USB ビデオクラス (UVC) フレームワーク内で EZ-USB™ FX3 を使用してイメージセンサー インタフェースを実装する方法

About this document

Scope and purpose

USB 3.0 がもたらす広帯域幅は、ペリフェラルを USB に接続する IC への要求を高度にします。本アプリケーション ノートでは、非圧縮データを PC にストリームするカメラ (EZ-USB™ FX3 と通信するイメージセンサー) というよく利用されている USB 3.0 の応用を中心に説明します。本アプリケーション ノートは、インタフェースの柔軟性を犠牲にせずにデータ処理能力を最大限にするよう設計された EZ-USB™ FX3 の機能を強調します。また、USB ビデオ クラス (UVC) の実装情報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 AMCap、MPC-HC、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。最後に、EZ-USB™ FX3 のフレキシブルなイメージ センサー インタフェースを使用して 2 つのイメージ センサーを接続することで 3D イメージング アプリケーションとモーショントラッキング アプリケーションを実装する方法を説明します。

Intended audience

本ドキュメントの主な対象は、USB ビデオクラス (UVC) フレームワークで EZ-USB™ FX3 を使用したイメージセンサインタフェースを実装する必要がある方です。

関連製品ファミリ

CYUSB301x、**CYUSB201x**

その他のサンプル コードが必要な場合は、以下を参照してください。

USB SuperSpeed のサンプルコードのリストにアクセスするには、**サンプル コードのウェブページ**を参照してください。

Table of contents

About this document **1**
Table of contents **1**
1 はじめに **4**
1.1 詳細情報 6
2 USB ビデオ クラス (UVC) **7**
2.1 エニュメレーション データ 7
2.2 オペレーション コード 7
2.3 USB ビデオ クラス要件 8
2.3.1 UVC 用の USB ディスクリプタ 8
2.3.1.1 ビデオ制御インタフェース 9
2.3.1.2 ビデオ ストリーム インタフェース 9
2.3.2 UVC 固有の要求 11
2.3.2.1 制御要求 – 輝度、PTZ および拡張ユニット制御機能 14
2.3.2.2 ストリーム要求 – PROBE および COMMIT 制御機能 15

Table of contents

2.3.3 ビデオ データ フォーマット: YUY2 16
2.3.4 UVC ビデオ データ ヘッダー 16
2.3.5 静止画キャプチャ 18
2.3.5.1 Method-2 静止画キャプチャ 18
3 GPIF II イメージ センサー インタフェース 21
3.1 イメージ センサー インタフェース 21
3.1.1 GPIF II インタフェース要件 21
3.2 イメージ センサー インタフェースのピン マッピング 22
3.3 ピンポン DMA バッファ 23
3.4 デザイン戦略 24
3.5 GPIF II ステート マシン 26
3.6 GPIF II Designer を使用したイメージ センサー インタフェースの構築 28
3.6.1 プロジェクトの作成 28
3.6.2 インタフェースの定義 30
3.6.3 ステート マシンの描画 34
3.6.4 GPIF II ステート マシンの描画 34
3.6.5 GPIF II インタフェースの詳細変更 43
4 DMA システムの設定 45
4.1 DMA バッファについて 50
5 EZ-USB™ FX3 のファームウェア 52
5.1 アプリケーション スレッド 54
5.2 初期化 55
5.3 列挙 55
5.4 I^2C インタフェースによるイメージ センサー設定 55
5.5 ビデオ ストリームの開始 55
5.6 DMA バッファの設定 56
5.7 ビデオ ストリーム中の DMA バッファの扱い 56
5.8 ビデオ ストリームの終了 57
5.9 「デバッグ」インタフェースの追加 57
5.9.1 デバッグ インタフェースの詳細 57
5.9.2 デバッグ インタフェースの使用 61
5.9.2.1 シングル読み出し: 61
5.9.2.2 シーケンシャル読み出し: 64
5.9.2.3 シングル書き込み: 65
5.9.2.4 シーケンシャル書き込み: 66
6 ハードウェア設定 67
6.1 ハードウェア要件 67
6.2 SuperSpeed Explorer Kit 基板のセットアップ 67
7 UVC ベース ホスト アプリケーション 70
7.1 デモの実行 70
8 トラブルシューティング 73
8.1 ホストアプリケーションで見える黒い画面または不正な色 73
8.2 ホストアプリケーションで観測されるフレーム/秒の減少 74
9 2 個のイメージ センサー同士の接続 75
9.1 UVC によるインタリーブされたイメージの転送 76
9.1.1 例 1: 2 個の 640x480 モノクロ センサー 76
9.1.2 例 2: 2 個の 640x480 カラー センサー 77
9.2 現在のプロジェクトに新しいビデオフォーマットを追加するファームウェア修正チェックリスト 78

Table of contents

9.3 現在のプロジェクトに新しいビデオ解像度を追加するファームウェア修正チェックリスト 78
9.4 新しいセンサーをサポートするためのファームウェア変更のチェックリスト 78
10 まとめ 80
11 付録 A: EZ-USB™ FX3 DVK (CYUSB3KIT-001) のハードウェア設定詳細 81
11.1 EZ-USB™ FX3 DVK 基板の設定 81
12 付録 B: EZ-USB™ FX3 Explorer キットおよび Aptina Image Sensor 相互接続基板 (CYUSB3ACC-004) のハードウェア設定詳細 84
12.1 ハードウェア設定 84
参考資料 86
改訂履歴 87

1 はじめに

USB 3.0 がもたらす広帯域幅は、ペリフェラルを USB に接続する IC への要求を高度にしています。一般的な例としては、非圧縮データを PC にストリームするカメラがあります。本アプリケーション ノートでは、一方がイメージ センサーに接続され、もう一方が USB 3.0 ホスト PC に接続されるコンバータがインフィニオン EZ-USB™ FX3 チップを使用して実装されます。EZ-USB™ FX3 は、その第 2 世代の汎用プログラマブル インタフェース (GPIF II) を使用してイメージ センサーとのインタフェースを提供し、その SuperSpeed USB ユニットを使用して PC に接続します。EZ-USB™ FX3 ファームウェアは、イメージ センサーからのデータを UVC に準拠した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、AMCap、MPC-HC、Webcamoid または 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。

本アプリケーション ノートで指定された手順は、EZ-USB™ FX3 ファミリの USB 2.0 製品である EZ-USB™ FX2G2 にも適用されます。

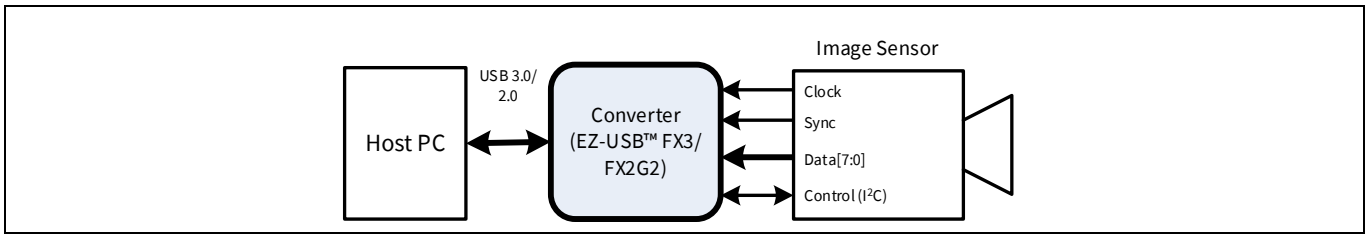


Figure 1 カメラ アプリケーション

*Note: EZ-USB™CX3 と MIPI CSI-2 イメージ センサーのインタフェースの情報は、**AN90369** を参照してください。*

Figure 1 にカメラ アプリケーションを示します。左側は SuperSpeed USB 3.0 ポートを搭載した PC です。右側は以下の特長を持っているイメージ センサーです。

- 8 ビット同期パラレル データ インタフェース
- 16 ビット/ピクセル (16bpp)
- YUY2 色空間
- 1280x720 の解像度 (720p)
- 30 フレーム/秒 (30fps)
- アクティブ HIGH のフレーム/ライン有効信号
- 正クロック エッジ極性

本アプリケーション ノートでは特定のイメージ センサー インタフェースについて説明しますが、これら特長はデータ バス幅や信号極性などのわずかな違いがあるだけで、多くのイメージ センサー インタフェースに共通しています。GPIF II ブロックのプログラマブル特性の結果として、これらの違いは容易に対応できます。また、EZ-USB™ FX3 はその I^2C インタフェースを使用し、イメージ センサーの設定用に制御バスを実装します。

Figure 2 は詳細なシステム ブロックダイヤグラムです。主なブロックに番号を付け、それぞれが実行するタスクを以下に説明します。

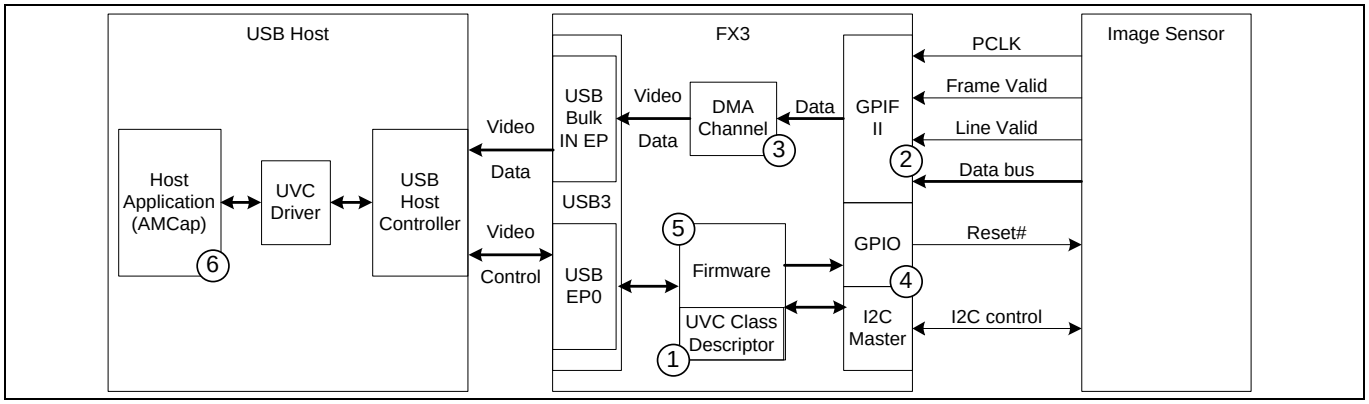


Figure 2　システム ブロックダイヤグラム

1. 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。詳細については **2 節**を参照してください。
2. パラレルバス インタフェースをイメージ センサーに実装します。これは EZ-USB™ FX3 GPIF II インタフェースを使用して行われます。**GPIF II Designer** と呼ぶインフィニオン ツールはグラフィカル ステート マシン エディターでカスタム波形の設計を可能にします。インタフェースがプログラム可能であるため、少し変更して、それを異なるイメージ センサー (例えば、16 ビット データ バスを持つイメージ センサー) 用にカスタマイズできます。詳細については **3 節**を参照してください。
3. イメージ センサー データを GPIF II ブロックから USB インタフェース 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イメージ センサーのビデオ データにヘッダー データを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ー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 にストリーム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。詳細については **4 節**を参照してください。
4. EZ-USB™ FX3 I^2C マスターを使用して、制御バスをイメージ センサーに実装します。I^2C と GPIO ユニットはインフィニオンの標準的なライブラリの呼び出しによりプログラムされます。それらは **5.4 節**で述べます。
5. EZ-USB™ FX3 ファームウェアは、EZ-USB™ FX3 ハードウェア ブロックを初期化し (**5.2 節**)、UVC カメラとして列挙し (**2.3.1 節**)、UVC 固有の要求を処理し (**2.3.2 節**)、ビデオ制御設定 (輝度など) を I^2C インタフェースを介してイメージ センサーに転送し (**5.4 節**)、UVC ヘッダーをビデオ データ ストリームに追加し (**2.3.4 節**)、ヘッダー付きのビデオ データを USB に転送し (**5.7 節**)、GPIF II ステート マシンを維持します (**5.8 節**)。
6. AMCap や MPC-HC、Webcamoid、VLC Media Player などのホスト アプリケーションは、UVC ドライバーにアクセスし、UVC 制御インタフェースを介してイメージ センサーを設定し、UVC ストリーム インタフェースを介してビデオ データを受信します。UVC ベースのホスト アプリケーションについては、**7 節**を参照してください。

カメラが USB 2.0 ポートに接続されている場合、EZ-USB™ FX3 ファームウェアは I2C 制御バスを使用し、フレームレートを 30FPS から 15FPS に、フレーム サイズを 1280x720 ピクセルから 640x480 ピクセルに縮小して、より低速な USB 帯域幅に対応します 。UVC の最大スループット(バイト単位)は、40MBps 以下です (フレーム高×フレーム幅×画素サイズ×フレーム/秒)。なお、測定されるフレーム/秒は、イメージセンサーのフレームブランキング時間、ラインブランキング時間に依存します。

PC ホストはオプションで制御インタフェースを使用し、カメラの輝度、パン、チルト、ズーム調整を送信できます。パン、チルトおよびズームは通常一緒に実装され、PTZ と呼ばれます。

1.1 詳細情報

インフィニオンは、**www.cypress.com** に大量のデータを掲載しており、ユーザーがデザインに対して適切なデバイスを選択し、デバイスをデザインに統合する手助けをしています。

- 概要: **USB ポートフォリオ**, **USB ロードマップ**
- USB3.0 製品セレクタ: **EZ-USB™ FX3**, **EZ-USB™ FX3S**, **EZ-USB™ CX3**, **EZ-USB™ HX3**, **EZ-USB™ SX3**
- アプリケーション ノート: インフィニオンは、基本レベルから高度なレベルまでの様々なトピックの USB アプリケーション ノートを提供しています。以下は EZ-USB™ FX3 入門用の推奨アプリケーション ノートです:
  - **AN75705** – EZ-USB™ FX3 入門
  - **AN231295** – EZ-USB™ SX3 入門
  - **AN70707** – EZ-USB™ FX3/FX3S ハードウェア設計ガイドラインおよび回路図チェックリスト
  - **AN65974** – EZ-USB™ FX3 スレーブ FIFO インタフェースを使った設計
  - **AN75779** – USB ビデオ クラス (UVC) フレームワーク内で EZ-USB™ FX3 を使用してイメージ センサー インタフェースを実装する方法
  - **AN86947** – EZ-USB™ FX3 で USB 3.0 のスループットを最適化
  - **AN84868** – EZ-USB™ FX3 を使用した USB 経由での FPGA コンフィギュレーション
  - **AN68829** – Slave FIFO interface for EZ-USB™ FX3: 5 ビットアドレスモード
  - **AN76348** – Differences in implementation of EZ-USB™ FX2LP and EZ-USB™ FX3 applications
  - **AN89661** – USB RAID 1 disk design using EZ-USB™ FX3S
- サンプルコード:
  - **USB Hi-Speed**
  - **USB Full-Speed**
  - **USB SuperSpeed**
- Knowledge base articles (KBA)
  - **UVC troubleshooting guide – KBA226722**
  - **Change bulk endpoint to isochronous in FX3 firmware – KBA231897**
  - **Invalid sequence error in multi-channel commit buffer - KBA218830**
  - **Handling commit buffer failures occurred during video transfers using EZ-USB™ FX3 – KBA231382**
  - **Modified AN75779 firmware for streaming video using cyusb3.sys driver – KBA233542**
  - **FX3/CX3: UVC extension unit application – KBA230466**
  - **Implementing extension unit control in AN75779 example project - KBA219280**
- 技術リファレンス マニュアル(TRM):
  - **EZ-USB™ FX3 technical reference manual**
- 開発キット:
  - **CYUSB3KIT-003, EZ-USB™ FX3 SuperSpeed explorer kit**
- モデル: **IBIS**

2 USB ビデオ クラス (UVC)

UVC に準拠するためには、2 つの EZ-USB™ FX3 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
- オペレーション コード

2.1 エニュメレーション データ

本アプリケーション ノートに添付されたコードは、UVC エニュメレーション データを格納する *cyfxuvcdscr.c* ファイル (**2.3.1 節**で説明する) を含んでいます。UVC ディスクリプタのフォーマットを定義する USB 仕様は、**usb.org** のビデオ クラスのセクションで入手できます。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:

1. カメラ入力端子 (IT)
2. 出力端子 (OT)
3. プロセッシング ユニット (PU)
4. 拡張ユニット (EU)

これら要素はディスクリプタで **Figure 3** に示すように接続しています。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ています。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。

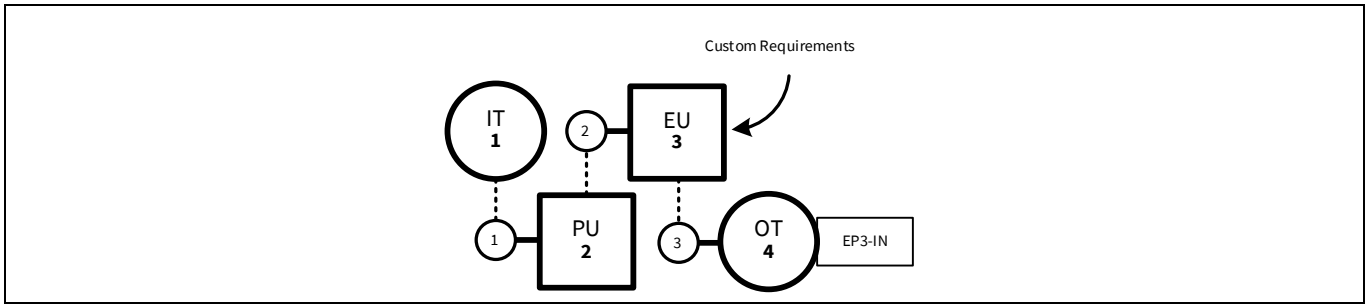


Figure 3 カメラ アーキテクチャの UVC 図

ディスクリプタは、ビデオ特性 (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、PTZ など) も含んでいます。標準のビデオ制御機能の他には、拡張ユニットはユーザー要件に基づいた追加の制御機能を提供します。

Note: UVC ドライバーが ISOC エンドポイントの BURST を 1 に制限するため、Windows7/8 マシンで USB 3.0 用の UVC で ISOC-IN エンドポイントを使用して達成できる最大帯域幅は 24Mbps です。しかし、Windows10 マシンでは ISOC 帯域幅を最大にするために BURST を 15 に設定できます。BULK エンドポイントにはこのような制限はありません。したがって、ISOC-IN エンドポイントはこのプロジェクトでは使用されません。詳細については、ここにアクセスしてください。

2.2 オペレーション コード

ホストがカメラを列挙した後、UVC ドライバーは一連の要求をカメラに送信して動作特性を判定します。これは機能要求フェーズと呼ばれます。このフェーズは、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 フェーズに先だって行われます。EZ-USB™ FX3 ファームウェアは、USB 制御エンドポイント (EP0) を介して到着した要求に応答します。

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することを示します。機能要求フェーズの間、UVC ドライバーはデバイスに照会し、関連する輝度パラメータを判定します。

ホスト アプリケーションが輝度値を変更するように要求すると、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。

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 サイズをストリームするために、UVC ドライバーはストリーム要求を発行します。要求には PROBE および COMMIT という 2 種類があります。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。ストリーム 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。

2.3 USB ビデオ クラス要件

本アプリケーション ノート用のファームウェア プロジェクトは **cyfxuvc_an75779** という名称のフォルダにあります。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 では次のようなデバイスが必要です:

- UVC 固有の USB ディスクリプタで列挙する
- 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
- ビデオ データを UVC 準拠のカラー フォーマットでストリームする
- イメージ ペイロードごとに UVC 準拠ヘッダーを追加する

これらの要件詳細は **UVC 仕様書**に記載されています。

2.3.1 UVC 用の USB ディスクリプタ

cyfxuvcdscr.c ファイルは USB ディスクリプタの表を含んでいます。バイト アレイ「CyFxUSBHSConfigDscr」 (Hi-Speed) および「CyFxUSBSSConfigDscr」 (SuperSpeed) は UVC 固有のディスクリプタを含んでいます。これら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んでいます:

コンフィギュレーション ディスクリプタ

- インタフェース関連ディスクリプタ
- ビデオ制御 (VC) インタフェース ディスクリプタ
  - VC インタフェース ヘッダー 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
    - 拡張ユニット 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 ストリーム (VS) インタフェース ディスクリプタ
  - VC インタフェース入力ヘッダー 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。インタフェース関連ディスクリプタは、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すために使用されます。ここで、このディスクリプタはビデオ制御 (VC) インタフェースとビデオ ストリーム (VS) インタフェースを備えた 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。

2.3.1.1 ビデオ制御インタフェース

VC インタフェース ディスクリプタとそのサブディスクリプタは、すべての制御インタフェース関連機能を報告します。例には、輝度、コントラスト、色相、露光、PTZ などの制御があります。

VC インタフェース ヘッダー ディスクリプタは、この VC インタフェースが属している VS インタフェースを指す UVC 固有のインタフェース 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、拡張ユニット 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 フィールドを含んでいます。

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、その中を流れるビデオの輝度、コントラスト、色相などのイメージ属性を制御します。

拡張ユニットは、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、拡張ユニットはありませんが、サンプル設計が実装されており、デバイス ファームウェア バージョンを取得または設定するために有効にできます。拡張ユニットを利用する場合、標準ホスト アプリケーションは、その機能を認識できるよう設計されない限り、認識できません。サンプル ホスト アプリケーションが、デバイス ファームウェア バージョンを取得または設定するために提供されます。独自の拡張制御機能とこれらの制御機能を照合するホスト アプリケーションを設計できます。サポートできる機能には、デバイス ファームウェアのバージョンとハードウェア ID の取得、センサー/イメージ シグナル プロセッサ (ISP) レジスタへの書き込みなどがあります。デバイスは複数の拡張ユニットをサポートできます。関連するファームウェア プロジェクトの UVC 拡張ユニットの詳細は **2.3.2 節** を参照してください。

出力端子はそれらユニット (IT、PU、EU) とホスト間のインタフェースを記述するために使用されます。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的な 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に、これらの機能を分割しています。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/ユニット ディスクリプタの「bmControls」ビット フィールド (**cyfxuvcdscr.c**) で、機能に対応するビットを「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、列挙の制御に関する詳細をポーリングします。詳細のポーリングは EP0 要求で実行されます。そのようなすべての要求 (ビデオ ストリーム要求を含む) は、uvc.c ファイル内の ***UVCAppEP0Thread_Entry*** 関数で処理されます。

2.3.1.2 ビデオ ストリーム インタフェース

ビデオ ストリーム インタフェース 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ム モードを切り替えます。

VS インタフェース入力ヘッダー 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または圧縮などのカラー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 レートを含んでいます。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 ストリームに使用されるバルク エンドポイントに関する情報を含む標準的な USB エンドポイント ディスクリプタです。

この例では唯一の解像度とフレーム レートを使用します。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)。

Table 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	SuperSpeed 値	Hi-Speed 値
23〜24	幅と高さの比	16:9	4:3

Table 2　　VS フォーマット 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	SuperSpeed 値	Hi-Speed 値
5〜8	解像度 (W、H)	1280x720	640x480
17〜20	最大イメージ サイズ (バイト単位)	1280x720x2 (2 バイト/ピクセル)	640x480x2 (2 バイト/ピクセル)
21〜24、26〜29	100ns 単位のフレーム間隔	0x51615 (30FPS)	0xA2C2A (15FPS)

複数のバイトの値が LSB ファースト (リトルエンディアン) でリストされていることに注意してください。したがって、例えば 0x00051615 のフレーム レートは 33.33 ミリ秒、すなわち 30FPS に相当します。

Table 3　　プローブ/コミット構造体の値

プローブ/コミット構造体のバイト オフセット	特性	SuperSpeed 値	Hi-Speed 値
2	フォーマット インデックス	1	1
3	フレーム インデックス	1	1
4〜7	100ns 単位のフレーム間隔	0x51615 (30FPS)	0xA2C2A (15FPS)
18〜21	最大イメージ サイズ (バイト単位)	1280x720x2 (2 バイト/ピクセル)	640x480x2 (2 バイト/ピクセル)

この設計は上記の表の項目を変更することで、1080p など異なるイメージ解像度に対応するように適合できます。

2.3.2 UVC 固有の要求

UVC 仕様では USB 制御エンドポイント EP0 は、制御とストリーム要求を UVC デバイスに通信するために使用されます。これらの要求は、ビデオ関連制御機能の特性を調べ、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機能は機能 (capability) 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ムできます。機能 (最初の特性) 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 ストリーム モードの特性 (カラー 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
- 最小値と最大値の間の値の数
- デフォルト値
- 現在の値

SET と GET は UVC 固有の要求の 2 種類です。SET は属性の現在の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。

以下に UVC 固有の要求一覧を示します:

- SET_CUR:SET 要求の唯一のタイプ
- GET_CUR:現在の値を読み出す
- GET_MIN:サポートされた最小値を読み出す
- GET_MAX:サポートされた最大値を読み出す
- GET_RES: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) を読み出す
- GET_DEF はデフォルト値を読み取ります(デフォルトのフレーム解像度など、任意のパラメータにデフォルトを設定するために適切な値を返します)。
- GET_LEN:バイト単位で属性のサイズを読み出す
- GET_INFO:特定の機能の状態/サポートを問い合わせる

UVC 仕様では、これらは特定の機能の必須または任意の要求として規定されています。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。カメラが特定の機能要求をサポートしていない場合、要求がホストからカメラに発行された時に制御エンドポイントを停止することによって、これを示す必要があります。

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。バイト フィールドは、**2.3.1 節**で説明された UVC 固有のディスクリプタと同じ構造に従う階層を持っています。第 1 レベルはインタフェースを識別します (ビデオ制御またはビデオ ストリーム)。

第 1 のレベルがインタフェースをビデオ制御として識別する場合、第 2 のレベルは端末またはユニットを識別し、第 3 のレベルはその端末または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:

- 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 レベルがインタフェースをビデオ ストリームと識別した場合、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ムを開始させる、またはストリーム 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ム 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ム 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る場合、ホストは第 2 レベルを COMMIT にセットして SET_CUR 要求を送信することによってそれを確認します。ホストとデバイスのやり取りを **Figure 6** に示します。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。

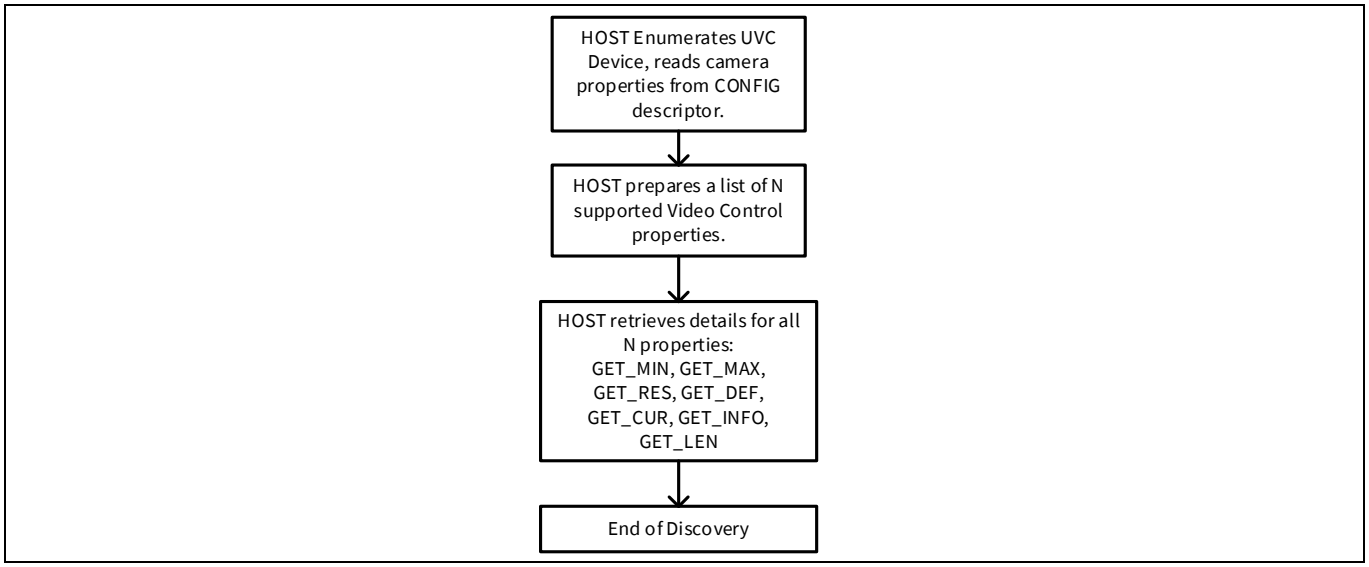


Figure 4　UVC 列挙および検出フロー

UVC デバイスが USB に差し込まれると、ホストはそれを列挙し、カメラがサポートする特性の詳細情報を調べます (**Figure 4**)。

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。**Figure 5** に、この相互作用を示します。

実装のこの種類は同期制御転送です。UVC 仕様では、デバイスは制御要求の 2 種類をサポートします:同期制御転送および非同期制御転送。

輝度、パン、チルト、ズームのような標準的 UVC 要求は、完了までに時間がかかりません (10ms 未満)。この場合、UVC デバイスはセンサー/ISP レジスタに書き込みを行い、センサー/ISP からの肯定応答を待ちません。ホスト ドライバーが SET_CUR 要求を送信すると、デバイスは 10ms 以内に成功 (要求を確認) または失敗 (要求を停止) の応答をします。UVC 仕様では、この要求は実質的に同期しています。すべての UVC ビデオ制御要求は同期制御転送でサポートできます。

ここで、ホスト ドライバーが露出値を設定すると仮定します。設計実装では、センサー/ISP は約 100ms で応答を返します。デバイスは常に SET_CUR 要求成功で SET_CUR 要求に応答し、センサー/ISP から確認応答を受信した後、デバイスは制御ステータス割込みエンドポイントに成功または失敗の値をロードします。要求のこの種類は非同期制御要求です。制御ステータス割り込みと関連する DMA チャネルは、このプロジェクトで既に設定されています(**uvc.c** ファイルの CY_FX_EP_CONTROL_STATUS を参照してください)。このプロジェクトでは、すべての要求が同期的であるため、非同期的な制御転送には使用されません。

Note:　ビデオコントロールインタフェースのコントロールステータス割り込みの使用方法については、UVC 仕様書の「Status Interrupt Endpoint」の章を参照してください。

*Note:　ハードウェアトリガーによるスチルキャプチャのための制御ステータス割り込みエンドポイントの使用例については、**uvc.c** ファイルの **CyFxSendStatusToHost** 関数を参照してください。非*

同期ビデオ制御の場合、表2-2 に基づいて、ステータス割り込みエンドポイントレスポンスを入力できます。UVC 規格のステータスパケット形式 (VideoControl Interface がオリジネータ) に、問い合わせたビデオコントロールに応じたコントロールセレクタ値 (bSelector) を設定します。

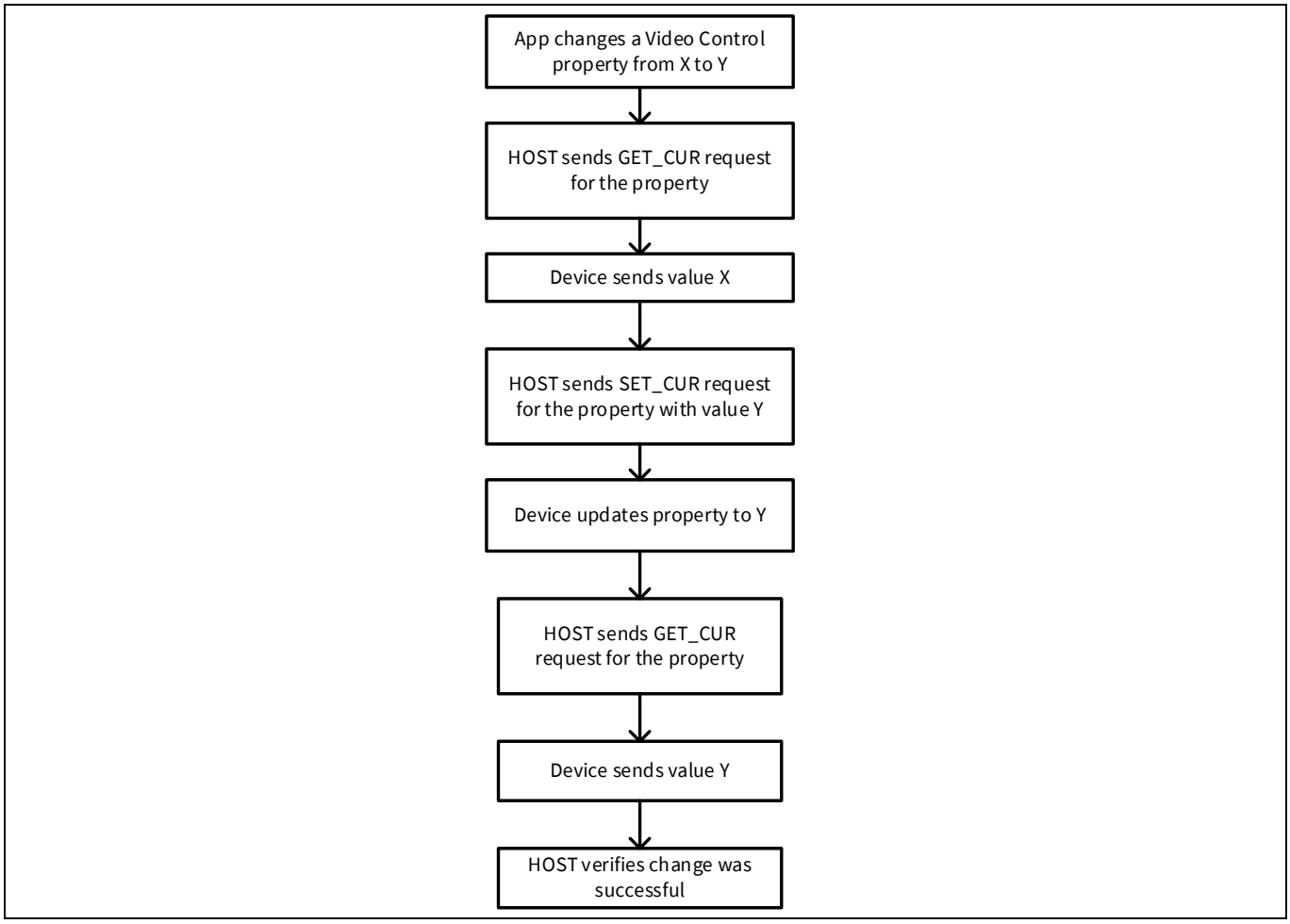


Figure 5　ホスト アプリケーションでカメラ設定を変更

ストリームを開始する前に、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ム モードを調べます。デフォルトのストリーム モードが決められた後、UVC ドライバーは COMMIT 要求を発行します。このプロセスは **Figure 6** 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ムする準備ができています。

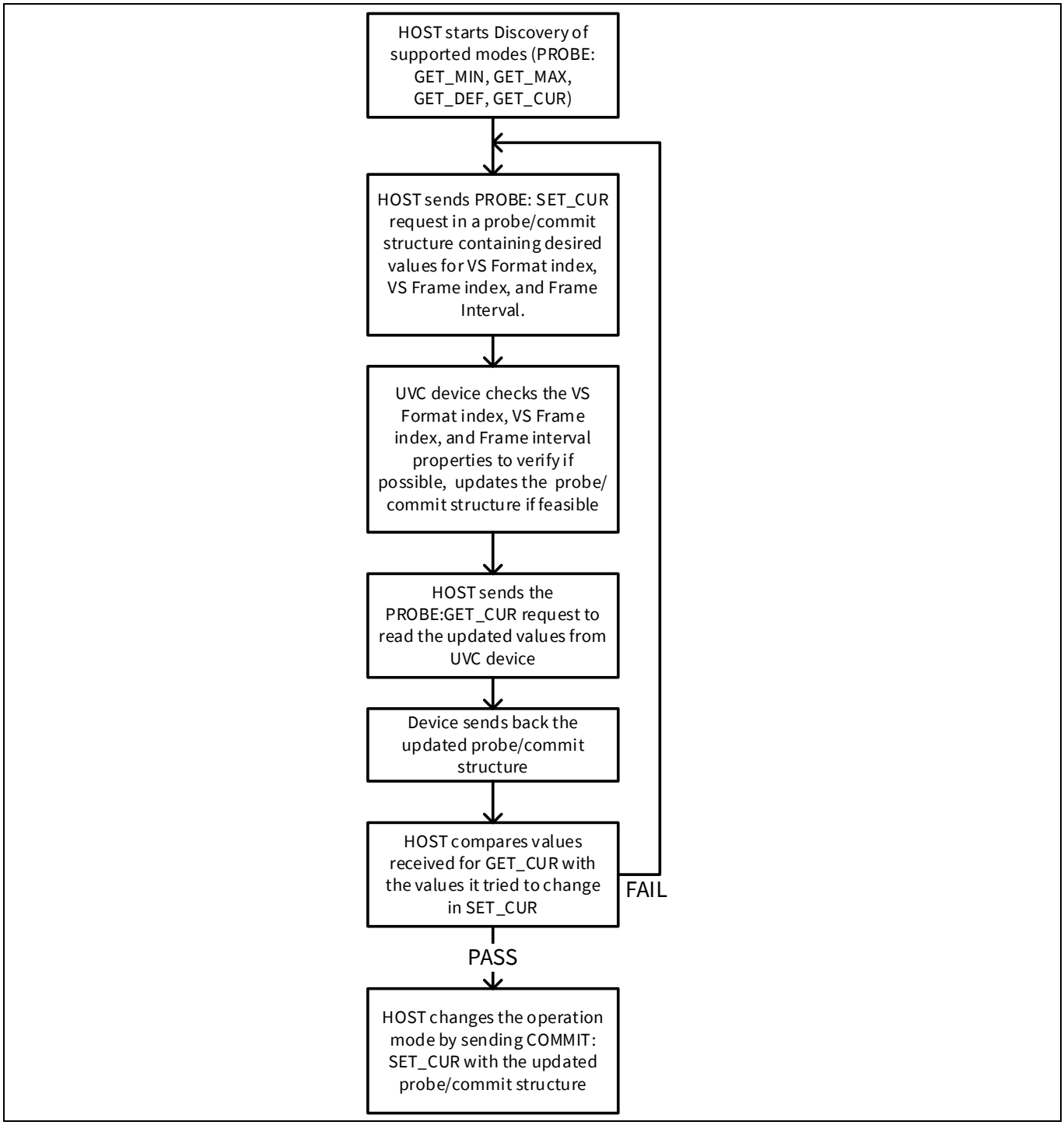


Figure 6　ホスト カメラのストリーム前のダイアログ

2.3.2.1　制御要求 – 輝度、PTZ および拡張ユニット制御機能

輝度と PTZ の制御機能は関連プロジェクトで実装されます。PTZ は *uvc.h* ファイルで「UVC_PTZ_SUPPORT」を定義することにより有効にできます。イメージ センサーはこれらの機能をサポートしない場合もあります。サポートしない場合、それらを実装するように特定のハードウェアを設計する必要があります。いずれの場合も、これらの機能の制御の USB 側ファームウェア実装は同じままです。ただし、イメージ センサー実装は異なります。したがって、これらの制御機能の USB 側を実装

