

如何使用 EZ-USB FX3 将图像传感器连接到 USB 视频类别 (UVC) 框架内

作者: Karnik Shah

相关器件系列: CYUSB301X、CYUSB201X

相关应用笔记 AN75705、AN90369

要想获得本应用笔记的最新版本或相关项目文件, 请访问 www.cypress.com/go/AN75779。

更多代码示例? 我们明白。

要想获取 USB 超高速代码示例的综合列表, 请访问 <http://www.cypress.com/101781>。

USB 3.0 所提供的高带宽对于将外设连接至 USB 的各个 IC 提出很高的要求。本应用笔记重点介绍了一种 USB 3.0 的流行应用: 摄像机 (即连接到 EZ-USB® FX3™ 的图像传感器) 将非压缩数据输送给 PC。应用笔记突出介绍了特定设计 FX3 的特性, 能够最大化数据吞吐量而不影响到接口的灵活性。本应用笔记也提供了有关 USB 视频类别 (UVC) 的实施细节。符合该类别的摄像设备能够使用内置的 PC 驱动程序和主机应用程序 (如 AMCap、MPC-HC 和 VLC 媒体播放器) 进行操作。最后, 本应用手册还指出了如何使用 FX3 中的灵活图像传感器接口来连接两个图像传感器, 以便实现三维 (3D) 图像和移动跟踪应用。

目录

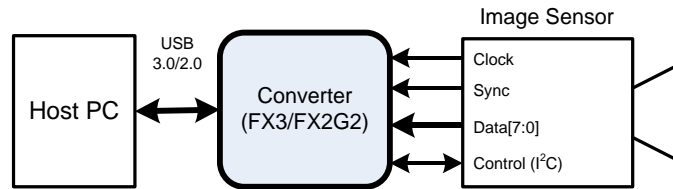
1 简介	2	5.7 在视频流期间中处理 DMA 缓冲区	46
2 USB 视频类别 (UVC)	3	5.8 终止视频流	46
2.1 枚举数据	3	5.9 增加“调试”接口	47
2.2 操作码	4	6 硬件设置	57
2.3 USB 视频类别要求	4	6.1 硬件要求	57
3 GPIF II 图像传感器接口	13	6.2 SuperSpeed Explorer 套件电路板安装包	57
3.1 图像传感器接口	13	7 基于 UVC 的主机应用	60
3.2 图像传感器接口的引脚映射情况	14	7.1 运行演示	60
3.3 交替 DMA 缓冲区	15	8 故障排除	62
3.4 设计策略	16	9 连接两个图像传感器	63
3.5 GPIF II 状态机	17	9.1 使用 UVC 传输交错图像	64
3.6 使用 GPIF II Designer 实现图像传感器接口	20	9.2 将新的视频分辨率添加到当前项目的固件修改检查表	65
4 设置 DMA 系统	38	9.3 支持新传感器的固件修改检查表	65
4.1 DMA 缓冲区的相关内容	42	10 总结	66
5 FX3 固件	43	附录 A. FX3 DVK (CYUSB3KIT-001) 的硬件安装详细内容	67
5.1 应用线程	45	A.1 FX3 DVK 板安装	67
5.2 初始化	45	附录 B. FX3 Explorer 套件和 Aptina 图像感应互联电路板 (CYUSB3ACC-004) 的硬件设置详细信息	70
5.3 枚举	45	B.1 硬件设置	70
5.4 使用 I ² C 接口配置图像传感器	45		
5.5 启动视频流	46		
5.6 设置 DMA 缓冲区	46		

1 简介

USB 3.0 提供的高带宽对于将外设连接至 USB 的 IC 提出很高的要求。一个流行的示例就是摄像机将非压缩数据输送到 PC 中。本应用笔记介绍了如何通过使用赛普拉斯的 EZ-USB®FX3™ 芯片来将一个转换器的一端连接到图像传感器，另一端则连接到 USB 3.0 主机 PC。FX3 使用它的第二代通用可编程接口（Gen 2（GPIF II））来提供图像传感器接口，将其超速 USB 单元连接至 PC。FX3 固件将图像传感器输出的数据转换为与 USB 视频类（UVC）兼容的格式。符合此类别的摄像设备能够使用 OS 内置的驱动程序进行操作，使摄像机与主机应用（如 AMCap、MPC-HC 和 VLC 媒体播放器）相兼容。

该应用笔记中所描述的各步骤也与 FX2G2（FX3 系列的 USB 2.0 器件）兼容。

图 1. 摄像机应用



注意： 请参考 [AN90369](#)，以了解如何将 MIPI CSI-2 图像传感器连接到 EZ-USB® CX3。

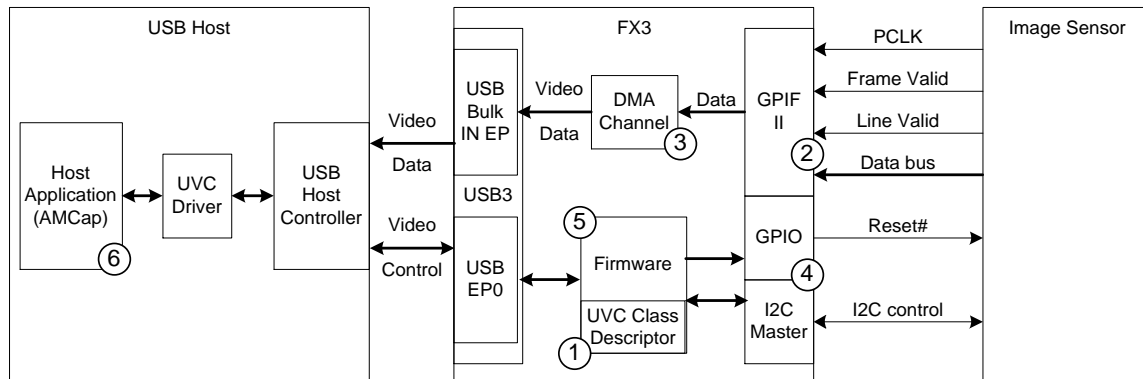
图 1 描述的是摄像机应用。左侧是一个带有超速 USB 3.0 端口的 PC 而右侧是一个具有下面各项特性的图像传感器：

- 8 位同步并行数据接口
- 每个像素为 16 位
- YUY2 颜色空间
- 分辨率为 1280 x 720 个像素（720p）
- 30 帧每秒
- 帧/行有效信号均为高电平有效
- 上升时钟边沿极性

虽然应用笔记讨论了一个特定的图像传感器接口，但这些特性是通用于多种图像传感器接口（即使可能有轻微差异，如数据总线宽度和信号极性）。作为 GPIF II 模块的可编程特性，这些差异是很容易调节适应的。此外，FX3 会使用其 I²C 接口实现控制总线，用作配置图像传感器。

图 2 显示了系统框图更加详细的信息。框图的主要子模块被编号，每个子模块进行的各任务如下面所述：

图 2. 系统框图



1. 提供正确的 USB 描述符，保证主机能够识别符合 UVC 的外设。有关详细信息，请参考第 2 节中的内容。
2. 实现连接到图像传感器的并行总线连接可通过使用 FX3 GPIF II 接口实现该操作。通过使用赛普拉斯的 [GPIF II Designer](#) 工具可以在图形状态机编辑器中自定义设计波形。由于该接口是可编程的，因此微小的更改也能将其自定义用于其他图像传感器（例如 16 位数据总线的图像传感器）。有关详细信息，请参考第 3 节中的内容。
3. 构建一个 DMA 通道，用以将 GPIF II 模块上的图像传感器数据转移到 USB 接口模块 (UIB) 内。在本应用中，必须将标头数据添加到图像传感器视频数据中，以符合 UVC 规范的要求。因此，通过配置 DMA，CPU 可以将所需的标头添加到 DMA 缓冲区内。该通道的设计要确保最大带宽可用于将视频从图像传感器输送到 PC。有关详细信息，请参考第 4 节中的内容。
4. 通过使用 FX3 I²C 主设备可以实现控制总线与图像传感器的连接。通过赛普拉斯的标准库调用（如第 5.4 节所介绍）可对 I²C 和 GPIO 单元进行编程。
5. FX3 固件初始化 FX3 的硬件模块（第 5.2 节），枚举成为一个 UVC 摄像机（第 2.3.1 节），处理 UVC 特定请求（第 2.3.2 节），通过 I²C 接口（第 5.4 节）将视频控制设置（如亮度）传输给图像传感器，将 UVC 标头添加到视频数据流（第 2.3.4 节），将带有标头的视频数据提交给 USB（第 5.7 节），以及维持 GPIF II 状态机（第 5.8 节）。
6. 主机应用（如 AMCap、MPC-HC 或 VLC 媒体播放器）将存取 UVC 驱动程序，以通过 UVC 控制接口来配置图像传感器，并通过 UVC 流接口接收视频数据。欲了解有关基于 UVC 的主机应用的详情，请参考第 7 节。

如果将摄像机插入 USB 2.0 端口，FX3 固件会使用 I²C 控制总线将帧率从 30 FPS 下降到 15 FPS，并将帧大小从 1280 x 720 个像素降低到 640 x 480 个像素，以匹配较低的 USB 带宽。PC 主机可以选用控制接口，将亮度、平移、倾斜和缩放等调整内容传输给摄像机。一般可同时实现平移、倾斜和缩放，并被称为 PTZ。

注意： 该应用笔记描述了如何使用 USB UVC 架构的 FX3 器件实现图像传感器连接。实现 USB 版本 3 连接的 FX3 实例固件已可用。更多详细信息，请参考[论坛帖子](#)。用户应开发自己的主机应用或使用第三方的应用。赛普拉斯不提供任何用于 USB 版本 3 主机应用和驱动程序的解决方案。

2 USB 视频类别 (UVC)

要符合 UVC，则需要下面两个 FX3 代码模块：

- 枚举数据
- 操作码

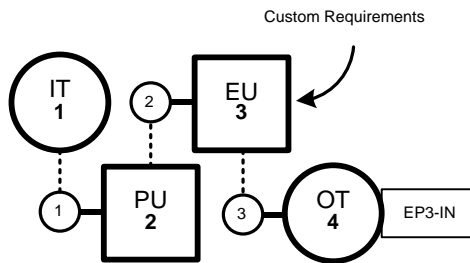
2.1 枚举数据

该应用笔记所附带的代码包括一个 `cyfxuvcdscr.c` 文件（在第 2.3.1 节中说明），该文件包含 UVC 的枚举数据。可从 usb.org 上的视频类别一节获取用于定义 UVC 描述符格式的 USB 规范。本节介绍了顶层视图的描述符。UVC 器件具有以下四个逻辑元素：

1. 输入摄像机终端 (IT)
2. 输出端 (OT)
3. 处理单元 (PU)
4. 扩展单元 (EU)

这些元素在描述符中互相连接，如图 3 所示。通过将描述符中的各终端编号连接起来，可连接上述各元素。例如，输入（摄像机）终端描述符声明了其 ID 为 1；处理单元描述符的输入连接的 ID 为 1，所以逻辑上表示他连接至输入端。输出端描述符将指定要使用的 USB 端点，在该情况下，将使用 BULK-IN 端点 3。

图 3. 摄像机架构的 UVC 图



描述符还包括了视频属性（如宽度、高度、帧率、帧大小和位深度）以及控制属性（如亮度、曝光、增益、对比度和 PTZ）等内容。除了标准的视频控制，扩展单元还根据用户要求提供附加控制。

注意：在 Windows 7/8 电脑上使用 USB 3.0 的 UVC 标准的 ISOC-IN 端点可获取的最大宽度将为 24 Mbps。这是因为 UVC 驱动程序将 ISOC 端点的 BURST 限制为 1。但在 Windows 10 电脑上，为了获取最大的 ISOC 宽度，可将 BURST 置为 15。BULK 端点并没有这个限制。因此，该项目将不使用 ISOC-IN 端点。请参考[论坛帖子](#)以了解更多信息。

2.2 操作码

主机枚举摄像机后，UVC 驱动程序会向摄像机发出一系列请求，以确定工作特征。该操作叫做能力请求阶段。请求阶段先于串流阶段，其中主机应用会启动串流视频。FX3 固件会响应到达 USB 控制端点（EP0）的请求。

例如，假设 UVC 器件表明它支持某个 USB 描述符中的亮度控制。在能力请求阶段，UVC 驱动程序将查询器件，以发现相关的亮度参数。

当主机发出要求更改亮度值的请求时，UVC 驱动程序将发出“SET”（设置）控制请求，用以更改亮度值（SET_CUR）

同样，当主机应用需要选择输送一个受支持的视频格式/帧率/帧大小时，它会发出输送请求。共有两种请求类别：PROBE 和 COMMIT。PROBE 请求用于确定 UVC 器件是否就绪于接收串流模式的更改。串流模式是图像格式、帧大小和帧率的组合。

2.3 USB 视频类别要求

该应用笔记的固件项目位于名称为 **cyfxuvc_an75779** 的文件夹内。本节介绍了示例项目如何满足 UVC 的要求。UVC 要求器件执行下列任务：

- 使用 UVC 特定的 USB 描述符进行枚举
- 处理 USB 描述符中所记录的 UVC 控制和串流能力的 UVC 特定 SET/GET 请求
- 以符合 UVC 的颜色格式进行视频数据串流
- 将符合 UVC 的标头添加到每个图像载荷内

请参考 [UVC 规范](#)，以了解这些要求的详细信息。

2.3.1 UVC 的 USB 描述符

cyfxuvc_dscr.c 文件包含 USB 描述符的表格。字节阵列“CyFxBUSConfigDscr”（高速）和“CyFxBUSConfigDscr”（超速）包含了 UVC 特定的描述符。这些描述符执行下面各子描述符树：

- 配置描述符
 - 接口关联描述符
 - 视频控制（VC）接口描述符
 - VC 接口标头描述符
 - 输入（摄像机）终端描述符
 - 处理单元描述符
 - 扩展单元描述符

- 输出端描述符
 - VC 状态中断端点描述符
- 视频流 (VS) 接口描述符
 - VS 接口输入标头描述符
 - VS 格式描述符
 - VS 帧描述符
- BULK-IN 视频端点描述符

配置描述符是一个标准的 USB 描述符，它定义了 USB 器件在其子描述符中的功能。接口关联描述符用于向主机指示器件符合标准 USB 类别。此处，该描述符还使用两个接口（包括视频控制 (VC) 接口和视频流 (VS) 接口）来报告符合 UVC 的器件。因为 UVC 器件具有两个独立的接口，因此它是一个 USB 符合器件。

2.3.1.1 视频控制接口

VC 接口描述符及其子描述符将报告所有与控制接口有关的能力。示例包括亮度、对比度、色调、曝光和 PTZ 等控制。

VC 接口标头描述符是一个 UVC 特定的接口描述符，它指向该 VC 接口所属的 VS 接口。

输入（摄像机）终端描述符、处理单元描述符、扩展单元描述符以及输出端描述符均包含各个位字段，这些位字段说明了相应终端或单元所支持的特性。

摄像机终端控制着机械（或等效数字）特性，比如传输视频流的器件的曝光和 PTZ。

处理单元控制着图像的各项属性，如通过该单元输送的视频亮度、对比度以及色调。

与标准的 USB 供应商请求相似，扩展单元允许添加供应商指定的特性。在该设计中，扩展单元是空的，但已经实现了一个示例设计，通过启用该示例可以获取或设置器件固件版本。如果优化扩展单元，则标准主机应用将无法识别其特性，除非设计主机应用使其能够识别这些特性。通过所提供的示例主机应用可以获取或设置器件固件版本。您可以设计自己的扩展控制和主机应用，从而可以查询这些控制。受支持的性能包括获取器件固件版本和固件 ID，写入到传感器/图像信号处理器 (ISP) 等等。该器件可以支持多个扩展单元。有关 UVC 扩展单元及相关固件项目的更多详细信息，请参考第 2.3.2 节。

输出端用于描述这些单元 (IT、PU、EU) 和主机之间的接口。VC 状态中断端点描述符是一个用于中断端点的标准 USB 描述符。该端点可用于传输 UVC 特定的状态信息。该端点的此功能属于本应用笔记范围外的内容。

UVC 规范对这些功能进行了分类，您可容易实现类别特定的控制请求。但执行这些功能是应用特定的。通过将相应的功能位设置为 '1'，可以在各自终端/单元描述符的位字段 "bmControls" (`cyfxuvcdscr.c`) 中记录所支持的控制功能。在枚举时，UVC 器件的驱动程序会轮询控制的详细信息。通过 EP0 请求实现轮询的细节。所有相似的请求（包括视频流请求）均由 `uvc.c` 文件中的 `UVCAppEP0Thread_Entry` 函数处理。

2.3.1.2 视频流接口

视频流接口描述符以及它的子描述符记录了各种帧格式（如非压缩、MPEG、H.264）、帧的分辨率（宽度、高度和位深度）以及帧率。根据所记录的值，主机应用可以通过选择所支持的帧格式、帧分辨率和帧率的组合来切换流模式。

VS 接口的输入标头描述符指定了 VS 格式描述符所跟随的数量。

VS 格式描述符包含图像的长宽比和颜色格式，如非压缩或压缩格式。

VS 帧描述符包含图像的分辨率和它的所有支持帧率。如果摄像机支持不同的分辨率，则多个 VS 帧描述符将遵循 VS 格式描述符。

BULK-IN 视频端点描述符是一个标准 USB 端点描述符，它包含有关用于输送视频的 Bulk 端点的信息。

本示例中使用了单一分辨率和帧率。其图像的特性包含在三个描述符中，如下面三表所示（只显示相关字节偏移）。

表 1. VS 格式描述符值

VS 格式描述符的字节偏移	特性	超速值	高速值
23-24	长宽比	16:9	4:3

表 2. VS 帧描述符值

VS 帧描述符字节偏移	特性	超速值	高速值
5-8	分辨率（宽、高）	1280x720	640x480
17-20	图像的最大尺寸，单位为字节	1280x720x2（2 个字节/像素）	640x480x2（2 个字节/像素）
21-24 或 26-29	帧间隔，单位为 100 ns	0x51615（30 FPS）	0xA2C2A（15 FPS）

请注意，多字节数值会先列出最低有效位（LSB）（低位优先），因此，例如帧率为 0x00051615，即 33.33 毫秒，或 30 FPS。

表 3. 探针/提交结构值

探针/提交结构的字节偏移	特性	超速值	高速值
2	格式索引	1	1
3	帧索引	1	1
4-7	帧间隔，单位为 100 ns	0x51615（30 FPS）	0xA2C2A（15 FPS）
18-21	图像的最大尺寸，单位为字节	1280x720x2（2 个字节/像素）	640x480x2（2 个字节/像素）

通过更改上述三表中的输入可以调整该设计，从而支持不同的图像分辨率，比如 1080 p。

2.3.2 UVC 特定的请求

UVC 规范使用了 USB 控制端点 EP0，使之能够向 UVC 器件进行控制和传输所请求的通信。这些请求用于发现并更改与视频相关的控制属性。UVC 规范将这些与视频相关的控制内容定义为能力。通过这些能力您可以更改图像属性或输送视频。一个能力（第一项内容）可以是视频控制属性（如亮度、对比度和色调），也可以是视频流模式的属性（如颜色格式、帧大小和帧率）。通过 USB 配置描述符的 UVC 特定部分，可以记录各种能力。每个能力都带有其属性。能力的各项属性如下：

- 最小值
- 最大值
- 最小值和最大值之间值的数值。
- 默认值
- 当前值

SET 和 GET 是两类 UVC 特定请求。SET 用于更改属性的当前值，而 GET 则用于读取属性。

下面是 UVC 特定请求的列表：

- SET_CUR 是 SET 唯一请求的类型
- GET_CUR 用于读取当前值
- GET_MIN 用于读取支持的最小值。
- GET_MAX 用于读取支持的最大值。
- GET_RES 用于读取分辨率（步长值表示最小值和最大值之间受支持的数值）
- GET_DEF 用于读取默认值
- GET_LEN 用于读取属性大小，单位为字节
- GET_INFO 用于查询特定能力的状态/支持情况。

针对已给的属性，UVC 规范将这些请求定义为强制或可选的。例如，如果 SET_CUR 请求对于一种特殊能力是可选的，则通过 GET_INFO 请求可确定它是否存在。如果摄像机不支持能力的某个请求，在主机向摄像机发出请求时必须通过停止控制端点来指出这一点。

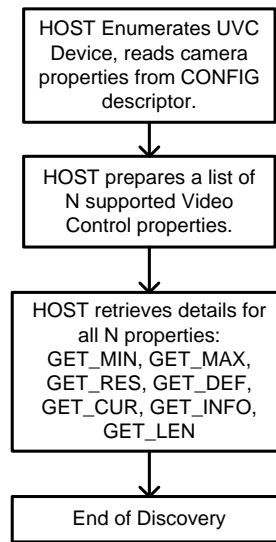
这些请求中的字节字段符合其目标能力。这些字节字段分层次，它具有与第 2.3.1 节中所述的 UVC 特定描述符相同的结构。第一层用以识别接口（视频控制还是视频流）

如果第一层确认接口是视频控制，则第二层将确认终端或者单元，并且第三层确认终端或单元的能力。例如，如果目标能力为亮度控制，则：

- 第一层为视频控制
- 第二层为处理单元
- 第三层为亮度控制

如果第一层确认接口为视频流，则第二层将为 PROBE（探针）或 COMMIT（提交）。这样便不存在第三层。当主机需要 UVC 器件启动输送或更改流模式时，主机将先确定器件是否支持新的流模式。要确定该项，主机将发出一系列的 SET 和 GET 请求，并将第二层置为 PROBE。器件可接收或拒绝流模式的更改。如果器件接收更改请求，则主机将通过发出 SET_CUR 请求并把第二层置为 COMMIT 进行确认。图 6 显示的是主机和器件间的互动。下面三个流程图显示的是主机与 UVC 器件间的交互情况。

