AN84868

サイプレスの EZ-USB FX3 を使用した USB 経由での Xilinx FPGA コンフィギュレーション

著者: Rama Sai Krishna Vakkantula

関連プロジェクト：あり

関連製品ファミリ：CYUSB3014

最新の FX3 SDK：こちらをクリック

更にサンプルコードをお求めでしょうか？以下をご参照ください。

豊富な FX3 サンプルコードにアクセスするには、USB SuperSpeed サンプルコードウェブページをご覧ください。

AN84868 は、次世代 USB 3.0 ペリフェラル コントローラーである EZ-USB® FX3™を使用してスレーブ シリアル インターフェース経由で Xilinx 社の FPGA をコンフィギュレーションする方法について説明します。このインターフェースにより、USB 2.0 または 3.0 経由で Xilinx 社の FPGA にコンフィギュレーション ファイルをダウンロードすることができます。本アプリケーション ノートと共に提供されたファームウェア ファイルは Xilinx 社の FPGA に対応して設計およびテストされていますが、同様のインターフェースを持つ他の FPGA に対応してそれらをカスタマイズすることができます。

Contents

1 はじめに

FX3 にはイメージセンサー, 外部プロセッサ, ASIC または FPGA のような外部デバイスに繋がれるコンフィギュレーション可能なパラレル汎用プログラマブルインターフェース (GPIF II) があります。その結果ユーザーは、ほぼどんなシステムにも USB 3.0 機能を統合することができます。

また FX3 は、UART, SPI, I^2C, および I^2S などのシリアル ペリフェラルに接続するインターフェースも提供しています。

FX3 により、スーパースピード機能をいかなる FPGA ベースの設計にも追加することができます。殆どのアプリケーションで、FPGA はマスターとして機能し、GPIF II は同期スレーブ FIFO インターフェースで動作します。スレーブ FIFO インターフェースの詳細については、AN65974 – EZ-USB® FX3™スレーブ FIFO インターフェースによる設計を参照してください。

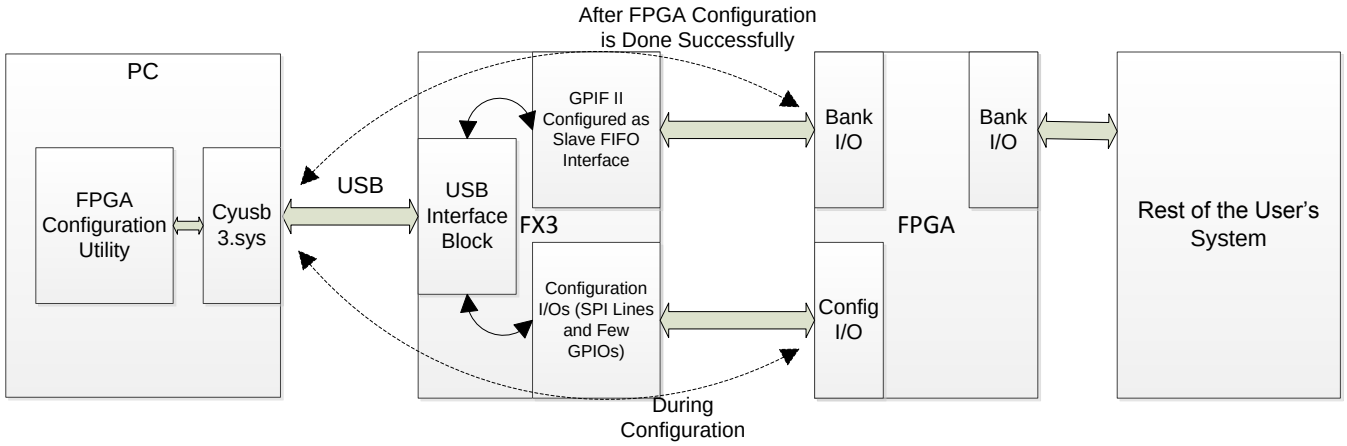
FPGA に接続されているコントローラー(この場合は FX3)を使用してその FPGA をコンフィギュレーションすることができます。FX3 を使用すると、FPGA 用に専用のコンフィギュレーション チップ (例えば PROM またはプロセッサ) が不要になります。さらに、この方法は基板上の JTAG コネクタを必要とする広く使用されている JTAG コンフィギュレーション インターフェースの代替として作動できます。この方法により、コストと基板スペースが低減されます。また、FPGA コンフィギュレーションファイルをプリロードされた外部 SPI フラッシュまたは EEPROM から FX3 を使用して FPGA コンフィギュレーションをロードすることもできます。FX3 + FPGA + HelionVision ISP ベース インダストリアルカメラ リファレンスデザイン – KBA222700 を参照して下さい。ここでは、FX3 はイメージング アプリケーション用の FPGA とインターフェースされています。

マスターとして機能する FX3 は、スレーブ パラレル (SelectMAP) とスレーブ シリアルの 2 モードで Xilinx 社の FPGA を設定することができます。FPGA をコンフィギュレーションする様々なオプションの情報については、Xilinx Spartan-6 FPGA コンフィギュレーションユーザーガイドを参照してください。このアプリケーション ノートでは、スレーブ シリアル モードのみについて説明します。FPGA のコンフィギュレーションが完了した後、FX3 ファームウェアがスレーブ FIFO インターフェースに切り替わる方法も説明します。図 1 には、FX3 がまず FPGA をコンフィギュレーションして、コンフィギュレーションが正常に完了した後にスレーブ FIFO インターフェースに切り替わるブロック図を示します。

このアプリケーションノートでは、ホストアプリケーションを使用してベンダーコマンドを使用して FPGA コンフィギュレーションファイルをロードします。 ベンダーコマンドは、SPI インターフェースを介してホストから受信した FPGA コンフィギュレーションデータにより FPGA コンフィギュレーションプロセスを開始します。

次のセクションでは、Xilinx 社のスレーブ シリアル コンフィギュレーション インターフェースの詳細と FX3 を使用した設計実装を説明します。

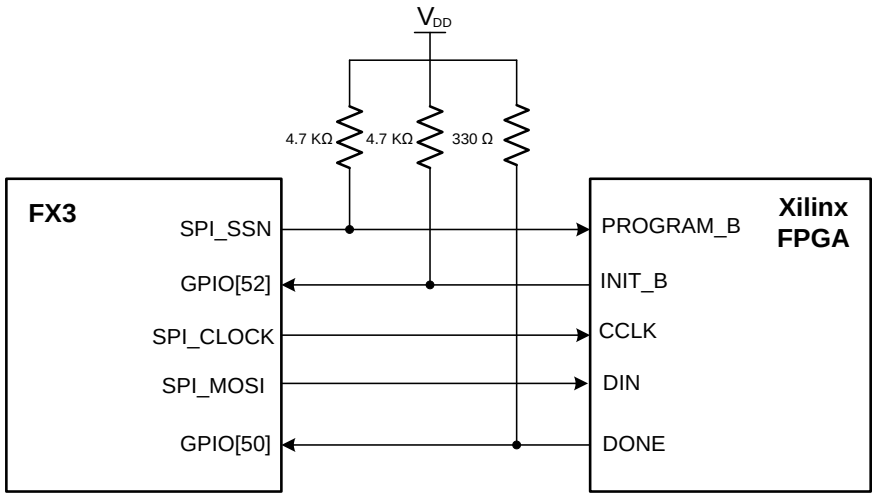
図 1. システム レベルのアプリケーションのブロック図



2 Xilinx 社のスレーブ シリアル コンフィギュレーション インタフェース

このセクションは Xilinx 社のスレーブ シリアル コンフィギュレーション インターフェースの詳細を説明します。図 2 は Xilinx 社のスレーブ シリアル インターフェースに関連したインターフェースのピンを示します。表 2 ではスレーブ シリアル インターフェース ピンを説明します。

図 2. FX3 と Xilinx Spartan-6 FPGA 間のハードウェア接続



PROGRAM_B、INIT_B および DONE はオープンドレイン信号です。これらのラインに適切な値のプルアップ抵抗を接続します。図 2 に記載されている抵抗値は Xilinx Spartan-6 FPGA コンフィギュレーション ユーザーガイドを参照しています。FX3 CYUSB3KIT-001 ボードを使用する場合は、これらのプルアップ抵抗を接続する必要がありませんが、最終のデザインではプルアップ抵抗を配置する必要があることに注意してください。

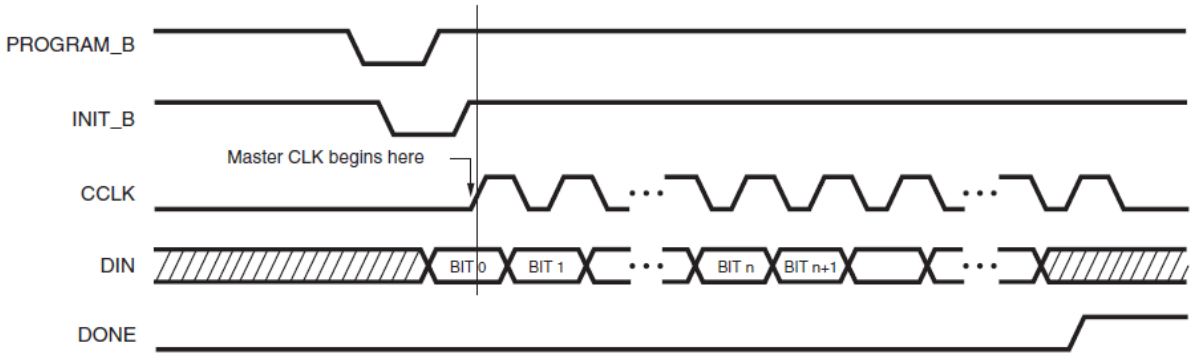
表 1 にはスレーブ シリアル インターフェースでのインターフェースの信号を示します。

