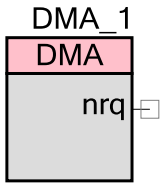
ダイレクト メモリアクセス (DMA)

1.50



特長

- 24 のチャネル
- 8 つの優先レベル
- 128 のトランザクション ディスクリプタ
- 8、16、32 ビットのデータ転送
- 設定可能な送信・受信アドレス
- エンディアン互換性対応
- データ転送完了時に割り込み生成が可能
- DMA ウィザードがアプリケーションの開発を補助

概要説明

DMA コンポーネントは、メモリ、コンポーネント、レジスタ間のデータ転送を可能にします。コントローラは、8、16、32 ビット幅のデータ転送をサポートし、エンディアンの異なる送信元と送信先の間でデータ転送を行うように構成できます。複雑な操作を行う目的で、TD をチェーンすることもできます。

DMA は、レベルトリガ、立ち上がりエッジトリガ信号に対応しています。詳細については、Hardware Request（ハードウェアのリクエスト）パラメータ セレクションを参照してください。

DMA コンポーネントをいつ使うか

データ転送の負荷を CPU から取り除きたい場合、または事前設定ができ予測可能な形でデータの転送が必要なとき、DMA コンポーネントは便利です。その例を下記に示します。

- メモリ→メモリ
- メモリ→ペリフェラル
- ペリフェラル→メモリ
- ペリフェラル→ペリフェラル

TD は個々にも実行でき、複雑な転送を行うにはチェーンして実行することもできます。

DMA Wizard (DMA ウィザード)

PSoC Creator は、DMA を使用したアプリケーションの開発が迅速かつ正確にできるよう支援する DMA ウィザードを提供します。ウィザードは、TD 定義から、またアプリケーションにコピー＆ペーストできる必要な C コードの生成までをガイドします。

ウィザードは、[PSoC Creator **Tools**] (PSoC Creator ツール) メニューから起動します。詳細については、PSoC Creator ヘルプを参照してください。

PSoC 3 のアドレス

PSoC 3 では、DMA 転送に関係するフラッシュ以外のすべてのロケーションが、メモリの最初の 64K 以内となります。フラッシュ以外のすべてのロケーションでは、上位 16 ビットのアドレスのために与えられた値は 0 でなければなりません。Keil 社製のコンパイラは最初の 64K 以外のアドレスは認識せず、上位 16 ビットを他の情報に使用するため、上位バイトがゼロ以外になります。このため、ポインタからその位置までの上位 16 ビットは直接使用することができません。詳細については、*「Generic Pointers in Keil (Keil 社の汎用ポインタ)」*を参照してください。フラッシュの場合は、アドレスの上位 16 ビットに使用する適切な値は次の通りです。

```
HI16(CYDEV_FLS_BASE)
```

これはコンパイラが行う特殊処理です。PSoC 3 と PSoC 5 の両方で適切に機能するコードを作成するには、以下のコードスタイルを使用します。"src" がフラッシュの変数、"dst" が SRAM の変数であると仮定すると

```
#if (defined(__C51__))
    /* PSoC 3 - Source is Flash */
    dmaChan = DMA_1_DmaInitialize(1, 0, HI16(CYDEV_FLS_BASE), 0);
#else
    /* PSoC 5 */
    dmaChan = DMA_1_DmaInitialize(1, 0, HI16(src), HI16(dst));
#endif
```

PSoC 5 SRAM アクセス

PSoC 5 では、DMA は 0x1FFF8000～0x1FFFFFFF の範囲の RAM にアクセスできません。しかし同じメモリで 0x20008000～0x2000FFFF はアクセス可能です。

CPU アクセス

```
0x1FFF8000 - 0x1FFFFFFF C-BUS 32KB
0x20000000 - 0x20007FFF S-BUS 32KB
```

DMA アクセス

```
0x20000000 - 0x20007FFF S-BUS 32KB
0x20008000 - 0x2000FFFF C-BUS 32KB
```

この再マッピングは、DMA を設定するために使用される API によって自動的にハンドルされます。API に渡されるパラメータは、DMA の API によって自動的にハンドルされるネイティブ CPU アドレスの上下 16 ビットです。注：DMA エンジンがアドレスをインクリメントする場合、増えるのは下位 16 ビットのみです。従って、0x2000FFFF の次のアドレスは 0x20000000 で、これは連続した 64K バイトのメモリブロックとして機能するメモリ空間となります。