图 4. UVC 枚举以及探索流程



当 UVC 插入到 USB 时，主机将枚举它，并发现摄像头支持的属性的细节（图 4）。

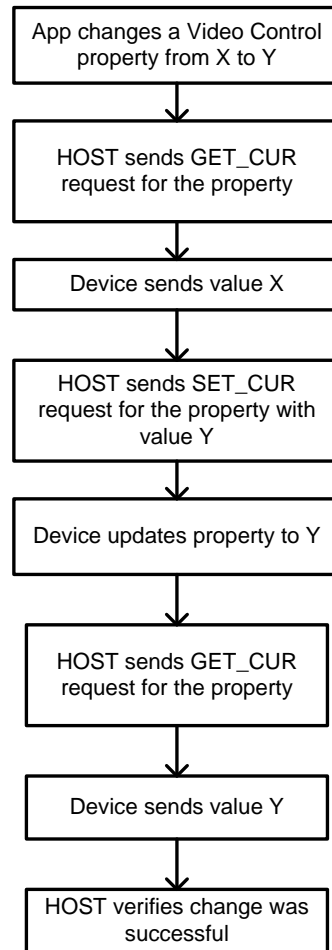
在视频运行期间，摄像机的操作者可以在主机应用所提供的显示对话框中更改摄像机的属性（如亮度）。图 5 显示了这种交互情况。

这种执行是同步控制传输。根据 UVC 规范，一个器件可以支持两种控制请求：同步控制传输和异步控制传输。

标准 UVC 请求（如亮度，平移，倾斜和缩放请求）需要较少的完成时间（少于 10 毫秒）。在这种情况下，UVC 器件将对传感器/ISP 寄存器进行写操作，并不等待传感器/ISP 的任何确认。当主机驱动程序发送 SET_CUR 请求时，器件将在 10 毫秒内响应成功（确认请求）或失败（停止请求）。根据 UVC 规范，该请求本质上是同步的。现在，假设主机驱动程序设置一个曝光值。根据设计实现，传感器/ISP 将在 100 ms 内响应确认。器件始终对 SET_CUR 请求发出成功响应，并且在收到传感器/ISP 的确认之后，将成功或失败地加载控制状态中断端点。这种请求是异步控制请求。请参考 UVC 视频类规范，以了解更多详细信息。该项目配置了控制状态中断（请参阅 `uvc.c` 文件中的 `CY_FX_EP_CONTROL_STATUS`），但并没有使用该中断，因为所有请求都是同步的。

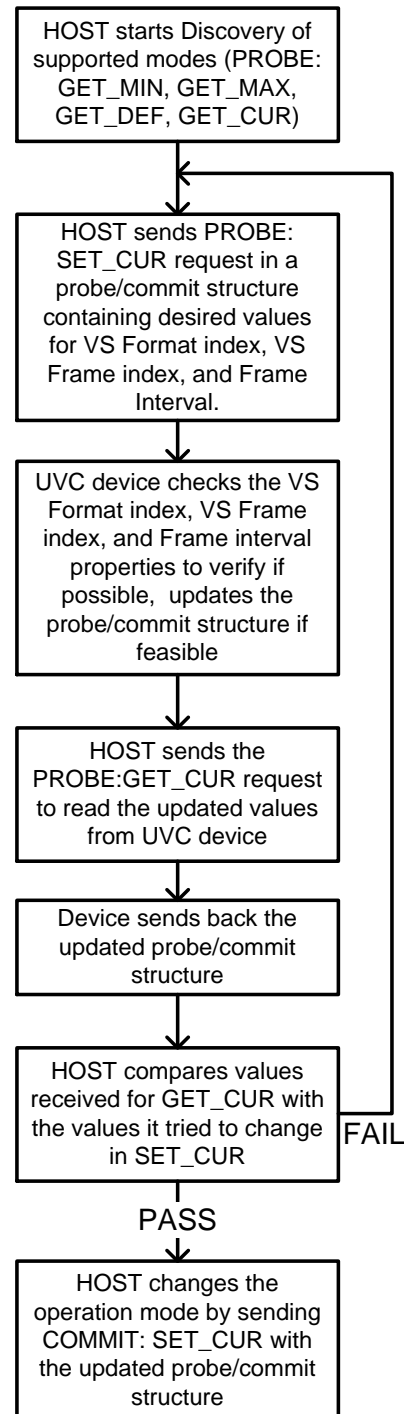
通过同步控制传输可以支持所有标准 UVC 视频控制请求。如果您要执行异步控制传输，但在执行过程中遇到问题，请创建[技术支持案例](#)。

图 5. 主机应用更改了摄像机的设置



进行输送前，主机应用将发出一组探针请求，用以查找可能的流模式。确定了默认流模式后，UVC 驱动程序将发出 COMMIT 请求。图 6 显示了该过程。这时，UVC 驱动程序可以从 UVC 器件输送视频。

图 6. 主机与摄像机之间的预输送对话框



2.3.2.1 控制请求 — 亮度、PTZ 控制以及扩展的单元控制

亮度和 PTZ 控制都在相关的项目中执行。通过定义 `uvc.h` 文件中的“UVC_PTZ_SUPPORT”，可以打开 PTZ。这些能力可能受图像传感器的支持，也可能不受它的支持。若不受支持，则必须设计特定硬件，以实现它们。在任何情况下，这些性能在所控制的 USB 侧的固件实现仍然相同。然而，图像传感器的实现会有区别。因此，只提供了占位符函数实现 USB 侧的控制。你必须写入代码，以实现图像传感器特定的 PTZ。

注意： 注意：本应用笔记中所描述的所有功能均在 `uvc.c` 文件中实现，除非另有提及。该文件是项目的一部分。它存储于本应用笔记附带的 `cyfxuvc_an75779` 文件夹中的源代码 zip 文件。

主机应用（通过 EP0）向处理单元发出视频控制请求，用以控制亮度。通过 `CyFxUVCAppInUSBSetupCB` 函数，可以处理所有设置请求。该函数可检测到主机是否已经发出了 UVC 特定的请求（控制或输送），然后设置一个事件标志：

“CY_FX_UVC_VIDEO_CONTROL_REQUEST_EVENT”或“CY_FX_UVC_VIDEO_STREAM_REQUEST_EVENT”。这时，`UVCAppEP0Thread_Entry` 函数（EP0 应用线程）将处理该标志。

亮度控制请求将触发视频控制请求的事件标志。正在等待触发这些标志的 EP0 应用线程将对视频控制请求进行解码，并调用相应的函数。EP0 应用线程调用 `UVCHandleProcessingUnitRqts` 函数用以处理亮度请求。

更改 `UVCHandleProcessingUnitRqts` 函数，以实现处理与单元相关的控制（亮度、对比度、色调等）。更改 `UVCHandleCameraTerminalRqts` 函数，以实现摄像机终端控制。更改 `UVCHandleExtensionUnitRqts` 函数，以实现所有供应商制定的请求。要支持这些控制，您必须设置 USB 描述符中的相应位。UVC 规范包括摄像机终端、处理单元和扩展单元 USB 描述符等详细信息。

扩展单元控制示例（获取或设置器件固件版本）在相关项目中实现。通过定义 `uvc.h` 文件中的“UVC_EXTENSION_UNIT”函数可以打开控制。请参考 [KBA219280](#) 以了解更多有关如何在固件中设计 UVC 扩展单元的详细信息。获取或设置固件版本允许您获取或设置应用笔记的固件版本。通过使用 `uvc_extension_app_x86.exe`（针对 32 位 Window）或 `uvc_extension_app_x64`（针对 64 位 Window）可以实现该操作。可以在附带项目的 `readme.txt` 文件中找到运行主机应用的指南。主机应用基于 Microsoft 媒体基础类和 DirectShow API。要了解更多有关主机应用的信息，请参阅[论坛讨论](#)。

2.3.2.2 串流请求 — 探针与提交控制

通过 `UVCHandleVideoStreamingRqts` 可以处理流相关的请求。当 UVC 驱动程序需要从 UVC 器件输送视频时，首先要进行协商。在该阶段，驱动程序将发出一个 PROBE（探针）请求，比如：GET_MIN、GET_MAX、GET_RES 和 GET_DEF。响应时，FX3 固件会返回一个 PROBE 结构。该结构包含 USB 描述符的索引，包括视频格式和视频帧、帧速率、帧的最大尺寸以及负载大小（即 UVC 驱动程序在每次转换中可获取的字节数量）等。

对于 USB 2.0 或 USB 3.0 的连接，“CY_FX_UVC_PROBE_CTRL”的 switch 语句将处理串流的协商阶段（在不同的模式下所支持的视频属性也不同）。请注意，因为在 USB 2.0 或 USB 3.0 连接中都支持同一个流模式，所有 GET_MIN、GET_MAX、GET_DEF 和 GET_CUR 的报告值均相同。如果需支持多个流模式，则这些值将不一样。

“CY_FX_UVC_COMMIT_CTRL”的 switch 语句处理流阶段的起始。COMMIT 控制的 SET_CUR 请求表示主机将接着开始输送视频。因此，COMMIT 控制的 SET_CUR 将设置“CY_FX_UVC_STREAM_EVENT”事件，表示主应用线程 `UVCAppThread_Entry` 将启动 GPIF II 状态机，从而进行输送视频。

2.3.3 视频数据格式：YUY2

UVC 规范只支持一个视频数据的颜色格式子集。因此，您需要选择一个符合 UVC 规范，并能够以颜色格式输送图像的图像传感器。本应用笔记包含了一个被称为 YUY2 的非压缩颜色格式。大部分（不是所有）图像传感器均支持这种格式。YUY2 颜色格式是 YUV 颜色格式中 4:2:2 的采样版本。亮度值 Y 是为所有像素采样的，而色度值 U 和 V 是仅针对偶像素进行采样的。这样会创建“宏像素”，每个宏像素将描述两个使用总共 4 个字节大小的两个图像像素。请注意，所有其他字节都是 Y 值，而 U 和 V 值仅用于表示偶像素。

Y0、U0、Y1、V0 （头两个像素）

Y2、U2、Y3、V2 （接下两个像素）

Y4、U4、Y5、V4 （接下两个像素）

请参考[维基百科](#)，了解有关颜色格式的详细信息。

注意：本应用笔记所附带的项目不支持 RGB 格式。有关 RGB 格式的支持，请参考 [FX3 SDK](#) 中的“cycx3_rgb16_as0260”和“cycx3_rgb24_as0260”示例项目（路径：<FX3 SDK installation path\EZ-USB FX3 SDK\1.3\firmware\cx3_examples>）。

尽管 UVC 规格不支持单色图像，但通过将单色图像数据作为 Y 值发送，并将所有 U 和 V 值都设置为 0x80，仍可以将一个 8 位的单色图像作为 YUY2 格式显示。

2.3.4 UVC 视频数据标头

UVC 类别非压缩视频负载需要一个 12 字节的标头。该标头描述了所传输图像数据的属性。例如，它包括一个“新帧”位，图像传感器控制器 (FX3) 可以为每个帧切换该位。FX3 代码还可以设置标头中的错误位，以表示在传输当前帧的过程中发生的错误。每个 USB 传输操作都需要该 UVC 数据标头。请参考 UVC 规范以了解详细信息。[表 4](#) 显示的是 UVC 视频标头数据的格式。

表 4. UVC 视频数据标头格式

字节偏移	字段名称	说明
0	HLF	标头长度字段指定了标头的长度，单位为字节
1	BFH	位字段标头表示图像数据的类型、视频流的状态以及其他字段的当前情况
2-5	PTS	呈现时间戳表示源时钟时间，单位为本地器件时钟周期
6-11	SCR	参考源时钟表示系统时间时钟和 USB 帧开始 (SOF) 标志计数器

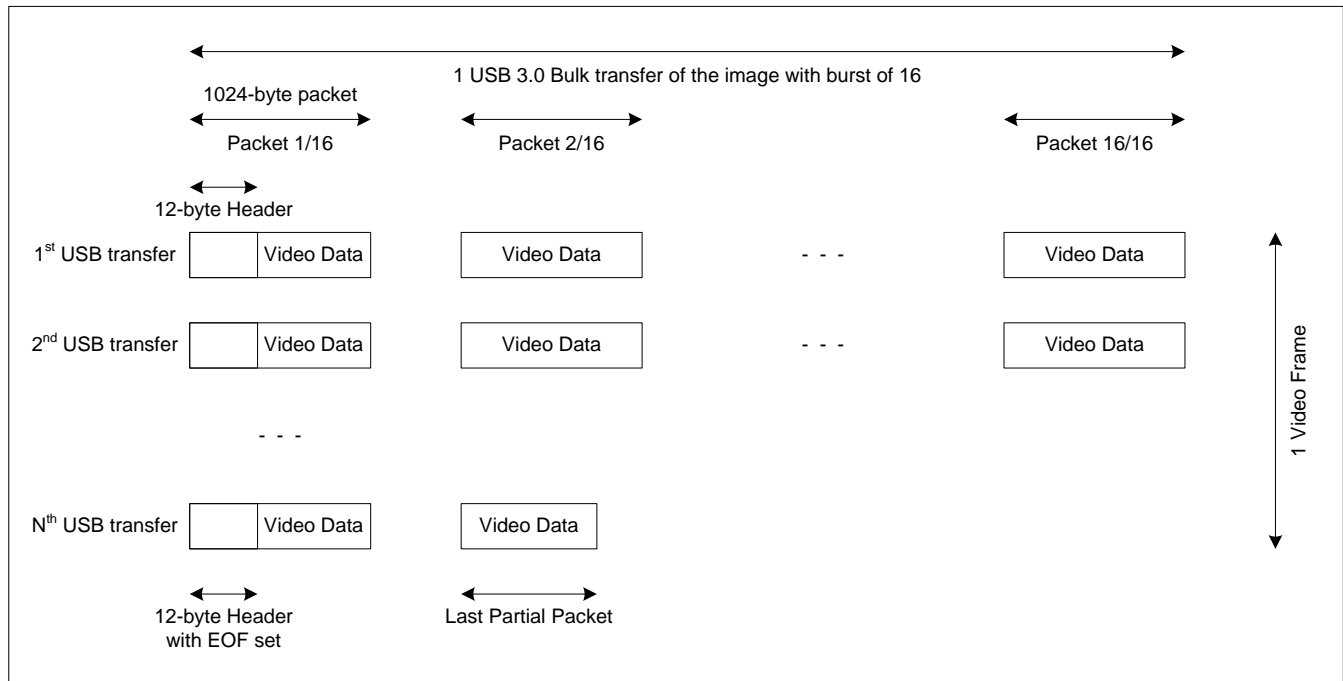
HLF 的值始终为 12。PTS 和 SCR 字段都是可选的。在固件示例中，这些字段的值都被置为零。位字段标头 (BFH) 将保持更改帧结束的值。[表 5](#) 显示的是 BFH 格式 (BFH 是视频标头数据的一部分)。

表 5. 位字段标头 (BFH) 格式

位偏移	字段名称	说明
0	FID	帧标识符位在每个图像帧的起始边界上进行切换，并在图像帧的其余部分保持不变
1	EOF	帧结束位表示视频的结束，仅在图像帧所执行的最后 USB 传输时设置该位。
2	PTS	呈现时间戳位表示 UVC 视频数据标头中 PTS 字段的当前情况 (1 表示存在)
3	SCR	源时钟参考位表示 UVC 视频数据标头中 SCR 字段的当前情况 (1 表示存在)
4	RES	保留，将其设置为 0
5	STI	静态图像位表示视频取样是否属于静态图像
6	ERR	错误位表示器件在串流时发生的错误
7	EOH	标头结束位 (如果被设置) 表示 BFH 字段的结束

[图 7](#) 显示的是如何将标头添加到本应用的视频数据中。进行每个 USB 批量传输时都要添加 12 字节的标头。这时，每个 USB 传输共有 16 个批量数据包。USB 3.0 的批量数据包大小为 1024 个字节。

图 7. UVC 视频数据传输



3 GPIF II 图像传感器接口

FX3 的 GPIF II 模块是一个很灵活的状态机，通过自定义该模块可以将 FX3 引脚连接至外部硬件（如图像传感器）。要设计状态机，需要了解该接口的要求以及 FX3 的 DMA 功能。

3.1 图像传感器接口

图 2 显示的是一个典型的图像传感器接口。图像传感器一般要求 FX3 控制器的复位信号。通过使用 FX3 通用输入/输出（GPIO）可以处理该操作。

图像传感器通常使用一个 I²C 接口，允许控制器对图像传感器的参数进行读写操作。图像传感器还会使用串行外设接口（SPI）或通用异步接收器/发送器（UART）接口实现同样操作。FX3 的 I²C、SPI 和 UART 模块都具有该功能。本应用使用了 I²C 来配置图像传感器。

为了传输图像，图像传感器要提供下面各信号：

1. FV：帧有效（表示帧的开始和结束）
2. LV：行有效（表示行的开始和结束）
3. PCLK：像素时钟（即同步接口的时钟）
4. Data（数据）：图像数据的八个数据线

图 8 显示的是 FV、LV、PCLK 和 Data 信号的时序图。图像传感器确认 FV 信号表示帧的开始。然后，逐行传输图像数据。在每行的传输操作中会激活 LV 信号，因为图像传感器会驱动 YUY2 格式的 8 位像素（在该格式中，每两个像素使用了 4 个字节）。在每个 PCLK 的上升沿上，字节数据将被提供给 GPIF II 单元。

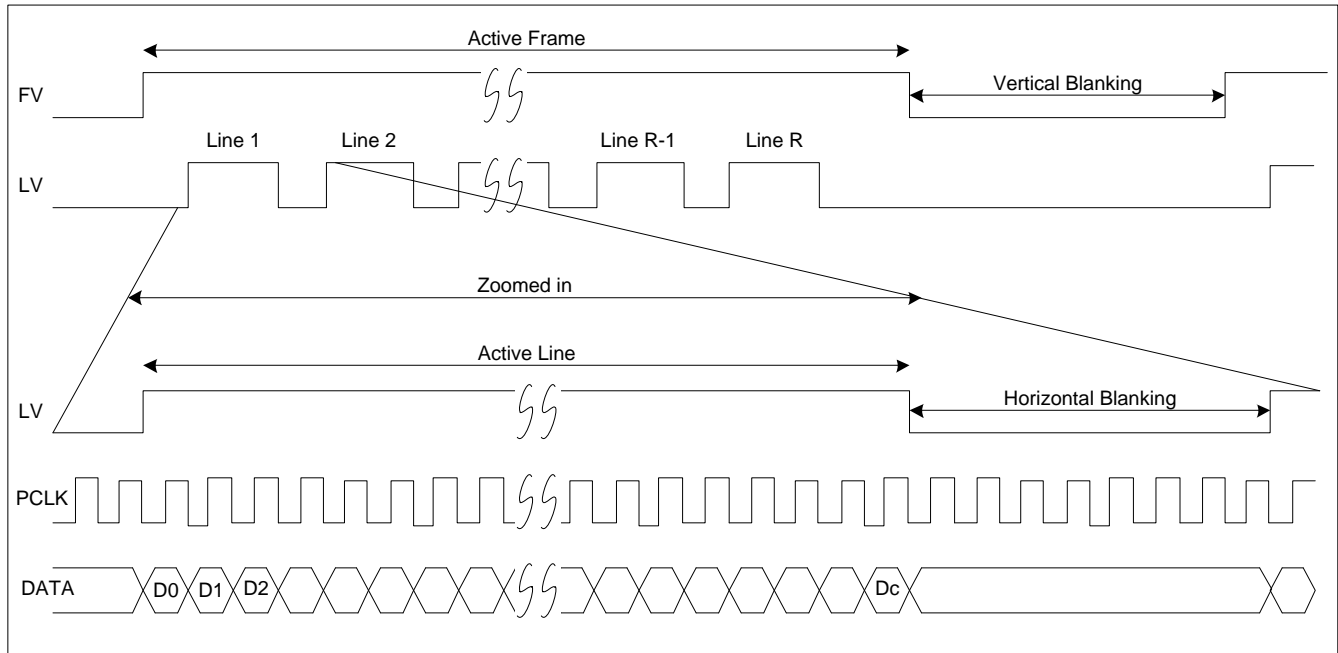
可将 FX3 GPIF II 总线配置为 8、16 或 32 位的总线。本应用使用了图像传感器的 8 位总线。如果图像传感器提供的总线是非字节对齐的（比如 12 位总线），请使用下一个更大的大小来填充或清除未使用的位。

3.1.1 GPIF II 接口的要求

根据时序图 (图 8)，GPIF II 的状态机有下面各项要求：

- 仅在 LV 及 FV 信号激活时，GPIF II 模块才会从数据引脚传输数据。
- 图像传感器不提供流量控制。因此，该接口必须传输一整行视频而不能间断。在该设计中，每一行包括 1280 个像素，每个像素需要两个字节，所以每行要传输 2560 个字节。
- 每一帧结束时都要通知 CPU，使其更新相应标头位。

图 8. 图像传感器接口时序图



3.2 图像传感器接口的引脚映射情况

表 66 显示的是 GPIF II 到图像传感器的引脚映射详情。

表 66. 并行图像传感器接口的引脚映射情况

EZ-USB FX3 引脚	具有 8 位数据总线的同步并行图像传感器接口
GPIO[28]	LV
GPIO[29]	FV
GPIO[0:7]	DQ[0:7]
GPIO[16]	PCLK
I ² C_GPIO[58]	I ² C SCL
I ² C_GPIO[59]	I ² C SDA