表 1. Xilinx スレーブ シリアル コンフィギュレーション ピンの説明

ピン名	ピン方向 (FPGA へ)	ピンの説明
PROGRAM_B	入力	**FPGA をプログラム。**アクティブ LOW。 500ns 以上 (Spartan-3 FPGA の場合は 300ns) LOW にアサートされた場合、コンフィギュレーション メモリをクリアすることによって FPGA にそのコンフィギュレーション プロセスを再起動させる
INIT_B	オープン ドレインの双方向 I/O	**FPGA の初期化インジケータ。**パワーオン リセット (POR) の後、または PROGRAM_B が LOW にパルスされ FPGA がそのコンフィギュレーション メモリをクリアしている間、LOW に駆動。コンフィギュレーション中に CRC エラーが検出されると、FPGA は再び INIT_B を LOW に駆動
CCLK	入力	**コンフィギュレーション クロック**
DIN	入力	**データ入力。**シリアル データ。FPGA は立ち上がり CCLK エッジでデータを取り込む
DONE	オープンドレインの双方向 I/O	**FPGA コンフィギュレーション完了。**コンフィギュレーション中に LOW。FPGA がコンフィギュレーションを正常に完了させた場合は HIGH になる

図 3 に、Xilinx 社のスレーブ シリアル コンフィギュレーションのクロッキング シーケンス図を示します。

図 3. Xilinx 社のスレーブシリアル コンフィギュレーションのクロッキング シーケンス



3 実装

FX3 は、PROGRAM_B をパルスし、INIT_B ピンを監視することによってコンフィギュレーションを開始します。INIT_B のピンは HIGH になると、FPGA はデータを受信する準備ができます。そして FX3 は、DONE ピンが正常なコンフィギュレーションを示す HIGH になるか、または INIT_B のピンがコンフィギュレーションエラーを示す LOW になるまで、データ信号とクロック信号の供給を開始します。コンフィギュレーション プロセスはコンフィギュレーション ファイル サイズから指定されたより多くのクロック サイクルを必要とします。これらの追加クロックは FPGA の起動中に必要です(図 3 を参照)。

3.1 ハードウェアの詳細

3.1.1 ハードウェア基盤

- Xilinx SP601 評価キット
- スーパー スピード エクスプローラー キット(CYUSB3KIT-003)または EZ-USB FX3 DVK(CYUSB3KIT-001)
- Samtec と FMC の相互接続基板(CYUSB3KIT-001 用)または CYUSB3ACC005 インターコネクション基板(CYUSB3KIT-003 用)
- コンフィギュレーション信号を相互接続するための配線

FX3 内の SPI ハードウェア ブロックは PC からのコンフィギュレーション データをシリアル化します。FX3 の SPI_SSN (スレーブ選択), SPI_CLOCK および SPI_MOSI は、それぞれ Xilinx 社 FPGA の PROGRAM_B, CCLK および DIN に接続されています。FPGA の INIT_B と DONE ピンはそれぞれ GPIO[52]と GPIO[50]に接続されています。図 2 は FX3 と Xilinx 社の FPGA との接続を示します。

3.2 FX3 ファームウェア

添付されているファームウェアには以下の部分があります。

- スレーブ シリアル インターフェースを介して FX3 に接続する Xilinx FPGA のコンフィギュレーション。
- Xilinx 社の FPGA のコンフィギュレーションが成功すると AN65974 で説明したのと同じ動作をするスレーブ FIFO インターフェース コンフィギュレーション。

FX3 内の SPI ハードウェア ブロックは、図 1 に示されるように、FX3 が PC アプリケーション“FPGA コンフィギュレーション ユーティリティ”から受信するデータをシリアル化します。FPGA コンフィギュレーション ユーティリティは、サイプレスの VID – 0x04B4 と PID – 0x00F1 の USB デバイスを識別するために設計されています。

アプリケーション ファームウェアの構造については、FX3 プログラマのマニュアルの FX3 アプリケーション構造の章を確認してください。FX3 SDK API の詳細については、FX3 ファーム ウェア API ガイドを参考資料として使用してください。