入出力接続

このセクションでは、DMA の様々な入力および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、I/O が、その I/O の説明でリストされている条件において、シンボルに隠れている可能性があることを示します。

nrq – 出力

nrq 端子は割り込みに接続することができます。また、コンポーネントに接続して、DMA 転送の完了を知らせることもできます。DMA 転送が完了すると、DMA は 2 バスクロックの幅を持つパルスを NRQ に生成します。

drq – 入力 *

drq 端子は、DMA トランザクションを要求できるコンポーネントに接続されています。

drq 入力は、レベルセンシティブかエッジセンシティブのいずれかです。drq がレベルセンシティブの場合、drq がアサートされると、DMA リクエストは連続して起こります。drq がエッジセンシティブの場合、DMA リクエストは少なくとも 1 バスクロックサイクル幅です。

trq – 入力 *

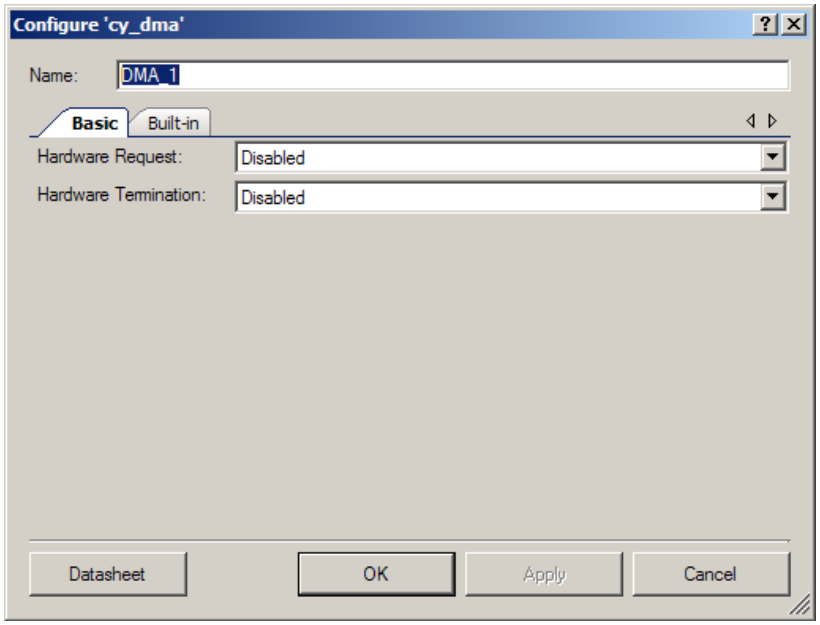
trq 端子は、DMA トランザクションを終了できるコンポーネントに接続します。利用できるデータがない場合、コンポーネントは DMA からデータを要求されることがあります。この信号を使用してトランザクションを終了します。

チェーンの現在の TD が終了すると、トランスファ カウントが完了したかのように終了します。従って、これは、チェーンに他の TD があるかないか、またどのタイプのトランザクション（例：ピンポン、円形、自動繰返しなど）が定義されているかに依存します。

この信号は、チャネルがデータ転送を試みているときにのみ使用されます。その他の場合は、このラインのポジティブエッジは無視されます。

コンポーネント パラメータ

DMA コンポーネントをデザイン上にドラッグし、ダブルクリックして**設定**ダイアログを開きます。



DMA は、次のパラメータを提供します。

Hardware Request(ハードウェアのリクエスト)

このパラメータは、DMA に対してトリガを実行する波形のタイプを設定します。「**Disabled(ディスエーブル)**」以外のオプションであれば、DMA リクエストの発行をハードウェアから許可する drq 端子を追加します。利用可能なオプションは以下の通りです。

- **Disabled(ディスエーブル) –** drq 端子は表示されません。この場合、DMA は CPU を介してのみトリガすることができます。
- **Derived(派生)** – drq のドライバを検査し、固定機能ブロック (I^2C, USB, CAN など)に接続していれば、何に接続しているかに基づいて DMA タイプが派生します。自動割り当ては、デバイスのデータシート記載の情報に基づいて行われます。固定機能ブロックに接続されていない場合は、**立ち上がりエッジ** オプションを使用します。
- **Rising Edge(立ち上がりエッジ) –** ソース信号の立ち上がりエッジで DMA をトリガします。DMA がイベントに基づいて発生する必要があるときは、このオプションを選択します。例えば、DMA を定期的に発生させるには立ち上がりエッジモードを設定し、「drq」信号は適切なレートに設定されたクロック信号に接続します。
- **Level –** DMA に接続されているソースをレベルセンシティブ リクエストとして選択します。この選択肢は、ある特定の状態がアクティブである限り DMA のトリガを続けなければならない場合に選択します。この典型的な

例としては、FIFO の FILL レベルに基づいて DMA がトリガされるようなケースになります。I2S などの通信コンポーネントと共に使用する場合の構成です。

