参 考 資 料

JAJA174 – 2008年08月

TMS320DM643x デバイスにおける VPFE及びVPBE ドライバの使い方

アプリケーション技術部

アブストラクト

本資料は、TMS320DM643xデバイスに搭載されているビデオ・プロセッシング・バック・エンド(VPBE)及びビデオ・プロセッシング・フロント・エンド(VPFE)ドライバの使い方について記載しています。VPBEブロックはオンスクリーン・ディスプレイ(OSD)とビデオ・エンコーダ(VENC)モジュールで構成されています。VPFEブロックはCCDコントローラ(CCDC)やリサイザ、プレビュー、ハードウェア3A統計処理ジェネレータ(H3A)、ヒストグラムの5つのモジュールで構成されますが、本資料で紹介するドライバではCCDCのみサポートします。VPFE内の他のモジュールのプログラミングは、それぞれ特定のドライバにてサポートされます。本資料に記載されていないドライバにおける詳細な情報は、デバイス固有の資料をご覧ください。

VPBE及びVPFEドライバは、製品登録されたお客様であれば、http://www.ti.com/davincisoftwareupdates から最新のソフトウェア: デジタル・ビデオ・ソフトウェア・デベロップメント・キット(DVSDK)がダウンロード可能です。
本資料に記載されているプロジェクトやソースコードは、http://www-s.ti.com/sc/techlit/spraap3.zipからダウンロード可能なプロジェクト・コードを使用しています。

目次

図

表

SPRAAP3A 翻訳版
最新の英語版資料
http://www-s.ti.com/sc/techlit/spraap3a.pdf

1 はじめに

本資料では、2つの例題を取り上げています。1つ目はVPBEドライバの使い方、2つ目はビデオデコーダからYUV422データをキャプチャするためのVPFEドライバの使い方に関するデモンストレーションです。

本資料には、VPBE及びVPFEハードウェアの詳細な説明はありませんが、重要な概念は例と共に説明されます。

さらに、VPBE及びVPFEドライバのアプリケーション・プログラミング・インターフェイス(API)の詳細説明もありません。VPBE/VPFEドライバのAPIに関する詳細な情報は、VPBE/VPFEドライバ・ユーザーズ・ガイドをご覧ください。本資料は実用例を通じてVPBE/VPFEユーザーズ・ガイドに対して充実したサービスを提供します。VPBE及びVPFEに関するより詳細な情報については、デバイス固有のユーザーズ・ガイドをご覧ください。

1.1 VPBE と VPFE の概要

ここでは、TM320DM643xデバイスのVPBE及びVPFEハードウェアの重要な技術的コンセプトについて説明します。VPBE及びVPFEそれぞれの詳細な情報は、TMS320DM643x DMP Video Processing Back End (VPBE) User's Guide (SPRU952) [1]、TMS320DM643x DMP Video Processing Front End (VPFE) User's Guide (SPRU977) [2] をご覧ください。

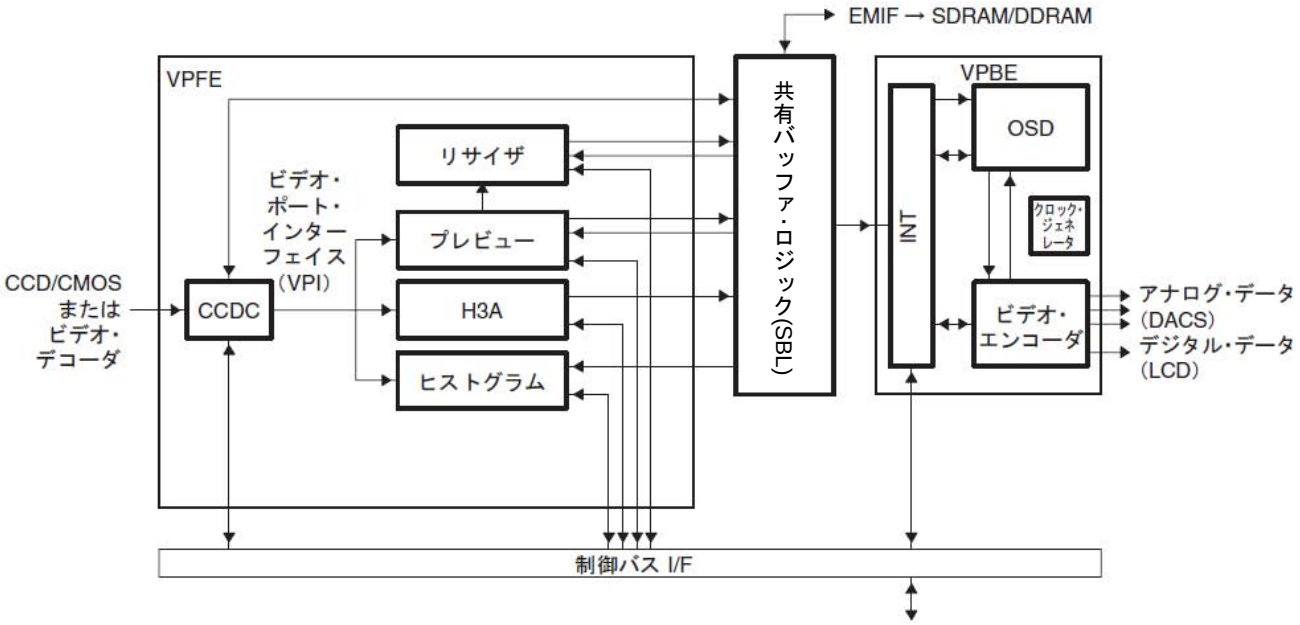

図 1. VPSS ブロック図

ビデオ・プロセッシング・サブシステム(VPSS)を図1に示します。VPSSは、イメージセンサーやビデオ・デコーダなどといった外部イメージング・ペリフェラル用の入力インターフェイス(VPFE)と、アナログSDTVディスプレイやデジタルLCDパネル、HDTVビデオ・エンコーダなどといったディスプレイデバイス用の出力インターフェイス(VPBE)を提供しています。これらのペリフェラルに加え、DDR2メモリ制御バースト・バンド幅を効率よく使用するために、共通のバッファメモリとダイレクト・メモリ・アクセス(DMA)のセットを持っています。共有バッファ・ロジック/メモリは、DDR2メモリ・コントローラへ/からのデータのリクエストまたは転送のどちらかを行なう全てのVPFEおよびVPBEモジュールに対する主要なソースまたはシンク(受信口)として動作します。

