です。

**void LCD\_Seg\_Enable(void)**

**説明:** LCD 固定ハードウェアへの出力を有効にし、UDB 信号の生成を有効にします。

**パラメータ:** なし

**戻り値:** uint8 cystatus: 標準 API 戻り値

戻り値	説明
CYRET_LOCKED	一部のDMA TDまたはチャンネルは既に使用中
CYRET_SUCCESS	正常終了

**副作用:** なし

**オプションのヘルパーAPI**

以下の API は、Configure (設定) ダイアログで対応ヘルパーが選択されている場合にのみ表示されます。

関数	説明
LCD_Seg_Write7SegDigit_n	7セグメント ディスプレイ エLEMENTのアレイ上に 16 進数を表示します。
LCD_Seg_Write7SegNumber_n	7 セグメント ディスプレイ エLEMENTの 1~5 桁のアレイ上に整数値を表示します。
LCD_Seg_WriteBargraph_n	リニアまたは円形(メーター状)バーグラフ上の整数位置を表示します。
LCD_Seg_PutChar14Seg_n	14セグメント英数字ディスプレイ ELEMENTのアレイ上に文字を表示します。
LCD_Seg_WriteString14Seg_n	14セグメント英数字ディスプレイ ELEMENTのアレイ上にヌル終端文字列を表示します。
LCD_Seg_PutChar16Seg_n	16セグメント英数字ディスプレイ ELEMENTのアレイ上に文字を表示します。
LCD_Seg_WriteString16Seg_n	16セグメント英数字ディスプレイ ELEMENTのアレイ上にヌル終端文字列を表示します。
LCD_Seg_PutCharDotMatrix_n	ドットマトリクス英数字ディスプレイ ELEMENTのアレイ上に文字を表示します。

関数	説明
LCD_Seg_WriteStringDotMatrix_n	ドットマトリクス英数字ディスプレイ エLEMENTのアレイ上にヌル終端文字列を表示します。

**注** サフィックス「n」を含む関数名は、コンポーネント マスタマイザ内で同じシンボル タイプのディスプレイ ヘルパーが複数作成されたことを示します。個々のディスプレイ ヘルパー エLEMENTは、関数名に「n」サフィックスが含まれる API 関数で制御されます。

### void LCD\_Seg\_Write7SegDigit\_n(uint8 digit, uint8 position)

**説明:** この関数は、7セグメント ディスプレイ エLEMENTのアレイ上に 16 進数を表示します。各桁は、0～9 と A～F の範囲の 16 進値です。7 セグメントのディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能を使用する必要があります。フレームバッファ内に複数の 7 セグメント ディスプレイ エLEMENTを定義し、関数名内のサフィックス (n) を通してアドレス設定できます。この関数は、コンポーネント カスタマイザ内で7セグメント ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** uint8 digit: 16進数として表示される、0から15の間の符号なし整数値  
uint8 position: 右端桁を 0 として右から左に数えた桁位置。位置が定義されている表示領域の外側であれば、表示されません。

**戻り値:** なし

**副作用:** なし

**void LCD\_Seg Write7SegNumber\_n(uint16 value, uint8 position, uint8 mode)**

**説明:** この関数は、7セグメント ディスプレイ エLEMENTの 1～5 桁のアレイ上に 16 ビット整数値を表示します。7セグメント ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の7セグメント ディスプレイ エLEMENT グループをフレーム バッファ内に定義し、関数名内のサフィックスを使ってアドレス設定することができます。符号変換、符号表示、小数点、その他のカスタム機能は、アプリケーション特有のユーザ コードで取り扱う必要があります。この関数は、コンポーネント カスタマイザ内で7セグメント ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** uint16 value: 表示する符号なし整数値。

uint8 position: 右端桁を0として右から左に数えた最下位桁位置。定義された表示領域の桁数が、値が必要とする桁数より少ない場合は、最上位桁が何桁か表示されないことがあります。

uint8 mode: ディスプレイ モードを設定します。0または1になります。

値	説明
0	頭の0を表示しません。
1	頭の0を表示します

**戻り値:** なし

**副作用:** なし



**void LCD\_Seg\_WriteBargraph\_n(uint8 location, uint8 mode)**

**説明:** この関数は、1 ～ 255 セグメントのバーグラフ(左から右)上の 8 ビット整数位置を表示します。バーグラフは 1 ～255 セグメントの範囲でユーザが定義したサイズとなります。バーグラフは、回転位置を表示するために円形にすることもできます。バーグラフ ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能を使用する必要があります。フレーム バッファ内で複数のバーグラフ ディスプレイ エLEMENTを定義し、関数名内のサフィックス (n) を通してアドレス設定できます。この関数は、コンポーネント カスタマイザ内でバーグラフ ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** uint8 location: 表示する符号なし整数位置。有効な値は、ゼロからバーグラフ内のセグメント数までです。ゼロの値は、すべてのバーグラフ エLEMENTをオフにします。バーグラフ内のセグメント数を超える値を指定すると、全ELEMENTがオンになります。

uint8 mode: バーグラフ ディスプレイ モードを設定します。

値	説明
0	指定した位置のセグメントがオンになります。
1	指定した位置のセグメントとその左側のすべてのセグメントがオンになります。
-1	指定した位置のセグメントとその右側のすべてのセグメントがオンになります。
2 ~ 10	指定した位置のセグメントとその右側 2～10 セグメントが表示されます。このモードは幅広い表示器の作成に使用します。