表 2 はこのアプリケーション ノートに添付されているファームウェア ソースコードにあるファイルを説明します。

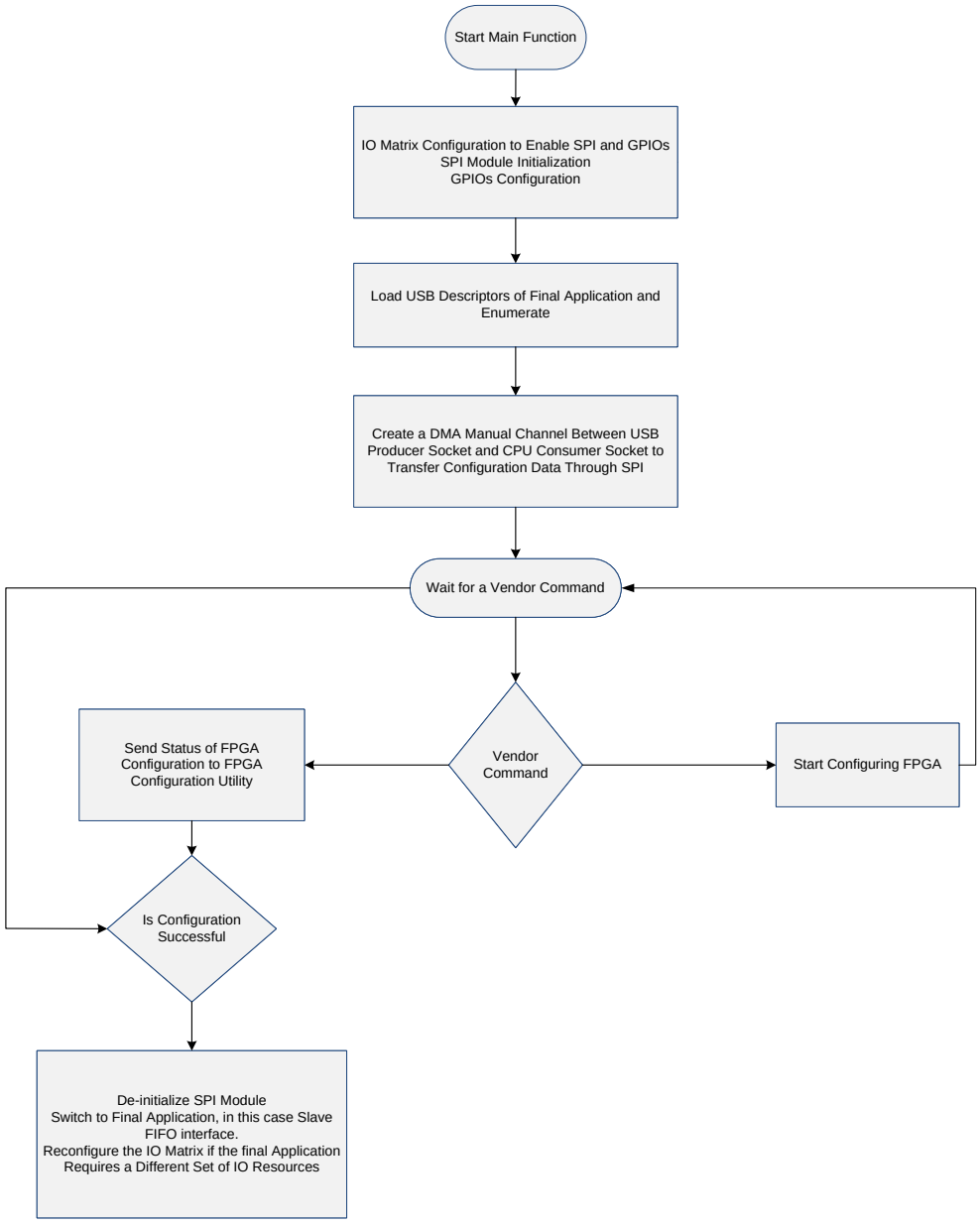
表 2. FX3 ファームウェア ソースファイルの説明

ファイル名	説明
cyfx_gcc_startup.S	サイプレスの FX3 ファームウェア起動コード
cyfxconfigfpga.c	このファイルはスレーブ シリアル モードでの FPGA のコンフィギュレーションの例を示す。以下の関数を含む • **Main**: FX3 デバイスを初期化し、キャッシュをセットアップし、FX3 I/O をコンフィギュレーションし、RTOS カーネルを起動 • **CyFxConfigFpgaApplnInit**: FX3 GPIO および SPI モジュールを初期化。GPIO[50]と GPIO[52]を入力信号として設定。エニュメレーションのために FX3 USB ブロックを初期化 • **CyFxConfigFpgaApplnStart**: USB 転送のためのエンドポイントのコンフィギュレーションと FX3 の USB ブロックから SPI ブロックへのデータ転送のための DMA チャネル コンフィギュレーション • **CyFxConfigFpgaApplnStop:** I/O マトリックスのリコンフィギュレーションを可能にするために FX3 GPIO および SPI モジュールを非初期化 • **CyFxConfigFpga**: スレーブ シリアル インターフェースを介してコンフィギュレーション データを Xilinx 社の FPGA に書き込む
cyfxconfigfpga.h	このファイルは Configure FPGA アプリケーションの例で使われる定数および定義を含む
cyfxslfifosync.c	このファイルはスレーブ FIFO 同期モードの例を示す。以下の関数を含む • **CyFxApplicationDefine**: スレーブ FIFO インターフェースを介してデータ転送を実行するためのアプリケーション スレッドを作成 • **SlFifoAppThread_Entry**: FX3 の内部ブロックのためのそれらの初期化関数を呼び出すアプリケーション スレッド関数。FPGA コンフィギュレーションの一連の動作が終わるのを待ち、FPGA のコンフィギュレーションが正常に完了したら、スレーブ FIFO インターフェースへ切り替える • **CyFxSwitchtoslFifo:** スレーブ FIFO インターフェースの要件ごとに FX3 I/O マトリックスをリコンフィギュレーション • **CyFxSlFifoApplnInit:** プロセッサ インターフェース ブロックを初期化し、スレーブ FIFO インターフェース用の GPIF コンフィギュレーションをロードし、GPIF のステートマシンを起動 • **CyFxConfigFpgaApplnStart:** USB 転送のためのエンドポイント コンフィギュレーションと FX3 の GPIF II ブロックから USB ブロックへのデータ転送のための DMA チャネル コンフィギュレーション • **CyFxSlFifoApplnStop**: この関数はスレーブ FIFO アプリケーションを中止。RESET または DISCONNECT イベントが USB ホストから受け取られる度に、この関数が呼ばれる。この関数によってエンドポイントは無効になり、DMA チャネルは破棄される • **CyFxUVCApplnUSBEventCB**: 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 • **CyFxSlFifoApplnUSBSetupCB**: USB セットアップの要求を処理するためのコールバック • **CyFxSlFifoApplnDebugInit:** デバッグ メッセージのプリント用の FX3 の UART ブロックを初期化。デバッグ プリントは UART にルートされ、115200 ボーで動作する UART コンソールを使用して見ることが可能
cyfxslfifosync.h	このファイルはスレーブ FIFO アプリケーションで使われる定数および定義を含む。
cyfxslfifousbdscr.c	このファイルはスレーブ FIFO の例に必要な USB ディスクリプタを含む
cyfxtx.c	このファイルは ThreadX RTOS 用に要求される移植を定義。このファイルはソース形式で提供され、アプリケーション ソースコードと共にコンパイルする必要がある
cyfxgpif2config.h	このファイルは 16 ビットと 32 ビットのスレーブ FIFO インターフェース用の GPIF II ディスクリプタを含む

注: FX3 に特定される用語については、「Getting Started with EZ-USB FX3」アプリケーション ノートの「FX3 Terminology」の節を参照してください。

図 4 のフローチャートは FX3 ファームウェアを説明します。

図 4. FX3 ファームウェアのフローチャート



3.3 I/O マトリックスのコンフィギュレーション

main() 関数では、I/O マトリックス (以下のコードに示す) をアプリケーションの要件に応じてコンフィギュレーションします。SPI インターフェースを有効にするために、GPIF II インターフェースは 16 ビットにコンフィギュレーションされます。GPIO[50]と GPIO[52]は有効にされ、Xilinx 社の FPGA の DONE ピンと INIT_B ピンに接続します (ハードウェア インターフェース図については図 2 を参照)。cyfxconfigfpga.c ファイルに存在する main() 関数内に、この抜粋したコードがあります。

```
io_cfg.useUart   = CyTrue;
io_cfg.useI2C    = CyFalse;
io_cfg.useI2S    = CyFalse;
io_cfg.useSpi    = CyTrue;
io_cfg.isDQ32Bit = CyFalse;
io_cfg.lppMode   = CY_U3P_IO_MATRIX_LPP_DEFAULT;
/* GPIOs 50 and 52 are enabled. */
io_cfg.gpioSimpleEn[0]  = 0x00000000;
io_cfg.gpioSimpleEn[1]  = 0x00140000;
io_cfg.gpioComplexEn[0] = 0;
io_cfg.gpioComplexEn[1] = 0;
status = CyU3PDeviceConfigureIOMatrix (&io_cfg);
```

3.3.1 SPI モジュールの初期化

SPI モジュールは以下のコードによって初期化され、コンフィギュレーションされます。25MHz のクロック周波数で動作するようにコンフィギュレーションされます。FX3 SPI ハードウェア ブロックは最大 33MHz のクロック周波数をサポートできます。cyfxconfigfpga.c ファイルにある **CyFxConfigFpgaApplnStart()** 関数内に、この抜粋したコードがあります。

```
/* Start the SPI module and configure the master. */
apiRetStatus = CyU3PSpiInit();

/* Start the SPI master block. Run the SPI clock at 25MHz and configure
   the word length to 8 bits. Also configure the slave select using FW. */
        CyU3PMemSet ((uint8_t *)&spiConfig, 0, sizeof(spiConfig));
        spiConfig.isLsbFirst = CyFalse;
        spiConfig.cpol       = CyTrue;
        spiConfig.ssnPol     = CyFalse;
        spiConfig.cpha       = CyTrue;
        spiConfig.leadTime   = CY_U3P_SPI_SSN_LAG_LEAD_HALF_CLK;
        spiConfig.lagTime    = CY_U3P_SPI_SSN_LAG_LEAD_HALF_CLK;
        spiConfig.ssnCtrl    = CY_U3P_SPI_SSN_CTRL_FW;
        spiConfig.clock      = 25000000; /* Maximum value of SPI clock is 33
MHz*/
        spiConfig.wordLen    = 8;

apiRetStatus = CyU3PSpiSetConfig (&spiConfig, NULL);
```

3.3.2 GPIO コンフィギュレーション

GPIO モジュールは以下のコードによって初期化され、コンフィギュレーションされます。GPIO[52]と GPIO[50]は入力として設定され、GPIO[52]が Xilinx 社の FPGA からの INIT_B のピンを、GPIO[50]が Xilinx FPGA からの DONE 信号を監視するために使われます。cyfxconfigfpga.c ファイルにある CyFxConfigFpgaApplnInit () 関数内に、この抜粋したコードがあります。

```
/* Init the GPIO module */
        gpioClock
.fastClkDiv = 2;
        gpioClock.slowClkDiv = 0;
      gpioClock.simpleDiv = CY_U3P_GPIO_SIMPLE_DIV_BY_2;
      gpioClock.clkSrc = CY_U3P_SYS_CLK;
      gpioClock.halfDiv = 0;
apiRetStatus = CyU3PGpioInit(&gpioClock, NULL);

/* Configure GPIO 52 as input */
      gpioConfig.outValue = CyTrue;
        gpioConfig.inputEn = CyTrue;
        gpioConfig.driveLowEn = CyFalse;
        gpioConfig.driveHighEn = CyFalse;
        gpioConfig.intrMode = CY_U3P_GPIO_INTR_BOTH_EDGE;
apiRetStatus = CyU3PGpioSetSimpleConfig(FPGA_INIT_B, &gpioConfig);

/* Configure GPIO 50 as input */
apiRetStatus = CyU3PGpioSetSimpleConfig(FPGA_DONE, &gpioConfig);
```

3.3.3 データ転送をセットアップするために DMA チャネルの作成

DMA マニュアル チャネルはプロデューサーUSB ソケットとコンシューマーCPU ソケット間で作成され、よって FX3 のバルク OUT エンドポイント (0x01) で受信したコンフィギュレーション データは手動で SPI モジュールに向けられることができます。DMA マニュアル チャネルを生成するために役立つコードを以下に示します。cyfxconfigfpga.c ファイルにある CyFxConfigFpgaApplnStart() 関数内に、この抜粋したコードがあります。

```
/* Create a DMA MANUAL channel for U2CPU transfer. The DMA size is set
based on the USB speed. */
    dmaCfg.size  = size;
    dmaCfg.count = CY_FX_SLFIFO_DMA_BUF_COUNT;
    dmaCfg.prodSckId = CY_FX_PRODUCER_USB_SOCKET;
    dmaCfg.consSckId = CY_U3P_CPU_SOCKET_CONS;
    dmaCfg.dmaMode = CY_U3P_DMA_MODE_BYTE;
/* Enabling the callback for produce event. */
    dmaCfg.notification = 0;
    dmaCfg.cb = NULL;
    dmaCfg.prodHeader = 0;
    dmaCfg.prodFooter = 0;
    dmaCfg.consHeader = 0;
    dmaCfg.prodAvailCount = 0;

apiRetStatus = CyU3PDmaChannelCreate
(&glChHandleUtoCPU,CY_U3P_DMA_TYPE_MANUAL_IN,
                                        &dmaCfg);
```