するプレースホルダ関数のみが提供されます。イメージ センサー固有の PTZ 実装用にコードを書く必要があります。

Note: *アプリケーション ノートに記載されているすべての関数は、別途指定されていない限り、**uvc.c** ファイルに実装されています。このファイルは、本アプリケーション ノートに添付される zip 形式ソース コード内の **cyfxuvc_an75779** フォルダ下に格納されているプロジェクトの一部です。*

ホスト アプリケーションは、輝度制御用にプロセッシング ユニットにビデオ制御要求を (EP0 を介して) 送信します。すべてのセットアップ要求は **CyFxUVCAppInUSBSetupCB** 関数で処理されます。この関数により、ホストが UVC 固有の要求 (制御かストリーム) を送信したかを検出して、それぞれ

「CY_FX_UVC_VIDEO_CONTROL_REQUEST_EVENT」または「CY_FX_UVC_VIDEO_STREAM_REQUEST_EVENT」のイベント フラグをセットします。すると、フラグは **UVCAppEP0Thread_Entry** 関数 (EP0 アプリケーション スレッド) で処理されます。

輝度制御要求は、ビデオ制御要求イベント フラグをトリガーします。これらフラグのいずれかがトリガーされるのを待っている EP0 アプリケーション スレッドは、ビデオ制御要求をデコードして適切な関数を呼び出します。EP0 アプリケーション スレッドは **UVCHandleProcessingUnitRqts** 関数を呼び出して輝度制御要求を処理します。

プロセッシング ユニット関連の制御機能 (輝度、コントラスト、色相など) を実装するために、**UVCHandleProcessingUnitRqts** 関数を修正します。カメラ端子の制御機能を実装するために、**UVCHandleCameraTerminalRqts** 関数を修正します。ベンダー固有の要求を実装するために、**UVCHandleExtensionUnitRqts** 関数を修正します。いずれかの制御機能をサポートするためには、USB ディスクリプタで対応するビットをセットしなければなりません。UVC 仕様はカメラ端子、プロセッシング ユニットおよび拡張ユニット USB ディスクリプタの詳細が含まれています。

デバイス ファームウェア バージョン取得/設定という拡張ユニットのサンプル制御機能は関連アプリケーション プロジェクトで実装されます。この制御機能は *uvc.h* ファイルの「UVC_EXTENSION_UNIT」の定義で有効にできます。ファームウェアでの UVC 拡張ユニットの設計方法は **KBA219280** を参照してください。ファームウェア バージョン取得/設定の制御機能により、アプリケーション ノートのファームウェア バージョンを取得および設定ができます。これは *uvc_extension_app_x86.exe* (Windows 32 ビット版) または *uvc_extension_app_x64* (Windows 64 ビット版) を使用して実行できます。ホスト アプリケーション実行ガイドラインは添付されたプロジェクトの *readme.txt* ファイルに示されています。ホスト アプリケーションはマイクロソフト社 Media Foundation クラスおよび DirectShow API に基づきます。ホスト アプリケーションの詳細は、ここをご覧ください。

2.3.2.2 ストリーム要求 – PROBE および COMMIT 制御機能

UVCHandleVideoStreamingRqts はストリーム関連要求を処理します。UVC ドライバーが UVC デバイスからビデオをストリームする必要がある場合、第 1 フェーズはネゴシエーションです。このフェーズでは、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、EZ-USB™ FX3 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 インデックスを含んでいます。

「CY_FX_UVC_PROBE_CTRL」の switch 文の case は、USB 2.0 または USB 3.0 接続機能の場合にストリームのネゴシエーション フェーズを担当します (サポートされたビデオの特性はモードによって異なります)。USB 2.0 または USB 3.0 のいずれも同じストリーム モードがサポートされるため、GET_MIN、GET_MAX、GET_DEF、GET_CUR の報告された値は同じであることに注意してください。複数のストリーム モードをサポートする必要がある場合、これら値は異なります。

「CY_FX_UVC_COMMIT_CTRL」の switch 文の case は、ストリーミング フェーズの開始を処理します。COMMIT 制御機能用の SET_CUR 要求は、次にホストがストリーム ビデオを開始することを示します。したがって、COMMIT 制御機能用の SET_CUR は、主なアプリケーション スレッド **UVCAppThread_Entry** が GPIF II ステート マシンにビデオ ストリームを開始させるように指示する「CY_FX_UVC_STREAM_EVENT」イベントをセットします。

2.3.3 ビデオ データ フォーマット: YUY2

UVC 仕様では、ビデオ データ用にカラー フォーマットの 1 サブセットだけがサポートされています。このため、UVC 仕様に準拠したカラー フォーマットでイメージをストリームするイメージ センサーを選択するべきです。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る YUY2 という非圧縮のカラー フォーマットについて説明します。YUY2 カラー フォーマットは YUV カラー 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 はすべてのピクセルでサンプリングされますが、クロミナンス値 U および V は偶数ピクセルでのみサンプリングされます。これにより、「マクロ ピクセル」が作成されます。各マクロ ピクセルは合計 4 バイトを使用して 2 つのイメージ ピクセルを記述します。他のすべてのバイトが Y 値であり、U および V 値が偶数ピクセルのみを表していることに注意してください:

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)

カラー 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。

Note: *アプリケーション ノートに附属するプロジェクトでは RGB フォーマットはサポートされていません。RGB フォーマットのサポートは **FX3 SDK** のサンプル プロジェクト「cycx3_rgb16_as0260」および「cycx3_rgb24_as0260」を参照してください (パス: <FX3 SDK インストール パス\EZ-USB™ FX3 SDK\1.3\firmware\cx3_examples)。*

モノクロ イメージは UVC 仕様の一部としてサポートされていませんが、モノクロ イメージ データを Y 値として送信し、すべての U および V 値に 0x80 を設定することで、8 ビット モノクロ イメージを YUY2 フォーマットで表わせます。

2.3.4 UVC ビデオ データ ヘッダー

UVC クラスは非圧縮ビデオ ペイロード用に 12 バイト ヘッダーを必要とします。ヘッダー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 (EZ-USB™ FX3) がフレームごとにトグルする「new frame」ビットを含んでいます。EZ-USB™ FX3 コードはヘッダー内でエラー ビットをセットして、現行フレームのストリームに問題が発生したことを示すこともできます。この UVC データ ヘッダーはすべての 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。**Table 4** に UVC ビデオ データ ヘッダーのフォーマットを示します。

Table 4　　UVC ビデオ データ ヘッダー フォーマット

バイト オフセット	フィールド名	説明
0	HLF	bHeaderLength - HLF (ヘッダー長フィールド) は、ヘッダーの長さをバイト単位で示す
1	BFH	bmHeaderInfo - BFH (ビット フィールド ヘッダー) 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

バイト オフセット	フィールド名	説明
2〜5	PTS	dwPresentationTime - 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	scrSourceClock - 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 フィールドはオプションです。このファームウェアの例ではこれらフィールドは 0 になっています。

ビット フィールド ヘッダー (BFH) はフレームの最後まで変化する値を保持します。**Table 5** に UVC ビデオ データ ヘッダーの一部である BFH のフォーマットを示します。

Table 5　ビット フィールド ヘッダー (BFH) 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 フレームの開始される境界でトグルし、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 ヘッダー内の 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 ヘッダー内の 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 にセット
5	STI	STI (スティル イメージ) ビットはビデオ サンプルが静止イメージに属するかを示す
6	ERR	ERR (エラー) ビットはデバイス ストリームのエラーを示す
7	EOH	EOH (ヘッダーの終了) ビットがセットされた場合、BFH フィールドの終了を示す

Figure 7 に、本アプリケーションでどのようにこれらのヘッダーがビデオ データに追加されるかを示します。12 バイト ヘッダーは各 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 パケットを持っています。USB 3.0 バルク パケット サイズは 1024 バイトです。

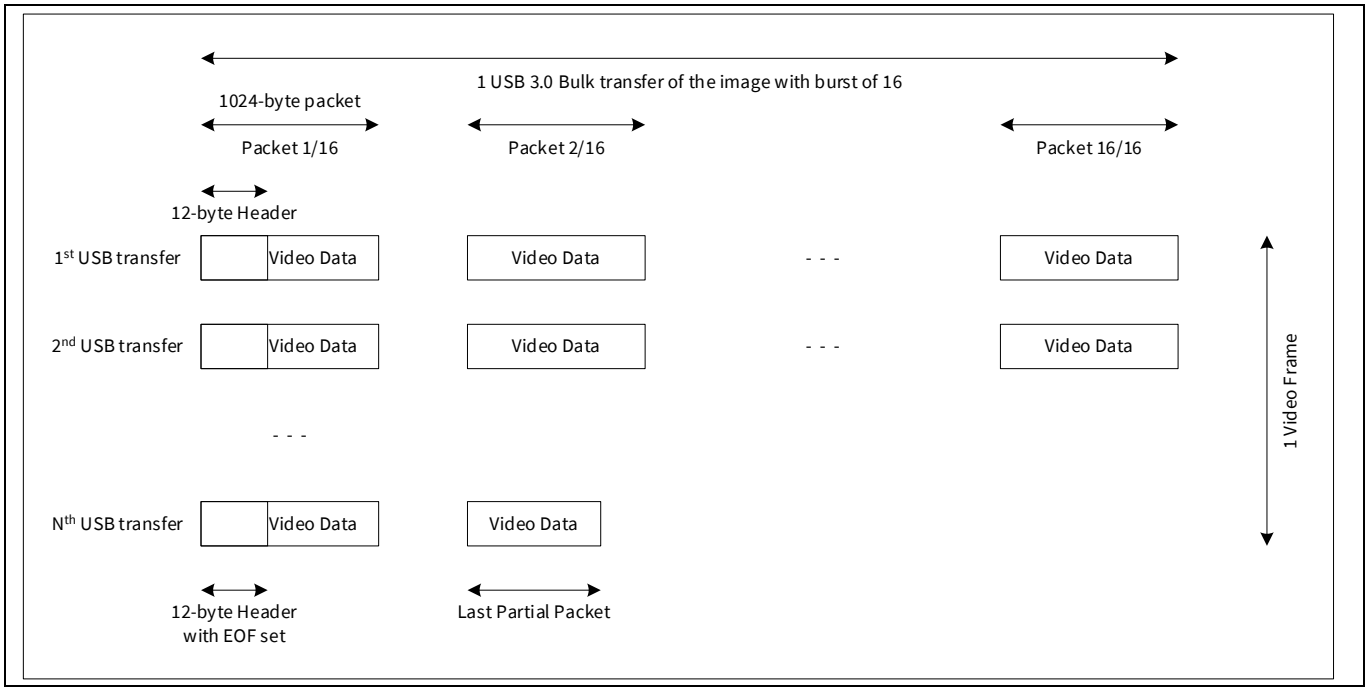


Figure 7　UVC ビデオ データ転送

2.3.5　静止画キャプチャ

ビデオカメラは、ビデオストリームからの静止画キャプチャにも対応します。これは、プログラムによるソフトウェアトリガーまたはハードウェアトリガーのいずれかによって開始されます。

静止画の撮影には 3 種類の方法があり、デバイスは該当する動画ストリーミング インタフェース記述子の中のクラス固有の記述子に対応する方法を指定しなければいけません。

このアプリケーションノートでは、Method-2 の静止画キャプチャを使用しています。その他の静止画キャプチャのモードについては、UVC 仕様書を参照してください。

2.3.5.1　Method-2 静止画キャプチャ

Method-2 静止画キャプチャは、ストリーミングの解像度とは異なる解像度の静止画をキャプチャする機能を追加します。この場合、ホストソフトは次のような動作をします。

1. 動画配信を一時的に停止します。
2. 静止プローブ/コミットネゴシエーションに基づいて最適な帯域代替設定を選択します。
3. 静止画転送オプションを指定して VS_STILL_IMAGE_TRIGGER_CONTROL 設定要求を送信します(UVC 仕様 4.3.1.4 章「静止画トリガー制御」を参照してください)。
4. 静止画データの受信を準備します。

このとき、ペイロードヘッダーにその旨を記載した静止画データを送信します。完全な静止画を受信すると、ホストソフトウェアは元の代替設定に戻し、ビデオストリーミングを再開します。この方式では、静止画フレームはストリーミングされる動画フレームと同じサイズである必要はありません。

ソフトウェアトリガーによる静止画キャプチャの場合、ホストソフトウェアがデバイスから静止画キャプチャを開始します。VS_STILL_IMAGE_TRIGGER_CONTROL SET リクエストに 「Transmit still image via dedicated bulk pipe」オプション(UVC 仕様の「Still Image Trigger Control」)を指定することによって、送信できます。この場合、要求を出した後、ホストは該当する動画ストリーミング インタフェースの静止

画一括エンドポイントから静止画の受信を開始します。静止画を撮影し、静止画一括エンドポイントにデータを送信します。

ハードウェアトリガー方式の静止画キャプチャの場合、ハードウェアトリガーを検出した後に静止画の送信を開始します。ボタン押下を検出すると、ステータス割り込みエンドポイントは、関連するビデオストリーミング インタフェースから割り込みを発行します。選択された**クラス別 VS インタフェース入力ヘッダー記述子**の **bTriggerUsage** フィールドが「initiating still image capture」に設定されている場合、デバイスは VS_STILL_IMAGE_TRIGGER_CONTROL の **bTrigger** フィールドを「Transmit still image via dedicated bulk pipe」に設定します。ホストソフトウェアは、割り込みを受けた後にデバイスが取り込んだ静止画データの受信を開始する必要があります。

静止画のプローブとコミットの制御要求は、**UVCHandleVideoStreamingRqts** 関数の CY_FX_UVC_STILL_PROBE_CONTROL と CY_FX_UVC_STILL_COMMIT_CONTROL ケース文で扱われます。静止画キャプチャは、同関数内の CY_FX_UVC_STILL_TRIGGER_CONTROL ケース文に実装されます。

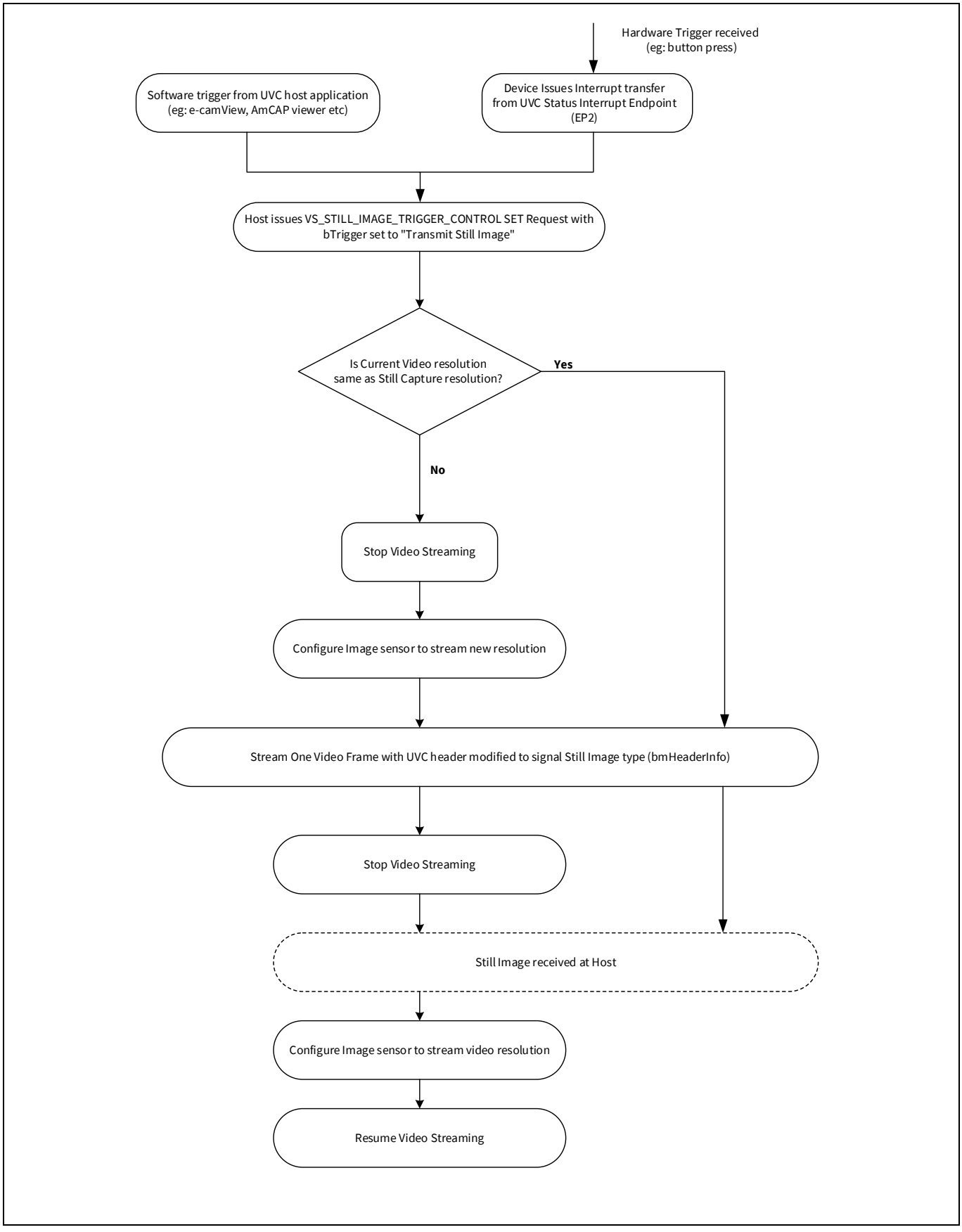


Figure 8　静止キャプチャのファームウェアフロー

3 GPIF II イメージ センサー インタフェース

EZ-USB™ FX3 の GPIF II ブロックは、フレキシブルなステート マシンであり、FX3 ピンをイメージ センサーなどの外部ハードウェアと通信するように駆動するためにカスタマイズできます。ステート マシンの設計には、インタフェース要件と EZ-USB™ FX3 の DMA 機能の理解が必要です。

3.1 イメージ センサー インタフェース

標準的なイメージ センサー インタフェースは **Figure 2** に示されます。通常、イメージ センサーは EZ-USB™ FX3 コントローラーからのリセット信号を必要とします。これは EZ-USB™ FX3 汎用入出力 (GPIO) を使用して実施されます。

一般的には、イメージ センサーは I^2C 接続を使用して、コントローラーがイメージ センサー パラメータの読み出し/書き込みを可能にします。イメージ センサーは同じ目的のために、シリアル ペリフェラル インタフェース (SPI) または汎用非同期レシーバ/トランスミッタ (UART) 接続を使用することもできます。EZ-USB™ FX3 の I^2C、SPI および UART ブロックはこの機能を提供できます。このアプリケーションでは、イメージ センサーを設定するために I^2C を使用します。

イメージを転送するために、イメージ センサーは以下の信号を提供します:

1. FV: フレーム有効 (フレームの開始と停止を示します)
2. LV: ライン有効 (ラインの開始と停止を示します)
3. PCLK: ピクセル クロック (同期インタフェース用のクロック)
4. DATA: イメージ データ用の 8 本のデータ ライン

Figure 9 は FV、LV、PCLK および DATA 信号のタイミング図を示します。イメージ センサーは FV 信号をアサートしてフレームの開始を示します。そして、イメージ データが 1 ラインごとに転送されます。各ライン転送中に、イメージ センサーが 8 ビット ピクセル データ (2 ピクセルごとに 4 バイトから成る) を YUY2 フォーマットで転送すると、LV 信号はアサートされます。バイト データは PCLK の各立ち上がりエッジで GPIF II ユニットにクロック入力されます。

EZ-USB™ FX3 GPIF II バスは 8 ビット、16 ビットまたは 32 ビット データ バスに設定できます。このアプリケーションでは、イメージ センサーの 8 ビット バスを使用します。イメージ センサーが 12 ビット バスなどのバイト単位に揃わないバスを提供する場合、 次のサイズ アップでパッドを使用するか、または未使用ビットを破棄します。

3.1.1 GPIF II インタフェース要件

タイミング図 (**Figure 9**) に基づいて、GPIF II ステート マシンは以下の要件があります:

- GPIF II ブロックは、LV と FV 信号がアサートされた時にのみデータをデータ ピンから転送する必要があります。
- イメージ センサーはフロー制御を備えていません。したがって、インタフェースは割込みなしでビデオをフル ラインで転送する必要があります。この設計では、ラインあたりに 1280 ピクセルがあり、各ピクセルには 2 バイトが必要なため、ラインごとに 2560 バイトが転送されます。
- CPU は、各フレームの終了時にそれに応じてヘッダー ビットを更新するために通知される必要があります。

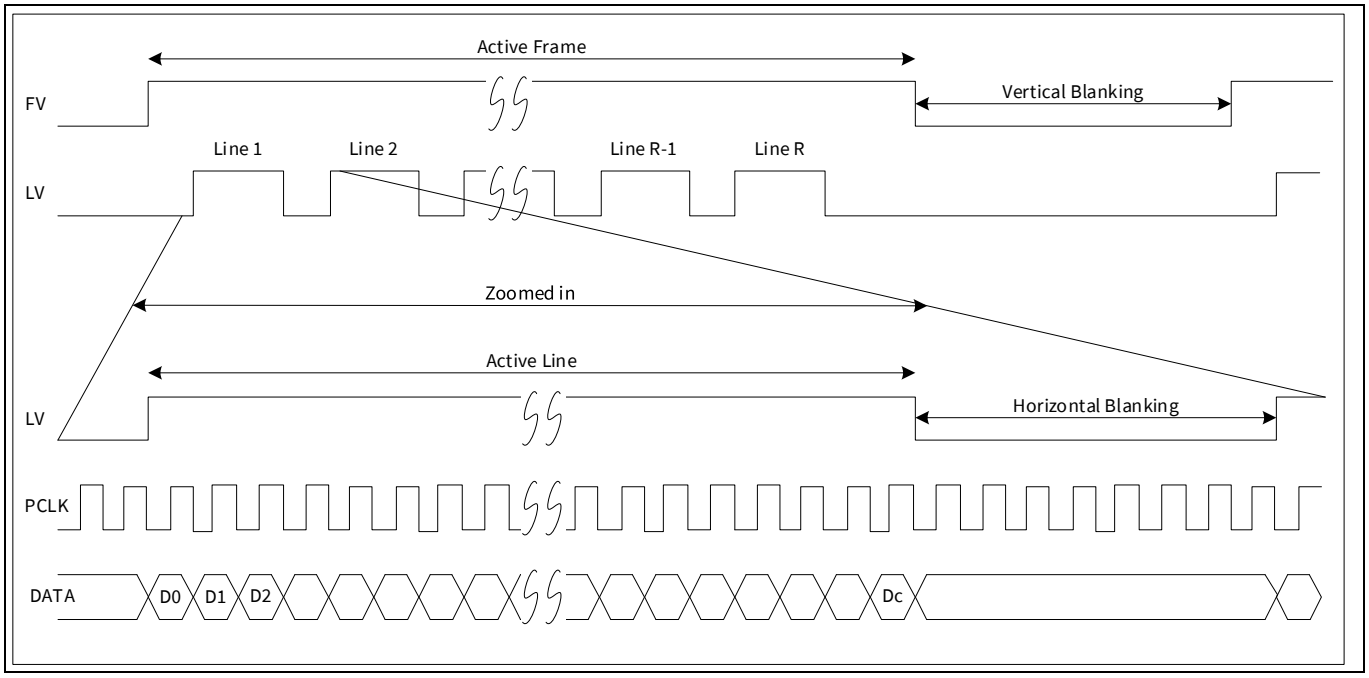


Figure 9　イメージ センサー インタフェースのタイミング図

3.2　イメージ センサー インタフェースのピン マッピング

GPIF II からイメージ センサー ピンへのマッピングは **Table 6** に示されます。

Table 6　パラレル イメージ センサー インタフェースのピン マッピング

EZ-USB™ FX3 のピン	8 ビット データ バスを使った同期パラレル イメージ センサー インタフェース
GPIO[28]	LV
GPIO[29]	FV
GPIO[0:7]	DQ[0:7]
GPIO[16]	PCLK
I^2C_GPIO[58]	I^2C SCL
I^2C_GPIO[59]	I^2C SDA

UART または SPI をインタフェースとして使用するイメージ センサーが 16 ビット データ バスを使う場合、**Table 7** に示されるピン マッピングを使用します。

Table 7　イメージ センサー用の追加ピン マッピング

EZ-USB™ FX3 のピン	イメージ センサー インタフェース (追加のピン)
GPIO[8:15]	DQ[8:15]
GPIO[46]	GPIO/UART_RTS
GPIO[47]	GPIO/UART_CTS
GPIO[48]	GPIO/UART_TX
GPIO[49]	GPIO/UART_RX
GPIO[53]	GPIO/SPI_SCK/UART_RTS

EZ-USB™ FX3 のピン	イメージ センサー インタフェース (追加のピン)
GPIO[54]	GPIO/SPI_SSN/UART_CTS
GPIO[55]	GPIO/SPI_MISO/UART_TX
GPIO[56]	GPIO/SPI_MOSI/UART_RX

*Note: EZ-USB™FX3 の完全なピン マッピングについては、データシート「**EZ-USB™FX3 SuperSpeed USB Controller**」を参照してください。*

3.3 ピンポン DMA バッファ

EZ-USB™ FX3 の入力と出力のデータ転送を理解するには、以下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
- GPIF スレッド

ソケットはペリフェラル ハードウェア ブロックと EZ-USB™ FX3 RAM の接続点です。USB、GPIF、UART、SPI など EZ-USB™ FX3 上の各ペリフェラル ハードウェア ブロックには、それぞれに対応する一定数のソケットがあります。各ペリフェラル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んでいます。

DMA ディスクリプタは、EZ-USB™ FX3 RAM に割り当てられたレジスター式のことです。これは DMA バッファのアドレスとサイズ情報および次の DMA ディスクリプタへのポインタを格納しています。これらのポインタは DMA ディスクリプタ チェーンを作ります。

DMA バッファは RAM の一部であり、EZ-USB™ FX3 デバイスを介して転送されるデータの中間記憶領域として使用されます。DMA バッファは EZ-USB™ FX3 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 ディスクリプタに格納されています。

GPIF スレッドは、ソケットに外部データ ピンを接続する GPIF II ブロックの専用データ パスです。

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 EZ-USB™ FX3 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。GPIF II ブロックから USB ブロックまでのデータ ストリームを例に取ってみます。GPIF ソケットはデータで DMA バッファが一杯になったことを 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、EZ-USB™ FX3 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ています。

一方、GPIF ソケットは EZ-USB™ FX3 CPU に割込みを送信して GPIF ソケットが DMA バッファを満たしたことを通知することもできます。EZ-USB™ FX3 CPU はこの情報を USB ソケットに伝達します。USB ソケットは EZ-USB™ FX3 CPU に割込みを送信して 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、EZ-USB™ FX3 CPU はこの情報を GPIF ソケットに伝達します。この実装は手動 DMA チャネルと呼ばれています。手動 DMA チャネルの実装は一般的に、EZ-USB™ FX3 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ています。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 ヘッダー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 ソケットと呼ばれています。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 ソケットと呼ばれています。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 (**4 節**)。ソケットは DMA バッファが一杯か空になった後、ある

DMA ディスクリプタから別の DMA ディスクリプタに切り替えるのに限られた時間 (数マイクロ秒) かかります。切り替え中、ソケットはデータを転送できません。この待ち時間は、フロー制御を備えていないインタフェースにとって問題になる場合があります。1 例としては、イメージ センサー インタフェースが挙げられます。

この問題は複数の GPIF スレッドを使用して GPIF II ブロックで解決されます。GPIF II ブロックは 4 個の GPIF スレッドを実装しています。1 度に 1 個の GPIF だけがデータを転送できます。GPIF II ステート マシンはデータを転送するためにアクティブな GPIF スレッドを選択する必要があります。

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。GPIF II ステート マシンは、内部制御信号または外部入力を使用してアクティブな GPIF スレッドを選択します。この例では、GPIF スレッド間の切り替えは内部制御信号によって制御されます。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。GPIF スレッドの切り替えに待ち時間はありません。GPIF II ステート マシンは、この切り替えを DMA バッファの境界で実行し、GPIF ソケットが新しい DMA ディスクリプタに切り替わるための待ち時間をマスクできます。これにより、DMA バッファが一杯になる時、GPIF II ブロックはセンサーからデータを失うことなく取り込めます。

Figure 10 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 バッファを満たす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する) および 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する) は、正しい順序でデータを読み出すために別の 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。

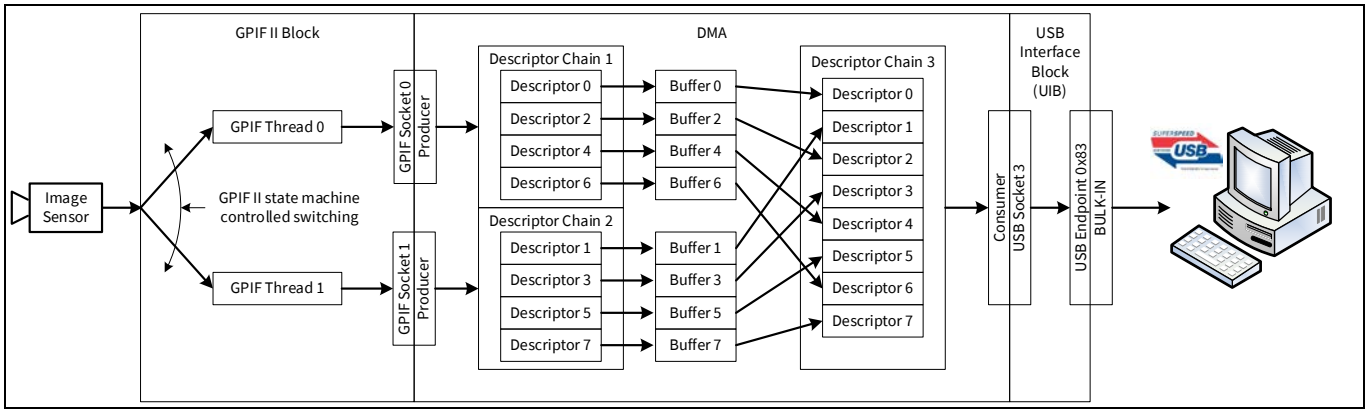


Figure 10　FX3 データ転送アーキテクチャ

3.4　デザイン戦略

詳細な GPIF II ステート マシンを構築する前に、基本的転送戦略を検討することは有用です。

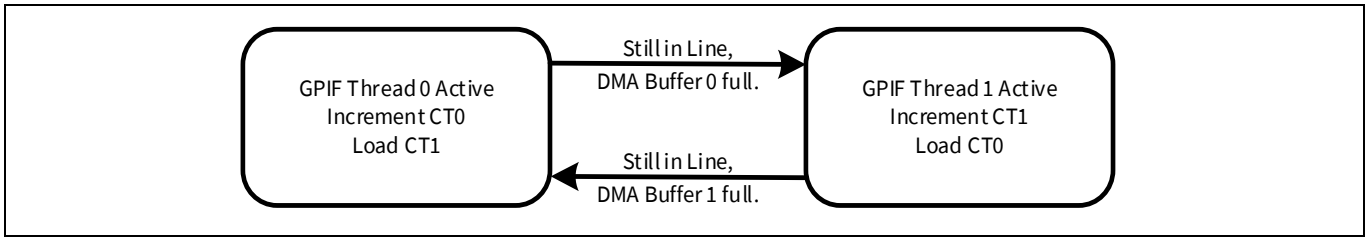


Figure 11　ビデオ ラインでのデータ転送

Figure 11 に、アクティブな水平ライン中に DMA バッファが一杯になる時の DMA バッファの基本的なピンポン動作を示します。GPIF II ステート マシンは 3 つの独立するカウンターを持ちます。この例では、GPIF II ユニットは 2 個のカウンター (CT0 と CT1) を使用して DMA バッファに書き込まれるバイト数をカウントします。カウントが DMA バッファ制限に到達すると、「DMA バッファ フル」分岐が実行され、GPIF II が GPIF スレッドを切り替えます。バイトが GPIF スレッド内で転送されている間、他の GPIF スレッドのカウンターは、次の GPIF スレッド切り替え後にバイトをカウントする準備ができるために初期化されます。

イメージ センサーはビデオ ラインの終了に達すると、LV をデアサートします。この時点では、Figure 12 のように、ステート マシンは幾つかのオプションがあります (ここで、LV はライン有効信号、FV はフレーム有効信号です)。選択は、DMA バッファが一杯になったかどうかおよびフレームが完了したかどうかに依存します。

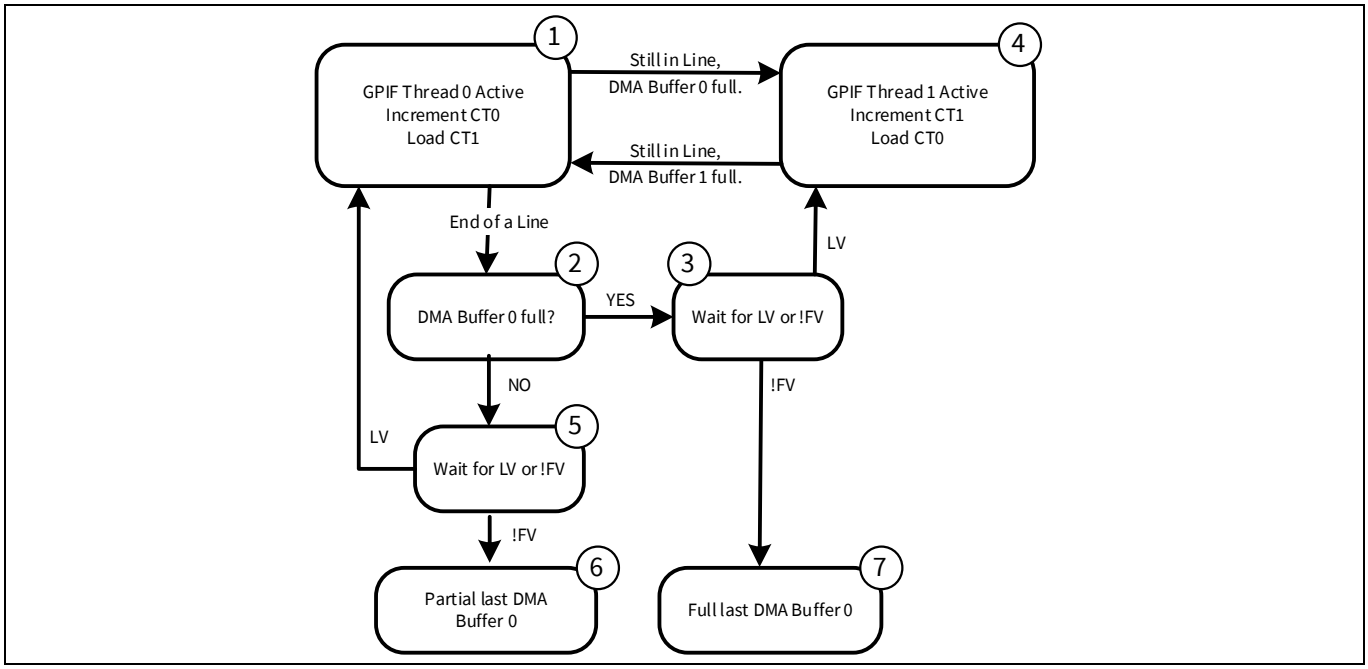


Figure 12　ビデオ ライン終了時のデータ転送の決定

Note:　「End of a Line」で始まるミラー イメージ決定ツリーもステート 4 から発します。これは、明確化のために図から省略されています。

ラインの終了 (LV=0) で、ステート マシンはステート 1 からステート 2 に遷移します。ステート 2 では、それは DMA バッファ バイト カウンターをチェックして DMA バッファが一杯になったかを確認します。そうであったら、ステート マシンはステート 3 に遷移します。ステート 3 では、完全なフレームが転送された場合に FV が LOW になり、ステート 7 に入ります。ステート 7 では、CPU は最後の DMA バッファが一杯になったことを示す割込み要求によって警告されます。CPU はこの情報を使用して GPIF II を次のフレーム用にセットアップできます。

ステート 3 で FV=1 の場合、イメージ センサーはまだフレームを転送しており、次のビデオ ラインを示すためにすぐに LV=1 に再アサートします。LV=1 になると、ステート マシンはステート 3 からステート 4 に遷移して、ステート 1 からステート 4 への遷移と同じように GPIF スレッド切り替えを行います。したがって、パス 1-4 とパス 1-2-3-4 の両方で GPIF スレッドが切り替わります。これらのパス間の違いは、2 番目のパスがラインの終了の休止があるため追加のサイクルを取ることです。

DMA バッファが一杯にならない場合、ステート マシンはステート 2 からステート 5 に遷移します。ステート 5 では、ステート 3 と同じように、イメージ センサーはフレームの終了または LV の再アサート (次

のビデオ ラインの開始) を待ちます。LV=1 の場合、ステート 1 の DMA バッファ 0 を満たし続けます。FV=0 の場合、イメージ センサーはビデオ フレームの送信を完了し、DMA バッファ 0 は部分的に一杯になります。ステート 6 では、ステート マシンは別の割込みを CPU に送信して、CPU が短い DMA バッファの構成を USB を介して送信できます。

3.5 GPIF II ステート マシン

EZ-USB™ FX3 GPIF II は、最大 256 ステートを持つプログラム可能なステート マシンです。各ステートは以下を含む動作を実行できます。

- マルチ制御ライン駆動
- データおよび/またはアドレスの送信または受信
- 内部 CPU への割込み送信

ステートの遷移は、DMA Ready やイメージ センサーのフレーム/ライン有効信号などの内部または外部信号に依存します。

GPIF II ステート マシン設計を始めるには、イメージ センサー波形でステート マシンの開始点を選びます。フレームの開始は、FV のポジティブ エッジへの遷移によって示され、論理的な開始点です。GPIF II はこのエッジを検出するために、まず FV=0 (最初のステート)、それから FV=1 (2 番目のステート) を待ちます。また 2 番目のステートは、ビデオ データで一杯になった 1 つの DMA バッファに対応するために転送カウンターを初期化します。ステート マシンはカウント値をテストし、カウンター限界値に達すると GPIF スレッド (DMA バッファ) を切り替えます。DMA バッファが一杯になると、カウンター限界値に達します。

ステート マシンは、GPIF II アドレス カウンターADDR とデータ カウンターDATA という 2 個の GPIF II 内部カウンターを使用して DMA バッファのバイト数をカウントします。GPIF II ステート マシンは GPIF スレッドを切り替えるたびに、他の GPIF スレッド用に適切なカウンターを初期化します。カウンター限界値をロードするのに 1 クロック サイクルかかるため、ロードされた値はターミナル カウントより 1 小さいです。