**戻り値:** なし

**副作用:** なし

**void LCD\_Seg\_PutChar14Seg\_n(uint8 character, uint8 position)**

**説明:** この関数は、14セグメント英数字ディスプレイ エLEMENTのアレイ上に 8 ビット文字を表示します。14セグメント ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の14セグメント ディスプレイ エLEMENT グループをフレーム バッファ内に定義し、関数名内のサフィックス (n) を使ってアドレス設定することができます。この関数は、コンポーネント カスタマイザ内で14セグメント ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** uint8 character: 表示する文字のASCII値 (ASCII 値 0 ～ 127 の印刷可能文字)

uint8 position: 文字の位置は左端を0として左から右に数えます。位置が定義されている表示領域の外側であれば、表示されません。

**戻り値:** なし

**副作用:** なし

**void LCD\_Seg\_WriteString14Seg\_n(\*uint8 character, uint8 position)**

**説明:** この関数は、14セグメント英数字ディスプレイ エLEMENTのアレイ上にヌル終端文字列を表示します。14セグメント ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の 14セグメント ディスプレイ エLEMENT グループをフレーム バッファ内に定義し、関数名内のサフィックス (n) を使ってアドレス設定することができます。この関数は、コンポーネント カスタマイザ内で14セグメント ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** \*uint8 character: ヌル終端文字列を示します。

uint8 position: 最初の文字の位置は左端を0として左から右に数えます。文字列の長さが定義した表示領域のサイズを超える場合は、表示領域に入りきらない文字は表示されません。

**戻り値:** なし

**副作用:** なし

**void LCD\_Seg\_PutChar16Seg\_n(uint8 character, uint8 position)**

**説明:** この関数は、16セグメント英数字ディスプレイ エLEMENTのアレイ上に8ビット文字を表示します。16セグメント ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の16セグメント ディスプレイ エLEMENT グループをフレーム バッファ内に定義し、関数名内のサフィックス (n) を使ってアドレス設定することができます。この関数は、コンポーネント カスタマイザ内で16セグメント ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** uint8 character: 表示する文字のASCII値 (値が 0 ~ 255 の印刷可能ASCIIおよび表拡張文字)。

uint8 position: 文字の位置は左端を0として左から右に数えます。位置が定義されている表示領域の外側であれば、表示されません。

**戻り値:** なし

**副作用:** なし

**(void) LCD\_Seg\_WriteString16Seg\_n(\*uint8 character, uint8 position)**

**説明:** この関数は、16セグメント英数字ディスプレイ エLEMENTのアレイ上にヌル終端文字列を表示します。16セグメント ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の16セグメント ディスプレイ エLEMENTグループをフレーム バッファ内に定義し、関数名内のサフィックス (n) を使ってアドレス設定することができます。この関数は、コンポーネント カスタマイザ内で 16セグメント ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** \*uint8 character: ヌル終端文字列を示します。

uint8 position: 最初の文字の位置は左端を0として左から右に数えます。文字列の長さが定義した表示領域のサイズを超える場合、表示領域に入りきらない文字は表示されません。

**戻り値:** なし

**副作用:** なし



**void LCD\_Seg\_PutCharDotMatrix\_n(uint8 character, uint8 position)**

**説明:** この関数は、ドットマトリクス英数字ディスプレイ エLEMENTのアレイ上に 8 ビット文字を表示します。ドットマトリクス ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の ドットマトリクス ディスプレイ エLEMENT グループをフレーム バッファ内に定義し、関数名内のサフィックス (n) を使ってアドレス設定することができます。この関数は、コンポーネント カスタマイザ内でドットマトリクス ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** uint8 character: 表示する文字の ASCII 値。

uint8 position: 文字の位置は左端を0として左から右に数えます。位置が定義されている表示領域の外側であれば、表示されません。

**戻り値:** なし

**副作用:** なし

**void LCD\_Seg\_WriteStringDotMatrix\_n(\*uint8 character, uint8 position)**

**説明:** この関数は、ドットマトリクス英数字ディスプレイ エLEMENTのアレイ上にヌル終端文字列を表示します。ドットマトリクス ディスプレイ エLEMENTに関連付けられているピクセル群を定義するには、カスタマイザ ディスプレイ ヘルパー機能が必要です。複数の ドットマトリクス ディスプレイ エLEMENT グループをフレーム バッファ内に定義し、関数名内のサフィックス (n) を使ってアドレス設定することができます。この関数は、コンポーネント カスタマイザ内でドットマトリクス ディスプレイ エLEMENTが定義されている場合にのみ含まれます。