ハードウェアの終了

このパラメータは**イネーブル** または **ディスエーブル**に設定できます。**イネーブル**に設定すると、DMA リクエストのハードウェアによる終了を可能にする trq 端子が追加されます。この端子がディスエーブルに設定されている場合、DMA 転送は、CPU リクエストによって、または DMA がデータ転送を完了したことによってのみ終了します。

リソース

DRQ	デジタル ブロック					API メモリ(バイト)		ピン (外部入出力ごと)
	データパス	マクロセル	ステータスレジスタ	コントロールレジスタ	Counter7	フラッシュ	RAM	
1	該当なし	該当なし	該当なし	該当なし	該当なし	2332	7	0

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構成できます。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。

デフォルトでは、PSoC Creator はインスタンス名「DMA_1」を設計上のコンポーネントの最初の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、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。読みやすいように、下表では「DMA」というインスタンス名を使用しています。

DMA インスタンスに対応する API

機能	説明
DMA_DmaInitialize	呼び出し側が使用する DMA チャネルを割り当て・初期化します。
DMA_DmaRelease	コンポーネントのインスタンスと関連付けられている DMA チャネルを開放し、ディスエーブルにします。

uint8 DMA_DmaInitialize(uint8 burstCount、uint8 requestPerBurst、uint16 upperSrcAddress、uint16 upperDestAddress)

説明: 呼び出し側が使用する DMA チャネルを割り当て・初期化します。

パラメータ: uint8 burstCount. この TD が分割された結果のバーストサイズ（1～127）を指定します。バーストサイズは、スポークサイズの倍数でなければなりません。

この値がゼロのとき、転送は 1 バーストで完了します。この場合、TD のトランスファ カウント パラメータが 1 バーストで転送されるバイト数を決定します。

uint8 requestPerBurst. トランザクションを完了するために必要な場合は、データ全体を複数のバーストに分割することができます。

値	動作
0	最初のバーストに続くすべてのバーストは、自動的にリクエスト・実行されます。
1	最初のバーストに続くすべてのバーストは、個別に要求されなければなりません。

uint16 upperSrcAddress。送信元アドレスの上位 16 ビット。

uint16 upperDestAddress。受信先アドレスの上位 16 ビット。

戻り値: uint8: 呼び出し側が DMA アクティビティのために使用できるチャネル。チャネルが残っていない場合は、DMA_INVALID_CHANNEL (0xFF) 。

副作用: なし

void DMA_DmaRelease(void)

説明: コンポーネントのインスタンスと関連付けられているチャネルを開放します。DMA_DmaInitialize が再び呼び出されない限り、チャネルを再び使用することはできません。

パラメータ: なし

戻り値: なし

副作用: なし

DMA ライブラリの API（すべての DMA インスタンスが共有）

DMA コントローラの関数

機能	説明
CyDmacConfigure()	DMAC 構成レジスタにデフォルト値をセットします。
CyDmacError()	DMAC からエラービットを取得します。
CyDmacClearError()	DMAC のエラーレジスタでエラービットをクリアします。

CyDmacErrorAddress()	最後の DMAC エラーが発生したアドレスを取得します。

void CyDmacConfigure(void)

説明: 割り当てされるすべての TD が記載されているリンクリストを作成します。この関数はスタートアップコードで呼び出されます。通常は呼び出す必要はありません。すべての DMA チャネルが非アクティブの場合、ユーザが呼び出すことができます。

パラメータ: なし

戻り値: なし

副作用: なし

uint8 CyDmacError(void)

説明: 最後に失敗した DMA トランザクションのエラータイプを含んだ DMA_ERROR タイプの値を返します。

パラメータ: なし

戻り値: DMA_ERROR タイプからエラーデータ(4 ビット)を返します。

ビット	定義	説明
ビット 3	DMAC_PERIPH_ERR	ペリフェラルがバス トランザクションに対してエラー反応を示したとき、1 にセットされます。1 を書き込むことによってクリアされます。
ビット 2	DMAC_UNPOP_ACC	無効なアドレスへのアクセスが試みられたとき、1 にセットされます。1 を書き込むことによってクリアされます。
ビット 1	DMAC_BUS_TIMEOUT	バスのタイムアウトが発生したとき、1 にセットされます。1 を書き込むことによってクリアされます。タイムアウトの値は PHUBCFG レジスタの BUS_TIMEOUT フィールドが決定します。

副作用: なし

void CyDmacClearError(uint8 error)