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、インタフェースのデータ バス幅に応じてカウンター限界値が変わります。この例では、**5.6 節**で説明されたように、データ バス幅が 8 ビット、DMA バッファ サイズが 16,368 バイトであるため、プログラムされる限界値は 16,367 となります。一般的に DMA バッファ カウント限界値は次のとおりです:

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

したがって、16 ビット インタフェースでは、DMA バッファ サイズが 8184 の 16 ビット ワードであるため、プログラムされる限界値は 8183 となります。32 ビット インタフェースでは、DMA バッファ サイズが 4092 の 32 ビット ワードであるため、プログラムされる限界値は 4091 となります。

詳細な GPIF II ステート マシンは **Figure 13** に示されます。2 つの DMA バッファ バイト カウンターは GPIF II DATA と ADDR カウンターです。**Figure 12** に示すように、これらのカウンターは CT0 と CT1 で対応します。

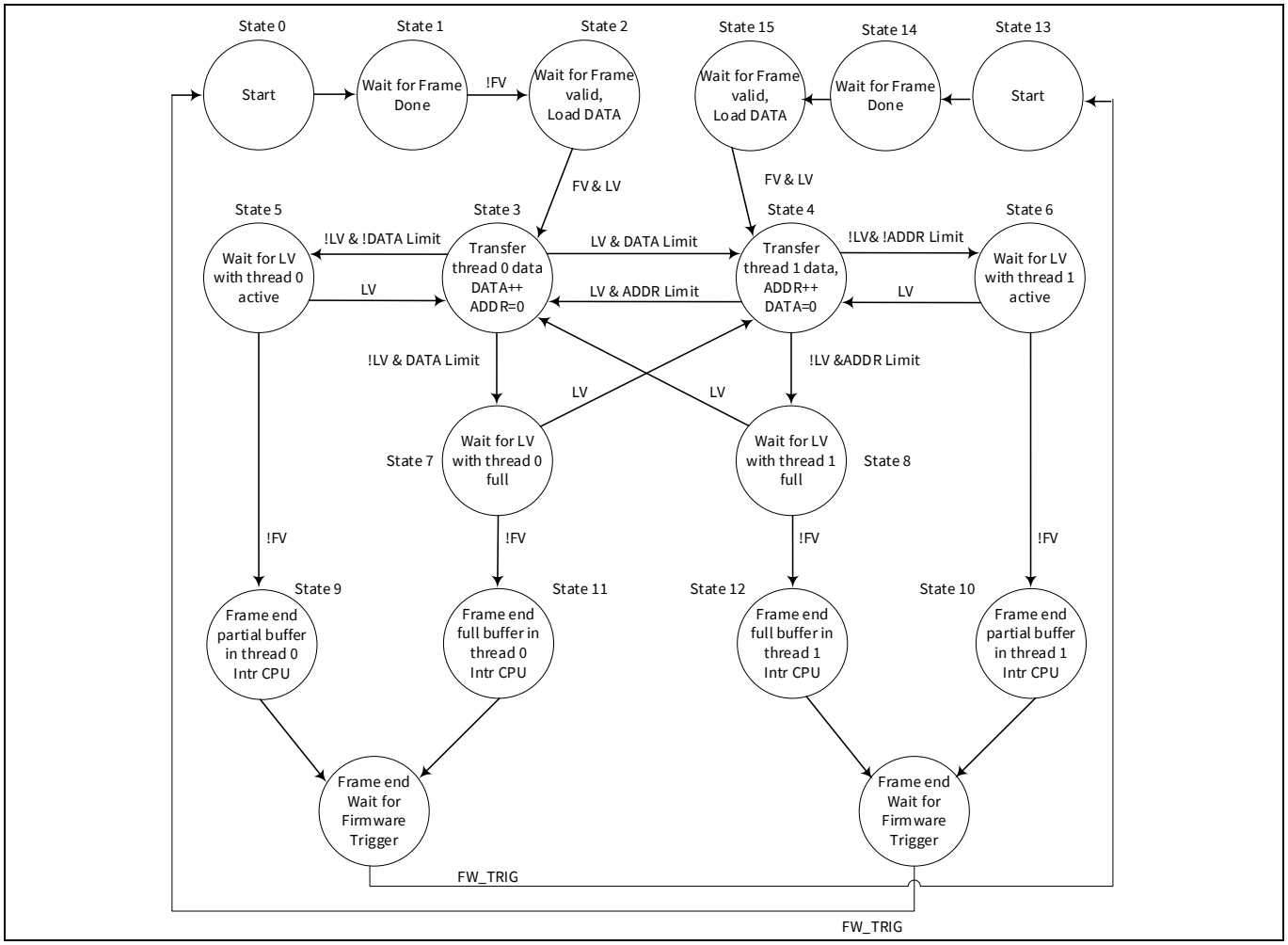


Figure 13　GPIF II ステート マシン図

各フレームの終了時に、CPU は 4 つのあり得る割込み要求の 1 つを受信して、DMA バッファの番号と一杯の状態を示します (ステート 9～12)。これらの要求を使用すると、コールバック関数を実装して、CPU が以下を含む幾つかのタスクを処理することを可能にすることもできます:

1. フレームの終了時に DMA バッファが一杯にならない場合、最後の DMA バッファを USB 転送用にコミットします (ステート 9 と 10)。フレームの終了時に DMA バッファが一杯になった場合、GPIF II はそれを USB 転送用に自動的にコミットします (ステート 11 と 12)。
2. コンシューマ ソケット (USB) が最後の DMA バッファ データを送信するのを待機してから、チャネルと GPIF II ステート マシンをステート 0 にリセットして、次のフレームの開始の準備をします。
3. アプリケーション固有の特別なタスクを処理して、次のフレームに進むことを示します。UVC では、12 バイト ヘッダー内のビットをトグルすることによってフレーム変更イベントを示す必要があります。

Figure 14 には、水平線の小さい部分でイメージ センサー波形を GPIF II と関係付けます。**4 節**の **Figure 46** のように、「Step」ラインが DMA システムとやり取りをします。

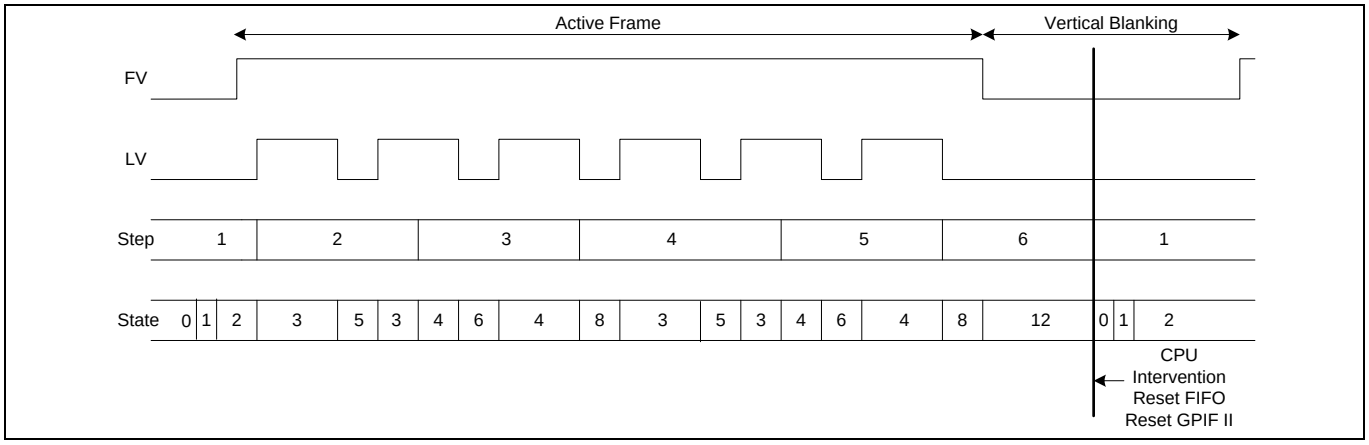


Figure 14　イメージ センサー インタフェース、データ パス実行およびステート マシンの相互関係

3.6　GPIF II Designer を使用したイメージ センサー インタフェースの構築

本節では、GPIF II Designer を使用したイメージ センサー インタフェースの設計を説明します。ご参考までに、完成したプロジェクトが添付ソースの zip ファイル内の 「**fx3_uvc.cydsn**」ディレクトリ パスにあります。

設計手順には 3 ステップがあります:

1. GPIF II Designer でプロジェクトを作成する
2. インタフェース定義を選択する
3. キャンバスにステート マシンを描画する

3.6.1　プロジェクトの作成

GPIF II Designer を起動します。GUI を **Figure 15** に示します。続く図に示すように、ステップバイステップの指示に従ってください。各図にはサブステップが含まれています。各ステップの詳細については、**GPIF II Designer ユーザー ガイド**を参照してください。

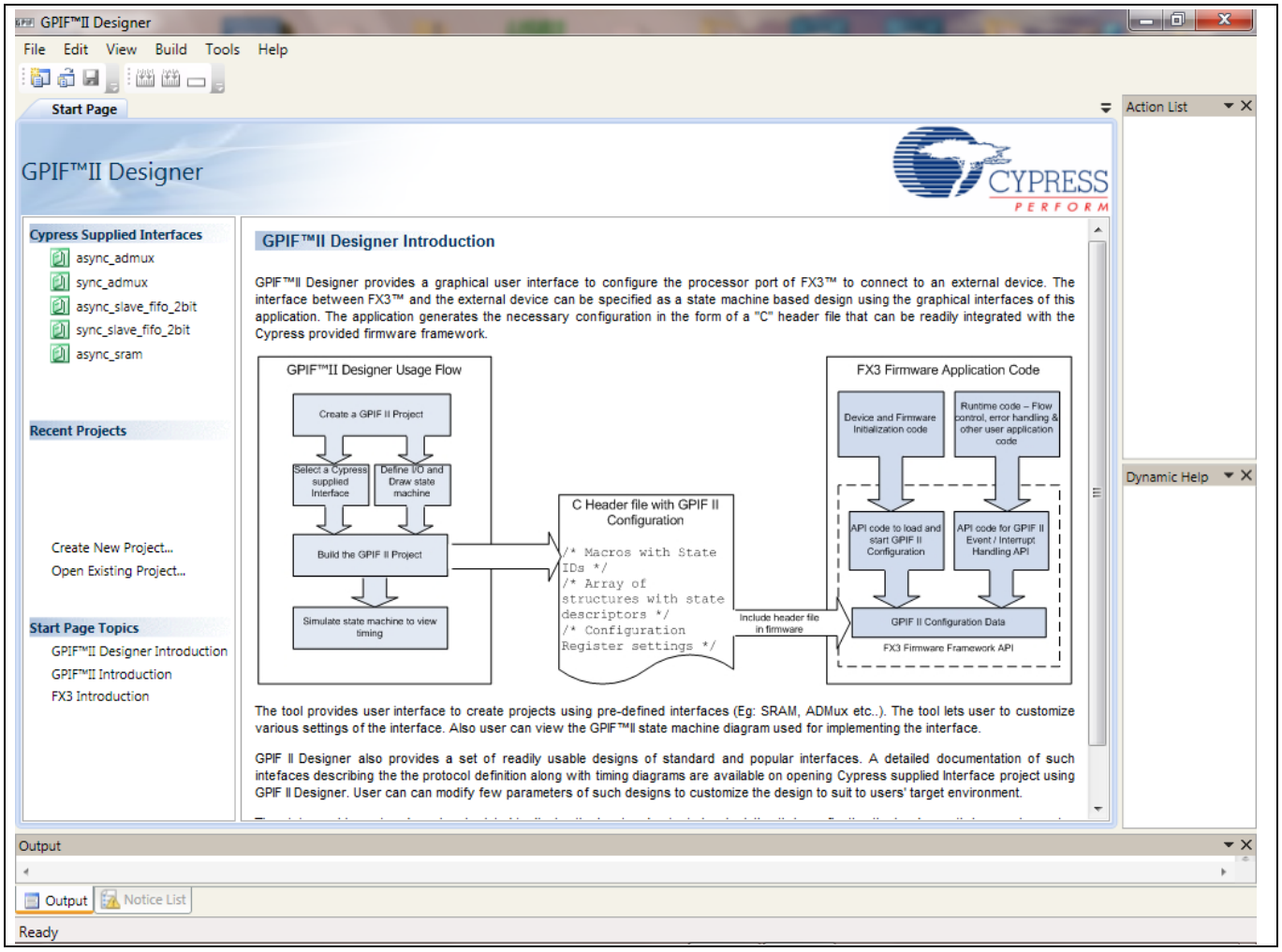


Figure 15　GPIF II Designer を起動します

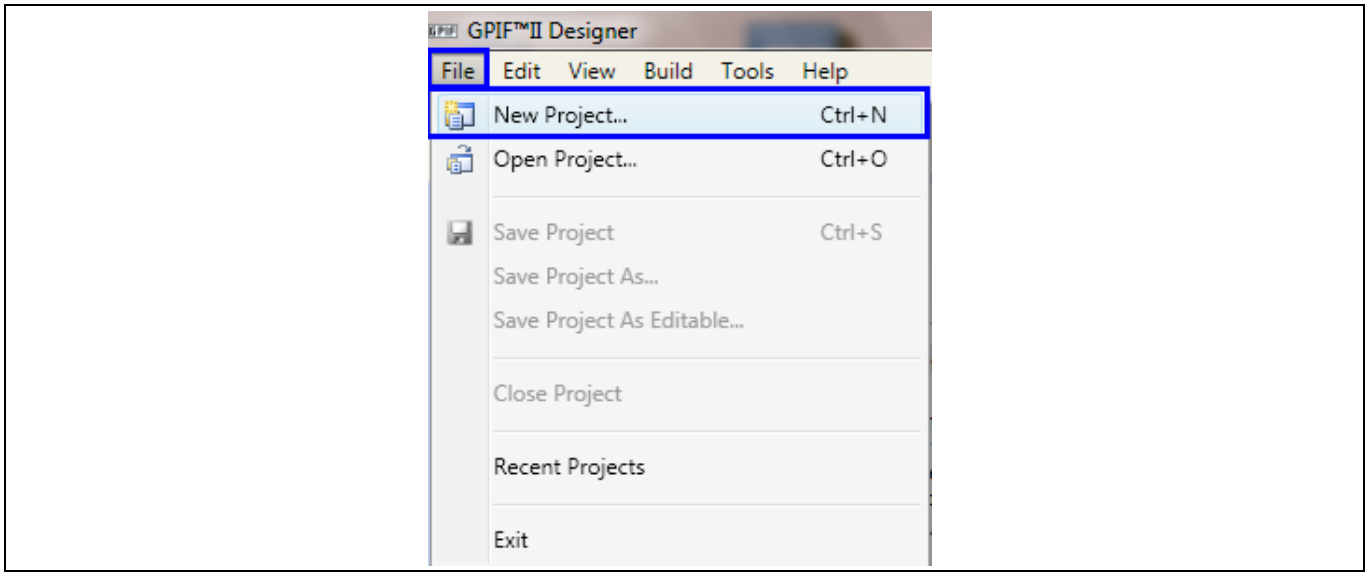


Figure 16　ファイル メニューを開いて「New Project」を選択

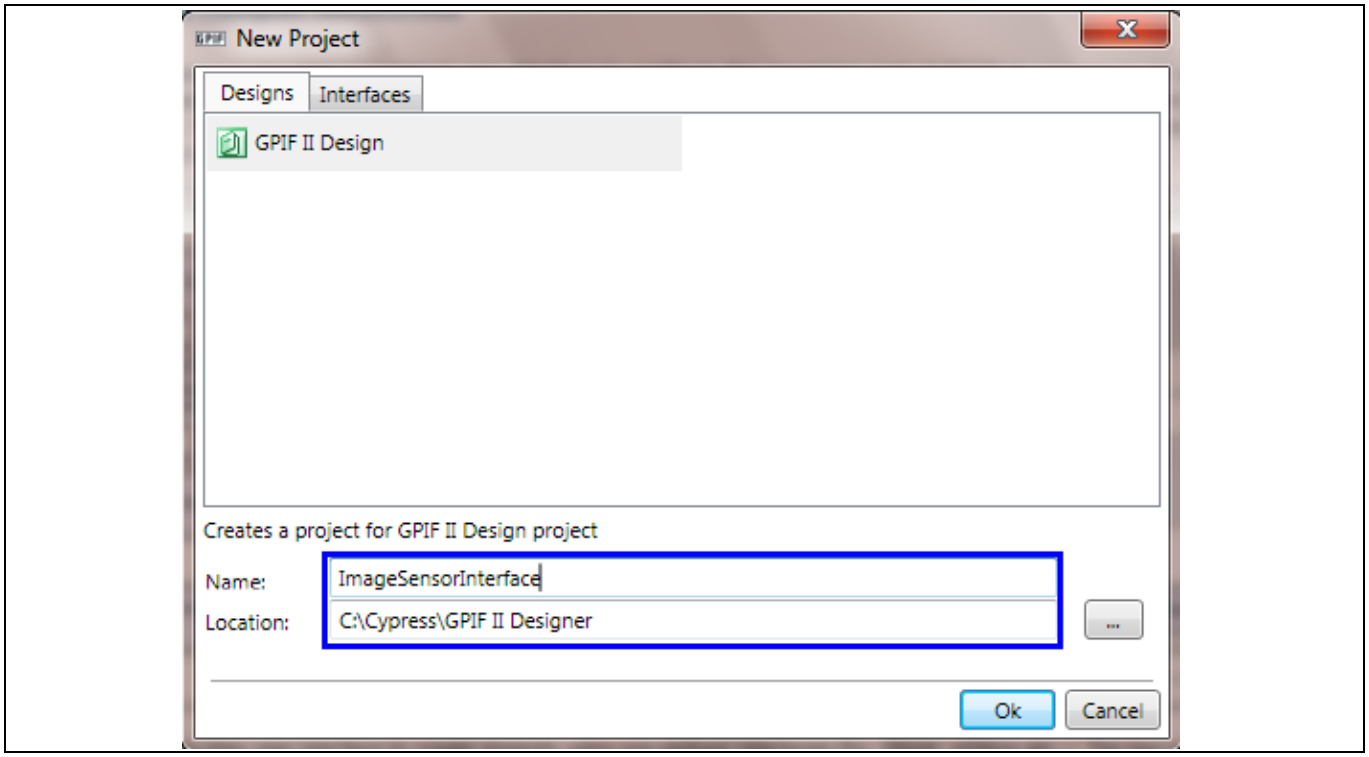


Figure 17　プロジェクト名と場所を入力

ここで、プロジェクト作成が完了し、GPIF II Designer では「**Interface Definition**」と「**State Machine**」タブへアクセスできます。「**Interface Definition**」タブで、EZ-USB™ FX3 は左側にあり、イメージ センサー (「Application Processor」ラベル付き) は右側にあります。次のステップは 2 つの間のインタフェースを設定することです。

3.6.2　インタフェースの定義

このプロジェクトでは、EZ-USB™ FX3 デバイスに接続したイメージ センサーは 8 ビット データ バスを持っています。これは LV 信号用に GPIO 28、FV 信号用に GPIO 29、センサー リセット用にアクティブ LOW 入力である GPIO 22 を使用します (**Table 6**)。このイメージ センサーは、EZ-USB™ FX3 への I^2C 接続も使用して、例えばセンサーを 720p モードに設定するためにイメージ センサー レジスタにアクセスします。「Interface Settings」列には、必要な選択肢があります。

さらに、指定された入力信号は次のフェーズで利用可能になり、遷移方程式を作成します。**Figure 18** に、選択するインタフェース設定を示します。

1. **Signals** で、LV と FV 用に 2 つの入力を選択します。
2. **Signals** で、nSensor_Reset 用に 1 つの出力を選択します。
3. **FX3 peripherals used** で、I^2C を選択します。これは、EZ-USB™ FX3 I^2C マスターをアクティブにします。
4. **Interface type** で、**Slave** を選択します。イメージ センサーがクロックを供給し、データ バスを駆動するため、GPIF II はスレーブ デバイスとして動作します。
5. **Communication type** で、**Synchronous** を選択します。これは、イメージ センサーからの同期クロックの存在を反映します。

6. **Clock settings** で、**External** を選択します。イメージ センサーはその PCLK 信号を GPIF II に送信します。
7. **Active clock edge** で、**Positive** を選択します。イメージ センサーはポジティブ エッジでデータ遷移をトリガーします。
8. **Endianness** で、**Little endian** を選択します。エンディアンは幅が 1 バイトより広いデータ バスのバイト順を示します。8 ビット インタフェースでは、この設定は関係ありません。
9. **Address/data bus usage** で、**8 bit** を選択します。イメージ センサーは 8 ビット データ バスを提供します。

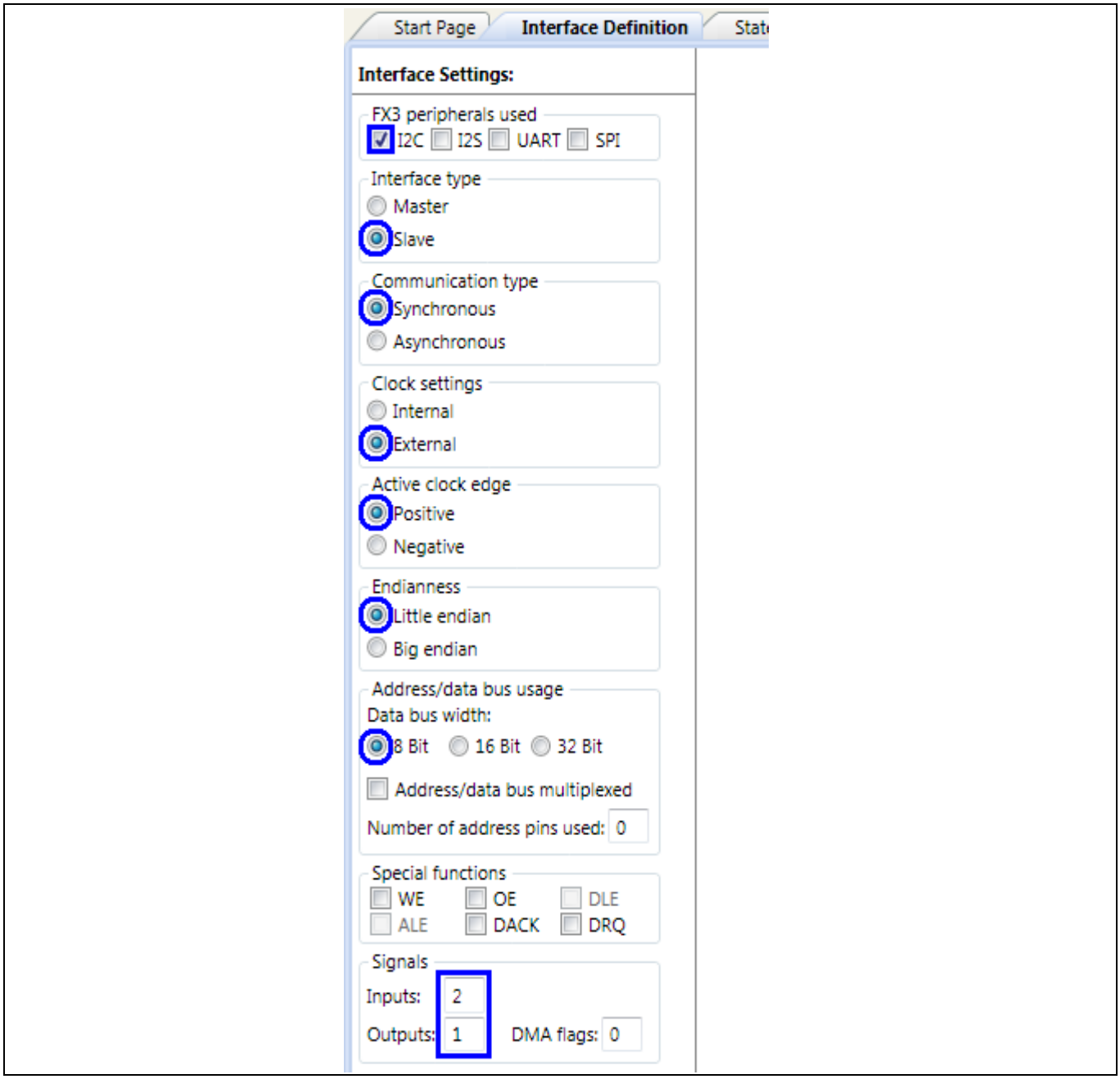


Figure 18　インタフェース設定の選択

「I/O Matrix Configuration」は **Figure 19** に示すようになります。次のステップは入力と出力の信号プロパティを変更することです。プロパティは、信号名、ピン マッピング (すなわち、どの GPIO が入力か出力として動作するか)、信号の極性、出力信号の初期値を含んでいます。

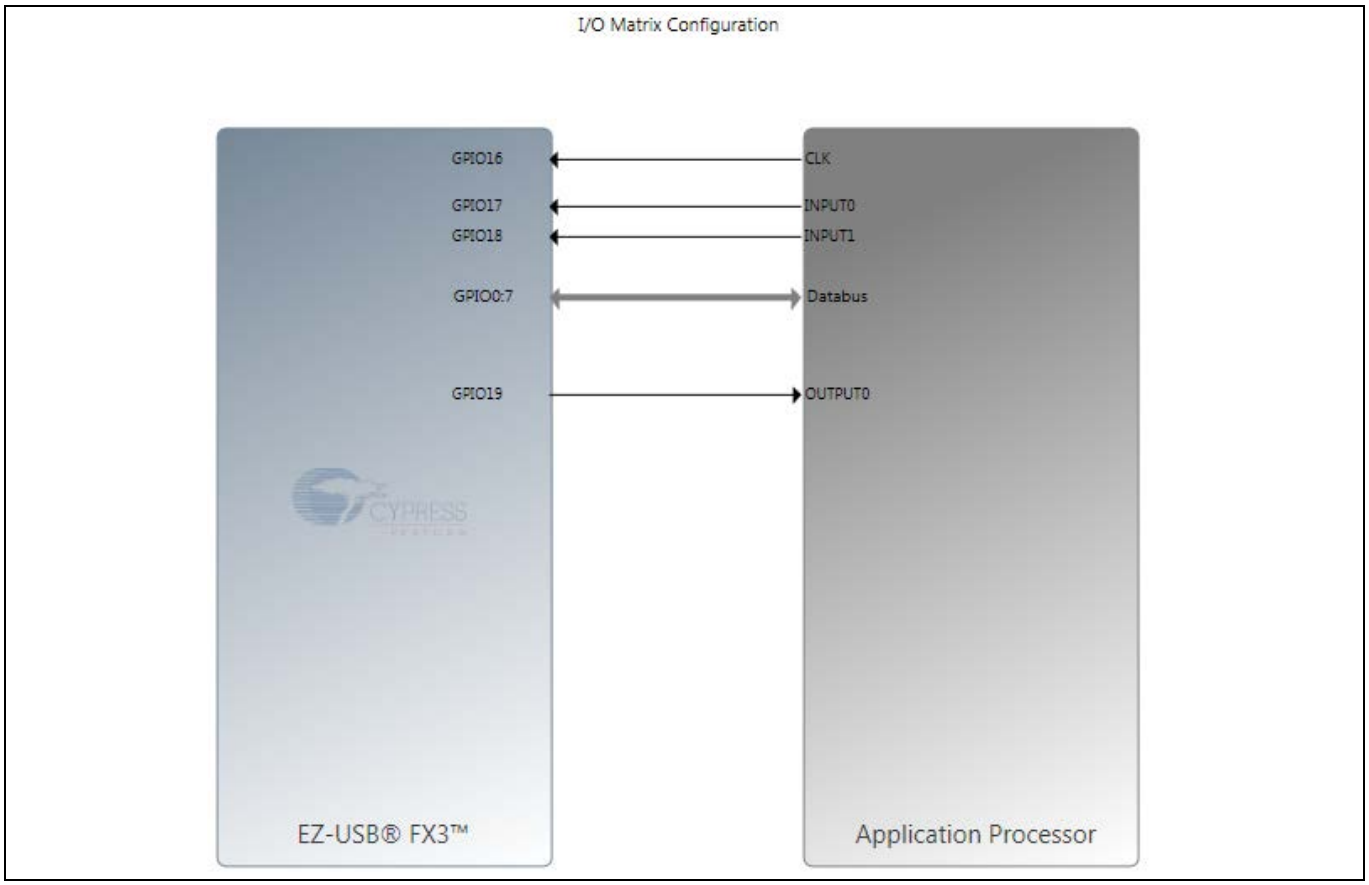


Figure 19　今までのブロックダイヤグラム

Figure 20 に示すように、**アプリケーション プロセッサ**領域の INPUT0 ラベルをダブルクリックし、その入力信号のプロパティを開きます。

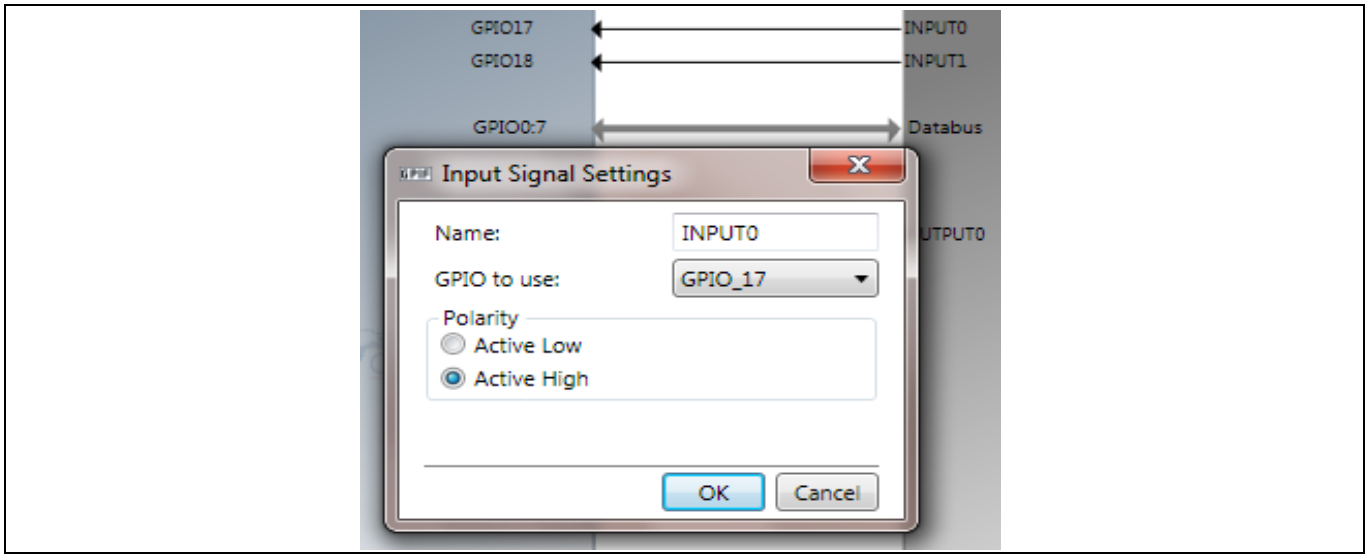


Figure 20　INPUT0 のデフォルト プロパティ

信号名を **LV** に変更し、**GPIO to use:**の値を **GPIO_28** に変更します (**Table 6**)。極性を **Active High** のままにします。遷移は **Figure 21** のようになります。「OK」をクリックします。

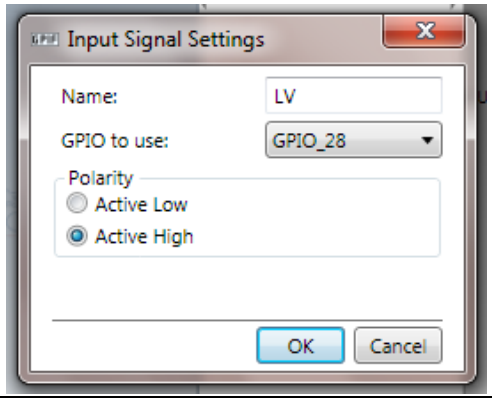


Figure 21　編集された INPUT0 のプロパティ

次に、INPUT1 ラベルをダブルクリックし、信号名を **FV** に変更し、GPIO の割当てを **GPIO_29** に変更し、極性を **Active High** のままにします (**Figure 22**)。

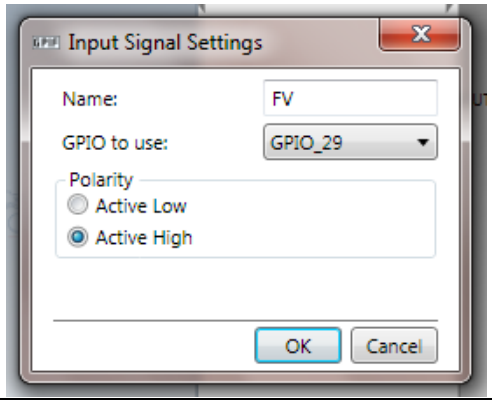


Figure 22　編集された INPUT1 のプロパティ

OUTPUT0 ラベルをダブルクリックし、**Figure 23** に示すとおり設定を変更します。

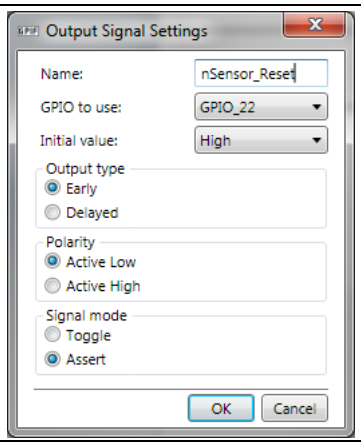


Figure 23　編集された OUTPUT0 のプロパティ

これで、インタフェース設定は完了です。信号名などのプロパティを設定すると、GPIF II Designer の他のすべての部分が更新されて変更を反映します。例えば、ブロックダイヤグラムには一般的な入力と出力名の代わりに信号名が現れます。また、ステート マシン デザイナーを使用する際、LV 信号や FV 信号などの利用可能な信号オプションが自動的にドロップダウンリストに選択肢として表示されます。

3.6.3 ステート マシンの描画

State Machine タブをクリックしてステート マシン キャンバスを開きます。ステート マシンの設計は次の 3 つの基本的な動作を含みます:

1. ステートを作成する
2. ステートにアクションを追加する
3. 遷移用の条件を使用してステート間の遷移を作成する

3.6.4 GPIF II ステート マシンの描画

State Machine タブをクリックしてキャンバスを開きます。未編集のキャンバスには START と STATE0 の 2 つのステートがあります。無条件の遷移 (LOGIC_ONE) でステートを接続させます (**Figure 24**)。

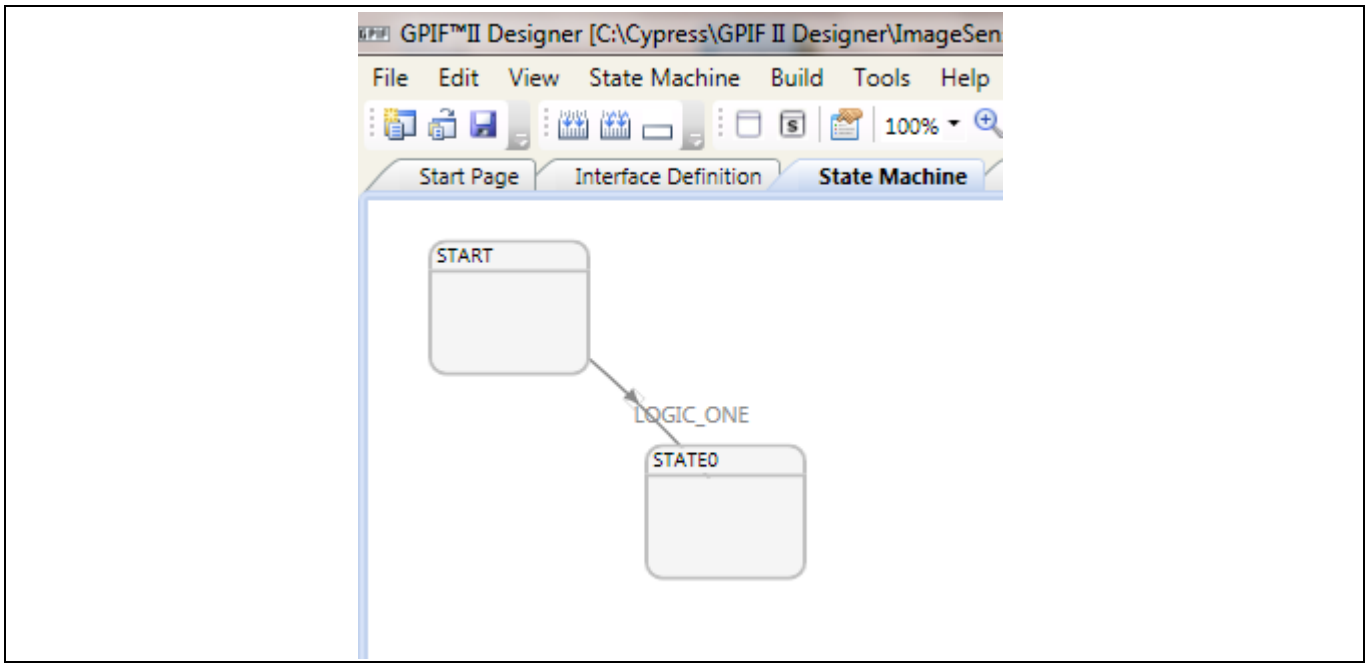


Figure 24 初期のステート マシン キャンバス

1. START ボックスの内側をダブルクリックし、名前のテキスト ボックスを編集することで START ステートの名称を START_SCK0 に変更します。同様に STATE0 ステートの名称を IDLE_SCK0 に変更します。「Repeat actions until next transition」チェックボックスでは、ステートに入ってからアクションを 1 回起こすか、またはステート中にすべてのクロックで起こすかを決めます。このようなアクションなしのステートでは、チェックボックス ステートは該当なしです。
2. キャンバス内の空いている場所を右クリックし、「Add State」を選択して新しいステートを追加します。新しいステートをダブルクリックし、その名称を WAIT_FOR_FRAME_START_0 に変更します。
3. IDLE ステートから WAIT_FOR_FRAME_START_0 ステートへの遷移を作成します。カーソルを IDLE ステート ボックスの中央に合わせます。カーソルは「+」記号に変化して、遷移のエントリを示します。マウスを WAIT_FOR_FRAME_START_0 ステートの中央にドラッグします。ステート ボックスの内側に小さいの四角形が現れ、その中心を見つけられます。マウスがステート ボックスの中心以外のとこ