3.3.4 FPGA コンフィギュレーション ユーティリティと FX3 ファームウェア間の通信

PC の FPGA コンフィギュレーション ユーティリティで動作するアプリケーションからの FX3 のファームウェア機能を制御するには 2 つのベンダー コマンドを使用します。FX3 のファームウェアは受信するベンダー コマンドに基づいてイベントをセットします。コンフィギュレーション ビット ファイルの長さと共にベンダー コマンド 0xB2 (VND_CMD_SLAVESER_CFGLOAD) を受信した後、FPGA コンフィギュレーションを開始するためにイベント CY_FX_CONFIGFPGAAPP_START_EVENT をセットします。ファームウェアはまた、ベンダー コマンド 0xB1 (VND_CMD_SLAVESER_CFGSTAT) を受信した後、FPGA コンフィギュレーションが正常に完了した場合だけ、スレーブ FIFO インターフェースに切り替えるためにイベント CY_FX_CONFIGFPGAAPP_SW_TO_SLFIFO_EVENT をセットします。以下のコードの抜粋は、それをするために使用されています。cyfxslfifosync.c ファイルにある CyFxSlFifoApplnUSBSetupCB () 関数内に、このコードがあります。

```
if (bRequest == VND_CMD_SLAVESER_CFGLOAD)
   {
     if ((bReqType & 0x80) == 0)
          {
          CyU3PUsbGetEP0Data (wLength, glEp0Buffer, NULL);
          filelen = uint32_t)(glEp0Buffer[3]<<24)|(glEp0Buffer[2]<<16)|
                              (glEp0Buffer[1]<<8)|glEp0Buffer[0];
          glConfigDone = CyTrue;
/* Set CONFIGFPGAAPP_START_EVENT to start configuring FPGA */
          CyU3PEventSet(&glFxConfigFpgaAppEvent,
                        CY_FX_CONFIGFPGAAPP_START_EVENT, CYU3P_EVENT_OR);
          isHandled = CyTrue;
         }

   }
if (bRequest == VND_CMD_SLAVESER_CFGSTAT)
   {
      if ((bReqType & 0x80) == 0x80)
           {
          glEp0Buffer [0]= glConfigDone;
          CyU3PUsbSendEP0Data (wLength, glEp0Buffer);
/* Switch to slaveFIFO interface when FPGA is configured successfully*/
          if (glConfigDone)
           CyU3PEventSet(&glFxConfigFpgaAppEvent,
                         CY_FX_CONFIGFPGAAPP_SW_TO_SLFIFO_EVENT,
                         CYU3P_EVENT_OR);
          isHandled = CyTrue;
          }
   }
```

3.3.5 イベントに基づいたアクション

FX3 のファームウェアは連続して上述のイベントを検索し、それらのイベントに対応するアクションを実施します。cyfxslfifosync.c 内の SlFifoAppThread_Entry() は以下のコードを含みます。

```
/* Wait for events to configure FPGA */
txApiRetStatus = CyU3PEventGet (&glFxConfigFpgaAppEvent,
                                (CY_FX_CONFIGFPGAAPP_START_EVENT |
                                 CY_FX_CONFIGFPGAAPP_SW_TO_SLFIFO_EVENT),
                                   CYU3P_EVENT_OR_CLEAR, &eventFlag,
                                 CYU3P_WAIT_FOREVER);
if (txApiRetStatus == CY_U3P_SUCCESS)
   {
         if (eventFlag & CY_FX_CONFIGFPGAAPP_START_EVENT)
          {
           /* Start configuring FPGA */
         CyFxConfigFpga(filelen);
          }
         else if ((eventFlag & CY_FX_CONFIGFPGAAPP_SW_TO_SLFIFO_EVENT))
          {
           /* Switch to SlaveFIFO interface */
      CyFxConfigFpgaApplnStop();
      CyFxSwitchtoslFifo();
      CyFxSlFifoApplnInit();
      CyFxSlFifoApplnStart();
         }
  }
```

3.4 スレーブ シリアル インターフェースの実装

CyFxConfigFpga は Xilinx 社のスレーブ シリアル インターフェースを実装するための関数です。コンフィギュレーション プロセスを開始するために、FX3 は PROGRAM_B を LOW に駆動します。FX3 は INIT_B が LOW になるまで待機し、INIT_B が再び HIGH になるとデータをクロックし始めます。FPGA にすべてのコンフィギュレーション データを送信した後、FX3 は DONE 信号に基づいてコンフィギュレーションが正常に完了したかどうかを判定します。コンフィギュレーションが正常に完了した場合、DONE 信号は HIGH にされます。タイミング図として、より判り易くした図 1 をご覧ください。この関数は cyfxconfigfpga.c にあります。

```
/* This is the function that writes configuration data to the Xilinx FPGA
*/
CyU3PReturnStatus_t CyFxConfigFpga(uint32_t uiLen)
{
      uint32_t uiIdx;
      CyU3PReturnStatus_t apiRetStatus;
      CyU3PDmaBuffer_t inBuf_p;
      CyBool_t xFpga_Done, xFpga_Init_B;

/* Pull PROG_B line to reset FPGA */
      apiRetStatus = CyU3PSpiSetSsnLine (CyFalse);
      CyU3PGpioSimpleGetValue (FPGA_INIT_B, &xFpga_Init_B);
      CyU3PGpioSimpleGetValue (FPGA_INIT_B, &xFpga_Init_B);
              if (xFpga_Init_B)
              {
              glConfigDone = CyFalse;
              return apiRetStatus;
              }
      CyU3PThreadSleep(10);
/* Release PROG_B line */
      apiRetStatus |= CyU3PSpiSetSsnLine (CyTrue);
      CyU3PThreadSleep(10);   // Allow FPGA to startup

/* Check if FPGA is now ready by testing the FPGA_Init_B signal */
    apiRetStatus |= CyU3PGpioSimpleGetValue (FPGA_INIT_B, &xFpga_Init_B);
    if( (xFpga_Init_B != CyTrue) || (apiRetStatus != CY_U3P_SUCCESS) ){

          return apiRetStatus;
    }
/* Start shifting out configuration data */
   for(uiIdx = 0; (uiIdx < uiLen) && glIsApplnActive; uiIdx +=
uiPacketSize )
      {
      if(CyU3PDmaChannelGetBuffer (&glChHandleUtoCPU, &inBuf_p, 2000) !=
      CY_U3P_SUCCESS){
        glConfigDone = CyFalse;
        apiRetStatus = CY_U3P_ERROR_TIMEOUT;
      break;
      }
      apiRetStatus = CyU3PSpiTransmitWords(inBuf_p.buffer , uiPacketSize);
      if (apiRetStatus != CY_U3P_SUCCESS)
      {
        glConfigDone = CyFalse;
        break;
      }
      if(CyU3PDmaChannelDiscardBuffer (&glChHandleUtoCPU) !=
CY_U3P_SUCCESS)
```

```
        {
         glConfigDone = CyFalse;
        apiRetStatus = CY_U3P_ERROR_TIMEOUT;
         break;
        }
    }
    CyU3PThreadSleep(1);

    apiRetStatus |= CyU3PGpioSimpleGetValue (FPGA_DONE, &xFpga_Done);
    if( (xFpga_Done != CyTrue) )
    {
      glConfigDone = CyFalse;
    apiRetStatus = CY_U3P_ERROR_FAILURE;
    }
    return apiRetStatus;
}
```

3.5 I/O マトリックスのリコンフィギュレーション

すべてのコンフィギュレーション データが FX3 に送信された後、FPGA コンフィギュレーション ユーティリティは自動的に 0xB1 (VND_CMD_SLAVESER_CFGSTAT) ベンダー コマンドを送信します。FX3 ファームウェアは、FPGA の設定が正常に完了した場合にのみ、スレーブ FIFO インターフェースに切り替えます。以下のコード部分は I/O マトリックスを再設定するために使用されます。同じ I/O リソースが最終アプリケーションで使用されている場合は、I/O マトリックスをリコンフィギュレーションする必要がありません。しかし、この場合では、(AN65974 から入手した) スレーブ FIFO ファームウェアが GPIF II で 32 ビットのインターフェースを使用するため、I/O マトリックスをリコンフィギュレーションする必要があります。影響を受けたすべての周辺モジュールは、I/O のマトリックスをリコンフィギュレーションする前に非初期化されていることを確認します。このアプリケーションでは、I/O のマトリックスを再設定する前に GPIO および SPI モジュールは非初期化されます。I/O マトリックスのコンフィギュレーションは以下に示す 32 ビットのスレーブ FIFO インターフェースとして動作する必要があります。cyfxslfifosync.c ファイルにある CyFxSwitchtoslFifo () 関数内に、この抜粋コードがあります。

```
    io_cfg.useUart   = CyTrue;
    io_cfg.useI2C    = CyFalse;
    io_cfg.useI2S    = CyFalse;
    io_cfg.useSpi    = CyFalse;
#if (CY_FX_SLFIFO_GPIF_16_32BIT_CONF_SELECT == 0)
    io_cfg.isDQ32Bit = CyFalse;
    io_cfg.lppMode   = CY_U3P_IO_MATRIX_LPP_UART_ONLY;
#else
    io_cfg.isDQ32Bit = CyTrue;
    io_cfg.lppMode   = CY_U3P_IO_MATRIX_LPP_DEFAULT;
#endif
    /* No GPIOs are enabled. */
    io_cfg.gpioSimpleEn[0]  = 0x00000000;
    io_cfg.gpioSimpleEn[1]  = 0;
    io_cfg.gpioComplexEn[0] = 0;
    io_cfg.gpioComplexEn[1] = 0;
    status = CyU3PDeviceConfigureIOMatrix (&io_cfg);
```