説明: DMAC のエラーレジスタでエラービットをクリアします。

パラメータ: uint8 エラー。DMA_ERROR タイプでクリアするエラービットのビットマスク。

ビット	定義	説明
ビット 3	DMAC_PERIPH_ERR	ペリフェラルがバス トランザクションに対してエラー反応を示したとき、1 にセットされます。1 を書き込むことによってクリアされます。
ビット 2	DMAC_UNPOP_ACC	無効なアドレスへのアクセスが試みられたとき、1 にセットされます。1 を書き込むことによってクリアされます。
ビット 1	DMAC_BUS_TIMEOUT	バスのタイムアウトが発生したとき、1 にセットされます。1 を書き込むことによってクリアされます。タイムアウトの値は PHUBCFG レジスタの BUS_TIMEOUT フィールドが決定します。

戻り値: なし

副作用: なし

uint32 CyDmacErrorAddress(void)

説明: BUS_TIMEOUT、UNPOP_ACC、PERIPH_ERR の発生時、エラーのアドレスがエラーアドレス レジスタに書き込まれ、この関数で読み取ることができます。エラーが複数の場合は、最初のエラーのアドレスだけが保存されます。

パラメータ: なし

戻り値: エラーを発生させたアドレス。

副作用: なし

チャネル特有の関数

機能	説明
CyDmaChAlloc()	呼び出し側が使用する DMA のチャネルを割り当てます。
CyDmaChFree()	CyDmaChAlloc() が割り当てたチャネルを開放します。
CyDmaChEnable()	DMA チャネルを実行するためイネーブルにします。
CyDmaChDisable()	DMA チャネルをディスエーブルにします。
CyDmaClearPendingDrq()	ペンディングの DMA データリクエストをクリアします。
CyDmaChPriority()	DMA チャネルの優先順位を設定します。
CyDmaChSetExtendedAddress()	送信元と受信先のアドレスの上位 16 ビットを設定します。

機能	説明
CyDmaChSetInitialTd()	チャネルの初期 TD を設定します。
CyDmaChSetRequest()	TD のチェーンまたは 1 つの TD を終了する、または DMA を開始するリクエスト。
CyDmaChGetRequest()	CyDmaChSetRequest リクエストが実行されたかどうかをチェックします。
CyDmaChStatus()	現在の TD のステータスを特定します。
CyDmaChSetConfiguration()	チャネル用の構成情報を設定します。

uint8 CyDmaChAlloc(void)

説明: チャネルハンドル必要とするすべての関数で使用するために、DMAC からチャネルを割り当てます。

パラメータ: なし

戻り値: 割り当てされたチャネル番号。ゼロは有効なチャネル番号です。利用可能なチャネルがない場合、DMA_INVALID_CHANNEL が返されます。

副作用: なし

cystatus CyDmaChFree(uint8 chHandle)

説明: CyDmaChAlloc が割り当てたチャネルハンドルを開放します。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChEnable(uint8 chHandle, uint8 preserveTds)

説明: DMA チャネルをイネーブルにします。ソフトウェアまたはハードウェアのリクエストは、チャネルの実前に発生させる必要があります。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

uint8 preserveTds. TD が完了したときに、オリジナルの TD の状態を保存します。このパラメータはチャネル内のすべての TD に適用されます。

値	動作
0	TD が完了すると、DMAC は TD 構成値を現在の状態に残し、オリジナルの状態に復元しません。
1	TD が完了すると、DMAC はオリジナルの構成値を TD に復元します。

preserveTds が設定されているときは、ワーキング レジスタが存在しており、チャネル番号と同じ TD スロットが RESERVED となります。例えば、CH06 を使用中で **preserveTds** が設定されている場合、TD スロット 6 の使用は許可されません。このスロットは DMA エンジンの内部使用のために取り戻されます。

注 チャネルの **preserveTds** が 0 に設定されているとき、完了した TD にチェーンバックしないでください。そのチャネルの **preserveTds** を 0 に設定し、TD が完了した場合、トランスファ カウントは 0 になります。トランスファ カウントが 0 の TD が開始されると、TD は無制限の数のデータを転送します。

preserveTds が 0 に設定されている場合、間違ったデータを要求する可能性があるため、ハードウェアリクエスト(DRQ)を使用する際は特に注意してください。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChDisable(uint8 chHandle)

説明: DMA チャネルをディスエーブルにします。この関数が呼び出されるときは、いつチャネルをディスエーブルにするか、またどの TD が実行されているか特定するために CyDmaChStatus が、呼び出されるでしょう。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合はCYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaClearPendingDrq(uint8 chHandle)

