AN90369

MIPI® CSI-2 イメージ センサーを EZ-USB® CX3™にインターフェースする方法

著者: Manu Kumar、Savan Javia

ソフトウェア バージョン: EZ-USB FX3 SDK1.3.3

関連リソース: こちらをクリックしてください。

その他のサンプル コードが必要な場合は、以下のとおり対応いたします。

USB SuperSpeedサンプルコードのリストについては、http://www.cypress.com/101781にアクセスしてください。

第 1 世代の USB 3.1 (以前に USB 3.0 と呼ばれた) が提供した広帯域幅は、周辺装置を USB に接続する IC への要求を高度にしています。本アプリケーション ノートでは、非圧縮データを PC にストリーミングするカメラ (EZ-USB CX3 と通信する MIPI CSI-2 のイメージ センサー) でよく利用されている第 1 世代の USB 3.1 のアプリケーションに焦点を当てて説明します。本アプリケーション ノートでは、USB ビデオ クラス (UVC) の実装についても詳しく説明します。このクラスに準拠したカメラ デバイスは、PC 内蔵ドライバーおよび e-CAMView、Media Player Classic や VLC Media Player などのホスト アプリケーションを使用して動作できます。

目次

1 はじめに

第 1 世代の USB 3.1 が提供した広帯域幅は、周辺装置を USB に接続する IC に対する高度な要求をします。一般的な例としては、非圧縮データを PC にストリーミングするカメラがあります。第 1 世代の USB 3.1 では USB 帯域幅が増加したため、PC に接続するカメラの解像度も高くなりました。広帯域幅のカメラ インターフェース要件に対応するために携帯機器プロセッサ インターフェース (MIPI) と呼ばれる協会がカメラ シリアル インターフェース 2 (CSI-2) という仕様を策定しました。

本アプリケーション ノートは、このために特別に設計された EZ-USB FX3™の応用品種であるサイプレス EZ-USB CX3 を使用する MIPI CSI-2－USB 3.1 コンバータの実装について詳しく説明します。

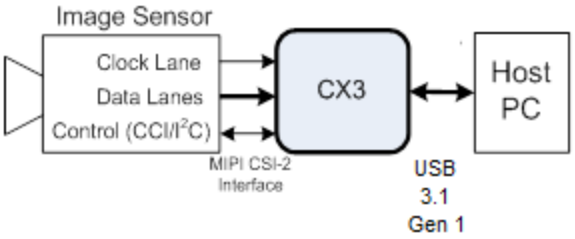
サイプレス EZ-USB 製品ファミリの初心者である場合、本アプリケーション ノートを理解しやすいように「AN75705 - Getting Started with EZ-USB® FX3™」アプリケーション ノートを参照してください。

CX3 はイメージ センサーからのデータを USB ビデオ クラス (UVC) と互換性のあるフォーマットに変換します。このクラスに準拠したカメラは、OS 内蔵ドライバーを使用して動作でき、e-CAMView、Media Player Classic や VLC Media Player などのホスト アプリケーションと互換性があります。

CX3 は FX3 からの派生品であり、FX3 との相違点は次のとおりです。

- MIPI CSI-2 レシーバー インターフェースを備えた MIPI CSI-2 コントローラーが追加されます。
- USB 2.0 OTG と充電器検出機能は取り除かれます。
- 2 つのクロック リファレンス (コア用に CLKIN、MIPI CSI-2 コントローラー用に REFCLK) が必要です。
- CLKIN については 19.2MHz の発振器のみがサポートされます。

図 1. カメラ アプリケーション



CX3 は FX3 の特殊バージョンであるため、すべての FX3 開発ツールとほとんどの FX3 リファレンスが CX3 に適用されます。この互換性の例としては、FX3 を使用するカメラ設計例 (パラレル インターフェース) について記載した AN75779 があります。双方のチップのデータ転送アーキテクチャが同一であるため、CX3 の内部動作についての参考資料の多くは、FX3 のアプリケーション ノートから持ってきています。

本アプリケーション ノートでは、利用可能なサイプレスのファームウェア プロジェクトを理解するのに役立つように詳しく説明します。お客様の設計がこのアプリケーション ノートと同じイメージ センサーを使用している場合は、コード変更はほとんど、あるいはまったく必要ありません。異なるイメージ センサーを使用している場合、本アプリケーション ノートも若干の変更が必要なコード モジュールを指摘します。

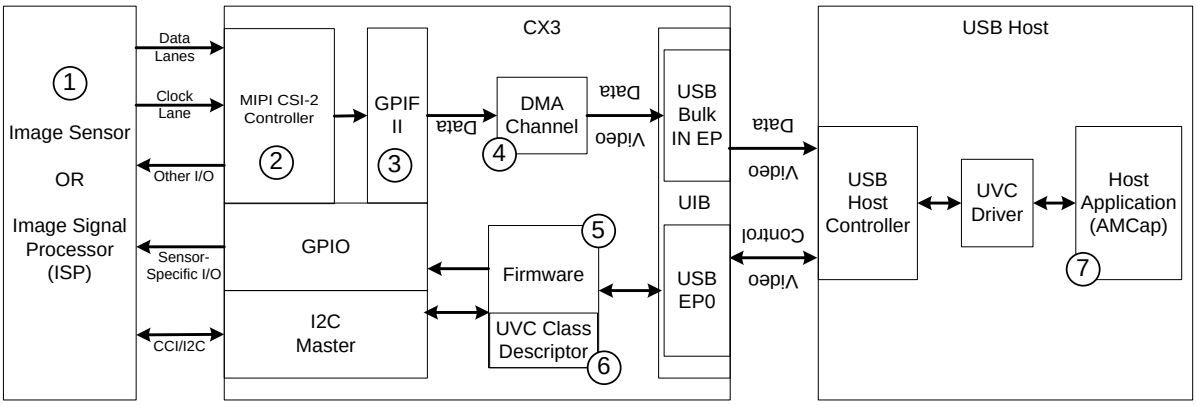
図 1 にカメラ アプリケーションを示します。右側はスーパースピード USB 3.1 ポートを搭載した PC です。左側は次のことに対応できる MIPI CSI-2 インターフェースを備えたイメージ センサーです。

- 1～4 本の MIPI CSI-2 データ レーン
- RAW8/10/12/14、YUV422 (CCIR/ITU 8/10 ビット)、RGB888/666/565、M-JPEG のような圧縮されたフォーマットとユーザー定義の 8 ビット、16 ビット、および 24 ビットの画像フォーマット

例えば、OV5640 センサーは VGA 60 fps、HD (720p) 60 fps、フル HD (1080p) 30 fps、5 M-Pixel 15 fps をサポートします。以下のセクションでは、*cycx3_uvc_ov5640* サンプルプロジェクト (FX3/CX3 SDK インストールで利用可能) を、VGA、HD、およびフル HD 解像度で YUV 形式をストリーミングするように設計する方法について詳しく説明します。

図 2 はサブブロックに番号を付けたブロックダイヤグラムです。各サブブロックで実行されるタスクを説明します。

図 2. システム ブロックダイヤグラム



1. MIPI CSI-2 に基づいたイメージ センサーを CX3 に接続し、カメラ制御インターフェース (CCI) バスを使用してイメージ センサーを設定します。
2. イメージ センサーから画像データを読み出し、パケットを取り出し、GPIF II ブロックに送信するよう、CX3 の MIPI CSI-2 コントローラーを設定します。2.1 節を参照してください。
3. 画像データフォーマットに従って GPIF II ブロックを設定します。2.2 節を参照してください。
4. 画像データを GPIF II ブロックから USB インターフェース ブロック (UIB) に転送する DMA チャネルを構築します。このアプリケーションでは、UVC 仕様に準拠するためにヘッダ データをイメージ センサーのビデオ データに追加する必要があります。このために、DMA は CPU が DMA バッファに必要なヘッダを追加できるように構成されています。このチャネルは、ビデオをイメージ センサーから PC にストリーミングするために最大帯域幅を使用できるように設計する必要があります。3 節を参照してください。
5. CX3 のファームウェアは、CX3 のハードウェアブロックを初期化し (5.3 節)、イメージ センサーと MIPI CSI-2 コントローラーを設定し (5.5 節)、UVC カメラとしてデバイスをエニュメレートし (4.3.1 節)、UVC 固有の要求を処理し (4.3.2 節)、CCI (I^2C) インターフェースを介してイメージ センサーにビデオ制御設定 (輝度など) を変換し (5.5 節)、ビデオ データ ストリームに UVC ヘッダを追加し (4.3.4 節)、USB にヘッダ付きのビデオ データを転送します (5.10 節)。
6. 適切な USB ディスクリプタを提供し、ホストが周辺装置を UVC に準拠したデバイスとして認識できるようにします。4 節を参照してください。
7. UVC 制御インターフェースを介してイメージ センサーを設定し、UVC ストリーミング インターフェースを介してビデオ データを受信するために、UVC ドライバーにアクセスするホスト アプリケーション (e-CAMView、Media Player Classic や VLC Media Player など) を使用します。7 節を参照してください。

カメラが USB 2.0 ポートに差し込まれたら、CX3 のファームウェアは CCI バスを使用して低い USB 帯域幅に適合するために下げられるフレーム レートとフレーム サイズを選びます。ホストはビデオ制御インターフェースを任意に使用して、輝度、コントラスト、色相、飽和、露光、オートフォーカスや PTZ (パン、チルト、ズーム) の調整をカメラに送信できます。

2 イメージ センサーを CX3 へインターフェースする

MIPI CSI-2 イメージ センサーを CX3 に接続し、そこからデータを読み出すには、CSI-2 インターフェースと CX3 の DMA 機能を理解する必要があります。

2.1 MIPI CSI-2 インターフェース

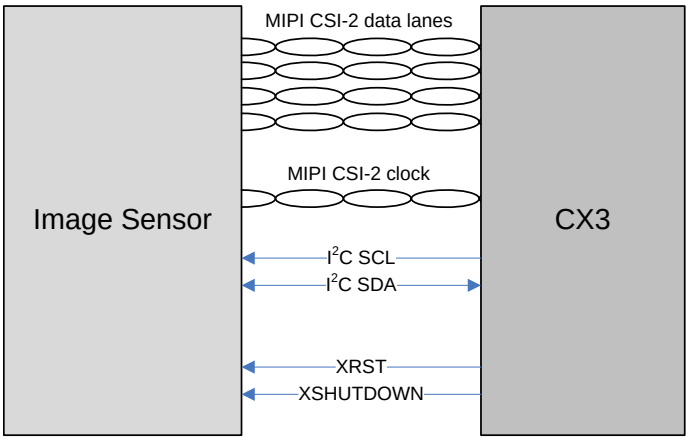
カメラ アプリケーションが精密になるため、高解像度のイメージセンサーに対して高い要求が寄せられます。この要求は、拡大が困難で、多くの相互接続を必要とする一般的なパラレル イメージ センサー インターフェースの限界に迫っています。そこで、MIPI アライアンス (携帯機器産業プロセッサ インターフェース協会) は、カメラ シリアル インターフェース 2 (CSI-2) 規格を作成し、広範囲のイメージング ソリューションをサポートする標準的、強固、低消費電力、高速シリアル インターフェースを策定しました。

MIPI CSI-2 インターフェースは、データ信号とクロック信号を持つ単方向の差動シリアル インターフェースです。最大 4 本のデータ レーンがあり、それぞれ最大 1Gbps でデータを転送します。

イメージ センサーを設定するために使用される制御インターフェースは、I²C 規格に準拠し、カメラ制御インターフェース (CCI) と呼ばれています。

CX3 の MIPI CSI-2 コントローラーは、イメージ センサーが一般的に必要とするこれら 2 種類のインターフェースと複数の追加信号を提供します。

図 3. CX3 とイメージ センサーとのインターフェース



このインターフェースにある 3 種類の追加信号を次に示します。

XRST: イメージ センサーはコンフィギュレーション前に CX3 からのリセット信号を常に必要とします。このリセットは CX3 の XRST ピンを使用して行います。

XSHUTDOWN: ホストがイメージ センサーからビデオ ストリームを要求していない場合は、センサーはシャットダウンまたはスタンバイ モードに入り消費電力を低減できます。これは、CX3 の XSHUTDOWN ピンを使用して制御できるセンサー内のシャットダウン端子を使用して行われます。

MCLK: 外部クロック発振器により供給される、イメージ センサーに必要なマスター (またはリファレンス) クロックです。

CSI-2、CCI および 3 つの追加信号のピン マッピングについては、データシート「EZ-USB® CX3™ MIPI CSI-2 to SuperSpeed USB Bridge Controller」を参照してください。また完全な例については、CX3 RDK 回路図を参照してください。

MIPI CSI-2 コントローラーは、1～4 つのデータ レーン、様々なデータ形式 (RAW、YUV や MJPEG など)、および様々なカメラの解像度に対応するように設定できます。詳細については、5.5 節を参照してください。このアプリケーションでは、4 つのデータレーンに YUV 画像データを送信するように構成されたセンサーを使用します。

構成された後、MIPI CSI-2 コントローラーは、イメージ センサーからシリアル画像データを受け取り、パケットを取り出し、さらにパラレル インターフェースを介して送信されるパラレル データにそれを変換します。このインターフェースは次の信号を使用します。

- FV: フレーム有効 (フレームの開始と終了を示します)
- LV: ライン有効 (ラインの開始と終了を示します)
- PCLK: ピクセル クロック
- データ: 画像データ用の 8 ビット 、16 ビット、24 ビットのデータ バス

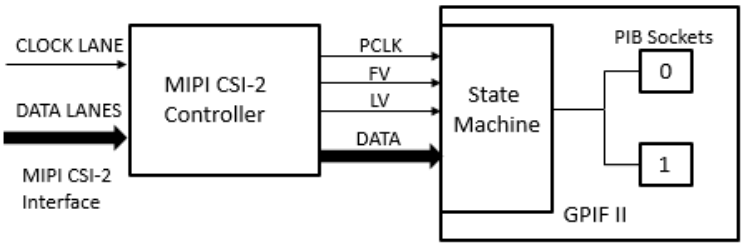
この信号のタイミングは AN75779 の 3.1 節で説明するパラレル インターフェースと同じです。

CX3 で使用できる最大 GPIF II データ バス幅が 24 ビットで、サポートされる最大 PCLK が 100MHz であるため、CX3 にサポートされる最大スループットは 2.4Gbps です。

2.2 GPIF II ブロック

MIPI CSI-2 コントローラーからのパラレル出力インターフェースは、プロセッサ インターフェース ブロック (PIB) の一部である GPIF II ブロックに接続されます。

図 4. GPIF II ブロック



GPIF II ブロックはステート マシンを使用してこのインターフェースからデータを 2 個の DMA ソケットに読み出します。データ転送の方法を把握するためには、CX3 の DMA 機能についての理解が必要となります。

2.2.1 DMA の基本

CX3 の DMA (ダイレクト メモリ アクセス) は、任意の 2 ブロック間で高速で中断しないデータ転送を可能にします。これらのデータ転送を理解するには次の用語を理解することが重要です。

- ソケット
- DMA ディスクリプタ
- DMA バッファ

ソケットは周辺装置のハードウェア ブロックと CX3 RAM との接点です。USB、GPIF、UART、SPI などの CX3 上の各周辺装置のハードウェア ブロックには、それぞれに関連付けられた一定数のソケットがあります。各周辺装置を流れる個別のデータ フロー数は、それに対応するソケット数に等しいです。ソケットの実装は、アクティブな DMA ディスクリプタを指すレジスター式とソケットに対応するイネーブルまたはフラグ割込みを含みます。

DMA ディスクリプタは CX3 RAM に割り当てられたレジスター式です。DMA バッファのアドレスとサイズの情報および次の DMA ディスクリプタへのポインタを格納してます。これらのポインタは DMA ディスクリプタ チェーンを形成します。

