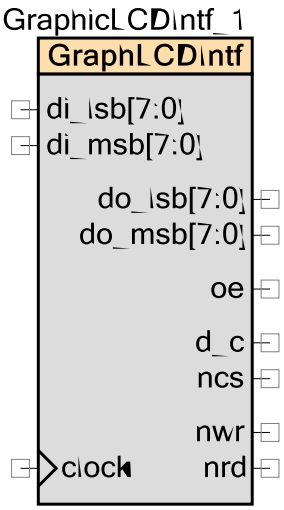
グラフィック LCD インターフェース (GraphicLCDIntf) 1.60

特長



- グラフィック LCD コントローラへの 8 または 16 ビットのインターフェース
- 多くのグラフィック コントローラデバイスに対応
- 読み取りおよび書き込みトランザクション
- 読み込み「LOW」パルス幅では、2ー255 サイクル
- 読み込み「HIGH」パルス幅では、1ー255 サイクル
- 典型的な i8080 インターフェースを実装

概要説明

グラフィック LCD インターフェース (GraphicLCDIntf) コンポーネントは、グラフィック LCD コントローラとドライバデバイスのインターフェースを提供します。これらのデバイスは、一般に、LCD パネルに統合されます。これらのデバイスへのインターフェースは、一般に、i8080 インターフェースと呼ばれています。これは、Intel i8080 マイクロプロセッサの歴史的なパラレル バス インターフェース プロトコルの参照です。

GraphicLCDIntf を使用する場合

LCD コントローラとドライバデバイスは、一般に、LCD パネルに統合されます。表示用のフレーム バッファへのインターフェースに含まれる、またはこのようなインターフェースを提供し、そのバッファを管理します。GraphicLCDIntf コンポーネントは、このコントローラとの読み取りおよび書き込みを実行します。これらのトランザクションは、以下のパラメータを持ちます。

- 読み取りまたは書き込み
- Address (アドレス)。d_c ピンで駆動される 1 ビットのアドレス
- Data (データ) (8 または 16 ビット)。書き込みでは「do」で送信、読み取りでは「di」で読み取られます。

GraphicLCDIntf コンポーネントは、多数のコントローラに対応します。このコンポーネントの設定では、3 つのパラメータを使用できます。

- クロック周波数: このコンポーネントを駆動するクロックの周波数は、書き込み信号の最小パルス幅「LOW」によって、しばしば制限されます。(この値は、該当するグラフィック LCD コントローラのデータシートに記載されています。) 書き込みパルスは、単一のクロック期間では「LOW」であるため、この要件を満たすようにクロック周波数を設定してください。
- 読み取りパルス幅「HIGH」: カスタマイザのこの設定は、クロック周期で測定されます。クロック期間にパルス幅「HIGH」で設定された周期数を掛けた値は、コントローラの読み取りパルス幅「HIGH」の要件を満たす必要があります。
- 読み取りパルス幅「LOW」: この設定は、読み取りパルス幅「HIGH」設定と同じ方法で実行されます。読み取りパルス幅「LOW」のタイミングは、読み取りパルス幅と読み取りアクセス時間の要件を満たす必要があります。データは、アクティブな「LOW」パルス幅が終わる 1 クロック周期前にサンプリングされるため、パルス幅は、十分なアクセス時間を持つよう設定する必要があります。

以下に、該当する LCD コントローラの設定を示します。

Solomon Systech SSD1289

- クロック周波数: 20 MHz (50 ns)
- 読み取りパルス幅「HIGH」: 10 クロック周期 (500 ns)
- 読み取りパルス幅「LOW」: 10 クロック周期 (500 ns)

Solomon Systech SSD2119

- クロック周波数: 25 MHz (40ns)
- 読み取りパルス幅「HIGH」: 13 クロック周期 (500 ns)
- 読み取りパルス幅「LOW」: 13 クロック周期 (500 ns)

Himax HX8347A

- クロック周波数: 28.5 MHz (35 ns)
- 読み取りパルス幅「HIGH」: 3 クロック周期 (105 ns)
- 読み取りパルス幅「LOW」: 11 クロック周期 (385 ns)

ILITEK ILI9325

- クロック周波数: 20 MHz (50 ns)

- 読み取りパルス幅「HIGH」: 3 クロック周期 (150 ns)
- 読み取りパルス幅「LOW」: 3 クロック周期 (150 ns)

Epson S1D13743

- クロック周波数 : 33 MHz (33.3 ns)
- 読み取りパルス幅「HIGH」: 2 クロック周期 (67 ns)
- 読み取りパルス幅「LOW」: 5 クロック周期 (167 ns)

入出力接続

ここでは、GraphicLCDIntf コンポーネントのさまざまな入出力接続について説明します。I/O リストのアスタリスク (*) は、その I/O の説明でリストされている条件において、記号に隠れている可能性があることを示します。

clock (クロック)

このコンポーネントを操作するクロック。このコンポーネントは、コンポーネントに接続されている単一のクロックから完全に操作されます。

di_lsb[7:0]