説明: ペンディングの DMA データリクエストをクリアします。

パラメータ: chHandle: DMA チャネルのハンドル。

戻り値: 成功の場合はCYRET_SUCCESS、それ以外はCYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChPriority(uint8 chHandle, uint8 priority)

説明: DMA チャネルの優先順位を設定します。この関数は、ユーザが実行中の優先順位を変更したいときに使用することができます。優先順位が DMA チャネルで同じままでも、ユーザは .cydwr ファイル内で優先順位を設定できます。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

uint8 priority: チャネルに設定する優先順位は 0～7 です。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChSetExtendedAddress(uint8 chHandle, uint16 source, uint16 destination)

説明: DMA チャネル用の送信元と受信先アドレスの上位 16 ビットを設定します（チェーン内のすべての TD について有効）。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

uint16 source. DMA 転送元の上位 16 ビットのアドレス。

uint16 destination. DMA 転送先の上位 16 ビットのアドレス。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChSetInitialTd(uint8 chHandle, uint8 startTd)

説明: 関数 CyDmaChEnable が呼び出されるとき、実行される初期 TD を設定します。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitialize で返された前回のハンドル。

uint8 startTd. チャネルと関連付けられている最初の TD を設定するための TD のインデックス。ゼロは有効な TD インデックスです。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChSetRequest(uint8 chHandle, uint8 request)

説明: 呼び出し元に、TD チェーンの終了、または 1 つの TD の終了、または DMA チャネルを開始するダイレクトリクエストを許可します。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

uint8 request. 以下の定数のうちの 1 つ。各定数は 3 ビットの値です。

リクエスト値	説明
CPU_REQ	DMA チャネルを開始するダイレクトリクエストを作成します。
CPU_TERM_TD	1 つの TD を終了します。
CPU_TERM_CHAIN	TD チェーンを終了します。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChGetRequest(uint8 chHandle)

説明: この関数により、CyDmaChSetRequest の呼び出し元が、リクエストが完了したかどうかを判定します。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

戻り値: 以前にポスティングされたリクエストの状態を示す 3 ビットのリクエストに対応する 3 ビットのフィールドを返します。リクエストが完了すると、値はゼロになります。

ハンドルが無効の場合は DMA_INVALID_CHANNEL。

副作用: なし

cystatus CyDmaChStatus(uint8 chHandle, uint8 * currentTd, uint8 * state)

説明: DMA チャネルのステータスを特定します。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitalize で返された前回のハンドル。

uint8 * currentTd. 現在の TD のインデックスを保存するアドレス。値が不要の場合には、NULL でも可。

uint8 state: チャネルのステートを保存するアドレス。値が不要の場合には、NULL でも可。

ビット 1	STATUS_TD_ACTIVE	0: チャネルは現在 DMAC によるサービスを実施していません。
		1: チャネルは現在 DMAC によるサービスを実施しています。
ビット 0	STATUS_CHAIN_ACTIVE	0: TD チェーンは非アクティブです。新しいチェーンをトリガする DMA リクエストがないか、前回のチェーンが完了した状態です。
		1: TD チェーンが DMA リクエストによってトリガされました。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaChSetConfiguration(uint8 chHandle, uint8 burstCount, uint8 requestPerBurst, uint8 tdDone0, uint8 tdDone1, uint8 tdStop)

説明: チャネル用の構成情報を設定します。

パラメータ: uint8 chHandle. CyDmaChAlloc または DMA_DmaInitialize で返された前回のハンドル。

uint8 burstCount. データ転送のバーストサイズ（1～127）を指定します。この値がゼロのとき、転送は 1 バーストで完了します。

(uint8) requestPerBurst. トランザクションを完了するために必要な場合は、データ全体を複数のバーストに分割することができます。

値	動作
0	最初のバーストに続くすべてのバーストは、自動的にリクエスト・実行されます。
1	最初のバーストに続くすべてのバーストは、個別に要求されなければなりません。

uint8 tdDone0. TERMOUT0 割り込みラインの 1 つを信号完了に選択します。nrq 端子に接続されているラインは、TERMOUT0_SEL 定義を特定し、*cyfitter.h* によって与えられたとおりに使用しなければなりません。

uint8 tdDone1: TERMOUT1 割り込みラインの1つを信号完了に選択します。nrq 端子に接続されているラインは、TERMOUT1_SEL 定義を特定し、*cyfitter.h* によって与えられたとおりに使用しなければなりません。

uint8 tdStop. TD の終了の信号を DMAC に通知するための TERMIN 割り込みラインの 1 つを選択します。Trq 端子に接続されている信号は、どの TERMIN （終了リクエスト）を使用するかを特定します。