DMA バッファは RAM の一部であり、CX3 デバイスを介して転送されるデータの中間記憶装置として使用されます。DMA バッファは CX3 ファームウェアによって RAM から割り当てられ、それらのアドレスは DMA ディスクリプタの一部として格納されます。

2.2.2 2 個のソケットが使用される理由

2 個のソケットが使用される理由を理解するにあたり、ソケットの詳細を理解する必要があります。

ソケットはイベントにより互いに直接通知するか、または、割込みにより CX3 CPU に通知します。この信号方式はファームウェアにより設定されます。例えば、GPIF II ブロックから USB ブロックまでのデータ ストリームを考えます。GPIF ソケットは DMA バッファをデータで充填したことを USB ソケットに通知でき、USB ソケットは DMA バッファが空になったことを GPIF ソケットに通知できます。この実装は *自動 DMA チャネル*と呼ばれています。自動 DMA チャネルの実装は通常、CX3 CPU がデータ ストリームでデータの修正が必要ない場合に使用されます。

一方、GPIF ソケットは CX3 CPU に割込みを送信し、GPIF ソケットが DMA バッファを充填したことも通知できます。CX3 CPU はこの情報を USB ソケットに送ります。USB ソケットは CX3 CPU に割込みを送信し、USB ソケットが DMA バッファを空にしたことを通知できます。そして CX3 CPU はこの情報を GPIF ソケットに送り返します。この実装は *手動 DMA チャネル*と呼ばれます。

手動 DMA チャネルの実装は通常、CX3 CPU がデータ ストリームのデータを追加、除去あるいは修正する必要がある場合に使用されます。本アプリケーション ノートのファームウェア例では、ファームウェアが UVC ビデオ データ ヘッダを追加する必要があるため、手動 DMA チャネルの実装を使用します。

DMA バッファにデータを書き込むソケットは、プロデューサ ソケット プロデューサ ソケット と呼ばれます。DMA バッファからデータを読み出すソケットは、*コンシューマ ソケット*と呼ばれます。ソケットはデータ管理のために DMA ディスクリプタに格納された DMA バッファ アドレス、DMA バッファ サイズおよび DMA ディスクリプタ チェーンを使用します。

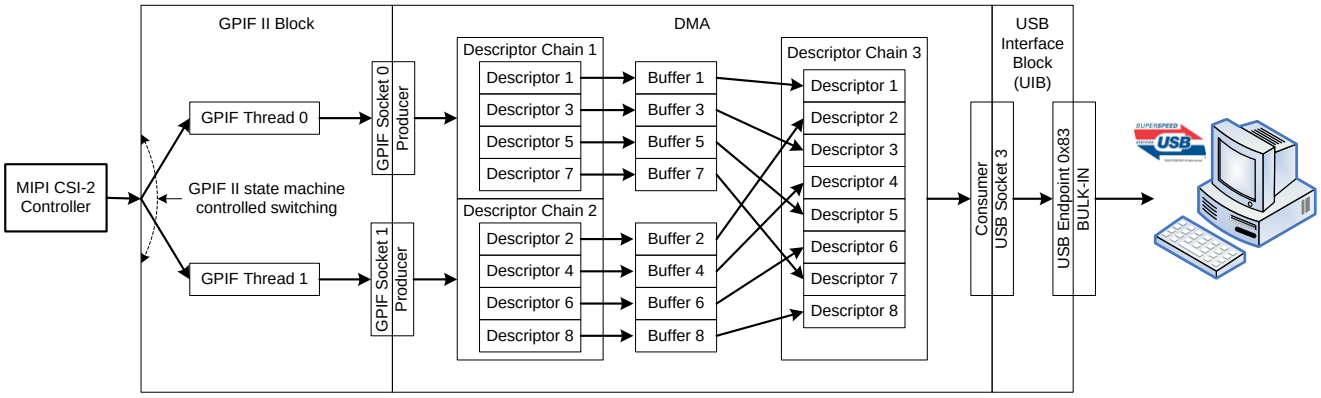
ソケットは、DMA バッファを満杯か空にした後、DMA ディスクリプタから他の DMA ディスクリプタに切り替えるのに限られた時間 (最大で数マイクロ秒) を要します。切り替えの間、ソケットはデータを転送できません。GPIF II ブロックでは 2 個の GPIF スレッドを使用して、このレイテンシを除去できます。

GPIF スレッドはソケットにデータ ピンを接続する GPIF II ブロックの専用データ パスです。一度に 1 個の GPIF スレッドだけがデータを転送できます。

GPIF スレッド選択のメカニズムはマルチプレクサに似ています。アクティブな GPIF スレッドを切り替えるとデータ転送用のアクティブなソケットが切り替わります。結果として、データ転送用の DMA バッファが変更されます。この切り替えは 1 クロック サイクルのレイテンシがあるため、本質的には瞬間的に行われます。GPIF II ステート マシンはこの切り替えを DMA バッファの境界で行うため、新しい DMA ディスクリプタへの GPIF ソケット切り替えのレイテンシをマスクできます。これにより DMA バッファが満杯になった時、GPIF II ブロックはセンサーからデータを損失せずに取り込めます。

図 5 には本アプリケーションで使用するソケット、DMA ディスクリプタ、DMA バッファ接続およびデータ フローを示します。2 個の GPIF スレッドは、代替 DMA バッファを充填するために使用されます。これら GPIF スレッドは、個別の GPIF ソケット (プロデューサ ソケットとして動作) と DMA ディスクリプタ チェーン (ディスクリプタ チェーン 1 とディスクリプタ チェーン 2) を使用します。USB ソケット (コンシューマ ソケットとして動作) は、正しい順序でデータを読み出すために 3 番目の DMA ディスクリプタ チェーン (ディスクリプタ チェーン 3) を使用します。ソケット、ソケット切り替えおよび関連する遅延についての詳細は 3 節を参照してください。

図 5. CX3 データの転送アーキテクチャ



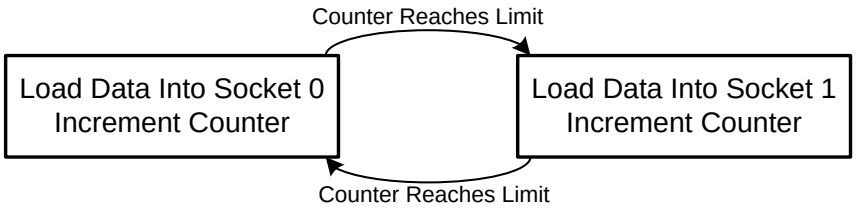
2.2.3 GPIF II ステート マシン

GPIF II ブロックは内部でステート マシンを使用して、MIPI CSI-2 コントローラーのパラレル出力インターフェースからビデオ データを読み出します。

前節 (2.2.2 節) で説明したように、ステート マシンは DMA ディスクリプタ (および実際には、DMA バッファ) 間で切り替える時にデータの損失を防ぐためにビデオ データを 2 個のソケットにロードします。

これはソケットに読み出したデータ量を継続して追跡するためにカウンターを使用し、カウンターが限界値に達すると、他のソケットに切り替わることで行われます。カウンターの限界値は DMA バッファのサイズに設定されます。

図 6. 2 個のソケットへのデータ転送



カウンターは、クロック サイクルごとに 1 増加します。したがって、インターフェースのデータ バス幅によってカウンター限界値が変わります。例えばデータ バス幅が 16 ビット、DMA バッファ サイズが 16KB (即ち、16,384) の場合、サイクルごとに 2 バイトの読み出しが行われるため、プログラムされる限界値は (16384/2) – 1 = 8191 になるはずです。

一般的に DMA バッファ カウント限界値は次のとおりです。

$$count = \left(\frac{producer\_buffer\_size(L)}{data\_bus\_width\ (in\ bytes)} \right) - 1$$

バス幅について:

GPIF II データバス幅は、イメージ センサーのデータ フォーマットに応じて選択される必要があります。このアプリケーション ノートでは、センサーが 16 ビットの YUV422 データを出力するように構成されているため、データ バスが 16 ビットに設定されます。各イメージ フォーマットのバス幅の詳細については、FX3 SDK API ガイドの「**CyU3PMipicsiDataFormat_t**」節を参照ください。

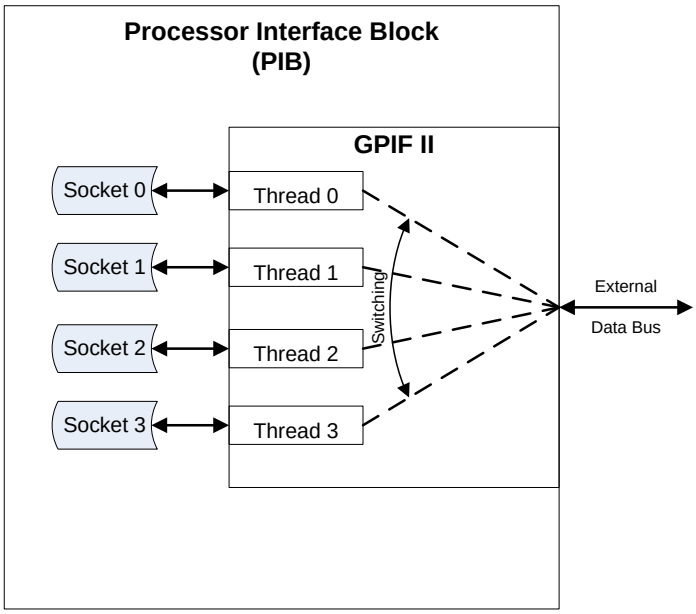
フレームが送信された後、ステート マシンはフレーム転送の正常終了を示すために CPU を割り込んで、その結果ヘッダを追加して他のフレームの終了タスクを実行できます。

次の節ではデータをストリーミングする DMA チャネルおよび UVC をサポートするファームウェアの詳細を説明します。

3 DMA システムのセットアップ

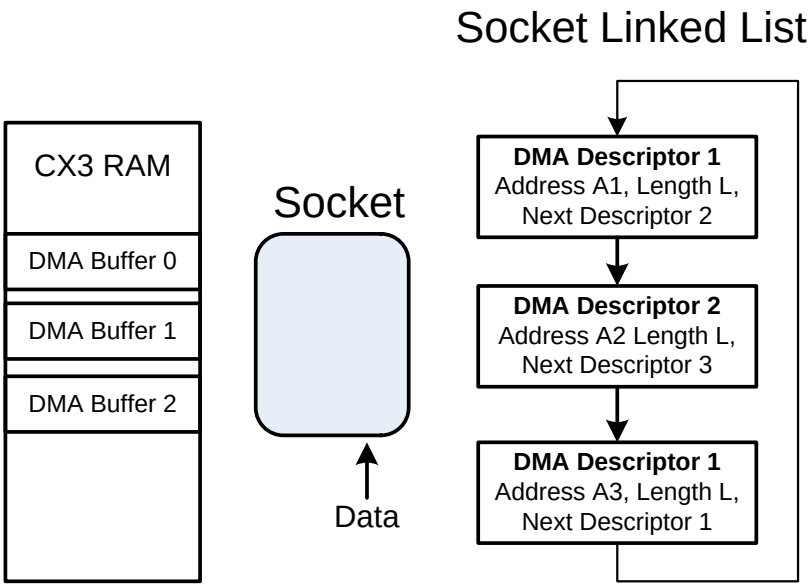
GPIF II ブロックは最大 100MHz で動作し、最大 24 ビットのデータ (300Mbps) を持ちます。内部 DMA バッファにデータを転送するには、GPIF II は、DMA プロデューサ ソケットに接続された 2 個の GPIF スレッドを使用します (2.2.2 節で説明されます)。ソケットと GPIF スレッドのデフォルトのマッピング (図 7) はこのアプリケーションに使用されます。ここでは、ソケット 0 が GPIF スレッド 0 に、ソケット 1 が GPIF スレッド 1 に接続されます。この場合で、利用可能なスレッドの 4 つの中で、2 つだけが使用されることにご注意ください。GPIF のスレッドの切り替えは、前節で説明した GPIF II ステート マシンで実行されます。

図 7. GPIF II ソケットとスレッドのデフォルト マッピング



DMA 転送を理解するために、2.2.1 節で説明されたソケットの概念をさらに次の 4 つの図に示します。図 8 に、ソケットの 2 つの主な属性であるリンク リストとデータ ルータを示します。

図 8. ソケットが DMA ディスクリプタのリストに従ってデータを送信



ソケット リンク リストは DMA ディスクリプタと呼ばれるメイン メモリ内の一連のデータ構造です。それぞれのディスクリプタは、DMA バッファのアドレス (An) と長さ (*L*) および次の DMA ディスクリプタへのポインタを指定します。ソケットは動作する時、DMA ディスクリプタを 1 つずつ読み出し、ディスクリプタのアドレスと長さで指定された DMA バッファにデータを転送します。*L* バイトが転送された時、ソケットは次のディスクリプタを読み出して、バイトを別の DMA バッファに引き続き転送します。

この構造により、DMA バッファをメモリ内の場所を問わず何個でも作って自動的に一緒に連結できるため、ソケットを多目的に使用できます。例えば図 9 のソケットは、繰り返しループで DMA ディスクリプタを読み出します。

図 9. DMA ディスクリプタ 1 と共に動作するソケット

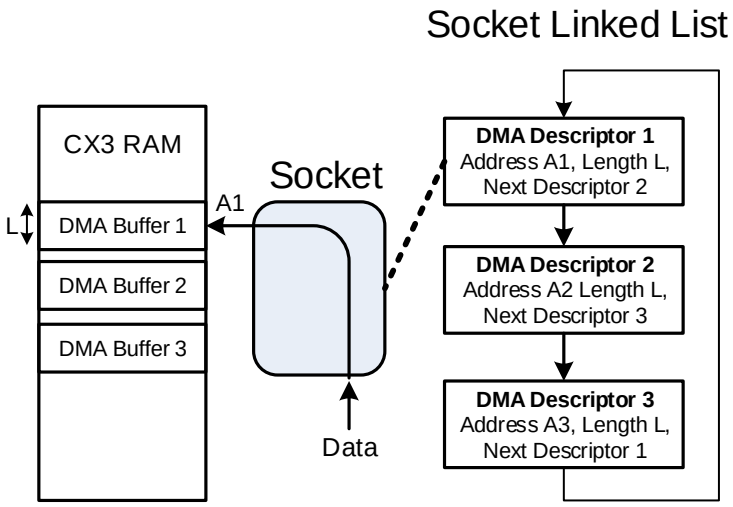


図 9 では、ソケットは DMA ディスクリプタ 1 をロードしました。DMA ディスクリプタはソケットが *L* バイトを転送するまで、A1 から始まるバイトを転送するように指示します。それに続いて、ソケットは DMA ディスクリプタ 2 を読み出し、*L* バイトを転送するまで、A2 から始まるバイトを転送します (図 10)。

図 10. DMA ディスクリプタ 2 と共に動作するソケット

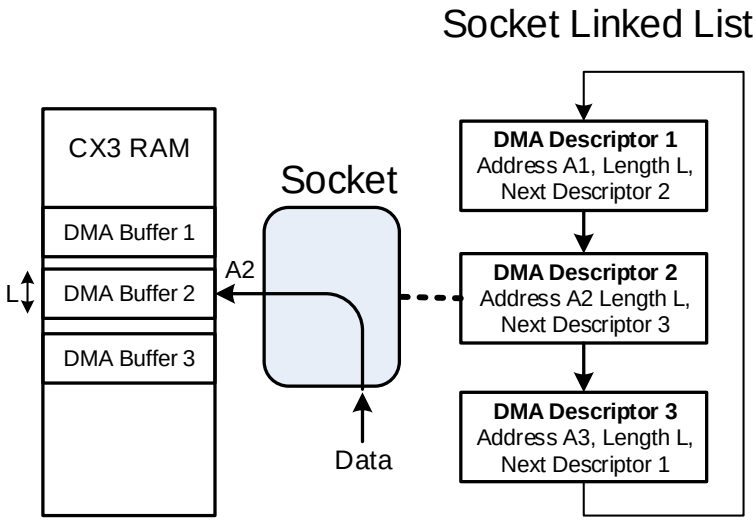


図 11 ではソケットは 3 番目の DMA ディスクリプタを読み出して、A3 から始まってデータを転送します。*L* バイトを転送した後、DMA ディスクリプタ 1 に戻って動作を繰り返します。