**パラメータ:** \*uint8 character: ヌル終端文字列を示します。

uint8 position: 最初の文字の位置は左端を0として左から右に数えます。文字列の長さが定義した表示領域のサイズを超える場合は、表示領域に入りきらない文字は表示されません。

**戻り値:** なし

**副作用:** なし

**ピンAPI**

これらの API 関数は、セグメント LCD コンポーネントで使用されるピンのドライブ モードを変更するために使用します。

関数	説明
LCD_Seg_ComPort_SetDriveMode	セグメントLCDコンポーネントのコモン ラインとして使用されるすべてのピンのドライブ モードを設定します。
LCD_Seg_SegPort_SetDriveMode	セグメントLCDコンポーネントのセグメント ラインとして使用されるすべてのピンのドライブ モードを設定します。

**void LCD\_Seg\_ComPort\_SetDriveMode(uint8 mode)**

説明:	セグメントLCDコンポーネントのコモンラインとして使用されるすべてのピンのドライブモードを設定します。
パラメータ:	uint8 mode: 使用するドライブモード。ドライブモードの詳細は、ピンコンポーネントデータシートを参照してください。
戻り値:	なし
副作用:	なし

**LCD\_Seg\_SegPort\_SetDriveMode(uint8 mode)**

説明:	セグメントLCDコンポーネントのセグメントラインとして使用されるすべてのピンのドライブモードを設定します。
パラメータ:	uint8 mode: 使用するドライブモード。ドライブモードの詳細は、ピンコンポーネントデータシートを参照してください。
戻り値:	なし
副作用:	なし

**マクロ**

- LCD\_Seg\_COMM\_NUM – コンポーネントの現在の設定におけるユーザ定義ディスプレイのコモンラインの数を定義します。
- LCD\_Seg\_SEG\_NUM – コンポーネントの現在の設定におけるユーザ定義ディスプレイのセグメントラインの数を定義します。
- LCD\_Seg\_BIAS\_TYPE – コンポーネントの現在の設定におけるユーザ定義ディスプレイのバイアスタイプを定義します。
- LCD\_Seg\_BIAS\_VOLTAGE – ユーザ定義ディスプレイのデフォルト バイアス電圧レベルを定義します。この値は、初期化プロセス中に LCDDAC コントロール レジスタで設定されます。
- LCD\_Seg\_FRAME\_RATE – コンポーネントの現在の設定におけるユーザ定義ディスプレイのリフレッシュ レートを定義します。
- LCD\_Seg\_EXTRACT\_ROW – フレーム バッファ内の指定ピクセルの行を計算します。
- LCD\_Seg\_EXTRACT\_PORT – フレームバッファ内の指定ピクセルのバイトオフセットを計算します。
- LCD\_Seg\_EXTRACT\_PIN – フレームバッファ内の指定ピクセルのビット位置を計算します。
- LCD\_Seg\_WRITE\_PIXEL – これは void 型の関数 LCD\_Seg\_WritePixel() のマクロ定義です。

- LCD\_Seg\_READ\_PIXEL – これは LCD\_Seg\_ReadPixel() 関数のマクロ定義です。
- LCD\_Seg\_FIND\_PIXEL – このマクロはフレームバッファ内のピクセル位置を計算します。カスタマイズのピクセルテーブル、および LCD 専用の物理ピンに関する情報を用います。このマクロはピクセルマッピング機構の基礎を成します。ピクセル テーブルの各ピクセル名は、フレームバッファ内の計算されたピクセル位置で定義されます。API はピクセル名を使用して、それぞれのピクセルにアクセスします。

## ファームウェアのソースコードのサンプル

PSoC Creator は、Find Example Project ダイアログに数多くのサンプルプロジェクトを提供しており、そこには回路図およびコード例が含まれています。コンポーネント特有の例を見るには、[Component Catalog] または回路図に置いたコンポーネント インスタンスからダイアログを開きます。一般例については、[Start Page] または **[File (ファイル)]** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

詳しくは、PSoC Creator ヘルプの「Find Example Project (サンプルプロジェクトを検索)」を参照してください。

**注** 特化したコンポーネント実装のため、ローパワーモードでは、CyPmSaveClocks() および CyPmRestoreClocks() API でこのコンポーネントを使用することはできません。その他の場合では、CyPmSaveClocks() および CyPmRestoreClocks() API を使用できます。

## 機能の説明

セグメント LCD コンポーネントは、タイプの異なる LCD ガラスを強力に柔軟に駆動する機構を提供します。Configuration (設定) ダイアログから、コンポーネントの機能をカスタマイズするのに使用できるパラメータにアクセスできます。標準 API ルーチン群により、ディスプレイと指定ピクセルを制御します。定義されたディスプレイ ヘルパーのタイプと数に基づいて追加のディスプレイ API が生成されます。