OSD及びVENCモジュールは、強力かつ柔軟なディスプレイ・インターフェイスを提供します:

- オンスクリーン・ディスプレイ(OSD)グラフィック・アクセラレータ：OSD は、各種のハードウェア・ディスプレイ・ウィンドウに対応した様々なフォーマットでディスプレイ・データを管理し、ディスプレイ・ウィンドウを 1 つのディスプレイ・フレームに組み合わせて処理します。その後、データはビデオ・エンコーダ(VENC)モジュールによって出力されます。OSDがサポートしている主要な機能は次のとおりです。
  - 同時表示可能な2つのビデオ・ウィンドウ及び2 つの OSD ウィンドウのサポート

(VIDWIN0/VIDWIN1 とOSDWIN0/OSDWIN1)
- ウィンドウごとに個別のイネーブル / ディスエーブル制御、プログラム可能なウィンドウサイズ、表示位置、外部メモリ･アドレスとオフセット･レジスタ等
- ウィンドウごとに水平方向と垂直方向の両方に対して 2 倍および 4 倍ズームをサポート
- OSDWIN1 を、OSDWIN0 のアトリビュート･ウィンドウに設定することが可能
- ウィンドウ（インターレース / プログレッシブ）に対してフィールド / フレーム･モードのいずれかを選択可能
- OSD とビデオ･ウィンドウ間で 8 段階のブレンディング処理
- OSD とビデオ･データの透過性サポート(ビットマップとビデオ間でのバックグラウンド･カラーと一致するピクセルのみのブレンディング)
- それぞれのウィンドウに対して、VGAからNTSC/PAL (640x480 から 720x576 へ)にリサイズする能力
- OSDウィンドウ(いずれか一方、同時に2つは不可)は、最大8ビットに制限されたYCbCrフォーマットのビットマップ･データではなく、16-bitフォーマットのRGBデータの表示が可能
- OSDビットマップ･データ幅は、1, 2, 4, 8bitのいずれかを選択可能
- 各OSDウィンドウはビットマップ用に16エントリをサポート(256エントリの RAM/ROM CLUTテーブルにインデックスするため)
- VPBE のラッパー･インターフェイスを介した 24 ビットの RGB 入力データ（これは 16 ビット YCbCr ビデオ･ウィンドウ･データに変換される）の間接サポート
- 256 色をサポートする RAM/ROM テーブル間で選択できる能力のあるプログラム可能なカラー・パレット。すべてのウィンドウに対して同時に選択可能なウィンドウには 2 つの ROM テーブルを所有
- カーソルの幅、高さ、カラーは、いずれも選択可能
- ディスプレイ優先順位は、矩形カーソル > OSDWIN1 > OSDWIN0 > VIDWIN1 > VIDWIN0 > バックグラウンド･カラーの順

- ビデオ･エンコーダ (VENC)は、オンスクリーン･ディスプレイ(OSD)からディスプレイ･フレームを取得し、ディスプレイ･デバイスとのインターフェイスに必要な指定出力形式や出力信号(データ、クロック、同期などを含む)にフォーマットします。VENC は、次の 3つの主要なサブブロックから構成されています。
  - NTSC/PAL 方式のテレビ･ディスプレイとインターフェイスするための信号を生成するアナログ･ビデオ･エンコーダ
  - 高品位ビデオ･エンコーダと DVI/HDMI インターフェイス･デバイスの両方またはそのいずれかとインターフェイスするために、標準デジタル YUV 出力だけでなく、さまざまなデジタル LCD ディスプレイ･フォーマットとのインターフェイスをサポートするデジタル LCD コントローラ
  - さまざまなデジタル･ビデオ出力モードだけでなく、アナログ･ビデオ出力に必要な固有のタイミングを生成するタイミング・ジェネレータ

VPFEは、CCDC、プレビュー･エンジン･イメージ･パイプ(IPIPE)、H3A、リサイザ及びヒストグラムブロックから構成されます。同時に、これらのモジュールは、強力かつ柔軟なフロントエンド･インターフェイスを提供します。ただし、既存のVPFEドライバはCCDCのみサポートされます。他のVPFEモジュールのプログラミングにおいては、本資料に記載されていない他の特定のドライバによってサポートされます。

- CCDCは、センサー(CMOSまたはCCD)からの raw (加工されていない) イメージ/ビデオ･データを受け付けます。また、CCDCは、通常、ビデオ･デコーダ･デバイスからのさまざまなフォーマットのYUVビデオ･データを受け付けます。raw入力の場合、CCDコントローラ出力では、raw入力イメージを最終的に処理されたイメージに変換するためにさらにイメージ処理が必要です。このイメージ処理は、プレビューエンジン、またはソフトウェアのいずれかで高速に行なわれます。CCDコントローラは、コントロール･レジスタ及びパラメータ･レジスタを使用してプログラムされます。CCDコントローラ･モジュールがサポートしている機能は次のとおりです。
  - 従来のBayerパターン･センサー･フォーマット
  - 外部タイミング･ジェネレータへのHD/VDタイミング信号及びフィールドIDの生成または同期
  - プログレッシブ･センサー及びインターレース･センサーをサポート(最大で2フィールドまでのハードウェア･サポート)
  - 最大で75MHzまでのセンサー･クロックをサポート
  - REC656/CCIR-656規格(8ビットまたは16ビットのYCbCr422フォーマット)をサポート
  - 個々のHSYNC及びVSYNC信号を含む8ビットまたは16ビットのYCbCr422フォーマットをサポート
  - 最大で16ビット入力をサポート