図 11. DMA ディスクリプタ 3 と共に動作するソケット

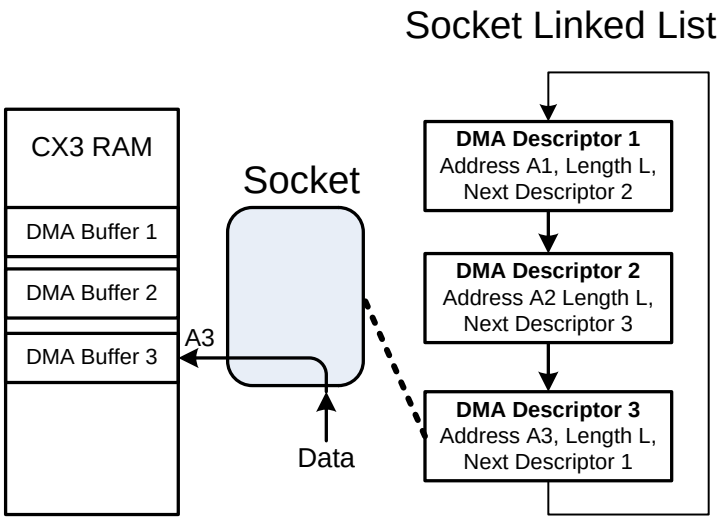
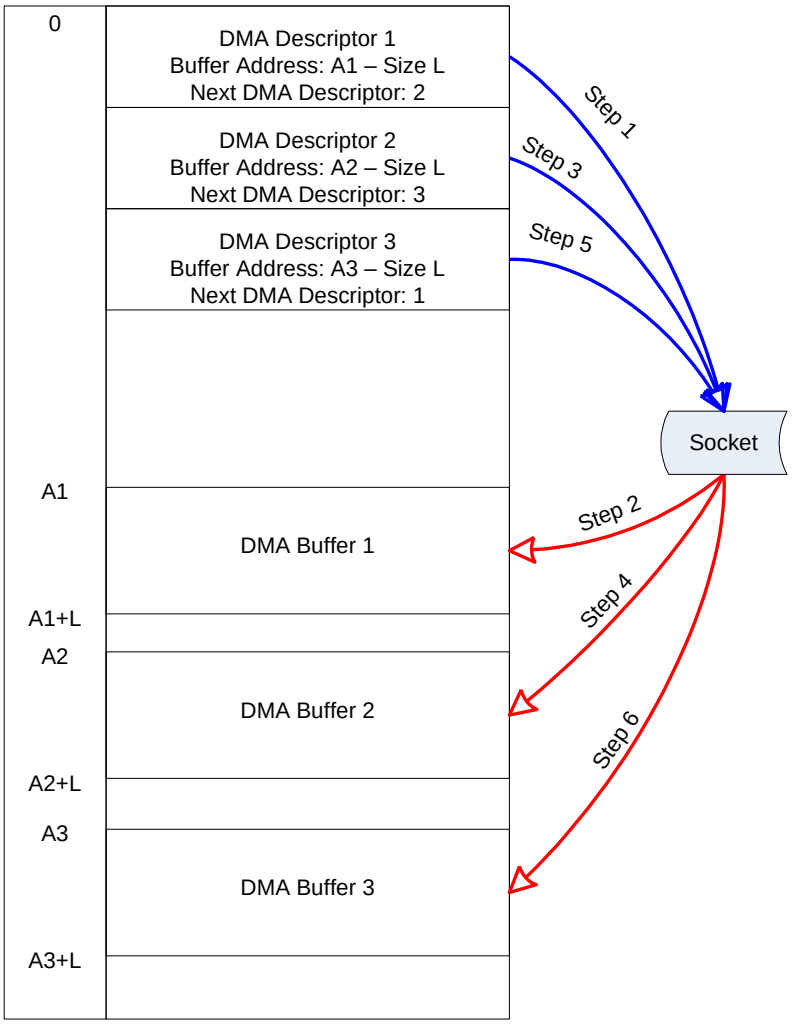


図 12 には、詳細な DMA データ転送を示します。この例は、円形ループで連結した長さ L の 3 つの DMA バッファを使用しています。CX3 メモリ アドレスは左側にあります。青色の矢印はソケットがソケット リンク リストのディスクリプタをメモリからロードしていることを示します。赤色の矢印は、結果となるデータ パスを示します。以下の手順は、データが内部 DMA バッファに転送される時のソケットのシーケンスを示します。

ステップ 1: DMA ディスクリプタ 1 をメモリからソケットにロードします。DMA バッファ位置 (A1)、DMA バッファ サイズ (L)、および次のディスクリプタ (DMA ディスクリプタ 2) の情報を取得します。ステップ 2 に進みます。

ステップ 2: A1 から始まって DMA バッファ位置にデータを転送します。DMA バッファ サイズである L のデータ量を転送した後、ステップ 3 に進みます。

図 12. DMA 転送例



ステップ 3: 現時点の DMA ディスクリプタ 1 が指している DMA ディスクリプタ 2 をロードします。DMA バッファ位置 (A2)、DMA バッファ サイズ (L) および次のディスクリプタ (DMA ディスクリプタ 3) の情報を取得します。ステップ 4 に進みます。

ステップ 4: A2 から始まって DMA バッファ位置にデータを転送します。DMA バッファ サイズである L のデータ量を転送した後、ステップ 5 に進みます。

ステップ 5: 現時点の DMA ディスクリプタ 2 が指している DMA ディスクリプタ 3 をロードします。DMA バッファ位置 (A3)、DMA バッファ サイズ (L) および次のディスクリプタ (DMA ディスクリプタ 1) の情報を取得します。ステップ 6 に進みます。

ステップ 6: A3 から始まって DMA バッファ位置にデータを転送します。DMA バッファ サイズである *L* のデータ量を転送した後、ステップ 1 に戻ります。

この単純な方式は、カメラ アプリケーションの問題があります。ソケットはメモリから次の DMA ディスクリプタを読み出すのに通常 1µs かかります。この転送では、データ転送の途中に一時停止が発生すればビデオ データは失われます。この損失を防ぐために、DMA バッファ サイズはビデオラインの長さの倍数に設定できます。これにより、DMA バッファ切り替えの一時停止時間はビデオ ラインの非アクティブ時間 (即ち、センサーの垂直帰線消去時間) と一致します。しかし、この方法は、例えばビデオの解像度が変更された場合、柔軟性を持ちません。

DMA バッファ サイズをライン サイズに全く等しく設定することは、BULK 転送にとって第 1 世代の USB 3.1 の最大バースト速度が良い効果を出さないため、良い解決策ではありません。第 1 世代の USB 3.1 は BULK エンドポイント上で最大 16 バースト (1 バーストが 1024 バイト) を可能にします。そのため、DMA バッファ サイズが 16KB に設定されます。

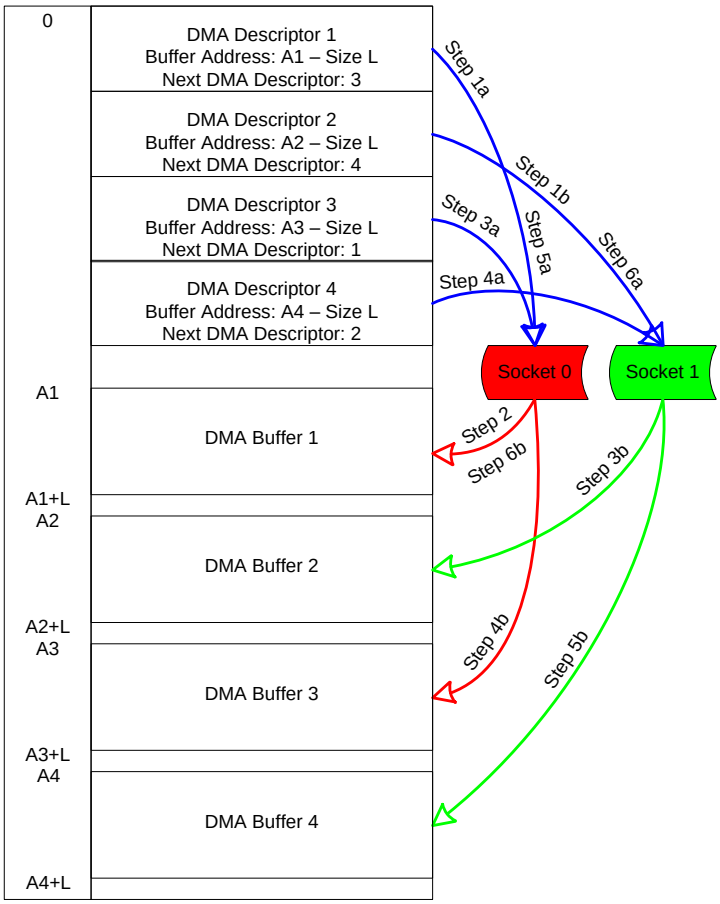
より良い解決策は、ソケットが 1 クロック サイクル以内にレイテンシなしに切り替わるという利点を利用することです。したがって、4 個のインターリーブ DMA バッファにデータを格納するために、*2* つのソケットを使用することは意味があります。

注: データの損失を防止するために、ソケットごとに少なくとも 2 つのバッファが必要となります (すなわち、ソケットが 1 つのバッファに書き込んでいる間にホストが他のバッファから読み出せるという 2 つのバッファです)。

注: 図 5 に示すもの (ソケットごとに 4 バッファ) と次の図に示すもの (ソケットごとに 2 バッファ) が異なります。それは説明を簡略化するためです。しかし、実際のファームウェアはソケットごとに 4 つのバッファを設定します。

デュアル ソケットを使用したデータ転送は、実行ステップを番号付けした図 13 で説明します。ソケット 0 とソケット 1 の DMA バッファへのアクセスは、それぞれ赤色と緑色の矢印 (個別のソケットのデータ経路) で区別されます。各ステップの「a」と「b」部分は同時に実行されます。このハードウェアの並行動作により、DMA ディスクリプタの回復デッド タイムが無くなり、GPIF II がデータを内部メモリに連続してストリーミングできます。これらのステップは、 図 12. に示す「ステップ」ラインに対応します。

図 13. デュアル ソケットを使用したシームレスな転送



ステップ 1: ソケットの初期化時に、ソケット 0 とソケット 1 は、それぞれ DMA ディスクリプタ 1 と DMA ディスクリプタ 2 をロードします。

ステップ 2: 転送データの準備ができると、ソケット 0 はすぐにデータを DMA バッファ 1 に転送します。転送の長さは *L* です。この転送が完了すると、ステップ 3 に進みます。

ステップ 3: GPIF II の固定機能ステート マシンは GPIF スレッドを切り替えるため、データ転送用のソケットも切り替わります。ソケット 1 が DMA バッファ 2 へのデータ転送を開始すると同時に、ソケット 0 は DMA ディスクリプタ 3 をロードします。ソケット 1 が L のデータ量の転送を終了するまでに、ソケット 0 は DMA バッファ 3 へのデータ転送の準備ができます。

ステップ 4: ここで、GPIF II は元の GPIF スレッドに切り替わります。ソケット 0 は長さ L のデータを DMA バッファ 3 に転送します。同時に、ソケット 1 は DMA ディスクリプタ 4 をロードして、DMA バッファ 4 へのデータ転送の準備ができます。ソケット 0 が長さ L のデータの転送を終了したら、ステップ 5 に進みます。

ステップ 5: GPIF II は、ソケット 1 のデータを DMA バッファ 4 に送信します。同時にソケット 0 は DMA ディスクリプタ 1 をロードして、DMA バッファ 1 にデータを転送する準備をします。ソケット 1 を初期化せずにデータを同時に転送する点を除き、ステップ 5a はステップ 1a と同様であることにご注意ください。

ステップ 6: GPIF II は再びソケットを切り替えて、ソケット 0 は DMA バッファ 1 への長さ L のデータの転送を開始します。DMA バッファは、ここまでは UIB コンシューマ ソケットにより使い果たされたから、空になっていることが想定されます。それと同時に、ソケット 1 は DMA ディスクリプタ 2 をロードして、DMA バッファ 2 へのデータ転送の準備ができます。ここで、サイクルは実行経路のステップ 3 に進みます。

コンシューマ側 (USB) が GPIF II から次のビデオ データのチャンクを受信するのに間に合うように DMA バッファを空にして解放した場合にのみ、GPIF II ソケットはビデオ データを転送できます。コンシューマの速度が十分に速くなかったら、ソケットの DMA バッファへの書き込みが無視されるためソケットはデータを失います。その結果、バイト カウンターは実際の転送との同期を失います。このような同期喪失は次のフレームに伝播することがあります。したがって、フレームが長時間転送されない場合は、クリーンアップ メカニズムが必要となります。このメカニズムを 5.1 節で説明します。

フレーム転送は、次の 4 つの可能なシナリオのいずれかで終了できます。

- ソケット 0 が一杯の DMA バッファを転送した
- ソケット 1 が一杯の DMA バッファを転送した
- ソケット 0 が部分的な DMA バッファを転送した
- ソケット 1 が部分的な DMA バッファを転送した

最後の 2 つシナリオでは、CPU は USB コンシューマに部分的な DMA バッファを転送する必要があります。

このアプリケーション ノートでは、*SDK_INSTALL_PATH*\firmware\cx3_examples\cycx3_uvc_ov5640 フォルダにあるプロジェクトを使用します。ここで *SDK_INSTALL_PATH* は FX3 SDK インストールのロケーションです。32 ビット システムの初期設定インストール パスは、*C:\Program Files\Cypress\EZ-USB FX3 SDK\1.3* です。64 ビット システムの場合、このパスは *C:\Program Files (x86)\Cypress\EZ-USB FX3 SDK\1.3* です。

DMA チャネルは、*cycx3_uvc.c* ファイル内の「CyCx3AppInit」と呼ばれるファイルの関数を使って初期化されます。DMA チャネル コンフィギュレーションの詳細は、同じ関数内にある「dmaCfg」構造体でカスタマイズされます。DMA チャネル タイプは MANUAL_MANY_TO_ONE に設定されます。

また、第 1 世代の USB 3.1 ホストへデータをストリーミングする USB エンドポイントは、BULK エンドポイントを介した 16 パケット (パケット サイズが 1024 バイト) のバースト転送を可能にするように設定されます。これは、エンドポイントが「CX3_EP_BULK_VIDEO」に固定して設定された「CyU3PSetEpConfig」関数内に渡された「endPointConfig」構造体を使用して設定されます。

3.1 DMA バッファ

本節では CX3 DMA バッファの作成方法とこのアプリケーション内での使用方法の概要を説明します。DMA チャネルに不可欠な部分を 2.2.1 節に説明します。

本アプリケーションでは、GPIF II ユニットが生産者 (プロデューサ) であり、USB ユニットが消費者 (コンシューマ) です。このアプリケーションでは、データ損失を避けるために、GPIF II ブロック内で GPIF スレッドの切り替え機能を使用します。

プロデューサ ソケットは、DMA ディスクリプタをロードすると、対応する DMA バッファをチェックして書き込み動作の準備ができているかを確認します。プロデューサ ソケットは、DMA バッファが空であることを検出した場合、CX3 RAM にデータを書き込むために *アクティブ* 状態に移行します。プロデューサ ソケットは、書き込み動作用に DMA バッファをロックします。