如果图像传感器使用的是 UART 或 SPI 接口，并使用了 16 位数据总线，那么请采用表 7 中描述的引脚映射。

表 7. 图像传感器的其他引脚映射情况

EZ-USB FX3 引脚	图像传感器接口 (额外引脚)
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
GPIO[54]	GPIO/SPI_SSN/UART_CTS
GPIO[55]	GPIO/SPI_MISO/UART_TX
GPIO[56]	GPIO/SPI_MOSI/UART_RX

注意：要了解有关完整的 EZ-USB FX3 引脚映射的信息，请参见“EZ-USB FX3 超速 USB 控制器”数据手册。

3.3 交替 DMA 缓冲区

要想了解 FX3 的数据输出和输入，需要明白下面各术语：

- 套接字
- DMA 描述符
- DMA 缓冲区
- GPIF 线程

套接字是外设硬件模块和 FX3 RAM 之间的连接点。FX3 上的每个外设硬件模块（如 USB、GPIF、UART 和 SPI）具有各自固定的套接字数量。传输过外设的独立数据流数量等于该外设上的套接字数量。套接字的执行包括一组寄存器，用于指向有效的 DMA 描述符，并使能或置位与该套接字相关的中断。

DMA 描述符是一组位于 FX3 RAM 中的寄存器。它保存了 DMA 缓冲区的地址和大小数据，以及指向下一个 DMA 描述符的指针。这些指针构建成了 DMA 描述符链。

DMA 缓冲区是 RAM 的一部分，用于存储通过 FX3 器件传输的中间数据。通过 FX3 固件，可将部分 RAM 空间分配给 DMA 缓冲区。这些缓冲区的地址被存储在 DMA 描述符中。

GPIF 线程是 GPIF II 模块内的专用数据路径，用来将外部数据引脚同套接字连接起来。

套接字可通过各个事件直接互相发出信号，或者通过中断向 FX3 CPU 发出信号。这些操作是由固件配置的。例如，将数据流从 GPIF II 模块传输给 USB 模块。GPIF 套接字可以通知 USB 套接字它已经向 DMA 缓冲区写满了数据，或者 USB 套接字可以通知 GPIF 套接字它已经清空了该 DMA 缓冲区。该操作被称为自动 DMA 通道。当 FX3 CPU 不用修改数据流中的任何数据时，通常会使用自动 DMA 通道。

或者，GPIF 套接字可以向 FX3 CPU 发送一个中断，来通知 GPIF 套接字已经写满了 DMA 缓冲区。FX3 CPU 可将该信息传递给 USB 套接字。USB 套接字可向 FX3 CPU 发送一个中断，来通知 USB 套接字已经清空了 DMA 缓冲区。此时，FX3 CPU 可以将此信息反馈给 GPIF 套接字。该操作被称为手动 DMA 通道。如果 FX3 CPU 需要添加、删除或修改数据流中的数据，通常需要执行该手动 DMA 通道操作。本应用笔记中的固件示例使用了手动 DMA 通道操作，因为该固件要求添加 UVC 视频数据标头。

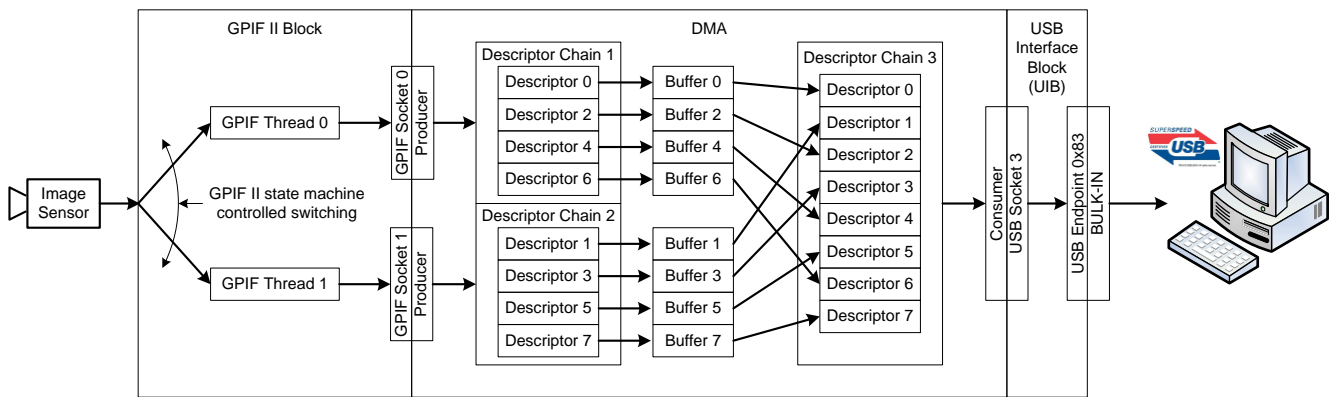
生产套接字指的是用于将数据写入 DMA 缓冲区内的套接字。消耗套接字指的是读取 DMA 缓冲区内数据的套接字。套接字使用存储于 DMA 描述符上的 DMA 缓冲区地址、DMA 缓冲区大小和 DMA 描述符链的值来管理数据（第 4 节）。套接字写满或清除 DMA 缓冲区后，需要一个有限的时间（最多几微秒）从一个 DMA 描述符切换到另一个描述符。切换过程中，套接字不能传输数据。对于没有流量控制的接口，该延迟是一个问题。图像传感器接口便属于这类情况。

通过使用多个 GPIF 线程，可以在 GPIF II 模块中处理这问题。GPIF II 模块实现四个 GPIF 线程。每次只有一个 GPIF 线程能够传输数据。GPIF II 状态机必须选择一个有效的 GPIF 线程来传输数据。

GPIF 线程选择机制同 MUX 一样。GPIF II 状态机使用内部控制信号或外部信号选择有效的 GPIF 线程。在这示例中，内部控制信号控制在各 GPIF 线程之间进行切换。切换有效的 GPIF 线程时会切换用于数据传输的有效套接字，从而修改用于数据传输的 DMA 缓冲区。GPIF 线程的切换没有延迟。GPIF II 状态机在 DMA 缓冲区边界上进行该切换，这样可以屏蔽 GPIF 套接字切换到新的 DMA 描述符时所要求的延迟。这时，当 DMA 缓冲区被写满时，GPIF II 模块可以获取传感器中的数据而没有发生数据丢失。

图 9 显示的是本应用中使用的各个套接字、DMA 描述符和 DMA 缓冲区连接，以及数据流。使用两个 GPIF 线程写满备用的 DMA 缓冲区。这些 GPIF 线程使用了单独的 GPIF 套接字（作为生产套接字使用）和 DMA 描述符链（描述符链 1 和描述符链 2）。USB 套接字（作为消耗套接字使用）使用了另一个 DMA 描述符链（描述符链 3）按正确顺序读取数据。

图 9. FX3 数据传输架构



3.4 设计策略

在编译 GPIF II 状态机的详细信息前，建议考虑基本的传输策略。

图 10. 在视频线内的数据传输

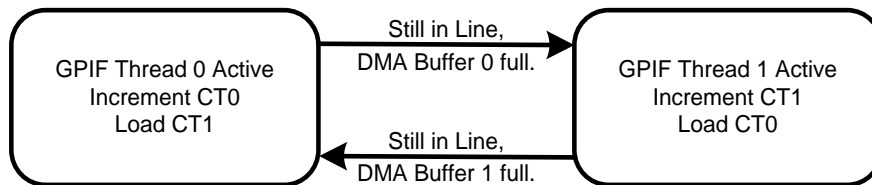
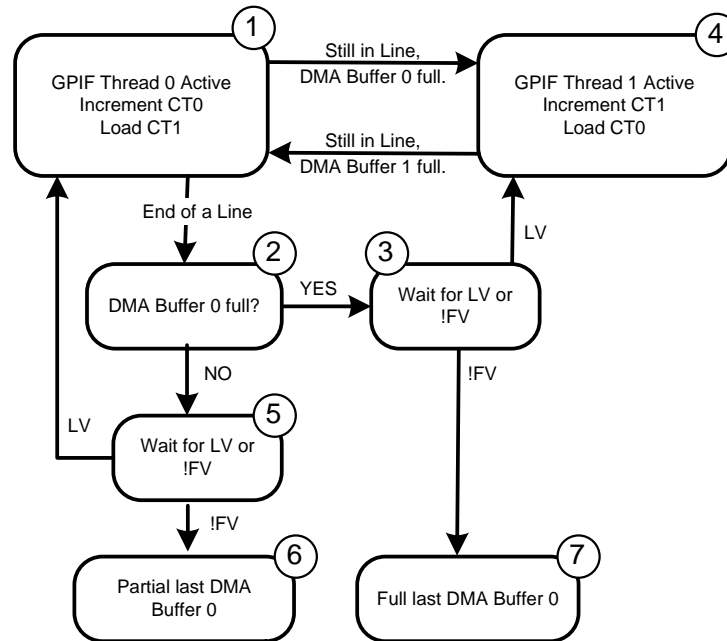


图 10 显示的是在活动的水平线期间，填满 DMA 缓冲区时基本 DMA 缓冲区的交替操作。GPIF II 状态机有三个独立的计数器。在该示例中，GPIF II 单元使用两个计数器（CT0 和 CT1）来计算写入到 DMA 缓冲区的字节数量。当计数达到 DMA 缓冲区的限制时，将采取“DMA 缓冲区已满”分支，并且 GPIF II 将切换 GPIF 线程。由于各字节是在一个 GPIF 线程中进行传输的，因此在下一个 GPIF 线程切换出现后，需要初始化用于其他 GPIF 线程的备用计数器。

当图像传感器取消确认 LV 信号时，则表示它已到达视频线的结尾。这时，状态机有几个选项，如图 11 所示，其中 LV 表示有效线信号和 FV 表示有效帧信号。各选项结果取决于 DMA 缓冲区是否填满和该帧是否完成。

图 11. 视频线结束的数据传输决策



注意： 由“线结尾”开始的映像决策树也是从状态 4 发出的；为了提高清晰度，在该框图中没有显示。

在线传输结束时 (LV=0)，状态机从状态 1 转换到状态 2，然后它将检查 DMA 缓冲区字节计数器，以确定 DMA 缓冲区是否被填满。如果 DMA 缓冲区已满，则状态机将转换到状态 3。在状态 3 中，如果 FV 为低电平，则表示已经传输完了全帧，并且进入状态 7。在状态 7 中，一个表示最后完整的 DMA 缓冲区的中断请求将提醒 CPU 通过这信息，CPU 可以设置用于下一个帧的 GPIF II。

在状态 3 中，如果 FV = 1，图像传感器将继续传输某一帧，并且它将重新确认 LV=1，以指出下个视频线。当 LV = 1 时，状态机从状态 3 转换到状态 4，并执行 GPIF 线程切换，该切换与从状态 1 转换到状态 4 的切换相同。因此，路径 1-4 和 1-2-3-4 都进行了一个 GPIF 线程切换。这些路径的区别在于第二个路径需要使用额外的周期，这是因为它线传输结束时有一个暂停。

如果 DMA 缓冲区未充满，则状态机将从状态 2 转换到状态 5。状态 5 与状态 3 相似，它将等待该帧结束或 LV 重新激活 (开始下一个视频线)。如果 LV = 1，它将在状态 1 中继续填写 DMA 缓冲区 0。如果 FV = 0，则表示图像传感器已完成传送一个视频帧，而且 DMA 缓冲区 0 只被部份填充。在状态 6 中，状态机向 CPU 发送另一个中断，允许它通过 USB 发送短 DMA 缓冲区。

3.5 GPIF II 状态机

FX3 GPIF II 是一个可多达 256 种状态的可编程状态机。每个状态可以执行下列操作：

- 驱动多个控制线
- 发送或接收数据和/或地址
- 向内部 CPU 发送中断

状态的转换取决于内部或外部信号 (如 DMA 就绪信号) 以及图像传感器的有效帧/线信号。

要开始设计 GPIF II 状态机，可在图像传感器波形中选择一点为状态机的起始位置。由上升 FV 跃变指出的帧起始位置是一个个逻辑的起始点。GPIF II 通过先等待 FV = 0 (第一个状态)，然后再等待 FV = 1 (第二个状态) 来检测该沿。第二个状态也将初始一个传输计数器，以回应一个填满视频数据的 DMA 缓冲区。状态机将测试该计数器的值，如果达到极限值，它将切换 GPIF 线程 (DMA 缓冲区)。当 DMA 缓冲区被填满时，将达到计数器的极限值。

状态机使用两个 GPIF II 内部计数器来计算 DMA 缓冲区字节：GPIF II 地址计数器 ADDR 和数据计数器 DATA。每当 GPIF II 状态机切换 GPIF 线程时，它将为其他 GPIF 线程启动合适的计数器。由于进行加载计数器的限制值需要一个时钟周期，因此该值比终端计数值小 1 个单位。

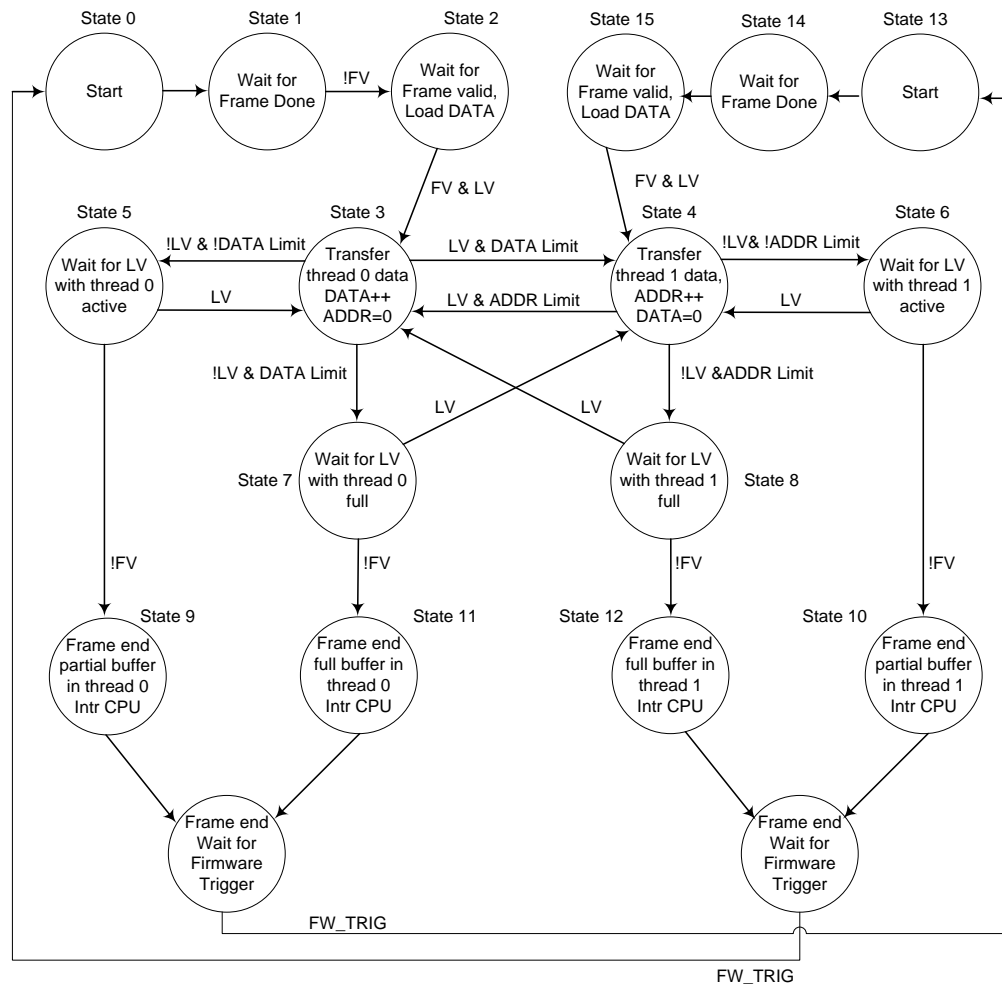
执行每一个时钟周期后，传输计数器的值将递增 1。因此，根据接口的数据总线宽度，计数器的限定值将不一样。在该示例中，数据总线的宽度为 8 位，DMA 缓冲区大小为 16,368 字节（如第 5.6 节中所述），因此编程限制值为 16,367。一般情况下，DMA 缓冲区的计数限制值为：

$$count = \left(\frac{producer_buffer_size(L)}{data_bus_width} \right) - 1$$

所以对于 16 位的接口，DMA 缓冲区大小为 8184 个 16 位字，编程限制值为 8183。对于 32 位的接口，DMA 缓冲区大小为 4092 个 32 位字，编程限制值为 4091。

图 12 显示的是 GPIF II 状态机的详细信息。两个 DMA 缓冲区字节计数器分别为 GPIF II 的 DATA 和 ADDR 计数器。这些计数器同图 11 中显示的 CT0 和 CT1 相对应。

图 12. GPIF II 状态机的框图

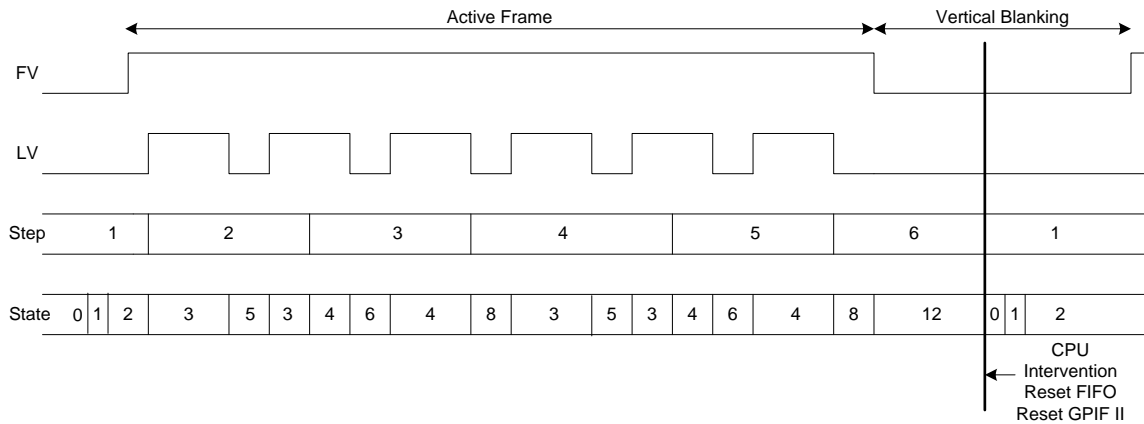


在每一帧的传输结束时，CPU 收到四个可能中断请求中的一个，表示 DMA 缓冲区的编号和已满状态（状态 9-12）。可以使用这些请求执行回调函数，以允许 CPU 处理任务，具体如下：

1. 如果帧结束时未有的 DMA 缓冲区（状态 9 和 10），则确认最后 DMA 缓冲区，以用于 USB 传输。如果在帧结束时 DMA 缓冲区已满，GPIF II 将自动确认它，以用于 USB 传输（状态 11 和 12）。
2. 等待消耗套接字（USB）传送最后的 DMA 缓冲区数据；然后，将通道和 GPIF II 状态机复位到状态 0，以预备下一个帧。
3. 处理所有特殊应用的任务，以指出帧的前进。UVC 需要通过切换 12 字节标头中的位来指示帧改变事件。

图 13 将图像传感器波形连接到 GPIF II 状态，显示一小部分水平线。“Step”线处理 DMA 系统，具体情况在第 4 节和图 45 中进行介绍。

图 13. 图像传感器接口、数据路径执行以及状态机的关联



3.6 使用 GPIF II Designer 实现图像传感器接口

本节介绍的是如何使用 GPIF II Designer 来设计图像传感器接口。有关参考材料，请访问附带源中压缩文件内的“fx3_uvc.cydsn”目录路径，以查找整个项目。

设计过程包括下面三个步骤：

1. 使用 GPIF II Designer 创建项目
2. 选择接口定义
3. 在设计图上绘制状态机

3.6.1 创建项目

启动 GPIF II Designer。GUI 如图 14 显示。按照下图所示的步骤指示进行操作。每个图包含了多个子步骤。有关每个步骤的预先信息，请参考 [GPIF II Designer 用户指南](#)。