- オプティカル・ブラック・クランピング信号の生成
- シャッター信号制御をサポート
- デジタル・ラインピング及びブラック・レベル補正機能をサポート
- 10ビットから8ビットへのA-law補正をサポート
- SDRAMへライトする前のローパス・フィルタをサポート．このフィルタがイネーブルの場合、各ラインの左右両端の2ピクセルがそれぞれ出力からクロップ
- 16ビット幅から8ビット幅までの出力の生成をサポート(8ビット幅があれば、保存領域を50%節約可能)
- プログラム可能なカリング・パターンを介してダウンサンプリングをサポート
- 外部ライト・イネーブル信号を介してDDR2への出力を制御できる機能
- 水平方向及び垂直方向の両方へ最大で32Kピクセル(イメージ・サイズ)までサポート

1.2 VPBE ドライバの概要

ここでは、VPBEドライバの簡単な概要を示します。

以下に示すサブコンポーネントを含むVPBEドライバのソフトウェア構造を図2に示します:

- ドライバ用のランタイム環境を提供するリアルタイム・オペレーション・システムであるDSP/BIOS™ ソフトウェア・カーネル
- IOMによるアプリケーションに対する共通のDSP/BIOSドライバ・インターフェイスを提供
- ハードウェアのレジスタを抽象化したCSLR(Registered CSL)
- DDCはドライバコアと独立したオペレーティング・システム(OS)
- PALOSはVPBEドライバがOS処理を要求するためのDSP/BIOSの抽象化層を提供
- DM6437評価モジュール(EVM)ボード上で動作するドライバ

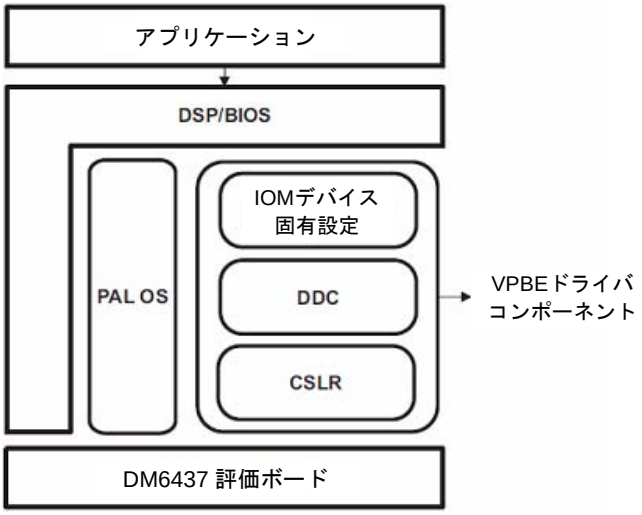

図 2. VPBEドライバ構造

VPBEドライバは以下に示す機能をサポートします:

- 各ビデオ/OSDウィンドウ(VIDWIN0/VIDWIN1及びOSDWIN0/OSDWIN1)、VENC及びCURSOR における個別のチャネル
- 割り込みモードでのみ動作する同期式ドライバ
- ビデオのシームレスなキャプチャ/ディスプレイを行なうマルチ・フレーム・バッファの切り替え
- メンテナンスや新たなプラットフォームへの移植が容易．異なるデバイスへのVPBEドライバのポーティングは、デバイス固有のコンフィグレーションレジスタを変更するのみ．異なるOSへのVPBEドライバへのポーティングにおいても、ターゲットOSに適したPAL OS層の修正で実現可能

1.3 VPFE ドライバの概要

ここでは、VPFEドライバの簡単な概要を示します。

VPFEドライバは、図2に示されるVPBEドライバと同様のソフトウェア構造になります。VPFEドライバは以下に示す特長をサポートします:

- CCDCチャネル
- 割り込みモードのみで動作する同期式ドライバ
- ビデオのシームレスなキャプチャ及びディスプレイを行なうマルチ・フレーム・バッファの切り替え
- メンテナンスや新たなプラットフォームへの移植が容易

2 VPBE ドライバの使用例

この節では、VPBEウィンドウの使用方法のデモンストレーションやVPBEがサポートする重要な機能をリストアップした例について説明します。図3に示される以下の画像は、そのディスプレイの例になります。