入力データ バスの下位 8 ビット。読み取りトランザクション中のデータで使用されます。

これらの信号は、デバイスの入力ピンに接続し、これらのピンでは「同期入力」を無効にする必要があります。信号は、同期出力信号を基にして駆動されるため、信号そのものは派生的に同期化されます。

di_msb[7:0] *

入力データ バスの上位 8 ビット。読み取りトランザクション中のデータで使用されます。16 ビット インターフェース モードでのみ存在します。

これらの信号は、デバイスの入力ピンに接続し、これらのピンでは「同期入力」を無効にする必要があります。信号は、同期出力信号を基にして駆動されるため、信号そのものは派生的に同期化されます。

do_lsb[7:0]

出力データ バスの下位 8 ビット。書き込みトランザクション中のデータで使用されます。

do_msb[7:0] *

出力データ バスの上位 8 ビット。書き込みトランザクション中のデータで使用されます。16 ビット インターフェース モードでのみ存在します。

oe

データ バスの出力イネーブル。通常、データ バスでは、入力/出力ピン コンポーネントの出力イネーブルに接続されます。この信号が使用される方法については、「図解マクロ」を参照してください。

d_c

データ/コマンド信号。「HIGH」の場合はデータ トランザクション、「LOW」の場合はコマンド トランザクションを示します。

ncs

アクティブ「LOW」チップの選択。

nwr

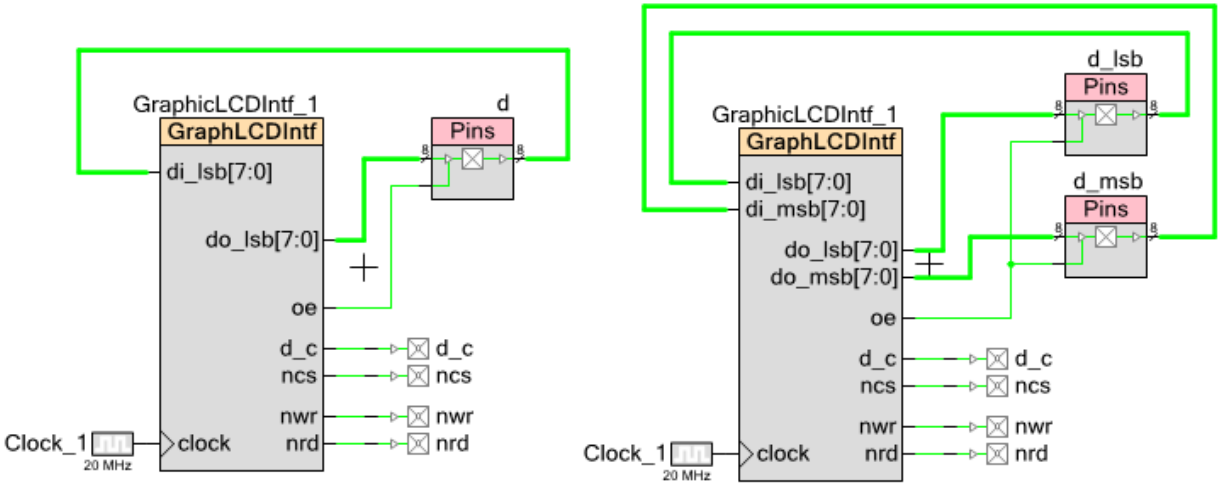
アクティブ「LOW」書き込み制御信号。

nrd

アクティブ「LOW」読み取り制御信号。

回路マクロ情報

カタログには、標準の記号エントリに加え、2 つのマクロが提供されています。1 つは、ピンおよびクロックに接続される 8 ビットの実装用です。もう 1 つは、ピンおよびクロックに接続される 16 ビットの実装用です。

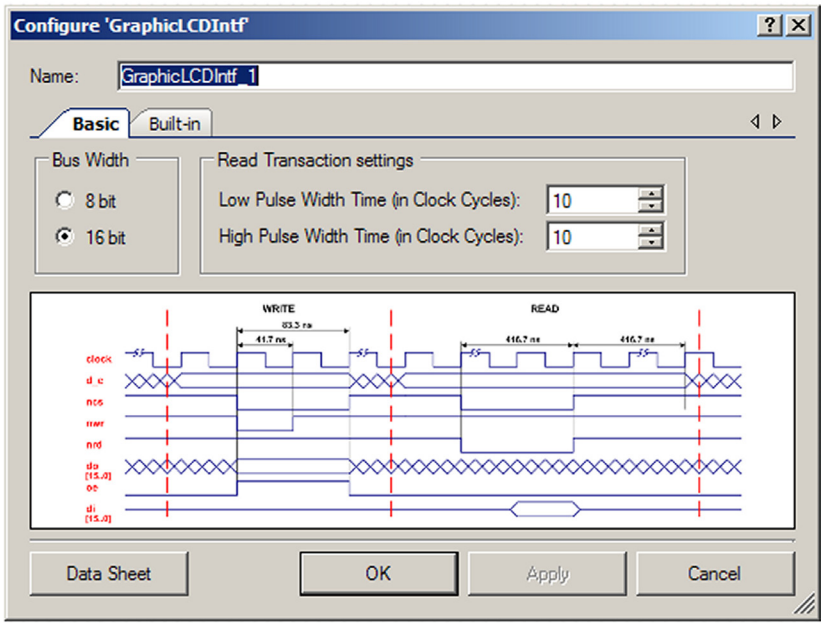


それぞれのマクロでは、クロックが 20 MHz に設定され、パルス幅設定はデフォルトのままになります。これらは、CY8CKIT-032 グラフィック LCD インターフェース キットで使用される SSD1289 コントローラでは、正しい設定です。