图 14. 启动 GPIF II Designer

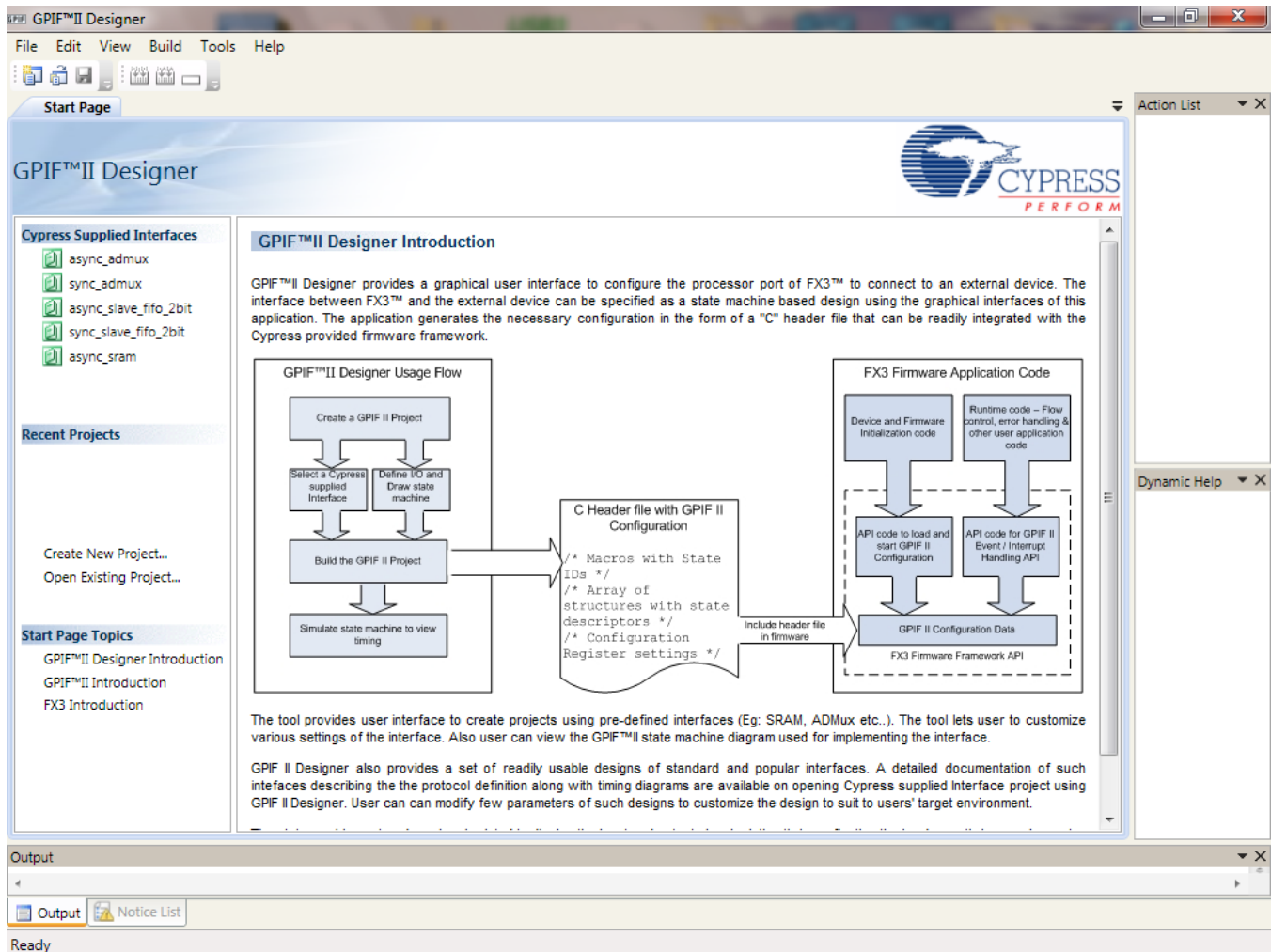


图 15. 打开 File 菜单并选择 New Project 项

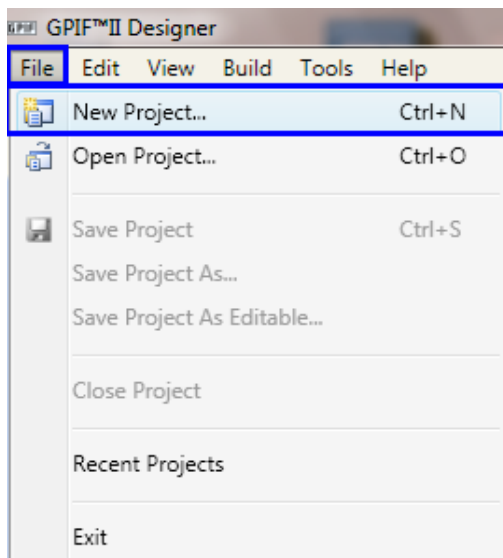
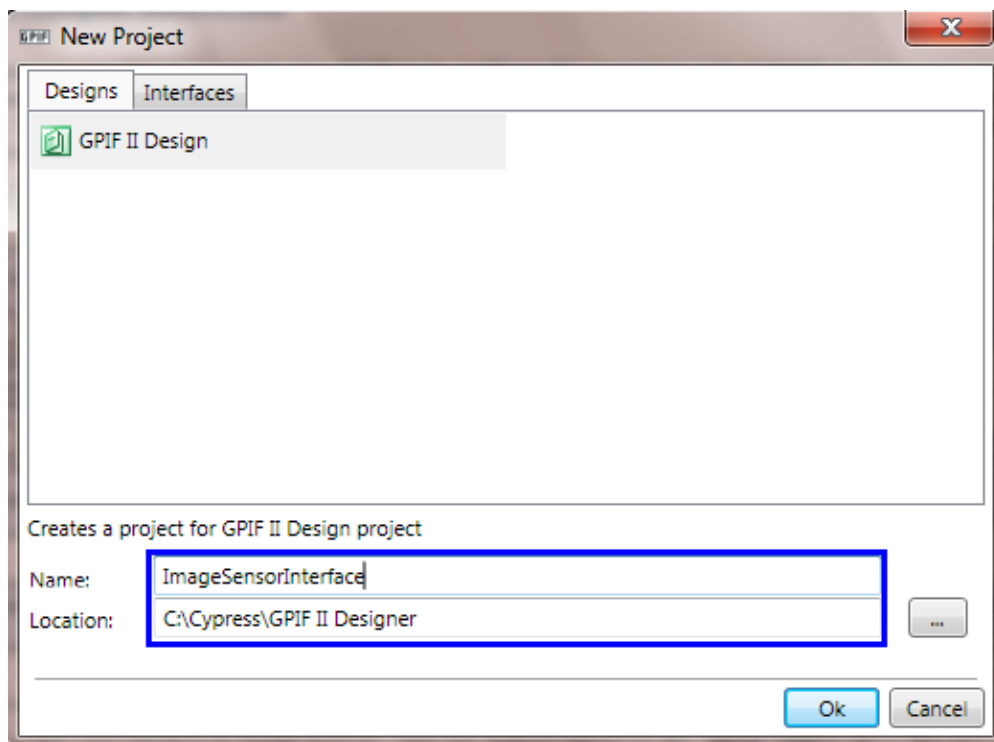


图 16. 输入项目名称和位置



现在完成了创建项目操作，打开 GPIF II Designer 后可以访问接口定义和状态机选项卡。在接口定义选项卡中，FX3 位于左侧，图像传感器（标签为“应用处理器”）位于右侧。接下来进行设置这两者之间的接口。

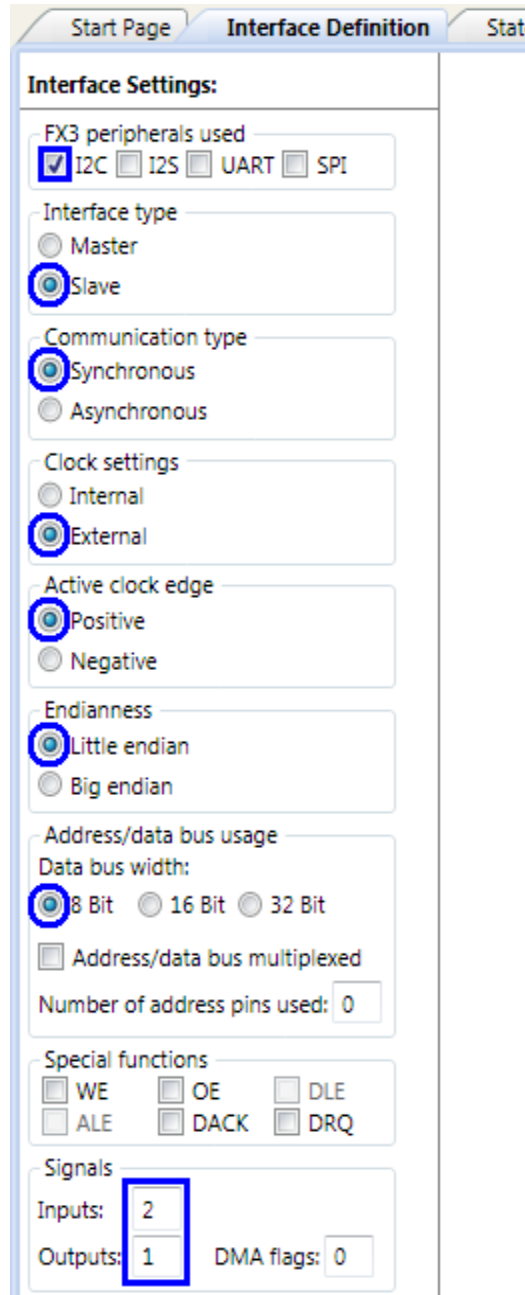
3.6.2 定义接口

在该项目中，与 FX3 器件相连的图像传感器具有 8 位数据总线。它将 GPIO 28 用于 LV 信号，GPIO 29 用于 FV 信号，另外 GPIO 22 为低电平有效输入的传感器复位（表 66）。该图像传感器还可通过连接到 FX3 的 I²C 来访问图像传感器寄存器，例如，为 720p 模式配置传感器。“接口设置”列显示各需要的选项。

另外，在下一个阶段中，可以使用指示输入信号来创建转换公式。图 17 显示的是可选的接口设置。

1. 在“Signals”（信号）项中，为 LV 和 FV 选择两个输入。
2. 在“Signals”（信号）项中，为 nSensor_Reset 选择一个输出。
3. 在“FX3 peripherals used”（使用的 FX3 外设）项中，选择 I²C。这样可以激活 FX3 I²C 主设备。
4. 在“Interface type”（接口类型）项中，选择“Slave”（从设备）。由于图像传感器提供时钟，并驱动数据总线，因此 GPIF II 将作为从设备使用。
5. 在“Communication type”（通信类型）项中，选择“Synchronous”（同步）。这样反映了在图像传感器中存在的同步时钟。
6. 在“Clock settings”（时钟设置）项中，选择“External”（外部）。图像传感器将为 GPIF II 提供它的 PCLK 信号。
7. 在“Active clock edge”（活动时钟沿）项中，选择“Positive”（上升沿）。图像传感器在它的上升沿上启用数据转换。
8. 在“Endianness”（字节顺序）项中，选择“Little endian”（低位优先）。字节顺序指的是数据总线中比一个字节的宽度更大的字节顺序。对于 8 位的接口，该设置不受影响。
9. 在“Address/data bus usage”（地址/数据总线的使用情况）项中，选择“8 bit”（8 位）。图像传感器提供了一个 8 位的数据总线。

图 17. 接口设置选项



Interface Settings:

FX3 peripherals used
☒ I2C ☐ I2S ☐ UART ☐ SPI

Interface type
☐ Master
☒ Slave

Communication type
☒ Synchronous
☐ Asynchronous

Clock settings
☐ Internal
☒ External

Active clock edge
☒ Positive
☐ Negative

Endianness
☒ Little endian
☐ Big endian

Address/data bus usage
 Data bus width:
☒ 8 Bit ☐ 16 Bit ☐ 32 Bit
☐ Address/data bus multiplexed
 Number of address pins used: 0

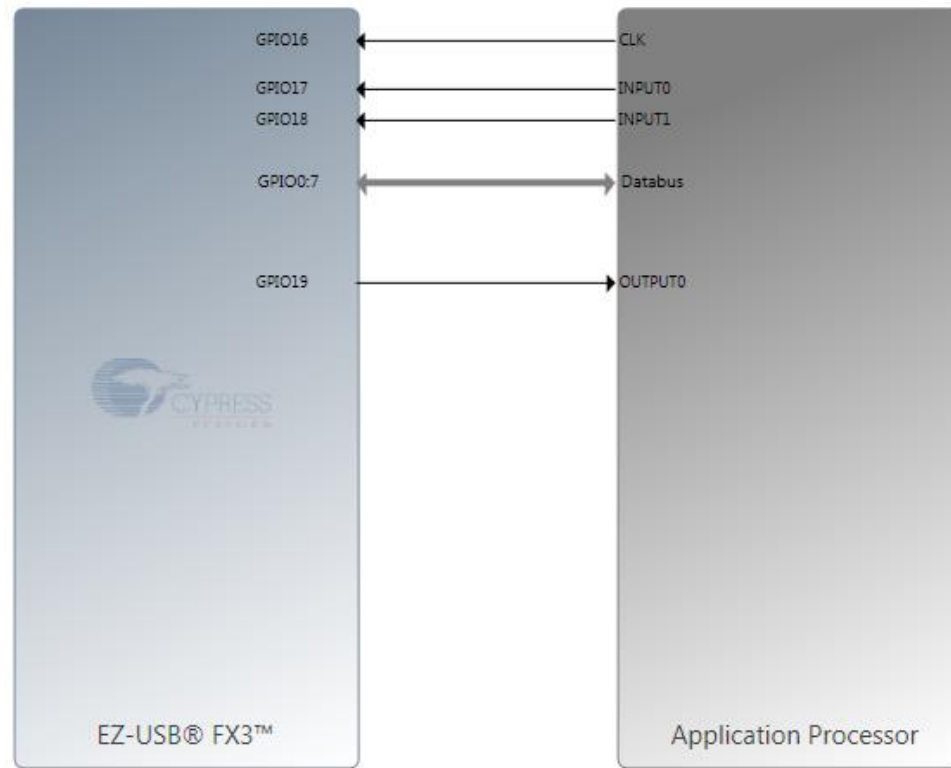
Special functions
☐ WE ☐ OE ☐ DLE
☐ ALE ☐ DACK ☐ DRQ

Signals
 Inputs: 2
 Outputs: 1 DMA flags: 0

“I/O Matrix configuration”（I/O 矩阵配置）现在的情况如图 18 所示。接下来需要修改输入和输出信号的属性，这些属性包括信号名称、引脚映像（例如，哪个 GPIO 作为输入或输出使用）、信号极性以及输出信号的初始值。

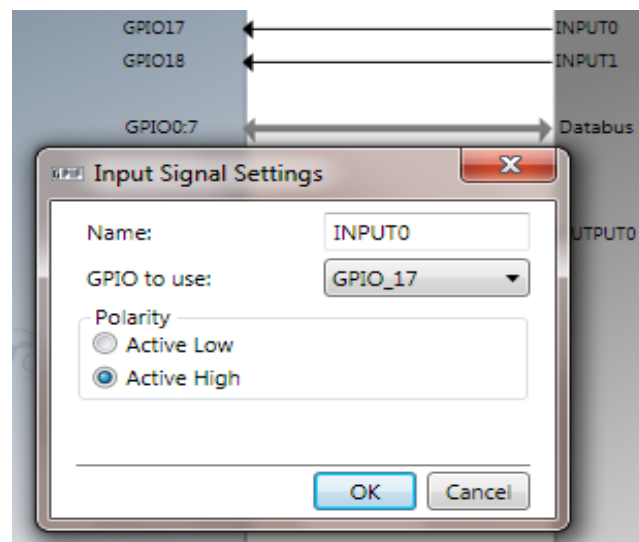
图 18. 当前的框图

I/O Matrix Configuration



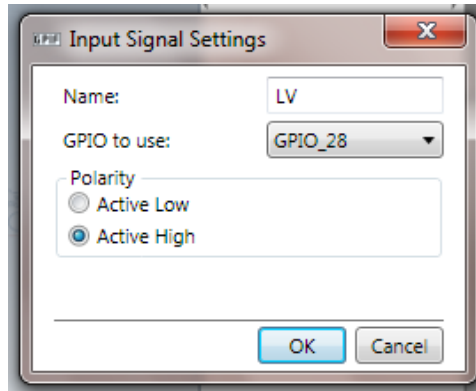
在应用处理器区中双击 INPUT0 标签可以打开输入信号的属性，如图 19 所示。

图 19. INPUT0 的默认属性



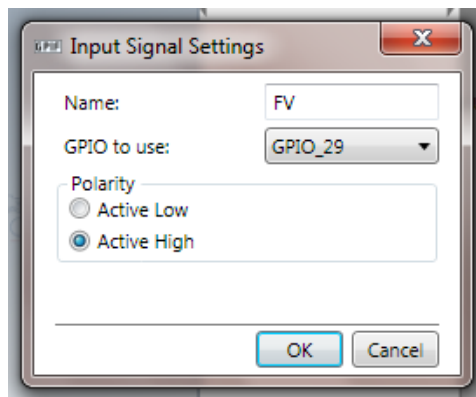
将信号名称修改为“LV”，并将“GPIO to use: ”（被使用的 GPIO）设置为 GPIO_28（表 66）。将极性保持为“Active High”（高电平有效）。现在，这些属性的情况如图 20 所示。点击“OK”。

图 20. 已编辑的 INPUT0 属性



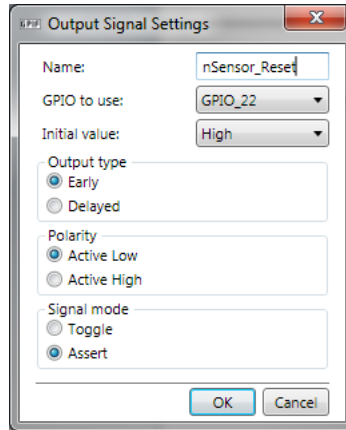
接下来，双击 INPUT1 标签，将信号名称改为 FV，并将 GPIO 设置为 GPIO_29，并保持极性为高电平有效（图 21）。

图 21. 已编辑的 INPUT1 属性



双击 OUTPUT0 标签并根据图 22 修改各设置内容。

图 22. 已编辑的 OUTPUT0 属性



这样便完成了接口设置。设置各属性（如信号名称）时，GPIF II Designer 的所有其他内容将更新，以反映该更改。例如，该框图现在包含了信号名称，而不是通用的输入和输出名称。另外，当您使用状态机设计工具时，可用的信号选项（如 LV 和 FV 信号）将自动显示为下拉列表中的选项。

3.6.3 绘制状态机

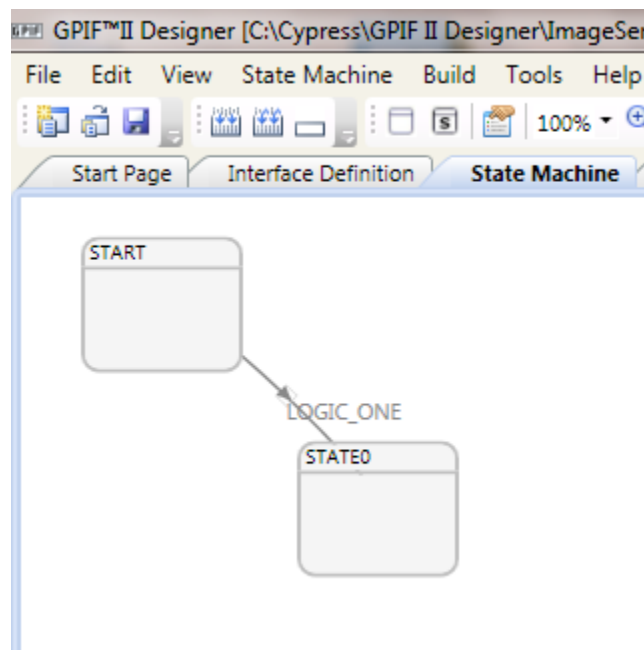
通过点击 **State Machine**（状态机）选项卡可以打开状态机设计图。状态机设计包括下面三个步骤：

1. 创建状态
2. 在各状态中添加操作
3. 使用转换操作所要求的条件创建各状态间的转换

3.6.4 绘制 GPIF II 状态机

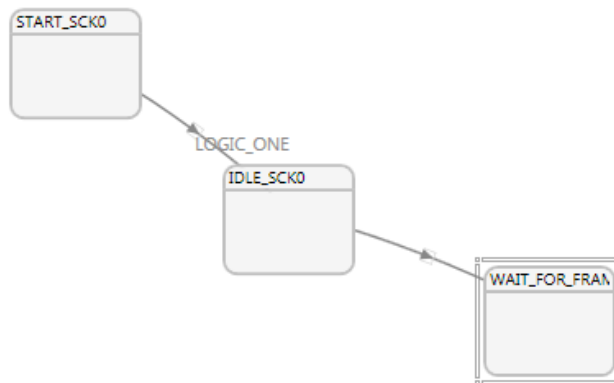
点击 **State Machine**（状态机）选项卡打开设计图。未编辑的设计图有两种状态：**START** 和 **STATE0**。无条件转换（**LOGIC_ONE**）将连接各状态（图 23）。

图 23. 初始状态机设计图



1. 通过双击 **START** 框并将其名称改为 **START_SCK0** 可以编辑 **START** 状态的名称。同样，将 **STATE0** 状态的名称编辑为 **IDLE_SCK0**。通过 “Repeat actions until next transition” (复操作直到发生下一个转换为止) 项可以确定在输入该状态时，仅发生一次操作还是在该状态内每个时钟上均发生。对于无操作的状态，如这状态，则选框状态是不可用的。
2. 通过右击设计图中的空白位置并选择 “Add State” (添加状态) 项，可以添加一个新的状态。双击该新状态，并将它的名称改为 **WAIT_FOR_FRAME_START_0**。
3. 创建从 **IDLE** 状态到 **WAIT_FOR_FRAME_START_0** 状态的转换将光标放置在 **IDLE** 状态框的中心位置。该光标将变为+形状，表示转换的输入。将鼠标拖放到 **WAIT_FOR_FRAME_START_0** 状态的中心位置。在状态框内显示一个小正方形，表示其中心的定位。如果在状态框中心任何位置中释放鼠标，那么不会创建转换。重新尝试。请注意，这状态切换还没有任何条件。