プロデューサ ソケットは DMA バッファへの書き込みを終了すると、コンシューマ ソケットが DMA バッファにアクセスできるようにロックを解除します。このアクションは「バッファ ラップアップ」または簡単に「ラップアップ」と呼ばれます。その後、DMA ユニットは DMA バッファを CX3 RAM に転送するように指示されます。プロデューサ ソケットは、DMA バッファを生成したといわれます。プロデューサが積極的に DMA バッファを満たさない時にのみ、DMA バッファをラップアップする必要があります。

DMA バッファが完全に満たされると、フレーム中に繰り返すため、ラップアップ動作は自動的に行われます。プロデューサ ソケットは、DMA バッファのロックを解除して CX3 RAM に DMA バッファを転送し、空の DMA バッファへ切り替え、ビデオ データ ストリームの書き込みを継続します。

コンシューマ ソケットが DMA ディスクリプタをロードすると、読み出し動作の準備ができたか確認するために対応する DMA バッファをチェックします。コンシューマ ソケットは、DMA バッファが転送されたことを検出した場合、CX3 RAM からデータを読み出すためにアクティブ状態に移行します。コンシューマ ソケットは、読み出し動作用に DMA バッファをロックします。

- コンシューマ ソケットは DMA バッファからすべてのデータを読み出した後、プロデューサ ソケットが DMA バッファにアクセスできるようにロックを解除します。コンシューマ ソケットは、DMA バッファを消費したといわれます。
- 同じ DMA ディスクリプタがプロデューサ ソケットとコンシューマ ソケットによって同時に使用されている場合、DMA バッファの満杯／空の状態は、DMA ディスクリプタとソケット間のイベントを介してプロデューサ ソケットとコンシューマ ソケット間で自動的にやり取りされます。
- このアプリケーションでは、CPU が 12 バイトの UVC ヘッダを追加する必要があるため、プロデューサ ソケットとコンシューマ ソケットは異なる DMA ディスクリプタ セットをロードする必要があります。プロデューサ ソケットによりロードされた DMA ディスクリプタは、コンシューマ ソケットによりロードされた DMA ディスクリプタが指定する対応 DMA バッファから 12 バイトのオフセット補正をした DMA バッファを指します。
- プロデューサ ソケットとコンシューマ ソケットに対応する異なる DMA ディスクリプタに起因して、CPU はプロデューサ ソケットとコンシューマ ソケット間の DMA バッファ状態の情報伝達を管理する必要があります。そのため、DMA チャネルの実装が「手動 DMA」チャネルと呼ばれます。
- DMA バッファが GPIF II ブロックにより生成された後、CPU は割込みを介して通知されます。CPU はその後、ヘッダ情報を追加し、コンシューマ (USB インターフェース ブロック) に DMA バッファを転送します。
- GPIF II 側では、DMA バッファ内のビデオ データは自動的にラップアップされ、フレーム内の最後の DMA バッファ以外、CX3 RAM にすべてのバッファ内のデータを転送します。
- フレームの終わりで最終の DMA バッファは、完全に満たされない場合があります。この場合 CPU が介入して手動で DMA バッファをラップアップし、CX3 RAM にそれを転送します。これは、「強制的なラップアップ」と呼ばれます。

4 USB ビデオ クラス（UVC）

本節では、ビデオ データを PC へ転送するための、ユニバーサル シリアル バス (USB) 仕様の上にあるプロトコルである USB ビデオ クラスについて説明します。

UVC クラスに準拠するには、2 個の CX3 コード モジュールが必要です。

- エニュメレーション データ
- 動作コード

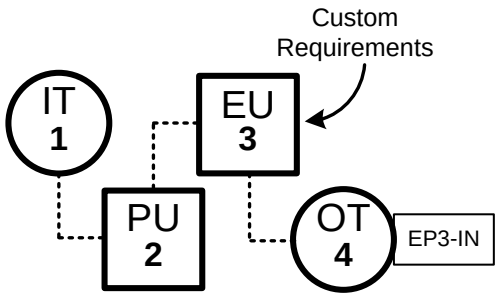
4.1 エニュメレーション データ

SDK インストールの *cycx3_uvc_ov5640* プロジェクトは、*cycx3_uvcdscr.c* と名付けられた、UVC エニュメレーション データを持つファイル (4.3.1 節を参照) を含みます。UVC ディスクリプタの形式を定義する USB 仕様は usb.org から入手できます。本節ではディスクリプタについて概説します。UVC デバイスには 4 つの論理要素があります。各要素は 1 つのディスクリプタにより定義されます。

- カメラ入力端子 (IT)
- 出力端子 (OT)
- プロセッシング ユニット (PU)
- エクステンション ユニット (EU)

これら要素はディスクリプタ内で図 14 のように接続します。要素間の接続はディスクリプタの端子番号を関連付けることで行われます。例えば、入力 (カメラ) 端子ディスクリプタはその ID を 1 と宣言し、プロセッシング ユニット ディスクリプタはその入力接続の ID が 1 と指定します。したがって、プロセッシング ユニット ディスクリプタは論理的には入力端子に接続します。出力端子ディスクリプタは、どの USB エンドポイントを使用するかを示します。この場合は BULK-IN エンドポイント 3 です。

図 14. カメラのアーキテクチャの UVC 図



ディスクリプタは、ビデオの特性 (幅、高さ、フレーム レート、フレーム サイズ、ビット深度など) および制御特性 (輝度、露光、増幅率、コントラスト、パン／チルト／ズーム即ち PTZ) も含みます。

4.2 動作コード

ホストがカメラをエニュメレートした後、UVC ドライバーは一連の要求をカメラに送信して動作の特性を判断します。これは、「機能要求段階」と呼ばれます。この段階は、ホスト アプリケーションがビデオのストリームを開始するストリーム段階に先行します。

例えば、UVC デバイスはその USB ディスクリプタのいずれかで輝度制御をサポートしていることを示します。機能要求段階中に、UVC ドライバーはデバイスに問い合わせて、関連する輝度パラメーターを得ます。

ホスト アプリケーションが輝度値を変更するよう要求すると、UVC ドライバーは SET 制御要求 (SET_CUR) を発行して輝度を変更します。これは USB コントロール エンドポイント (EP0) により実行されます。

同様に、ホスト アプリケーションがサポートされたビデオ フォーマット、フレーム レート、フレーム サイズをストリーミングすることにすると、UVC ドライバーはストリーミング要求を発行します。要求は、PROBE および COMMIT という 2 種類あります。COMMIT 要求は変更を実行するために使用されますが、PROBE 要求は、UVC デバイスがストリーム モードへの変更を受け入れる準備ができたかを判断するために使用されます。ストリーミング モードは、イメージ フォーマット、フレーム サイズ、フレーム レートの組合せです。

4.3 USB ビデオ クラスの要件

本節ではどのように UVC 要件が満たされるかについてサンプル プロジェクトで説明します。UVC はデバイスに次の動作を要求します。

- UVC 固有の USB ディスクリプタでエニュメレートする
- USB ディスクリプタで報告された UVC 制御とストリーム機能の SET/GET UVC 固有の要求を処理する
- UVC 準拠の色フォーマットでビデオ データをストリーミングする
- 各イメージ ペイロードに UVC 準拠のヘッダを追加する

これらの要件の詳細は、UVC 仕様に記載されています。

4.3.1 UVC の USB ディスクリプタ

cycx3_uvcdscr.c ファイルは USB ディスクリプタの表を含みます。配列「CyCx3USBHSConfigDscr」(ハイスピード) および「CyCx3USBSSConfigDscr」(スーパースピード) は UVC 固有のディスクリプタを含みます。これらのディスクリプタは以下のツリー構造のサブディスクリプタを含みます。

- コンフィギュレーション ディスクリプタ
  - インターフェース アソシエーション ディスクリプタ
  - ビデオ制御 (VC) インターフェース ディスクリプタ
    - VC インターフェース ヘッダ ディスクリプタ
      - 入力 (カメラ) 端子ディスクリプタ
      - プロセッシング ユニット ディスクリプタ
      - エクステンション ユニット ディスクリプタ
      - 出力端子ディスクリプタ
    - VC ステータス割込みエンドポイント ディスクリプタ
  - ビデオ ストリーミング (VS) インターフェース ディスクリプタ
    - VS インターフェース入力ヘッダ ディスクリプタ
      - VS フォーマット ディスクリプタ
    - VS フレーム ディスクリプタ
  - BULK-IN ビデオ エンドポイント ディスクリプタ

    コンフィギュレーション ディスクリプタは、そのサブディスクリプタで USB デバイスの機能を定義する標準 USB ディスクリプタです。インターフェース アソシエーション ディスクリプタは、デバイスが標準 USB クラスに準拠していることをホストに示します。ここで、このディスクリプタはビデオ制御 (VC) インターフェースとビデオ ストリーミング (VS) インターフェースを備えた UVC 準拠のデバイスを報告します。2 つの個別のインターフェースを備えた UVC デバイスは USB 複合デバイスとなります。

ビデオ制御 (VC) インターフェース

VC インターフェース ディスクリプタおよびそのサブディスクリプタは、制御インターフェースに関連するすべての機能を報告します。例としては、輝度、コントラスト、色相、露光、PTZ 制御などがあります。

VC インターフェース ヘッダ ディスクリプタは、この VC インターフェースが属している VS インターフェースを指す UVC 固有のインターフェースディスクリプタです。

入力 (カメラ) 端子ディスクリプタ、プロセッシング ユニット ディスクリプタ、エクステンション ユニット ディスクリプタ、出力端子ディスクリプタは、それぞれの端子またはユニットがサポートしている機能を記述するビット フィールドを含みます。

カメラ端子は、ビデオ ストリームを送信するデバイスの露光や PTZ などの機械的な機能 (または相当するデジタル機能) を制御します。

プロセッシング ユニットは、それを介してストリーミングされるビデオの輝度、コントラスト、色相などのイメージ特性を制御します。

エクステンション ユニットは、標準の USB ベンダー要求のようにベンダー固有の機能を追加することを可能にします。この設計ではエクステンション ユニットは空ですが、ディスクリプタはカスタム機能用のプレースホルダとして含まれます。エクステンション ユニットを利用する場合、認識するよう変更しなければ標準ホスト アプリケーションはその機能を認識しません。

出力端子はそれらユニット (IT、PU、EU) とホスト間のインターフェースを記述します。VC ステータス割込みエンドポイント ディスクリプタは、割込みエンドポイント用の標準 USB ディスクリプタです。このエンドポイントは、UVC 固有のステータス情報を通信するために使用できます。このエンドポイントの機能は本アプリケーション ノートの範囲外です。

UVC 仕様ではクラス固有の制御要求の実装を容易に構成できるよう、これらの機能を分割します。ただし、これらの機能の実装はアプリケーションによって異なります。サポートされた制御機能は、それぞれの端子またはユニット ディスクリプタの「bmControls」ビット フィールド (*cycx3_uvcdscr.c*) で、対応する機能のビットを「1」にセットすることで報告されます。UVC デバイス ドライバーは、エニュメレーション時に制御の情報をポーリングします。情報のポーリングは EP0 要求により実行されます。そのようなすべての要求 (ビデオ ストリーミング要求を含む) は、*cycx3_uvc.c* ファイル内の CyCx3AppUSBSetupCB 関数により処理されます。

ビデオ ストリーミング (VS) インターフェース

VS インターフェース ディスクリプタおよびそのサブディスクリプタは、様々なフレーム フォーマット (非圧縮、MPEG、H.264 など)、フレーム解像度 (幅、高さ、ビット深度) およびフレーム レートを報告します。報告された値に基づいてホスト アプリケーションは、サポートされたフレーム フォーマット、フレーム解像度およびフレーム レートの組合せを選択してストリーミング モードを切り替えます。

VS インターフェース入力ヘッダ ディスクリプタは、それに続く VS フォーマット ディスクリプタの数を指定します。

VS フォーマット ディスクリプタは、イメージのアスペクト比および非圧縮や圧縮などの色フォーマットを含みます。

VS フレーム ディスクリプタは、イメージ解像度およびそれのサポートされたすべてのフレーム レートを含みます。カメラが異なる解像度をサポートする場合、複数の VS フレーム ディスクリプタが VS フォーマット ディスクリプタに続きます。

BULK-IN ビデオ エンドポイント ディスクリプタは、ビデオをストリーミングするのに使用するバルク エンドポイントの情報を含む標準 USB エンドポイント ディスクリプタです。

この例では、2 組の解像度とフレーム レートをサポートするために、2 つのフレーム ディスクリプタを使用します。そのイメージ特性は、以下の 3 つの表に示されるように 3 つのディスクリプタに含まれます (関連のあるバイト オフセットだけが示されます)。

表 1. VS フォーマット ディスクリプタの値

VS フォーマット ディスクリプタのバイト オフセット	特性	スーパースピードの値	ハイスピードの値
23～24	幅対高さ比	16:9	4:3

表 2. 第 1 の VS フレーム ディスクリプタの値

VS フレーム ディスクリプタのバイト オフセット	特性	スーパースピードの値	ハイスピードの値
5～8	解像度 (W、H)	0x780、0x438 (1920 × 1080)	0x140、0xF0 (320 × 240)
17～20	バイト単位での最大イメージ サイズ	0x3F4800 (1920 × 1080 × 2)	0x25800 (320 × 240 × 2)
21～24 および 26～29	100ns 単位でのフレーム間隔	0x51615 (30fps)	0x1B207 (90fps)

表 3. 第 2 VS フレーム ディスクリプタの値

VS フレーム ディスクリプタのバイト オフセット	特性	スーパースピードの値	ハイスピードの値
5～8	解像度 (W、H)	0x500、0x2D0 (1280 × 720)	0x280、0x1E0 (640 × 480)
17～20	バイト単位での最大イメージ サイズ	0x1C2000 (1280 × 720 × 2)	0x96000 (640 × 480× 2)
21～24、および 26～29	100ns 単位でのフレーム間隔	0x28B0A (60fps)	0x28B0A (60fps)

複数バイトの値が LSB ファースト (リトルエンディアン) で格納され送信される点にご注意ください。したがって、例えば最初のスーパースピード フレーム レートを 30fps とすると、フレーム間隔は次のように計算されます。

$$FrameInterval = \frac{1}{30\ fps \times 100ns} = 333333 = 0x51615$$

これはディスクリプタで 0x15、0x16、0x05 と 0x00 として保存されます。この設計は、上記の表の項目を変更することで、異なるイメージ解像度に対応できるように適用できます。

4.3.2 UVC 固有の要求

UVC 仕様では、USB 制御エンドポイント EP0 を使用して制御とストリーミング要求を UVC デバイスに通信します。これらの要求は、ビデオ関連の制御特性を調べ、変更するのに使用されます。UVC 仕様では、これらのビデオ関連の制御が機能として定義されています。これらの機能により、イメージ特性を変更する、またはビデオをストリーミングできます。

機能は、ビデオ制御特性 (輝度、コントラスト、色相など) またはビデオ ストリーミング モードの特性 (色フォーマット、フレーム サイズ、フレーム レートなど) です。機能は USB コンフィギュレーション ディスクリプタの UVC 固有のセクションを介して報告されます。各機能は属性を持っています。機能の属性は以下のとおりです。

- 最小値
- 最大値
- 最小値と最大値間の値の数
- 初期設定値
- 現行値

SET と GET は、2 タイプの UVC 固有の要求です。SET は属性の現行値を変更するのに使用され、GET は属性を読み出すのに使用されます。

UVC 固有の要求のリストは以下のとおりです。

- SET_CUR は唯一の SET 要求タイプです。
- GET_CUR は現在の値を読み出します。
- GET_MIN はサポートされた最小値を読み出します。
- GET_MAX はサポートされた最大値を読み出します。
- GET_RES は、解像度 (最小値と最大値間のサポートされた値を示すステップ値) を読み出します。
- GET_DEF は初期設定値を読み出します。
- GET_LEN はバイト単位で特性のサイズを読み出します。
- GET_INFO は状態／特定の機能のサポートを問い合わせます。