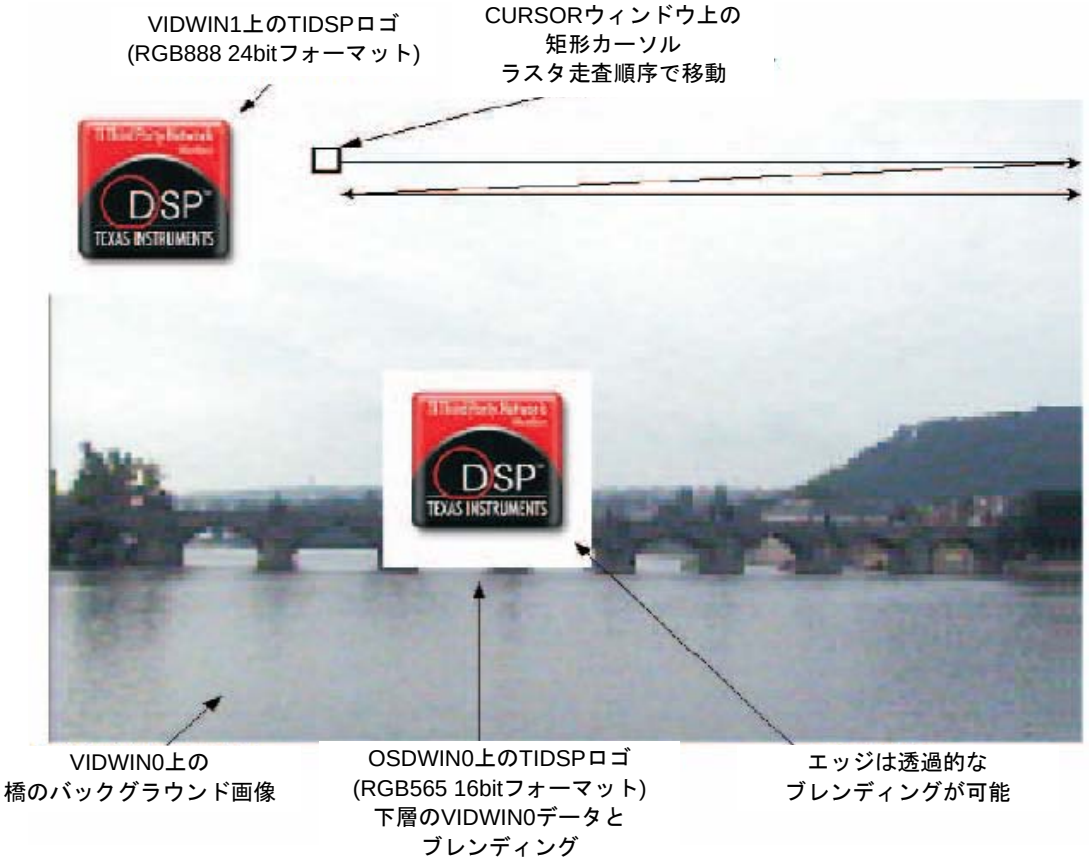

図 3. VPBEサンプルにおける表示画面

- 橋が表示されるビデオデータは、バックグラウンドとしてVIDWIN0に表示されます。ビデオデータはimages.libファイル内の *BRIDGE30f* という配列に格納され、180x120の解像度でYUV422データの30フレームを構成しています。水平/垂直方向の両方向に対して4xにアップスケーリングされ、NTSC D1解像度(720x480)となります。バックグラウンド画像は常に高品質を必要としないので少ないデータ量でメモリに格納されますが、アップスケーリング機能によりNTSC D1解像度を維持します。各列における有効データサイズは360バイトです。VPBEハードウェアはピッチが32バイト・アラインされていることを要求するので、列サイズが384バイトになるように各列の後ろに24バイトが付加されています。つまり、BRIDGE30fの配列のサイズは、384x120x30 = 1,382,400バイトになります。

- 画像左上部のTI DSPロゴは、フォーマットがRGB888(24bitmap)であり、VIDWIN1上に表示されます。この画像はimages.libファイル内の *TIDSP146x146RGB888* という配列に格納され、146x146のサイズで、各列が146x3=438バイトとなっています。同様に、列サイズが448バイトになるように各列の後ろに10バイトが付加されています。つまり、TIDSP146x146RGB888の配列のサイズは、448x146 = 65,408バイトになります。
- 画面中央のもう一つのTI DSPロゴは、フォーマットがRGB565(16bitmap)であり、OSDWIN0上に表示されます。この画像はimages.libファイル内の *TIDSP146x146RGB565* という配列に格納され、146x146のサイズで、各列が146x2=292バイトとなっています。同様に、列サイズが320バイトになるように各列の後ろに28バイトが付加されています。つまり、TIDSP146x146RGB565の配列のサイズは、320x146 = 46,720バイトになります。ロゴ画像全体は、OSDモジュールのブレンディング機能を用いることで下層のVIDWIN0データと部分的にブレンドすることが可能です。ロゴに対してある特定の値でピクセルを下層のVIDWIN0データとブレンディングすることも可能です。サンプルでは、0xFFFFという値で全てのピクセルをVIDWIN0とブレンドするようにプログラムされています。この場合では、ロゴのエッジは透過的にブレンドされます。
- 矩形カーソルは、ラスタ走査順序でCURSORウィンドウ上に表示かつ移動するようにプログラムされています。その大きさは20x20になります。プログラムによってCURSORのカーソルエッジや幅、カラーを変更できる機能をサポートしていますが、ドライバでは、ROM/RAM CLUTテーブルでエントリーゼロとなる1画素幅のエッジ幅と色に固定しています。

上記に記載されたデータを表示するVPBEドライバの設定手順を以下に示します:

1. 橋のビデオデータを表示するようにVIDWID0を設定
   a. コンフィグレーション・パラメータ *PSP_VPBEOsdConfigParams winParams* を用いてFVID_create()をコールすることでVIDWIN0 の *vid0Handle* チャネルを生成します。winParamsのbitsPerPixelとcolorFormatがそれぞれPSP_VPSS_BITS16とPSP_VPBE_YCbCr422にセットされているので、ドライバは入力データがYUV422フォーマットであることがわかります。ウィンドウサイズは720x480にセットされ、hScalingとvScalingは両方ともPSP_VPBE_ZOOM_4Xであるので、オリジナルの180x120フレームはNTSC D1解像度にアップスケールすることができます。ピッチは384にセットされるので、入力ビデオデータが列毎に32バイトにアラインされます。コールバックもwinParamsで提供されます。ドライバによって毎回フィールドがコールされるので、インターレース・フレームの半分が出力側に送られ、表示されているフレーム数をカウントするために使用されます。全30フレームが表示されると、先頭からデータ表示を繰り返します。
   b. FVID_allocBuffer()を2度コールし、VIDWIN0チャネル用のフレーム・バッファを2つ確保します。
   c. フレーム・バッファに最初のフレームを2つコピーします。
   d. FVID_queue()をコールして、VIDWIN0に表示される最初の橋画像フレームを含んだフレーム・バッファを転送します。
2. RGB888フォーマットのTIDSPロゴ画像を表示するようにVIDWIN1を設定
   a. コンフィグレーション・パラメータ *PSP_VPBEOsdConfigParams winParams* を用いてFVID_create()をコールすることでVIDWIN1 の *vid1Handle* チャネルを生成します。winParamsのbitsPerPixelとcolorFormatがそれぞれPSP_VPSS_BITS24とPSP_VPBE_RGB_888にセットされているので、ドライバは入力データが24-bitmapフォーマットであることがわかります。ウィンドウサイズは146x146にセットされ、hScalingとvScalingは両方ともPSP_VPBE_ZOOM_IDENTITYのズームなしとなります。ピッチは448にセットされるので、入力ビデオデータが列毎に32バイトにアラインされます。画像は更新する必要がないので、この場合コールバックの指定は必要ありません。
   b. FVID_allocBuffer()をコールし、VIDWIN1チャネル用のフレーム・バッファを2つ確保します。
   c. フレーム・バッファに画像データをコピーします。
   d. FVID_queue()をコールして、VIDWIN1にフレーム・バッファを転送します。VENCを設定すると、即座にロゴが表示されます。
3. RGB565フォーマットのTIDSPロゴ画像を表示するようにOSDWIN0を設定
   a. コンフィグレーション・パラメータ *PSP_VPBEOsdConfigParams winParams* を用いてFVID_create()をコールすることでOSDWIN0の *osd0Handle* チャネルを生成します。winParamsのbitsPerPixelとcolorFormatがそれぞれPSP_VPSS_BITS16とPSP_VPBE_RGB_565にセットされているので、ドライバは入力データが16-bitmapフォーマットであることがわかります。ウィンドウサイズは146x146にセットされ、hScalingとvScalingは両方