图 24. 步骤 1 到 3 的结果



4. 通过双击转换线可以从 **IDLE_SCK0** 到 **WAIT_FOR_FRAME_START_0** 编辑转换公式 (图 25)。请注意，**LV** 和 **FV** 显示为公式项，以对应已重新命名的框图信号。要想选择低电平 **FV**，请使用按键和信号选择来创建公式。点击 **Not** 按键，在公式输入窗口内将显示 “!” 符号。然后选择 **FV** 信号并点击 “Add” (添加) 按键 (或双击 **FV** 输入)，以创建最后的 “!FV” 公式。也可以在公式输入窗口中键入 “!FV” 直接输入该公式。这时，该转换如图 26 所示。

图 25. 双击转换线后将显示该窗口

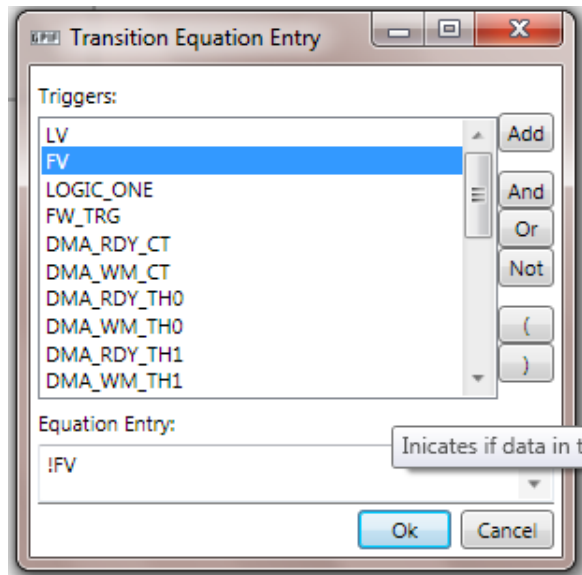
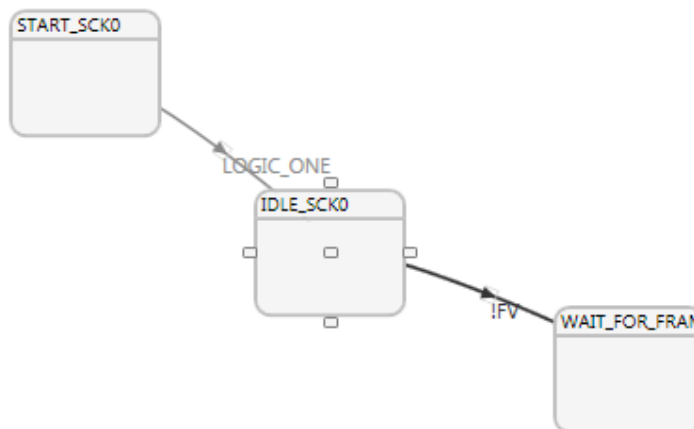
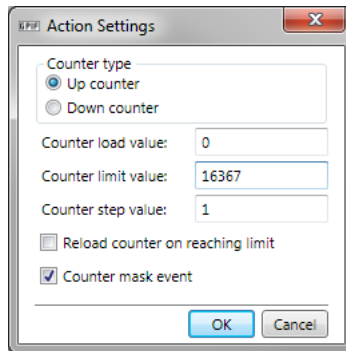


图 26. 已编辑的转换



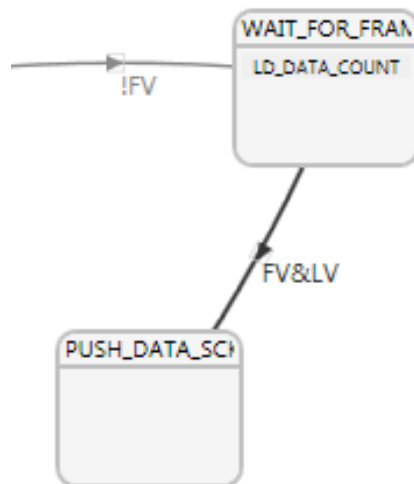
- 在 WAIT_FOR_FRAME_START_0 状态下，我们要初始化两个字节计数器： 因为计数器在其递增的状态前必须被初始化。我们重新回顾一下，本设计的 DATA 计数器使用的是套接字 0 缓冲区，而 ADDR 计数器则使用的是套接字 1 缓冲区。GPIF II Designer 右上窗口的“Action List”（操作列表）窗口内显示的是各操作选项。要想将某个操作添加到状态中，需要将它名称拖放到状态框中的操作列表内。将 LD_DATA_COUNT 和 LD_ADDR_COUNT 操作拖放到 WAIT_FOR_FRAME_START_0 状态框内。
- 要想编辑各操作属性，请双击状态框中的操作名称。然后编辑两个操作的属性，如图 27 所示。如果勾选“Counter mask event”（计数器屏蔽事件）复选框，当计数器达到它的限定值时可以禁用其中断请求。

图 27. LD_DATA/ADDR_COUNT 操作设置



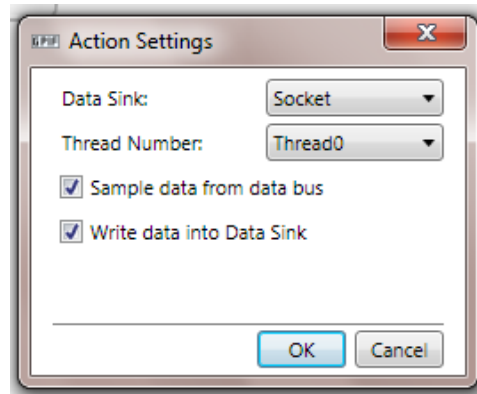
7. 类似于上一步骤，创建 START_SCK1、IDLE_SCK1 和 WAIT_FOR_FRAME_START_1 等状态。
8. 添加一个名称为 PUSH_DATA_SCK0 的新状态（将图像传感器数据推入套接字 0 内）。该状态每个时钟将一个图像传感器字节传输到 GPIF II 接口上，这个接口连接到套接字 0。
9. 创建一个从 WAIT_FOR_FRAME_START_0 state 到 PUSH_DATA_SCK0 状态的转换。
10. 编辑该转换公式输入，当确认 FV 和 LV 时通过创建 FV&LV 公式使其发生。结果状态转换如图 28 所示。

图 28. PUSH_DATA_SCK0 和其转换条件 FV&LV



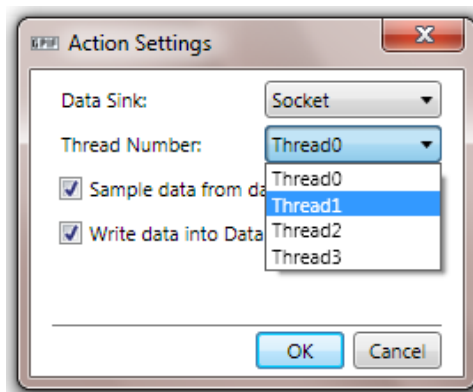
11. 将 COUNT_DATA 操作添加到 PUSH_DATA_SCK0 状态内。这样，DATA（套接字 0）计数器的值将在每个 PCLK 的上升沿上递增。
12. 将 IN_DATA 操作添加到 PUSH_DATA_SCK0 状态内。使用该操作可以从数据总线上读取数据，并将数据写入到 DMA 缓冲区内。
13. 将 LD_ADDR_COUNT 操作添加到 PUSH_DATA_SCK0 状态内，以重载 ADDR 计数器。如上所述，计数器加载是在实际递增计数器的状态中完成的。ADDR 计数器将计算传输到 SCK1 内的字节。
14. 在 PUSH_DATA_SCK0 状态中，编辑操作 IN_DATA 的属性，如图 29 所示。

图 29. PUSH_DATA_SCK00 操作设置



15. 添加名称为 PUSH_DATA_SCK1 的新状态。创建从 WAIT_FOR_FRAME_START_1 状态到 PUSH_DATA_SCK1 状态的转换，并编辑该转换公式输入，以在 FV 和 LV 都被确认时使其发生。将 COUNT_ADDR、IN_DATA 和 LD_DATA_COUNT 操作添加到该状态内。
16. 在 PUSH_DATA_SCK1 状态中，编辑 IN_DATA 操作的属性，使其能够使用 Thread1，如图 30 所示。

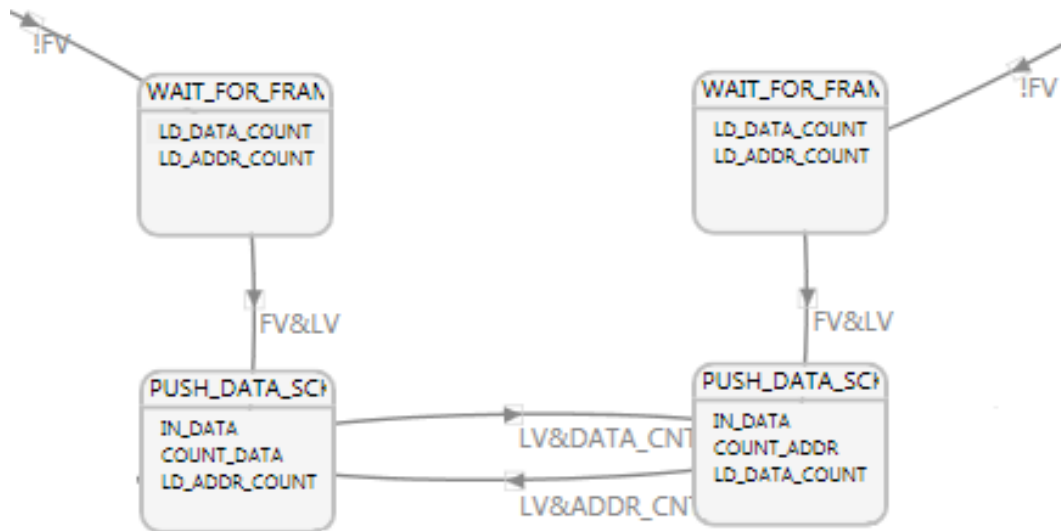
图 30. PUSH_DATA_SCK1 IN_DATA 操作



17. 创建从 PUSH_DATA_SCK0 状态到 PUSH_DATA_SCK1 状态的转换。通过使用“LV and DATA_CNT_HIT”公式可以编辑该转换公式输入。
18. 创建从 PUSH_DATA_SCK1 状态到 PUSH_DATA_SCK0 状态的转换。反转方向。通过使用“LV and ADDR_CNT_HIT”公式可以编辑该转换的公式输入。

这些转换都是在线有效期间於 DMA 缓冲区边界，各 GPIF 线程之间进行的切换。图 31 显示这部分状态机。

图 31. 图 10 显示了交替操作



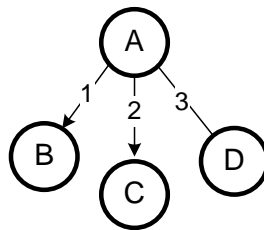
19. 将名称为“LINE_END_SCK0”的新状态添加到 PUSH_DATA_SCK0 状态的左侧。

20. 将名称为“LINE_END_SCK1”的新状态添加到 PUSH_DATA_SCK1 状态的右侧。

当图像传感器切换到下一个视频线时，同时 LV 信号被取消确认，则会进入这两个状态。由于使用不同的 GPIF 线程分别执行相同的操作，因此套接字 0 和套接字 1 两侧都需要这些状态。

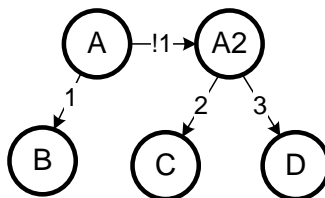
21. PUSH_DATA 状态需要三个可能的退出转换，但 GPIF II 硬件在每个状态中最多只能实现两个转换。通过创建一个虚拟状态可以处理三转换要求，该状态不进行任何操作，它只是在两个状态之间分配三个退出条件。为了演示该操作，图 32 显示的是一个状态 A 根据条件 1、2 或 3 转换到状态 B、C 或 D。

图 32. 状态 A 要求退出条件 1、2 和 3



22. 要使它 GPIF II 兼容，需要添加虚拟状态 A2，如图 33 所示。

图 33. 添加虚拟状态 A2 以与 GPIF II 兼容



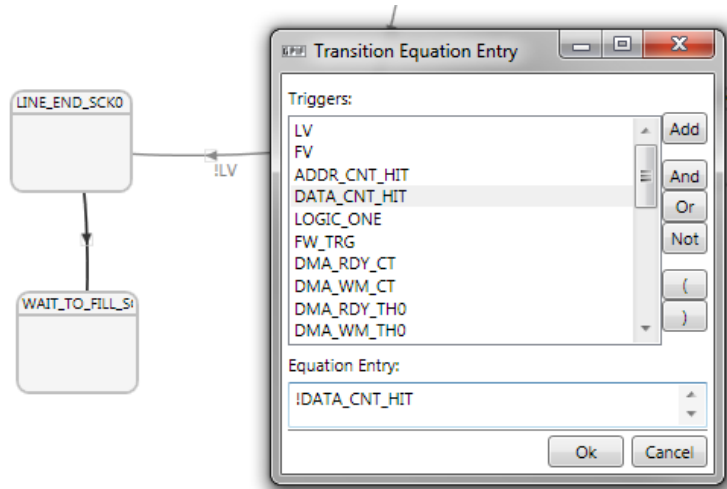
23. 创建作为虚拟状态使用的 LINE_END 状态。

24. 创建从 PUSH_DATA_SCK0 状态到 LINE_END_SCK0 状态的转换（该转换使用转换公式“(not LV)”）。

25. 创建从 PUSH_DATA_SCK1 状态到 LINE_END_SCK1 状态的转换（该转换使用转换公式“(not LV)”）。

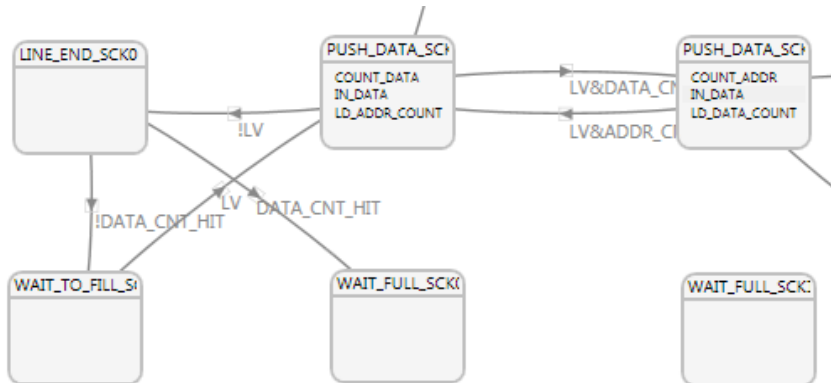
26. 在 LINE_END_SCK0 状态下, 创建新状态“WAIT_TO_FILL_SCK0”。
 27. 在 LINE_END_SCK1 状态下创建新状态“WAIT_TO_FILL_SCK1”。
- 当 DMA 缓冲区未满载却取消确认有效线时, 会进入这两个状态。
28. 创建从 LINE_END_SCK0 状态到 WAIT_TO_FILL_SCK0 状态的转换 (该转换使用转换公式“(not DATA_CNT_HIT)”), 如图 34 所示。

图 34. DATA_CNT_HIT 表示计数器已达到它的编程极限



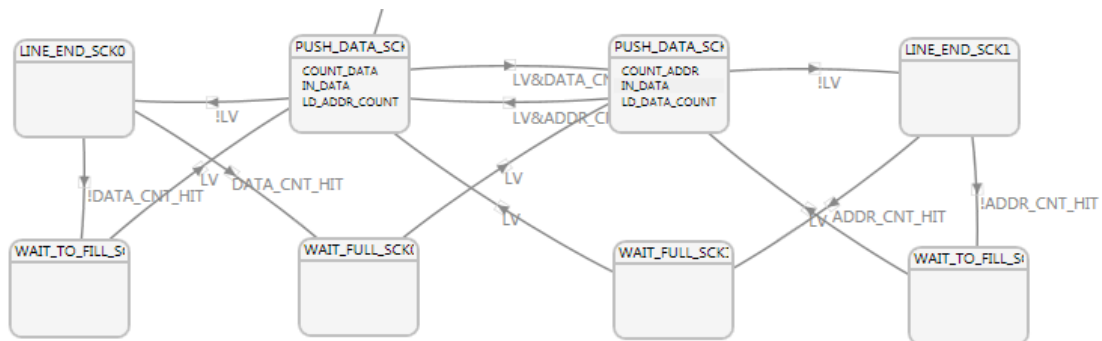
29. 创建从“WAIT_TO_FILL_SCK0”状态到“PUSH_DATA_SCK0”状态的回转 (该回转使用公式“LV”)。一旦该线有效, 数据传输将在同一个套接字内恢复。
30. 创建从“LINE_END_SCK1”状态到“WAIT_TO_FILL_SCK1”状态的转换 (该转换使用转换公式“(not ADDR_CNT_HIT)”)。
31. 创建从“WAIT_TO_FILL_SCK1”状态到“PUSH_DATA_SCK1”状态的转换 (该转换使用公式“LV”)。
32. 在“PUSH_DATA_SCK0”状态下添加新状态“WAIT_FULL_SCK0_NEXT_SCK1”。
33. 在“PUSH_DATA_SCK1”状态下添加新状态“WAIT_FULL_SCK1_NEXT_SCK0”。
34. 在这两个状态期间 (WAIT_FULL_), 图像传感器在 DMA 缓冲区边界上切换各线。因此, 下一个字节传输必须使用当前无效的 GPIF 线程。
35. 创建从“LINE_END_SCK0”状态到“WAIT_FULL_SCK0_NEXT_SCK1”状态的转换 (该转换使用公式“DATA_CNT_HIT”)。图 35 显示的是状态机的这个部分。

图 35. 已添加的 WAIT_FULL 状态



36. 创建从“LINE_END_SCK1”状态到“WAIT_FULL_SCK1_NEXT_SCK0”状态的转换（该转换使用公式“ADDR_CNT_HIT”）。
37. 创建从“WAIT_FULL_SCK0_NEXT_SCK1”状态到“PUSH_DATA_SCK1”状态的转换（该转换使用公式“LV”）。请注意，该操作会在框图中创建交叉链接。
38. 创建从“WAIT_FULL_SCK1_NEXT_SCK0”状态到“PUSH_DATA_SCK0”状态的转换（该转换使用公式“LV”）。图 36 显示的是该状态机的一部分。

图 36. DMA 缓冲区填满时的等待



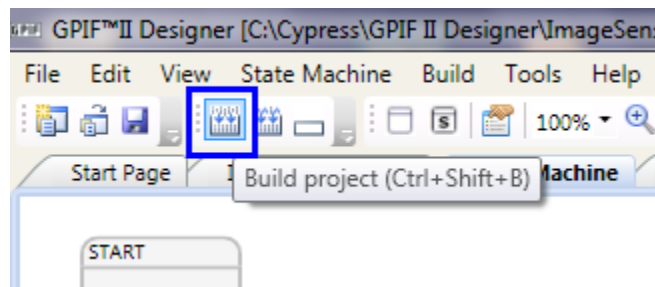
39. 在“WAIT_TO_FILL_SCK0”状态下创建新状态“PARTIAL_BUF_IN_SCK0”。
40. 在“WAIT_TO_FILL_SCK1”状态下创建新状态“PARTIAL_BUF_IN_SCK1”。
- PARTIAL_BUF_状态表示帧结束，这时，最后 DMA 缓冲区是未满足的。当 CPU 响应 GPIF II 生成的中断请求时，它必须手动传送该未满足的 DMA 缓冲区。
41. 在“WAIT_FULL_SCK0_NEXT_SCK1”状态下添加新状态“FULL_BUF_IN_SCK0”。
42. 在“WAIT_FULL_SCK1_NEXT_SCK0”状态下添加新状态“FULL_BUF_IN_SCK1”。
- FULL_BUF_的状态表示帧结束的数据是 DMA 缓冲区中的最后字节（该缓冲区与相应的 GPIF 线程相关联）。由于 GPIF II 硬件负责传送已满的 DMA 缓冲区，因此任何其他操作都是应用特定的。
43. 创建从“WAIT_TO_FILL_SCK0”状态到“PARTIAL_BUF_IN_SCK0”状态的转换（该转换使用公式“not FV”）。
44. 创建从“WAIT_TO_FILL_SCK1”状态到“PARTIAL_BUF_IN_SCK1”状态的转换（该转换使用公式“not FV”）。
45. 创建从“WAIT_FULL_SCK0_NEXT_SCK1”状态到“FULL_BUF_IN_SCK0”状态的转换（该转换使用公式“not FV”）。
46. 创建从“WAIT_FULL_SCK1_NEXT_SCK0”状态到“FULL_BUF_IN_SCK1”状态的转换（该转换使用公式“not FV”）。
47. 在“PARTIAL_BUF_IN_SCK0”、“PARTIAL_BUF_IN_SCK1”、“FULL_BUF_IN_SCK0”和“FULL_BUF_IN_SCK1”状态中添加操作“Intr_CPU”。

48. 添加新状态“FRAME_END_SCK_0”，并且添加从“PARTIAL_BUF_IN_SCK0”状态到“FULL_BUF_IN_SCK0”状态的转换（该转换使用“firmware trigger asserted (FW_TRIG)”公式）。然后，添加一个从“FRAME_END_SCK_0”状态到“IDLE_1”状态的转换（转换使用的公式为“firmware trigger de-asserted (!FW_TRIG)”）。
49. 类似于上一步骤，创建新状态“FRAME_END_SCK_1”，并添加从“PARTIAL_BUF_IN_SCK1”状态到“FULL_BUF_IN_SCK1”状态的转换（转换使用的公式为“firmware trigger asserted (FW_TRIG)”）。然后，添加从“FRAME_END_SCK_1”状态到“IDLE_0”状态的转换（转换使用的公式为“firmware trigger de-asserted (!FW_TRIG)”）。

图 38 显示的是最后状态机。与图 12 相比，主要区别在于该图添加了 PUSH_DATA 状态，这样可以适应任何状态中最多两个转换。

50. 通过选择“File-Save Project As”保存该项目。
51. 使用图 37 中加亮显示的 Build 图标编译该项目。项目输出窗口指的是编译的状态。

图 37. BUILD 按键



52. 检查项目的输出。在项目目录中，该输出显示为一个名称为 `cyfxgpiif2config.h` 的头文件。如果检查该头文件，您会发现 GPIF Designer II 已经创建了各 GPIF II 内部设置阵列。FX3 固件将使用这些设置来配置 GPIF II 并定义其状态机。请勿直接编辑该文件，而应该通过 GPIF II Designer 进行操作。

The diagram illustrates the state transitions for two channels, SCK0 and SCK1. The states and transitions are as follows:

- START_SCK0** transitions to **IDLE_SCK0** on **LOGIC_ONE**.
- IDLE_SCK0** transitions to **WAIT_FOR_FRAM** on **!FV**.
- WAIT_FOR_FRAM** (LD_DATA_COUNT, LD_ADDR_COUNT) transitions to **PUSH_DATA_SCK** on **FV&LV**.
- PUSH_DATA_SCK** (IN_DATA, COUNT_DATA, LD_ADDR_COUNT) transitions to **LINE_END_SCK** on **!LV**.
- PUSH_DATA_SCK** transitions to **WAIT_TO_FILL_S** on **!LV&DATA_CNT** and **!LV&ADDR_CNT**.
- LINE_END_SCK** transitions to **WAIT_TO_FILL_S** on **!LV&DATA_CNT_HIT**.
- WAIT_TO_FILL_S** transitions to **PARTIAL_BUF_IN** on **!FV**.
- PARTIAL_BUF_IN** (INTR_CPU) transitions to **FULL_BUF_IN_SC** on **FW_TRG**.
- FULL_BUF_IN_SC** (INTR_CPU) transitions to **WAIT_FULL_SCK** on **!FV**.
- WAIT_FULL_SCK** transitions to **LINE_END_SCK** on **!LV&ADDR_CNT_HIT**.
- WAIT_FULL_SCK** transitions to **WAIT_TO_FILL_S** on **!LV&ADDR_CNT_HIT**.
- WAIT_TO_FILL_S** transitions to **PARTIAL_BUF_IN** on **!FV**.
- PARTIAL_BUF_IN** transitions to **FULL_BUF_IN_SC** on **FW_TRG**.
- FULL_BUF_IN_SC** transitions to **FRAME_END_SCK** on **FW_TRG**.
- FRAME_END_SCK** transitions to **IDLE_SCK0** on **FW_TRG**.