## デフォルト設定

LCD\_Seg コンポーネントのデフォルト設定は、汎用 LCD ダイレクト セグメント ドライブ コントローラを提供します。デフォルト LCD\_Seg 設定:

- 4 コモン ライン
- 8 セグメント ライン
- 1/3 バイアス モード
- 60 Hz リフレッシュ レート
- No Sleep power mode (スリープなしパワー モード)



- 3 V バイアス電圧
- LCD ガラスサイズ: 10 平方 cm
- 968.99  $\mu$ s ハイドライブ時間
- seg - 1x、com - 2x ハイドライブ強度
- 7.57  $\mu$ s ロードライブ時間
- seg – 0.06x、com 0.12x ロードライブ 強度
- ディスプレイ ヘルパーの定義なし。デフォルト API 生成時、サポートされているディスプレイ エLEMENTの関数含まず。

## カスタム設定

セグメント LCD コンポーネントの主な特長は異なる文字やレイアウトを持つ LCD の柔軟なサポートです。

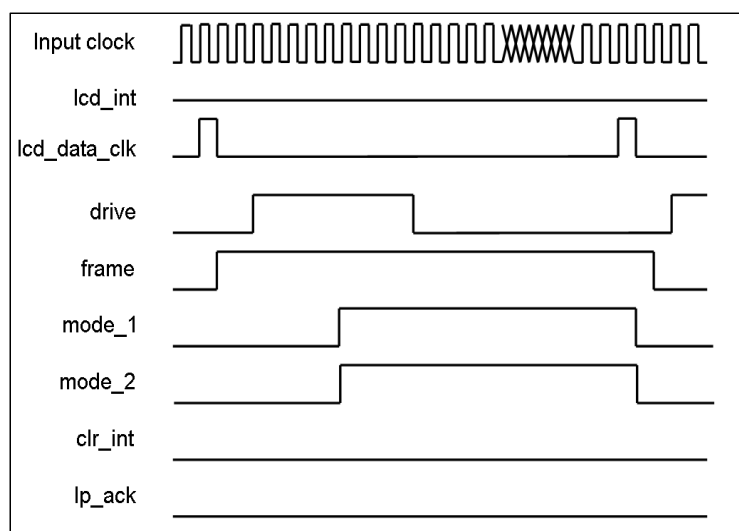
## ドライバ パワー モード

セグメント LCD は、No Sleep (スリープなし) と Low Power (ローパワー) の 2 つのパワーモードで動作できます。No Sleep (スリープなし) がデフォルト動作モードです。

### No Sleep (スリープなし) モード

このモードでは、セグメントLCDは決してスリープ モードにならず、LCD はフレーム全体を通して駆動されます。図 1 に、No Sleep (スリープなし) モードでの LCD\_Seg コンポーネントの UDB 生成 (内部) 信号の波形を示します。

図 1. セグメント LCD コントロール信号 (No Sleep (スリープなし) モード)



### ローパワーモード

このモードでは、LCD は電圧遷移時にのみアクティブに駆動され、LCD システムのアナログ コンポーネントは電圧遷移と次の電圧遷移との間にはスリープ モードになります。内部 LCD タイマは、LCD 画面を更新するためにデバイスをスリープ モードから短時間ウェイクアップします。その後、内部ロジックは次のリフレッシュ シーケンスまでデバイス全体をスリープ モードに戻します。

ローパワー モードでは、Frame Rate (フレームレート) パラメータは制限され、LCD タイマ用のクロックソース (1 kHz [ILO] または 8 kHz [32 kHz クリスタル])、コモンライン数、波形タイプなど、その他のパラメータに依存します。フレームレートの制限は、LCD タイマの入力周波数の制限によって説明できます。以下に、8 個のコモン、ローパワーILO モード、タイプ A 波形の設定用の最大フレームレートの計算例を示します。