UVC 仕様はこれらの要求が与えられた機能ごとに必須または任意であるかを規定しています。例えば、SET_CUR 要求が特定の機能において任意である場合、その存在は GET_INFO 要求により判断されます。カメラがある機能に対する特定の要求をサポートしていない場合、これを示すために、要求がホストからカメラに発行される時、制御エンドポイントをストールします。

これらの要求には、対象となる機能を検証するバイト フィールドがあります。これらバイト フィールドは、4.3.1 節で説明された UVC 固有のディスクリプタと同じ構造を持つ階層を持っています。第 1 レベルはインターフェースを識別します (ビデオ制御またはビデオ ストリーミング)。

第 1 レベルがインターフェースをビデオ制御と識別した場合、第 2 レベルは端子かユニットを識別し、第 3 レベルはその端子かユニットの機能を識別します。例えば、対象となる機能が輝度制御である場合を下記に示します。

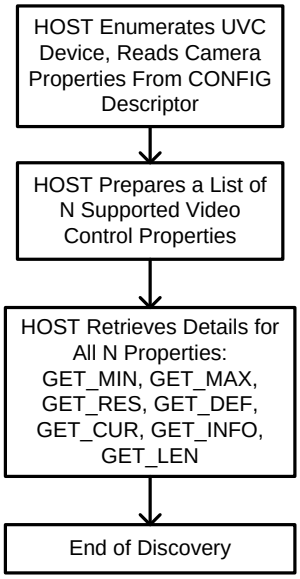
- 第 1 レベル=ビデオ制御
- 第 2 レベル=プロセッシング ユニット
- 第 3 レベル=輝度制御

第 1 レベルがインターフェースをビデオ ストリーミングと識別した場合、第 2 レベルは PROBE か COMMIT となります。第 3 レベルはありません。

UVC デバイスにストリーミングを開始させる、またはストリーミング モードを変更させるために、まずホストはデバイスが新しいストリーミング モードをサポートするかを判断します。このために、ホストは第 2 レベルを PROBE にセットして一連の SET と GET 要求を送信します。デバイスはストリーミング モードへの変更を受け入れるかまたは拒否します。デバイスが変更要求を受け入れた場合、ホストはこれを確認するために、第 2 レベルを COMMIT にセットして SET_CUR 要求を送信します。

以下の 3 つのフローチャートは、ホストがどのように UVC デバイスとやり取りするかを示します。

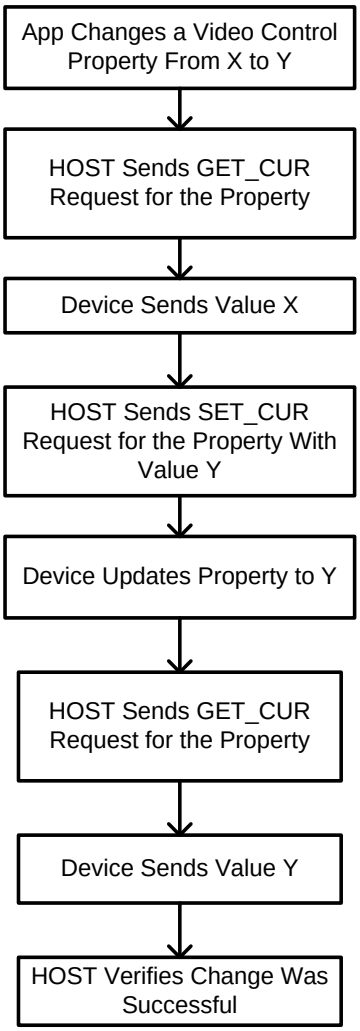
図 15. UVC エニュメレーションと検出フロー



UVC デバイスが USB に差し込まれると、ホストはそれをエニュメレートし、カメラがサポートする特性の詳細情報を調べます（図 15）。

操作中にカメラマンはホスト アプリケーションによって表示されたダイアログから、輝度などカメラの特性を変更できます。図 16 はこのやり取りを示します。

図 16. ホスト アプリケーションがカメラ設定を変更



ストリーミングを開始する前に、ホスト アプリケーションは、一連の PROBE 要求を発行して可能なストリーミングモードを調べます。デフォルト ストリーミング モードが決定された後、ホストは希望する解像度とフレームレートのフレームおよびフォーマット ID を持つ COMMIT 要求を発行します。このプロセスを図 17 に示します。COMMIT 要求と共にホストによって送信される構造を表 4 に示します。

この時点で、UVC ドライバーは UVC デバイスからビデオをストリーミングする用意ができます。

図 17. ホスト カメラのストリーミング前のダイアログ

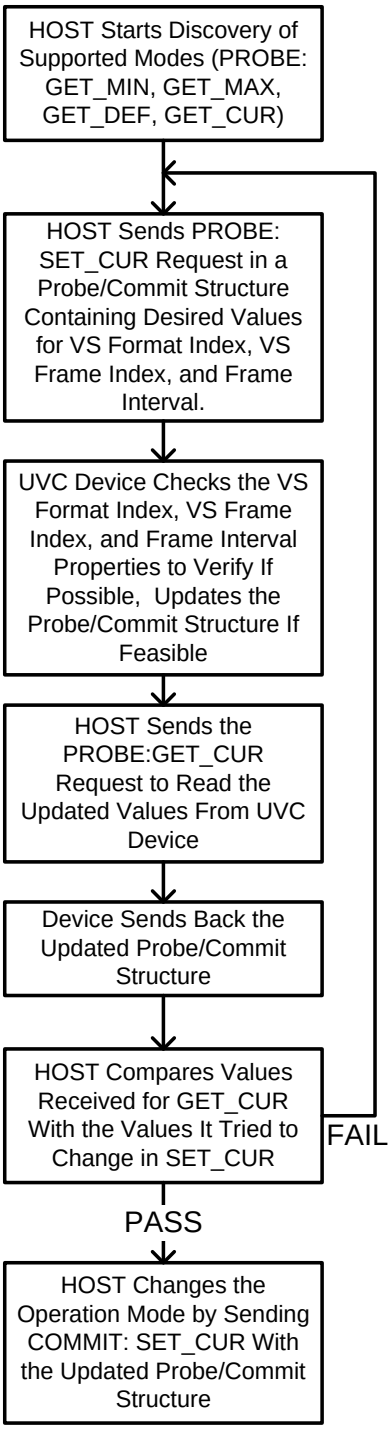


表 4. 1080p @ 30fps ストリームを開始する前にホストによって送信されるコミット構造の値

プローブ／コミット構造のバイト オフセット	特性	値
2	フォーマット インデックス	1
3	フレーム インデックス	1
4～7	100ns 単位でのフレーム間隔	0x51615 (30fps)
18～21	バイト単位での最大イメージ サイズ	0x3F4800 (1920 × 1080 × 2)

制御要求: 輝度、PTZ、色相、彩度、その他

イメージ センサーは、輝度、色相、彩度などの画像の機能の制御をサポートでき、外付けハードウェアが PTZ (パン、チルト、ズーム) のサポートのために加えられることがあります。

CX3 ファームウェアでこれをサポートするには、ビデオ制御ディスクリプタが特定の制御のサポートを示すよう修正する必要があります。例えば、輝度の制御をサポートするには、プロセッシング ユニット ディスクリプタの bmControls フィールドのビット 0 をセットする必要があります。詳細については、4.1 節で言及される UVC 仕様を参照してください。

これでホストは指定された要求を特定の端子に送信できるようになります。ホストが送信できる様々な要求のタイプについて 4.3.2 節で説明します。これらの要求は、CyCx3AppUSBSetupCB 関数により処理されます。

要求を受信すると、ファームウェアはイメージ センサーまたは関連するハードウェアに要求を送信できます。実装はアプリケーションによって異なります。

ストリーミング要求: プローブとコミット制御

CyCx3AppUSBSetupCB 関数はストリーミング関連の要求を処理します。UVC ドライバーが UVC デバイスからビデオをストリーミングする必要がある場合、第 1 段階は折衝です。この段階では、ドライバーは GET_MIN、GET_MAX、GET_RES、GET_DEF などの PROBE 要求を送信します。これに応じて CX3 ファームウェアは PROBE 構造体を返します。この構造体は、ビデオ フォーマット、ビデオ フレーム、フレーム レート、最大フレーム サイズ、ペイロード サイズ (UVC ドライバーが 1 回の転送で取り出せるバイト数) の USB ディスクリプタを含みます。

この例では、GET_CUR 要求が処理され、それに続いて現時点のアクティブな接続を確認し、適切な PROBE 構造体を送信します。

GET_CUR の処理に続き、SET_CUR 要求が処理されます。これらの要求はストリーミング段階の最初の時点で送信されます。

COMMIT 制御用の SET_CUR 要求は、要求が完了した後にホストがビデオのストリーミングを開始することを示します。ホストが送信したフレーム ディスクリプタ インデックスの種類 (ホストに送信されたデータ構造の詳細については、表 4 を参照してください) に応じ、イメージ センサーは、要求された解像度とフレーム レートでのビデオ データを送信するよう設定され、続いて、適切な MIPI CSI-2 インターフェース パラメーターがセットされます。これでビデオストリーミングが開始します。

ビデオ データ フォーマット: YUY2

UVC 仕様では、ビデオ データ用に色フォーマットの 1 サブセットだけがサポートされます。このため、UVC 仕様に準拠した色フォーマットでイメージをストリーミングするイメージ センサーを選択する必要があります。本アプリケーション ノートでは、ほとんど (すべてではない) のイメージ センサーによってサポートされた、YUY2 という非圧縮の色フォーマットについて説明します。YUY2 色フォーマットは YUV 色フォーマットをダウンサンプルした 4:2:2 バージョンです。輝度値 Y はピクセルごとにサンプリングされますが、色度 U と V は偶数ピクセルごとにのみサンプリングされます。これは「マクロ ピクセル」を形成します。各マクロ ピクセルは、合計 4 バイトを使う 2 イメージ ピクセルを表します。他のすべてのバイトは Y 値であり、U と V 値は偶数ピクセルのみを表します。

Y0、U0、Y1、V0 (最初の 2 ピクセル)

Y2、U2、Y3、V2 (次の 2 ピクセル)

Y4、U4、Y5、V4 (次の 2 ピクセル)

色フォーマットの詳細情報についてはウィキペディアを参照してください。

注: UVC 仕様において RGB フォーマットはサポートされません。しかし、マイクロソフトは Windows 向けに RGB888 や RGB565 を含む様々な種類の他のフォーマットをサポートするために UVC 仕様の拡張版を提供しています。それらはビデオ ストリーミング形式のディスクリプタに適切な GUID を含めるよう指定されます。詳細については、Microsoft の Web ページのメディア タイプを参照してください。

注: モノクロ画像が UVC の仕様の一部としてサポートされませんが、8 ビット RAW データを Y 値に使用し、すべての U と V 値を 0x80 にセットすることで、8 ビットのモノクロ画像を YUY2 フォーマットで表示できます。これはイメージ センサー (ISP) によって処理されます。

UVC ビデオ データ ヘッダ

UVC クラスは非圧縮ビデオ ペイロード用に 12 バイト ヘッダを必要とします。ヘッダは転送されるイメージ データの特性を記述します。例えば、イメージ センサー コントローラー (CX3) がフレームごとにトグルする「新フレーム」ビットを含みます。CX3 コードはヘッダ内でエラー ビットをセットして、現行フレームのストリーミングに問題が発生したことも示せます。この UVC データ ヘッダはすべての USB 転送に必要です。詳細情報については UVC 仕様を参照してください。

表 5 に UVC ビデオ データ ヘッダのフォーマットを示します。

表 5. UVC ビデオ データ ヘッダ フォーマット

バイト オフセット	フィールド名	説明
0	HLF	HLF (ヘッダ長フィールド) は、ヘッダの長さをバイト単位で示す
1	BFH	BFH (ビット フィールド ヘッダ) は、イメージ データのタイプ、ビデオ ストリームの状態、他のフィールドの有無を示す
2～5	PTS	PTS (プレゼンテーション タイム スタンプ) は、ソース クロック時間をネイティブなデバイス クロック単位で示す
6～11	SCR	SCR (ソース クロック リファレンス) は、システム時刻および USB のフレームの開始 (SOF) トークン カウンターを示す

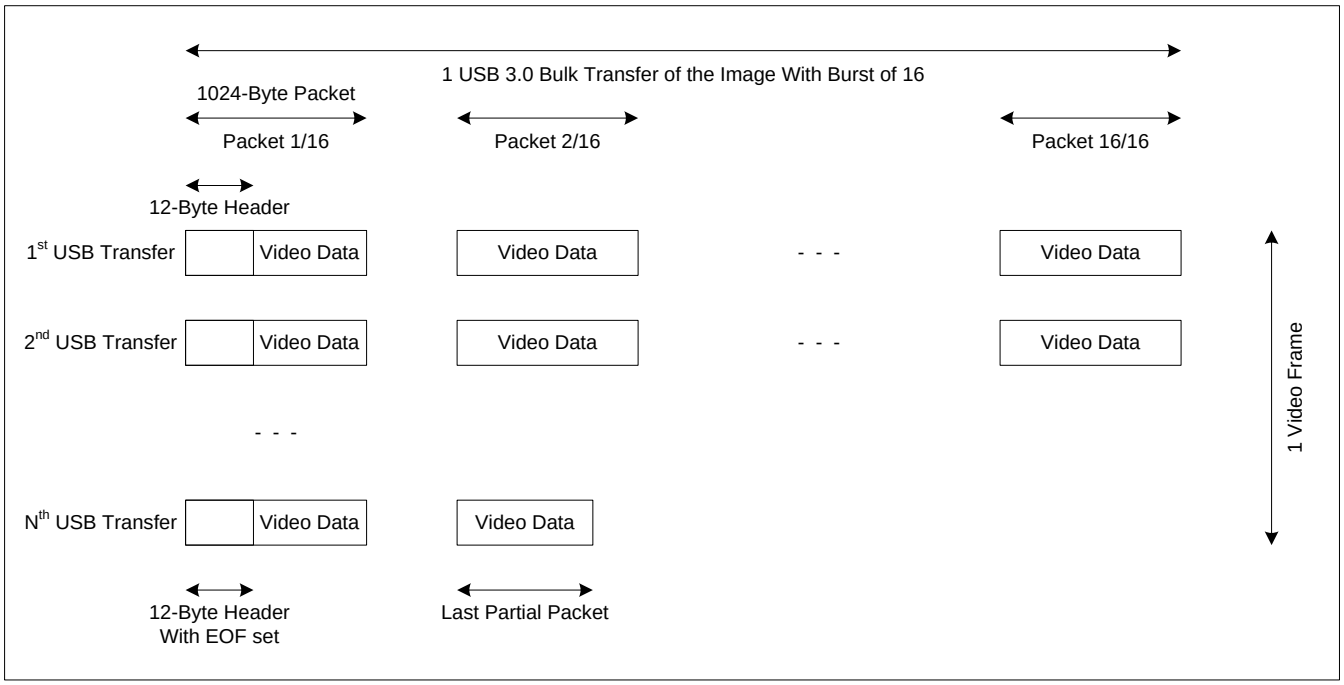
HLF の値は常に 12 です。PTS と SCR フィールドは任意です。このファームウェアの例ではこれらフィールドは 0 になります。ビット フィールド ヘッダ (BFH) はフレームの最後で変化する値を保持します。表 6 に UVC ビデオ データ ヘッダの一部である BFH のフォーマットを示します。