3.5.1 エンドポイント コンフィギュレーションおよびシーケンス番号の復元

同じプロデューサーのエンドポイント (EP1 OUT BULK) は FPGA コンフィギュレーションのためにまたは FPGA のコンフィギュレーションが正常に完了した後、スレーブ FIFO インターフェースを介して FX3 に接続している FPGA に USB からのデータを転送するために使用されます。しかし、スレーブ FIFO インターフェースが有効にされた後、EP1 はバースト転送で広帯域のデータ転送をサポートできるように再設定されます。従って、CyU3PSetEpConfig API は同じエンドポイントを設定するために二度呼び出されます。この API は、エンドポイントと関連したシーケンス番号をクリアします。USB 3.0 のホストと FX3 デバイスがシーケンス番号で不適当な組合せを発見すれば、データ転送は失敗します。従って、USB 3.0 のホストが EP1 を再設定した後にさえ正常にデータ転送を行うことができるようにユーザーはシーケンス番号を復元する必要があります。これは USB3.0 データの転送だけに有効です。

`CyU3PUsbGetEpSeqNum` API は現時点のシーケンス番号をエンドポイントに取り、`CyU3PUsbSetEpSeqNum` はアクティブシーケンス番号をエンドポイントに設定します。

3.6 コンフィギュレーション ファームウェアをユーザーの設計に統合

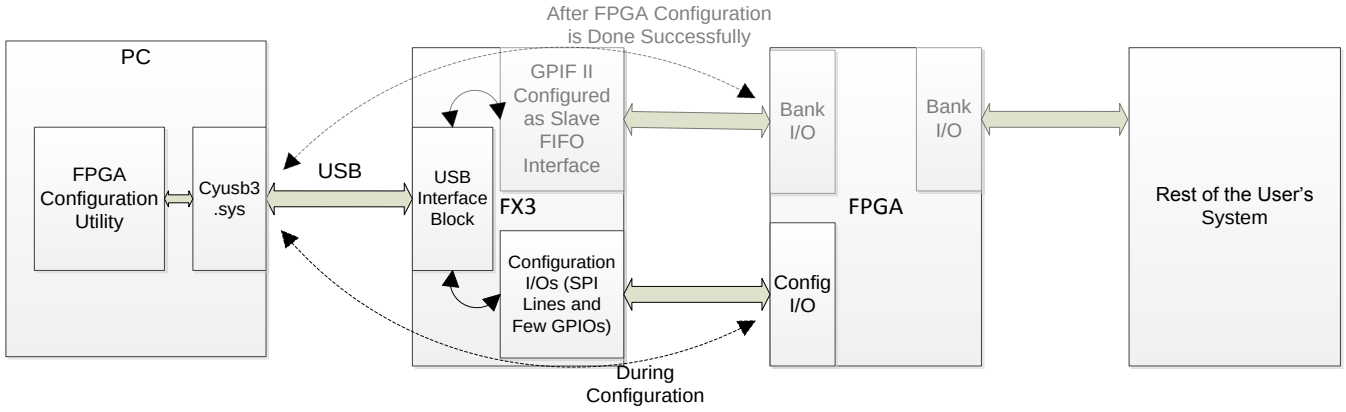
本節では、コンフィギュレーション ファームウェアをユーザーの設計に統合する方法について説明します。以下に説明する手順を読む際、本アプリケーション ノートに添付されているプロジェクトを参照してください。

1. `cyfxconfigfpga.c` および `cyfxconfigfpga.h` ファイルをユーザーのプロジェクトにインポートします。
2. `main`() 関数が `cyfxconfigfpga.c` に実装されているため、ユーザー設計の中で `main`() をコメントにします。
3. ユーザー アプリケーションの初期化関数の代わりにスレッドのエントリ関数内で `CyFxConfigFpgaApplnInit`() を呼び出します。この例では、`CyFxConfigFpgaApplnInit`() は `CyFxSlFifoApplnInit`()の代わりに関数 `SlFifoAppThread_Entry`()内で呼び出されます。
4. USB イベント コールバック関数内でユーザーのアプリケーション スタート関数の代わりに `CyFxConfigFpgaApplnStart`() を呼び出します。この例では、`CyFxConfigFpgaApplnStart`() は、`CyFxSlFifoApplnStart`() の代わりに `CyFxSlFifoApplnUSBEventCB` 関数内で呼び出されます。
5. `CyFxConfigFpgaApplnInit`() が既に USB エニュメレーション部分を処理しているため、同じ処理をするコード部分である `CyFxSlFifoApplnInit`() をコメントアウトします。
6. このサンプルに実装されているように、ベンダー コマンドおよびイベントに応じたサポートを追加します。
7. ユーザーのアプリケーションが異なるリソース セットを必要とする場合は、I/O マトリックスを再設定する必要があります。この例では、I/O マトリックスのリコンフィギュレーション コードは `cyfxslfifosync.c` にある `CyFxSwitchtoslFifo` () 関数で見つけられます。
8. `SlFifoAppThread_Entry`() と同様に、アプリケーションのスレッドのエントリ関数を変更します。

3.7 ソフトウェアの詳細

本節では、本アプリケーションノートに添付されたプロジェクト ファイルを実行するために必要な USB 3.0 ドライバとホスト アプリケーションを説明します。図 5 は PC で FPGA を FX3 にインターフェースするための設定に必要となるドライバとホスト アプリケーションを含むシステム レベルのブロック図を示します。

図 5. PC 側でソフトウェア詳細を示すシステム レベルのブロック図



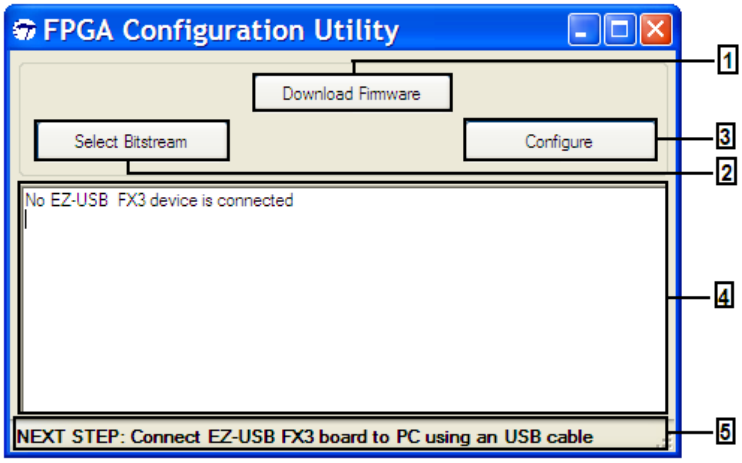
3.7.1 ホストアプリケーション

FPGA コンフィギュレーション ユーティリティはこのアプリケーション専用に開発され、添付ファイルとして提供されています。

USB ドライバ: *cyusb3.inf* および *cyusb3.sys* は EZ-USB FX3 SDK の一部です。

FPGA をコンフィギュレーションするために作成されたサンプルのホスト アプリケーションである FPGA コンフィギュレーション ユーティリティはデザインに含まれています。アプリケーションはサイプレスの SuperSpeed USBSuite に含まれるサイプレス アプリケーション開発ライブラリ *CyUSB.dll* を使用して Visual C# 2008 Express Edition 内で開発されます。デバイスは、サイプレスによって開発された汎用ドライバである *CyUSB3.sys* に結合される必要があります。このアプリケーション ノートと共に提供されるホスト アプリケーションは FPGA コンフィギュレーション ユーティリティを開発するためのリファレンスとして役立ちます。これは、FX3 の RAM 内にファームウェア イメージをダウンロードするための一つのオプション、および Xilinx 社の FPGA コンフィギュレーションに対応した bitstream (*.bin*) ファイルを選択するための柔軟性を提供しています。さらに、デモを正常に実行するために、このアプリケーションは各ステップの状態を付与して次のステップを示します。FPGA コンフィギュレーション ユーティリティの要素の注釈を図 6 に示します。

図 6. PGA コンフィギュレーション ユーティリティの要素の注釈



1: FX3 の RAM にファームウェア イメージをダウンロード

2: Xilinx FPGA のコンフィギュレーション ファイル (*.bin ファイル*) を選択

3: 選択したコンフィギュレーション ファイルを FX3 経由でダウンロード

4: Xilinx FPGA のコンフィギュレーション中に各ステップの状態を表示

5: 次のステップを表示

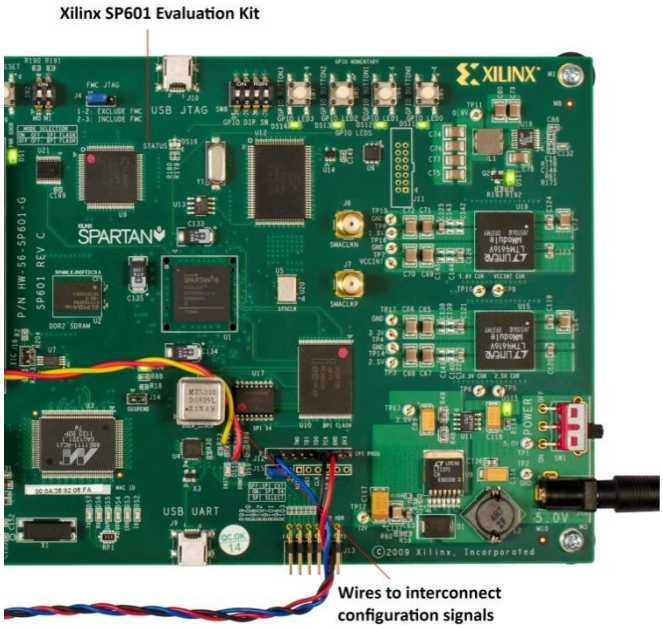
4 動作説明

このセクションでは、このアプリケーション ノートに添付されているソフトウェアとファームウェア プロジェクトを使用して FX3 スーパー スピード エクスプローラー キットに接続する Xilinx 社の FPGA を設定する方法について説明します。表 3 に示すように、Xilinx Spartan-6 SP601 評価キットと FX3 スーパー スピード エクスプローラー キット(または CYUSB3KIT-001)間のハードウェア接続を行います。これらの接続はハードウェア相互接続図 (図 2) に示されたものと同様です。また、Samtec—FMC コネクタを使用して FX3 スーパー スピード エクスプローラー キット(または CYUSB3KIT-001)を Xilinx Spartan-6 SP601 評価キットに接続します。このアプリケーション ノートに使用されるハードウェア セットアップは AN65974 に使用されるものと同じですが、FPGA の設定に必要な信号を接続するために 5 本のワイヤーが必要なことに注意してください。