戻り値: CYRET_SUCCESS : 成功した場合

chHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

トランザクション・デスクリプタ・ファンクション

機能	説明
CyDmaTdAllocate()	使用できるフリーリストから TD を割り当てます。
CyDmaTdFree()	フリーリストへ TD を戻します。
CyDmaTdFreeCount()	利用可能な空の TD 数を取得します。
CyDmaTdSetConfiguration()	TD に構成を設定します。
CyDmaTdGetConfiguration()	TD に対して構成を取得します。
CyDmaTdSetAddress()	送信元と受信先のアドレスの下位 16 ビットを設定します。
CyDmaTdGetAddress()	送信元と受信先のアドレスの下位16ビットを取得します。

uint8 CyDmaTdAllocate(void)

説明: 割り当てられた DMA チャネルと共に使用する TD を割り当てます。

パラメータ: なし

戻り値: 呼び出し元が使用するゼロベースの TD インデックス。TD は 128 から予約されている TD（0～23）を引くため、返される戻り値は 24～128 ではなく、24～127 となります。

使用できる空のTDがない場合は DMA_INVALID_TD。

副作用: なし

void CyDmaTdFree(uint8 tdHandle)

説明: フリーリストへ TD を返します。

パラメータ: uint8 tdHandle. CyDmaTdAllocate API によって返される TD ハンドル。

戻り値: なし

副作用: なし

uint8 CyDmaTdFreeCount(void)

説明: 割り当てできる使用可能な TD 数を返します。

パラメータ: なし

戻り値: 空の TD 数。

副作用: なし

cystatus CyDmaTdSetConfiguration(uint8 tdHandle, uint16 transferCount, uint8 nextTd, uint8 configuration)

説明: TDを設定します。

パラメータ: uint8 tdHandle. 以前に CyDmaTdAlloc から返されたハンドル。

uint16 transferCount. この TD 用のデータ転送サイズ（単位：バイト）。サイズがゼロの場合、転送は無制限に続きます。このパラメータは 4095 バイトが限度です。それよりも高い値が通った場合、TD は全く初期化されません。

uint8 nextTd. TD チェーンの次の転送ディスクリプタのゼロベースインデックス。ゼロは次の TD の有効なポインタです; DMA_END_CHAIN_TD はチェーンの最後です。

uint8 configuration. 構成ビットのビットフィールドを保存します。

構成オプション	説明
TD_SWAP_EN	エンディアンスワップを実行します。
TD_SWAP_SIZE4	スワップサイズ = 4 バイト
TD_AUTO_EXEC_NEXT	現在の TD が完了すると、自動的にチェーンの次の TD がトリガされます。
TD_TERMIN_EN	「trq」入力ラインでポジティブエッジが発生すると、この TD を終了します。ポジティブエッジはバースト時に発生する必要があります。その場合だけ、DMAC が検知します。
DMA__TD_TERMOUT_EN	このTDが完了すると、TERMOUT 信号がパルスを生成します。注：このオプションはインスタンスに特有で、インスタンス名のつぎに下線が2文字分続きます。この例では、インスタンス名は DMA です。
TD_INC_DST_ADR	バーストの各データトランザクションのサイズに応じて、DST_ADR がインクリメントします。
TD_INC_SRC_ADR	バーストの各データ トランザクションのサイズに応じて、SRC_ADR がインクリメントします。

戻り値: CYRET_SUCCESS : 成功した場合

tdHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaTdGetConfiguration(uint8 tdHandle, uint16 * transferCount, uint8 * nextTd, uint8 * configuration)

説明: TD の構成を再取得します。NULL ポインタがパラメータとして渡された場合、このパラメータはスキップされます。ユーザは興味のある値のみリクエストできます。

パラメータ: uint8 tdHandle. 以前に CyDmaTdAlloc から返されたハンドル。

uint16 transferCount. この TD 用のデータ転送サイズ（単位：バイト）を保存するアドレス。サイズがゼロの場合、TD が転送を完了した、または TD が無制限の転送を行っている可能性があります。

uint8 nextTd. TD チェーン中の次の TD のインデックスを保存するアドレス。

uint8 * configuration. 構成ビットのビットフィールドを保存するアドレス。CyDmaTdSetConfiguration を参照してください。

戻り値: CYRET_SUCCESS : 成功した場合

tdHandle が無効な場合は CYRET_BAD_PARAM。

副作用: TD が N 個のトランスファー を持ち、これを実行した場合、トランスファー カウントは 0 になります。これが再実行され、転送数がゼロの場合、無制限転送と解釈されます。トランスファー カウントがゼロの TD を要求する場合には注意が必要です。

cystatus CyDmaTdSetAddress(uint8 tdHandle, uint16 source, uint16 destination)