表 6. ビット フィールド ヘッダ (BFH) のフォーマット

ビットオフセット	フィールド名	説明
0	FID	FID (フレーム識別子) ビットは各イメージ フレーム開始境界でトグルし、イメージ フレームの残りの期間で変わらない
1	EOF	EOF (フレームの終了) ビットはビデオの終了を示し、イメージ フレームに属す最後の USB 転送でのみセット
2	PTS	PTS (プレゼンテーション タイム スタンプ) ビットは UVC ビデオ データ ヘッダ内の PTS フィールドの有無を示す (1=存在している)
3	SCR	SCR (ソース クロック リファレンス) ビットは UVC ビデオ データ ヘッダ内の SCR フィールドの有無を示す (1=存在している)
4	RES	予約済み、0 にクリア
5	STI	STI (スティル イメージ) ビットはビデオ サンプルが静止画像であるかを示す
6	ERR	ERR (エラー) ビットはデバイス ストリーミングのエラーを示す
7	EOH	EOH (ヘッダの終了) ビットは、セットされた場合、BFH フィールドの終了を示す

図 18 に、このアプリケーションでヘッダがどのようにビデオ データに追加されるかを示します。12 バイト ヘッダは各 USB バルク転送に追加されます。ここでは、各 USB 転送は合計 16 バルク パケットを持ちます。第 1 世代の USB 3.1 バルク パケット サイズは 1024 バイトです。

図 18. UVC ビデオ データ転送



5 CX3 ファームウェア

本アプリケーション ノートは FX3/CX3 SDK に添付されるサンプル プロジェクト *cycx3_uvc_ov5640* を使用します。ファームウェアは、まず CX3 の CPU を起動してその I/O を設定します。その後、ThreadX RTOS を開始するために CyU3PKernelEntry 関数を呼び出します。RTOS は自ら初期化し、いくつかの内部スレッドを作成した後、CyFxApplicationDefine を呼び出してユーザーにアプリケーション固有のスレッドを作成させます。

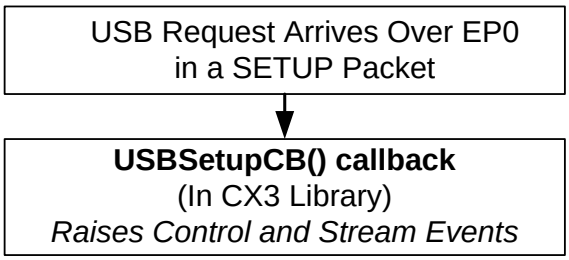
この例では 2 つのアプリケーション スレッドが作成されます。uvcAppThread と uvcMipiErrorThread です。RTOS は、これらのアプリケーション スレッドを実行するためにリソースを割り当て、2 つのアプリケーション スレッド関数 (CyCx3UvcAppThread_Entry と CyCx3UvcMipiErrorThread) のそれぞれの実行をスケジュールします。図 19 に基本プログラム構造を示します。

本アプリケーション ノートで記述するサンプル ファームウェアには、OV5640 イメージ センサーを使用した基本的な画像ストリーミング機能だけが実装されます。別のセンサーを使用する場合、または OV5640 よりも多くの機能を使用する場合は、それに対応するよう *cycx3_uvc.c* ファイルを修正する必要があります。表 7 は各モジュールに実装するコード モジュールと関数をまとめたものです。

表 7. サンプル プロジェクトのファイル

ファイル	説明
cyfxtx.c	変更は不要。このファイルをそのままプロジェクトに使用 これは、RTOS と CX3 API ライブラリがメモリ マッピングに使用する定数および CX3 API ライブラリがメモリ管理に使用する関数を含む
cycx3_uvc.c	UVC アプリケーションの主なソース ファイル。輝度や PTZ などの制御をサポートするためにコードを修正する際、および異なるビデオ ストリーミング モードへのサポートを追加するために修正する際には変更が必要 以下の関数を含む: • *main :*CX3 デバイスを初期化し、キャッシュをセットアップし、CX3 I/O を設定し、RTOS カーネルを起動 • *CyFxApplicationDefine:* RTOS が実行する 2 つのアプリケーション スレッドを定義 • *CyCx3AppThread_Entry:* この関数は最初のアプリケーション スレッドにより実行される。CX3 の内部ブロック用の初期化関数を呼び出し、デバイスをエニュメレートして、デバイスの一時停止とフレームの完了タスクを処理 • *CyCx3AppMipiErrorThread:* この関数は 2 番目のアプリケーション スレッドにより実行される。MIPI CSI-2 コントローラー内のエラーを示すイベントを待った後、CyU3PMipicsiGetErrors でそれを読み出す • *CyCx3AppDebugInit:* デバッグ メッセージのプリント用の CX3 UART ブロックを初期化 • *CyCx3AppInit:* CX3 の GPIO ブロック、プロセッサ ブロックまたは PIB (GPIF II は PIB の一部)、I2C/CCI インターフェース、MIPI CSI-2 Rx インターフェースおよびセンサー (スーパースピード モードでは 1080p 30fps に設定) を初期化。また、エニュメレートのために USB ブロックを、USB 転送のためにエンドポイント コンフィギュレーション メモリを初期化するほか、2 つの GPIF II ソケットから USB ソケットにデータを転送するために DMA チャネル コンフィギュレーションを作成 • *CyCx3AppGpifCB:* GPIF II ステート マシンから生成される CPU 割込みを処理 • *CyCx3AppDmaCallback:* CX3 からホストへの出力ビデオ データを継続的にトラック。イメージ データにヘッダを追加するほか、最初のアプリケーション スレッドに通知してフレームの完了を示せる • *CyCx3AppUSBSetupCB:* ホストが送信するすべての制御要求を処理し、UVC 固有の要求がホストから受け取られたことを示すイベントを設定し、ストリーミングが停止する時点を検出。ビデオ ストリーミングの開始と停止に必要となる、UVC クラス固有の PROBE と COMMIT 制御要求も処理 • *CyCx3AppUSBEventCB:* 一時停止、ケーブル切断、リセット、再開などの USB イベントを処理 • *CyCx3AppErrorHandler:* エラー処理関数。必要に応じてエラー処理を実施するために使用されるプレースホルダ関数 • *CyCx3AppAddHeader:* 有効なストリーミング中に UVC ヘッダをビデオ データに追加
cycx3_uvcdscr.c	UVC アプリケーションの USB エニュメレーション ディスクリプタを含む。フレームレート、画像解像度、ビット深度、またはサポートされているビデオ制御を変更する場合は、このファイルも変更が必要。UVC 仕様に必要な詳細情報が含まれている
cycx3_uvc.h	フレーム カウントと MIPI エラー スレッドのデバッグ プリントをオン／オフにすることでアプリケーション動作を修正するスイッチを含む。 *cycx3_uvc.c* と *cycx3_uvcdscr.c* ファイルに共通に使用される定数を含む
cyfx_gcc_startup.s	CX3 CPU の起動コードを含むアセンブリ ソース ファイル。スタックと割込みベクタをセットアップする関数を含む。 変更は不要。

図 19. 高レベルカメラ プロジェクトの構造



uvc.c

CyCx3UvcAppInUSBSetupCB()
Handles Control & Stream events
CyCx3GpifCB()
Handles Frame Termination
CyCx3UvcAppDmaCallback()
Handles DMA Transfers

5.1 アプリケーション スレッド

2 つのアプリケーション スレッドが並列機能を有効にします。

uvcAppThread (アプリケーション スレッド) は CX3 ペリフェラルの初期化を行い、2 つの内部イベントを処理します: USB サスペンドと DMA チャネルのリセット。

uvcMipiErrorThread は MIPI CSI-2 のエラーを監視し、エラー カウントを返します。

5.1.1 USB サスペンド イベント

CX3 がホストにより一時停止状態にされた時、USB サスペンド イベント (CX3_USB_SUSP_EVENT_FLAG) がトリガーされ下記を実行します。

1. MIPI CSI-2 コントローラーのクロックを無効にし、インターフェースを低消費電力モードにします。
2. イメージ センサーをスリープ状態にします。
3. USB バスの動作を復帰ソースにして CX3 コアを低消費電力サスペンド モードに移行します。

USB バスで動作が検出された場合、デバイスは復帰します。復帰後、イメージ センサーは電源投入され、MIPI CSI-2 コントローラーは再びアクティブになります。

5.1.2 DMA チャネル リセット イベント

フレーム転送に時間が掛かりすぎて完了できない場合、古いデータが消されビデオ ストリームが再起動されます。500ms タイマー (ウォッチドッグとして使われる) が切れるとセットされた CX3_DMA_RESET_EVENT フラグを使ってこれが行われます。

このタイマーは、フル フレームがホストに転送される度にリセットされます。また、1 フレームが DMA バッファ に長時間滞留した時にのみ切れます。ホストへの DMA バッファの転送に失敗したら、ビデオ ストリームも再起動されます (DMA チャネルがシーケンスから外れてしまう結果となる可能性があります)。

CX3_ERROR_THREAD_ENABLE プリプロセッサ マクロが有効になったら、CSI-2 エラーを監視するために他のスレッド (uvcMipiErrorThread) が作られます。

このスレッドのエントリ関数 (CyCx3UvcMipiErrorThread) は MIPI CSI-2 コントローラーからエラー件数を取得するように周期的に CyU3PMipicsiGetErrors を呼び出します。空白スクリーンが見えたら、エラー カウンターはデバッグ処理のため使用されます (MIPI CSI-2 エラー カウンターで修正する方法の詳細な情報は、8 節を参照してください)。

5.2 UVC 要求の処理

CX3 ファームウェアはコントロール エンドポイント (EP0) を介して UVC 固有の制御要求 (4.3.2 節に記述された SET_CUR、GET_CUR、GET_MIN、GET_MAX) を処理します。クラス固有の制御要求は CyCx3AppUSBSetupCB (CB=コールバック) 関数で処理されます。CX3 がこれらいずれかの制御要求を受け取る度に、この関数が受け取ったセットアップ データをデコードして処理します。

ビデオ ストリーミング インターフェースに直接送信したプローブ制御要求 (4.3.2.2 節を参照) の場合、ホストに該当するプローブ制御構造体を送信します。コミット制御要求が生成されたら、ビデオ ストリーミングが起動されます。

この例ではビデオ コントロール機能に対応してないため、ビデオ コントロール インターフェースに送信されたすべての要求 (Get Status 要求以外) (4.3.2.1 節を参照) が機能停止します (USB ホストが STALL ハンドシェークを受け取ります)。サンプルが輝度、露出、PTZ もしくはいずれか他の制御に対応するなら、ここで処理する必要があります。

5.3 初期化

CyCx3UvcAppThread_Entry 関数は、UART デバッグ処理機能を初期化するために CyCx3AppDebugInit を呼び出し、残りの必要なブロック、DMA チャネル、USB エンドポイントを初期化するために CyCx3AppInit を呼び出します。

5.4 エニュメレーション

CyCx3AppInit 関数では、CyU3PUsbSetDesc 関数への呼び出しは CX3 を UVC デバイスとしてエニュメレートします。UVC ディスクリプタは *cycx3_uvcdscr.c* ファイルで定義されます。これらのディスクリプタは、非圧縮 YUY2 フォーマットを使用して、ピクセルあたり 16 ビットを送信する、30fps の 1920x1080 ピクセルと 60fps の 1280x720 ピクセルの 2 つの解像度のイメージ センサー用に定義されます。これらの設定を変更する必要がある場合は 4.3.1 節を参照してください。

5.5 MIPI CSI-2 コントローラーの設定

CyCx3AppInit 関数は最初に MIPI CSI-2 コントローラーにインターフェースする I^2C、GPIO および PIB ブロックを初期化します。

DMA チャネルが作成された後、GPIF II ステート マシンをロードするために、データ バス幅とバッファ サイズをパラメーターとして取り入れる CyU3PMipicsiGpifLoad を呼び出します。プロデューサ ソケットから見て、パラメーターとして渡されるバッファ サイズは DMA バッファ サイズとなります。つまりロードされる値は以下となります。

dma_buffer_size –(dma_header_size + dma_footer_size)

選定されたバス幅は、CSI-2 を介して送信されているイメージ データのタイプと一致するべきです。各々の対応するイメージ データのタイプに応じて必要とされるバス幅は SDK API ガイドで定義されます。

それで、ステート マシンが起動され、USB ホストによってイメージ ストリーミングが要求されるまで休止状態を続けます。

ステート マシンはロードされ起動された後、MIPI CSI-2 コントローラーは初期化されます。これは、XRES ピンをアサートしてイメージ センサーをリセット状態にする CyU3PMipicsiInit を呼び出すことによって行われます。センサーをリセット状態から復帰させるには、CyU3PMipicsiSetSensorControl 関数を使用して XRES ピンをディアサートします。

MIPI CSI-2 コントローラーが (CyU3PMipicsiSetIntfParams 関数により) 設定されます。この関数はコンフィギュレーション パラメーターを含む CyU3PMipicsiCfg_t 型の構造体を取り込みます。

サイプレスは CX3 プロジェクトを簡単に生成できるように、**CX3 コンフィギュレーション ツール** (そのスクリーンショットを図 20～23 に示す) を提供します。このツールは、ユーザー入力に基づいて、完全な CX3 ファームウェア プロジェクト (UVC／ベンダー クラス ディスクリプタ、UVC 要求の処理、ビデオ ストリーミング DMA エンジン、および MIPI CSI-2 コントローラーのコンフィギュレーションを含む) を生成できます。ツールにアクセスして操作させる手順は「CX3 Application Software / USB Driver: Frequently Asked Questions – KBA91298」、とサイプレス EZ-USB Suite (Eclipse) の CX3 コンフィギュレーション ツール ヘルプ (*Help* > *Help Contents* > *Cypress EZ-USB Guides* > *EZ-USB Suite Guide* > *CX3 Configuration Utility*) で記述されます。