ともPSP_VPBE_ZOOM_IDENTITYのズームなしとなります。ピッチは320にセットされるので、入力ビデオデータが列毎に32バイトにアラインされます。画像は更新する必要がないので、この場合コールバックの指定は必要ありません。

b. 例では3つのブレンディング・オプションがあります。入力に応じて以下の設定から1つを選択してください。
   - オプション0 (blending = PSP_VPBE_BLEND0, transparency = TRUE, transparencyColor = 0xFFFF): 0xFFFFとしたピクセルは完全にマスクされます。ロゴのエッジカラーが0xFFFF値(白)なので、このオプションにおけるエッジは表示されません。
   - オプション1 (blending = PSP_VPBE_BLEND2, transparency = TRUE, transparencyColor = 0xFFFF): 0xFFFFとしたピクセルは下層のVIDWIN0データと部分的にブレンディングされます。そのブレンディング率は2/8(25%)にセットされます。この場合、ロゴのエッジはVIDWIN0と部分的にブレンディングされます。
   - オプション2 (blending = PSP_VPBE_BLEND4, transparency = FALSE): ロゴ画像全体が下層のVIDWIN0 データと部分的にブレンディングされます。そのブレンディング率は4/8(50%)にセットされます。

c. FVID_allocBuffer()をコールし、OSDWIN0チャネル用のフレーム・バッファを2つ確保します。

d. フレーム・バッファに画像データをコピーします。

e. FVID_queue()をコールして、OSDWIN0にフレーム・バッファを転送します。VENCを設定すると、即座にロゴが表示されます。

4. 矩形カーソルを表示するようにCURSORを設定
   a. コンフィグレーション・パラメータ *PSP_VPBECursorConfigParams cursorParams* を用いてFVID_create()をコールすることでCURSORの *cursorHandle* チャネルを生成します。cursorParamsのカーソルサイズが20x20にセットされています。ROM CLUTテーブルは、CursorCLUTSをPSP_VPBE_CLUTSOURCE_ROMにセットすることで選択されます。
5. VENCモジュールを設定
   a. コンフィグレーション・パラメータ *PSP_VPBEVencConfigParams vencParams* を用いてFVID_create()をコールすることでVENCの *vencHandle* チャネルを生成します。出力テストにコンポジット入力の外部モニタを使用します。vencParamsのdisplayStandardは、PSP_VPBE_DISPLAY_NTSC_INTERLACED_COMPOSITEにセットされています。
6. 全ての与えられたビデオや画像データを表示するために指定された回数のループを開始します。そのループ内では次のことを実行します。
   a. FVID_queue()をコールして、表示するための次のブリッジフレームを含む新たなフレームバッファをVIDWIN0に渡します。
   b. FVID_dequeue()をコールして、VIDWIN0からフレームバッファを受け取ります。フレームのデータが表示されると、VPBEドライバはフレームバッファを返します。そのフレームバッファは表示するために次のブリッジフレームに使用されます。
   c. ラスタースキャンオーダーにおいて、次の所定の位置にカーソルを移動させるために、CURXPとCURYPレジスタを直接プログラムします。
7. プログラムがループを抜けたら、全てのチャネル及びフレームバッファをフリー(解放)します。
   a. VIDWIN0における2つのフレームバッファをフリーするために、FVID_free()を2度コールします。VIDWIN0チャネルをフリーするために、FVID_delete()をコールします。
   b. 同様に、VIDWIN1におけるフレームバッファ、VIDWIN1チャネルをフリーするために、FVID_free()及びFVID_delete()をコールします。
   c. 同様に、OSDWIN0におけるフレームバッファ、OSDWIN0チャネルをフリーするために、FVID_free()及びFVID_delete()をコールします。
   d. CURSORチャネルをフリーするために、FVID_delete()をコールします。

3 VPFE ドライバ使用例: ビデオ･デコーダから YUV422 データをキャプチャ

この節では、図4に示される設定のビデオ･デコーダからYUV422データをキャプチャするためのVPFEドライバの使い方について説明します。CCDカメラは、アナログコンポジット信号をデジタルYVU422データにデコードするビデオデコーダ tvp5146に接続されます。YUV422データは8-bitデータバスを介してDM6437プロセッサに送られます。tvp5416とDM6437の間では、I2Cバスがあり、DM6437はtvp5146を設定するためのマスターとしても動作します。

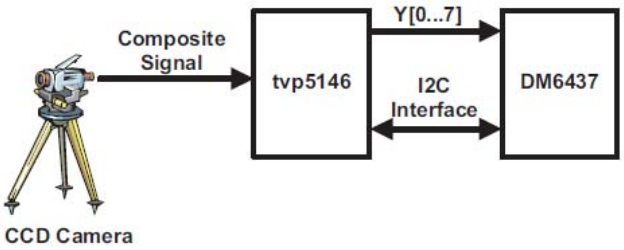

図 4. YUV422データをキャプチャするためのセットアップ

この例は、tvp5416デコーダの設定とビデオ入力データをキャプチャするためのVPFEドライバの使い方に関するデモンストレーションです。キャプチャされたデータもVPBEドライバを使用することでモニタに出力できます。CCDカメラはNTSC D1解像度でデータを出力します。しかし、VPFEのCCDCモジュールは、NTSC D1に限らずどんなフレーム･サイズをもキャプチャできるだけの十分な柔軟性を持っています。