ろに動くと、遷移は作成されません。やり直してください。ステート遷移にはまだ条件がないことに注意してください。

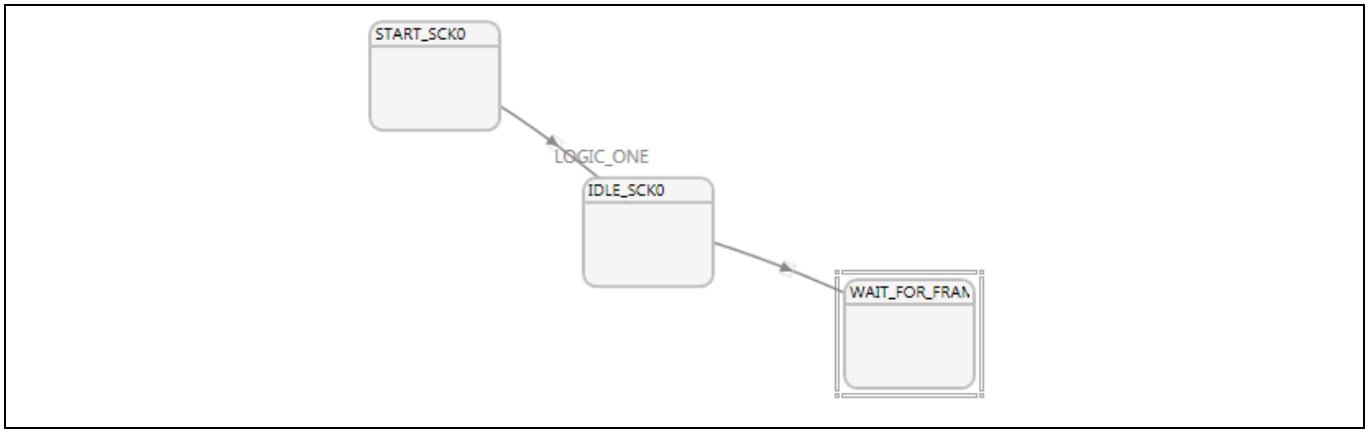


Figure 25　ステップ 1〜3 の結果

4. IDLE_SCK0 から WAIT_FOR_FRAME_START_0 への遷移方程式を編集するために、遷移ラインをダブルクリックします (**Figure 26**)。LV と FV は、ブロックダイヤグラムの名称変更された信号に対応する方程式として表示されます。FV を LOW に選択するために、ボタンと信号の選択肢を使用して方程式を作成します。「Not」ボタンをクリックすると、「Equation Entry」ウィンドウに「!」記号が現れます。そして、FV 信号を選択し、「Add」ボタンをクリックして (または FV エントリをダブルクリックする)、最終の方程式「!FV」を作成します。また、方程式入力ウィンドウに 「!FV」と入力して方程式を直接入力することもできます。遷移は **Figure 27** のようになります。

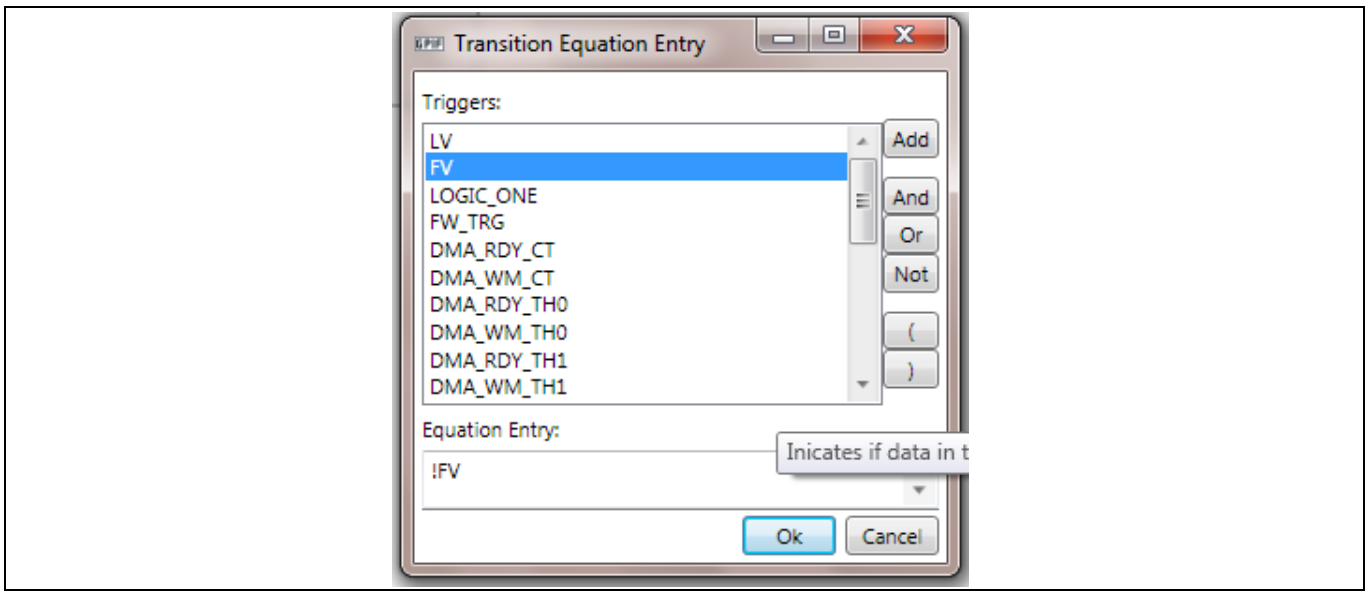


Figure 26　遷移ラインをダブルクリックしてこのウィンドウを開く

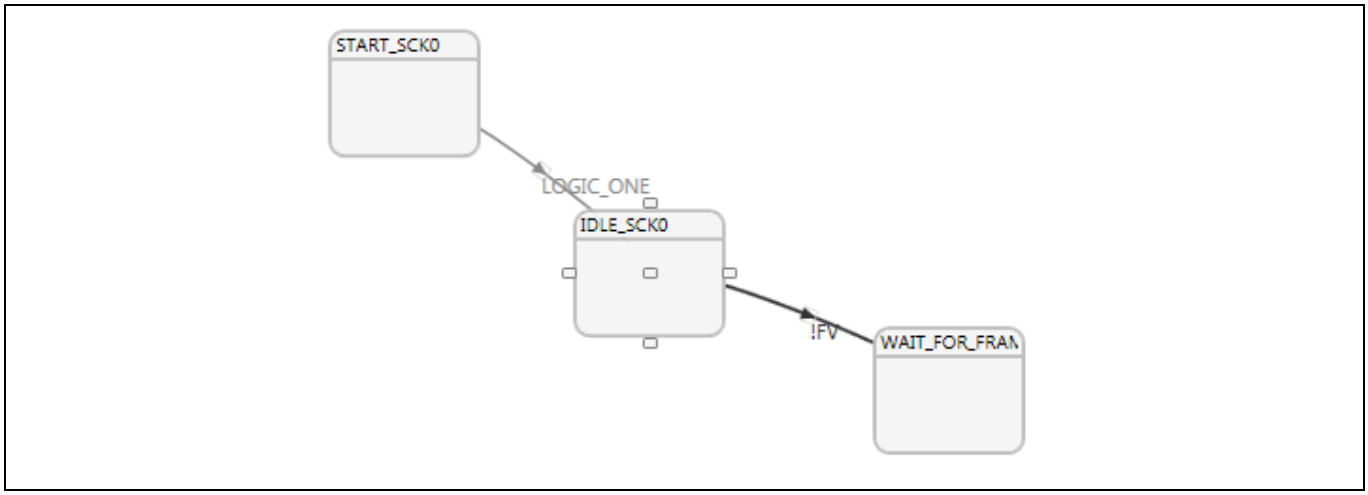


Figure 27　編集された遷移

5. インクリメントに先立ち、カウンターを初期化する必要があるため、WAIT_FOR_FRAME_START_0 ステートでは、2 つのバイト カウンターを初期化する必要があります。この設計では、DATA カウンターをソケット 0 バッファに、ADDR カウンターをソケット 1 バッファに使用することに注意してください。GPIF II Designer の上右のウィンドウ内の「Action List」ウィンドウはアクションのオプションを表示します。ステートにアクションを追加するために、その「Action List」内の名前をステート ボックスにドラッグします。LD_DATA_COUNT と LD_ADDR_COUNT のアクションを WAIT_FOR_FRAME_START_0 ステート ボックスにドラッグします。
6. アクション プロパティを編集するために、ステート ボックス内側のアクション名をダブルクリックします。**Figure 28** に示すように、両方のアクションのプロパティを編集します。「Counter mask event」チェックボックスは、カウンターが限界値に達した際の割込み要求を無効にします。

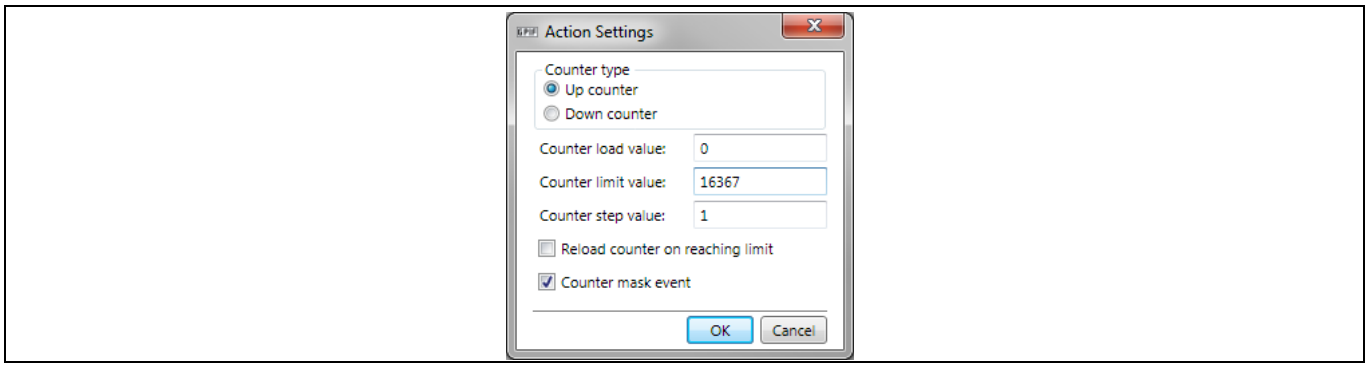


Figure 28　LD_DATA/ADDR_COUNT のアクション設定

7. 前のステップと同様に、ステート START_SCK1、IDLE_SCK1 および WAIT_FOR_FRAME_START_1 を作成します。
8. PUSH_DATA_SCK0 (イメージ センサー データをソケット 0 にプッシュする)という新しいステートを追加します。このステートは、クロックごとにイメージ センサーの 1 バイトを GPIF II インタフェースに転送して、ソケット 0 に送信します。
9. WAIT_FOR_FRAME_START_0 ステートから PUSH_DATA_SCK0 ステートへの遷移を作成します。
10. 「FV&LV」方程式を作成することで、FV と LV 両方がアサートされる時にこの遷移方程式のエントリを発生させるように変更します。ステート遷移は **Figure 29** のようになります。

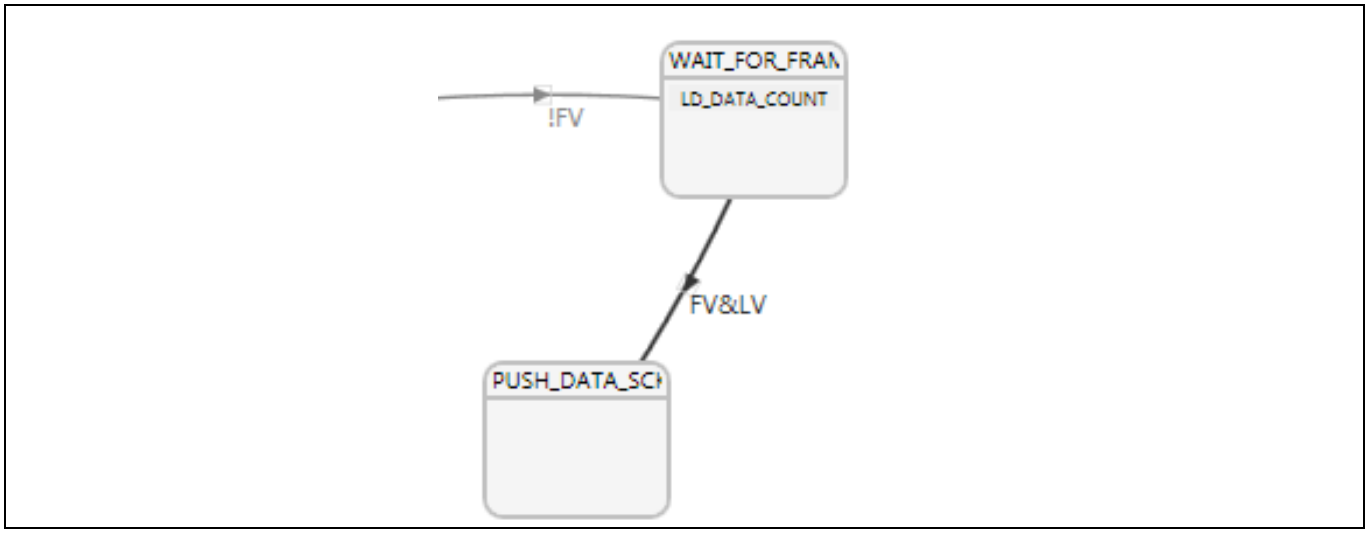


Figure 29　PUSH_DATA_SCK0 ステートおよびその遷移条件 FV&LV

11. COUNT_DATA アクションを PUSH_DATA_SCK0 ステートに追加します。これにより、PCLK の各立ち上がりエッジで DATA (ソケット 0) カウンター値がインクリメントされます。
12. IN_DATA アクションを PUSH_DATA_SCK0 ステートに追加します。このアクションは、データをデータ バスから読み出して DMA バッファに書き込みます。
13. ADDR カウンターをリロードするために LD_ADDR_COUNT アクションを PUSH_DATA_SCK0 ステートに追加します。前述のように、カウンター ロードは実際にカウンターをインクリメントするステートで実行されます。ADDR カウンターは SCK1 に転送されたバイト数をカウントします。
14. **Figure 30** に示すように、PUSH_DATA_SCK0 ステートで IN_DATA アクションのプロパティを編集します。

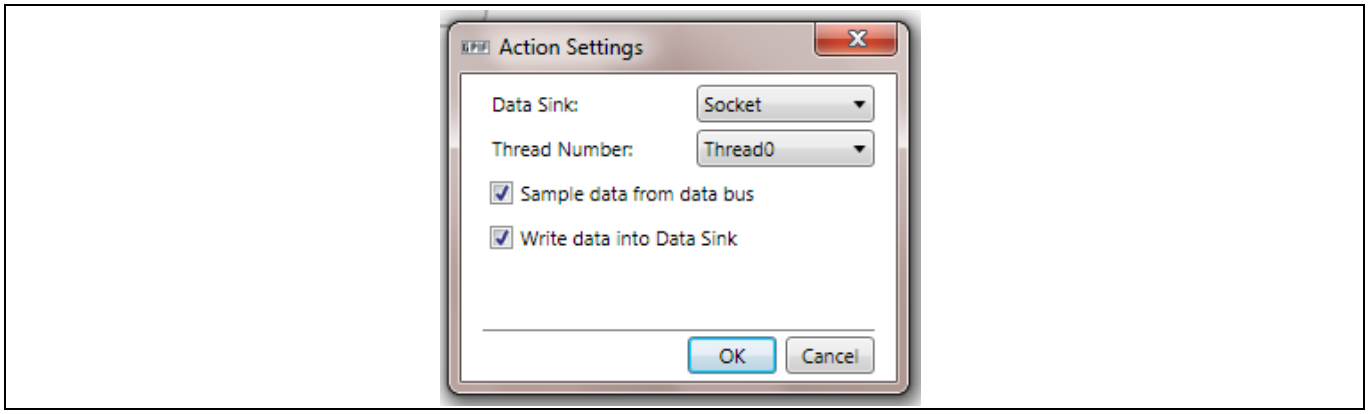


Figure 30　PUSH_DATA_SCK0 アクションの設定

15. PUSH_DATA_SCK1 と名付けられたステートを追加します。WAIT_FOR_FRAME_START_1 ステートから PUSH_DATA_SCK1 ステートへの遷移を作成し、FV と LV が両方ともアサートされた時に起きるように遷移方程式エントリを編集します。COUNT_ADDR、IN_DATA および LD_DATA_COUNT アクションを PUSH_DATA_SCK1 ステートに追加します。
16. **Figure 31** に示すように、「Thread1」を使用するために PUSH_DATA_SCK1 ステートで IN_DATA アクションのプロパティを編集します。

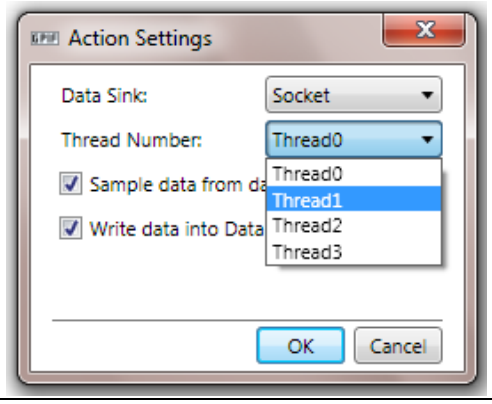


Figure 31　PUSH_DATA_SCK1 での IN_DATA アクション

17. PUSH_DATA_SCK0 ステートから PUSH_DATA_SCK1 ステートへの遷移を作成します。「LV&DATA_CNT_HIT」方程式を使用してこの遷移の方程式エントリを編集します。
18. PUSH_DATA_SCK1 ステートから PUSH_DATA_SCK0 ステートへの遷移を作成します。方向を反転させます。「LV&ADDR_CNT_HIT」方程式を使用してこの遷移の方程式エントリを編集します。

これら遷移は、アクティブなラインの間 DMA バッファ境界で GPIF スレッド間で切り替わるステート遷移を定義します。**Figure 32** にステート マシンのこの動作を示します。

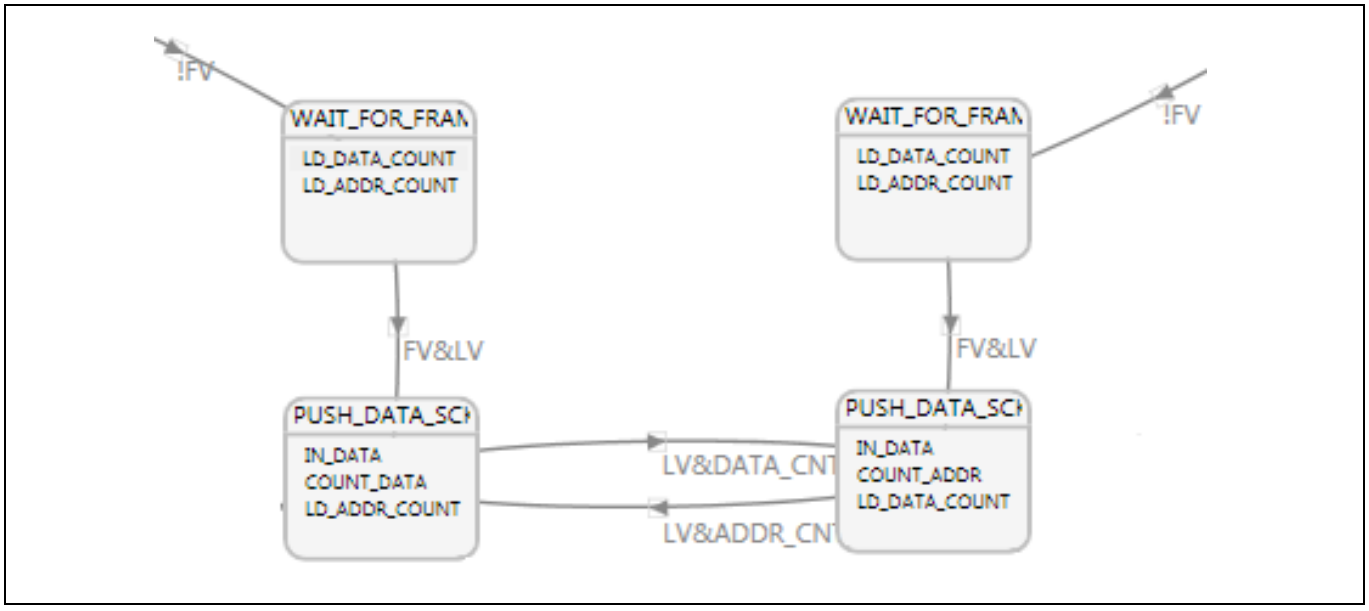


Figure 32　Figure 11 からのピンポン アクション

19. 「LINE_END_SCK0」という新しいステートを PUSH_DATA_SCK0 ステートの左側に追加します。
20. 「LINE_END_SCK1」という新しいステートを PUSH_DATA_SCK1 ステートの右側に追加します。

    イメージ センサーが次のビデオ ラインへの切り替え中に、LV 信号がデアサートされるとこれらの 2 つのステートに入ります。同様の動作は異なる GPIF スレッドを使用して交互に実行されるため、ステートはソケット 0 とソケット 1 の両側に必要とされます。

21. PUSH_DATA ステートは終了のために 3 つあり得る遷移を必要としますが、GPIF II ハードウェアはステートごとに最大 2 つの遷移を実装します。3 つの遷移の要件を処理するために、2 つのステートに

3 つの終了条件を割り当てるだけのダミー ステートを作成します。これをデモする目的として、**Figure 33** に条件 1、2 または 3 に基づいたステート A からステート B、C または D への遷移を示します。

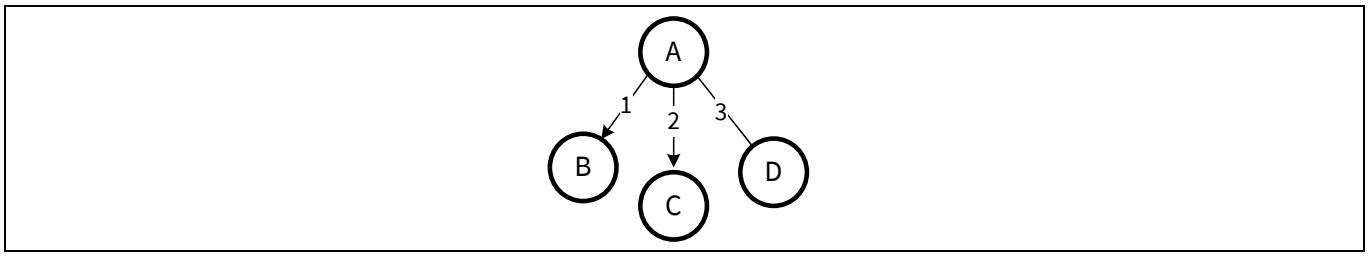


Figure 33　ステート A は終了条件 1、2、3 を必要とする

22. **Figure 34** に示すように、GPIF II の互換性を確保するためにダミー ステート A2 を追加します。

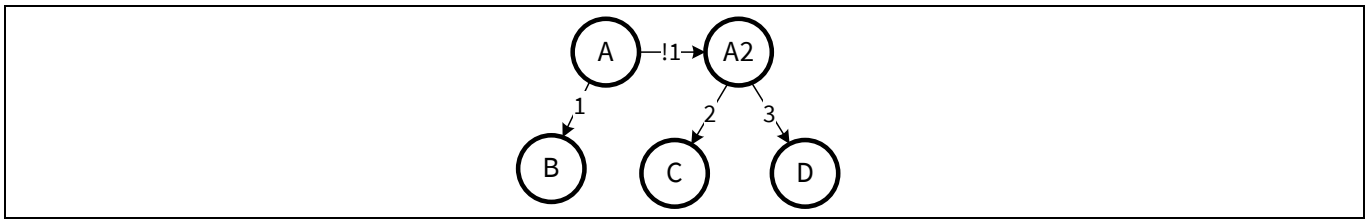


Figure 34　GPIF II の互換性のためにダミー ステート A2 を追加

23. ダミー ステートとして機能する LINE_END ステートを作成します。
24. 「(not LV)」遷移方程式を使って PUSH_DATA_SCK0 ステートから LINE_END_SCK0 ステートへの遷移を作成します。
25. 「(not LV)」遷移方程式を使って PUSH_DATA_SCK1 ステートから LINE_END_SCK1 ステートへの遷移を作成します。
26. 「WAIT_TO_FILL_SCK0」という新しいステートを LINE_END_SCK0 ステートの下に追加します。
27. 「WAIT_TO_FILL_SCK1」という新しいステートを LINE_END_SCK1 ステートの下に追加します。

    DMA バッファが一杯でなく、ライン有効信号がデアサートされる時に、これら 2 つのステートに入ります。

28. **Figure 35** に示すように、遷移方程式「(not DATA_CNT_HIT)」を使って LINE_END_SCK0 ステートから WAIT_TO_FILL_SCK0 ステートへの遷移を作成します。

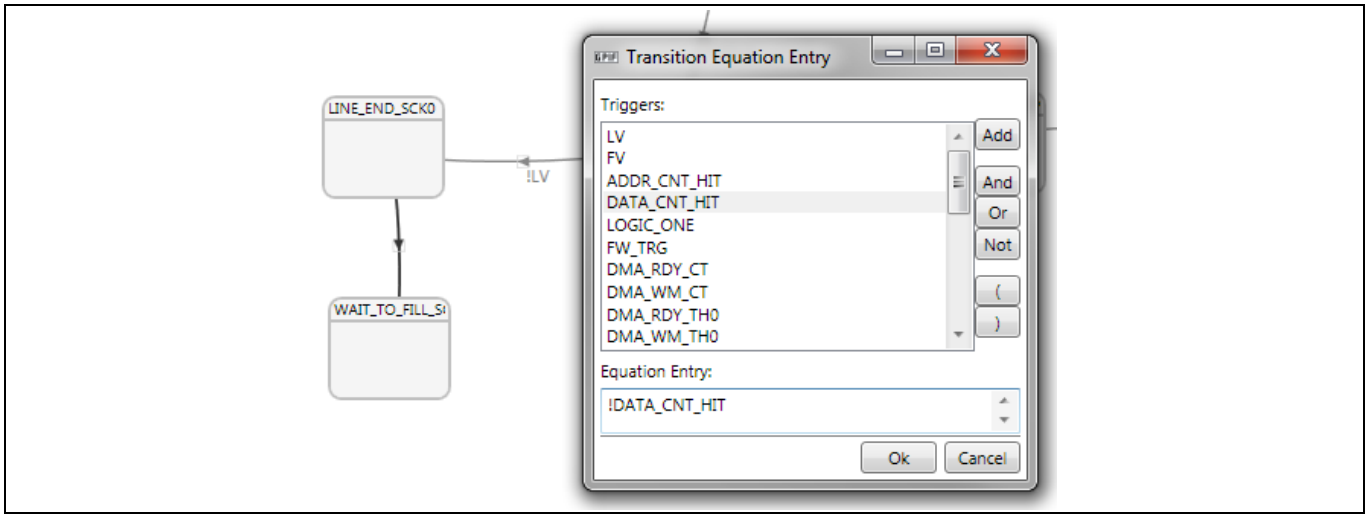


Figure 35　DATA_CNT_HIT はカウンターがそのプログラムされた限界値に達したことを示す

29. 「LV」方程式を使って「WAIT_TO_FILL_SCK0」ステートから「PUSH_DATA_SCK0」ステートへの逆の遷移を作成します。データ転送は、ラインが有効になるとすぐに同じソケットで再開します。
30. 「(not ADDR_CNT_HIT)」遷移方程式を使って「LINE_END_SCK1」ステートから「WAIT_TO_FILL_SCK1」ステートへの遷移を作成します。
31. 「LV」方程式を使って「WAIT_TO_FILL_SCK1」ステートから「PUSH_DATA_SCK1」ステートへの逆の遷移を作成します。
32. 「WAIT_FULL_SCK0_NEXT_SCK1」という新しいステートを「PUSH_DATA_SCK0」ステートの下に追加します。
33. 「WAIT_FULL_SCK1_NEXT_SCK0」という新しいステートを「PUSH_DATA_SCK1」ステートの下に追加します。
34. これら 2 つのステート (WAIT_FULL_) の間、イメージ センサーは DMA バッファの境界でラインを切り替えます。したがって、次のバイト転送は現時点で無効な GPIF スレッドを使用しなければなりません。
35. 「DATA_CNT_HIT」方程式を使って「LINE_END_SCK0」ステートから「WAIT_FULL_SCK0_NEXT_SCK1」ステートへの遷移を作成します。ステート マシンのこの部分を **Figure 36** に示します。

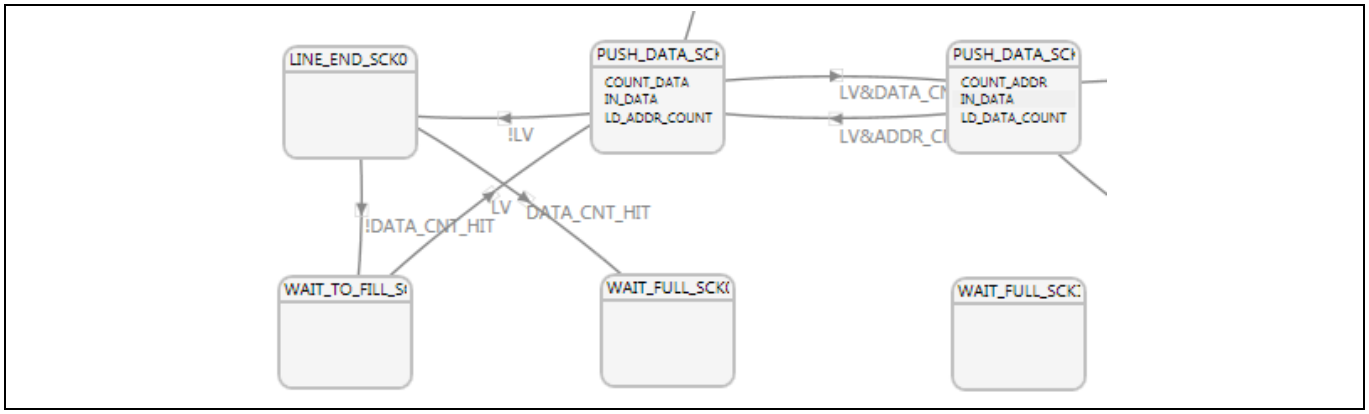


Figure 36　WAIT_FULL ステートを追加

36. 「ADDR_CNT_HIT」方程式を使って「LINE_END_SCK1」ステートから「WAIT_FULL_SCK1_NEXT_SCK0」ステートへの遷移を作成します。
37. 「LV」方程式を使って「WAIT_FULL_SCK0_NEXT_SCK1」ステートから「PUSH_DATA_SCK1」ステートへの遷移を作成します。これにより、ステート図に交差リンクが作成されます。
38. 「LV」方程式を使って「WAIT_FULL_SCK1_NEXT_SCK0」ステートから「PUSH_DATA_SCK0」ステートへの遷移を作成します。ここで、ステート マシンは **Figure 37** のようになります。

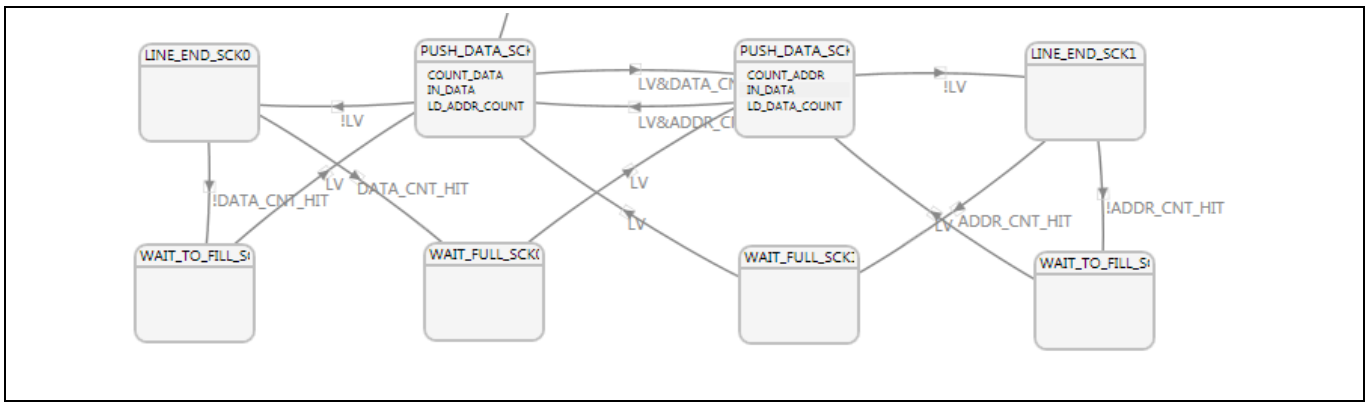


Figure 37　DMA バッファが一杯になっている間待機

39. 「PARTIAL_BUF_IN_SCK0」という新しいステートを「WAIT_TO_FILL_SCK0」ステートの下に追加します。
40. 「PARTIAL_BUF_IN_SCK1」という新しいステートを「WAIT_TO_FILL_SCK1」ステートの下に追加します。

    各 PARTIAL_BUF_ステートは、最後の DMA バッファが完全に満たされていないフレームの終了を示します。CPU は、GPIF II が生成した割込み要求に応答する時に手動でこの一杯にならない DMA バッファをコミットする必要があります。

41. 「FULL_BUF_IN_SCK0」という新しいステートを「WAIT_FULL_SCK0_NEXT_SCK1」ステートの下に追加します。
42. 「FULL_BUF_IN_SCK1」という新しいステートを「WAIT_FULL_SCK1_NEXT_SCK0」ステートの下に追加します。

    各 FULL_BUF_ステートは、フレーム データの終了が対応する GPIF スレッドに関連付けられた DMA バッファの最後のバイトであることを示します。GPIF II ハードウェアはこの一杯になった DMA バッファのコミットを担当するため、これ以上のアクションはアプリケーション固有になります。

43. 「not FV」方程式を使って「WAIT_TO_FILL_SCK0」ステートから「PARTIAL_BUF_IN_SCK0」ステートへの遷移を作成します。
44. 「not FV」方程式を使って「WAIT_TO_FILL_SCK1」ステートから「PARTIAL_BUF_IN_SCK1」ステートへの遷移を作成します。
45. 「not FV」方程式を使って「WAIT_FULL_SCK0_NEXT_SCK1」ステートから「FULL_BUF_IN_SCK0」ステートへの遷移を作成します。
46. 「not FV」方程式を使って「WAIT_FULL_SCK1_NEXT_SCK0」ステートから「FULL_BUF_IN_SCK1」ステートへの遷移を作成します。
47. 「Intr_CPU」アクションを「PARTIAL_BUF_IN_SCK0」、「PARTIAL_BUF_IN_SCK1」、「FULL_BUF_IN_SCK0」および「FULL_BUF_IN_SCK1」ステートに追加します。
48. 「FRAME_END_SCK_0」という新しいステートを追加し、「ファームウェア トリガー アサート (FW_TRIG)」を遷移条件としてステート「PARTIAL_BUF_IN_SCK0」および「FULL_BUF_IN_SCK0」から

の遷移を追加し、そして「ファームウェア トリガー デアサート (!FW_TRIG)」を遷移条件としてステート「FRAME_END_SCK_0」から「IDLE_1」への遷移を追加します。

49. 前のステップと同様に、「FRAME_END_SCK_1」ステートを作成し、「ファームウェア トリガー アサート (FW_TRIG)」を遷移条件としてステート「PARTIAL_BUF_IN_SCK1」と「FULL_BUF_IN_SCK1」からの遷移を追加し、そして「ファームウェア トリガー デアサート (!FW_TRIG)」を遷移条件としてステート「FRAME_END_SCK_1」から「IDLE_0」への遷移を追加します。

    最終のステート マシンは **Figure 39** のようになります。これを **Figure 13** と比較すると、唯一の違いはステートごとに最大 2 つの遷移の要件に適合するための PUSH_DATA ステートの追加です。

50. 「File-Save Project As」を選択し、プロジェクトを保存します。
51. **Figure 38** に強調表示するビルド アイコンを使用し、プロジェクトをビルドします。プロジェクトの出力ウィンドウはビルドの状態を示すところです。

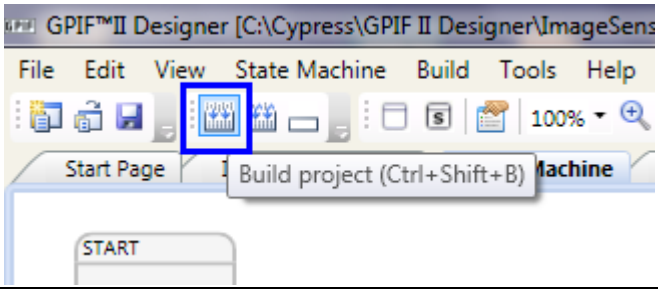


Figure 38　ビルド ボタン

52. プロジェクトの出力を確認します。これは、プロジェクトのディレクトリ配下のヘッダー ファイル *cyfxgpif2config.h* として表示されます。このヘッダー ファイルを読むと、GPIF Designer II が内部 GPIF II 設定のアレイを作成したことが分かります。EZ-USB™ FX3 ファームウェアはこれらの設定を使用して GPIF II を設定し、そのステート マシンを定義します。このファイルを直接編集せず、GPIF II Designer に任せてください。