$$\text{max FR} = 1000 / (2 * \text{Num Commons}) = 62.5 \text{ Hz}$$

LCD タイマが入力周波数の各クロックサイクルでウェイクアップ信号を生成する場合、62.5 Hz が提供できる最大フレームレートです。

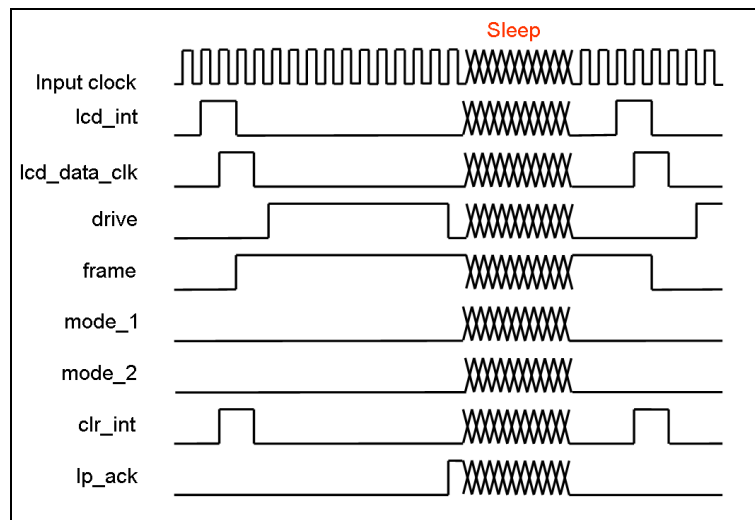


実際には、LCD タイマはウェイクアップ イベントの設定に 1 期間とそれをクリアするために 1 期間必要なだけです。これにより、最大フレームレートを 2 で割ることにより、31.25 Hz が得られます。この値の整数部分を取り、実際の最大フレームレートとして 31Hz が得られます。

タイプ B 波形には 1/2 と少ないウェイクアップ イベントしか必要ないので、タイプ B 波形の実際のフレームレートはタイプ A 波形の 2 倍となります。

図 2 に、ローパワーモードでの LCD\_Seg コンポーネントの UDB 生成信号の波形を示します。

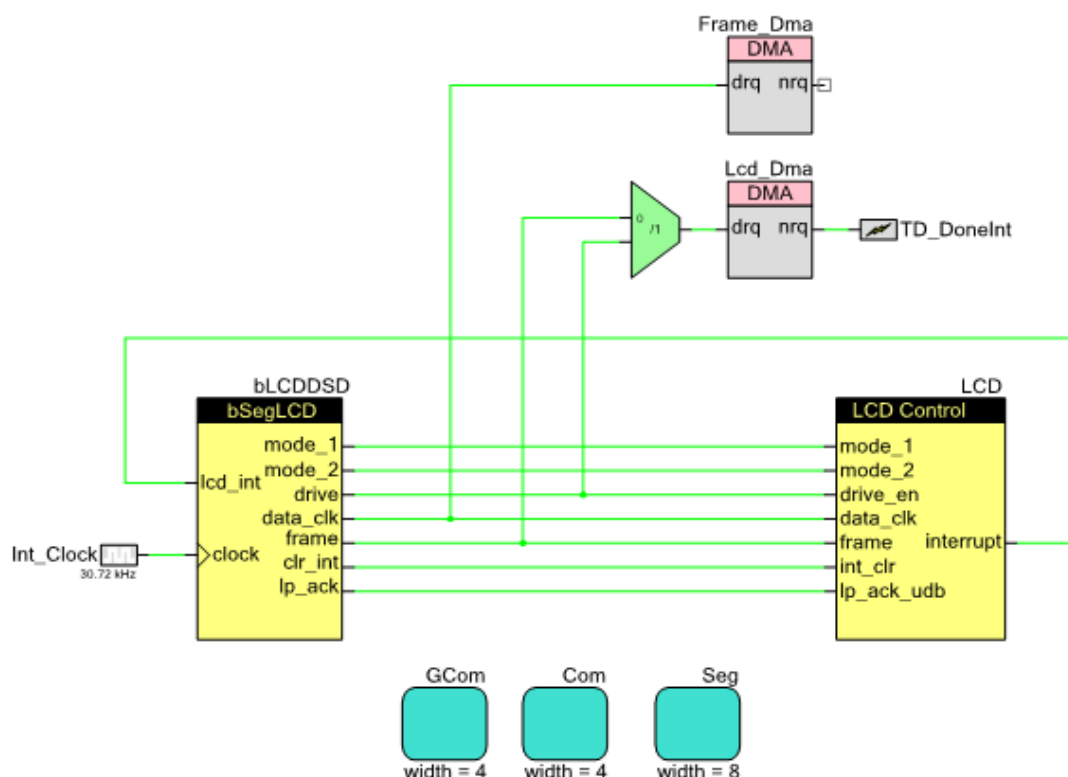
図 2. セグメント LCD コントロール信号 (ローパワーモード)



## ブロックダイアグラムと設定

図 3 に、セグメント LCD コンポーネントの内部回路図を示します。これは、基本セグメント LCD コンポーネント、LCD コントロール ブロック (LCD) コンポーネント、DMA コンポーネント、3 つの LCD ポート、1 つのデジタル ポート、1 つの ISR コンポーネント、1 つのクロックで構成されています。