The SCK1 channel follows a similar sequence of states and transitions, with the final **FRAME_END_SCK1** state also transitioning to **IDLE_SCK1** on **FW_TRG**.

3.6.5 编辑 GPIF II 接口的详细内容

本章节介绍的是如何修改接口设置 (若需要)。例如, 如果图像传感器/ASIC 具有 16 位宽的数据总线, 那么您需要修改 GPIF II 接口, 使其符合于该数据总线。请按照下列步骤进行操作:

1. 打开 GPIF II Designer 中的 **ImageSensorInterface.cyfx** 项目。(不能直接编译该项目)。
2. 依次选择 File->Save Project As。
3. 在将出现的对话框内合适的位置, 键入合适的名称保存该项目。
4. 关闭当前打开的项目 (File->Close Project)。
5. 打开步骤 3 所执行保存的项目。
6. 在 **Interface Definition** (接口定义) 选项卡中选择 **16 Bit** 选项, 以用于 **Address/Data Bus Usage** (地址/数据总线的使用情况) 设置。
7. 打开 **State Machine** (状态机) 选项卡。
8. 在状态机设计图中, 双击 WAIT_FOR_FRAME_START 状态中的 LD_DATA_COUNT 操作。将计数器的限定值改为 8183。
9. 对 LD_ADDR_COUNT 操作进行相同的操作。
10. 保存项目。
11. 编译项目。
12. 在步骤 3 所选择的位置内, 将生成的新 **cyfxgpiif2config.h** 头文件复制到固件的项目目录中。如果有 **cyfxgpiif2config.h** 文件存在, 请覆盖它。在附带源的 zip 文件中寻找固件项目目录 **cyfxuvc_an75779**。

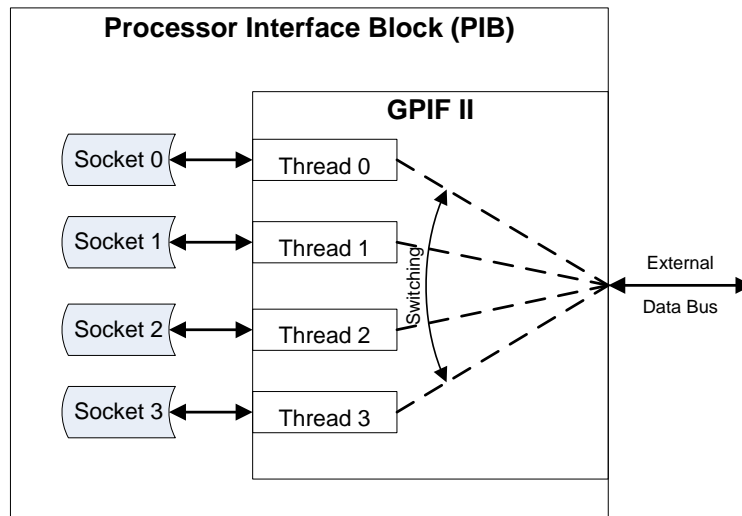
注意: 如果您将 GPIF II 总线的宽度改为 32 位, 请确保固件中 iomatrix 配置的 isDQ32Bit 参数被设置为 CyTrue。

下一节将介绍用于数据流和支持 UVC 的固件的 DMA 通道。

4 设置 DMA 系统

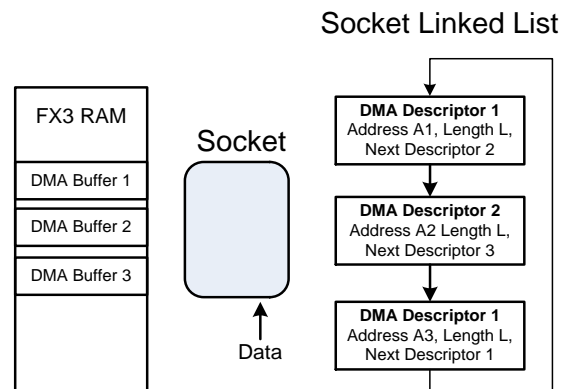
GPIF II 模块 (处理器接口模块 (PIB) 的一部分) 的工作频率可达 100 MHz, 而且数据总线宽度为 32 位 (400 MBps)。为了将数据传输到内部 DMA 缓冲区内, GPIF II 使用多个 GPIF 线程连接到 DMA 生产套接字 (如章节 3.3 中所述)。本应用使用了四个 GPIF 线程中的两个。它使用插座和 GPIF 线程的默认映射 (图 39) — 插座 0 与 GPIF 线程 0 相连, 插座 1 与 GPIF 线程 1 相连。在上一章节中设计的 GPIF II 状态机内实现 GPIF 线程切换。

图 39. GPIF II 套接字/线程的默认映像



为了理解 DMA 传输, 下面四个图中继续使用了套接字的概念。图 40 显示的是两个主插座属性、一个链接列表和一个数据路由器。

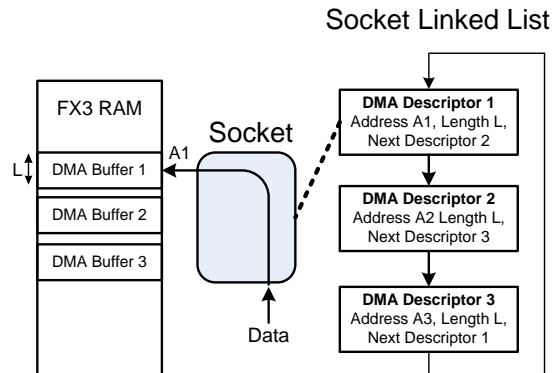
图 40. 套接字根据 DMA 描述符列表路由数据



套接字链接列表是主存储器中的一组数据结构, 这些结构又被称为 DMA 描述符。每个描述符指定了 DMA 缓冲区的地址和长度, 以及指向下个 DMA 描述符的指针。套接字运行时, 每次仅检索各 DMA 描述符中的一个描述符, 这样可以将数据路由到描述符地址和长度所指定的 DMA 缓冲区。传输 L 个字节后, 该套接字将检索下一个描述符, 并将继续将各字节传输到另一个 DMA 缓冲区内。

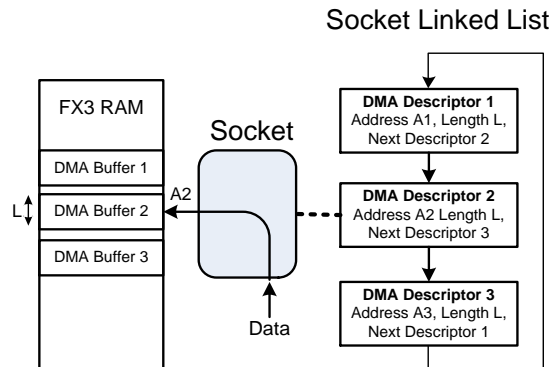
这结构使套接字变得非常灵活, 因为可以在存储器中的任何位置上创建任何 DMA 缓冲区数量, 而且这些缓冲区可以自动被链接在一起。例如, 图 40 中的插座以重复循环检索 DMA 描述符。

图 41. 使用 DMA 描述符 1 运行的套接字



在图 41 中，该插座加载了 DMA 描述符 1，并且传输操作开始于 A1 地址的字节，直到传输 L 字节结束。这时，它将检索 DMA 描述符 2 并对其地址和长度设置 (A2 和 L) 进行相同的操作 (图 42)。

图 42. 使用 DMA 描述符 2 运行的套接字



在图 43 中，该插座检索第三个 DMA 描述符，并传输从 A3 地址开始的数据。传输 L 字节后，将使用 DMA 描述符 1 重复该序列。

图 43. 使用 DMA 描述符 3 运行的套接字

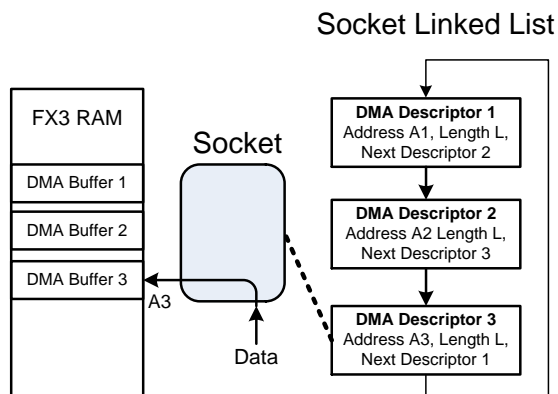
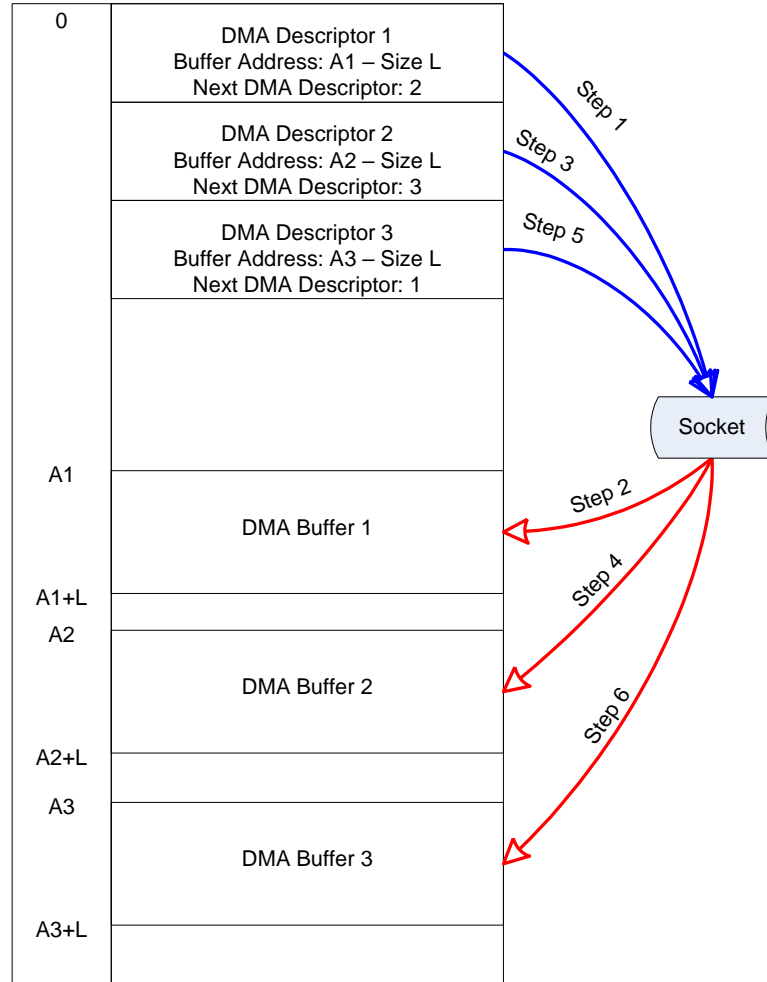


图 44 显示的是 DMA 数据传输的详细内容。本示例使用了三个 L 长度的 DMA 缓冲区，形成一个圆形循环。FX3 存储器地址位于左边。蓝色箭头表示套接字从存储器加载套接字链接列表的描述符。红色箭头显示的是结果数据的路径。下面各步骤描述了在数据转移到内部 DMA 缓冲区时，需要对套接字进行的操作序列。

步骤 1：将存储器中的 DMA 描述符 1 加载到套接字内。获取有关 DMA 缓冲区位置（A1）、DMA 缓冲区大小（L）和下一个描述符（DMA 描述符 2）的信息。转到步骤 2。

步骤 2：将数据传输到开始于 A1 地址的 DMA 缓冲区。传输完 DMA 缓冲区大小 L 的数据量后，转到步骤 3。

图 44. DMA 传输示例



步骤 3：加载当前 DMA 描述符 1 所指向的 DMA 描述符 2。获取有关 DMA 缓冲区的位置（A2）、DMA 缓冲区的大小（L）以及下一个描述符（DMA 描述符 3）的信息。转到步骤 4。

步骤 4：将数据传输到开始于 A2 地址的 DMA 缓冲区。传输 DMA 缓冲区 L 数据完成后，转到步骤 5。

步骤 5：加载当前 DMA 描述符 2 所指向的 DMA 描述符 3。获取有关 DMA 缓冲区的位置（A3）、DMA 缓冲区的大小（L）以及下一个描述符（DMA 描述符 1）的信息。转到步骤 6。

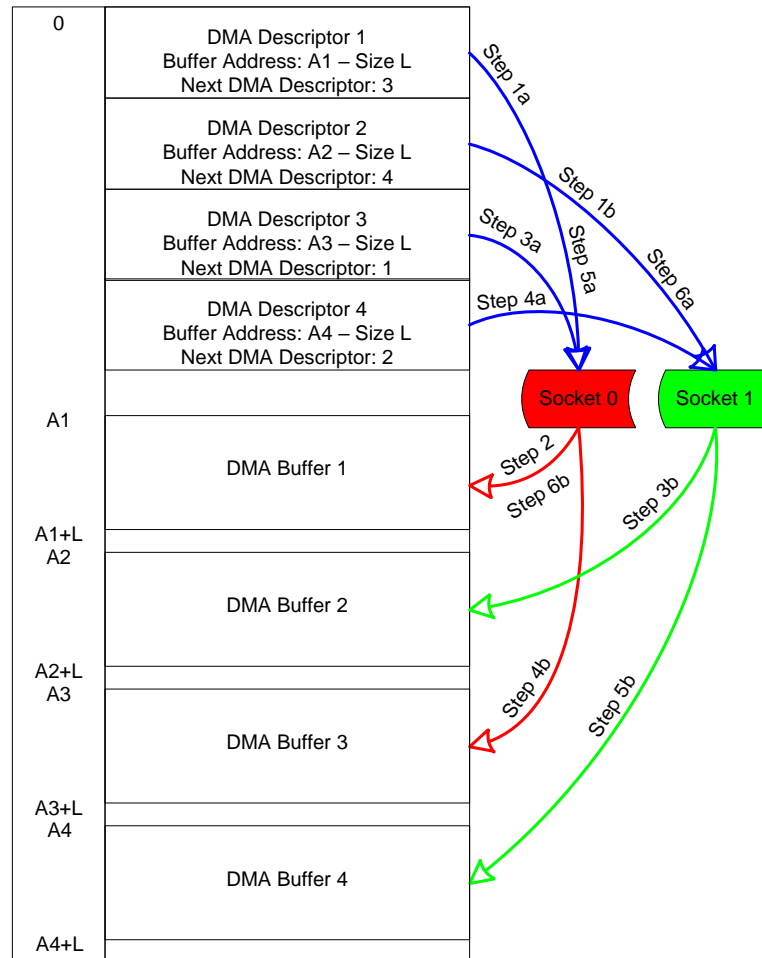
步骤 6：将数据传输到开始于 A3 地址的 DMA 缓冲区。传输 DMA 缓冲区 L 数据完成后，转到步骤 1。

该简单方案在摄像机应用中存在问题。套接字检索存储器中下一个 DMA 描述符的时间通常为 1 微秒。如果该传输的暂停发生在视频线中间，则视频数据将丢失。为了避免发生这种情况，可以将 DMA 缓冲区大小设置为视频线长度的倍数。这样将使 DMA 缓冲区切换造成的暂停与视频线无效事件（LV=0）同步发生。然而，在某些情况（如视频分辨率发生改变时），则该方法缺少灵活性。

设置 DMA 缓冲区大小等于线大小也不是一个好方案，因为 BULK 传输将不会利用 USB 3.0 的最大突发速率。USB 3.0 允许各 BULK 端点最大可达 16 个突发 x 1024 个字节。这就是将 DMA 缓冲区大小设置为 16 KB 的原因。

更优势的方案就是使套接字在一个时钟周期内进行切换而不需要延迟的时间。因此，建议使用两个套接字将数据存储四个交错的 DMA 缓冲区内。图 45 显示的是使用了双插座的数据传输以及执行的各步骤。套接字 0 和套接字 1 对 DMA 缓冲区的访问分别区分使用红色和绿色箭头（数据路径的单独套接字）。每一步骤中的 ‘a’ 和 ‘b’ 部分均同时发生。该硬件并行操作可以消除 DMA 描述符的检索死区时间，并允许 GPIF II 连续将数据串流到内部存储器内。这些步骤同图 13 中的 “Step” 线相对应。

图 45. 双套接字的无缝传输



步骤 1：启动套接字时，套接字 0 和套接字 1 分别加载 DMA 描述符 1 和 DMA 描述符 2。

步骤 2：当数据可用时，套接字 0 将数据传输到 DMA 缓冲区 1 内。传输长度为 L。传输结束后，请转到步骤 3。

步骤 3：GPIF II 切换 GPIF 线程，从而切换用于数据传输的套接字。套接字 1 开始将数据传输到 DMA 缓冲区 2 内，同时套接字 0 将加载 DMA 描述符 3。在套接字 1 完成传输长度为 L 的数据时，套接字 0 将就绪把数据传输到 DMA 缓冲区 3 内。

步骤 4：GPIF II 现在切换回到原始的 GPIF 线程。这时，套接字 0 将长度为 L 的数据传输到 DMA 缓冲区 3 内。同时，套接字 1 将数据传输到 DMA 描述符 4 内，以便就绪于将数据传输到 DMA 缓冲区 4 内。套接字 0 完成传输 L 长度的数据后，请转到步骤 5。

步骤 5: GPIF II 将套接字 1 中的数据路由到 DMA 缓冲区 4 内。同时, 套接字 0 通过加载 DMA 描述符 1 以准备将数据传输到 DMA 缓冲区 1 内。请注意, 步骤 5a 与步骤 1a 基本相同, 只区别在于它的套接字 1 未被启动, 另外数据传输是同步进行的。

步骤 6: GPIF II 再次切换套接字后, 套接字 0 开始将长度为 L 的数据传输到 DMA 缓冲区 1 内。这时, 由于 UIB 消耗套接字已经用尽, 因此 DMA 缓冲区为空。同时, 套接字 1 加载 DMA 描述符 2 并就绪将数据传输到 DMA 缓冲区 2 内。这时, 周期将转到执行路径的步骤 3。

只有消耗端 (USB) 为空并及时释放了 DMA 缓冲区 (用以接收 GPIF II 的下个视频数据块) 时, GPIF II 套接字才能传输视频数据。如果消耗端的速度不够快, 套接字将丢失数据, 因为 DMA 缓冲区的写操作被忽略。这样, 字节计数器将不再与实际传输同步, 而且此现象将传播到下一帧。因此, 在每一帧的结束后需要一个清除机制。请参考清除节, 以了解该机制。

根据图 12 所示的流程图, 一个帧传输会以下面的四个可能状态结束:

- 套接字 0 传输了已满的 DMA 缓冲区
- 套接字 1 传输了已满的 DMA 缓冲区
- 套接字 0 传输了未滿的 DMA 缓冲区
- 套接字 1 传输了未滿的 DMA 缓冲区