「同期入力」オプションは、すべてのデータ ピンで選択が解除され、すべてのピンの API 生成はオフになります。

パラメータおよびセットアップ

GraphicLCDIntf コンポーネントを設計上にドラッグし、クリックして [Configure] (設定) ダイアログを開きます。GraphicLCDIntf のデフォルト設定は、CY8CKIT-032 グラフィック LCD インターフェース キットで使用される Solomon Systech SSD1289 デバイスとの操作に適しています。



Bus Width (バス幅)

グラフィック LCD コントローラへの 8 または 16 ビットのパラレル インターフェースが提供されるかどうかを決定します。デフォルト設定は 16 です。

Low Pulse Width Time (「LOW」パルス幅時間)

コントローラで、読み取りパルス幅「LOW」に必要なクロック周期数を決定します。この値は、2ー255 クロック周期に設定できます (読み取り値は、パルス終了の 1 クロック前にサンプリングされる必要があるため、最小は 2 です)。デフォルト設定は 10 です。

High Pulse Width Time (「HIGH」パルス幅時間)

コントローラで、読み取りパルス幅「HIGH」に必要なクロック周期数を決定します。この値は、1ー255 クロック周期に設定できます。デフォルト設定は 10 です。

Clock Select (クロック選択)

このコンポーネントには、内部クロックはありません。クロック源を取り付ける必要があります。このコンポーネントは、コンポーネントに接続されている単一のクロックから操作されます。

配置

GraphicLCDIntf は、UDB アレイ全体に配置され、すべての配置情報は、*cyfitter.h* ファイルを通して API に提供されます。

リソース

リソース	リソースのタイプ			API メモリ (バイト数)		ピン (外部入出力ごと)
	データパス セル	PLD	ステータス セル	Flash	RAM	
8 ビット インターフェイス	1	4	2	109	1	12
16 ビット インターフェイス	2	4	3	120	1	20

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。その次のセクションでは、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「GraphicLCDIntf_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。インスタンス名は、識別子の構文ルールに従った固有の値に変更できます。インスタンス名は、コンポーネントで生成されるすべてのグローバル関数名、変数、定数記号の接頭辞になります。分かりやすいよう、次の表では、インスタンス名「GraphicLCDIntf」を使用しています。

関数	説明
void GraphicLCDIntf_Start(void)	GraphicLCDIntf インターフェースを開始します。
void GraphicLCDIntf_Stop(void)	GraphicLCDIntf インターフェースを無効にします。
void GraphicLCDIntf_Write8(uint8 d_c, uint8 data)	8 ビット パラレル インターフェースで、書き込みトランザクションを開始します。

関数	説明
void GraphicLCDIntf_Write16(uint8 d_c, uint16 data)	16 ビット パラレル インターフェースで、書き込みトランザクションを開始します。
uint8 GraphicLCDIntf_Read8(uint8 d_c)	8 ビット パラレル インターフェースで、読み取りトランザクションを開始します。
uint16 GraphicLCDIntf_Read16(uint8 d_c)	16 ビット パラレル インターフェースで、読み取りトランザクションを開始します。
void GraphicLCDIntf_Sleep(void)	設定を保存し、GraphicLCDIntf を無効にします。
void GraphicLCDIntf_Wakeup(void)	設定を復元し、GraphicLCDIntf を有効にします。
void GraphicLCDIntf_Init(void)	デフォルトの GraphicLCDIntf 設定を初期化または復元します。
void GraphicLCDIntf_Enable(void)	GraphicLCDIntf を有効にします。
void GraphicLCDIntf_SaveConfig(void)	GraphicLCDIntf の設定を保存します。
void GraphicLCDIntf_RestoreConfig(void)	GraphicLCDIntf の設定を復元します。

グローバル変数

変数	説明
GraphicLCDIntf_initVar	グラフィック LCD インターフェースが初期化されたかどうかを示します。変数は、0 に初期化され、最初に GraphicLCDIntf_Start() が呼び出されると 1 に設定されます。これで、GraphicLCDIntf_Start() ルーチンを最初に呼び出した後で、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は、GraphicLCDIntf_Start() ルーチンを呼び出す前に変数を 0 に設定する必要があります。または、グラフィック LCD インターフェースは、GraphicLCDIntf_Init() および GraphicLCDIntf_Enable() 関数を呼び出すことで再初期化することもできます。

void GraphicLCDIntf_Start(void)

説明　アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングを有効にします。操作するコンポーネントを設定します。

パラメータ　なし

戻り値　なし

副作用　なし

void GraphicLCDIntf_Stop(void)

説明　アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングを無効にします。

パラメータ　なし

戻り値　なし

副作用　なし

void GraphicLCDIntf_Write8(uint8 d_c, uint8 data)

説明　8 ビット パラレル インターフェースで、書き込みトランザクションを開始します。書き込みは、指定された書き込みであるため、この関数は、書き込みがインターフェースで実際に完了する前に戻り値を返します。コマンドのキューがいっぱいである場合は、この関数は、この書き込みリクエストのスペースが空くまで、戻り値を返しません。

パラメータ　d_c: データ (1) またはコマンド (0) を示します。d_c ピンに渡されます。

データ: do_lsb[7:0] ピンで送信されるデータ

戻り値　なし

副作用　なし

void GraphicLCDIntf_Write16(uint8 d_c, uint16 data)