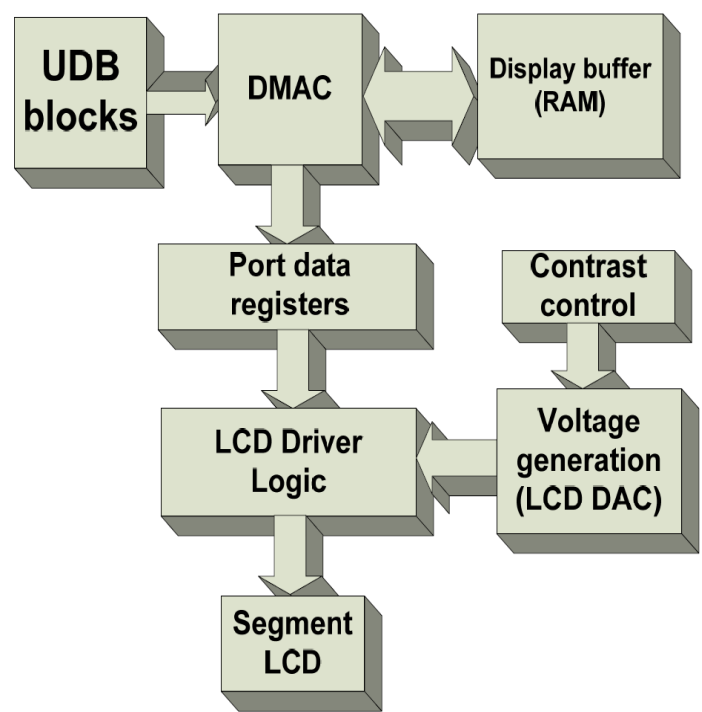
図 3. セグメント LCD コンポーネントの回路図



- 基本セグメント LCD コンポーネントは、LCD ポートと DMA コンポーネントの適正なタイミング信号の生成を行います。
- DMA コンポーネントは、フレームバッファからエイリアス メモリ領域経由でデータを LCD データ レジスタに転送するために使用します。
- LCD コンポーネントは、必要な DSI ルーティングを処理します。このブロックはまた、必要なレジスタ名を *cyfitter.h* ファイル中に定義して提供します。
- LCD ポート (GCom、Com、Seg) は、論理信号を物理ピンにマップするために使用します。LCD ポートには 2 つのインスタンスがあります。その 1 つはコモン ラインでもう 1 つはセグメント ラインです。コモン信号用の LCD ポートは 16 ピン幅に制限されており、セグメント信号用の LCD ポートは 48 ピン幅に制限されています。また、追加のコモン ポート GCom もありますが、これは Ganging (連結) オプションを有効にした場合にのみ含まれます。

トップ レベル アーキテクチャ

図 4. セグメント LCD トップ レベル



レジスタ

LCD\_Seg\_CONTROL\_REG

ビット	7	6	5	4	3	2	1	0
値	フレーム	予約済み			フレーム終了	モード2	モード1	クロック有効

- クロック有効:これまでのセクションで説明したすべての内部信号の生成を有効にします。
- モード 1: mode[2:0] ビット フィールド中央のビット。ハイ ドライブ強度とロー ドロイブ強度を定義します。
- モード 2: mode[2:0] ビット フィールドの上位ビット。ハイ ドライブ強度とロー ドロイブ強度を定義します。
- フレーム終了:DMA フレーム トランザクション完了後、同期パルスを生成します。
- フレーム:LCD ドライバ用のフレーム信号を生成します。

**LCD\_Seg\_CONTRAST\_CONTROL\_REG**

適正なバイアス電圧を生成するために LCD DAC で使用するバイアス電圧レベルを保存します。バイアス電圧レベルを変更するための API が提供されています。

ビット	7	6	5	4	3	2	1	0
値	予約済み	予約済み	コントラスト レベル					

- コントラスト レベル: 上記のバイアス電圧レベル。

**LCD\_Seg\_LCDDAC\_CONTROL\_REG**

ビット	7	6	5	4	3	2	1	0
値	lp イネーブル	予約済み			連続ドライブ	予約済み	バイアス選択	

- バイアス選択: バイアスを選択。
- 連続ドライブ: チップがスリープ状態になっても LCDDAC がアクティブ状態を維持できるようにします。
- lp イネーブル: UDB が LCD サブシステム用の Low Power Ack をゲートできるようにします。

**LCD\_Seg\_TIMER\_CONTROL\_REG**

ビット	7	6	5	4	3	2	1	0
値	期間						clk select (クロック選 択)	enable timer (タイマのイ ネーブル)

- enable timer (タイマのイネーブル) LCD タイマを有効にします。
- clk select (クロック選択): LCD タイマソース選択クロック。
- 期間: LCD タイマ期間。

**LCD\_Seg\_DRIVER\_CONTROL\_REG**

ビット	7	6	5	4	3	2	1	0
値	予約済み			バイパス イネーブル	pts	反転	モード0	スリープ モード

- スリープ モード: ローパワー モードでは、「1」に設定されている場合は LCD ドライバの出力バッファは接地に設定され、それ以外の場合は HI-Z に設定されます。
- モード 0: mode[2:0] ビット フィールドの下位ビット。ハイ ドライブ強度とロー ドライブ強度を定義します。
- 反転: 設定すると、セグメント ピン上のデータを反転します。