对于未滿的 DMA 缓冲区, CPU 需要将该缓冲区传送到 USB 接收器。

通过使用 `uvc.c` 文件中名称为 `CyFxDMAApplInit` 的函数可以启动 DMA 通道。在 `CyFxDMAApplInit` 函数中的 “dmaMultiConfig” 结构对 DMA 通道的配置详细内容进行了自定义。它的类型被设置为 `MANUAL_MANY_TO_ONE`。

另外, 将数据串流到 USB 3.0 主机的 USB 端点被配置为使能 16 突发 x 1024 字节的 BULK 端点。通过使 “CyU3PSetEpConfig” 函数中的 “endPointConfig” 结构可以实现该操作, 并且端点常量置为 “CY_FX_EP_BULK_VIDEO”

4.1 DMA 缓冲区的相关内容

本节介绍了如何创建 FX3 DMA 缓冲区, 并将其用于该应用。

- DMA 通道的组成部分在第 3.3 节中进行了介绍。
- 在该应用中, GPIF II 单元作为发送器, 而 FX3 USB 单元作为接收器。
- 该应用使用了 GPIF II 模块中的 GPIF 线程切换性能以避免丢失数据。
- 当发送套接字加载一个 DMA 描述符时, 它将检查相关的 DMA 缓冲区以确定该缓冲区是否能够进行写操作。发送套接字将其状态改为 “活动”, 如果 DMA 缓冲区为空, 它会把数据写入到 FX3 RAM 内。发送套接字锁定 DMA 缓冲区, 用以进行写操作。
- 当发送套接字完成写入 DMA 缓冲区后, 它会释放锁定, 以使接收套接字可以访问 DMA 缓冲区。该操作被称为 “缓冲包装” 或简单叫做 “包装”。然后, DMA 单元将 DMA 缓冲区传送给 FX3 RAM。发送套接字发送了一个 DMA 缓冲区。缓冲区只在发送器不对其填充时才能被包装。这是状态框中存在 `FV=0` 测试的原因。
- 如果 DMA 缓冲区完全被填满, 并且在帧传输期间重复填写操作, 则包装操作会自动进行。发送套接字释放 DMA 缓冲区上的锁定, 将其传送到 FX3 RAM 内, 切换到一个空的 DMA 缓冲区, 然后继续写入视频数据流。
- 当接收套接字加载了一个 DMA 描述符时, 它将检查相关的 DMA 缓冲区以确定该缓冲区是否能够进行读操作。如果 DMA 缓冲区已被传输, 接收套接字会将它的状态改为 “活动”, 以读取 FX3 RAM 中的数据。接收套接字锁定 DMA 缓冲区, 以执行读操作。
- 当接收套接字读取了 DMA 缓冲区中所有的数据后, 它会释放锁定, 以使发送套接字能够访问 DMA 缓冲区。接收套接字接收了 DMA 缓冲区。
- 如果发送套接字和接收套接字使用了相同的 DMA 描述符, 那么 DMA 缓冲区的满/空状态将通过 DMA 描述符和套接字间的事件在发送套接字和接收套接字之间自动通信。
- 在该应用中, 由于 CPU 需要添加一个 12 字节的 UVC 标头, 因此发送套接字和接收套接字需要加载不同的 DMA 描述符组。发送套接字所加载的 DMA 描述符将指向 DMA 缓冲区, 这些缓冲区位于离接收套接字所加载的 DMA 描述符指向的相应 DMA 缓冲区 12 字节偏移的位置上。

- 由于发送套接字和接收套接字使用了不同的 DMA 描述符，因此 CPU 必须管理发送套接字和接收套接字之间 DMA 缓冲区状态的通信。这就是 DMA 通道的实现被称为“手动 DMA”通道的原因。
- GPIF II 模块发送 DMA 缓冲区后，将通过中断通知 CPU。然后，CPU 添加标头信息并将 DMA 缓冲区传送到接收器 (USB 接口模块)。
- 在 GPIF II 侧，DMA 缓冲区中的视频数据被自动包装，并被传送到 FX3 RAM (帧的最后 DMA 缓冲区除外)。不需要 CPU 的干预。
- 在帧传输结束时，最后的 DMA 缓冲区可能没有完全填满。这时，CPU 会手动包装 DMA 缓冲区，并将该缓冲区传送到 FX3 RAM。该操作被称为“强制包装”。

5 FX3 固件

本应用笔记的固件示例位于源压缩文件中的 `cyfxuvc_an75779` 文件夹下面。请参考 [AN75705 — “EZ-USB FX3 入门”](#)，了解如何将固件项目导入到 Eclipse 工作区内。该固件示例可进行编译及枚举，但用户必须通过使用 `sensor.c` 和 `sensor.h` 文件为特定的图像传感器定制固件，以便可以从图像传感器串流视频。这些文件被作为示例项目中的占位符，应该在该位置上实现代码来配置图像传感器。

如果使用安森半导体 (Aptina) 传感器板，如本应用笔记的 [第 6 节](#) 所述的内容，则赛普拉斯所提供的 `sensor.c` 和 `sensor.h` 文件将同该特定传感器相对应。通过该项目所提供的预编译加载图像可以尝试配置 FX3-Aptina ([第 7 节](#))。[表 8](#) 汇总了各种代码模块以及每个模块中所执行的各函数。

表 8. 示例项目文件

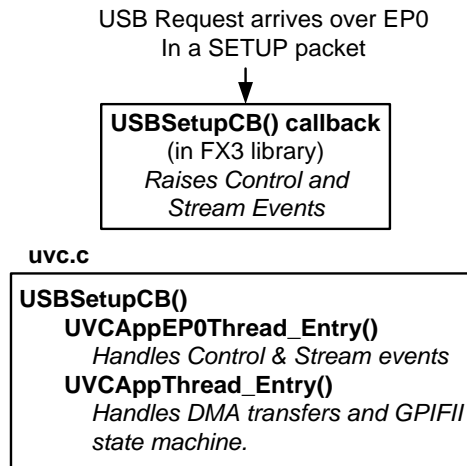
文件	说明
<code>sensor.c</code>	<ul style="list-style-type: none"> 定义 <code>SensorWrite2B</code>、<code>SensorWrite</code>、<code>SensorRead2B</code> 和 <code>SensorRead</code> 函数，以通过 I²C 接口对图像传感器配置进行读写操作。这些函数假设图像传感器的 I²C 总线上的寄存器地址的宽度是 16 位。 定义 <code>SensorReset</code> 函数可以控制图像传感器的复位线，另外通过使用 <code>SensorInit</code> 函数可以检测 I²C 连接并将图像传感器配置为默认流模式 (使用 <code>SensorScaling_HD720p_30fps</code> 函数) 定义 <code>SensorScaling_VGA</code> 和 <code>SensorScaling_HD720p_30fps</code> 函数可以将图像传感器配置为所需要的流模式。这些函数是占位符函数，因此，必须使用图像传感器特定的配置指令来填充它们。 通过定义 <code>SensorGetBrightness</code> 可以从图像传感器的亮度控制寄存器读取亮度值，同时通过定义 <code>SensorSetBrightness</code> 函数可以将亮度值写入该寄存器内。这些函数是占位符函数，因此，必须使用图像传感器特定的配置指令来填充它们。
<code>sensor.h</code>	<ul style="list-style-type: none"> 包含用于图像传感器的常量 (它的 I²C 从设备地址和 GPIO 复位编号)。在该文件内，您必须定义图像传感器的 I²C 从地址。 包含定义在 <code>sensor.c</code> 中的所有函数声明。
<code>camera_ptzcontrol.c</code>	<ul style="list-style-type: none"> 定义对摄像机的 PTZ 值进行读写操作的函数。这些函数是占位符函数，因此，必须使用图像传感器特定的配置指令来填充它们。 取消注释 <code>uvc.h</code> 中的 <code>#define UVC_PTZ_SUPPORT</code> 线，以使能 PTZ 控制代码的占位符。
<code>camera_ptzcontrol.h</code>	<ul style="list-style-type: none"> 包含用于 PTZ 控制实现的常量 包含了定义在 <code>camera_ptzcontrol.c</code> 文件中的所有函数声明。
<code>cyfctx.c</code>	<ul style="list-style-type: none"> 无需更改。将本应用笔记所提供的该文件使用于相关的项目。 它包含了 RTOS 和 FX3 API 库所用于存储器映射的变量，以及 FX3 API 库所用于存储器管理的函数。
<code>cyfxgpi2config.h</code>	<ul style="list-style-type: none"> 头文件由 GPIF II Designer 工具生成。无需更改。如果需要修改接口，则 GPIF II Designer 工具会生成一个新的头文件。它将取代旧的头文件。 包含传送到 <code>uvc.c</code> 文件中的 API 调用的结构和常量，以启动和运行 GPIF II 状态机。
<code>uvc.c</code>	<ul style="list-style-type: none"> UVC 应用的主源文件。修改代码以支持其他控制功能 (亮度和 PTZ 控制除外) 或添加各种视频数据流模式的支持时，需要进修改该文件。 它包括下面各函数： <ul style="list-style-type: none"> ○ <code>Main</code>: 启动 FX3 器件，设置缓存，配置 FX3 I/O 和启动 RTOS 内核。 ○ <code>CyFxApplicationDefine</code>: 定义 RTOS 执行的两个应用线程

文件	说明
	<ul style="list-style-type: none"> ○ UVCAppThread_Entry: 是第一个应用线程函数，用于调用 FX3 内部模块的启动函数，枚举器件，然后处理视频数据传输 ○ UVCAppEP0Thread_Entry: 是第二个应用线程函数，等待表示已收到 UVC 特定请求的事件并调用相应的函数进行处理这些请求 ○ UVCHandleProcessingUnitRqts: 处理 VC 请求，用于处理单元能力 ○ UVCHandleCameraTerminalRqts: 处理 VC 请求，用于输入终端能力 ○ UVCHandleExtensionUnitRqts: 处理 VC 请求，用于扩展单元能力 ○ UVCHandleInterfaceCtrlRqts: 处理不用于任何终端或单元的 VC 请求 ○ UVCHandleVideoStreamingRqts: 处理 VS 请求以更改流模式 ○ CyFxCyUVCAppInDebugInit: 初始用于打印调试信息的 FX3 UART 模块 ○ CyFxCyUVCAppInI2CInit: 初始用于图像传感器配置的 FX3 I2C 模块 ○ CyFxCyUVCAppInInit: 初始化 FX3 GPIO 模块，处理器模块（GPIF II 为处理器模块的一部分），传感器（配置为 720p 30fps），用于枚举的 USB 模块，用于 USB 传输的端点配置存储器，以及用于将数据从 GPIF II 传输到 USB 的 DMA 通道配置 ○ CyFxCyUVCAppGpifInit: 初始 GPIF II 状态机并启动它 ○ CyFxCyGpifCB: 处理 GPIF II 状态机所生成的 CPU 中断 ○ CyFxCyUVCAppCommitEOF: 将 GPIF II 状态机提供的部分 DMA 缓冲区发送到 FX3 RAM ○ CyFxCyUVCAppInDmaCallback: 跟踪从 FX3 到主机的输出视频数据 ○ CyFxCyUVCAppInUSBSetupCB: 处理所有由主机发送的控制请求，设置表示已接收主机所发送的 UVC 特定请求的事件，同时检测串流停止的时间 ○ CyFxCyUVCAppInUSBEventCB: 处理各个 USB 事件，如暂停、线缆断开连接、复位和恢复等事件 ○ CyFxCyUVCAppInAbortHandler: 当发生任何错误或检测到串流停止请求时，使用该函数将停止流视频数据 ○ CyFxCyAppErrorHandler: 错误处理函数。这是一个占位符函数，它允许您执行错误处理（若需要） ○ CyFxCyUVCAddHeader: 在有效串流期间将 UVC 标头添加到视频数据中 ○ CyFxCyUVCAppInStop: 停止 GPIF II 状态机并重置 DMA 和端点。 ○ CyFxCyUVCAppInStart: 开始 DMA 通道传输和 GPIF 状态机。 ○ CyFxCyUVCAppProgressTimer: 当传感器/ISP（图像信号处理器）在预定义的帧间隔时间内无法发送视频数据时，使用该函数将重置 DMA 和端点并重新开始串流。
cyfxuvcdscr.c	<ul style="list-style-type: none"> ▪ 包含用于 UVC 应用的 USB 枚举描述符。如果需要修改帧速率、图像分辨率、位深度或支持的视频控制，则需要修改该文件。UVC 规范包含了所有需要的详细信息。
uvc.h	<ul style="list-style-type: none"> ▪ 包含各个用于更改应用状态的开关，以便打开/关闭 PTZ 支持，调试接口以及调试用于帧计数和帧计数器的打印。 ▪ 包含通常用于 <i>uvc.c</i>、<i>cyfxuvcdscr.c</i>、<i>camera_ptzcontrol.c</i> 以及 <i>sensor.c</i> 文件的各常量。
cyfx_gcc_startup.S	<ul style="list-style-type: none"> ▪ 该汇编源文件包含了 FX3 CPU 启动代码。它具有用于设置堆栈和中断向量的函数。 ▪ 无需更改。

首先，固件将初始化 FX3 CPU 并配置它的 I/O。然后，通过调用函数（**CyU3PKernelEntry**）来启动 ThreadX 实时操作系统（RTOS）。它创建了两个应用线程：**uvcAppThread** 和 **uvcAppEP0Thread**。RTOS 将调配资源执行这些应用线程，并安排执行应用线程函数的顺序，即：分别执行 **UVCAppThread_Entry** 和 **UVCAppEP0Thread_Entry**。

图 46 显示的是基本的程序结构。

图 46. 摄像机项目结构



5.1 应用线程

两个应用线程允许实现并行功能。**UVCAppThread**（应用）线程用于管理视频数据流。它在串流开始前等待串流事件，在串流开始后发送 DMA 缓冲区，并且在每一帧或串流流结束后清除 FIFO。

固件将通过 EP0 处理 UVC 特定控制的请求（SET_CUR、GET_CUR、GET_MIN 和 GET_MAX），如亮度、PTZ 和 PROBE/COMMIT 控制。特殊类别的控制请求由应用线程中的 **CyFxUVCAppInUSBSetupCB**（CB=Callback）函数处理。每当 FX3 收到某个控制请求时，该函数将引发相应的事件，并立即释放主应用线程，以执行它的其他并行任务。然后，EP0 线程将触发这些事件，以处理类别特定的请求。

5.2 初始化

UVCAppThread_Entry 将通过调用 **CyFxUVCAppInDebugInit** 来初始化 UART 的调试能力，调用 **CyFxUVCAppInI2CInit** 来初始化 FX3 的 I²C 模块，调用 **CyFxUVCAppInInit** 来初始化其余所需模块、DMA 通道和 USB 端点。

5.3 枚举

在 **CyFxUVCAppInInit** 函数中，通过调用 **CyU3PUsbSetDesc** 函数可以确保 FX3 被枚举为一个 UVC 器件。在 **cyfxuvcdscr.c** 文件中定义了 UVC 描述符。这些描述符的定义用于使用非压缩的 YUY2 格式传输每像素为 16 位的图像传感器，其分辨率为 1280 x 720 像素，频率为 30 FPS。若需要修改这些设置，请参考第 2.3.1 节中的内容。

5.4 使用 I²C 接口配置图像传感器

使用 FX3 的 I²C 主控模块来配置图像传感器。使用 **SensorWrite2B**、**SensorWrite**、**SensorRead2B** 和 **SensorRead** 函数（定义在 **sensor.c** 文件中）通过 I²C 接口对图像传感器配置进行读写操作。**SensorWrite2B** 和 **SensorWrite** 函数将调用标准的 API **CyU3PI2cTransmitBytes**，向图像传感器写入数据。**SensorRead2B** 和 **SensorRead** 将调用标准的 API **CyU3PI2cReceiveBytes**，读取图像传感器内的数据。有关更多这些 API 的详细信息，请查阅 **FX3 SDK API 指南** 中的内容。

5.5 启动视频流

USB 主机应用 (如 VLC 播放器、AMCap 或 VirtualDub) 通过使用 UVC 驱动程序来显示视频, 将 USB 接口和 USB 备用设置组合到一个流视频 (通常为接口 0 备用设置 1), 并发送 PROBE/COMMIT 控制。该主机指示符表示快要启动一个视频数据流。发生流事件时, USB 主机应用会向 FX3 请求图像数据; FX3 应将图像数据从图像传感器发送给 USB 3.0 主机。在固件中, **UVCAppThread_Entry** 函数是一个无限循环。当没有流事件时, 主应用线程将在该循环下等待, 直到发生流事件为止。

注意: 如果没有流事件, FX3 不需要传输任何数据。因此, GPIF II 状态机将被禁用。否则, 主机应用从 DMA 缓冲区读取数据前, DMA 缓冲区将被填满, 并且 FX3 将传送一个坏帧。因此, 只有发生了流事件时, 才能起始 GPIF II 状态机。

当 FX3 收到流事件时, 主应用线程将调用 **CyU3PGpifSMSwitch** 函数来启动 GPIF II 状态机。在该函数中, 固件将从任何状态切换到启动状态 (**START_SCK0**), 并重新开始 GPIF 状态机。将启动状态名和启动条件作为参数使用, 并将其传递到 **CyU3PGpifSMSwitch** 函数内。启动状态和结束状态可以是任何无效状态 (请见项目附带的 257 个状态), 因为它保证 GPIF II 状态机将从任何状态切换到启动状态。在 **cyfxgpif2config.h** 文件中定义了启动状态 (**START_SCK0**) 和启动条件 (**ALPHA_START_SCK0**)。 **cyfxgpif2config.h** 文件由 GPIF II Designer 工具生成, 如第 3.6 节中所述。

注意: FX3 SDK API 指南包含 GPIF II 相关函数的详细信息, 如 **CyU3PGpifLoad** 和 **CyU3PGpifSMSwitch**。

5.6 设置 DMA 缓冲区

UVC 规范要求每个 USB 传输 (在本应用中即为每个 16 KB DMA 传输) 增加一个大小为 12 字节的标头。但是, FX3 架构要求所有与 DMA 描述符相关的 DMA 缓冲区的大小必须为 16 字节的倍数。

由于在 DMA 缓冲区中保留了供 FX3 CPU 填充的 12 字节, 因此 DMA 缓冲区大小将不是 16 字节的倍数。所以, DMA 缓冲区大小应为 16,384 减去 16, 而不是减去 12。DMA 大小 (不包括 FX3 固件所添加的 12 字节标头) 为 16,384-16=16,368 字节。DMA 缓冲区的结构如下: 12 字节的标头, 16,368 视频字节, 以及在 DMA 缓冲区传输结束时的 4 个未使用字节。这样, 所创建的 DMA 缓冲区可以利用最大的 USB BULK 突发空间为 16 x 1024 (即 16,384) 字节数据包。

5.7 在视频流期间中处理 DMA 缓冲区

CyFxUVCAppInInit 函数为发送和接收事件创建了一个具有回调通知的 DMA 手动通道。该通知用于跟踪传感器所传输的数据量以及主机所读取的数据量。当 GPIF II 传送 DMA 缓冲区, 并且 FX3 通过 **CyU3PDmaMultiChannelGetBuffer** 获得指向该缓冲区的指针时, FX3 CPU 可以使用 DMA 缓冲区。该缓冲区通过 **CyU3PDmaMultiChannelCommitBuffer** 被传输到 USB。当 USB 吞吐量比缓冲区填满 (通过 GPIF II 状态机) 的速度更慢时, 具有实际视频数据的缓冲区的数量则会增加。对于所有缓冲区都被填满的情况下, 传输多于一个缓冲区将导致错误。这情况叫做缓冲区传输失败。在该情况下, 将调用 DMA 复位事件, 并且 FX3 将停止和重新启动串流。

通常, 帧结束时, 最后的 DMA 缓冲区将未满载。在这种情况下, 固件必须在发送器上对 DMA 缓冲区实现强制包装, 并且触发一个发送事件, 然后将 DMA 缓冲区中合适的字节数传输到 USB。DMA 缓冲区的强制包装 (由 GPIF II 发送) 由 GPIF II 回调函数 **CyU3PDmaMultiChannelSetWrapUp** 执行。当 GPIF II 设置一个 CPU 中断时, 将触发 **CyFxGpifCB** 回调函数。取消确认帧有效信号时, 将生成该中断, 如 GPIF II 状态图所示。

UVC 标头包含了有关帧标识符和帧结束标志的信息。在帧的结束时, FX3 固件设置第二个 UVC 标头字节的位 1, 并切换它的位 0 (请查看“**CyFxUVCAddHeader**”)。仅在帧结束后, 才能切换 UVC 标头 FID 位。

一个慢速的 USB 主机将导致缓冲区传送失败, 并引起 DMA 重置和视频流的重启。如果传感器/ISP 无法按时传输视频, 视频将冻结或显示抖动。帧计数器是避免这种情况的基本实体。当视频串流开始时, 帧计数器也将开始。当 FX3 接收传感器的第一个缓冲区并重新启动以确保每个发送缓冲区准时到达时, 该计数器被复位。

5.8 终止视频流

共有三种方法可以终止图像流: 断开主机与摄像机的连接, 关闭 USB 主机程序, 或 USB 主机向 FX3 发送复位或暂停请求。这些操作将触发 **CY_FX_UVC_STREAM_ABORT_EVENT** 事件 (请参考 **CyFxUVCAppInAbortHdlr** 函数)。在 FX3 FIFO 中没有数据时, 将不会发生这些操作。换句话说, 需要做适当的清除。固件会复位与串流相关的变量, 并重置 **UVCAppThread_Entry** 环中的 DMA 通道。由于不需要任何流, 所以不会调用 **CyU3PGpifSMSwitch** 函数。这时, 固件将等待发下一个流事件发生。