説明　16 ビット パラレル インターフェースで、書き込みトランザクションを開始します。書き込みは、指定された書き込みであるため、この関数は、書き込みがインターフェースで実際に完了する前に戻り値を返します。コマンドのキューがいっぱいである場合は、この関数は、この書き込みリクエストのスペースが空くまで、戻り値を返しません。

パラメータ　d_c: データ (1) またはコマンド (0) を示します。d_c ピンに渡されます。

データ: do_msb[7:0] (最上位バイト) と do_lsb[7:0] (最下位バイト) ピンで送信されるデータ

戻り値　なし

副作用　なし

uint8 GraphicLCDIntf_Read8(uint8 d_c)

説明　8 ビット パラレル インターフェースで、読み取りトランザクションを開始します。読み取りは、現在指定されている書き込みがすべて完了した後で実行されます。この関数は、読み取りが完了し、読み取り値が返されるまで待機します。

パラメータ　d_c: データ (1) またはコマンド (0) を示します。d_c ピンに渡されます。

戻り値　di_lsb[7:0] ピンからの 8 ビットの読み取り値

副作用　なし

uint16 GraphicLCDIntf_Read16(uint8 d_c)

説明　16 ビット パラレル インターフェースで、読み取りトランザクションを開始します。読み取りは、現在指定されている書き込みがすべて完了した後で実行されます。この関数は、読み取りが完了し、読み取り値が返されるまで待機します。

パラメータ　d_c: データ (1) またはコマンド (0) を示します。d_c ピンに渡されます。

戻り値　do_msb[7:0] (最上位バイト) と do_lsb[7:0] (最下位バイト) ピンからの 16 ビットの読み取り値

副作用　なし

void GraphicLCDIntf_Sleep(void)

説明　これは、コンポーネントのスリープを準備するための、好ましいルーチンです。GraphicLCDIntf_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。次に、GraphicLCDIntf_Stop() 関数、さらに GraphicLCDIntf_SaveConfig() 関数を呼び出して、ハードウェアの設定を保存します。アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングを無効にします。
CyPmSleep() または CyPmHibernate() 関数を呼び出す前に、GraphicLCDIntf_Sleep() 関数を呼び出します。電源管理関数については、『PSoC Creator *システム リファレンス ガイド*』を参照してください。

パラメータ　なし

戻り値　なし

副作用　なし

void GraphicLCDIntf_Wakeup(void)

説明　これは、GraphicLCDIntf_Sleep() が呼び出された際に、コンポーネントにその状態を復元するための、好ましいルーチンです。GraphicLCDIntf_Wakeup() 関数は、GraphicLCDIntf_RestoreConfig() 関数を呼び出して、設定を復元します。GraphicLCDIntf_Sleep() 関数が呼び出される前にコンポーネントが有効になった場合は、GraphicLCDIntf_Wakeup() 関数も、コンポーネントを再度有効にします。アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングを有効にします。

パラメータ　なし

戻り値　なし

副作用　最初に GraphicLCDIntf_Sleep() または GraphicLCDIntf_SaveConfig() 関数を呼び出さずに GraphicLCDIntf_Wakeup() 関数を呼び出すと、予期せぬふるまいを引き起こす場合があります。

void GraphicLCDIntf_Init(void)

説明　カスタマイザの [Configure] (設定) ダイアログの設定に従って、コンポーネントを初期化または復元します。GraphicLCDIntf_Start() ルーチンが GraphicLCDIntf_Init() を呼び出し、これがコンポーネントの動作を開始する好ましい方法であるため、GraphicLCDIntf_Init() を呼び出す必要はありません。読み取り「LOW」および「HIGH」パルス幅を定義する静的なコンポーネント設定のみが、初期値に復元されます。

パラメータ　なし

戻り値　なし

副作用　これは、コンポーネントを再初期化しますが、FIFO のデータはクリアせず、コンポーネントのハードウェア状態マシンをリセットしません。現在のトランザクションは、バスで実行されます。

void GraphicLCDIntf_Enable(void)

説明　ハードウェアの使用を開始し、コンポーネントの動作を開始します。GraphicLCDIntf_Start() ルーチンが GraphicLCDIntf_Enable() を呼び出し、これがコンポーネントの動作を開始する好ましい方法であるため、GraphicLCDIntf_Enable() を呼び出す必要はありません。

パラメータ　なし

戻り値　なし

副作用　なし

void GraphicLCDIntf_SaveConfig(void)

説明　この関数は、コンポーネントの設定と維持されないレジスタを保存します。この関数は、[Configure] (設定) 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、GraphicLCDIntf_Sleep() 関数によって呼び出されます。読み取り「LOW」および「HIGH」パルス幅を定義するコンパイル時間コンポーネント設定が復元されます。

パラメータ　なし

戻り値　なし

副作用　なし

void GraphicLCDIntf_RestoreConfig(void)

説明　GraphicLCDIntf の維持されないレジスタ設定を復元します。GraphicLCDIntf_Wakeup によって API が呼び出され、コンポーネントの維持されないレジスタを復元します。

パラメータ　なし

戻り値　なし

副作用　この API が GraphicLCDIntf_SaveConfig の前に呼び出される場合は、読み取り「LOW」および「HIGH」パルス幅のコンポーネント設定が、カスタマイザで提供されている値に復元されます。