説明: この TD だけを対象として、送信元と受信先のアドレスの下位 16 ビットを設定します。

パラメータ: uint8 tdHandle. 以前に CyDmaTdAlloc から返されたハンドル。

uint16 source. データ転送元の下位 16 ビットのアドレス。

uint16 destination. データ転送先の下位 16 ビットのアドレス。

戻り値: CYRET_SUCCESS : 成功した場合

tdHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

cystatus CyDmaTdGetAddress(uint8 tdHandle, uint16 * source, uint16 * destination)

説明: この TD だけを対象として、送信元と受信先のアドレスの下位 16 ビットを取得します。NULL がポインタパラメーターに接続している場合、値はスキップします。ユーザは興味のある値のみリクエストできます。

パラメータ: uint8 tdHandle. 以前に CyDmaTdAlloc から返されたハンドル。

uint16 * source. データ転送元の下位 16 ビットのアドレスを保存するアドレス。

uint16 destination. データ転送先の下位 16 ビットのアドレスを保存するアドレス。

戻り値: CYRET_SUCCESS : 成功した場合

tdHandle が無効な場合は CYRET_BAD_PARAM。

副作用: なし

ファームウェア ソースコードの例

PSoC Creator は、「Find Example Project」 (プロジェクト例を検索) ダイアログに数多くのプロジェクト例を提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、「Component Catalog (コンポーネント カタログ) 」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 (スタート ページ) 」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 (フィルタのオプション) 」**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 」(プロジェクト例を検索) を参照してください。

DMA アドレスの移植性

DMA_DmaInitialize 関数または CyDmaChSetExtendedAddress 関数に渡された上位アドレスの意味は、PSoC 3 と PSoC 5 デバイスでは異なります。DMA アドレスは、PSoC 3 と PSoC 5 デバイス間での移植性はありません。上位アドレスは、物理的送信元と受信先アドレスの上位 16 ビットを設定するために必要です。

次のセクションでは、コードの例を提供します。

PSoC 3 デバイス

PSoC 3 デバイスでは、上位アドレス値はポインタの上位バイトと対応しないため、マニュアルで設定する必要があります。Keil 社製のコンパイラ(PSoC 3 と共に使用)は、物理アドレスではなく、ポインタの上位バイトを使用して、メモリタイプを示します。

フラッシュから読み取るために、PSoC 3 デバイスの送信元の上位アドレスは、次の定義で設定しなければなりません。`CYDEV_FLS_BASE`.

Ram から Ram へ

```
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];

/* DMA Configuration for DMA_1 */
#define DMA_1_BYTES_PER_BURST 16
#define DMA_1_REQUEST_PER_BURST 1
#define DMA_1_SRC_BASE (CYDEV_SRAM_BASE)
#define DMA_1_DST_BASE (CYDEV_SRAM_BASE)
DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST,
    HI16(DMA_1_SRC_BASE), HI16(DMA_1_DST_BASE));
DMA_1_TD[0] = CyDmaTdAllocate();
CyDmaTdSetConfiguration(DMA_1_TD[0], 128, DMA_INVALID_TD, DMA_1__TD_TERMOUT_EN |
TD_INC_SRC_ADR | TD_INC_DST_ADR);
CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)memory1), LO16((uint32)memory2));
CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
CyDmaChEnable(DMA_1_Chan, 1);
```

フラッシュから Ram へ

```
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];

/* DMA Configuration for DMA_1 */
#define DMA_1_BYTES_PER_BURST 1
#define DMA_1_REQUEST_PER_BURST 1
#define DMA_1_SRC_BASE (CYDEV_FLS_BASE)
#define DMA_1_DST_BASE (CYDEV_SRAM_BASE)

DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST,
    HI16(DMA_1_SRC_BASE), HI16(DMA_1_DST_BASE));

DMA_1_TD[0] = CyDmaTdAllocate();
CyDmaTdSetConfiguration(DMA_1_TD[0], 128, DMA_INVALID_TD, DMA_1__TD_TERMOUT_EN |
TD_INC_SRC_ADR | TD_INC_DST_ADR);
CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)buf1), LO16((uint32)buf2));
CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
CyDmaChEnable(DMA_1_Chan, 1);
```