データの入出力するためのVPFEドライバを設定する手順を以下に示します。

1. パラメータccdcParamsを引数として、FVID_create()をコールし、CCDCチャネルの *ccdcHandle* を生成します。ccdcParamsには、dataFlowやffMode, 高さ, 幅がそれぞれPSP_VPFE_CCDC_YCBCR_8, PSP_VPSS_FRAME_MODE, 480, 720にセットされているので、CCDCモジュールはインターレースのNTSC D1フレームをキャプチャする設定になります。次の3つの関数はtvp5146デコーダの設定に関連し、ccdcParamsのVPFEドライバに渡されます。
   a. PSP_VPFE_TVP5146_Open()は、CCDCチャネルが生成されるとき、DM6437とtvp5146デコーダ間のI2Cインターフェイスを初期化するためにコールされます。
   b. PSP_VPFE_TVP5146_Close()は、CCDCチャネルが削除されるとき、I2Cインターフェイスを削除するためにコールされます。
   c. PSP_VPFE_TVP5146_Control()は、tvp5146の設定するために特定の制御コマンドを発行するときにコールされます。
2. パラメータtvp5146Paramsを引数として、FVID_control()をコールし、tvp5146を設定します。tvp5146デコーダはオートでコンポジット信号をキャプチャするように設定されます。tvp5146のほとんどのコンフィグレーションパラメータは、デフォルトで典型的な使用法に最適な設定になるので、ここで設定するだけとなります。
3. FVID_alloc()をコールしてCCDCモジュールに必要な2つのフレーム･バッファを確保します。FVID_queue()をコールし、ドライバへバッファをエンキューします。
4. NTSC D1解像度かつYUV422フォーマットでキャプチャされたデータを表示するために、VIDWIN0を設定するパラメータvid0Paramsを引数としたFVID_create()をコールし、VPBEのVIDWIN0チャネルの *vid0Handle* を生成します。
5. FVID_alloc()をコールしてVIDWIN0に必要な2つのフレーム･バッファを確保します。FVID_queue()をコールしドライバへバッファをエンキューします。
6. コンポジット信号を表示するようにVENCモジュールを設定するパラメータvencParamsを引数としてFVID_create()をコールし、VPBEのVENCチャネルの *vencHandle* を生成します。
7. 入力ビデオ信号をキャプチャおよびディスプレイするために指定された回数のループを開始します。そのループ内では次のことを実行します。
   a. FVID_exchange()をコールし、キャプチャされたビデオフレームと取得し、CCDCチャネルへ空のフレーム･バッファを渡します。プログラムは、CCDCモジュールが新たなフレームをキャプチャした後に、次の処理に移ります。

b. FVID_exchange()をコールし、キャプチャされたデータを含むフレームバッファをVIDWIN0に渡し、フレームバッファを受け取ります。プログラムは、フレームのデータが表示された後に、次の処理に移ります。

8. プログラムがループを抜けたら、全てのチャネルとフレーム･バッファをフリー(解放)します。
   a. VIDWIN0における2つのフレームバッファをフリーするために、FVID_free()を2度コールします。VIDWIN0チャネルをフリーするために、FVID_delete()をコールします。
   b. 同様に、CCDCにおけるフレームバッファ、CCDCチャネルをフリーするために、FVID_free()及びFVID_delete()をコールします。
   c. VENCチャネルをフリーするために、FVID_delete()をコールします。

4 パッケージの使用ガイド

この節では、サンプルの実行およびリコンパイルの方法について説明します。

4.1 ハードウェア及びソフトウェア要求事項

サンプルコードをコンパイル及び実行するために必要な環境とソフトウェアを以下に示します。

- Microsoft Windows XPがインストールされているPC
- Code Composer Studio 3.3ソフトウェアがインストールされていること
- TMS320DM6437 評価用モジュール(EVM) ボード
- コンポジット端子を有したNTSC D1解像度でビデオデータをキャプチャするCCDカメラ
- コンポジット端子のモニタ

4.2 パッケージ内容

このサンプル･パッケージは、*spraap3.zip* というzipファイルに圧縮されています。解凍するとPC上に *VPBE_VPFE_Examples* として展開されます。*VPBE_VPFE_Examples* は、*CSL_inc, drivers, Pal_os, VPBE_example* および *VPFE_tvp5146_example* の5つのサブフォルダから構成されています。

CSL_inc には、TMS320DM643xの全てのペリフェラルに対するchip support library (CSL)のヘッダが含まれています。これは全てのデバイスレジスタにアクセスするための抽象化層を提供します。VPBEを例にとると、CURSOR位置をセットするためにCURXPおよびCURYPを直接プログラムでするために使用されます。

drivers には、サンプルで使用する様々なペリフェラルのライブラリが含まれています。*drivers* サブフォルダの内容を表1 に示します。

表1. VPBE_VPFE_Examples/drivers のコンテンツ

コンテンツ	説明
i2c\lib\	• i2c_drv_bios_dbg.lib: デバッグ用のI2Cドライバ • i2c_drv_bios_rel.lib: リリース用のI2Cドライバ
i2c\inc\psp_i2cApi.h	I2CドライバのDDCレイヤーAPIを定義するヘッダファイル VPFEの例では、tvp5146デコーダと通信するためのAPIを使用しています。
previewer\lib\	• prev_drv_bios_dbg.lib: デバッグ用のpreviewerドライバ • prev_drv_bios_rel.lib: リリース用のpreviewerドライバ CCDCが初期化される際に、VPFEドライバはpreviewerをディスエーブルするためにpreviewerドライバをコールします。
VPBE\lib\	• vpbe_drv_bios_dbg.lib: デバッグ用のVPBEドライバ • vpbe_drv_bios_rel.lib: リリース用のVPBEドライバ
VPFE\lib\	• vpfe_drv_bios_dbg.lib: デバッグ用のVPFEドライバ • vpfe_drv_bios_rel.lib: リリース用のVPFEドライバ
inc\	PSPドライバに共通な全てのヘッダファイルが含まれています。