ファームウェア ソースコードの例

CY8CKIT-032 グラフィック LCD インターフェース キットに付属のサンプルを参照してください。コントローラの初期化以外に、このコンポーネントは、典型的に、Segger emWin グラフィック コンポーネントにより、独占して使用されます。

機能説明

バスのトランザクション

このインターフェースは、読み取りまたは書き込みトランザクションのいずれかを実行します。これらのトランザクションは、以下のパラメータを持ちます。

- 読み取りまたは書き込み
- Address (アドレス)。この場合は、d_c ピンで駆動される 1 ビットのアドレスです。
- Data (データ) (8 または 16 ビット)。書き込みでは「do」で送信、読み取りでは「di」で読み取られます。

実装は、CPU が FIFO (データで使用されるのと同じ FIFO) を使用して、コマンド バイトをコンポーネントに送信することを想定しています。このコマンド バイトが読み取りまたは書き込みを示し、d_c ビットを提供します。

アイドル条件

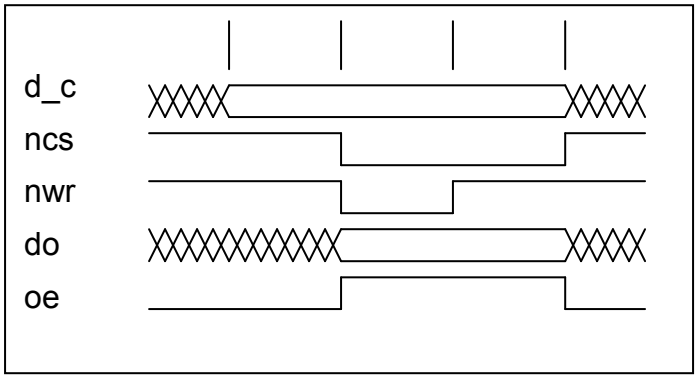
インターフェースで、読み取りも書き込みも実行されない場合は、インターフェースがアイドル状態になります。この条件での出力ピンの値は、以下の通りです。

- d_c: 無視 (最後の状態のまま)
- ncs: 1
- nwr: 1
- nrd: 1
- do: 無視 (最後の状態のまま)
- oe: 0

読み取りおよび書き込みトランザクションの説明で、リストされていない信号はアイドルです。

書き込みトランザクション

パラレル インターフェースでの書き込みトランザクションのタイミングは、以下の通りです。

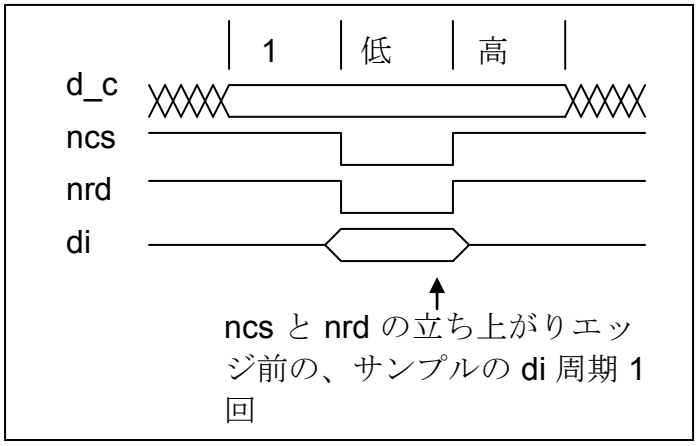


この図は、書き込みトランザクションが 3 つのクロック周期を必要としていることを示しています。タイミング図は、ビット幅に関わらず同じです。このトランザクションは、直ちに、または別の読み取りまたは書き込みトランザクションに続いて実行できます。または、書き込みトランザクションの前または後で、アイドル状態になる場合があります。

CPU へのインターフェースにより、CPU が書き込みリクエストを指定できるようになります (アドレスとデータを提供する書き込みをリクエストしてから、トランザクションがパラレル バスで実際に完了する前に実行する)。実装により、CPU を著しくストールさせることなく、CPU が 2 つの書き込みリクエストを持つことができます。