図 20. CX3 コンフィギュレーション ツールのショートカット

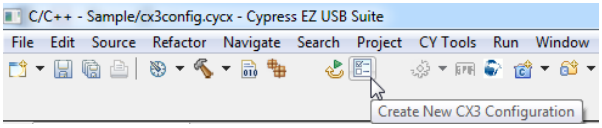


図 21. CX3 コンフィギュレーション ツールのスタート ページ

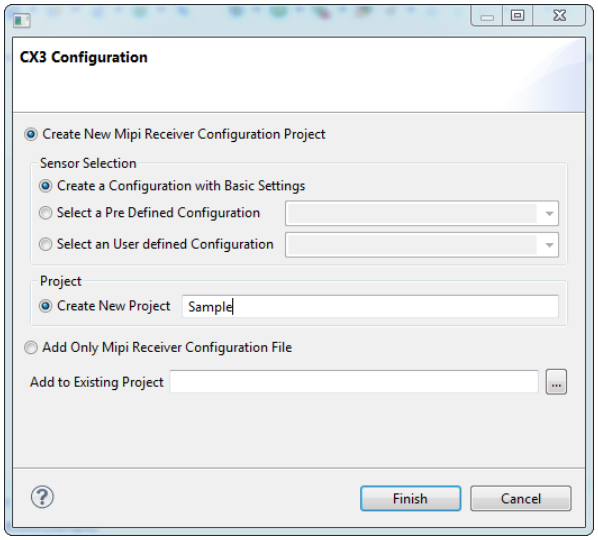


図 22. CX3 コンフィギュレーション ツールのセンサー コンフィギュレーション ページ

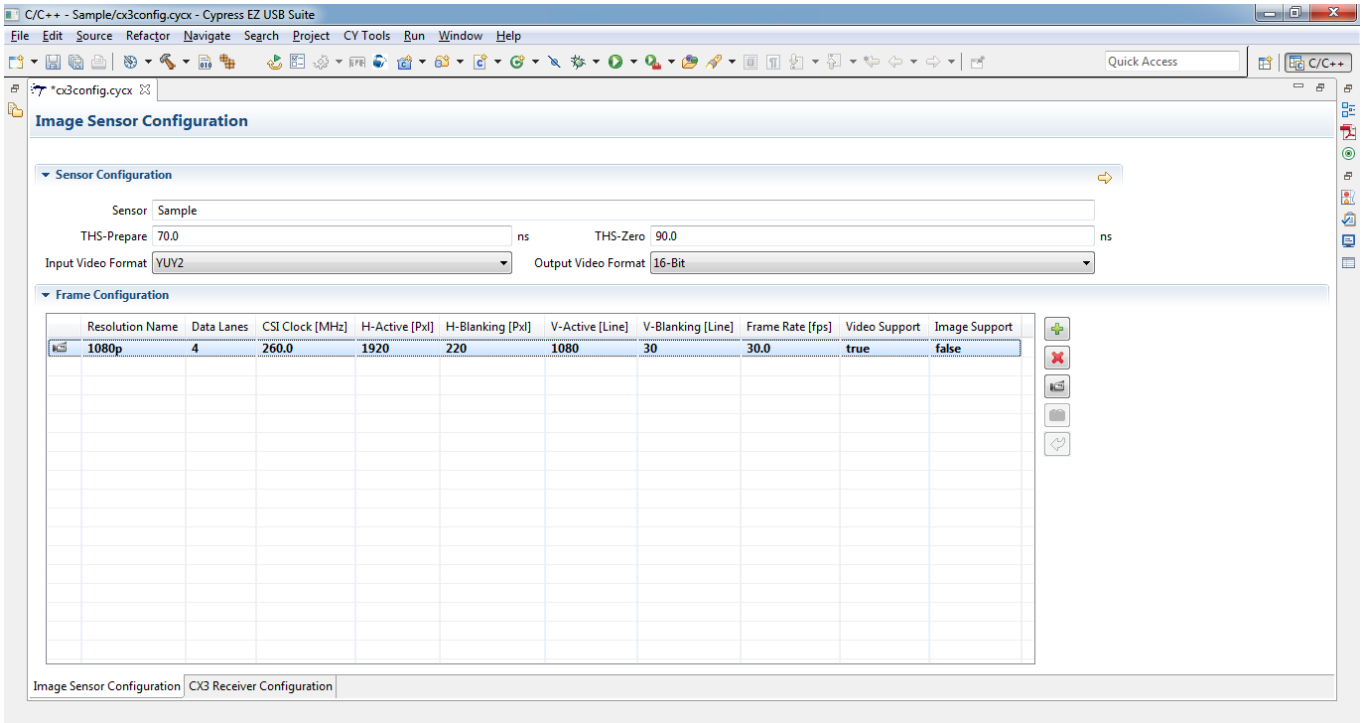
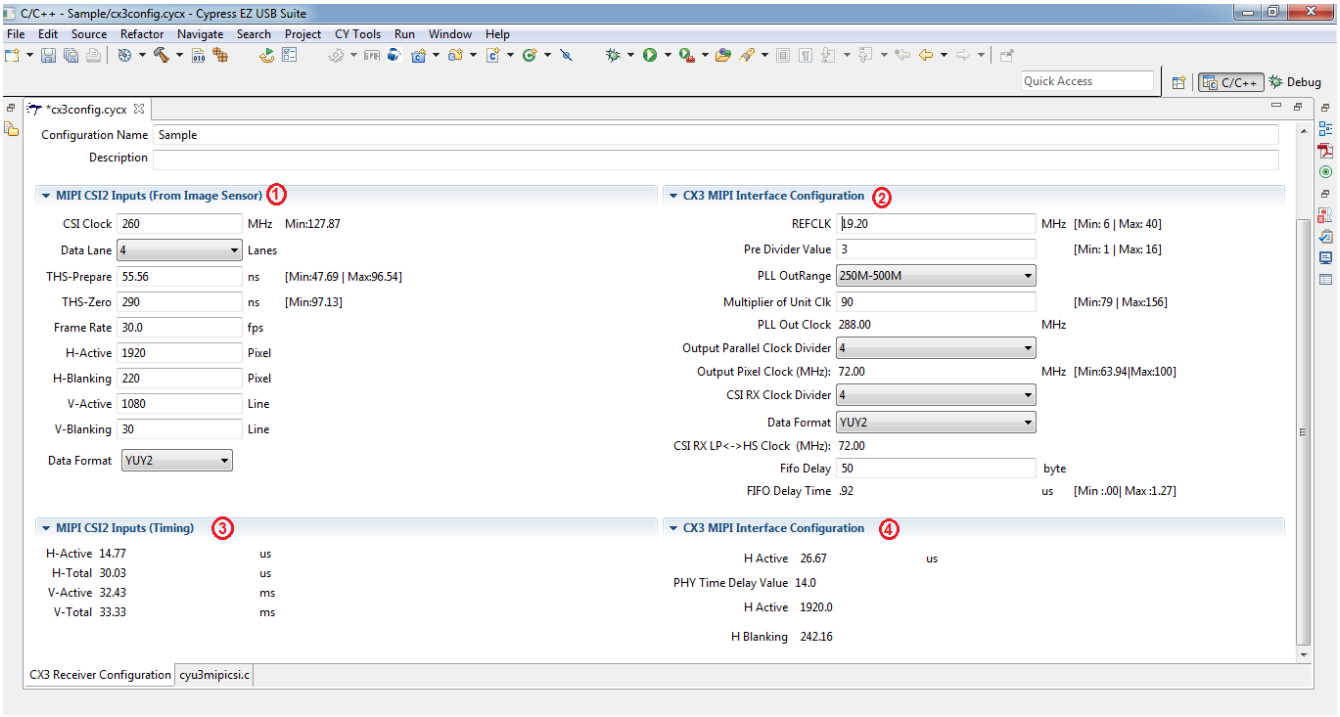


図 23. CX3 MIPI レシーバー コンフィギュレーション ツール



5.5.1 CX3 MIPI レシーバー コンフィギュレーション

CX3 コンフィギュレーション ツールは完全な CX3 プロジェクトまたは MIPI レシーバー コンフィギュレーション (既存のプロジェクトに追加できるもの) を生成するために使用できます。MIPI レシーバー コンフィギュレーション タブ (図 23) でのフィールドの概要をここで説明します。

MIPI CSI2 Inputs (MIPI CSI2 入力): 図 23 でのマーク「1」を参照してください。これらのフィールドは CSI-2 入力パラメーターです。例えば、図 23 に CSI-2 センサー インターフェース ストリーミング YUY2 フル HD (1080p) (CSI クロックが 260MHz で、H-Blanking が 220 ピクセルで、V-Blanking が 30 ラインである) を示します。これらの入力に基づく計算が MIPI コントローラー コンフィギュレーション パラメーターを検証するために使用されるため、これらの値は実際の CSI-2 インターフェース パラメーターに一致する必要があります。

ここでのすべてのパラメーター (データ フォーマットを含む) は計算の目的のみに使用されます。ここでのデータ フォーマットはストリーミングされているフォーマットのビット／ピクセルを得るために使用されます。

現在、ツールは「THS-Prepare」と「THS-Zero」の値フィールドを使用しません。SDK は「THS-Prepare」と「THS-Zero」の値を変更するための API 「CyU3PMipicsiSetPhyTimeDelay」をサポートします。この API の詳細については、FX3 SDK API のガイドを参照してください。ほとんどのセンサー／ISP は 9 の PHY 時間遅延 (上記の API の第 2 の 引数) で動作します。

MIPI CSI2 Input Timings (MIPI CSI2 入力 タイミング): 図 23 にマークされた「3」を参照してください。これらは CSI-2 入力パラメーターに基づいて計算された時間です。

CX3 MIPI Interface Configuration (CX3 MIPI インターフェース コンフィギュレーション): 図 23 でのマーク「2」を参照してください。ここでのパラメーターは MIPI CSI-2 コントローラーで使用される 3 つのクロック (PLL クロック、CSI RX LP <-> HS クロック、パラレル出力ピクセル クロック) を設定するために使用されます。これらのクロックの詳細については、CX3 TRM の 1.7 節を参照してください。

出力ピクセル クロック (PCLK)、データ フォーマット、Fifo 遅延は CX3 での MIPI コントローラーが正常に機能することを決定する重要なパラメーターです。

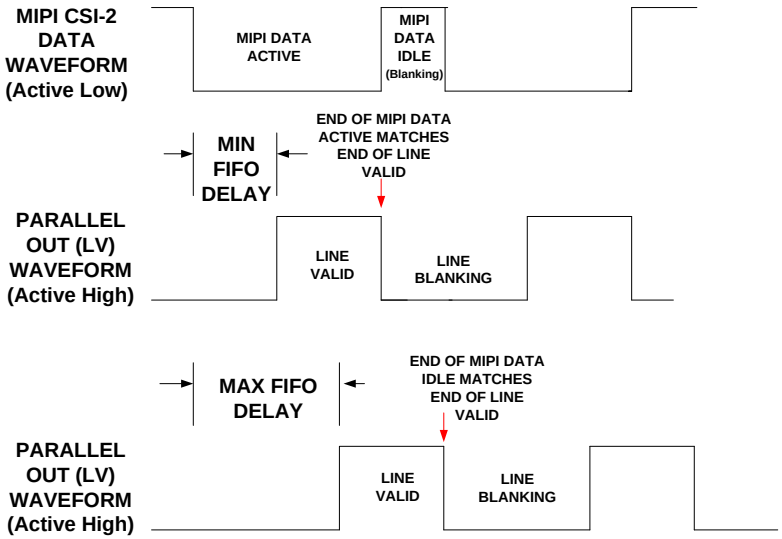
出力ピクセル クロックはツールに指定された範囲内に設定する必要があります。

データ フォーマットはパラレル出力インターフェースのデータ バス幅を決定します。サポートされたデータ フォーマット リストと各フォーマットによりサポートされたパラレル出力データ バス幅については、CX3 TRM の 1.6 節を参照してください。

Fifo 遅延パラメーターは設定した後、各ラインの始まりにパラレル出力に遅延を追加します。設定された出力パラレル ピクセル クロックはツールに提案されている出力パラレル ピクセル PCLK の「最小」値より高い場合、Fifo 遅延が必要になります。図 23 に Fifo 遅延の概念を示します。しかし、MIPI H-Active 時間と出力パラレル H-Active 時間の簡単な減算で fifo 遅延の最大値と最小値を計算できないことに注意してください。MIPI コントローラーには 1 つのライン バッファがあります。Fifo 遅延期間を計算する時、これも利用する必要があります。ツールはこの調整を実行します。

注: 次の MIPI ライン データが到着する前に、現在の MIPI ライン データはパラレル インターフェースで完全に送信する必要があります。このルールに違反すると、ビデオ フレームに垂直分割が発生します。したがって、Fifo 遅延は慎重に選択する必要があります。

図 23. Fifo 遅延の概念



他のセンサーを構成して他のフォーマットをストリーミングするには、FX3 SDK で新しいプロジェクトを作成し、センサー構成ページ (図 22 を参照) が表示されたら入力ビデオフォーマットを選択します。RGB 形式は、ピクセルのバイト数に応じて、YUV 形式と同じ方法でストリーミングできます。RGB 形式のストリーミングの詳細については、FX3 SDK の他のサンプルプロジェクトを参照してください。MJPEG 形式をさまざまな解像度でストリーミングし、CX3 デバイスを ISP (イメージシグナルプロセッサ) とインターフェースする方法については、CX3 ベースの Ascella RDK のソースコードを参照してください。CX3 ベースの Ascella RDK のファームウェアソースを入手するには、Ascella - Cypress® CX3™THine® ISP 13MP リファレンスデザイン キット (RDK) を購入する必要があることに注意してください。キットについて、ステップバイステップで説明されたガイドラインはここを参照してください。

OV5640 センサーを使用した MJPEG ファームウェアサンプルを探している場合は、OmniVision で NDA に署名し、テクニカルサポートケースを作成する必要があります。

5.6 イメージ センサーの設定

MIPI CSI-2 インターフェースが設定された後、イメージ センサーを適切に設定する必要があります。これは CyCx3AppInit 関数の終わりの近くで行われます。サイプレスは OV5640 イメージ センサーを初期化、設定する関数を持つあらかじめ編集されたライブラリを備えます。

他のイメージ センサーの場合、イメージ センサーを設定するのに使える *cyu3imagesensor.c* ファイルがいくつかのヘルパー関数を持ちます。

イメージ センサーは CX3 の I^2C マスター ブロックを使って設定されます。*cyu3imagesensor.c* ファイル内の SensorWrite2B、SensorWrite、SensorRead2B、および SensorRead 関数は、I^2C を介してイメージ センサーのコンフィギュレーションを読み書きするために使用できます。

データをイメージ センサーへ書き込むために関数 SensorWrite2B と SensorWrite は標準 API CyU3PI2cTransmitBytes を呼び出します。データをイメージ センサーから呼び出すために関数 SensorRead2B と SensorRead は標準 API CyU3PI2cReceiveBytes を呼び出します。これらの API の詳細は、FX3 SDK API ガイドを参照してください。

5.7 ビデオ ストリーミングの開始

VLC Player、e-CAMView、Media Player Classic、または VirtualDub などの USB ホスト アプリケーションは、USB インターフェースと USB alternate setting (代替設定) の組合せを、ビデオをストリーミングするもの (通常 Interface 0 Alternate setting 1 と呼ばれる) に設定し、PROBE/COMMIT 制御要求を送信する事を目的として UVC ドライバーの最上部に配置されます。これはビデオ データのストリーミングをまもなく開始するようホストによる指示です。ストリーミング イベントにおいて、USB ホスト アプリケーションは CX3 に画像データの要求を開始します。CX3 はその後イメージ センサーから第 1 世代の USB 3.1 ホストへ画像データの送信を開始するものとします。

CX3 がストリーミング イベントを受信する場合、USB イベントのコールバック (CyCx3cAppUSBSetupCB) は、受信したフレーム インデックスに従ったイメージ センサーと MIPI CSI-2 インターフェースを設定します。コンフィギュレーション パラメーターはホストが要求した解像度とフレーム レートに依存します。これは次の節 (5.8 節) で説明します。

GPIF II ステート マシンはホストがイメージ データを要求するまで動作する必要がないため、一時停止します。ストリーミングイベントが届くと、CyCx3UvcStart 関数が呼び出されます。これは、ステート マシン (CyU3PGpifSMControl を通じて) を再開し、MIPI CSI-2 インターフェース (CyU3PMipicsiWakeup を通じて) を復帰させ、イメージ センサーに電源を入れます。

注: SDK API ガイドは前後の節で使用される MIPI CSI-2 と GPIF II に関連した関数の詳細な情報を含みます。

5.8 フレーム設定の選択と切り替え

カメラは複数の解像度とフレーム レートに対応できます。対応される解像度／フレーム レート設定は、4.3.1.2 節で説明されるようにそのビデオ ストリーミング フレーム ディスクリプタに格納されます。

解像度かフレーム レートの設定が USB ホスト アプリケーションで選ばれると、USB ホストは SET_CUR コミット制御要求をビデオ ストリーミング インターフェースへ送信します。受信した構造体の 4 バイト目は、選択されるフレーム ディスクリプタを示します。

提供されたサンプル プロジェクトでは、デバイスがスーパースピードで動作する場合、インデックス 0 は 720p@60fps のビデオ ストリーミングに対応し、インデックス 1 は 1080p@30fps のビデオ ストリーミングに対応します。USB ホストは 1080p のビデオを流すために、フレーム インデックスが「1」に設定された SET_CUR 要求を送信します。

CyCx3UvcAppUSBSetupCB 関数はこのインデックスを読み出して、受信した内容に応じてイメージ センサーと MIPI CSI-2 コントローラーを設定して、要求された設定のビデオを流します。

5.9 DMA バッファの設定

UVC の仕様では、各 USB 転送 (このアプリケーションでは、各 16KB の DMA バッファ) に、12 バイトのヘッダを追加する必要があります。しかし、CX3 アーキテクチャでは、DMA ディスクリプタに関連付けられている DMA バッファが 16 バイトの倍数の大きさを備えている必要があります。