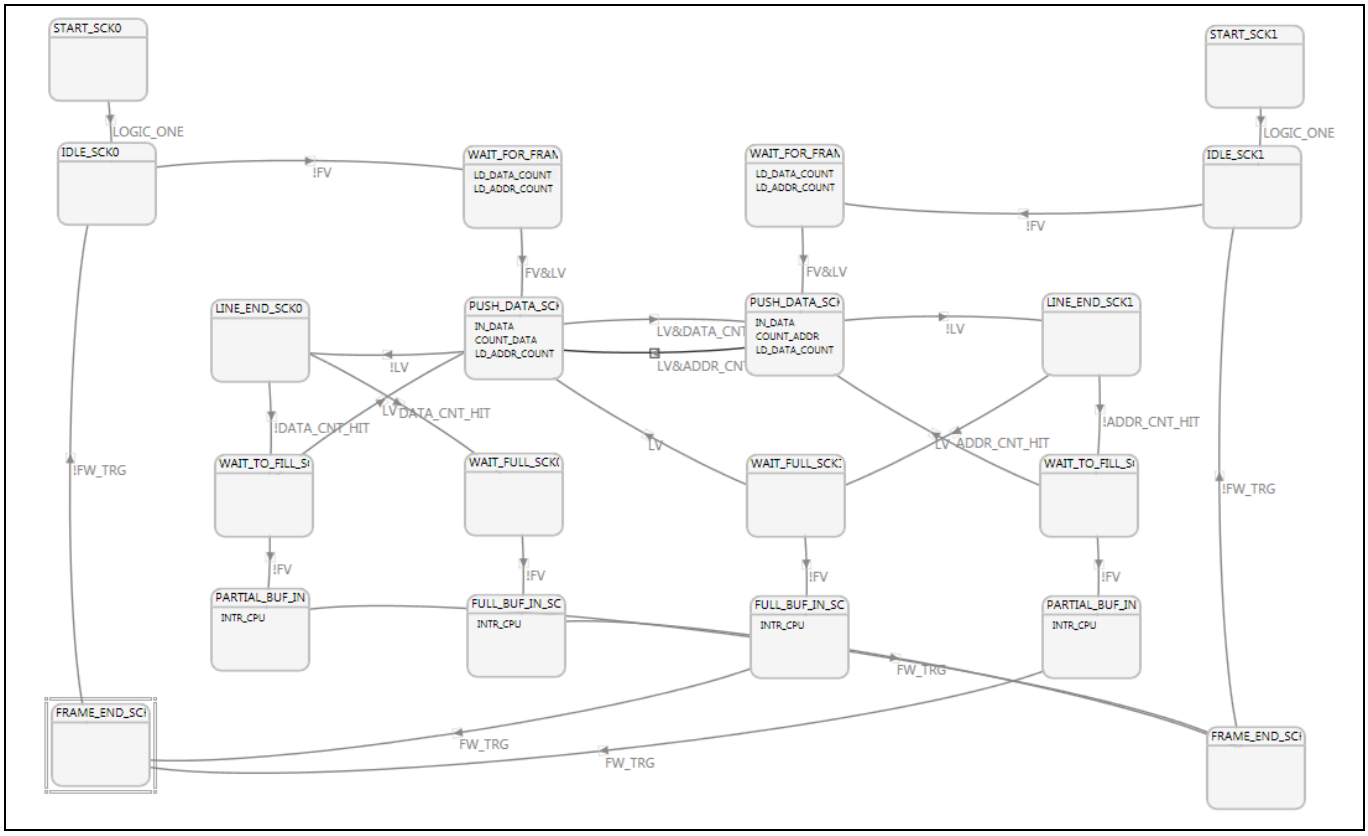


Figure 39　最終ステート マシンの図

3.6.5　GPIF II インタフェースの詳細変更

本節では、インタフェース設定の変更をしたい場合にその方法を説明します。例えば、イメージ センサー/ASIC が 16 ビット幅のデータ バスを有する場合、GPIF II インタフェースをそのデータ バスに適合するように変更する必要があります。これを実施するには次の手順を取ります:

1. GPIF II Designer で **ImageSensorInterface.cyfx** プロジェクトを開きます(このプロジェクトは直接コンパイルされないことがあります)。
2. File->Save Project As に移動します。
3. 表示されるダイアログ ボックスにおいてプロジェクトに適切な名前を付け、プロジェクトを便利な場所に保存します。
4. 現時点で開いているプロジェクトを閉じます (File->Close Project を選択します)。
5. ステップ 3 で保存したプロジェクトを開きます。
6. **Interface Definition** タブに移動し、**Address/Data Bus Usage** の設定で **16 Bit** のオプションを選択します。
7. **State Machine** タブに移動します。
8. ステート マシンのキャンバスで、WAIT_FOR_FRAME_START ステート内の LD_DATA_COUNT アクションをダブルクリックします。カウンター限界値を 8183 に変更します。
9. LD_ADDR_COUNT アクションも同様にします。
10. プロジェクトを保存します。
11. プロジェクトをビルドします。
12. 新規作成した *cyfxgpif2config.h* ヘッダー ファイルをステップ 3 で選択した場所からファームウェア プロジェクトのディレクトリにコピーします。既存の *cyfxgpif2config.h* ファイルがある場合、それを上

書きします。添付ソースの zip ファイルで、ファームウェア プロジェクトのディレクトリ *cyfxuvc_an75779* を探します。

Note: GPIF II バス幅を 32 ビットに変更する場合、ファームウェア内の iomatrix コンフィギュレーションでは isDQ32Bit パラメータが CyTrue に設定されたことをご確認ください。

次の節ではデータをストリームする 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。

4 DMA システムの設定

プロセッサ インタフェース ブロック (PIB) の一部である GPIF II ブロックは、32 データ ビットで 100MHz (400Mbps) まで実行できます。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複数の GPIF スレッドを使用します (**3.3 節**で説明しました)。このアプリケーションでは、4 つの GPIF スレッドのうちの 2 つが使用されます。ソケットと GPIF スレッドのデフォルトのマッピング (**Figure 40**) 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 は GPIF スレッド 1 に接続されています。GPIF スレッドの切り替えは、前のセクションで設計された GPIF II ステート マシンで行われます。

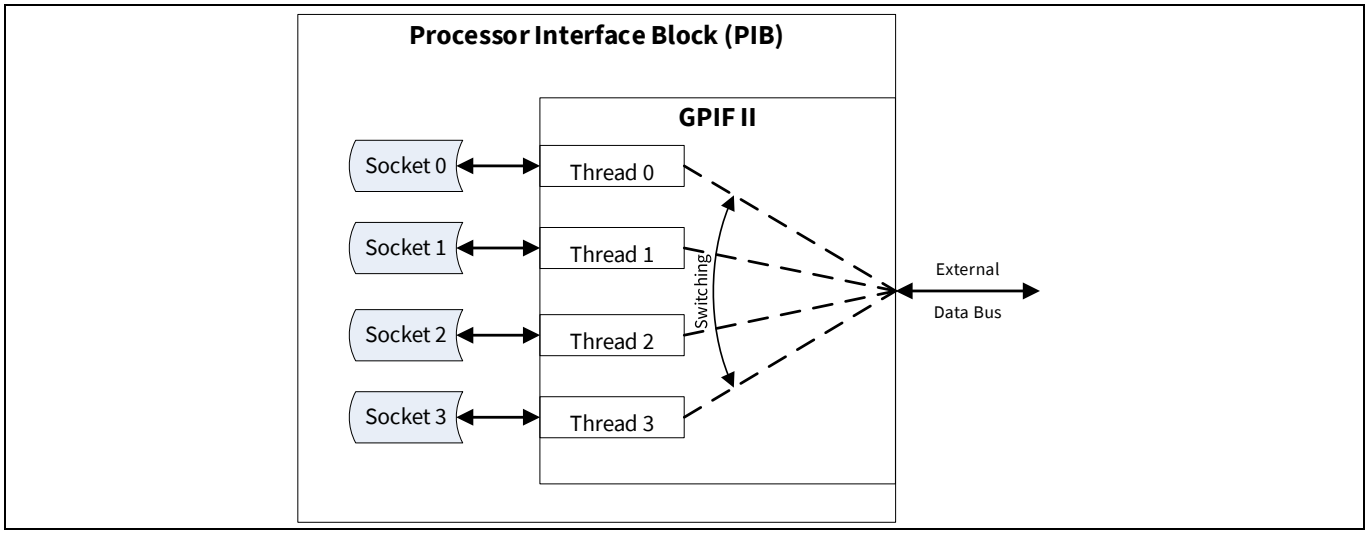


Figure 40 GPIF II ソケット/スレッドのデフォルト マッピング

DMA 転送を理解するために、次の 4 つの図でソケットの概念をさらに詳しく説明します。**Figure 41** に、2 つの主なソケット属性であるリンク リストおよびデータ ルーターを示します。

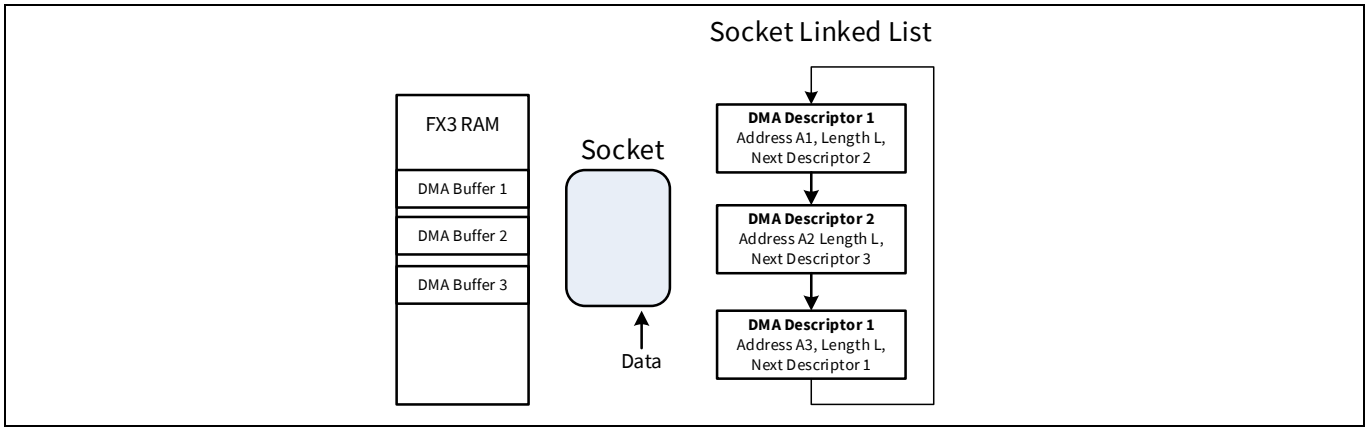


Figure 41 ソケットが DMA ディスクリプタのリストに従ってデータを転送

ソケット リンク リストはメイン メモリ内のデータ構造体の組であり、 DMA ディスクリプタと呼ばれています。それぞれのディスクリプタは、DMA バッファのアドレスと長さおよび次の 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、1 度に 1 つの DMA ディスクリプタを読み出し、ディスクリプタ内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、任意の数の DMA バッファをメモリ内の任意の場所に作成し、自動的に連結できるため、ソケットを汎用的に使用できます。例えば **Figure 41** のソケットは、繰返しループで DMA ディスクリプタを読み出します。

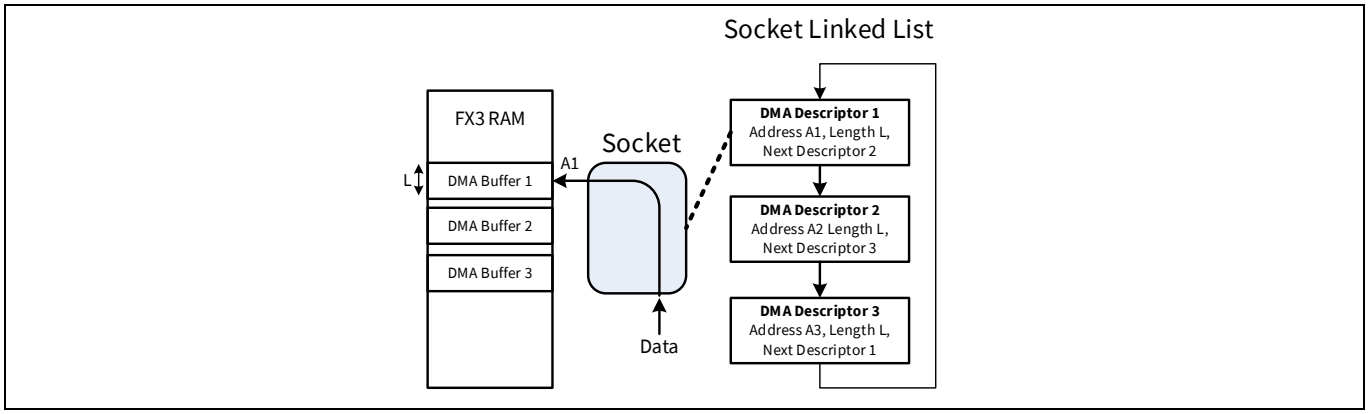


Figure 42　DMA ディスクリプタ 1 で動作するソケット

Figure 42 では、ソケットは DMA ディスクリプタ 1 をロードします。この DMA ディスクリプタ 1 の指示でソケットは L バイトが転送される時まで A1 から始まるバイトを転送します。ソケットは L バイトの転送を完了すると、DMA ディスクリプタ 2 を読み出し、アドレス A2 と長さ L で前と同様の処理を繰り返します (**Figure 43**)。

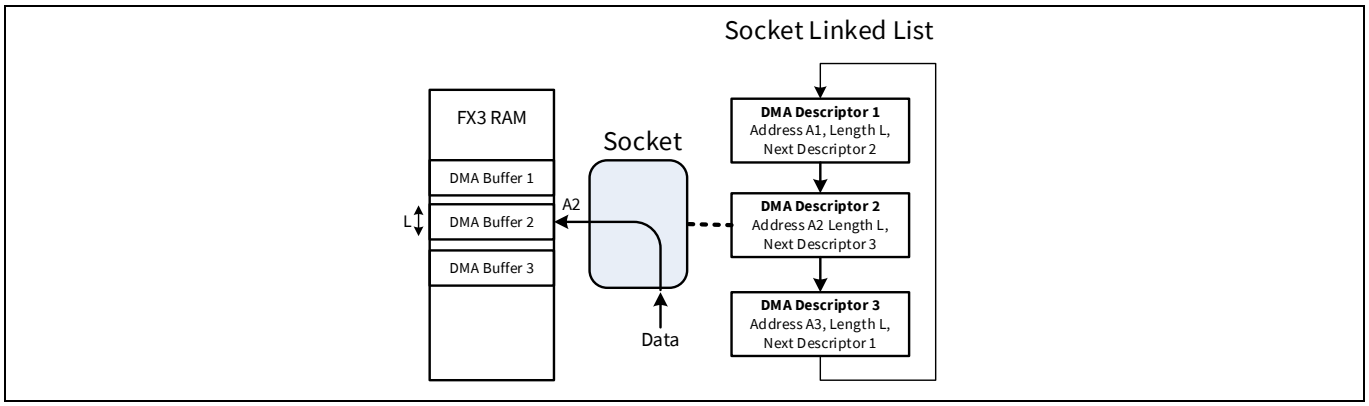


Figure 43　DMA ディスクリプタ 2 で動作するソケット

Figure 44 では、ソケットは 3 番目の DMA ディスクリプタを読み出し、A3 から始まるデータを転送します。L バイトを転送した後、シーケンスは DMA ディスクリプタ 1 から繰り返します。

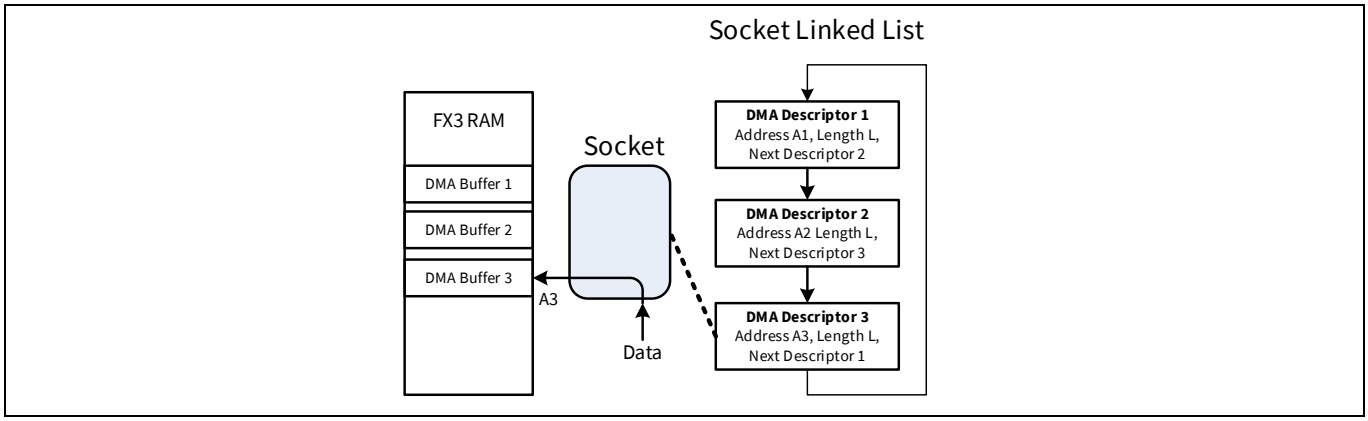


Figure 44　DMA ディスクリプタ 3 で動作するソケット

Figure 45 に、DMA データ転送を詳しく説明します。この例は長さ L の DMA バッファを 3 つ使用しています。これらのバッファは環状ループで連結されています。EZ-USB™ FX3 メモリ アドレスは左側にあります。青色矢印はソケット 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。赤色矢印は得られたデータ転送を示します。以下の手順は、データを内部 DMA バッファに転送するソケット 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 から始まる 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。

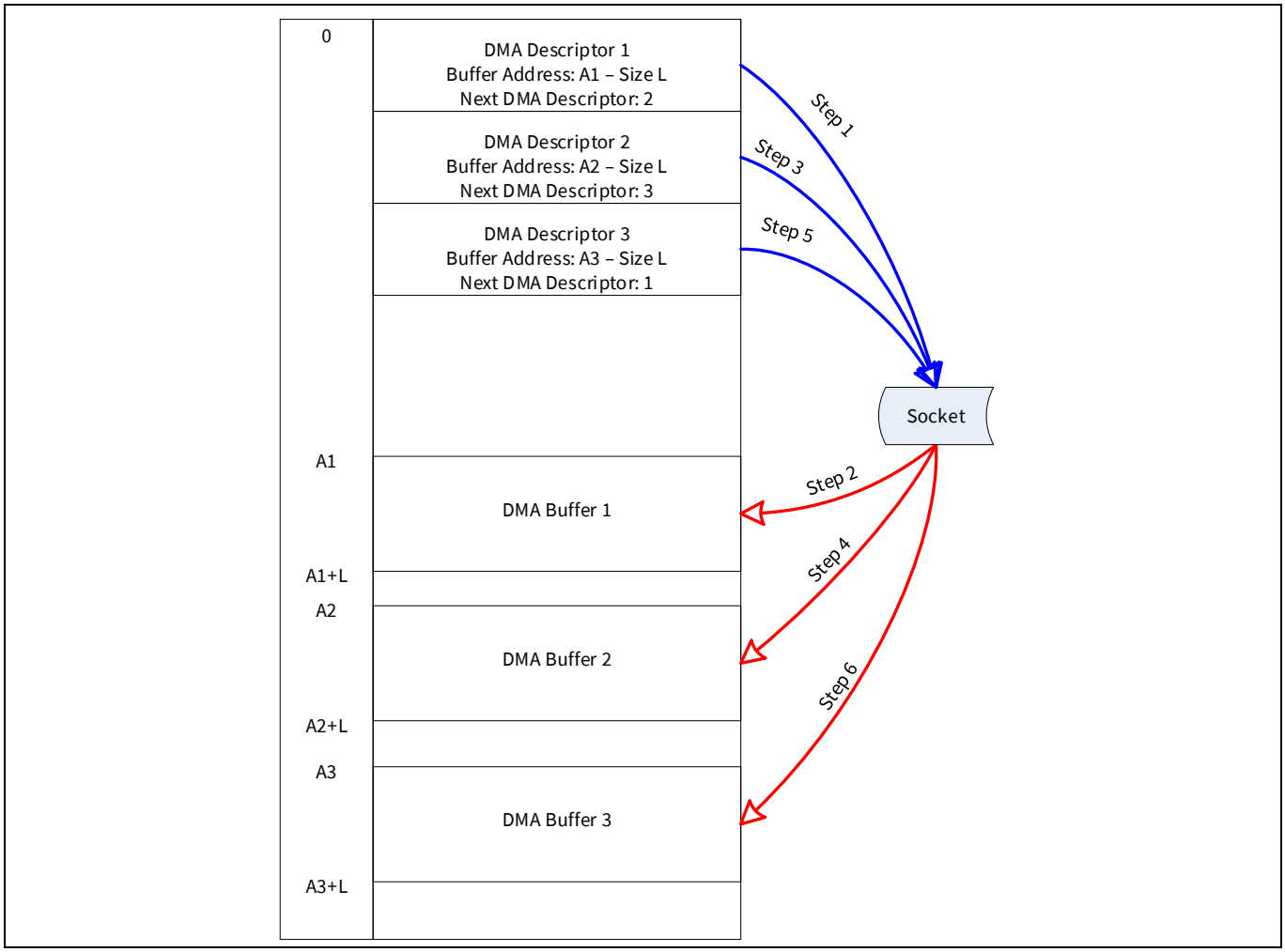


Figure 45　DMA 転送の例

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 から始まる 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 から始まる 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 に進みます。

この単純なスキームはカメラ アプリケーションで問題があります。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には通常 1 マイクロ秒がかかります。この転送ではビデオ ライン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 バッファの切り替えはビデオ ラインが非アクティブ (LV=0) になった時と同時に発生します。しかし、この方法は例えばビデオ解像度が変更された場合に対応する柔軟性を持っていません。

DMA システムの設定

DMA バッファ サイズをライン サイズと正確に等しく設定することは、BULK 転送の USB 3.0 最大バースト レートを利用しないため、良い解決策ではありません。USB3.0 は BULK エンドポイント上で 1024 バイトの最大 16 バーストを可能にします。これは DMA バッファ サイズが 16KB に設定されている理由です。

より良い解決策は、ソケットが 1 クロック サイクル以内で待ち時間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。デュアル ソケットを使用したデータ転送は、**Figure 46** で実行ステップに番号付けし説明します。ソケット 0 とソケット 1 の DMA バッファへのアクセスは、それぞれ赤色矢印と緑色矢印 (個別のソケットのデータ経路) で区別されています。各ステップの「a」と「b」の部分は同時に起きます。このハードウェアの並行動作により、DMA ディスクリプタ読み出しデッド タイムは無くなり、GPIF II がデータを内部メモリに連続してストリームできます。これらのステップは **Figure 14** の「Step」ラインに対応しています。

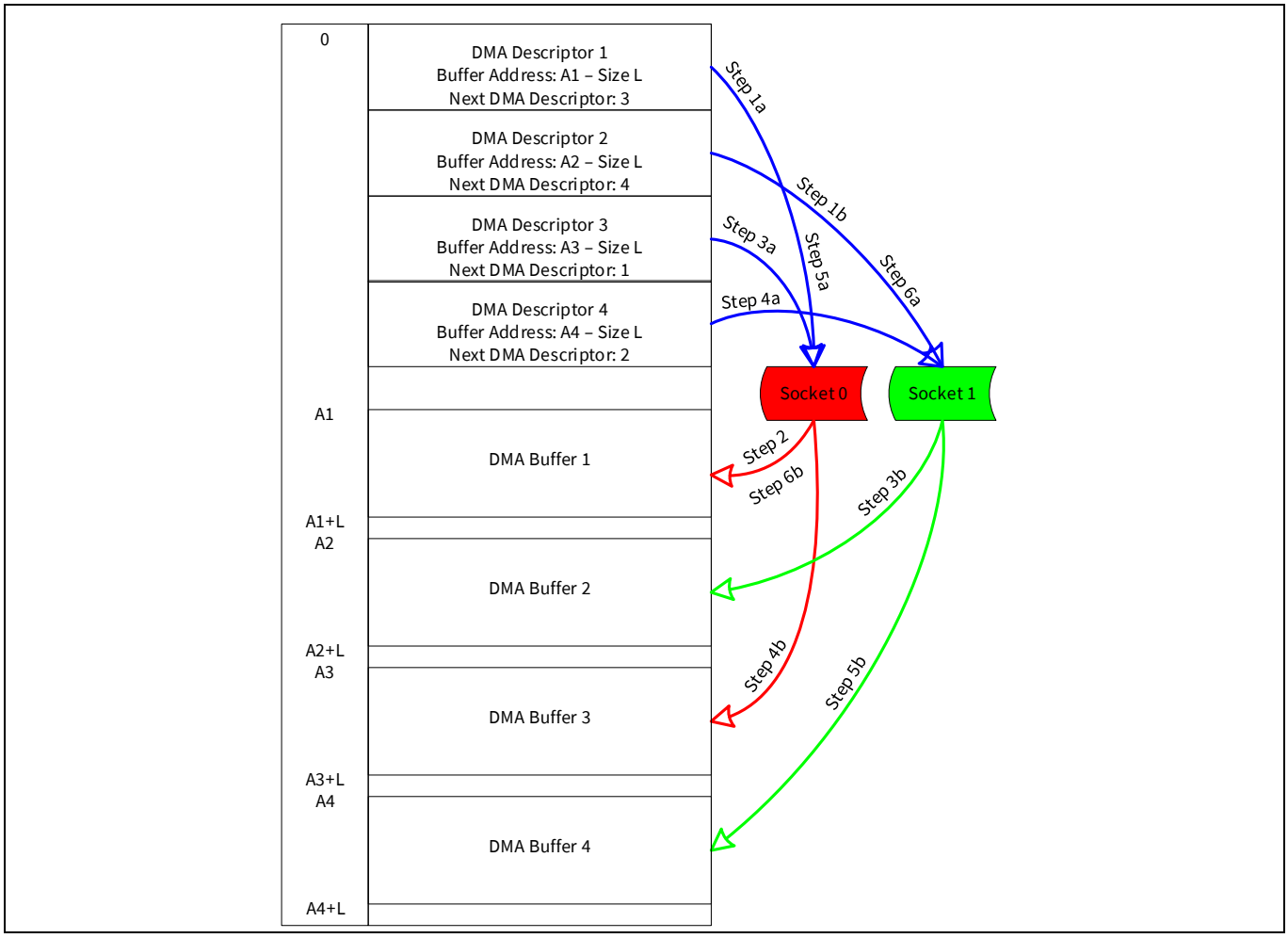


Figure 46　デュアル ソケットによるシームレスな転送

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。この転送が終わると、ステップ 3 に進みます。

ステップ 3: GPIF II は GPIF スレッドを切り替えます。したがって、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完了した時には、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: この時点で、GPIF II は元の GPIF スレッドに戻します。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、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 の長さ L のデータ転送を終わると、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 に転送します。同時に、ソケット 0 は DMA ディスクリプタ 1 をロードし、DMA バッファ 1 へのデータ転送の準備をします。ステップ 5a は、ソケット 1 を初期化せずにデータを同時に転送すること以外は、ステップ 1a に似ていることに注意してください。

ステップ 6: GPIF II は再びソケットを切り替え、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 バッファが UIB コンシューマ ソケットによって使い果たされてから、この時点では空の状態であることを想定しています。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、DMA バッファ 2 へのデータ転送の準備を完了します。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い、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い場合、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、次のフレームに伝播する可能性があります。このため、クリーンアップ メカニズムは、すべてのフレームの終わりに必要とされます。このメカニズムは「クリーン アップ」節で説明しています。

Figure 13 のフローチャートに従えば、フレーム転送は 4 つの可能な状態で終了します:

- ソケット 0 が一杯になった DMA バッファを転送した
- ソケット 1 が一杯になった DMA バッファを転送した
- ソケット 0 が一杯になっていない DMA バッファを転送した
- ソケット 1 が一杯になっていない DMA バッファを転送した

一杯になっていない DMA バッファの場合、CPU はそれを USB コンシューマにコミットする必要があります。

DMA チャネルは *uvc.c* ファイル内の **CyFxUVCApplnInit** 関数を使用して初期化されます。DMA チャネル設定の詳細は **CyFxUVCApplnInit** 関数の 「dmaMultiConfig」構造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、USB 3.0 ホストへデータをストリームする USB エンドポイントは、1024 バイトの BULK エンドポイント上で 16 のバーストを可能にするように構成されます。このために、エンドポイントの定数を「CY_FX_EP_BULK_VIDEO」に設定し、「CyU3PSetEpConfig」関数に渡された「endPointConfig」構造を使用します。

4.1 DMA バッファについて

本節では EZ-USB™ FX3 DMA バッファについて、作成およびこのアプリケーションでの使用方法の概要を説明します。

- DMA チャネルの不可欠な部分を **3.3** 節に説明しています。
- 本アプリケーションでは、GPIF II ユニットがプロデューサであり、EZ-USB™ FX3 USB ユニットがコンシューマです。

DMA システムの設定

- 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 ブロックで GPIF スレッド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 バッファで書き込み動作の準備ができているかを確認するためにチェックします。プロデューサ ソケットは、DMA バッファが空であることが確認できると、EZ-USB™ FX3 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のために 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。この動作は「バッファ ラップアップ」または単に「ラップアップ」と呼ばれます。DMA ユニットはその後、EZ-USB™ FX3 RAM に DMA バッファを転送するプロデューサ ソケットは DMA バッファをプロデュースしたと言います。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べきです。これがステート図の FV=0 テストの理由です。
- DMA バッファが完全に一杯になると、フレーム中に繰返し同じように、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、EZ-USB™ FX3 RAM にコミットし、空の DMA バッファに切り替え、ビデオ データ ストリームの書き込みを続けます。
- コンシューマ 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 バッファで読み出し動作の準備ができているかを確認するためにチェックします。コンシューマ ソケットは、DMA バッファがコミットされたことを検出した場合、EZ-USB™ FX3 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のために 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 バッファを使い果たしたと言われています。
- プロデューサ ソケットとコンシューマ ソケットが同時に同じ DMA ディスクリプタを使用している場合、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 ヘッダー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 ソケットによってロードされた DMA ディスクリプタは、コンシューマ ソケットによってロードされた DMA ディスクリプタが指す対応 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 ソケットが使用する異なる DMA ディスクリプタに応じて、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。これが DMA チャネルの実装が「手動 DMA」チャネルと呼ばれている理由です。
- DMA バッファが GPIF II ブロックによって生成された後、CPU は割込みを介して通知されます。CPU はその後、ヘッダー情報を追加し、コンシューマ (USB インタフェース ブロック) に DMA バッファをコミットします。
- GPIF II 側では、自動的に DMA バッファ内のビデオ データはラップアップされ、フレーム内の最後の DMA バッファを除くすべてのバッファは EZ-USB™ FX3 RAM にコミットされます。CPU の介入は必要ありません。
- 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、EZ-USB™ FX3 RAM にそれをコミットします。これは「強制的なラップアップ」と呼ばれます。

5 EZ-USB™ FX3 のファームウェア

本アプリケーション ノートで提供されているサンプル ファームウェアは、ソース zip ファイルの *cyfxuvc_an75779* フォルダの下にあります。Eclipse ワークスペースでこのファームウェア プロジェクトをインポートするためには **AN75705, Getting Started with EZ-USB™ FX3** を参照してください。このサンプル ファームウェアはコンパイルし、列挙しますが、イメージ センサーからビデオをストリームするためには、ユーザーは *sensor.c* と *sensor.h* ファイルを使って特定のイメージ センサー用にファームウェアを調整しなければなりません。

本アプリケーション ノートの **6 節**で説明したように ON Semiconductor (Aptina) センサー基板を使用する場合、インフィニオンが提供した *sensor.c* および *sensor.h* ファイルはこの特定のセンサーに対応します。プリコンパイルされたロード イメージは EZ-USB™ FX3-ON Semiconductor Sensor コンフィギュレーションを試すためにプロジェクト内に提供されています (**7 節**)。**Table 8** に、コード モジュールおよびそれぞれに実装する関数をまとめます。

Table 8 サンプル プロジェクト ファイル

ファイル	説明
sensor.c	I²C 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の SensorWrite2B、SensorWrite、SensorRead2B および SensorRead 関数を定義。これらの関数はイメージ センサーの I²C バス上のレジスタ アドレスが 16 ビット幅であることを前提にする イメージ センサーのリセット ラインを制御するための SensorReset 関数と、I²C の接続を確認し、(SensorScaling_HD720p_30fps 関数を使って) イメージ センサーをデフォルトのストリーム モードに設定するための SensorInit 関数を定義 イメージ センサーを所望のストリーム モードに設定するための SensorScaling_VGA と SensorScaling_HD720p_30fps 関数を定義。 センサー輝度調整レジスタの輝度値を読み書きするための SensorGetBrightness と SensorSetBrightness 関数を定義。
sensor.h	イメージ センサー用に使用する定数 (I2C スレーブ アドレスおよびリセット用の GPIO 番号) を含む。このファイルでイメージ センサーの I2C スレーブ アドレスを定義する必要がある sensor.c で定義されたすべての関数の宣言を含む
camera_ptzcontrol.c	カメラの PTZ 値を読み書きするための関数を定義。これらはプレースホルダ関数であり、イメージ センサー固有の設定コマンドで実行する必要がある uvc.h の「#define UVC_PTZ_SUPPORT」行のコメントを外し、PTZ 制御のコード プレースホルダを有効にする
camera_ptzcontrol.h	PTZ 制御の実装に使用される定数を含む camera_ptzcontrol.c で定義されたすべての関数の宣言を含む
cyfxtx.c	変更は不要。本アプリケーション ノートの添付プロジェクトでこのファイルをそのまま使用 これは RTOS と EZ-USB™ FX3 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 FX3 API ライブラリがメモリ管理に使用する関数を含む
cyfxgpif2config.h	GPIF II Designer ツールで生成されるヘッダーファイル。変更は不要。インタフェースを変更する必要がある場合、新しいヘッダーファイルは GPIF II Designer ツールから生成されなければならない。このファイルを新しいファイルで置き換える必要がある GPIF II ステート マシンを開始、実行するための、uvc.c ファイル内の API 呼び出しに渡された構造体と定数を含む

ファイル	説明
uvc.c	UVC アプリケーション用のメイン ソース ファイル。輝度と PTZ 制御機能以外の制御機能をサポートする、または異なるビデオ ストリーム モードのサポートを追加するためにコードを修正する際にこのファイルを変更する必要がある 以下の関数を含む: Main: EZ-USB™ FX3 デバイスを初期化し、キャッシュをセットアップし、EZ-USB™ FX3 I/O を設定し、RTOS カーネルを初期化 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 UVCAppThread_Entry: 最初のアプリケーション スレッド関数であり、EZ-USB™ FX3 の内部ブロック用の初期化関数を呼び出し、デバイスを列挙し、ビデオ データの転送と USB 一時停止を処理 UVCAppEP0Thread_Entry: 2 番目のアプリケーション スレッド関数であり、UVC 固有の要求が受け取られたことを示すイベントを待機し、これら要求を処理するために対応関数を呼び出す UVCHandleProcessingUnitRqts: プロセッシング ユニットの機能を対象とした VC の要求を処理 UVCHandleCameraTerminalRqts: 入力端子の機能を対象とした VC の要求を処理 UVCHandleExtensionUnitRqts: 拡張ユニットの機能を対象とした VC の要求を処理 UVCHandleInterfaceCtrlRqts: いかなる端子やユニットも対象外とした一般的な VC の要求を処理 UVCHandleVideoStreamingRqts: ストリーム モードを変更する VS 要求を処理 CyFxUVCApplnDebugInit: デバッグ メッセージのプリント用に FX3 の UART ブロックを初期化 CyFxUVCApplnI2CInit: イメージ センサー設定用に EZ-USB™ FX3 の I2C ブロックを初期化 CyFxUVCApplnInit: 列挙用に FX3 の GPIO ブロック、プロセッサ ブロック (GPIF II がプロセッサ ブロックの一部)、センサー (コンフィギュレーションを 720p 30fps にセット) および USB ブロックを、USB 転送用にエンドポイント コンフィギュレーション メモリを、GPIF II から USB へのデータ転送用に DMA チャネル コンフィギュレーションを初期化 CyFxUvcAppGpifInit:GPIF II ステートマシンを初期化し、起動 CyFxGpifCB: GPIF II ステートマシンから生成された CPU 割込みを処理 CyFxUvcAppCommitEOF: GPIF II ステートマシンが生成した一杯になっていない DMA バッファを EZ-USB™ FX3 RAM にコミット CyFxUvcApplnDmaCallback: EZ-USB™ FX3 からホストへの出力ビデオ データを追跡 CyFxUVCApplnUSBSetupCB: ホストが送信した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、ストリームが停止する時点を検出 CyFxUVCApplnUSBEventCB: 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 CyFxUVCApplnAbortHandler: エラーやストリーム停止要求が検出された時にストリーム ビデオ データを中止 CyFxAppErrorHandler: エラー処理関数。これは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 CyFxUVCAddHeader: ストリームが有効になっている間に UVC ヘッダーをビデオ データに追加

ファイル	説明
	CyFxUvcApplnStop: GPIF II ステート マシンを停止し、DMA およびエンドポイント バッファをリセット CyFxUvcApplnStart: DMA チャネル転送および GPIF ステート マシンを起動 CyFxUvcAppProgressTimer: センサー/ISP (イメージ信号プロセッサ) が所定のフレーム間隔時間内にビデオ データ送信が失敗した場合、DMA およびエンドポイントをリセットし、ストリームをやり直す
cyfxuvcdscr.c	UVC アプリケーションの USB 列挙ディスクリプタを含む。フレームレートやイメージ解像度、ビット深度、サポートされているビデオ制御機能を変更する場合は、このファイルも変更が必要。UVC 仕様にすべての必要な詳細情報が記載されている
uvc.h	PTZ サポート、デバッグ インタフェース、フレーム カウント用のデバッグ プリント、およびフレーム タイマーをオン/オフにするようにアプリケーション動作を変更するための switch を含む uvc.c、cyfxuvcdscr.c、camera_ptzcontrol.c および sensor.c ファイルで共通に使用される定数を含む
cyfx_gcc_startup.s	このアセンブリ ソース ファイルには、EZ-USB™ FX3 CPU の起動コードが含まれている。これは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

ファームウェアは最初に EZ-USB™ FX3 の CPU を初期化し、その I/O を設定します。その後、ThreadX リアルタイム オペレーティング システム (RTOS) を起動するために **CyU3PKernelEntry** 関数を呼び出します。**uvcAppThread** と **uvcAppEP0Thread** の 2 つのアプリケーション スレッドを作成します。RTOS はこれらのアプリケーション スレッドを実行し、アプリケーション スレッド関数 (**UVCAppThread_Entry**、**UVCAppEP0Thread_Entry**) の実行をスケジュールするためにリソースを割り当てます。

Figure 47 に基本プログラム構造を示します。

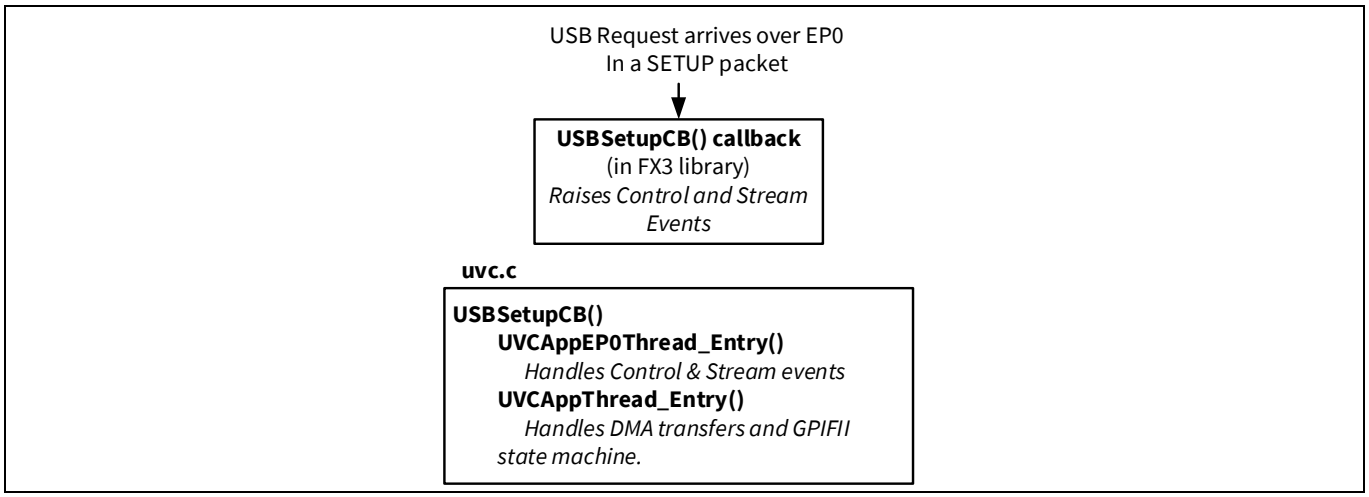


Figure 47　カメラ プロジェクト構造

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) スレッドはビデオ データのストリームを管理します。ストリーム開始前にストリーム イベントを待機し、エラーの場合に FIFO をクリーンアップします。