- pts: クリアして「0」 - 通常動作にすると、 $V_{OUT} = V_{IO} - 0.5V$ 。「1」に設定すると、 $V_{OUT} = V_{IO}$ 。
- バイパス イネーブル:「1」に設定すると、LCD ドライバ内のドライブ バッファをバイパスし、代わりに選択された電圧入力に直接接続します。クリアして「0」 - 標準動作にすると、電圧はドライバ バッファを介して渡されます。

**LCD\_Seg\_LCDDAC\_SWITCH\_REG[0..4]**

ビット	7	6	5	4	3	2	1	0
値	予約済み						スイッチ コントロール[0..4]	

- スイッチ コントロール[0..4] このビットフィールド群は LCD ドライバの電圧源を選択します。

**DC 電気的特性と AC 電気的特性****LCD 直接駆動 DC 仕様**

パラメータ	説明	条件	Min	Typ	Max	単位
$I_{CC}$	LCDシステム動作電流	デバイススリープモード、400 Hz でウェイクアップして LCD をリフレッシュ、バス クロック = 3 MHz、 $V_{DDIO} = V_{DDA} = 3V$ 、4 コモン、16 セグメントライン、1/4 デュティサイクル、50 Hz フレームレート、LCD ガラス接続なし	—	38	—	$\mu A$
$I_{CC\_SEG}$	セグメント ドライバあたりの電流	ストロングドライブモード	—	260	—	$\mu A$
$V_{BIAS}$	LCD バイアス範囲 ( $V_{BIAS}$ はLCD DACのメイン出力電圧 ( $V_0$ ))	$V_{DDA} \geq 3V$ および $V_{DDA} \geq V_{BIAS}$	2	—	5	V
	LCD バイアスステップサイズ	$V_{DDA} \geq 3V$ および $V_{DDA} \geq V_{BIAS}$	—	$9.1 \times V_{DDA}$	—	mV
	セグメント/コモン ドライバあたりの LCD 静電容量	ドライバは連結可能	—	500	5000	pF
	長時間セグメント オフセット		—	—	20	mV
$I_{OUT}$	セグメント ドライバあたりの出力駆動電流	$V_{DDIO} = 5.5V$ 、ストロングドライブモード	355	—	710	$\mu A$

**LCD ダイレクト ドライブ AC 仕様**

パラメータ	説明	条件	Min	Typ	Max	単位
$f_{LCD}$	LCD フレームレート		10	50	150	Hz

## コンポーネントの変更

ここでは、過去のバージョンからコンポーネントに加えられた主な変更を示します。

バージョン	変更の説明	変更の理由 / 影響
3.0	WRITE_PIXEL() マクロ定義を、 #define LCD_Seg_WRITE_PIXEL(pixelNumber, pixelState) LCD_Seg_WritePixel(pixelNumber, pixelState) から #define LCD_Seg_WRITE_PIXEL(pixelNumber, pixelState) (void)LCD_Seg_WritePixel(pixelNumber, pixelState) へ変更	コンポーネント内での処理が実現され LCD_Seg_WritePixel() API の戻り値を解析する必要はないので、マクロはそうに(void型に)変更されました。
	コンポーネントをローパワー モードで使用すると現れる STA の組み合わせサイクル警告を避けるため、コンポーネントの verilog 実装を変更	非同期 SR ラッチが同期ロジックに置き換えられました。
	割り当て解除されたヘルパー ピクセルの確認が LCD_Seg_ReadPixel() API に追加	0xFF の新しいリターン ステータス (LCD_Seg_Stat_PIXEL_UNKNOWN_STATE) が LCD_Seg_ReadPixel() に追加されました。このステータスは、ピクセルがヘルパーに存在するのに割り当てられていない場合に、返されます。
	関数 LCD_Seg_WriteBargraph()の最適化	リファクタリング(コード再利用等)により、フラッシュ メモリ内のスペースが節約されました。
	関数 LCD_Seg_WriteString14Seg() および LCD_Seg_WriteString16Seg()の最適化	配列のインデックス方式をポインタのインクリメント方式に変更されたことにより、RAM スペースが節約され、これらの関数の実行時間が向上します。
	API 対 LCD_Seg_Write7SegNumber()の最適化	LCD_Seg_Write7SegNumber() API の一部に無駄なコードがあり、この API の実行時間が長くなっていました。
	LCD_Seg_Wakeup() API への修正	LCD_Seg_Wakeup() API は返すべきステータスを返さないで、適切なステータスを返すように修正されました。
2.10	LCD_Seg_Start() ルーチン修正	実装にエラーがあったため、グローバル割り込みが前にイネーブルされていた場合、これを黙ってディスエーブルしていました。
	カスタマイズの Driver Power Settings (ドライバ パワー設定) タブを変更、2 つの新しいフィールド Custom Step (カスタム ステップ) と Default Step (デフォルト ステップ) を追加	ローパワー モードでの High Drive Time (ハイドライブ 時間) パラメータの新しいオプションが使用可能です。現状選択できるデフォルト ステップに加え、より精密なカスタム ステップを選択できるようになりました。これはマスタ クロックの周波数に基づきます。