表 3. Xilinx SP601 評価キットと FX3 エクスプローラキット(または CYUSB3KIT-001) 間のハードウェア接続

信号名	SP601 評価キットのピン配置	FX3 スーパースピード エクスプローラキットのピン配置	FX3 DVK のピン配置
PROGRAM_B	J12 のピン 1	J7 のピン 23	J102 のピン 2
INIT_B	抵抗 R90 の一端 (図 7 を参照)	J7 のピン 31	J20 のピン 6
CCLK	J12 のピン 7	J7 のピン 27	J101 のピン 2
DIN	J12 のピン 6	J7 のピン 19	J104 のピン 2
DONE	R113 または LED DS9 の一端 (図 7 を参照)	J7 のピン 37	J20 のピン 4

図 7. Xilinx SP601 評価キットのハードウェア接続



1. *FPGA_Config_Utility\bin\Debug* フォルダにある *Template.exe* を実行すると、画面にユーティリティが表示されます。次のステータス メッセージが表示されます： **No EZ-USB FX3 device is connected**
2. 図 8 のように USB ケーブルを使用して EZ-USB FX3 エクスプローラキットまたは CYUSB3KIT-001 を PC に接続します。そして、図 9 が示すように、テキストボックスに表示される「**EZ-USB FX3 Bootloader device connected**」 (EZ-USB FX3 ブートローダ デバイスが接続中) というステータス メッセージを確認します。
3. 図 10 に示すように、FX3 RAM にファームウェア イメージをダウンロードする、または *ConfigFpgaSlaveFifoSync.img* ファイルのロケーションにブラウズするために「**Download Firmware**」(ファームウェアをダウンロード) をクリックします。そして、「**Open**」をクリックします。

図 8. EZ-USB FX3 デバイスが接続されていない時の FPGA コンフィギュレーション ユーティリティ

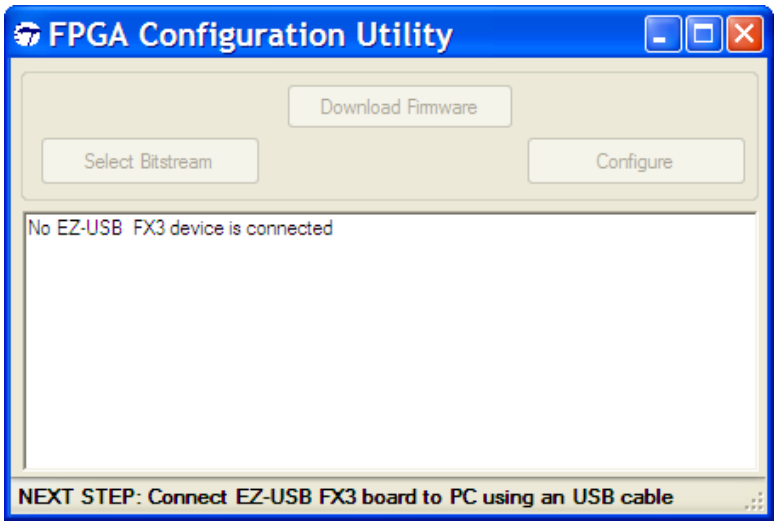


図 9. EZ-USB FX3 DVK を PC に接続した後の FPGA コンフィギュレーション ユーティリティ

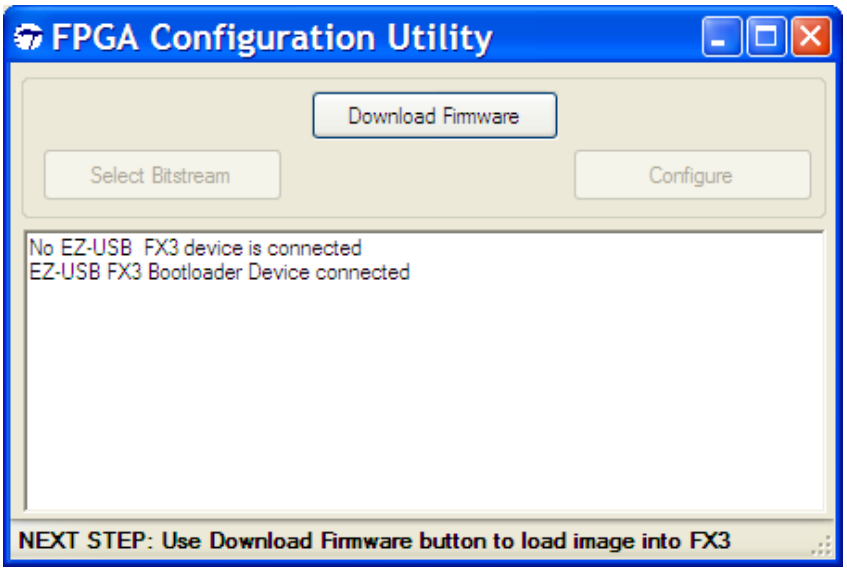


図 10. FX3 のファームウェア イメージを選択

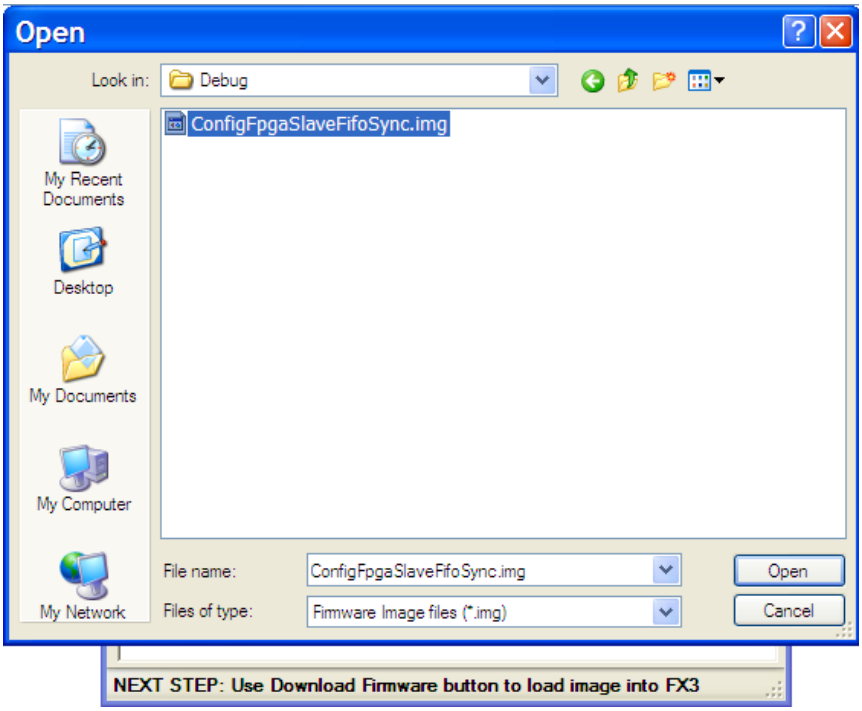


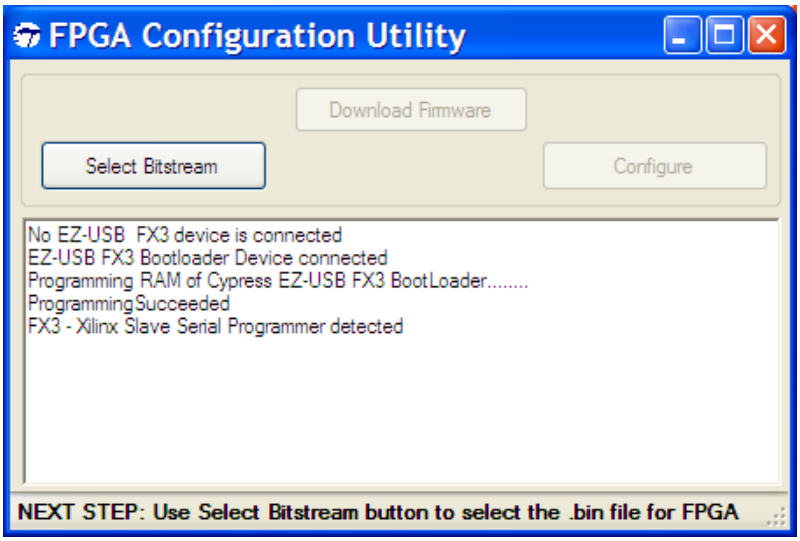
図 11 に示すように、以下のステータス メッセージが表示されます。

Programming RAM of Cypress EZ-USB FX3 BootLoader.....