ファームウェアは EP0 を介して輝度、PTZ、PROBE/COMMIT 制御機能などの UVC 固有の制御要求 (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 **CyFxUVCApplnUSBSetupCB** (CB=コールバック) 関数を使って処理されます。これらの制御要求の 1 つが FX3 によって受信されると、この関数は対応するイベントを発生させます。EP0Thread はその後、クラス固有の要求を処理するためにこれらイベントでトリガーします。

5.2 初期化

UVCAppThread_Entry は UART デバッグ機能の初期化用に **CyFxUVCApplnDebugInit** を、EZ-USB™ FX3 の I^2C ブロックの初期化用に **CyFxUVCApplnI2CInit** を、残りの必要なブロック、DMA チャネル、USB エンドポイントの初期化用に **CyFxUVCApplnInit** はを呼び出します。

5.3 列挙

CyFxUVCApplnInit 関数では、**Cy3PUsbSetDesc** 関数を呼び出し、EZ-USB™ FX3 が UVC デバイスとして列挙されることを保証します。UVC ディスクリプタは cyfxuvcdscr ファイルで定義されます。これらディスクリプタは、非圧縮 YUY2 形式を使用してピクセルあたり 16 ビットを送信する、30FPS での 1280×720 ピクセルの解像度のイメージ センサー用に定義されます。これら設定を変更する必要がある場合は **2.3.1 節**を参照してください。

5.4 I^2C インタフェースによるイメージ センサー設定

イメージ センサーは EZ-USB™ FX3 の I^2C マスター ブロックを使って設定されます。**SensorWrite2B**、**SensorWrite**、**SensorRead2B** および **SensorRead** 関数 (sensor.c ファイルで定義される) は、I^2C を介してイメージ センサー コンフィギュレーションを読み書きするために使用されます。データをイメージ センサーへ書き込むために **SensorWrite2B** と **SensorWrite** 関数は標準 API **CyU3PI2cTransmitBytes** を呼び出します。データをイメージ センサーから読み出すために **SensorRead2B** と **SensorRead** 関数は標準 API **CyU3PI2cReceiveBytes** を呼び出します。これらの API 詳細は **FX3 SDK API ガイド**を参照してください。

5.5 ビデオ ストリームの開始

VLC Player、AMCap、Webcamoid または VirtualDub などの USB ホスト アプリケーションは、UVC ドライバーを使用してビデオを表示し、USB インタフェースと USB 代替設定の組み合わせをビデオ ストリーム (通常はインタフェース 0 代替設定 1) に設定し、PROBE/COMMIT 制御を送信します。これは、ホストがビデオ データのストリームをまもなく開始するというホストの指示です。ストリーム イベント時に、USB ホスト アプリケーションは EZ-USB™ FX3 からビデオ データの要求を開始します。EZ-USB™ FX3 がイメージ センサーから USB 3.0 ホストへのイメージ センサーからのビデオ データ送信を開始することを前提にします。ファームウェアでは、**UVCAppThread_Entry** 関数は無限ループです。ストリームがない間、メイン アプリケーション スレッドはストリーム イベントが起きるまでこのループで待機します。

Note: ストリーム イベントがない場合、EZ-USB™FX3 はデータを転送する必要はありません。したがって、GPIF II 状態マシンは無効にされます。そうでない場合、ホスト アプリケーションが DMA バッファからデータを取り出す前に、すべての DMA バッファが一杯になってしまい、EZ-USB™ FX3 は不良フレームを送信します。したがって、GPIF II ステート マシンはストリーム イベントがある場合にのみ開始する必要があります。

EZ-USB™ FX3 がストリーム イベントを受信すると、メイン アプリケーション スレッドは **CyU3PGpifSMSwitch** 関数を呼び出し、GPIF II ステート マシンを開始します。この関数の中で、ファームウェアは任意の状態から開始状態 (**START_SCK0**) に切り替わり、GPIF ステート マシンを再起動します。**CyU3PGpifSMSwitch** 関数の引数として開始ステート名と開始条件を渡します。GPIF II ステート マ

シンが任意のステートから開始ステートに切り替わることを保証するために、遷移元ステートと終了ステートは任意の無効なステート (添付プロジェクトの 257) です。開始ステート (**START_SCK0**) と開始条件 (**ALPHA_START_SCK0**) は **cyfxgpif2config.h** ファイルで定義されます。**cyfxgpif2config.h** ファイルは、**3.6 節**で説明した GPIF II Designer ツールから生成されました。

*Note: **EZ-USB™FX3 SDK API ガイド**は **CyU3PGpifLoad** と **CyU3PGpifSMSwitch** などの GPIF II 関連関数についての詳細情報を記載します。*

5.6 DMA バッファの設定

UVC 仕様では、各 USB 転送 (すなわち、本アプリケーションでは各 16KB の DMA バッファ) に 12 バイトのヘッダーを追加する必要があります。しかし、EZ-USB™ FX3 アーキテクチャでは DMA ディスクリプタに関連付けられる DMA バッファの大きさが 16 バイトの倍数になっている必要があります。

EZ-USB™ FX3 CPU が書き込む DMA バッファの 12 バイトの予約は、DMA バッファ境界を 16 バイトの倍数にしません。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。これにより、EZ-USB™ FX3 ファームウェアが追加する 12 バイトのヘッダーを除いた DMA バッファ サイズは 16,384-16=16,368 バイトになります。DMA バッファは、最初は 12 バイトのヘッダー、次は 16368 ビデオ バイト、そして最後は 4 個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、すなわち 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。

5.7 ビデオ ストリーム中の DMA バッファの扱い

CyFxUVCApplnInit 関数はプロデューサ イベントとコンシューマ イベントのコールバック通知を含む手動 DMA チャネルを作成します。この通知はセンサーが送信したデータ量と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。GPIF II が DMA バッファを生成し、EZ-USB™ FX3 が **CyU3PDmaMultiChannelGetBuffer** によりこのバッファへのポインタを取得すると、DMA バッファが FX3 CPU によって使用できるになります。このバッファは **CyU3PDmaMultiChannelCommitBuffer** により USB にコミットされます。USB スループットが、バッファが (GPIF II ステート マシンによって) 一杯にされる速度よりも遅い時はいつでも、実際のビデオ データを持つバッファの数が増えます。ある時点で、すべてのバッファが一杯になり、もう 1 つのバッファをコミットするとエラーが発生します。これはコミット バッファ障害と呼ばれます。このような場合、DMA リセット イベントが呼び出され、EZ-USB™ FX3 がストリームを停止してやり直します。

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、生成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で DMA バッファを USB にコミットします。DMA バッファ (GPIF II によって生成される) の強力なラップアップは、GPIF II コールバック関数 **CyU3PDmaMultiChannelSetWrapUp** で実行されます。**CyFxGpifCB** コールバック関数は、GPIF II が CPU 割込みをセットした時にトリガーされます。GPIF II 状態図に示すように、この割込みはフレーム有効信号がデアサートされると発生します。

UVC ヘッダーはフレーム識別子とフレーム終了マーカーに関する情報を持っています。フレーム終了時に、EZ-USB™ FX3 ファームウェアは 2 番目の UVC ヘッダー バイト (「CyFxUVCAddHeader」を参照してください) のビット 1 をセットし、ビット 0 をトグルします。フレームが終了した後にのみ、UVC ヘッダーの FID ビットをトグルします。

USB ホストの速度が遅くなると、コミット バッファ障害が発生し、DMA がリセットされ、ビデオ ストリームがやり直されます。センサー/ISP がビデオを時間とおりに送信できなかった場合、ビデオは最終的にフリーズするか、またはジッタが生じます。これを避けるために、フレーム タイマーが不可欠です。フレーム タイマーはビデオ ストリーム開始時に開始されます。これは、EZ-USB™ FX3 がセンサーから最初のバッファを受信するとリセットされ、各プロデューサ バッファが確実に到着するように再起動されます。

5.8 ビデオ ストリームの終了

イメージ ストリームを終了させるために、カメラをホストから切断する、USB ホストプログラムを終了する、USB ホストが EZ-USB™ FX3 にリセットまたは一時停止要求を発行するという 3 つの方法があります。これらのアクションは CY_FX_UVC_STREAM_ABORT_EVENT イベントをトリガーします (**CyFxUVCApplnAbortHdlr** 関数を参照してください)。このアクションは EZ-USB™ FX3 FIFO にデータが無い場合は常に起きません。これは適切なクリーンアップが必要であることを意味します。ファームウェアはストリーム関連の変数をリセットし、**UVCAppThread_Entry** ループの DMA チャネルをリセットします。ストリームが必要ないため、**CyU3PGpifSMSwitch** 関数は呼び出されません。ファームウェアはその後、次のストリーム イベントが発生するまで待機します。

アプリケーションが終了すると、Windows プラットフォームではクリア機能の要求を、または Mac プラットフォームでは代替設定=0 のインタフェース設定要求を発行します。この要求が受信されるとストリームが停止します。この要求は **CyFxUVCApplnUSBSetupCB** 関数で、switch 文 case の CY_U3P_USB_TARGET_ENDPT と CY_U3P_USB_SC_CLEAR_FEATURE 要求で処理されます。

5.9 「デバッグ」インタフェースの追加

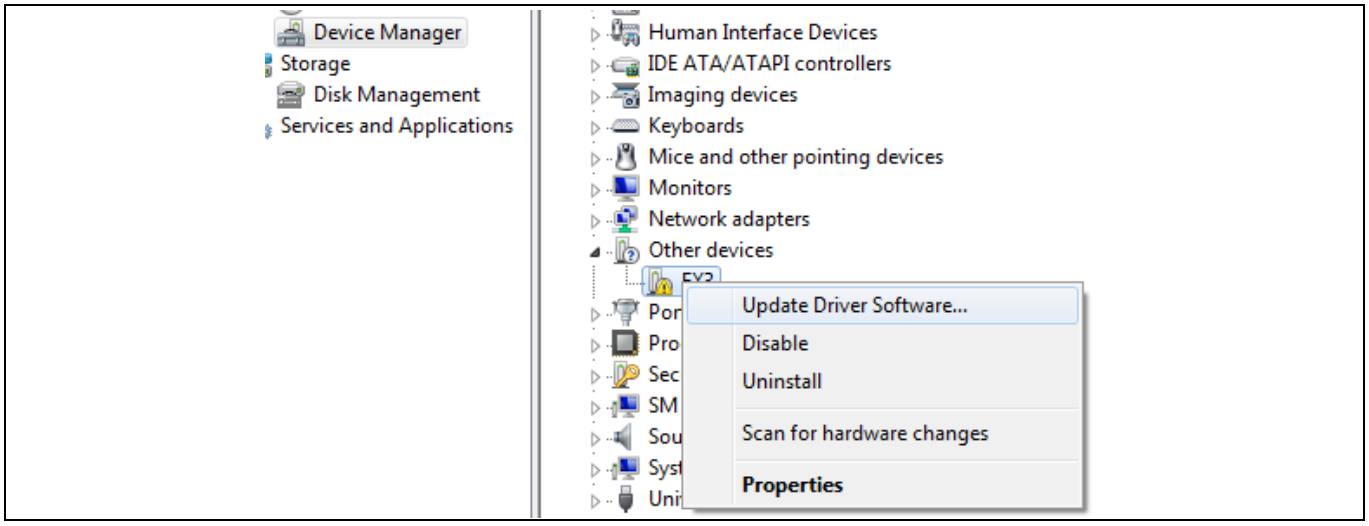
本節は、カメラ アプリケーションに 3 番目の USB インタフェースを追加し、デバッグやその他のカスタム目的で使用する方法を示します。

uvc.h ファイルの#define USB_DEBUG_INTERFACE は、**cyfxuvcdscr.c**、**uvc.h** および **uvc.c** のコードを有効にします。提供される例では、I^2C を介してイメージ センサー レジスタを読み書きします。

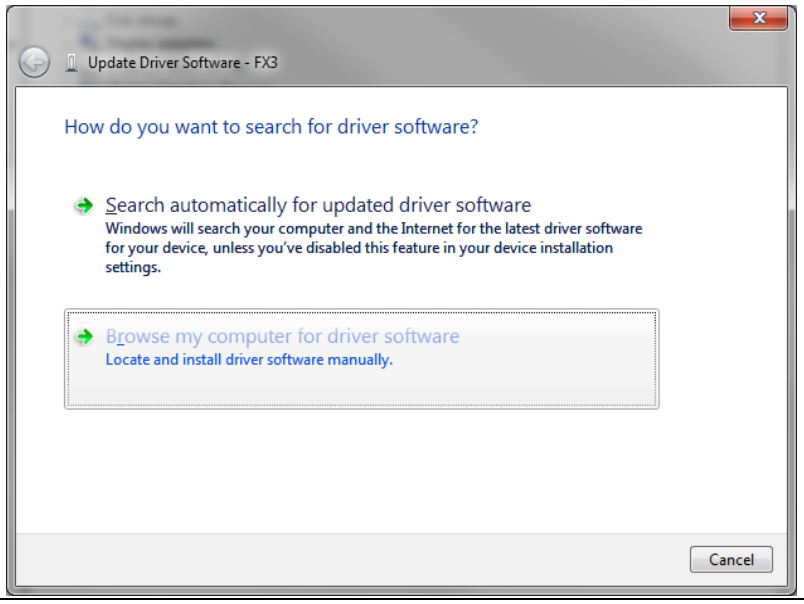
5.9.1 デバッグ インタフェースの詳細

2 つの BULK エンドポイントがこのインタフェース用に定義されます。EP 4 OUT はデバッグ コマンド BULK エンドポイントに設定され、EP 4 IN はデバッグ応答 BULK エンドポイントに設定されます。有効にされたデバッグ インタフェースを持つファームウェアが実行されると、EZ-USB™ FX3 は列挙中に 3 つのインタフェースを報告します。最初の 2 つは UVC 制御インタフェースとストリーム インタフェースで、3 番目はデバッグ インタフェースです。3 番目のインタフェースは EZ-USB™ FX3 SDK の一部として提供される CyUSB3.sys ドライバーにバインドする必要があります。以下の手順に従ってドライバーをインストールできます(64 ビット版 Windows7 のシステムを例として使用します。XP ユーザーは VID/PID を含めるように.inf ファイルを修正しなければなりません。その後、修正された.inf ファイルを使ってこのインタフェースに cyusb3.sys ドライバーをバインドします)。

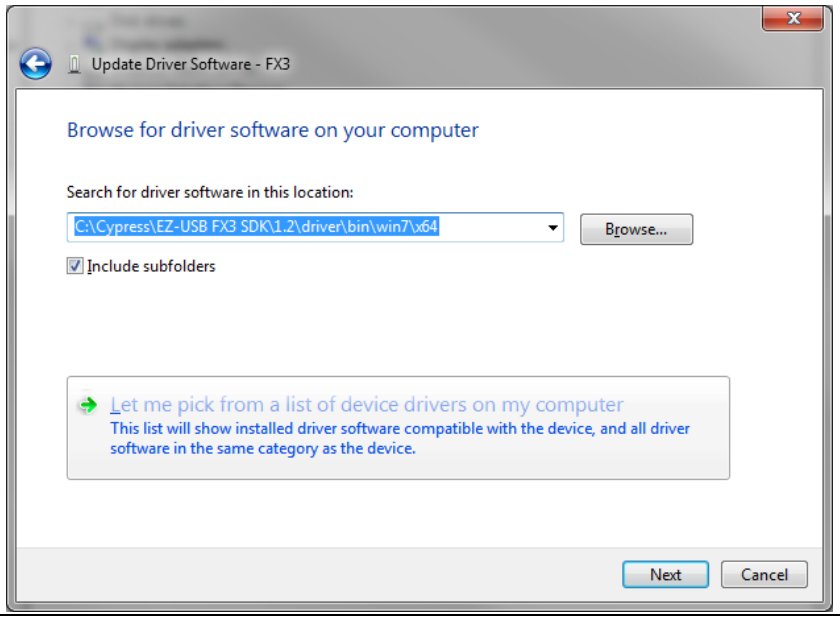
1. Device Manager を開き、**Other devices** の下の EZ-USB™ FX3 (または同等のもの) を右クリックし、**Update Driver Software…**を選択します。



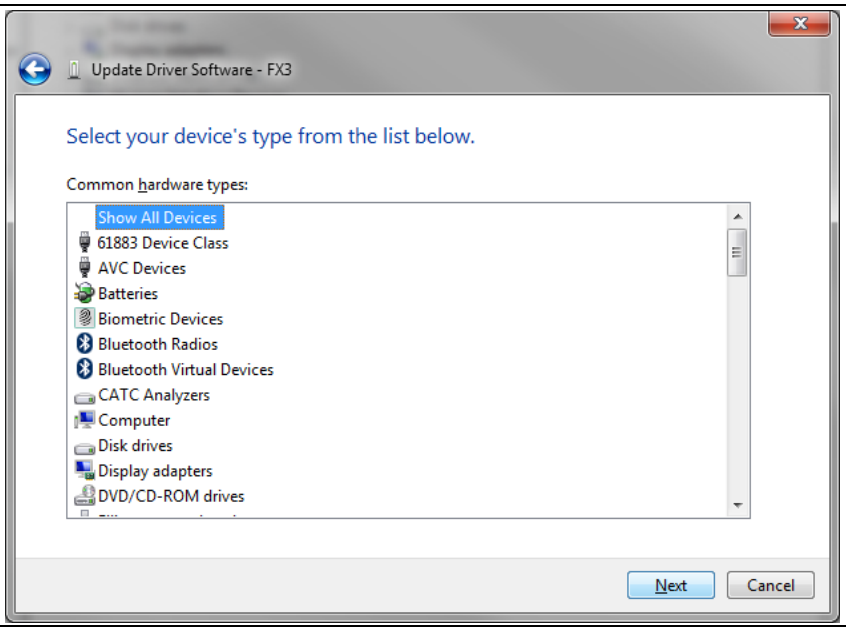
2. 次の画面で **Browse my computer for driver software** を選択します。



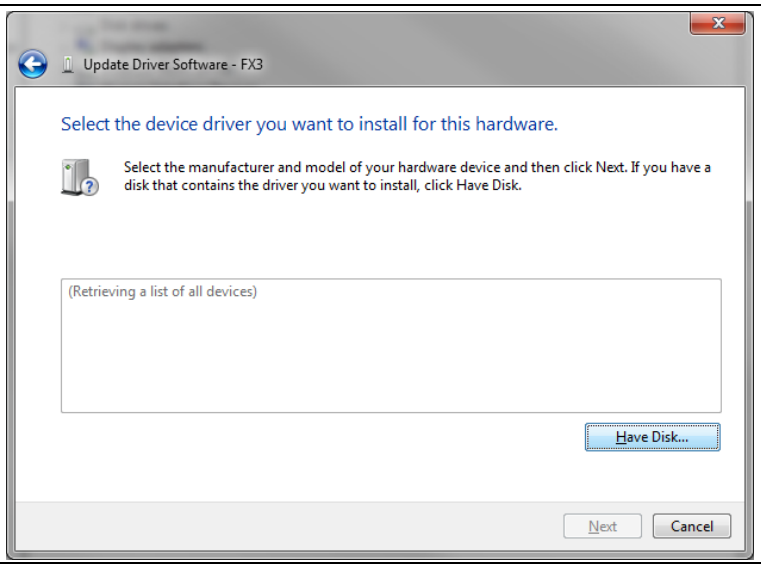
3. 続く画面で **Let me pick from a list of device drivers on my computer** を選択します。



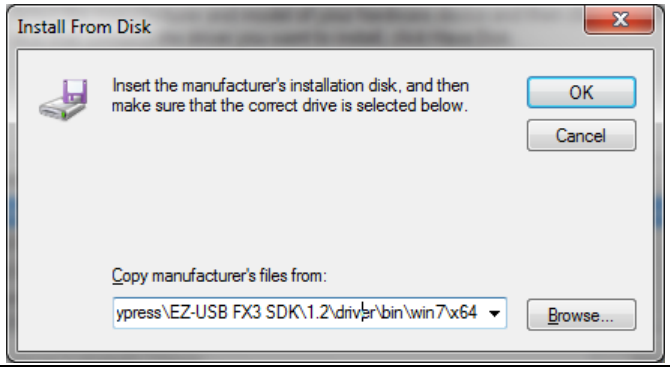
4. 続く画面で **Next** をクリックします。



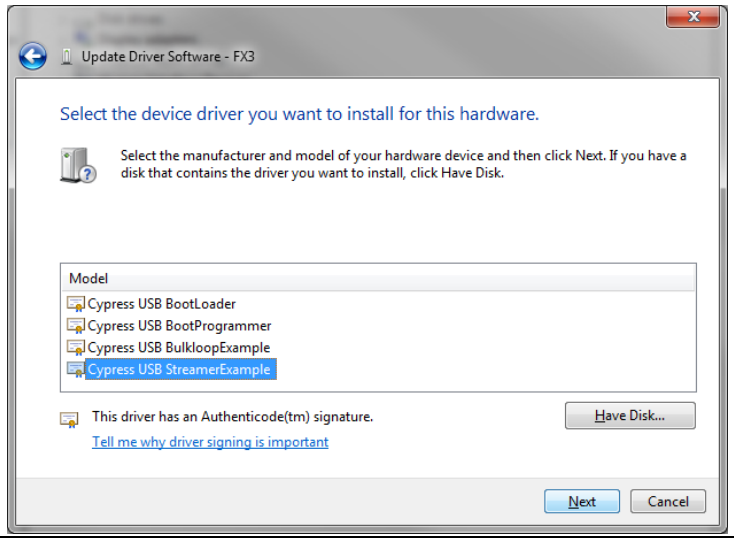
5. **Have Disk…**をクリックしてドライバーを選択します。



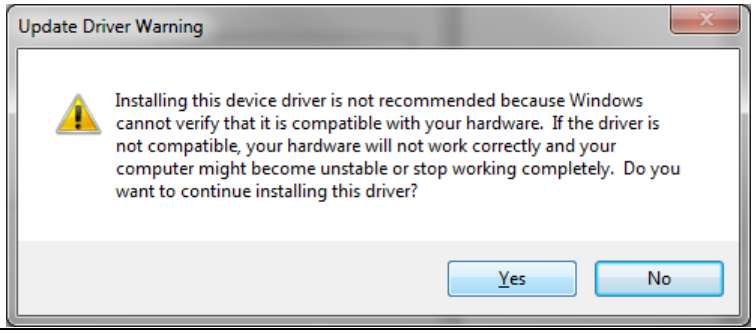
6. *<SDK installation>\<version>\driver\bin\<OS>\x86* または*\x64* フォルダ内の cyusb3.inf ファイルをブラウズし、**OK** をクリックします。



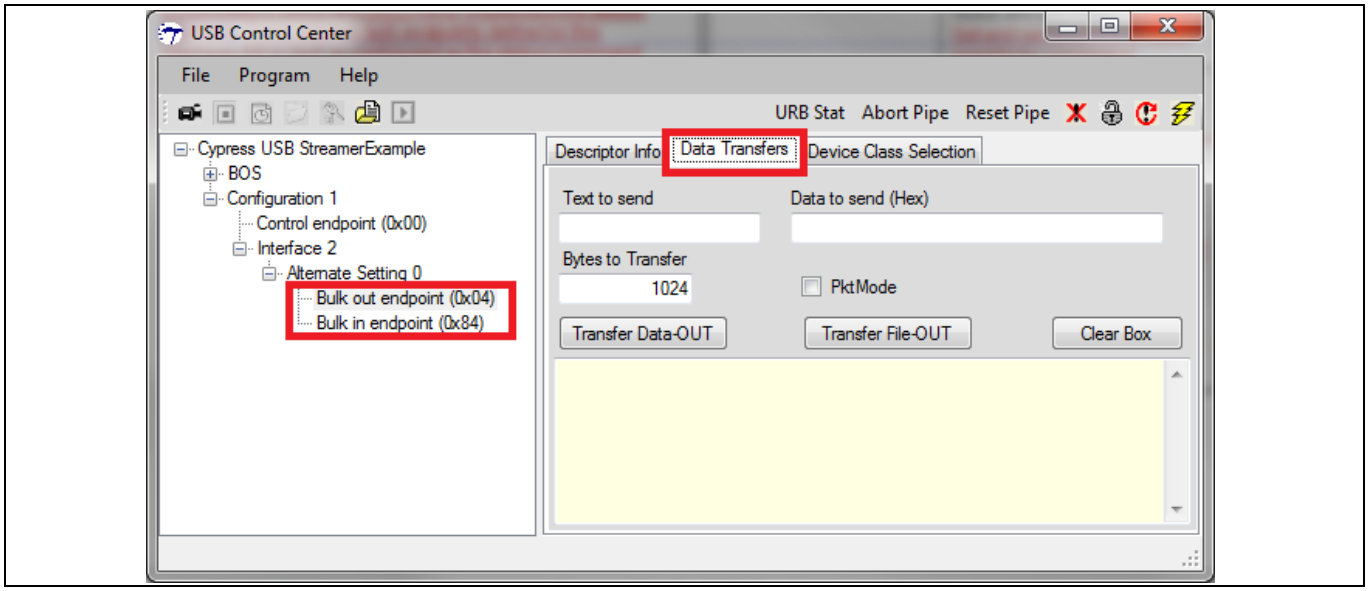
7. OS バージョンを選択し、**Next** をクリックしてインストールを行います。



8. 警告ダイアログ ボックスが現れた場合、**Yes** をクリックします。



ドライバーが 3 番目のインタフェースにバインドされると、デバイスはインフィニオン USB Control Center アプリケーションに表示されます。ユーザーはここからベンダー固有のデバイスとして FX3 ファームウェアにアクセスできます。デバッグ インタフェースの現在の実装は EP4-OUT コマンドおよび EP4-IN 応答エンドポイントを使用してイメージ センサー レジスタの読み書きを可能にします。これらのエンドポイントは Control Center で下図に示しているように、<FX3 デバイス名> → Configuration 1 → Interface 2 → Alternate Setting 0 を選択しアクセスできます。コマンドを送信したり、コマンド送信後に応答を取得したりするために、Data Transfers タブを使用します。

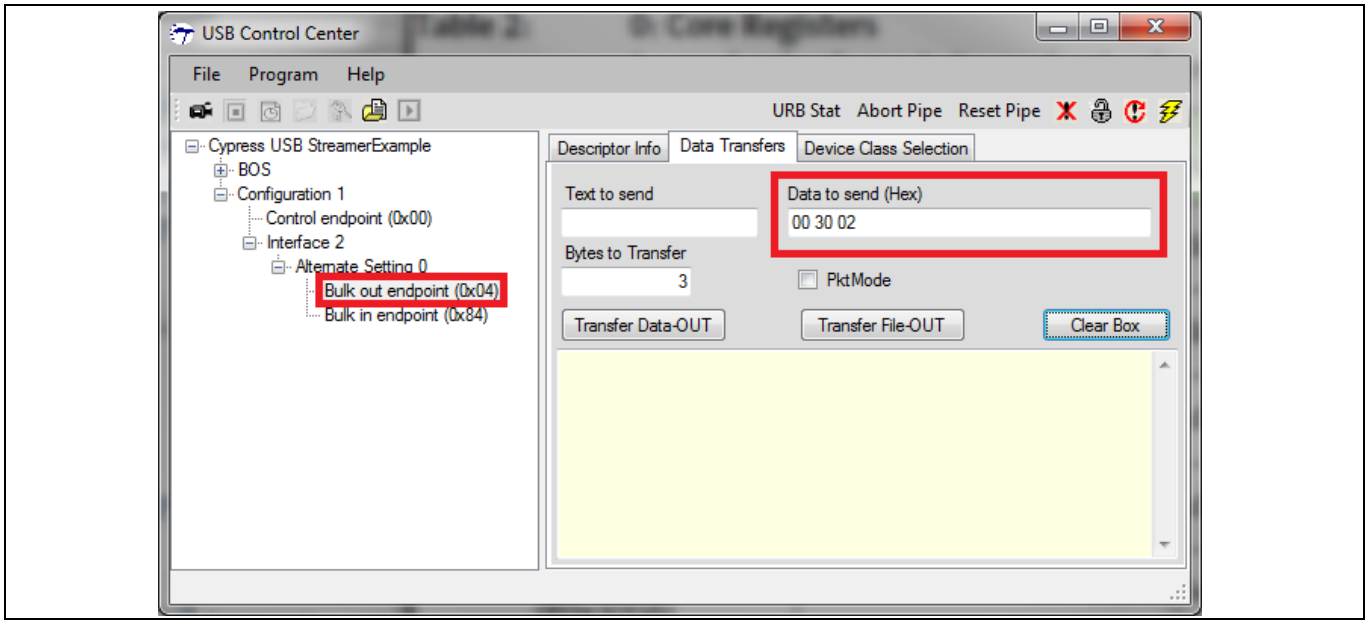


5.9.2 デバッグ インタフェースの使用

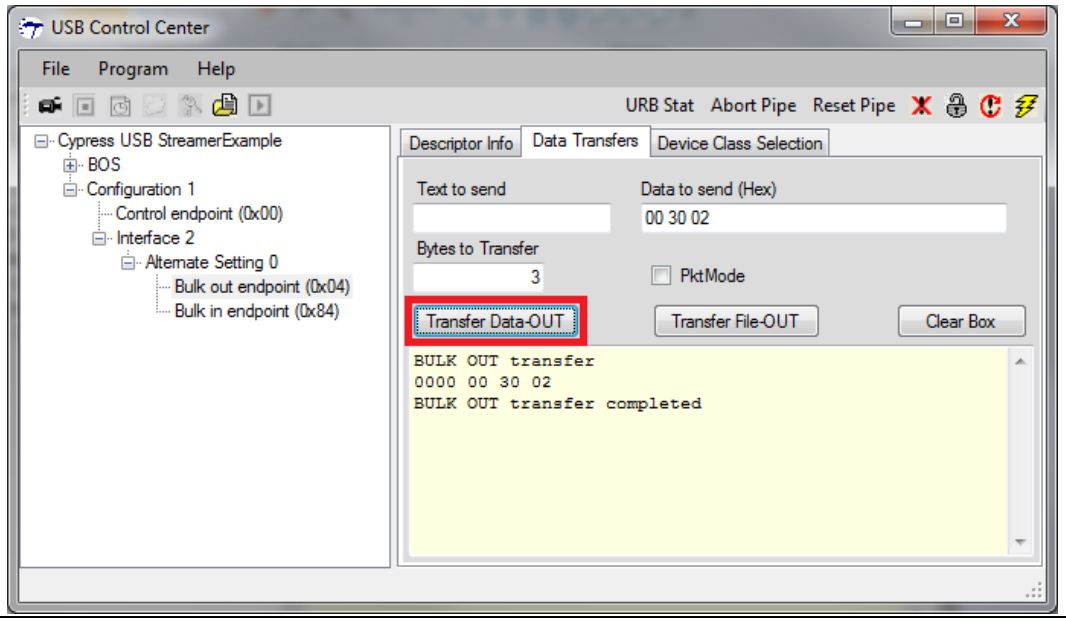
デバッグ インタフェースでは、シングル読み出し、シーケンシャル読み出し、シングル書き込み、シーケンシャル書き込みという 4 つのコマンドが実装されます。エラー チェックが無いため、コマンド入力には注意してください。ユーザーは適切な機能を確保するために、エラー チェックを実装できます。I^2C レジスタは 16 ビット幅であり、16 ビットでアドレス指定されます。

5.9.2.1 シングル読み出し:

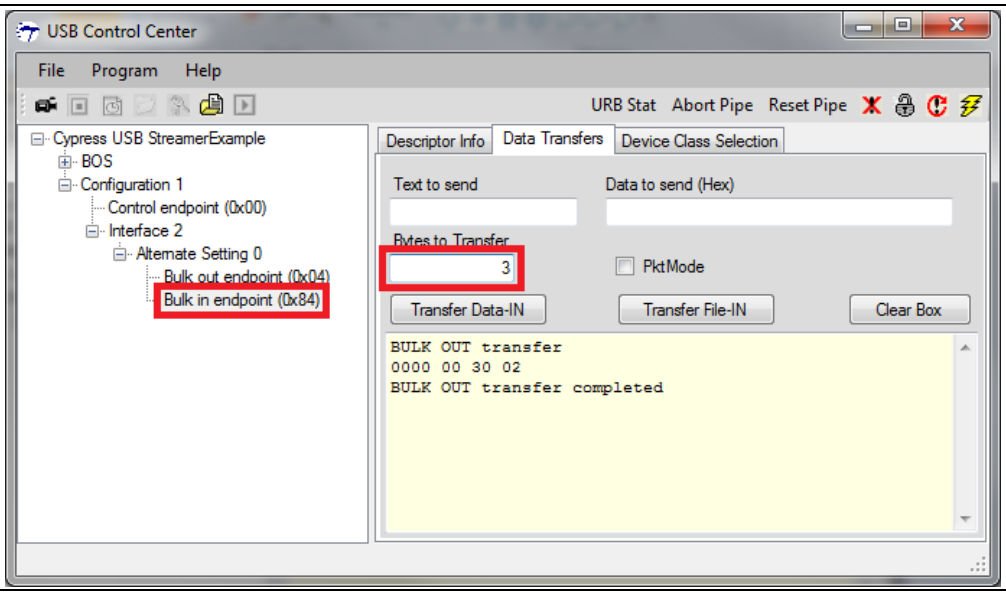
1. コマンド エンドポイントを選択し、Data Transfers タブの下に 16 進数でコマンドを入力します。シングル読み出しのフォーマットは、0x00 <レジスタ アドレス上位バイト><レジスタ アドレス下位バイト> です。下図にレジスタ アドレス 0x3002 の読み出しコマンドを示します。16 進数データ ボックスの入力中にスペースを使用しないでください。例えば、16 進数データ フィールドをクリックし、「003002」を入力します。



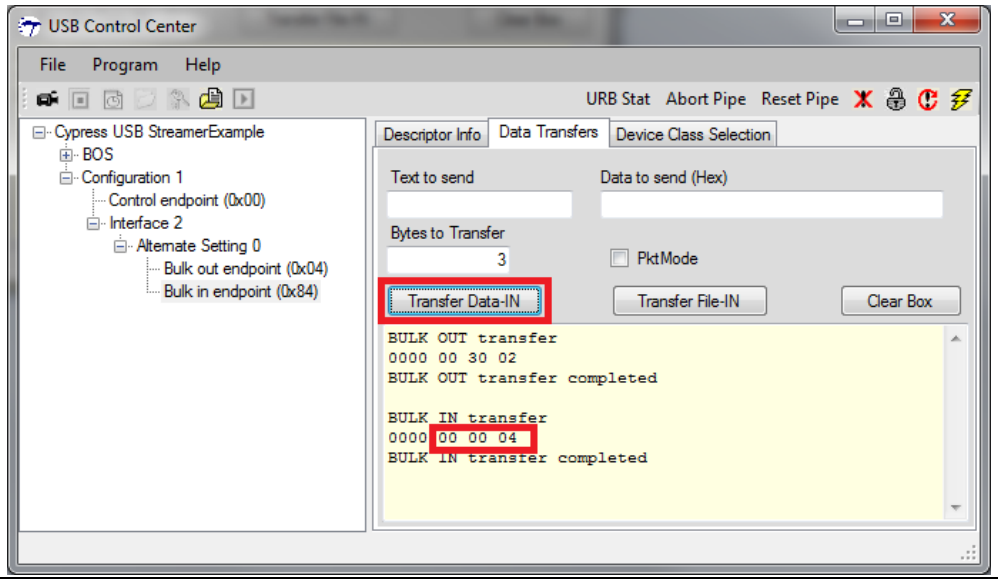
2. **Transfer Data-OUT** をクリックしてコマンドを送信します。



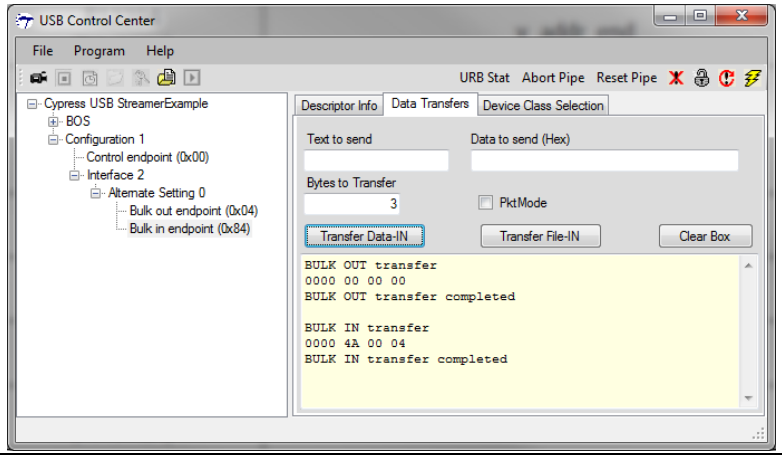
3. 応答エンドポイントを選択し、**Bytes to Transfer** フィールドに「3」を設定し、シングル読み出しコマンドの応答を読み取ります。



4. 「**Transfer Data-IN**」をクリックして応答を受信します。

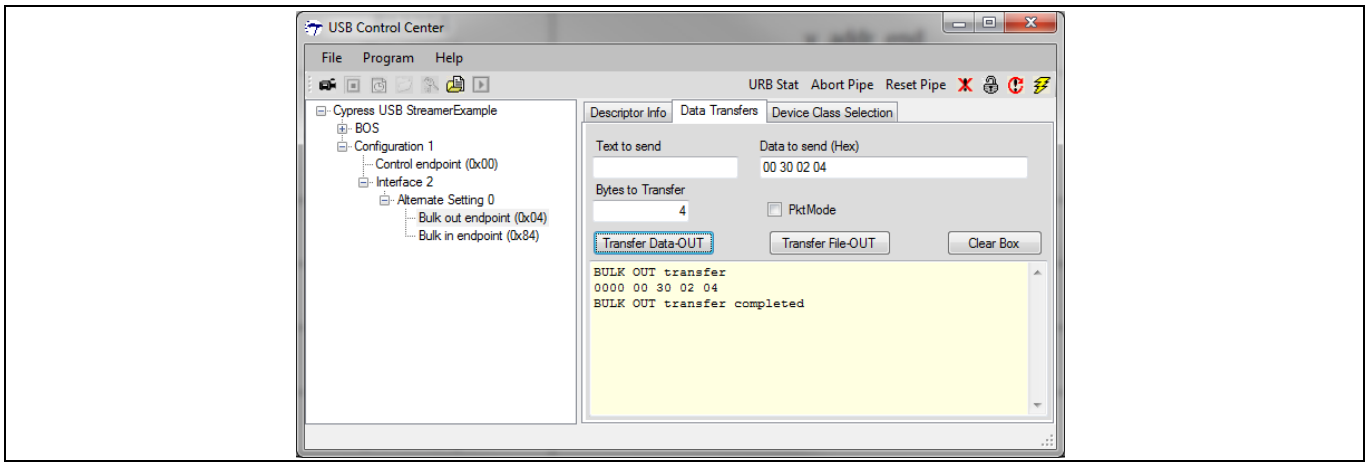


5. 応答の最初のバイトはステータスです。ステータス=0 はコマンドの成功を意味し、それ以外の値はコマンドが失敗したことを意味します。続くバイトは読み出されたレジスタの値が 0x0004 であることを示します。この例では失敗した転送を示し、その状態は非ゼロであり、残りのバイトは以前の転送の古い値です。