読み取りトランザクション

パラレル インターフェースでの読み取りトランザクションのタイミングは、以下の通りです。

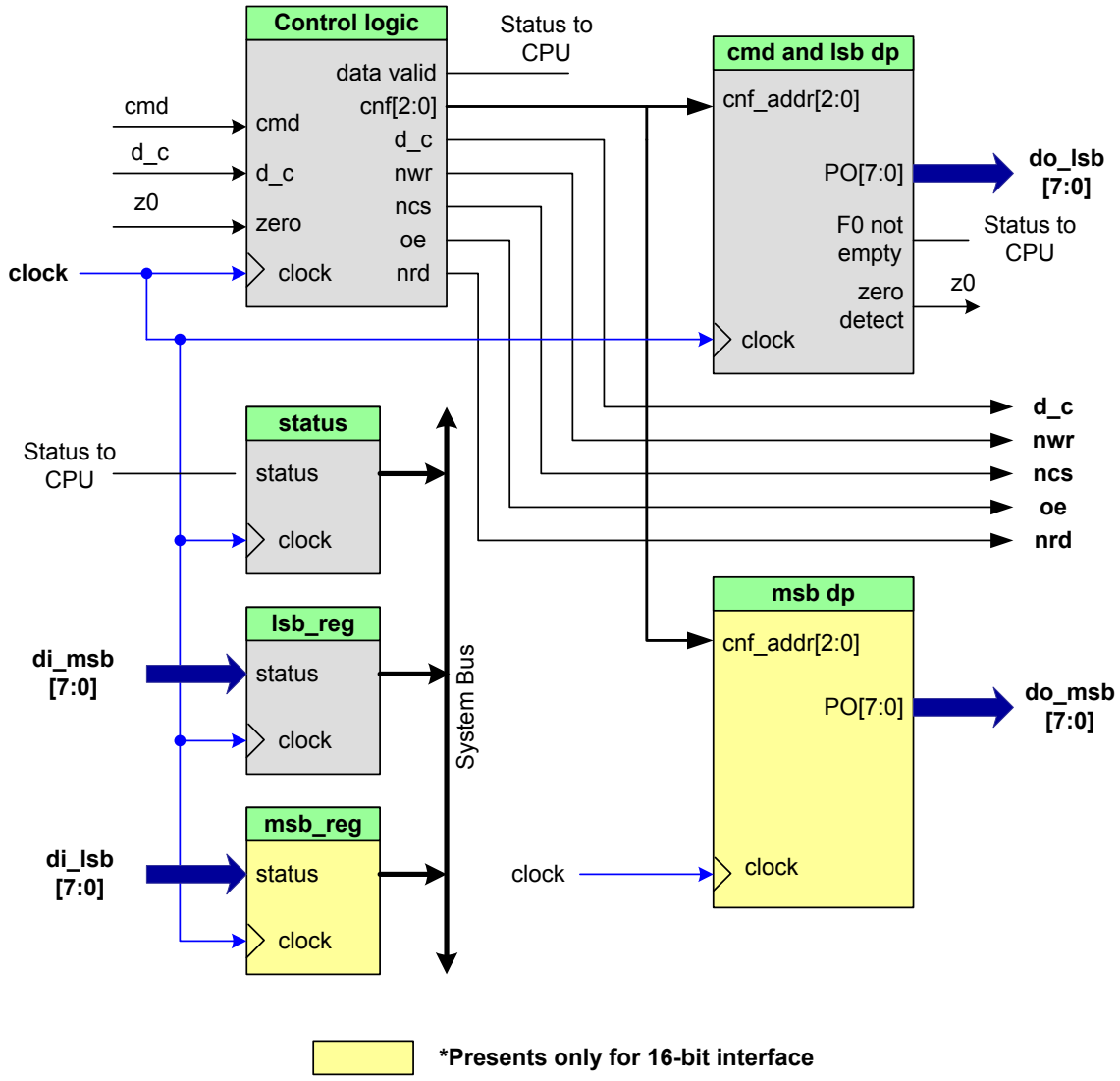


この図は、読み取りトランザクションが、読み取りパルス幅の「LOW」および「HIGH」設定により、さまざまなクロック周期数を必要とすることを示しています。タイミング図は、ビット幅に関わらず同じです。データ入力は、ncs と nrd 「LOW」パルスの終わりの前に、クロック周期 1 回をサンプリングすることに注意してください。このトランザクションは、直ちに、または別の読み取りまたは書き込みトランザクションに続いて実行できます。または、読み取りトランザクションの前または後で、アイドル状態になる場合があります。

読み取りおよび書き込みの順序は維持されます (読み取りは、指定された書き込みが完了する前に実行されます)。読み取りでは、CPU が実行前に、読み取りトランザクションの完了を待つ必要があります。

ブロック図と設定

GraphicLCDIntf コンポーネントは、設定されている UDB セットとして実装されます。以下のブロック図に実装を示します。



レジスタ

GraphicLCDIntf_STATUS_REG

ビット	7	6	5	4	3	2	1	0
値	予約済み						data_valid	F0_half_ empty

- F0_half_empty: 設定されると、コマンド/データ FIFO に少なくとも 2 バイトの容量が存在するようになります。
- data_valid: 読み取りデータが CPU で有効な場合に設定されます。このビットは、CPU がレジスタを読み取る際にクリアされます。

GraphicLCDIntf_DIN_LSB_DATA_REG

ビット	7	6	5	4	3	2	1	0
値	di_lsb[7:0]							

- 読み取りトランザクションの、入力データ バスの下位 8 ビット。

レジスタ値は、8 ビット インターフェースでは、GraphicLCDIntf_Read8() API 関数を使って、ユーザにより読み取ることができ、16 ビット インターフェースでは、GraphicLCDIntf_Read16() API 関数から返される値の最下位バイトです。

GraphicLCDIntf_DIN_MSB_DATA_REG

ビット	7	6	5	4	3	2	1	0
値	di_msb[7:0]							

- 読み取りトランザクションの、入力データ バスの上位 8 ビット。

レジスタ値は、16 ビット インターフェースでは、GraphicLCDIntf_Read16() API 関数から返される値の最上位バイトです。

注意事項：DIN_LSB_DATA_REG および DIN_MSB_DATA_REG ビットは、CPU ファームウェアがこれらのレジスタを読み取る際にクリアされます。