Programming Succeeded

FX3 – Xilinx Slave Serial Programmer detected

図 11. イメージ ファイルが FX3 の RAM にダウンロードされた後の FPGA コンフィギュレーション ユーティリティ

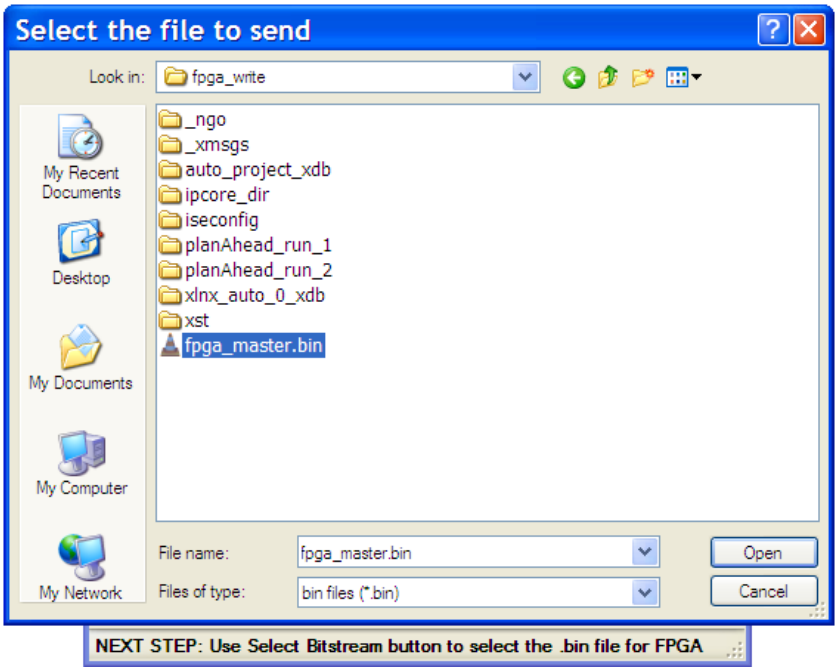


4. メッセージがユーティリティに表示された時に、「**Select Bitstream**」 (ビットストリームを選択) をクリックして FPGA 用の*.bin* ファイルを選択します。

   .bin ファイルが無く、*.bit* ファイルだけがある場合は、PromGen コマンドラインを使用して、*.bit* を*.bin* に変換してください。あるいは、iMPACT PROM ファイルフォーマッタを使用して Xilinx 社の PROM 用の*.bin* を作成できます。*.bin* ファイルの生成のサポートは www.xilinx.com/support.html を参照してください。

5. 図 12 に示すように、*fpga_master.bin* ファイルのロケーションをブラウズします。**Open** をクリックしてください。

図 12. Xilinx 社の FPGA に対応したコンフィギュレーション ビット ファイル (.bin) を選択します。



6. 図 13 が示すように、Xilinx 社の FPGA をコンフィギュレーションするために「**Configure**」ボタンをクリックします。FPGA が正常にコンフィギュレーションされた場合は、FX3 のファームウェアはスレーブ FIFO インターフェースに切り替わります。図 14 に示すように、以下のステータス メッセージが表示されます。

   Writing data to FPGA

   Configuration data has been sent to FPGA

   Configurations Successful

   FX3 Slave FIFO interface is activated

図 13. .bin ファイルを選択した後の FPGA コンフィギュレーション ユーティリティ

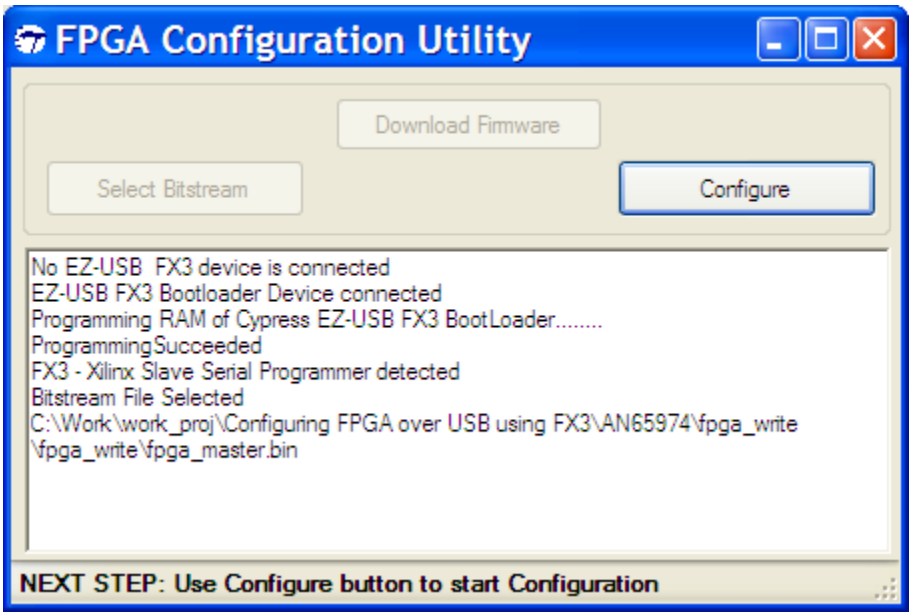
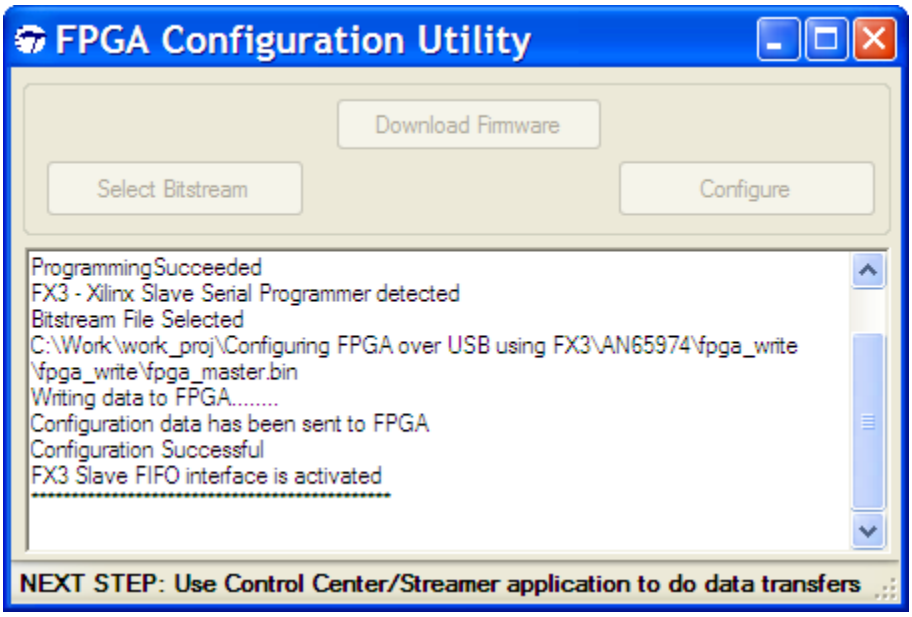
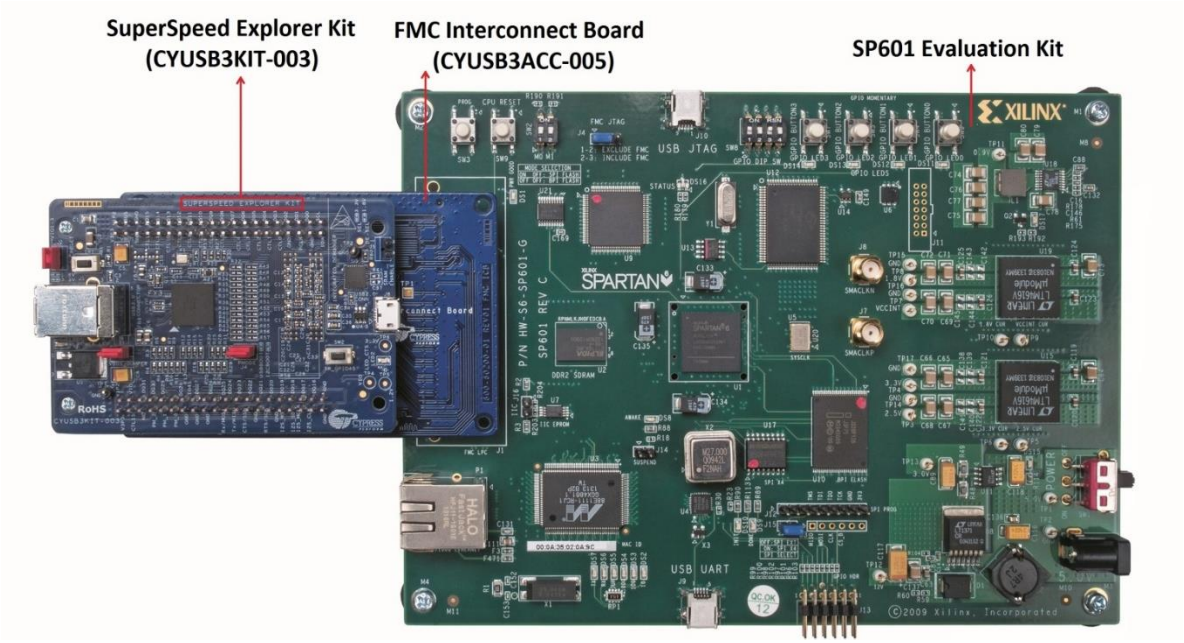