DMA バッファ内の 12 バイトを CX3 CPU が書き込むものとして予約すると、DMA バッファの境界が 16 バイトの倍数にならない場合があります。このため、DMA のバッファ サイズは、16,384 から 12 を引くのではなく 16 を引いた値にする必要があります。CX3 ファームウェアが追加する 12 バイトのヘッダを含まない DMA のバッファ サイズは、16,384-16=16,368 バイトです。DMA バッファは、先頭に 12 バイト ヘッダ、次に 16,368 ビデオ バイト、そして DMA バッファの最後に 4 つの未使用バイトの順で構成されます。こうして 16×1024 バイトのパケット (16,384 バイト) である USB BULK の最大バースト サイズを利用できる DMA バッファを生成します。

5.10 ビデオ ストリーミング中の DMA バッファの処理

CyCx3AppInit 関数は、プロデューサとコンシューマのイベント用のコールバック通知付きの手動 DMA チャネルを作成します。

コンシューマの通知はホストが読み出したデータ量を追跡するために使用されます。ホストがフレーム全体を読み出した後で、追跡変数がリセットされ、GPIF II ステート マシンが再起動されます。プロデューサ イベント通知が 12 バイトのヘッダを付加し、ホストにデータを転送するために使用されます。

DMA コールバックでは、ファームウェアは CyU3PDmaMultiChannelGetBuffer を介して DMA バッファが生成されたかを確認します。GPIF II プロデューサの DMA バッファが転送されるか、または強制的に CX3 CPU によってラップアップされる場合、DMA バッファは CX3 CPU に使用可能になります。アクティブ フレーム期間中に、イメージ センサーは、データをストリームし、GPIF II は、一杯の DMA バッファを生成します。この時点では、CX3 CPU は USB に 16,380 バイトのデータを転送します。

フレームの最後で、通常、最後の DMA バッファが部分的に満たされます。この場合、プロデューサ イベントをトリガーするためにファームウェアは強制的にプロデューサ側の DMA バッファをラップアップする必要があり、その後適切なバイト数を DMA バッファから USB に転送します。CyU3PDmaMultiChannelSetWrapUp の呼び出しを使う DMA バッファ (GPIF II で生成された) の強制的ラップアップは GPIF II コールバック関数 (CyCx3GpifCB) の中で実行されます。1 フレームが終了する時に発生する CPU の割込みを GPIF II が設定すると、このコールバック関数がトリガーされます。

注: UVC ヘッダは、フレーム識別子およびフレーム終了マーカーに関する情報を格納します。フレーム送信の終わりで、ファームウェアは 2 番目の UVC ヘッダ バイト (CyCx3UvcAddHeader を参照) のビット 1 をセットし、ビット 0 をトグルします。さらに「hitFV」変数がセットされ、イメージ センサーから取り込まれるフレームが終了したことを示します。

glDmaDone 変数は DMA バッファを追跡して、すべてのデータが CX3 FIFO から取り出されたことを確保します。

5.11 フレーム終端でのストリーミングの再開

フレームの最後で、GPIF II ステート マシンは前述した一連のイベントを開始する CPU 割込みを生成します。フレームの最後の DMA バッファは USB ホストにより読み出されると、glDmaDone 変数がゼロになり、以下のような作業が行われます:

- 古いイメージ データを消去するウォッチドッグとして動作する 500ms DMA チャネル リセット タイマーをリセットします。
- START 状態の 1 つで GPIF II ステート マシンを再起動するために、CyU3PGpifSMSwitch 関数を呼び出します。フレームの最後のバッファがソケット 0 に読み出された場合、GPIF II ステート マシンは ALPHA_CX3_START_SCK1 で再起動され、ソケット 1 へ次のフレームを読み出し開始します。またその逆もあります。

また UVC 仕様はヘッダ内のフレーム ID ビットを必要としフレームごとに切り替えられます。これは最後のバッファが USB ホストに転送される直前に、CyCx3UvcAddHeader 関数の中で行われます。

5.12 ビデオ ストリーミングの終了

イメージ ストリーミングを終了する方法は 3 つあります。

- カメラをホストから切り離す
- USB ホスト プログラムを終了する
- USB ホストがリセットを発行するか CX3 への要求を一時停止する

FIFO に残留データがある時にもビデオ ストリーミングを終了できるため、適正なクリーンアップが必要となります。

ビデオ ストリーミングを終了する時、ファームウェアはストリーミング関連の変数をリセットし、DMA チャネルをリセットし、そして GPIF II ステート マシンを一時停止します。カメラと MIPI CSI-2 インターフェースの電源も切断します。これらは CyCx3AppStop 関数の中で行います。

USB ホストのアプリケーションが終了する場合、ファームウェアは Windows プラットフォーム上で関数のクリア要求を出したり、Mac プラットフォーム上で代替設定=0 のインターフェースの設定要求を出します。この要求を受信するとストリーミングが停止します。この要求は、ターゲットが CY_U3P_USB_TARGET_ENDPT で、かつ要求が CY_U3P_USB_SC_CLEAR_FEATURE の場合、CyCx3AppUSBSetupCB 関数の中で処理されます。

ビデオ ストリーミングを終了した後で、イメージ センサーは電源切断され、CX3 コアは CyU3PSysEnterSuspendMode を使って一時停止されます。USB バス上で動作がある時に、CX3 コアが復帰しイメージ センサーに電源を投入してビデオ ストリーミングを再開します。

CCI (I^2C) コマンドか CX3 の XSHUTDOWN 端子のいずれかを使うことによって、イメージ センサーは電源を切断できます。このピンは最初のパラメーターとして、CY_U3P_CSI_IO_XSHUTDOWN を持った CyU3PMipicsiSetSensorControl 関数を使って制御できます。

6 ハードウェアのセットアップ

6.1 CX3 RDK を使うテスト

現時点のプロジェクトは、OmniVision OV5640 イメージ センサーを内蔵する CX3 リファレンスのデザイン キット (RDK) を備えた装置でテストされます。キットに関する詳細はサイプレス ウェブサイトの www.cypress.com/cx3 にある **Kits** タブから入手できます。

6.1.1 キットの調達

1. e-Con Systems 社から http://www.e-consystems.com/CX3-Reference-Design-Kit.asp にて RDK を購入してください。
2. サイプレスは OV5640 センサーの基本的な操作を実行できる CX3 SDK を備えたライブラリを提供します。追加機能をサポートする必要がある場合には、OmniVision (迅速な処置のために usb3@cypress.com にお客様の要求を電子メールでご送信ください) と NDA を締結します。NDA の締結後 usb3@cypress.com へ送信してください。サイプレスは NDA の確認後、OmniVision 固有のソース ファイルを提供します。
3. SuperSpeed 性能を評価するために、第 1 世代の USB 3.1 ホスト対応のコンピュータを使用します。

CX3 RDK にはクイック スタート ガイド、ハードウェア ユーザー マニュアルとファームウェア ビルド マニュアルが付属しており、お客様が RDK の使用を開始する手助けをします。

6.2 独自基板の設計

独自に基板を設計する際に、基板の動作を確実にするためにいくつかのガイドラインに従う必要があります。「FX3/FX3S Hardware Design Guidelines」アプリケーション ノートでは、CX3 にも適用できる FX3/FX3S のハードウェア設計例に対応した推奨デザイン プラクティスを説明します。

MIPI CSI-2 信号については、以下に示す配線ガイドラインに従う必要があります。

- 基板上の配線長は、100mm を超えてはいけません。
- 差動信号配線の伝送線路インピーダンス (Z0) は、100Ω +/- 10%にしてください。
- P－N が組となる配線長の差は 0.5mm 未満にしてください。
- MIPI CSI レーン信号の差動配線長の差は 1.5mm 未満にしてください。
- P－N が組となる信号の配線間隔は線幅の 2 倍にします。

7 UVC ベースのホスト アプリケーション

様々なホスト アプリケーションにより、UVC デバイスからビデオを表示して取り込めます。通常、Windows OS で Media Player Classic を選択します。他に 2 つの Windows アプリケーションがあります。VirtualDub (オープンソース アプリケーション) と e-CAMView です。

Linux システムは、ビデオをストリーミングするために V4L2 ドライバーと VLC メディア プレーヤーを使用できます。VLC メディア プレーヤーはインターネットからダウンロードできます。

MAC プラットフォームは、ビデオをストリームするために、UVC デバイスとのインターフェースの作成に FaceTime、iChat、Photo Booth、および Debut Video Capture ソフトウェアを使用できます。

8 トラブルシューティング

デバイスまたはファームウェアで問題に直面した場合、最初のステップはより多くのデータを取得し問題の原因を絞り込むことです。これをさらに実現するには、UART と JTAG デバッグを有効にできます。

cycx3_uvc.h ファイルの「CX3_DEBUG_ENABLED」スイッチを有効にして、様々なカウンター、エラー メッセージ、色々なデバッグ データを印刷できるようにします。UART ケーブルまたは USB-UART 間ブリッジを介して基板 (または CX3 RDK) 上の UART ポートを PC に接続します。ハイパーターミナル、Tera Term、または PC 上の COM ポートにアクセスする別のユーティリティを開きます。転送を始める前に UART コンフィギュレーションを 115,200 ボー、パリティなし、1 ストップ ビット、フロー制御なし、8 ビット データに設定します。この設定はデバッグ プリントを取り込むのに十分です。

また JTAG デバッガは各ステップでファームウェアをデバッグするために使用できます。JTAG 経由のデバッグ処理に関する詳細については、プログラマー マニュアル (Start > All Programs > Cypress > EZ-USB FX3 SDK) の 12.2.2.3 節「Executing and Debugging」を参照してください。

CX3 のファームウェア、ハードウェア、アプリケーション ソフトウェアについての FAQ は KBA91297、KBA91295 と KBA91298 で文書化されました。

いくつかの一般的な問題やこれらの原因、対策を一覧表示します。

問題: デバイスが PC 上でエニュメレートしません。

原因 1: ファームウェア内の API にエラー状態が発生した可能性があります。 これにより、エラー ハンドラの呼び出しが発生しており、ファームウェアが停止するか、またはエニュメレートに失敗する原因となります。

対策 1:JTAG デバッガや UART/RS232 ケーブルを使用し、すべての呼び出しの返り値の状態を検査してその中のどれかに失敗が出たかを確認します。その後、呼び出しが失敗した理由を理解するために API ガイドを使用します。

原因 2: 基板内の USB 信号品質が悪いです。

対策 2: 設計ガイドラインのアプリケーションノートを読み、適切な基板の設計を確認します。

原因 3: 敵切なドライバーがインストールされていません。

対策 3: デバイスをアンインストールして Device Manager で改めてインストールしてください。 あるいは、新しいデバイスとしてドライバーのインストールを強制するために、VID/PID のペアも変更できます。

原因 4:ディスクリプタに誤った値があるかも知れません。

対策 4:エラーの一般的な原因は長さフィールドです。コンフィギュレーション ディスクリプタ内の wTotalLength フィールドが全体の長さに一致していることを確認してください。また、クラス固有のビデオ コントロールとビデオ ストリーミングのディスクリプタについても、同じようにフィールドを確認してください。

問題: デバイスはエニュメレートしましたが、USB ホスト アプリケーション (e-CAMView などで) では黒い画面を表示します。

原因 1: CyCx3UvcAppThread_Entry 関数の glDMATxCount 変数を検査し、それが増加することを確認します。それでなければ CX3 とイメージ センサーとの間のインターフェースに問題がある可能性があります。

対策 1: イメージ センサーが CX3 に正しく接続されていることを確認します。 また、そのセンサーの初期化、MIPI CSI-2 コントローラーのコンフィギュレーションおよび GPIF II バス幅の選定が正しく行われることを確認します。

原因 2:glDMATxCount のプリント値が増加することが見えた場合、出力中のイメージ データを確認する必要があります。まずフレームごとに送出されるデータの合計量を確認してください。この量を知るために、ヘッダでのフレーム終了ビットがセットされたデータ パケットの長さを見つけます。(ヘッダの 2 番目のバイトはフレーム終了の転送用の 0x8E または 0x8F です)。他のデータ パケットの場合、転送されたデータの長さは DMA バッファのサイズをフッタサイズで引いた値になります。

(UVC ヘッダを含まない) フレームで転送された合計のイメージ データは 幅 × 高さ × 2 で計算されます。

対策 2: イメージ サイズの合計が要求されたものよりも少ない (またはそれより多い) 場合に、イメージ センサーは、おそらく必要とされるよりも少ない (またはそれ以上) のデータを送信しています。イメージ センサーと MIPI CSI-2 コントローラーのコンフィギュレーションを確認してください。

問題: ビデオはハイスピード接続でストリーミングできますがスーパースピード接続ではできません。

原因: これは SSTX/SSRX 線の信号品質が悪い可能性があります。スーパースピード配線が AN70707 に記述されたガイドラインに従っていることを確認してください。また、認定済みのケーブルを使用していることを確認します。

さらに、CX3_ERROR_THREAD_ENABLE スイッチは、CSI-2 インターフェースでエラーが発生したかどうかを確認するために使用できる MIPI CSI-2 のエラーを記録できます。エラーの種類に関する詳しい情報は SDK API ガイドを参照してください。

問題がまだ解決されない場合は、サイプレステクニカル サポート ケースを作成してください。

9 まとめ

本アプリケーション ノートでは、MIPI CSI-2 インターフェース付きの USB ビデオ クラスに準拠したイメージ センサーが、どのような方法でサイプレスの EZ-USB CX3 を使用して実装できるかを説明しています。特に、以下のことを示します。

- ホスト アプリケーションおよびドライバーが UVC デバイスとやり取りする方法
- UVC デバイスが UVC 固有の要求を管理する方法
- 一般的なイメージ センサーからのデータを受信するために MIPI CSI-2 インターフェースを設定する方法
- ホストアプリケーションでビデオのストリームを表示し、カメラのプロパティを変更する方法
- デバッグ目的で、UVC デバイスに USB インターフェースを追加する方法
- オープン ソースのホスト アプリケーション プロジェクトを含む様々なプラットフォーム上で使用可能なホスト アプリケーションを検索する方法
- 必要に応じて、CX3 ファームウェアの問題を解決しデバッグする方法

10 関連リソース

- AN75705 - Getting Started with EZ-USB® FX3™
- AN70707 - EZ-USB ® FX3™/FX3S™ Hardware Design Guidelines and Schematic Checklist
- AN75779 - How to Implement an Image Sensor Interface Using EZ-USB ® FX3™ in a USB Video Class (UVC) Framework
- Designing FX3™/CX3-Based USB Type-C Products - KBA218460
- Ascella - Cypress® CX3™ THine® ISP 13MP reference design kit (RDK)
- Denebola – USB 3.0 UVC Reference Design Kit (RDK)
- Video: Cypress EZ-USB CX3 Introduction
- Video: Introduction to EZ-USB CX3 Solution for HD Video

改訂履歴

文書名: AN90369 - MIPI® CSI-2 イメージ センサーを EZ-USB® CX3™にインターフェースする方法

文書番号: 001-92480

版	ECN	発行日	変更内容
**	4377476	05/22/2014	これは英語版 001-90369 Rev. **を翻訳した日本語版 001-92480 Rev. **です。
*A	4928471	10/06/2015	これは英語版 001-90369 Rev. *C を翻訳した日本語版 001-92480 Rev. *A です。
*B	5789029	06/28/2017	最新のテンプレートおよびロゴに変更しました。
*C	6895497	06/17/2020	これは英語版 001-90369 Rev. *D を翻訳した日本語版 001-92480 Rev. *C です。

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店および販売代理店の世界的なネットワークを持っています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

Arm® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチセンシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

サイプレス開発者コミュニティ

コミュニティ | サンプルコード | Projects | ビデオ | ブログ | トレーニング | Components

テクニカル サポート

cypress.com/support

本書で言及するその他すべての商標または登録商標は、それぞれの所有者に帰属します。

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709