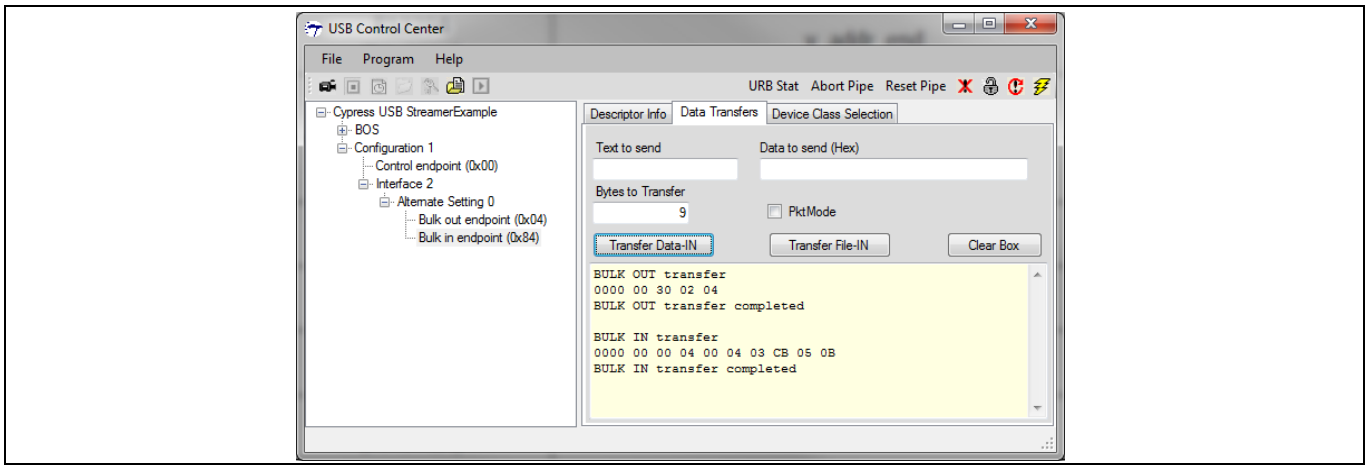


5.9.2.2 シーケンシャル読み出し:

1. コマンド エンドポイントを選択し、Data Transfers タブの下に 16 進数でコマンドを入力します。シーケンシャル読み出しのフォーマットは、0x00 <レジスタ アドレス上位バイト> <レジスタ アドレス下位バイト> <N>です。下図にレジスタ アドレス 0x3002 から始まる 4 つ (N=4) のレジスタの読み出しコマンドを示しています。

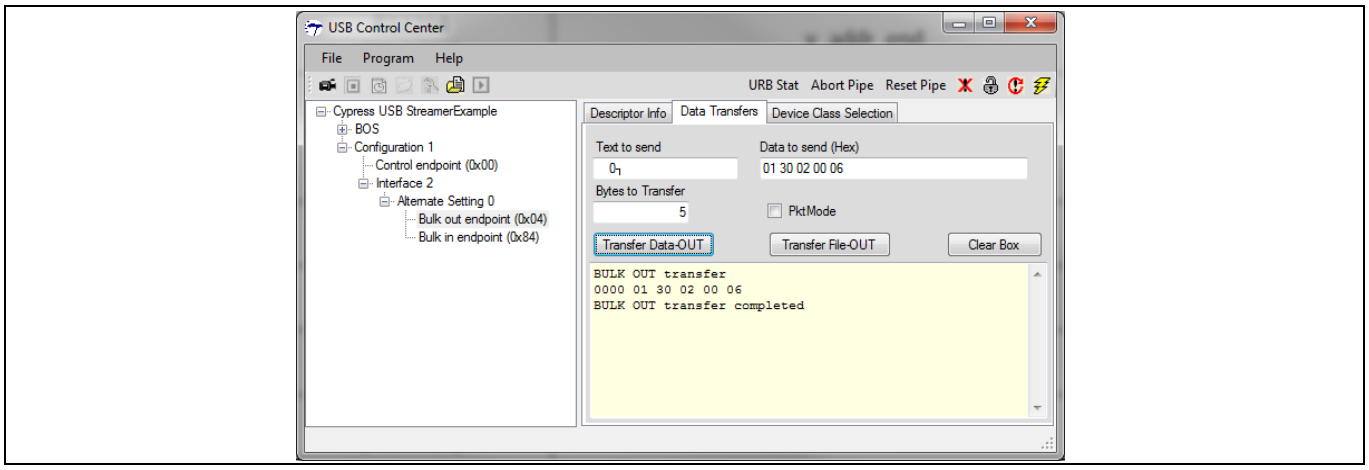


2. この場合、応答のための「Bytes to Transfer」 (転送するバイト) は (N*2+1)=9 です。この図は EZ-USB™ FX3 が読み出した値を示しています。

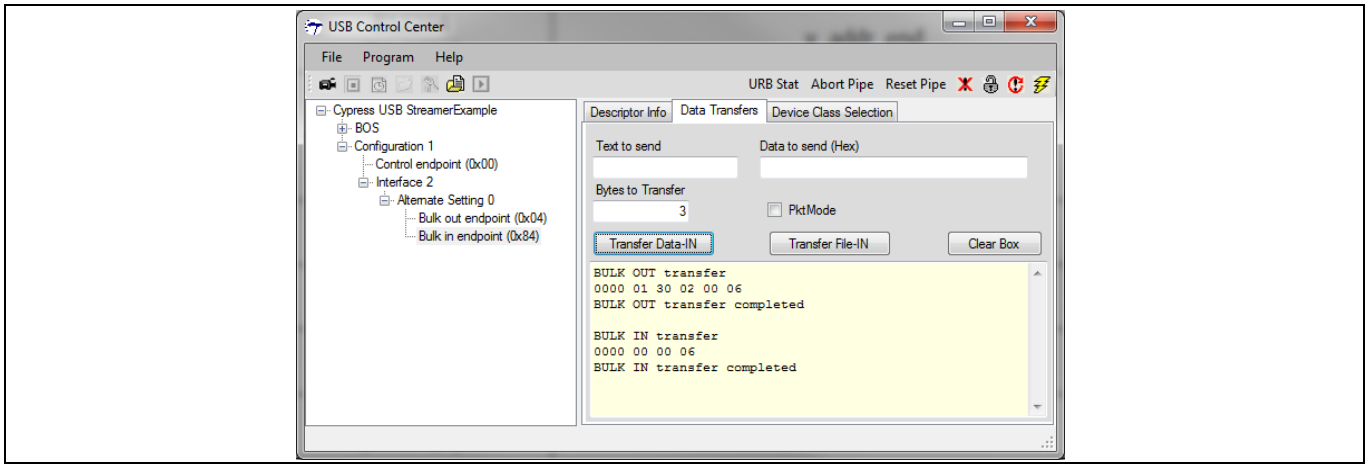


5.9.2.3 シングル書き込み:

1. コマンド エンドポイントを選択し、Data Transfers タブの下に 16 進数でコマンドを入力します。シングル書き込みのフォーマットは、0x01 <レジスタ アドレス上位バイト> <レジスタ アドレス下位バイト> <レジスタ値上位バイト> <レジスタ値下位バイト>です。下図はレジスタ アドレス 0x3002 に 0x0006 の値を書き込むための書き込みコマンドを示します。

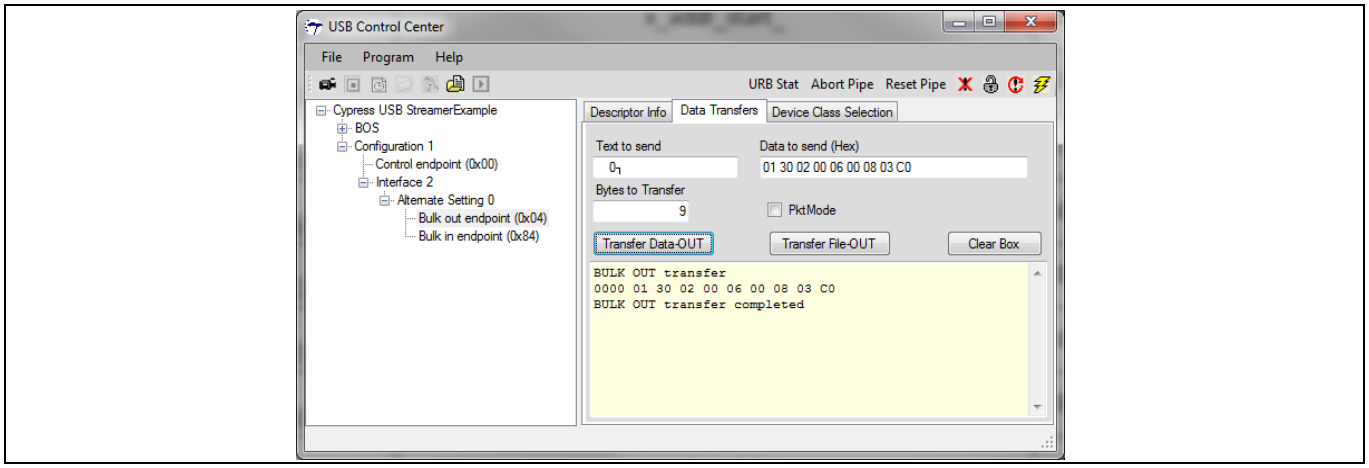


2. シングル書き込みの応答は 3 個のバイトを含みます: <ステータス> <レジスタ値上位バイト> <レジスタ値下位バイト>。これらレジスタ値が書き込まれた後に読み戻されるため、ユーザーは同じ値がコマンドで送信されたことを観察できます。

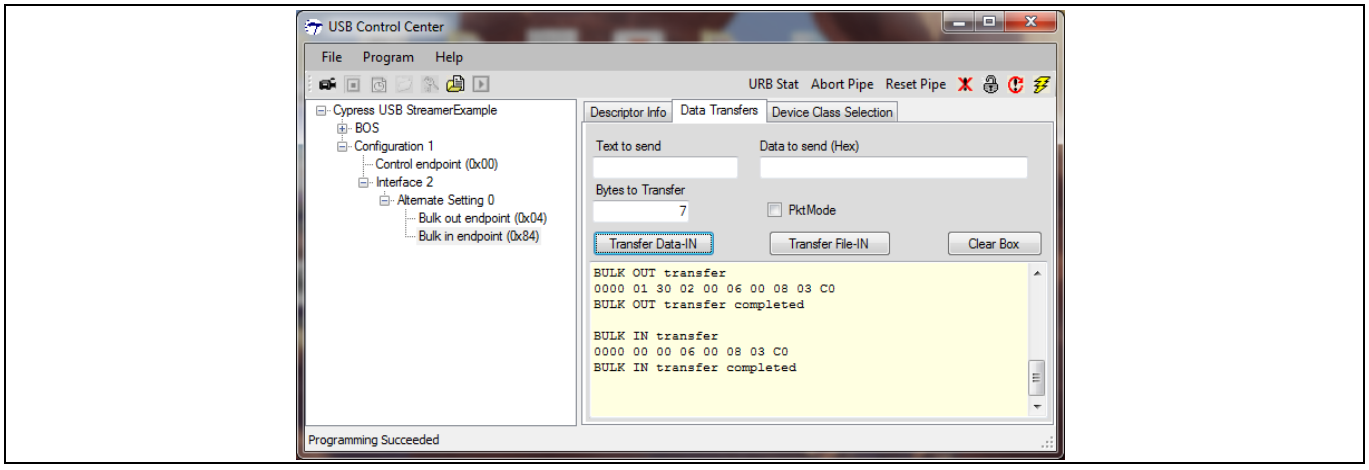


5.9.2.4 シーケンシャル書き込み:

1. コマンド エンドポイントを選択し、Data Transfers タブの下に 16 進数でコマンドを入力します。N 個のレジスタに書き込むシーケンシャル書き込みのフォーマットは 0x01 <レジスタ アドレス上位バイト> <レジスタ アドレス下位バイト> ((<レジスタ値上位バイト> <レジスタ値下位バイト>) *N 回) です。下図に、順次にレジスタ 0x3002、0x3004、0x3006 に書き込み値 0x0006、0x0008、0x03C0 を示します (N=3)。



2. 応答は <ステータス> (<レジスタ値上位バイト> <レジスタ値下位バイト>) * N 値です。この例では、転送するバイト数の合計は(2*N+1)=7 です。



6 ハードウェア設定

現在のプロジェクトは、SuperSpeed Explorer キット (CYUSB3KIT-003), ON Semiconductor イメージセンサー相互接続基板 (CYUSB3ACC-004A), および ON Semiconductor イメージセンサー基板 (MT9M114EBLSTCH3-GEVB / MT9M114_55CSP_HB_DEMO3_REV0) を含む構成でテストされています。これらの部品の入手方法の詳細は以下のとおりです。

6.1 ハードウェア要件

1. ON Semiconductor MT9M114 イメージセンサー基板 (MT9M114EBLSTCH3-GEVB/MT9M114_55CSP_HB_DEMO3_REV0) (ON Semiconductor 販売代理店から購入します)
2. SuperSpeed Explorer キット (CYUSB3KIT-003)[1]
3. SuperSpeed Explorer キット用の CYUSB3ACC-004A[2] (ON Semiconductor イメージセンサー用相互接続基板)
4. SuperSpeed の性能を評価するために、USB3.0 ホスト対応コンピュータを使用します。

6.2 SuperSpeed Explorer Kit 基板のセットアップ

以下の手順でビデオ アプリケーション テスト用の基板を準備します:

1. **Table 9** および **Table 10** に示すように、SuperSpeed Explorer キット (CYUSB3KIT-003) および ON Semiconductor イメージセンサー基板 (MT9M114_55CSP_HB_DEMO3_REV0) のジャンパ設定を設定します。

Table 9 ジャンパ設定 – ON Semiconductor イメージセンサー基板 (MT9M114_55CSP_HB_DEMO3_REV0)

ジャンパ/ヘッダー No.	ジャンパ/ヘッダー 名	ピン設定	説明
P1	OE_L	1-2	パラレルインタフェース有効
P2	CONFIG	2-3	ノーマルモードに設定
P4	FLASH	Open	外部フラッシュに接続
P5	TRST	1-2	ノーマルモードに設定
P6	SADDR	1-2	スレーブアドレス: 0x90
P7	+VDDIO	2－3	2.8V 動作のセンサー
P8	UART	Open	UART シャットダウン (Tristate)
P11	EEPROM	Closed (A1 column)	アクティブ low(A1)
P15	I2C CONNECTOR	1－2 および 3－4	マスターおよびセンサーI2C 接続
P32	CLOCK SELECTION	3－5 および 1－2	マルチカメラ, マスターモード; オンボード発振器

1 FX3 DVK (CYUSB3KIT-001) の使用の詳細については、11 を参照してください。

2 Aptina™イメージセンサーと Aptina™イメージセンサー相互接続基板 (CYUSB3ACC-004) の使用の詳細については、12 を参照してください。

Table 10 SuperSpeed Explorer キット (CYUSB3KIT-003)

ジャンパ/ヘッダー No.	ジャンパ/ヘッダー 名	ピン設定	説明
J2	VIO1_3	Closed	VDDIO 電圧選択 (2.8V)
J3	VBUS_JUMPER	Closed	バスパワー
J4	PMODE	Closed	USB ブート
J5	SRAM_ENABLE	Open	オンボード SRAM 無効

2. **Figure 48** に示すように、SuperSpeed Explorer キット、相互接続基板、および ON Semiconductor イメージセンサー基板を組み立てます。

Figure 48 SuperSpeed Explorer キット、ON Semiconductor イメージセンサー相互接続およびセンサー基板の組み立て

3. キットに付属された USB 3.0 ケーブルを使用し、SuperSpeed Explorer キット基板をコンピュータの USB ポートに差し込みます。
4. EZ-USB™ FX3 SDK の一部として提供された Control Center アプリケーションを使用し、基板にファームウェアをロードします。ファームウェアとビルド済みイメージは、**AN75779.zip** を参照してください。詳細な手順は **AN75705, Getting Started with EZ-USB™ FX3** を参照してください。操作概要は以下のとおりです。

   a) Control Center アプリケーションを開始します。EZ-USB™ FX3 Explorer 基板を接続すると、インフィニオン EZ-USB™ FX3 USB ブートローダデバイスとして認識されます (**Figure 49**)。

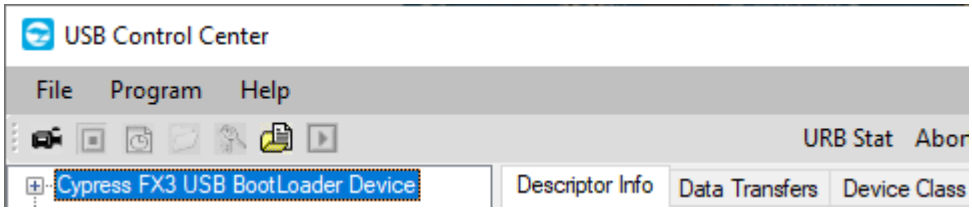


Figure 49 EZ-USB™ FX3 がブートローダとして列挙

b) **Program** > **FX3** > **RAM** を選択し、本アプリケーション ノートに添付される *cyfx_uvc_an75779.img* ファイルへ移動します (を参照してください)。デバイスは、プログラミング後に電源を入れ直した場合、ロードされたファームウェアを失い、インフィニオン FX3 USB ブートローダデバイスとして再列挙されます。

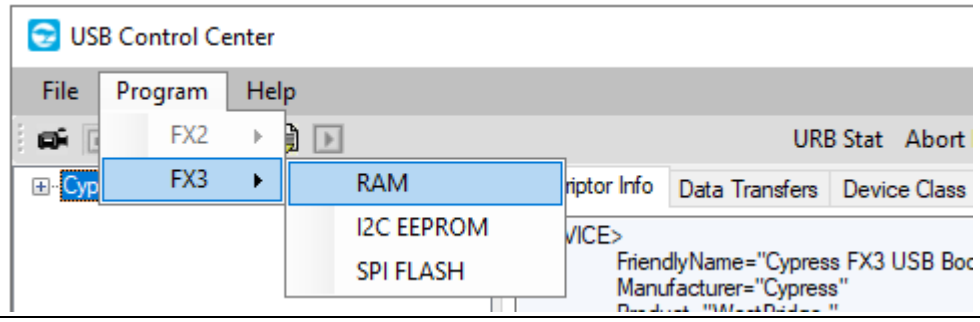


Figure 50 EZ-USB™ FX3 RAM へのコードのロード

Control Center アプリケーションは、ステータスバーにプログラミングのステータスを表示します。プログラミングが正常に完了すると、デバイスは UVC クラスデバイスとして列挙されるため、コントロールセンターのデバイスリストにも表示されなくなります。デバイスは、Windows デバイスマネージャーの **Cameras** または **Imaging Devices** タイプで確認できます (**Figure 51**)。

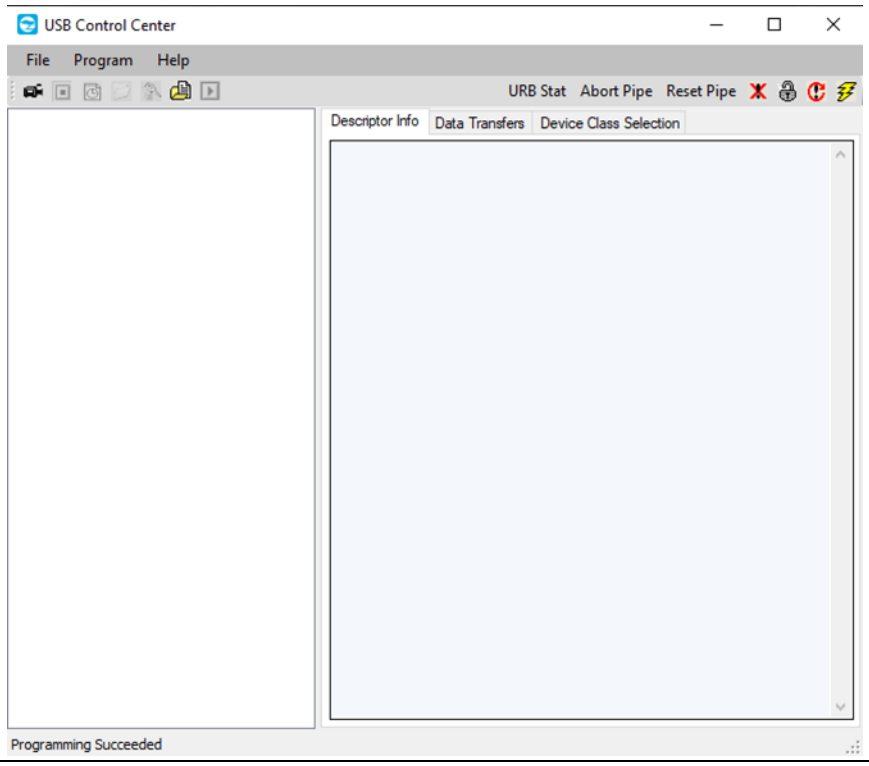


Figure 51 ステータスバーに表示されるプログラミング成功メッセージ

7 UVC ベース ホスト アプリケーション

UVC デバイスからのビデオ表示およびキャプチャを可能にするさまざまなホスト アプリケーションがあります。通常、VLC Media Player が選択されます。もう 1 つの広く使用されている Windows アプリケーションは **VirtualDub** (オープンソースアプリケーション) です。その他の追加の Windows アプリケーションは、**MPC-HC** Player (オープンソースアプリケーション)、**AmCap** (バージョン 8.0)、**Webcamoid** および **Debut Video Capture ソフトウェア**です。

Linux システムは、ビデオをストリームするために V4L2 ドライバーと VLC Media Player を使用できます。VLC Media Player はインターネットからダウンロードできます。**Webcamoid** は Linux プラットフォームでも利用可能です。

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

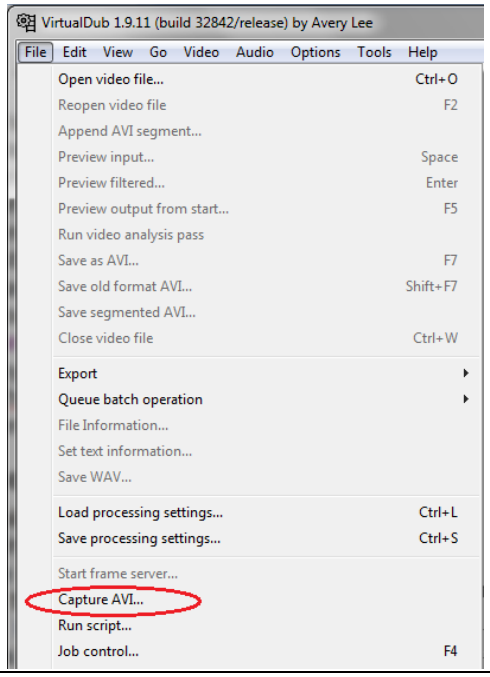
*Note: **VirtualDub** や **MPC-HC** などのオープン ソース アプリケーションを使用し、独自のホスト アプリケーションを設計します。**AmCap サンプル**は Windows SDK でも入手できます。*

7.1 デモの実行

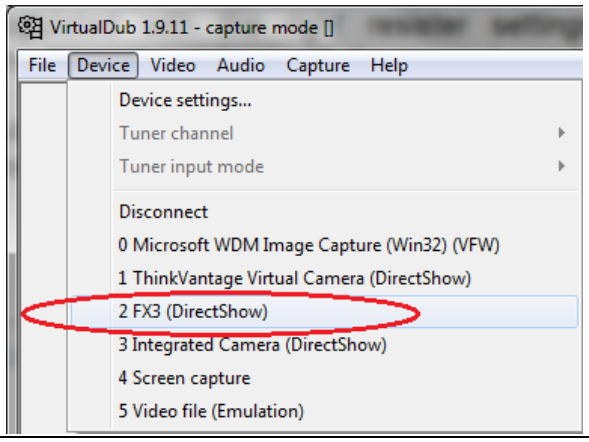
EZ-USB™ FX3 Explorer キットでのデモ用の事前コンパイルしたコード イメージ ファイルは **AN75779.zip** で入手できます。

このコードを実行するには、以下の手順を実施します:

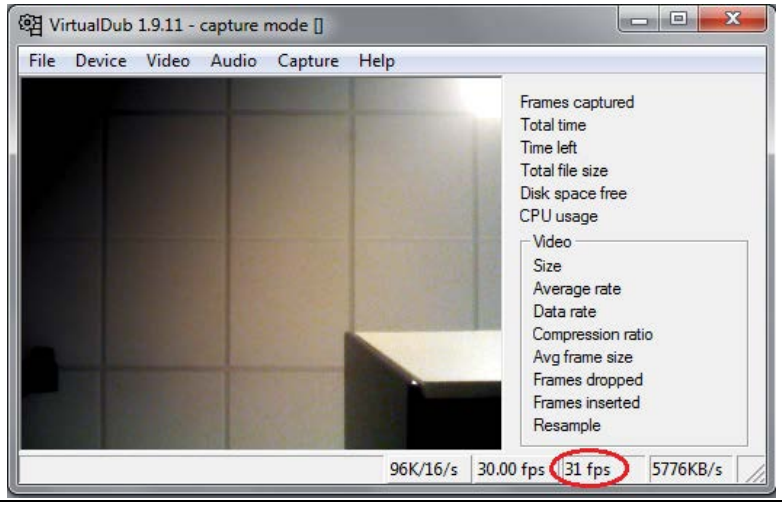
1. **6.2 節**に説明したとおり、EZ-USB™ FX3 UVC セットアップに事前コンパイルされたファームウェア イメージをにロードします。
2. この時点で、セットアップは UVC デバイスとして再列挙されます。オペレーティング システムは UVC ドライバーをインストールします。追加のドライバーは必要ありません。
3. ホスト アプリケーション (VirtualDub など) を開きます。
4. **File** > **Capture AVI** を選択します。



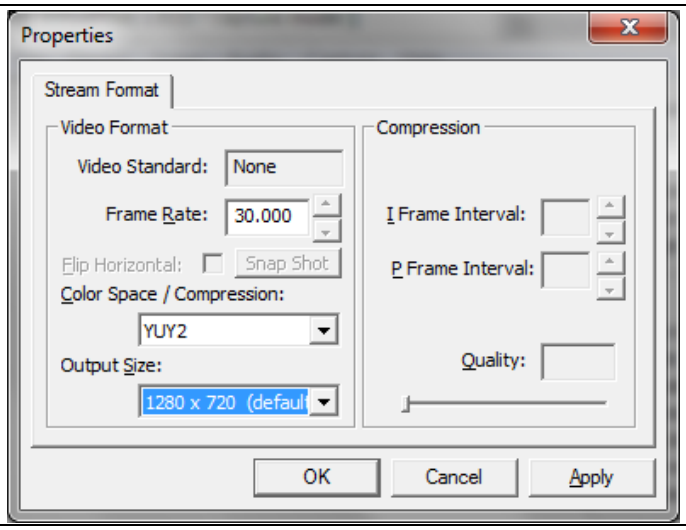
5. **Device** > **FX3 (Direct Show)**を選択すると、イメージのストリームが開始されます。



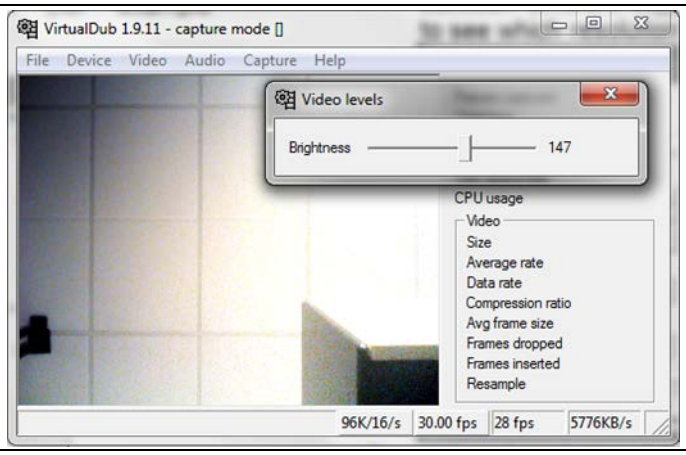
6. 右下の値は実際のフレーム レートを示します。



7. また、**Video** > **Capture Pin** を使用して、サポートされているさまざまな解像度の内から対象解像度を選択したり、現在アクティブな解像度を確認できます。



8. また **Video** > **Levels** を使用して、スライダ位置を変更して値を変更することにより輝度、または他のサポートされている制御コマンドを変更できます。**Video** > **Capture Filter** の下に追加の制御コマンドを検索します。



Note: *露出、シャープネスなどのビデオ制御を有効にするには、ON Semiconductor との秘密保持契約 (NDA) が必要です。EZ-USB™FX3 ファームウェアからのこれらの追加制御を有効にするための支援が必要な場合は、実行された NDA とともに FAE やお近くの FAE または販売代理店にお問い合わせください。*

8 トラブルシューティング

8.1 ホストアプリケーションで見える黒い画面または不正な色

1. 性能は最適化されているため、ファームウェアのリリース ビルドを使用してください。
2. EZ-USB™ FX3 が画像をストリーミングしているかどうかを調べるために、uvc.h ファイルの「DEBUG_PRINT_FRAME_COUNT」スイッチを有効にしてください。この switch 文は、フレーム カウント用の UART プリントを可能にします。SuperSpeed Explorer キットでは、UART はオンボードの統合デバッガを介して常に利用可能です。ハイパーターミナル、Tera Term、または PC の COM ポートにアクセスできる別のユーティリティを開きます。

   転送開始前に、UART 設定を次のようにしてください。**115200 ボー、パリティなし、1 ストップ ビット、フロー制御なし、および 8 ビット データ**。これはデバッグ プリントをキャプチャするのに十分です。PC ターミナル プログラムで、増分のフレーム カウンターが表示されない場合は、EZ-USB™ FX3 とイメージ センサー (GPIF またはセンサー制御/初期化) 間のインタフェースに問題がある可能性があります。
3. PC 端末プログラムで増分のフレーム カウンターのプリントが表示された場合、送出されるイメージ データを検証する必要があります。USB トレースはフレームごとに転送されるデータ量を表示できます。
4. フレームごとに送出されるデータの合計量を確認するには、ヘッダーにフレーム終了ビットがセットされているデータ パケットを探します(フレーム終了の転送では、ヘッダーの 2 番目のバイトは 0x8e または 0x8f です)。フレームで転送されたイメージ データ総量 (UVC ヘッダーを含まない) は次の式で計算されます: **幅 * 高さ * ピクセルサイズ** (バイト)。これは、USB トレースからの量でない場合、イメージ センサーの設定、または GPIF のインタフェースに問題があります。

Note: LeCroy、Ellisys、Beagle などのハードウェアベースの USB プロトコルアナライザツールや Wireshark、USBlyzer などのソフトウェアベースのツールを使用して USB トレースをキャプチャして分析できます。

5. デバッグプリントで DMA リセットイベントが観測されているかどうかを確認してください。Yes の場合、イメージセンサーからデータが正しく送信されているかどうか確認してください。
6. イメージセンサーからのフレームバリデーション (FV) とラインバリデーション (LV) の信号が、選択された解像度とフレームレートに従って正しく動作しているかどうかを確認してください。イメージセンサーのインタフェース要件については、「イメージセンサーのインタフェース要件」を参照してください。
7. EZ-USB™ FX3 が受信したフレームサイズが、特定の解像度の USB ディスクリプタで報告されたフレームサイズと同じかどうかを確認してください。
8. ビデオプローブとコミット制御パラメータ dwMaxPayloadTransferSize と dwMaxVideoFrameSize が選択した解像度に対して正しく設定されているかを確認してください。
9. CyU3PDmaSocketSetWrapUp は 4 の倍数のデータしかコミットできないため、観測された部分 DMA バッファサイズが 4 の倍数であるかどうかを確認してください。
10. GPIF のステートマシンで使用されているカウンターが正しく設定されているかを確認してください。
11. ビデオフォーマット GUID (guidFormat) がフォーマットディスクリプタに正しく設定され、ホスト OS の UVC ドライバーがサポートしているかどうかを確認してください。
12. イメージ データの総量が正しく、ホスト アプリケーションでイメージが表示されない場合、異なるホスト コンピュータと見なします。
13. **Debug UVC application firmware in EZ-USB™ FX3 – KBA226722** を参照してください。

8.2 ホストアプリケーションで観測されるフレーム/秒の減少

1. イメージセンサーからのフレームバリデーション (FV) とラインバリデーション (LV) の信号が、選択された解像度とフレームレートに従って正しく動作しているかどうかを確認してください。
2. UART デバッグプリントでコミットバッファの失敗が観察されるかどうかを確認してください。これらの失敗の処理方法の詳細については、**Handling commit buffer failures occurred during video transfers using EZ-USB™ FX3 – KBA231382** を参照してください。
3. 垂直ブランキング時間が 200 µs 以上であるかどうか確認してください。

問題がまだ解決されない場合は、**サイプレス開発者コミュニティ**で同様のケースを参照してください。

9 2 個のイメージ センサー同士の接続

3 次画像処理またはモーション トラッキングなどのホスト アプリケーションは、EZ-USB™ FX3 が 2 個のイメージ センサーからビデオデータを同時にストリームすることを必要とします。センサーが異なった場合、フォーマットを調整し、単一のビデオ データ チャネルを合成するために、イメージ センサー モジュールと EZ-USB™ FX3 間に FPGA を挿入しなければなりません。2 個の異なったイメージ センサーのそのようなセットアップは、本アプリケーション ノートの範囲外です。

より実用的なアプローチは、同一のイメージ センサーを使用することです。このアプローチを本節に示します。

Figure 52 に接続の詳細を示します。緑色のブロックは、GPIF II 内部にあり、赤色ブロックが EZ-USB™ FX3 低帯域幅ペリフェラル (I²C/GPIO) の一部です。その目的は、2 個のセンサーを同期化して同一のフレーム タイミングを達成し、GPIF II が 16 ビットのデータ バス上で各 8 ビット ビデオ ストリームを同時に入力することです。

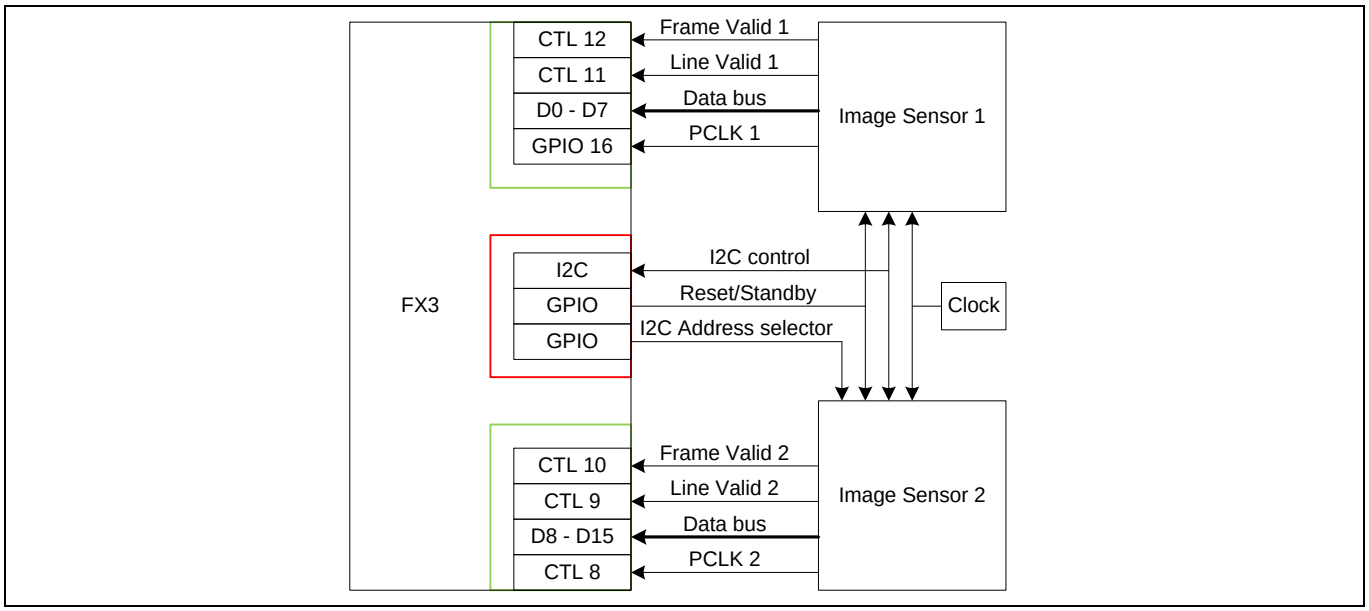


Figure 52 2 個のイメージ センサーを EZ-USB™ FX3 に接続

Figure 52 は 2 個のイメージ センサーで以下の状態を前提にします:

- 各イメージ センサーのバス幅は 8 ビットであるため、GPIF II バス幅は 16 ビットになります。
- 2 個のイメージ センサーが同期されます。これは、両方のイメージ センサーが同じであるクロック、LV と FV 遷移、およびピクセル タイミングを使用してることを意味します。言い換えれば、両方のイメージ センサーからのフレームを正確に重ね合わせられます。いくつかのイメージ センサーには 2 つのビデオ ストリームを同期させるために使用できる外部トリガー入力を備えています。他のイメージ センサーは、別の同期方法を使用することもあります。詳細は該当するセンサー データシートを参照してください。
- 両方のイメージ センサーは、I²C を設定に使用します。本アプリケーション ノートで使用されるイメージ センサーは、イメージ センサー モジュール間の同期を達成することを目的として、I²C 制御レジスタが正確に同時に書き込まれることを必要とします。これは、EZ-USB™ FX3 GPIO ピンを使用して 2 個のセンサーの内のいずれかの I²C アドレスを制御することにより達成されます。EZ-USB™ FX3 は I²C の書き込みを介して同時に 2 個のイメージ センサーを設定します。EZ-USB™ FX3 は設定可能なアドレス ピンを使ってイメージ センサー上の異なる I²C アドレスに切り替えることにより、個々のセンサーから読み出します。