図 14. FPGA コンフィギュレーションが正常に完了した後の FPGA コンフィギュレーション ユーティリティ



コンフィギュレーションに成功した後に Xilinx 社の FPGA 基板上で DS9 LED が点灯することが確認できます。コンフィギュレーション中に何か問題が発生した場合は点灯しません。図 15 は点灯している DS9 LED を示します。

図 15. FPGA が正常にコンフィギュレーションされた後のハードウェア セットアップ

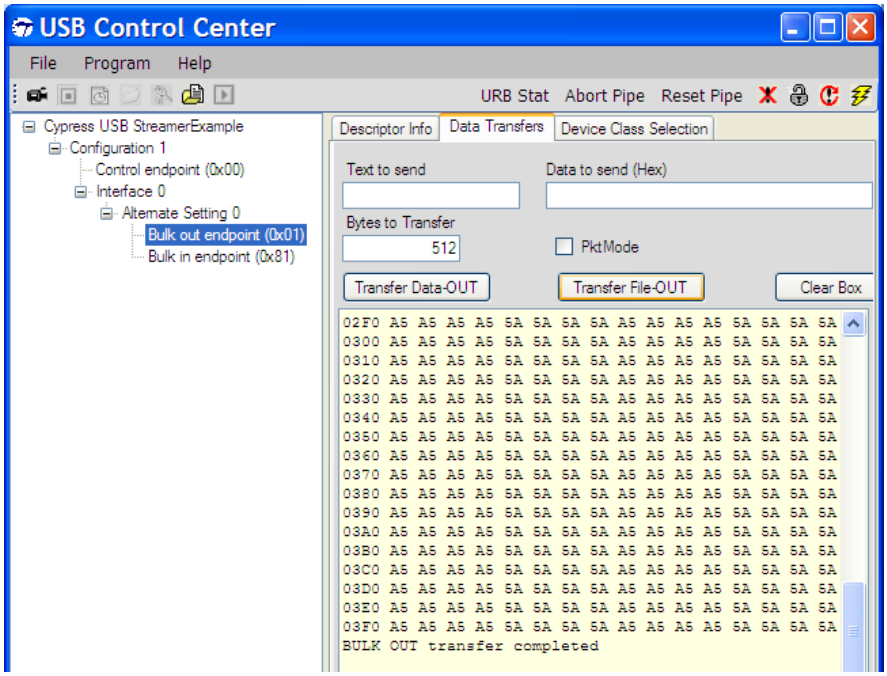


7. FX3 と Xilinx 社の FPGA の間のループバック動作を確認するために、Control Center アプリケーションを使用します。SP601 評価キットの SW8 スイッチが次のモードで保持されていることを確認してください。

SW8[1]	SW8[2]	SW8[3]	SW8[4]
OFF	OFF	OFF	ON

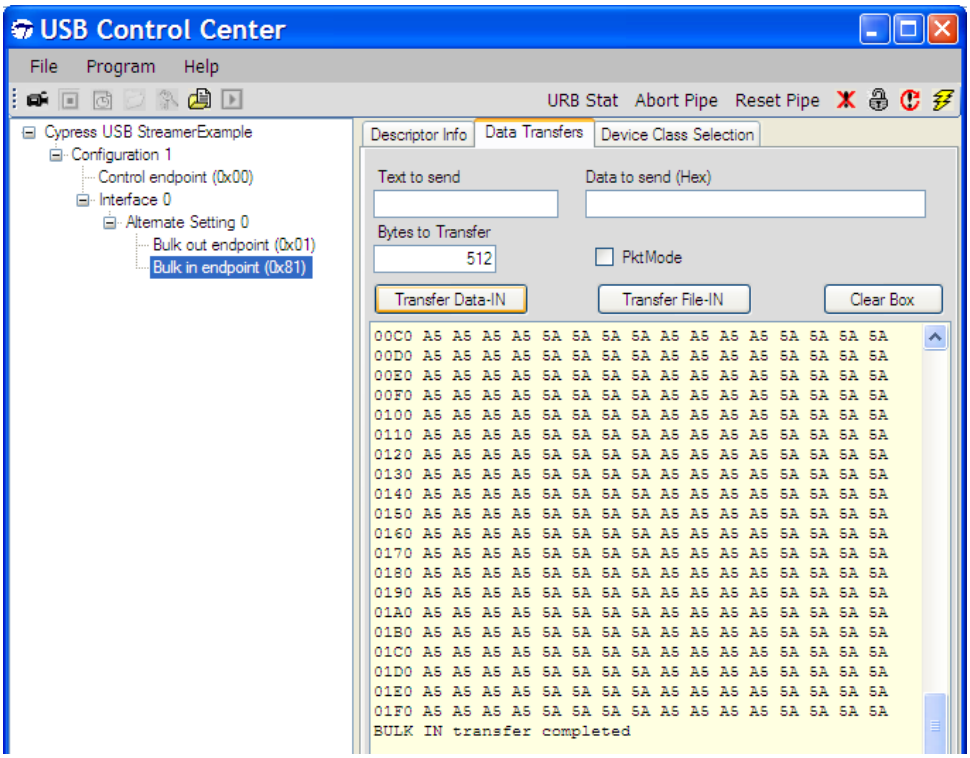
「**Bulk out endpoint (0x01)**」 (バルク OUT エンドポイント(0x01)) に移動して、「**Transfer File-OUT**」ボタンをクリックし同じフォルダにある *TEST.txt* と呼ばれるファイルを転送します。そして、図 16 に示すように、一連の **A5 A5 A5 A5 5A 5A 5A 5A** が「**Bulk out endpoin**」 (バルク OUT エンドポイント) に正常に転送されることを確認できます。

図 16. バルク OUT エンドポイント経由で TEST.txt ファイルを転送した後の USB Control Center



8. 「**Bulk in endpoint**」 (バルク IN エンドポイント) を選択し、「**Transfer Data-IN**」をクリックします。図 17 に示すように、受信したデータが「**Bulk out endpoint**」 (バルク OUT エンドポイント) に転送されたデータと同じであることを観察できます。データパスは、**Control Center > Bulk out endpoint of FX3 > FPGA reads the data from Bulk out endpoint > FPGA writes the same data to Bulk in endpoint of FX3 > Control Center** です。

図 17. Data-IN を実行してバルク IN エンドポイントからデータを取得した後の USB Control Center



5 まとめ

本アプリケーション ノートは、サイプレスの FX3 を使用して USB 経由で Xilinx 社の FPGA を効率的にコンフィギュレーションするためのソリューションを説明しました。FPGA が USB 3.0 機能に対応して FX3 とのインターフェースとして作動するシステムにこのソリューションを統合できます。これにより、FPGA をコンフィギュレーションする専用のプログラミング回路が不要になります。

6 関連プロジェクト ファイル

表 4 は、本アプリケーション ノートに添付されているファイルについて説明します。

表 4. 添付のファイルの説明

フォルダ名	内容
FPGA Configuration Utility	PC 側のアプリケーションのソースコード
FX3 Firmware	FX3 のファームウェアのソースコード
fpga_write	マスター デバイスとして機能する Xilinx 社 FPGA のソースコード。AN65974。で紹介した FPGA のコードと同じです。
bin	以下のファイルを含みます。 *TEST.txt* – FX3 と FPGA 間のループバック動作をテストするために使えるデータ ファイル。 *ConfigFpgaSlaveFifoSync.img* – FX3 のファームウェアのイメージ ファイル *Template.exe* – FPGA コンフィギュレーション ユーティリティの実行ファイル

7 参考資料

- CYUSB3014 データシート
- EZ-USB FX3 入門
- スレーブ FIFO インターフェース
- Spartan-6 ジェネレーション コンフィギュレーション ユーザーガイド– Xilinx 社の UG380
- Spartan-6 FPGA SP601 評価キット
- マイクロプロセッサを使用してスレーブ シリアルまたは SelectMAP モードで Xilinx FPGA をコンフィギュレーション
- FX3 + FPGA + HelionVision ISP ベースインダストリアルカメラ リファレンスデザイン – KBA222700

著者について

氏名: Rama Sai Krishna

役職: アプリケーション エンジニアスタッフ

改版履歴

文書名: AN84868 – サイプレスの EZ-USB FX3 を使用した USB 経由での Xilinx FPGA コンフィギュレーション

文書番号: 001-98028

版	ECN	変更者	発行日	変更内容
**	4802516	HZEN	07/13/2015	これは英語版 001-84868 Rev. *B を翻訳した日本語版 001-98028 Rev. **です。
*A	5801166	AESATMP8	07/06/2017	更新されたロゴと著作権。
*B	5814743	HIKA	07/13/2017	これは英語版 001-84868 Rev. *D を翻訳した日本語版 001-98028 Rev. *B です。
*C	6250166	HIKA	0724//2018	これは英語版 001-84868 Rev. *E を翻訳した日本語版 001-98028 Rev. *C です。

セールス，ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

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

製品

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

PSoC® ソリューション

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

サイプレス開発者コミュニティ

コミュニティ | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709