リファレンス

該当なし

DC 電気的特性と AC 電気的特性

以下の値は、期待されるパフォーマンスを示しており、初期特性データを基にしています。

「公称ルーティングでの最大」タイミング特性

パラメータ	説明		最小値	典型値	最大値 [1]	単位
f_{clock}	コンポーネントのクロック周波数		–	–	33	MHz
t_{AS}	アドレス セットアップ時間		1	–	–	t_{CY_clock} [2]
t_{PWLW}	パルス幅「LOW」時間		–	1	–	t_{CY_clock}
t_{PWHW}	パルス幅「HIGH」時間		3	–	–	t_{CY_clock}
t_{PWLR}	パルス幅「LOW」読み取り		2	–	255	t_{CY_clock}
t_{PWHR}	パルス幅「HIGH」読み取り		1	–	255	t_{CY_clock}
t_{AH}	アドレス ホールド時間					
		書き込み	2	–	–	t_{CY_clock}
		読み取り	t_{PWHR}	–	–	t_{CY_clock}
t_{CYCLE}	クロック周期時間					
		書き込み周期	4	–	–	t_{CY_clock}
		読み取り周期	t_{PWLR} + t_{PWRH} + 1	–	–	t_{CY_clock}
t_{DSW}	データ セットアップ時間		–	1	–	t_{CY_clock}
t_{DHW}	データ ホールド時間		–	1	–	t_{CY_clock}
t_{ACC}	データ アクセス時間		–	t_{PWHR} - 1	–	t_{CY_clock}
t_{DHR}	出力ホールド時間		–	0	–	t_{CY_clock}

[1] これらの「公称」値は、公称ルーティング状態における、コンポーネントの最大安全動作周波数を提供します。より高いクロック周波数でコンポーネントを実行することが可能です。この場合、STA の結果でタイミング要件を検証する必要があります。

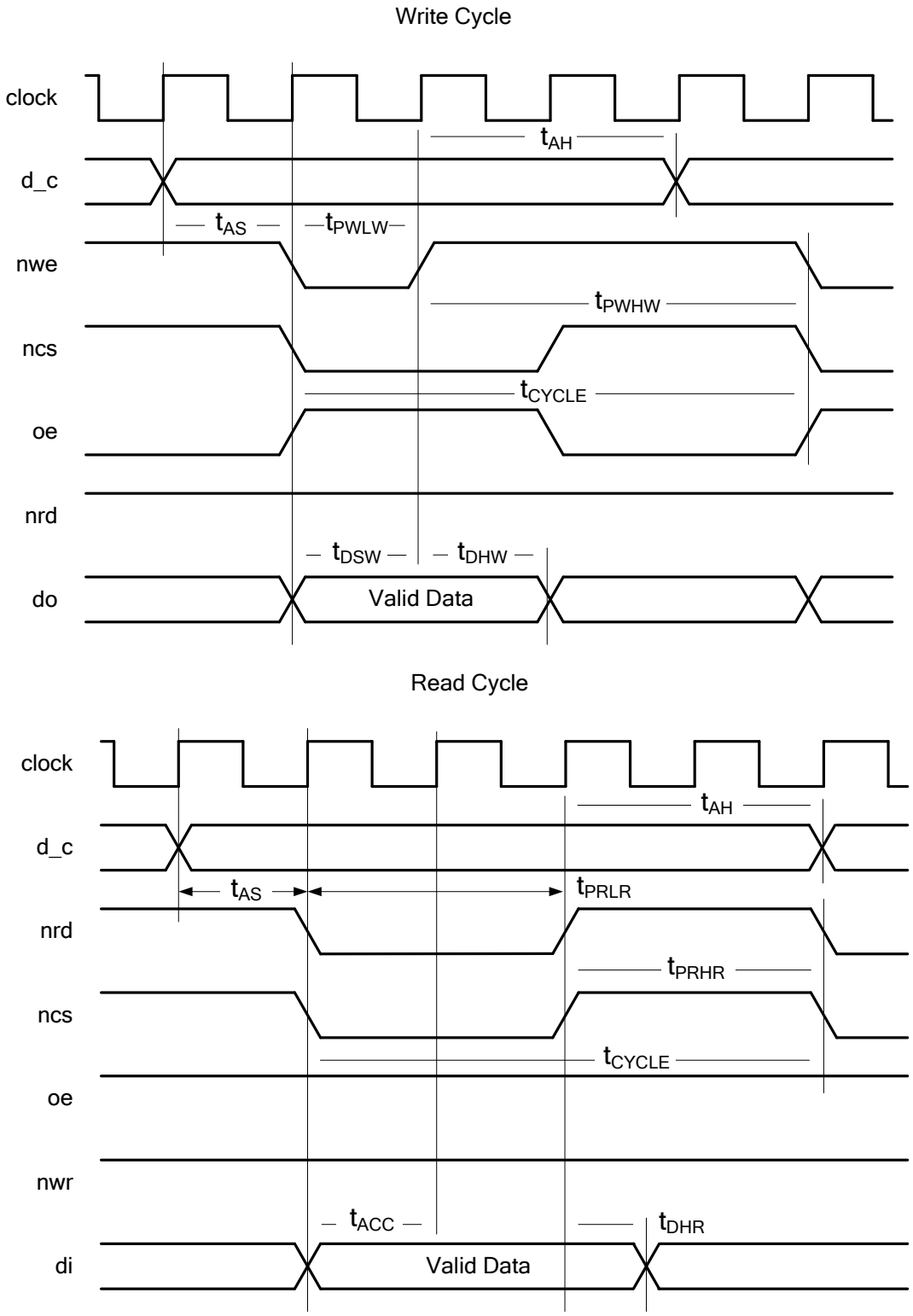
[2] t_{CY_clock} = 1 / f_{clock} - 1 回のクロック期間の周期時間