- 各センサーのリセット信号は、同じ EZ-USB™ FX3 GPIO 出力ピンにより駆動する必要があります。同様に、各センサーのスタンバイ ピンが存在する場合、別の EZ-USB™ FX3 GPIO 出力ピンを共有する必要があります。
- 自動設定はイメージ センサーにおいて無効にされます。例えば、自動露光、オート ゲイン、オート ホワイト バランスなどの機能は、両方のセンサーにおいてオフにされます。これらのオフにより、イメージ センサー上の統合と任意のイメージ処理は同じ時間を要することが保証されます。その結果、両方のセンサーからのフレームは同時にイメージ センサーから出力されます。

接続図に示していとおり、フレーム有効 2、ライン有効 2 および PCLK 信号は EZ-USB™ FX3 に接続されていますが、イメージ センサーが同期されることを前提にしているため、これらは GPIF II ブロックによって使用されません。これらの信号が FX3 に接続されているため、EZ-USB™ FX3 は信号を監視してデバッグおよび開発時にイメージ センサー同士間の同期の精度をチェックできます。

9.1 UVC によるインタリーブされたイメージの転送

UVC 仕様は 1 つのイメージを送信するのに使用されるイメージ センサーの数を識別できません。これは、フレームの幅、フレームの高さおよびフレームあたりの総バイト数のイメージ パラメータの 1 組だけを扱います。複数のイメージ センサーに対応するために、UVC ドライバーは内部整合性の試験を実行してそれに合格できるように、修正されたディスクリプタを読み出さなければなりません。これらの整合性チェックに失敗すると、UVC ドライバーはイメージ データをホスト アプリケーションに渡せず、アプリケーションは失敗します。UVC ドライバーが追加フレームをシングル イメージ センサーとして認識するようにディスクリプタを修正する必要があります。

取り扱い方の詳細は以下の 2 つの例に示しています。

9.1.1 例 1: 2 個の 640x480 モノクロ センサー

2 個のイメージ センサーは 640×480 モノクロ (ピクセル当たり 1 バイト) のデータを提供します。**Figure 53** に示すように、EZ-USB™ FX3 はフレームごとに 2 つの完全なイメージを同時に受信します。

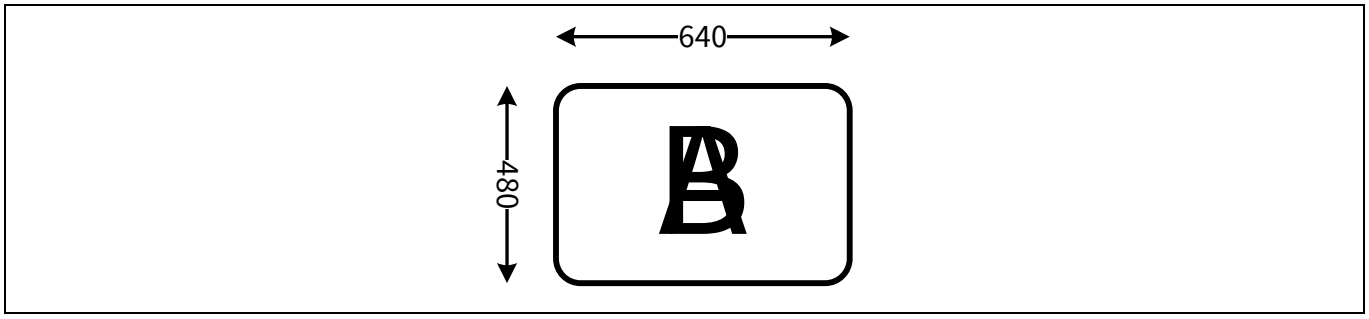


Figure 53 EZ-USB™ FX3 が 2 つのイメージ センサーから受信するもの

ディスクリプタはまだ 640x480 イメージ サイズを報告します。ダブル データ サイズは、A イメージを Y データに入れ、B イメージを U と V のデータに入れて調整されます。

フレームあたりのバイト数は以下のように計算できます。

フレーム当たりのバイト数=ピクセル当たりのバイト数x イメージ センサー数×解像度

ここでは、

解像度 = 幅 (ピクセル単位) x 高さ (ピクセル単位)

例:

フレーム当たりのバイト数=1x2x640x480=614,400 バイト

9.1.2 例 2: 2 個の 640x480 カラー センサー

2 つのカラー センサーが 640×480 の解像度で YUY2 データを送信することを前提にします。EZ-USB™ FX3 はフレームごとに 2 つの完全なカラー イメージを受信します (**Figure 54**)。

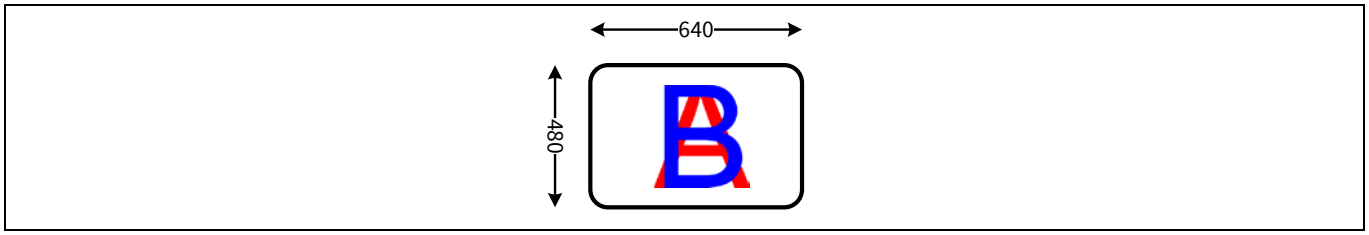


Figure 54 EZ-USB™ FX3 が 2 枚のカラー イメージを受信

例 1 との唯一の違いは、YUY2 形式の各ピクセルが 1 つではなく 2 つのバイトを使用することです。ピクセルの倍増に対応するために、報告されたイメージ サイズも倍になります (**Figure 55**)。

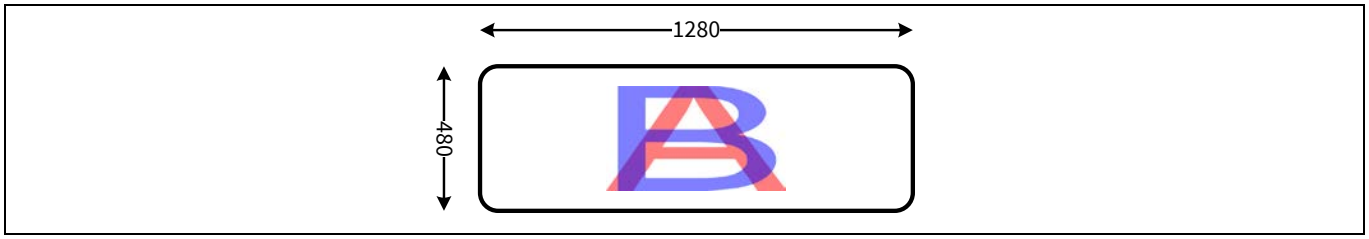


Figure 55 報告されるイメージ サイズ

各ピクセルが 2 バイトを占めるため、1 フレームあたりのデータは次のようになります:

フレームあたりのバイト数=2x2x640x480=1,228,800 バイト

任意のフレームの幅と高さは、それらがピクセルの倍増を反映する限り報告されることに注意してください。1 次元だけを 2 倍にすると、ダウンストリームの計算が簡単になります。高さを変更せずに幅を倍にすることは、垂直線を重ね合わせる利点をもたらします。これによりアプリケーション イメージ処理は簡素化されます。

これらのディスクリプタ変更は、2 重イメージが任意の 2 つの連続バイトが異なる画像センサーから送信されるインタリーブ方式でイメージ センサーからホスト アプリケーションに搬送することを可能にします。

上記の変更は UVC ドライバーの整合性チェックをパスし、ドライバーがビデオ データをホスト アプリケーションに渡すことを可能にします。この方法でストリームされたビデオは、直接見たときに理解できることを意味しません。標準の UVC ホスト アプリケーションを使用し、実装の健全性チェックを実行できます。しかし、イメージは非標準的な方法でストリームされるため、アプリケーションは正しく表示しません。これらのイメージを分離し、インタリーブされたビデオから有用な情報を表示および計算するために、カスタム アプリケーションを作る必要があります。

9.2 現在のプロジェクトに新しいビデオフォーマットを追加するファームウェア修正チェックリスト

概要として、ファームウェアを変更して新しいビデオフォーマットを追加する場合は、以下の手順を実行して必要な変更がすべて行われていることを確認してください:

1. 新しいフレーム ディスクリプタを追加し、High-speed および SuperSpeed USB コンフィギュレーション ディスクリプタを更新します。フォーマット記述子のパラメータの詳細については、USB ビデオ ペイロード仕様書を参照してください。
2. クラス固有のインタフェース入力ヘッダー記述子の bNumFormats と wTotallength フィールドを更新してください。
3. 新しい解像度を追加する方法については、**9.3 節**を参照してください。

9.3 現在のプロジェクトに新しいビデオ解像度を追加するファームウェア修正チェックリスト

概要として、ファームウェアを変更して新しい解像度を追加する場合は、以下の手順を実行して必要な変更がすべて行われていることを確認してください:

1. 使用可能なフォーマット ディスクリプタに新しいフレーム ディスクリプタを追加し、High-speed および SuperSpeed USB コンフィギュレーション ディスクリプタを更新します。新しいフレーム ディスクリプタは次の新しいビデオ解像度の詳細を反映する必要があります:フレーム インデックス、幅解像度、高さ解像度、フレーム レート、最大ビット レート、最小ビット レートおよびアスペクト比。High-Speed および SuperSpeed USB コンフィギュレーション ディスクリプタの長さおよびビデオ ストリーム入力ヘッダー ディスクリプタの長さを更新し、サポートされているすべてのビデオ フレーム ディスクリプタを含めます。
2. 追加されたビデオ フレーム解像度に対する PROBE/COMMIT 制御構造を追加します。新しく追加されたビデオ フレーム解像度の SET_CUR および GET_CUR 要求を処理します。
3. 新しく追加されたビデオ フレーム解像度を設定するセンサー制御コマンドを追加します。

9.4 新しいセンサーをサポートするためのファームウェア変更のチェックリスト

概要として、ファームウェアを変更して新しいセンサーをサポートする場合は、以下の手順を実行して必要な変更がすべて行われていることを確認してください:

1. イメージ センサー制御: この例では、イメージ センサーにコマンドを送信するための制御インタフェースとして基準 I2C コードがあります。これは修正が必要な場合や、新しいイメージ センサーでサポートされる異なる種類のインタフェースが必要になる場合があります。
2. 制御インタフェースのためのイメージ センサー セレクタとして機能する GPIO を制御するコードを追加します。EZ-USB™ FX3 が本アプリケーション ノートで使用されるセンサーに書き込む場合 (**Figure 52** 参照)、センサーに書き込まれたメッセージは両方のイメージ センサーへのマルチキャスト メッセージですが、FX3 はイメージ センサーから I2C レジスタを読み出す時、排他的アクセスを必要とします。
3. イメージ センサーのスタンバイまたは低消費電力モードを制御する GPIO を制御するコードを追加します。
4. フレームおよびフォーマット用の UVC 固有の High-speed と SuperSpeed USB コンフィギュレーション ディスクリプタを変更します。これはサポートされているビデオ フォーマット、およびサポートされているすべてのビデオ フレーム解像度のフレーム インデックス、幅解像度、高さ解像度、フレーム レート、最大ビット レート、最小ビット レートおよびアスペクト比を報告するためのものです。

5. High-Speed および SuperSpeed USB コンフィギュレーション ディスクリプタの長さおよびビデオ ストリーム入力ヘッダー ディスクリプタの長さを更新し、追加されたすべてのビデオ フレーム ディスクリプタを含めます。
6. サポートされている各ビデオ解像度に対する PROBE/COMMIT 制御構造を追加します。すべてのビデオ フレーム解像度に対する SET_CUR および GET_CUR 要求を処理します。
7. GPIF II ディスクリプタを変更し、バス幅を増やし、バス幅に応じてカウンター制限を更新します。これらの変更はヘッダー ファイルを生成するために GPIF II Designer で作られます。この新しく生成されたファイルをプロジェクトにコピーし、変更が有効になるようにします。

10 まとめ

本アプリケーションノートでは、USB ビデオ クラスに準拠したイメージ センサーは、どのようにインフィニオン EZ-USB™ FX3 を使用して実装できるかを説明します。特に、以下のことを示します:

- ホスト アプリケーションとドライバーが 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
- 標準イメージ センサーからデータを受信するために GPIF II Designer を使用して EZ-USB™ FX3 インタフェースをプログラムする方法
- ホスト アプリケーションでビデオ 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 インタフェースを追加する方法
- オープンソース ホスト アプリケーション プロジェクトを含むさまざまなプラットフォームで利用可能なホスト アプリケーションの検索方法
- UVC で複数イメージ センサーを接続したり、外部で同期化させたり、それらセンサーから同期にストリームしたりする方法
- 必要に応じて、EZ-USB™ FX3 ファームウェアの問題を修正しデバッグする方法

11 付録 A: EZ-USB™ FX3 DVK (CYUSB3KIT-001) のハードウェア設定詳細

11.1 EZ-USB™ FX3 DVK 基板の設定

ビデオ アプリケーション テスト用基板の準備は、以下の手順に従います。

1. **Figure 56** に示すように、EZ-USB™ FX3 DVK 基板上のジャンパを設定します。図で強調表示されていないジャンパをセットしないでください。

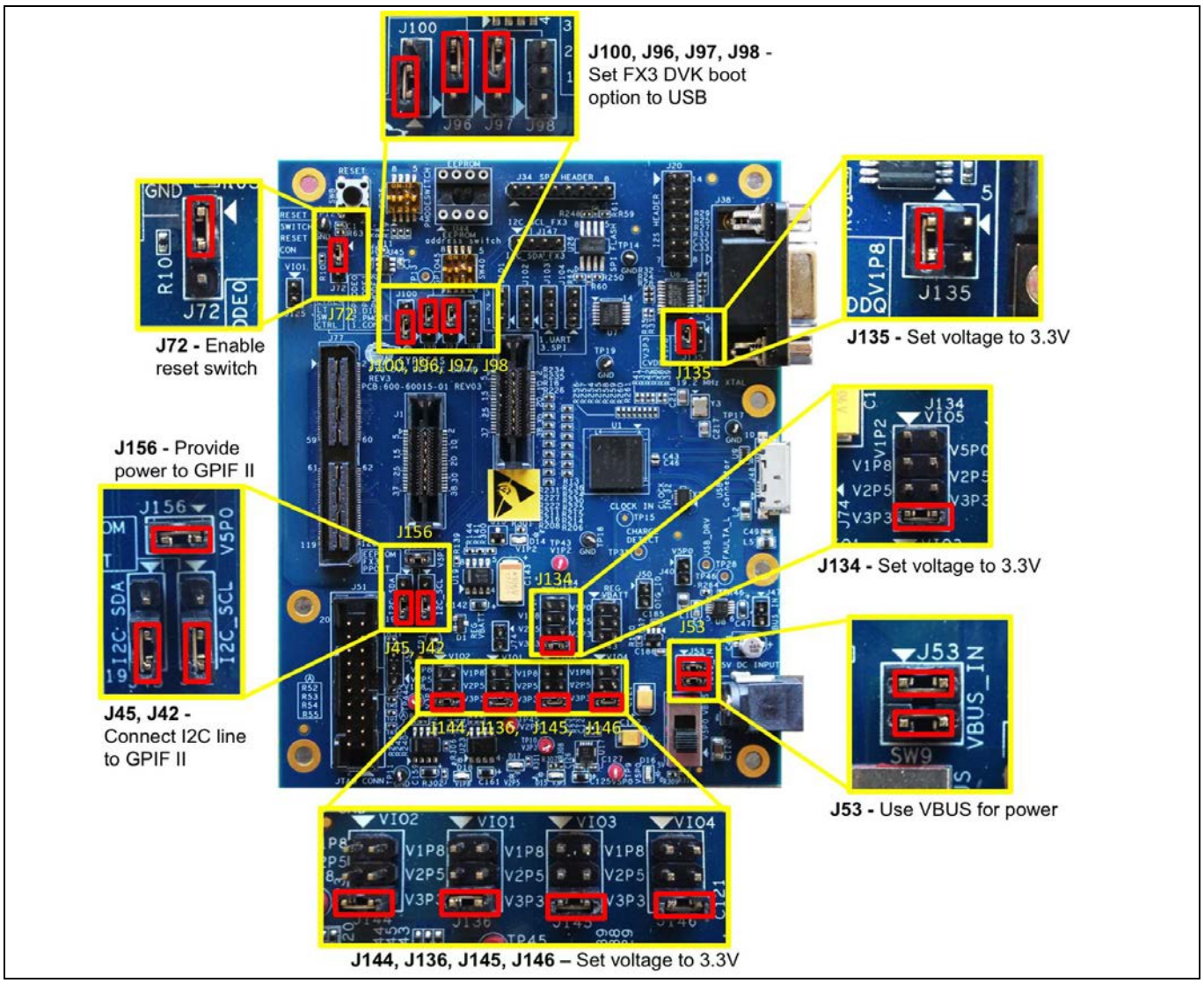


Figure 56 EZ-USB™ FX3 DVK ジャンパ

2. Aptina Image Sensor 相互接続基板 (CYUSB3ACC-004) を EZ-USB™ FX3 DVK J77 に差し込みます。相互接続基板上のコネクタの種類は一意で、ソケットは正しい向きでのみ接続できるようにキーが付けられています。
3. 相互接続基板に Aptina Image Sensor モジュールを接続します。**Figure 57** に 3 個の基板のアセンブリを示します。

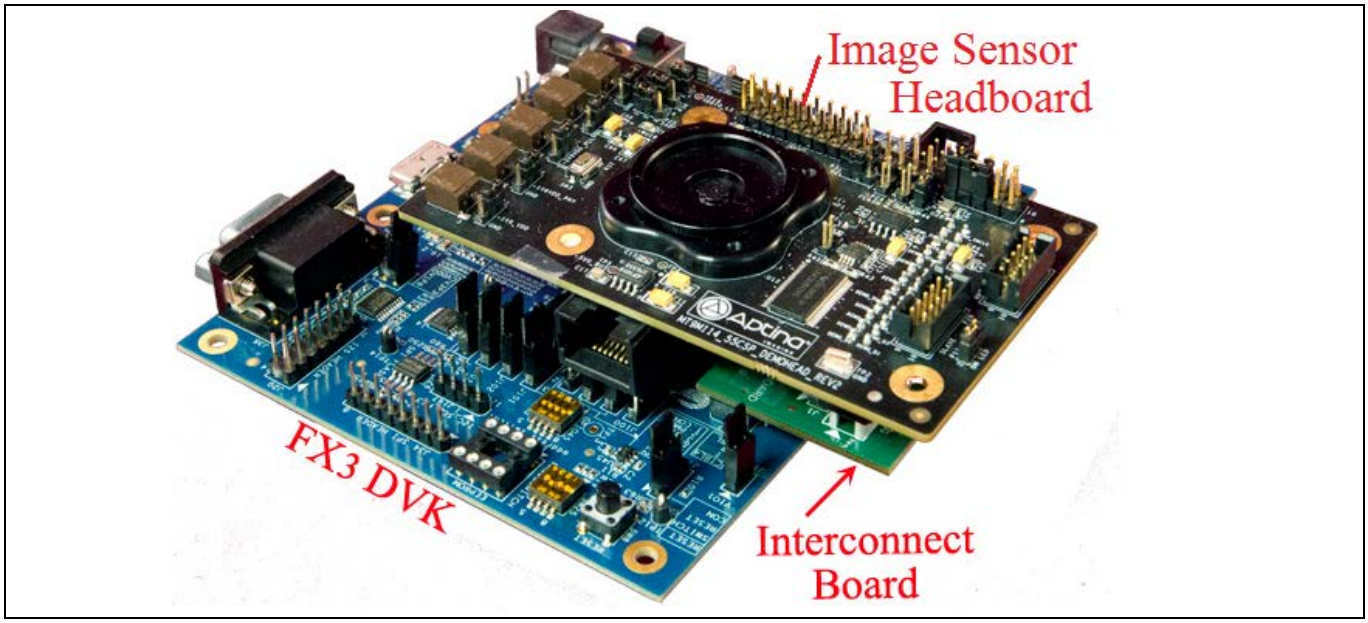


Figure 57　3-Board 720p カメラアセンブリ

4. DVK に付属する USB 3.0 ケーブルを使用し、EZ-USB™ FX3 DVK 基板をコンピュータの USB ポートに接続します。
5. EZ-USB™ FX3 SDK の一部として提供された Control Center アプリケーションを使用し、基板にファームウェアをロードします。ファームウェアソースとビルド済みイメージは、**AN75779.zip** で提供されます。詳細な手順は **AN75705, Getting Started with EZ-USB™ FX3** を参照してください。操作概要は以下のとおりです。

Figure 58　FX3 がブートローダとして列挙

a) Control Center アプリケーションを起動します。EZ-USB™ FX3 DVK を接続すると、インフィニオン EZ-USB™ FX3 USB ブートローダとして認識されます (**Figure 58** を参照してください)。

b) **Program** > **FX3** > **RAM** を選択し、本アプリケーション ノートに添付される *cyfx_uvc_an75779.img* ファイルへ移動します (を参照してください)。デバイスは、プログラミング後に電源を入れ直した場合、ロードされたファームウェアを失い、インフィニオン EZ-USB™ FX3 USB ブートローダデバイスとして再列挙されます。

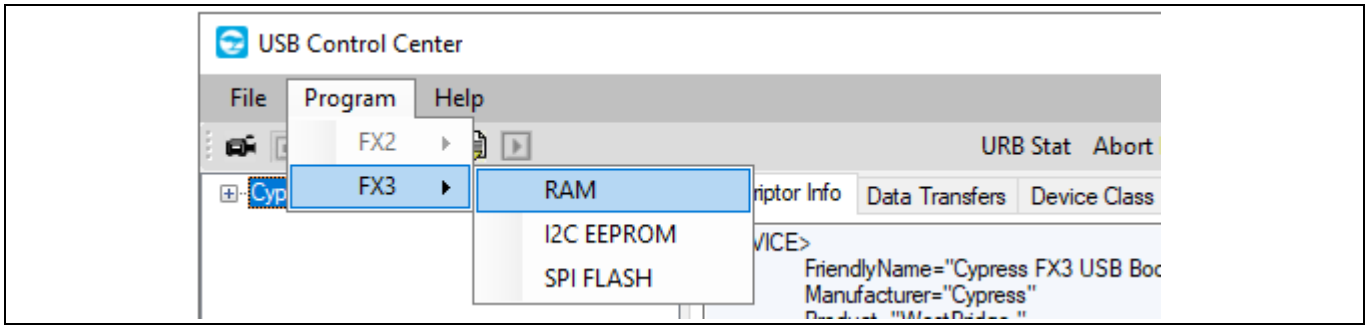


Figure 59　FX3 RAM へのコードのロード

c) Control Center アプリケーションは、ステータスバーにプログラミングのステータスを表示します。プログラミングが成功すると、デバイスは UVC クラスデバイスとして列挙されるため、コントロールセンターのデバイスリストからも消えます。デバイスは、Windows デバイスマネージャーの Cameras または Imaging Devices タイプで確認できます (**Figure 60**)。

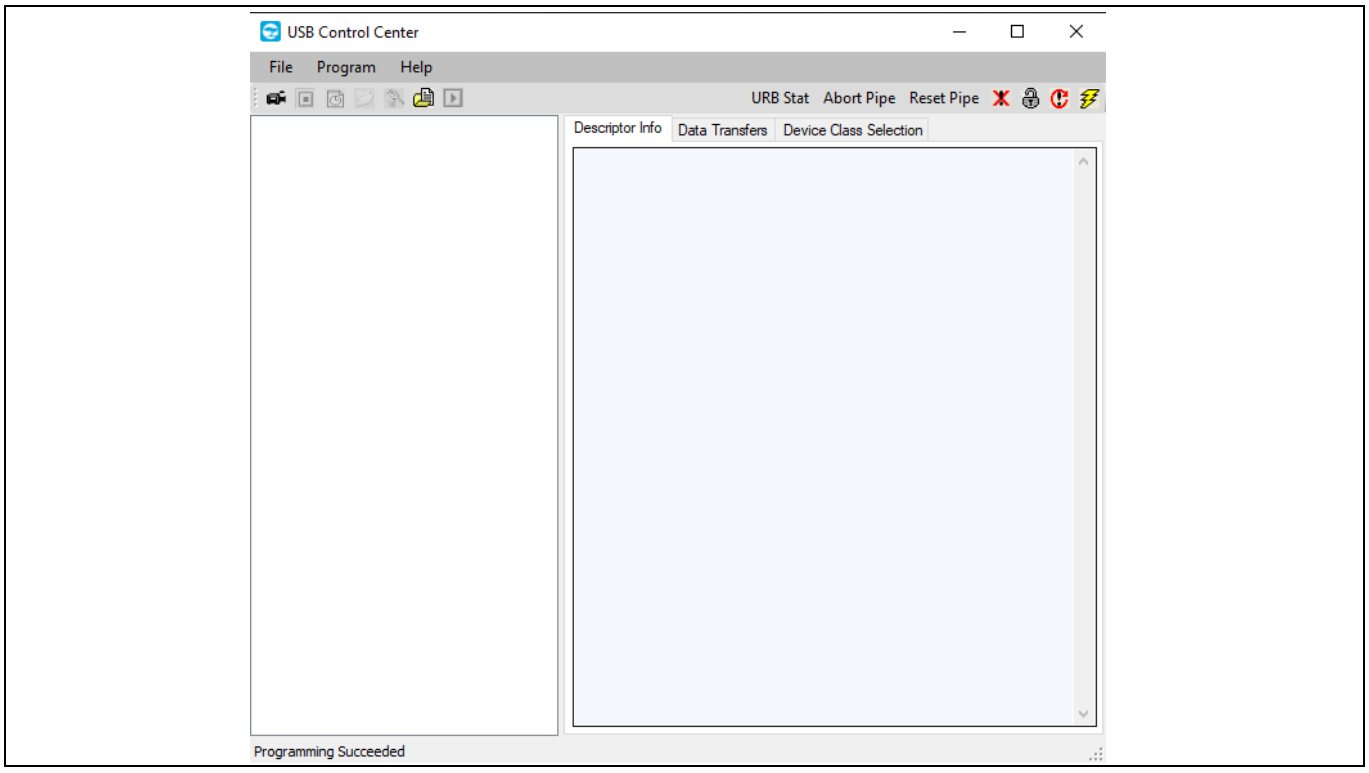


Figure 60　ステータスバーに表示されるプログラミング成功メッセージ

12 付録 B: EZ-USB™ FX3 Explorer キットおよび Aptina Image Sensor 相互接続基板 (CYUSB3ACC-004) のハードウェア設定詳細

12.1 ハードウェア設定

1. EZ-USB™ FX3 Explorer Kit でジャンパ J2、J3、および J4 を閉じ、ジャンパ J5 を開きます。
2. Aptina Interconnect Board Quick Start Guide の指示に従って、SuperSpeed Explorer キット、相互接続ボード、および Aptina Image Sensor 基板を組み立てます。**Figure 61** にアセンブリを示します。

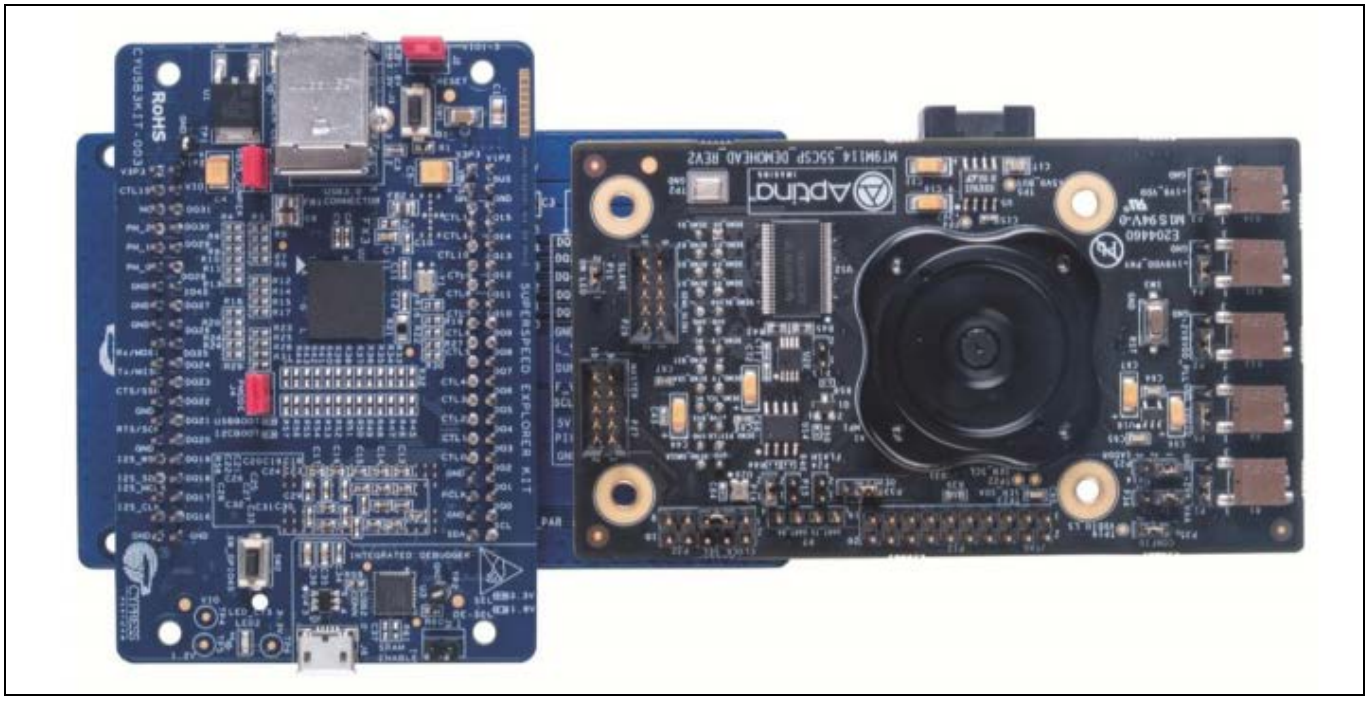


Figure 61 SuperSpeed Explorer キット、Aptina 相互接続、および Aptina センサー基板アセンブリ

3. キットに付属の USB 3.0 ケーブルを使用して、EZ-USB™ FX3 Explorer キット基板をコンピュータの USB ポートに差し込みます。
4. EZ-USB™ FX3 SDK の一部として提供される Control Center アプリケーションを使用して、ボードにファームウェアをロードします。ファームウェアソースとビルド済みイメージは、**AN75779.zip** で提供されます。詳細な手順は **AN75705, Getting Started with EZ-USB™ FX3** を参照してください。操作概要は以下のとおりです。

Control Center アプリケーションを開始します。EZ-USB™ FX3 Explorer 基板を接続すると、インフィニオン EZ-USB™ FX3 USB ブートローダデバイスとして認識されます (**Figure 62**)。

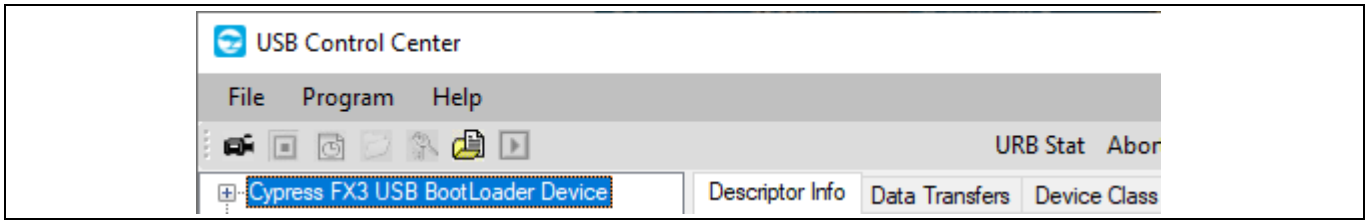


Figure 62 EZ-USB™ FX3 がブートローダとして列挙

Program > **FX3** > **RAM** を選択し、本アプリケーション ノートに添付される *cyfx_uvc_an75779.img* ファイルへ移動します (を参照してください)。デバイスは、プログラミング後に電源を入れ直した場合、ロードされたファームウェアを失い、インフィニオン EZ-USB™ FX3 USB ブートローダデバイスとして再列挙されます。

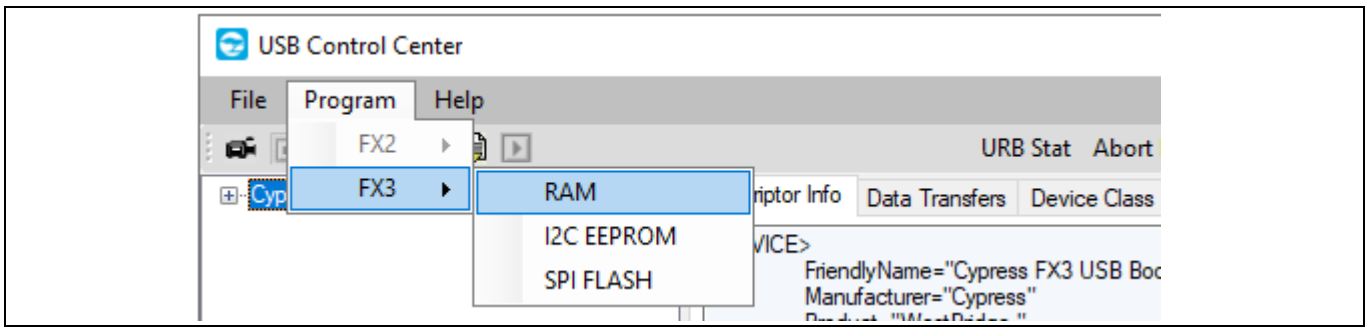


Figure 63 EZ-USB™ FX3 RAM へのコードのロード

Control Center アプリケーションは、ステータスバーにプログラミングのステータスを表示します。プログラミングが成功すると、デバイスは UVC クラスデバイスとして列挙されるため、コントロールセンターのデバイスリストからも消えます。デバイスは、Windows デバイスマネージャーの Cameras または Imaging Devices タイプで確認できます (**Figure 64**)。

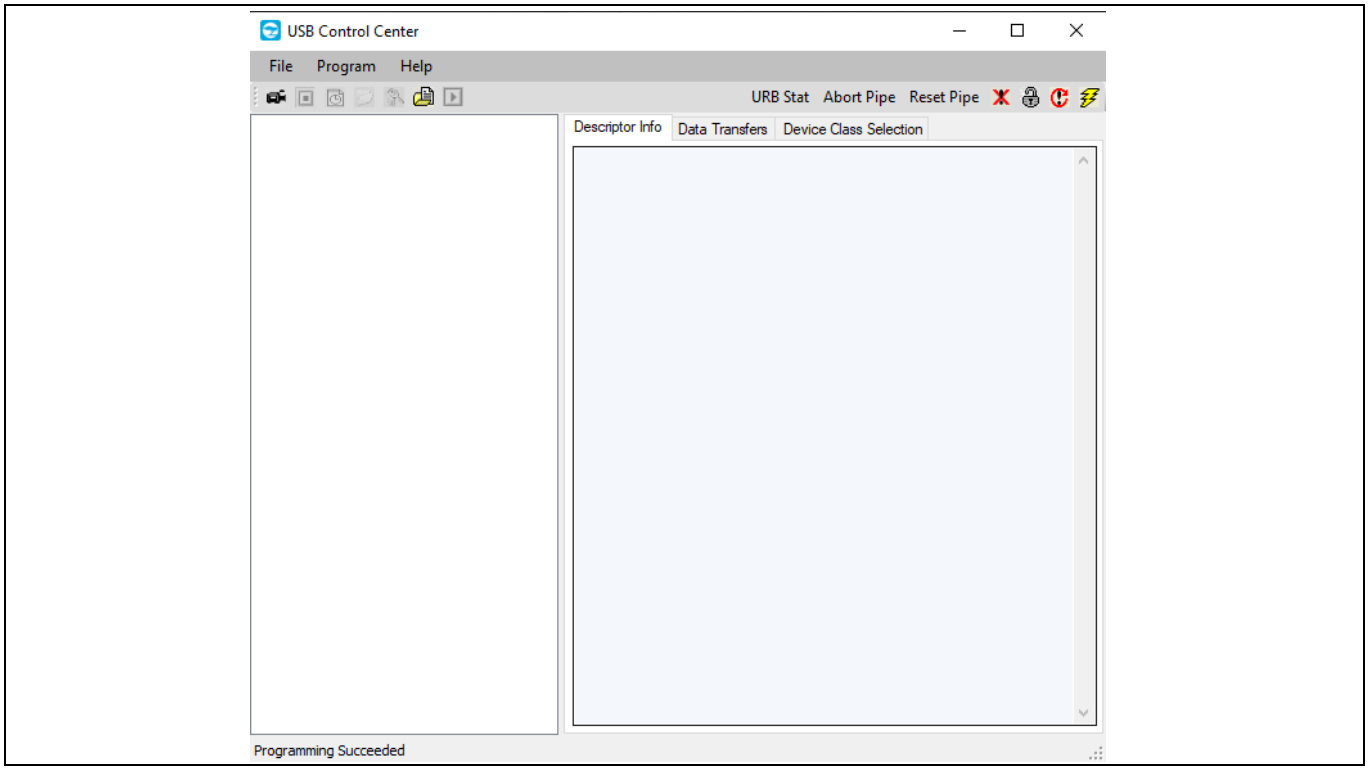


Figure 64 ステータスバーに表示されるプログラミング成功メッセージ

参考資料

[1] **AN75705**

[2] **AN90369**

改訂履歴

Document version	Date of release	Description of changes
**	2014-05-29	これは英語版 001-75779. Rev *D を翻訳した日本語版 Rev. **です。
*A	2017-04-27	Updated logo and copyright.
*B	2017-12/21	これは英語版 001-75779. Rev *J を翻訳した日本語版 Rev. *B です。
*C	2019-12-11	これは英語版 001-75779. Rev *K を翻訳した日本語版 Rev. *C です。
*D	2022-04-19	これは英語版 001-75779. Rev *L を翻訳した日本語版 Rev. *D です。

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-04-19
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
001-92696 Rev. *D

重要事項
本文書に記載された情報は、いかなる場合も、条件または特性の保証とみなされるものではありません（「品質の保証」）。本文に記された一切の事例、手引き、もしくは一般的価値、および／または本製品の用途に関する一切の情報に関し、インフィニオンテクノロジーズ（以下、「インフィニオン」）はここに、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

さらに、本文書に記載された一切の情報は、お客様の用途におけるお客様の製品およびインフィニオン製品の一切の使用に関し、本文書に記載された義務ならびに一切の関連する法的要件、規範、および基準をお客様が遵守することを条件としています。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

本製品、技術、納品条件、および価格についての詳しい情報は、インフィニオンの最寄りの営業所までお問い合わせください（**www.infineon.com**）。

警告事項
技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。