バージョン	変更の説明	変更の理由 / 影響
	LCD リフレッシュ シーケンスの完了前に SegLCD がスリープ状態になるという問題を修正	リフレッシュ シーケンスの開始時に内部コンポーネント信号 lp_ack がハイになっていましたが、正しくはリフレッシュ シーケンスの終了時にハイになる必要があります。
	ハイドライブ強度が “seg=2x com=2x” または “seg=1x com=4x” に設定されたときに LCD ガラス画像が反転するという問題を修正	この問題は、LCD ドライバ コントロール レジスタに不正な値が書き込まれることに起因していました。この値はマスクとのAND演算で生成されていましたが、このマスクが正しくありませんでした。このマスクを正しいものに変えることでこの問題が修正できました。
	LCD_Seg_WriteBargraph() API を修正	モードパラメータが 10 に設定されたときに LCD のバークラフピクセルの不正な出力として反映されていた LCD_Seg_WriteBargraph() API の問題を修正しました。これは、ポジションパラメータが “Position + 10 > Bar Graph pixels” (位置 + 10 > バークラフピクセル数) という条件を満たすときに発生していました。この場合、バークラフの最後のピクセルが設定されていませんでしたが、これは設定する必要がありました。
	データシートに特性データを追加	
	シリコン リビジョンとの互換性をアピールする情報をコンポーネントに追加	このツールは互換性のないシリコンでこのコンポーネントを使うとエラーを表示します。エラーが表示されたら、対象デバイスをサポートするリビジョンにアップデートしてください。
	データシートのマイナーな編集と更新	
2.0.a	データシートの LCD_Seg_Init()、LCD_Seg_Start()、LCD_Set_Stop() の API に注を追加	
	シリコン リビジョンとの互換性をアピールする情報をコンポーネントに追加	このツールは互換性のないシリコンでこのコンポーネントを使うとエラーを表示します。エラーが表示されたら、対象デバイスをサポートするリビジョンにアップデートしてください。
2.0	Sleep/Wakeup (スリープ/ウェイクアップ) と Init/Enable (初期化/イネーブル) API を追加	ローパワーモードをサポートし、ほとんどのコンポーネントの初期化とイネーブル化の制御を分離する共通インターフェースを提供するためです。
	スリープモードAPIの宣言を含む新しい API ファイル - SegLCD_PM.c を追加	ローパワーモードをサポートするための新しい要件です。



バージョン	変更の説明	変更の理由 / 影響
	<p>PSoC 3 ES3 以降をサポートするようにコンポーネントをアップデート、[Configure]ダイアログを以下のように更新:</p> <ul style="list-style-type: none"> <li>■ 削除された古いコントロール: Enable Debug Mode (デバッグモードを有効)、Low Drive Mode (ロードライブモード)</li> <li>■ 追加された新しいコントロール: Glass Size (ガラスサイズ)、High Drive Strength (ハイドライブ強度)、Low Drive Strength (ロードライブ強度)</li> <li>■ 古いモード Always Active (常に有効)と Low Power (ローパワー) を新しいモード No Sleep (スリープなし)、Low Power ILO (ローパワー ILO)、Low Power 32XTAL (ローパワー 32XTAL) に変更することで、ドライバパワーモードの選択を変更</li> <li>■ Custom Characters (カスタム文字) タブを追加し、ユーザがドットマトリクスヘルパー用のカスタム文字を作成可能に</li> <li>■ バイアスタイプコントロールを編集可能とし、バイアス電圧の選択値を変更</li> </ul>	<p>PSoC 3 ES3 デバイスと LCD HW アーキテクチャをサポートするという新しい要件を満たすために新しい 2.0 バージョンが作成されました。</p> <p>バージョン 1.xx は PSoC 3 ES2 および PSoC 5 シリコン改版をサポートします。</p>
	<p>Added `=ReentrantKeil(LCD_Seg_ . "_...")」を次の関数に追加</p> <p>LCD_Seg_Stop()  LCD_Seg_EnableInt()  LCD_Seg_DisableInt()  LCD_Seg_SetBias()  LCD_Seg_WriteInvertState()  LCD_Seg_ReadInvertState()  LCD_Seg_RedPixel()  LCD_Seg_SaveConfig()  LCD_Seg_RestoreConfig()</p>	<p>必要な場合に、これらの API を再入可能にするためです。</p>

Copyright © 2005-2012 Cypress Semiconductor Corporation. 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporation は、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することはありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものではありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™ 及び Programmable System-on-Chip™ は、Cypress Semiconductor Corp. の商標、PSoC® は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)は Cypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンスに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンスの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項: サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。