Pal_os には、ドライバでアクセスされるOS抽象化層の実装が行なわれています。*Pal_os* サブフォルダの内容を表2に示します。

表 2. VPBE_VPFE_Examples/Pal_os のコンテンツ

コンテンツ	説明
lib\	• palos_bios_dbg.lib: デバッグ用のpalosドライバ • palos_bios_rel.lib: リリース用のpalosドライバ
inc\	palosの全てのヘッダが含まれています。

VPBE_example フォルダは、第 2 節に記載されているサンプルが含まれています。表 3 に内容を示します。

表 3. VPBE_VPFE_Examples/VPBE_example のコンテンツ

コンテンツ	説明
Debug\	デバッグオプションでビルドする際に生成されるコードが含まれます。 • VPBEexample.out: デバッグ時の実行ファイル • VPBEexample.map: デバッグ時のメモリマップ・ファイル
Release\	リリースオプションでビルドする際に生成されるコードが含まれます。 • VPBEexample.out: リリース時の実行ファイル • VPBEexample.map: リリース時のメモリマップ・ファイル
images\	images.lib: 表示される全ての画像データのライブラリ
src\	サンプルの全てのソース・コードが含まれています。 • main.c: PINMUXの初期化を行なうメイン関数 • test.c: 2章で記載されている手順にてタスクを実装 • configureCursorWin.c: CURSORウィンドウを設定するコード • configureOsdWin0_TIDSP_RGB565.c: RGB565(16ビット)フォーマットのTI DSPロゴを表示するOSDWIN0を設定するコード • configureVidWin0_BRIDGE_YUV422.c: バックグランドとしてYUV422フォーマットの橋の画像を表示するVIDWIN0を設定するコード • configureVidWin1_TIDSP_RGB888.c: RGB888(24ビット)フォーマットのTI DSPロゴを表示するVIDWIN1を設定するコード
VPBEexample.pjt	このサンプルのCCS用プロジェクトファイル
VPBEexample.tcf	DM643x上でDSP/BIOSを実行するための設定ファイルです。VPBEドライバを設定する *dm6437_vpbe0.tci* が含まれます。

VPFE_tvp5146_example フォルダは、第 3 章に記載されているサンプルが含まれています。内容を表 4 に示します。

表 4. VPBE_VPFE_Examples/VPFE_tvp5146_example のコンテンツ

コンテンツ	説明
Debug\	デバッグオプションでビルドする際に生成されるコードが含まれます。 • VPFEtvp5146Example.out: デバッグ時の実行ファイル • VPFEtvp5146Example.map: デバッグ時のメモリマップ・ファイル
Release\	リリースオプションでビルドする際に生成されるコードが含まれます。 • VPFEtvp5146Example.out: リリース時の実行ファイル • VPFEtvp5146Example.map: リリース時のメモリマップ・ファイル
src\	サンプルの全てのソース・コードが含まれています。 • main.c: PINMUXの初期化を行なうメイン関数 • test.c: 3章で記載されている手順にてタスクを実装

	• psp_i2c_interface.c & psp_i2c_interface.h: I2Cドライバを用いたtvp5146と通信するためのコード • i2cParams_evmdm6437.c: I2Cドライバを初期化するコード • tvp5146_extDecoder.c: tvp5146デコーダに対して、オープン、クローズ、動的な制御を行なうためのコード
VPFEtvp5146Example.pjt	このサンプルのCCS用プロジェクトファイル
VPFEtvp5146Example.tcf	DM643x上でDSP/BIOSを実行するための設定ファイルです。 • VPBEドライバを設定する *dm6437_vpbe0_tci* が含まれます。 • I2Cドライバを設定する *dm6437_i2c0.tci* が含まれます。 • VPFEドライバを設定する *dm6437_vpfe0_tci* が含まれます。

4.3 VPBE サンプルの実行

この節では、VPBEサンプルを実行するための手順を説明します。

4.3.1 VPBE サンプルの実行方法

第2節に記載されているVPBEサンプルの実行方法について、以下に手順を示します。

1. コンポジットケーブルを用いて、DM647EVMボードとモニタを接続します。ボード上では *DAC D* に接続し、ボードに電源を入れてください。
2. Code Composer Studioを起動し、VPBEexample.pjtというプロジェクトを開きます。
3. サンプルのdebug(release)バージョンを実行するために、*Debug/VPBEexample.out (Release/VPBEexample.out)* をロードします。CCSのメニューから *File → Load program* を選択するとファイルのロードができます。
4. メニューの *Debug → Reset CPU* を選択し、CPUをリセットしてください。
5. メニューの *Debug → Restart* を選択し、プログラムをリスタートしてください。
6. *F5* ボタンを押し、プログラムを実行してください。図3のようにモニタに画像が表示されるはずです。

4.3.2 TVP5146 を用いた YUV422 データをキャプチャする VPFE サンプルの実行方法

第3節に記載されているVPFEサンプルの実行方法について、以下に手順を示します。

1. コンポジットケーブルを用いて、DM6437EVMボードとモニタを接続します。ボード上では *DAC D* に接続します。
2. CCDカメラをDM6437EVMボードの *VIDEO IN* ポートに接続し、ボードに電源を入れます。
3. Code Composer Studioを起動し、VPFEtvp5146Example.pjtというプロジェクトを開きます。
4. サンプルのdebug（release)バージョンを実行するために、*Debug/VPFEtvp5146Example.out (Release/VPFEtvp5146Example.out)* をロードします。CCSのメニューから *File → Load program* を選択するとファイルのロードができます。
5. メニューの *Debug → Reset CPU* を選択し、CPUをリセットしてください。
6. メニューの *Debug → Restart* を選択し、プログラムをリスタートしてください。
7. *F5* ボタンを押し、プログラムを実行してください。モニタにキャプチャしたカメラの画像が表示されるはずです。

4.4 VPBE/VPFE サンプルのコンパイル

この節では、VPBE/VPFEサンプルをコンパイルするための手順を説明します。

4.4.1 VPBE サンプルをコンパイルする方法

第2章で記載されているVPBEサンプルのコンパイル方法について、手順を以下に示します。

1. Code Composer Studioを起動し、VPBEexample.pjtを開きます。
2. サンプルを *Debug (Release)* バージョンでコンパイルするために、CCSのコンフィグレーションウィンドウを *Debug (Release)* に選択してください。それから、メニュー: *Project → Build* を選択して、コードをビルドします。