フラッシュから DAC へ

```
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];

/* DMA Configuration for DMA_1 */
#define DMA_1_BYTES_PER_BURST 1
#define DMA_1_REQUEST_PER_BURST 1
#define DMA_1_SRC_BASE (CYDEV_FLS_BASE)
#define DMA_1_DST_BASE (CYDEV_PERIPH_BASE)
DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST,
    HI16(DMA_1_SRC_BASE), HI16(DMA_1_DST_BASE));
DMA_1_TD[0] = CyDmaTdAllocate();
```

```
CyDmaTdSetConfiguration(DMA_1_TD[0], 64, DMA_INVALID_TD, TD_INC_SRC_ADR);
CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)fFashMem),
LO16((uint32)VDAC8_1_Data_PTR));
CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
CyDmaChEnable(DMA_1_Chan, 1);
```

PSoC 5 デバイス

PSoC 5 デバイスでは、上位アドレスは、送信元と受信先のアドレスの上位 16 ビットに設定することができます。

Ram から Ram へ

```
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];

/* DMA Configuration for DMA_1 */
#define DMA_1_BYTES_PER_BURST 1
#define DMA_1_REQUEST_PER_BURST 1
#define DMA_1_SRC_BASE (buffer1)
#define DMA_1_DST_BASE (buffer2)
DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST,
    HI16(DMA_1_SRC_BASE), HI16(DMA_1_DST_BASE));
DMA_1_TD[0] = CyDmaTdAllocate();
CyDmaTdSetConfiguration(DMA_1_TD[0], 128, DMA_INVALID_TD, DMA_1__TD_TERMOUT_EN |
TD_INC_SRC_ADR | TD_INC_DST_ADR);
CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)buffer1), LO16((uint32)buffer2));
CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
CyDmaChEnable(DMA_1_Chan, 1);
```

フラッシュから DAC へ:

```
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];

/* DMA Configuration for DMA_1 */
#define DMA_1_BYTES_PER_BURST 1
#define DMA_1_REQUEST_PER_BURST 1
#define DMA_1_SRC_BASE (FlashMem)
#define DMA_1_DST_BASE (CYDEV_PERIPH_BASE)
DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST,
    HI16(DMA_1_SRC_BASE), HI16(DMA_1_DST_BASE));
DMA_1_TD[0] = CyDmaTdAllocate();
CyDmaTdSetConfiguration(DMA_1_TD[0], 64, DMA_INVALID_TD, DMA_1__TD_TERMOUT_EN |
TD_INC_SRC_ADR);
CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)FlashMem),
LO16((uint32)VDAC8_1_Data_PTR));
CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
CyDmaChEnable(DMA_1_Chan, 1);
```

境界の条件

単一の DMA チャネルは 64K の境界を越えることができません。従って、DMA が 64K の境界を越えた場合、動作は警告なしに失敗します。これは PSoC 5 にとってのみ重要です。その理由は、PSoC 3 デバイスのメモリ空間はフラッシュの 64K や Ram の 8K を超えることがないからです。

このデータ構成を使用する場合、64 K の境界を越えないように注意してください。それには次のキーワードを利用することができます。

```
__attribute__((section()))
__attribute__((aligned()))
```

これらのキーワードはいずれも、GCC の「Specifying Attributes of Variables（変数の属性指定）」セクションのヘルプで説明されています。変数を特定のセクションと位置に表示したい場合は、section のキーワードを使用します。コンパイラを使って変数を特定の境界に置くには、aligned のキーワードを使用します。

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
1.50.b	データシートのマイナーな編集と更新	
1.50.a	CyDmaClearPendingDrq API を追加	
	データシートのマイナーな編集と更新	
1.50	DRQ タイプを特定する機能を追加。	旧機能は DRQ タイプを正しく判定できない場合もあった（新バージョンで「デリバー」された）ので、これをマニュアルで特定する機能を追加しました。 このため、構成ダイアログが変更され、デフォルトエディタを使用できなくなりました。
	パラメータ num_tds を削除。	使用される TD 数は書かれたコードによって異なります。この数字をコメントに補正する前に指定できるようにします。しかし実際に使用される数字を反映しているという保証はありません。この数字は DWR Editor（DWR エディタ）に表示されなくなりました。
	シンボル サマリを追加。	コンポーネントの概要を説明します。
	コンポーネントの形を整形。	会社のスタイルガイドに適合するよう更新しました。
	関数 CyDmaTdSetConfiguration() を更新。	NRQ 信号が簡略図でルーティングされている・されていない場合に、API は自動的にタームアウト信号をハンドルします。
	#define DMA__TD_TERMOUT_EN をヘッダファイルに追加。	この値は、コンポーネントが使用するタームアウトをイネーブルにする他の TD フラグと共に使用できます。

バージョン	変更の説明	変更の理由 / 影響
	uint8 DMA_DmaInitialize() からアサートを削除。	すでに行われた他のコードの変更のため、アサートが不要となりました。