「すべてのルーティングでの最大」タイミング特性

パラメータ	説明		最小値	典型値	最大値[3]	単位
f_{CLOCK}	コンポーネントのクロック周波数		–	–	25	MHz
t_{AS}	アドレス セットアップ時間		1	–	–	t_{CY_clock}
t_{PWLW}	パルス幅「LOW」時間		–	1	–	t_{CY_clock}
t_{PWHW}	パルス幅「HIGH」時間		3	–	–	t_{CY_clock}
t_{AH}	アドレス ホールド時間					
		書き込み	2	–	–	t_{CY_clock}
		読み取り	t_{PWHR}	–	–	t_{CY_clock}
t_{CYCLE}	クロック周期時間					
		書き込み	4	–	–	t_{CY_clock}
		読み取り	t_{PWLR} + t_{PWRH} + 1	–	–	t_{CY_clock}
t_{DSW}	データ セットアップ時間		–	1	–	t_{CY_clock}
t_{DHW}	データ ホールド時間		–	1	–	t_{CY_clock}
t_{ACC}	データ アクセス時間		–	t_{PWHR} - 1	–	t_{CY_clock}
t_{DHR}	出力ホールド時間		–	0	–	t_{CY_clock}

[3]「すべてのルーティング」の最大値は、<公称値>/2 で最近似の整数に切り上げ/切り下げられます。この値により、このコンポーネント周波数以下で実行される場合に、ユーザがタイミング条件を気にする必要がなくなります。

図 1. データ トランザクションのタイミング図



特性データ用の STA 結果の使用方法

公称ルーティング最大値は、静的タイミング分析 (STA) を使って、複数のテスト パスから収集されます。最大値は、以下のメカニズムで、STA 結果を使って、それぞれの設計用に計算できます。

f_{CLOCK} **最大コンポーネント クロック周波数は、名前の付いたコンポーネント クロック (この場合は CLK) として、クロック要約でのタイミング結果で提供されます。_timing.html ファイルからの、コンポーネント クロックの制限例は、以下の通りです。**

-Clock Summary

Clock	Actual Freq	Max Freq	Violation
BUS_CLK	66.000 MHz	UNKNOWN	
CLK	33.000 MHz	43.579 MHz	

残りのパラメータは、実装に特有で、クロック周期で測定されます。これらは、2 つのカテゴリに分類できます。

コンポーネントの設定に使用するパラメータ:

t_{PWLW} 書き込み信号の最小パルス幅「LOW」時間。

t_{PWLR} 読み取り信号の最小パルス幅「LOW」時間。

t_{PWHR} 書き込み信号の最小パルス幅「HIGH」時間。

コンポーネントを設定する際に、これらのパラメータをどのように使用する必要があるかについての詳しい説明は、1 ページの「GraphicLCDIntf を使用する場合」に記載されています。

コンポーネント実装を基にして固定されるパラメータは、以下の通りです。

t_{PWHW} 書き込み信号の最小パルス幅「HIGH」時間。

t_{AS} nwe/nrd 信号の立ち下がりエッジ前に、アドレス信号が有効になる最小時間。

t_{AH} nwe/nrd 信号の立ち上がりエッジ後に、アドレス信号が有効になる最小時間。

t_{CYCLE} 信号トランザクション (書き込み/読み取り) がインターフェースで実行される期間。

t_{DSW} 書き込み信号の立ち上がりエッジ前に、データが有効になる最小時間。

t_{DHW} 書き込み信号の立ち上がりエッジ後に、データが有効になる最小時間。

t_{ACC} 読み取り信号の負のエッジ後に、データがサンプリングされる最小時間。

t_{DHR} nrd 信号の立ち上がりエッジ後に、データが有効になる最小時間。

コンポーネントの変更

ここでは、前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
1.60	FIFO ブロックの状態信号が DP クロックに再サンプルされます。	これで、すべての PSoC 3 および PSoC 5 シリコンの同じタイミング結果で、コンポーネントが機能します。
	データシートに特性データを追加	
	データシートのマイナーな編集と更新	

© Cypress Semiconductor Corporation, 2011. 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 (サイプレス セミコンダクタ社) 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。特許またはその他の権限下で、ライセンスを譲渡または暗示することもありません。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、または安全の用途のために使用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もありません。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。

PSoC® は、サイプレス セミコンダクタ社の登録商標であり、PSoC Creator™ およびプログラマブル System-on-Chip™ は、サイプレス セミコンダクタ社の商標です。本書で言及するその他すべての商標または登録商標は、各社の所有物です。

全てのソース コード (ソフトウェアおよび / またはファームウェア) はサイプレス セミコンダクタ社 (以下「サイプレス」) が所有し、全世界の特許権保護 (米国およびその他の国)、米国の著作権法なら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りライセンシー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シーの製品のみをサポートするカスタム ソフトウェアおよび / また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コピー、使用、変更そして作成するためのライセンス、ならびにサイプレスのソース コードおよ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、また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、明示的または黙示的を問わず、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、商品性または特定目的への適合性の黙示的な保証か • 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または回路を適用また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。