4.4.2 TVP5146 を用いた YUV422 でキャプチャする VPFE サンプルをコンパイルする方法

第3章で記載されているVPBEサンプルのコンパイル方法について、手順を以下に示します。

1. Code Composer Studioを起動し、VPFEtvp5146Example.pjtを開きます。
2. サンプルを *Debug (Release)* バージョンでコンパイルするために、CCSのコンフィグレーションウィンドウを *Debug (Release)* に選択してください。それから、メニュー: *Project* → *Build* を選択して、コードをビルドします。

5 参考資料

英語資料:

1. *TMS320DM643x DMP Video Processing Back End (VPBE) User's Guide (SPRU952)*
2. *TMS320DM643x DMP Video Processing Front End (VPFE) User's Guide (SPRU977)*

ご注意

日本テキサス・インスツルメンツ株式会社（以下TIJといいます）及びTexas Instruments Incorporated（TIJの親会社、以下TIJないしTexas Instruments Incorporatedを総称してTIといいます）は、その製品及びサービスを任意に修正し、改善、改良、その他の変更をし、もしくは製品の製造中止またはサービスの提供を中止する権利を留保します。従いまして、お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての製品は、お客様とTIJとの間に取引契約が締結されている場合は、当該契約条件に基づき、また当該取引契約が締結されていない場合は、ご注文の受諾の際に提示されるTIJの標準販売契約約款に従って販売されます。

TIは、そのハードウェア製品が、TIの標準保証条件に従い販売時の仕様に対応した性能を有していること、またはお客様とTIJとの間で合意された保証条件に従い合意された仕様に対応した性能を有していることを保証します。検査およびその他の品質管理技法は、TIが当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、政府がそれ等の実行を義務づけている場合を除き、必ずしも行なわれておりません。

TIは、製品のアプリケーションに関する支援もしくはお客様の製品の設計について責任を負うことはありません。TI製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI製部品を使用したお客様の製品及びアプリケーションについて想定されうる危険を最小のものとするため、適切な設計上および操作上の安全対策は、必ずお客様にてお取り下さい。

TIは、TIの製品もしくはサービスが使用されている組み合せ、機械装置、もしくは方法に関連しているTIの特許権、著作権、回路配置利用権、その他のTIの知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TIが第三者の製品もしくはサービスについて情報を提供することは、TIが当該製品もしくはサービスを使用することについてライセンスを与えるとか、保証もしくは是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない場合もあり、またTIの特許その他の知的財産権に基づきTI からライセンスを得て頂かなければならない場合もあります。

TIのデータ・ブックもしくはデータ・シートの中にある情報を複製することは、その情報に一切の変更を加えること無く、かつその情報と結び付られた全ての保証、条件、制限及び通知と共に複製がなされる限りにおいて許されるものとします。当該情報に変更を加えて複製することは不公正で誤認を生じさせる行為です。TIは、そのような変更された情報や複製については何の義務も責任も負いません。

TIの製品もしくはサービスについてTIにより示された数値、特性、条件その他のパラメーターと異なる、あるいは、それを超えてなされた説明で当該TI製品もしくはサービスを再販売することは、当該TI製品もしくはサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、かつ不公正で誤認を生じさせる行為です。TIは、そのような説明については何の義務も責任もありません。

TIは、TIの製品が、安全でないことが致命的となる用途ないしアプリケーション（例えば、生命維持装置のように、TI製品に不良があった場合に、その不良により相当な確率で死傷等の重篤な事故が発生するようなもの）に使用されることを認めておりません。但し、お客様とTIの双方の権限有る役員が書面でそのような使用について明確に合意した場合は除きます。たとえTIがアプリケーションに関連した情報やサポートを提供したとしても、お客様は、そのようなアプリケーションの安全面及び規制面から見た諸問題を解決するために必要とされる専門的知識及び技術を持ち、かつ、お客様の製品について、またTI製品をそのような安全でないことが致命的となる用途に使用することについて、お客様が全ての法的責任、規制を遵守する責任、及び安全に関する要求事項を満足させる責任を負っていることを認め、かつそのことに同意します。さらに、もし万一、TIの製品がそのような安全でないことが致命的となる用途に使用されたことによって損害が発生し、TIないしその代表者がその損害を賠償した場合は、お客様がTIないしその代表者にその全額の補償をするものとします。

TI製品は、軍事的用途もしくは宇宙航空アプリケーションないし軍事的環境、航空宇宙環境にて使用されるようには設計もされていませんし、使用されることを意図されておりません。但し、当該TI製品が、軍需対応グレード品、若しくは「強化プラスティック」製品としてTIが特別に指定した製品である場合は除きます。TIが軍需対応グレード品として指定した製品のみが軍需品の仕様書に合致いたします。お客様は、TIが軍需対応グレード品として指定していない製品を、軍事的用途もしくは軍事的環境下で使用することは、もっぱらお客様の危険負担においてなされるということ、及び、お客様がもっぱら責任をもって、そのような使用に関して必要とされる全ての法的要求事項及び規制上の要求事項を満足させなければならないことを認め、かつ同意します。

TI製品は、自動車用アプリケーションないし自動車の環境において使用されるようには設計されていませんし、また使用されることを意図されておりません。但し、TIがISO/TS 16949の要求事項を満たしていると特別に指定したTI製品は除きます。お客様は、お客様が当該TI指定品以外のTI製品を自動車用アプリケーションに使用しても、TIは当該要求事項を満たしていなかったことについて、いかなる責任も負わないことを認め、かつ同意します。

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊/劣化、または故障を起こすことがあります。

弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. 静電気
   - 素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。
   - 弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。
   - マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。
   - 前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。
2. 温・湿度環境
   - 温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）
   - 直射日光があたる状態で保管・輸送しないこと。
3. 防湿梱包
   - 防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。
4. 機械的衝撃
   - 梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。
5. 熱衝撃
   - はんだ付け時は、最低限260℃以上の高温状態に、10秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）
6. 汚染
   - はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。
   - はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）

以上