当关闭应用时，它将在 Windows 平台上发送一个清除请求，或在 Mac 平台上触发一个备用设置为 0 的设置界面请求。收到该请求时，将停止串流操作。在 CY_U3P_USB_TARGET_ENDPT 的 switch 语句及 CY_U3P_USB_SC_CLEAR_FEATURE 请求下，CyFxCvUVCAppInUSBSetupCB 函数将处理该请求。

5.9 增加“调试”接口

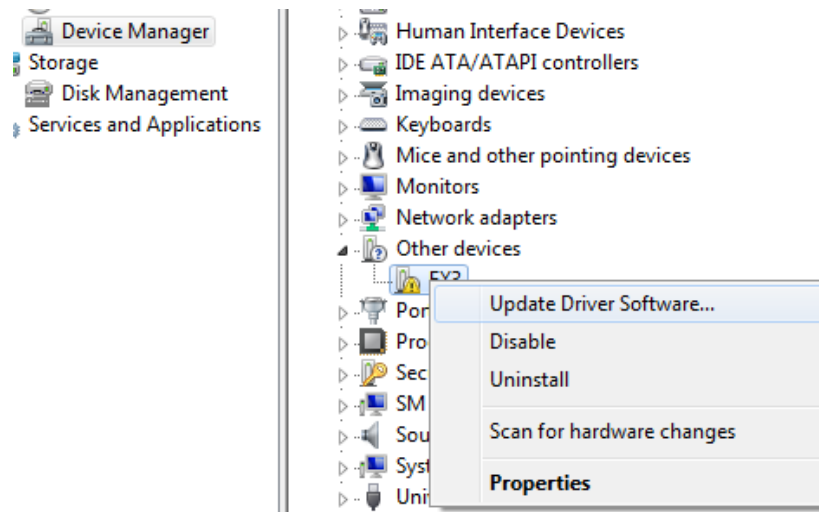
该部分介绍了如何向摄像机应用增加第三个 USB 接口用于调试或其他自定义目的。

uvc.h 文件中的#define USB_DEBUG_INTERFACE switch 语句使能了 cyfxuvcdscr.c、uvc.h 和 uvc.c 文件中的代码。该示例介绍了通过 I²C 对图像传感器寄存器进行的读和写操作。

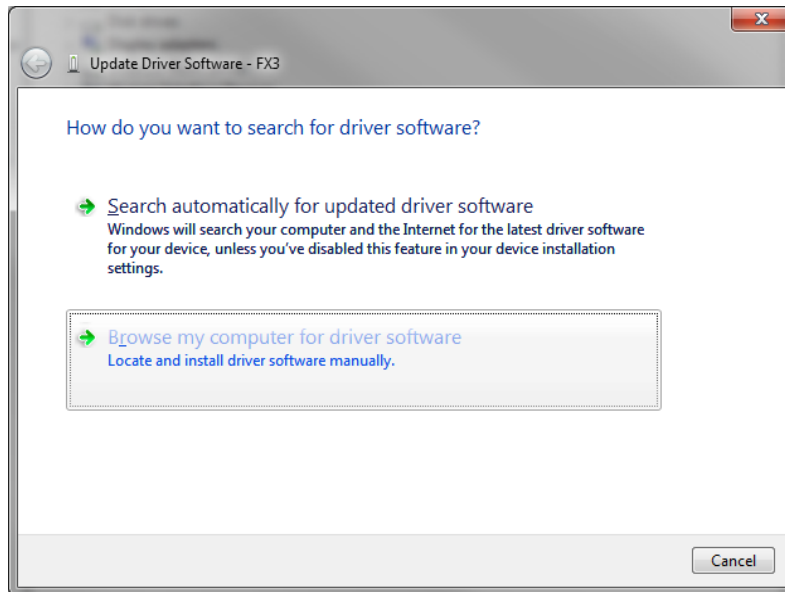
5.9.1 调试接口的详细内容

两个 BULK 端点被定义用于该接口。EP 4 OUT 被配置为调试指令 BULK 端点，EP 4 IN 则被配置为调试响应 BULK 端点。当运行使能了调试接口的固件时，FX3 将在枚举过程中报告三个接口。前两个接口是 UVC 的控制和串流接口，第三个则是调试接口。需要将第三个接口绑定到 FX3 SDK 的 CyUSB3.sys 驱动程序。您可以根据下面介绍的内容安装驱动程序。（64 位 Windows 7 系统作为示例使用。XP 系统用户需要在“.inf”文件中添加 VID/PID，然后使用修改好的“.inf”文件在该接口上安装 cyusb3.sys 驱动程序。）

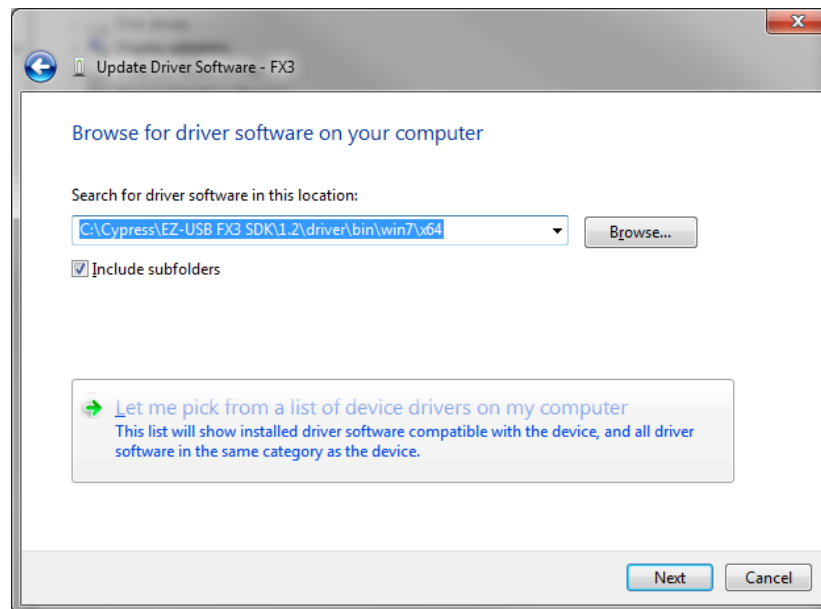
1. 打开器件管理器，右键点击“Other devices”下的 FX3（或同等选项），然后选中“Update Driver Software...”项



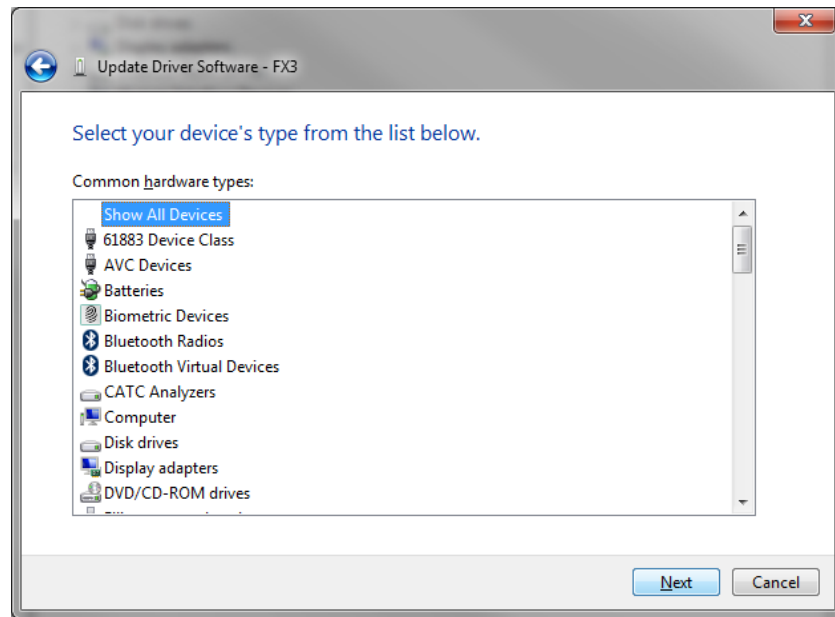
- 在接下来显示的屏幕上，选中“Browse my computer for driver software”项



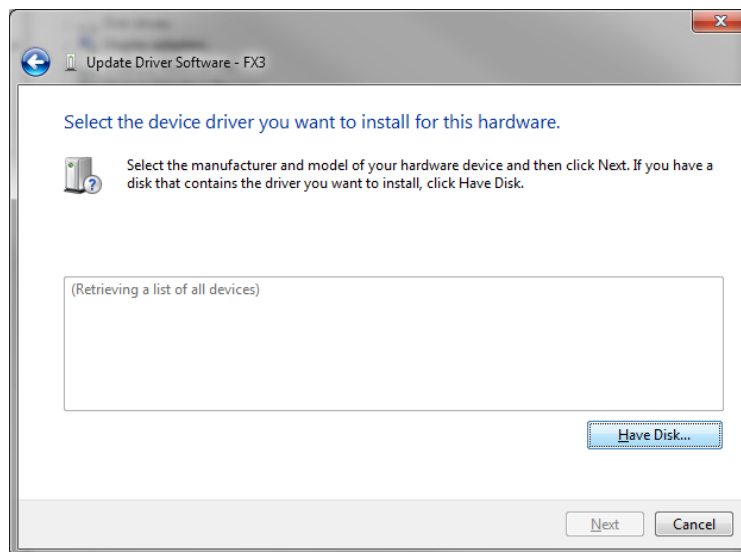
- 然后，选择“Let me pick from a list of device drivers on my computer”



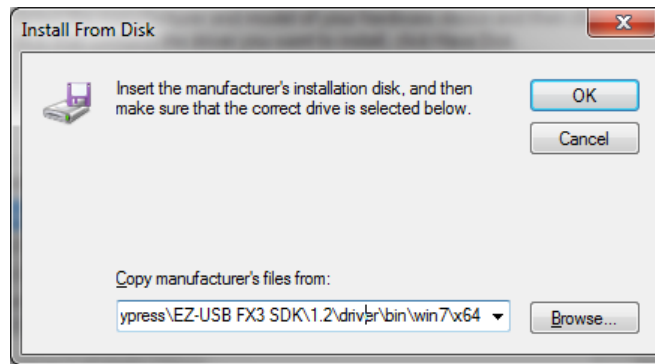
- 在接下来显示的屏幕上点击“Next”



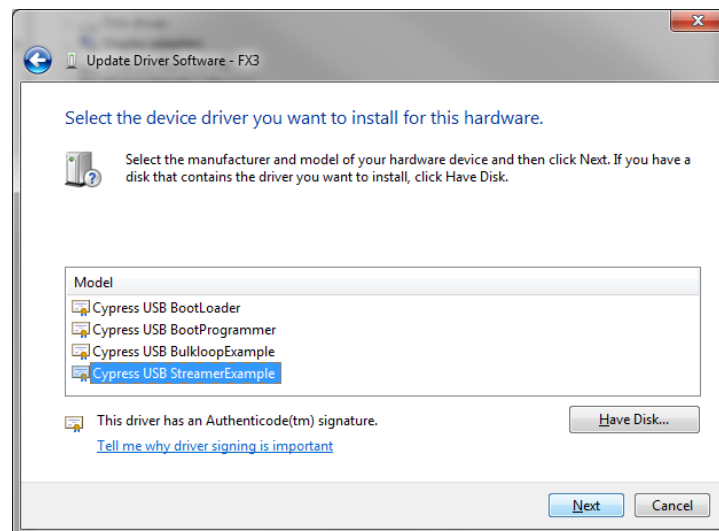
5. 点击“Have Disk...”，选中驱动程序



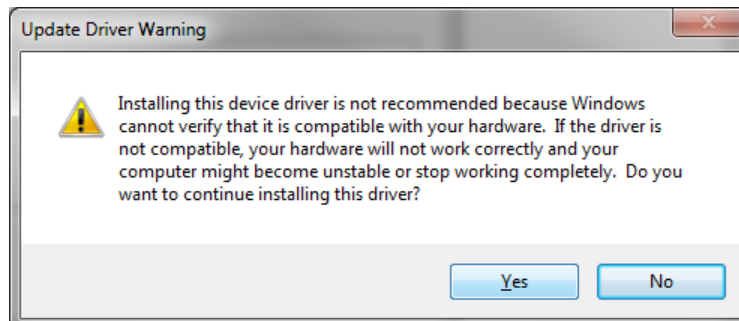
6. 浏览<SDK installation>\<version>\driver\bin\<OS>\x86 或\x64 文件夹中的 cyusb3.inf 文件，然后点击“OK”



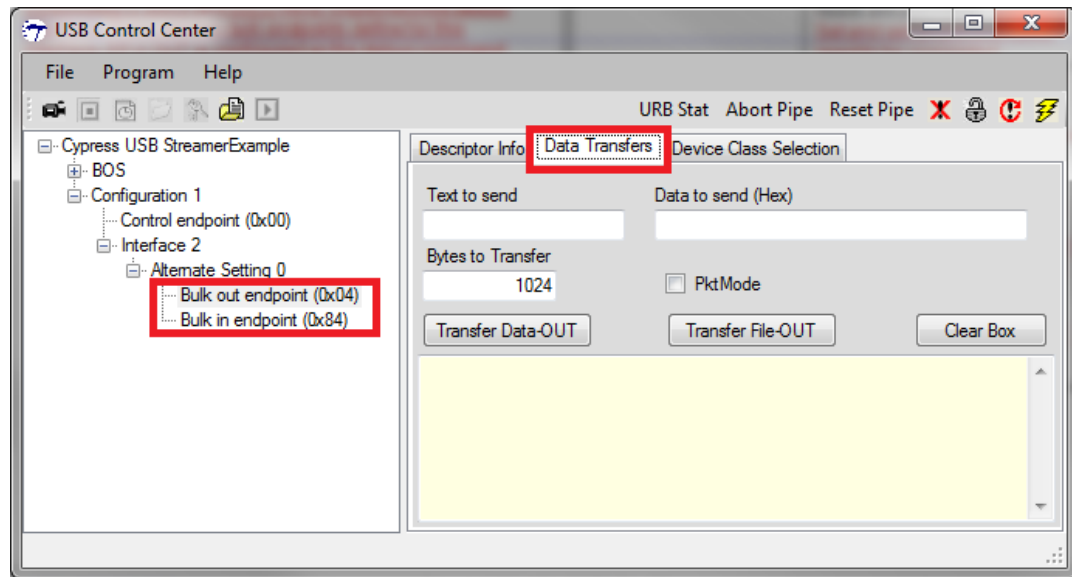
7. 选中一个操作系统版本，并通过点击“Next”进行安装



8. 点击警报对话框中的“Yes”（若出现）



在第三个接口上安装驱动程序后，器件将显示在赛普拉斯 USB 控制中心应用中。您可以在这里将 FX3 固件作为供应商特定的器件进行访问。当前实现的调试接口允许使用 EP4-OUT 指令和 EP4-IN 响应端点对图像传感器寄存器进行读和写操作。通过 Control Center 中的 <FX3 device name> → Configuration 1 → Interface 2 → Alternate Setting 0，以访问这些端点，如下图所示。使用“Data Transfer”选项卡来发送一个指令或发送指令后接收响应。

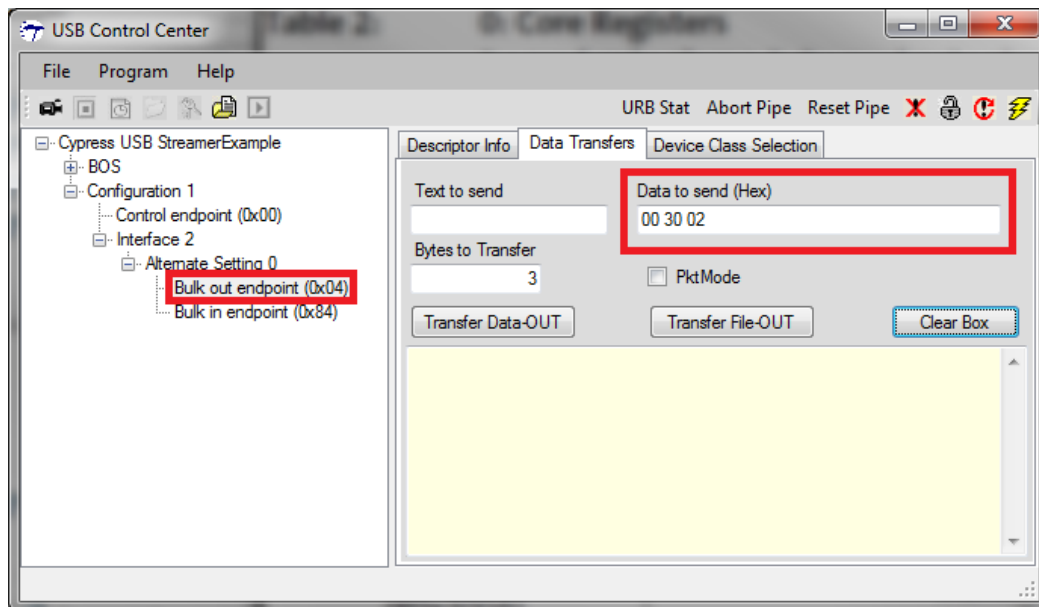


5.9.2 使用调试接口

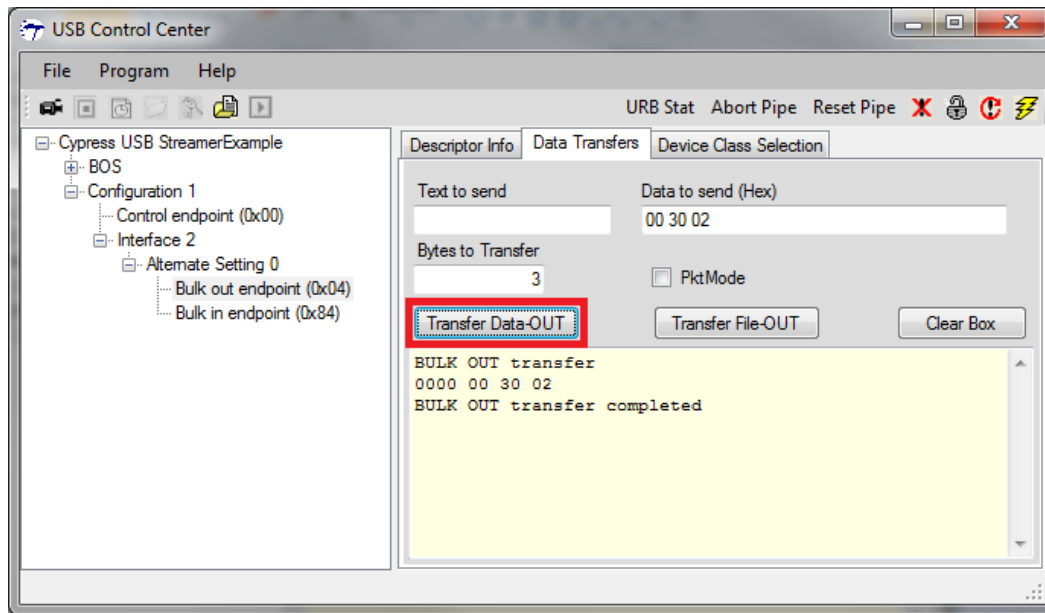
在调试接口中执行的四个指令分别是：单一读取、连续读取、单一写入和连续写入。因为没有错误检查，所以在输入指令时需要特别小心。您可以执行错误检查，以确保功能正确。I²C 寄存器的宽度为 16 位，通过使用 16 位可以寻址这些寄存器。

单一读取：

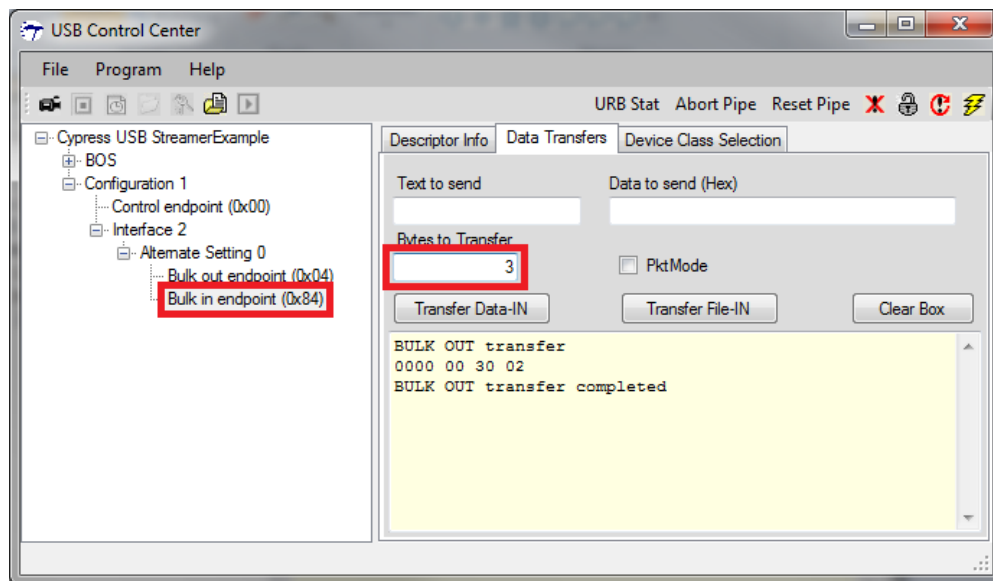
1. 在“Data transfers”选项卡下，选中指令端点并以十六进制的格式输入指令。单一读取的指令格式为 0x00 <寄存器地址高字节> <寄存器地址低字节>。下图显示的是寄存器地址 0x3002 的读指令。向十六进制的数据框中输入指令时，请勿使用空格。例如，点击十六进制数据字段并输入“003002”。



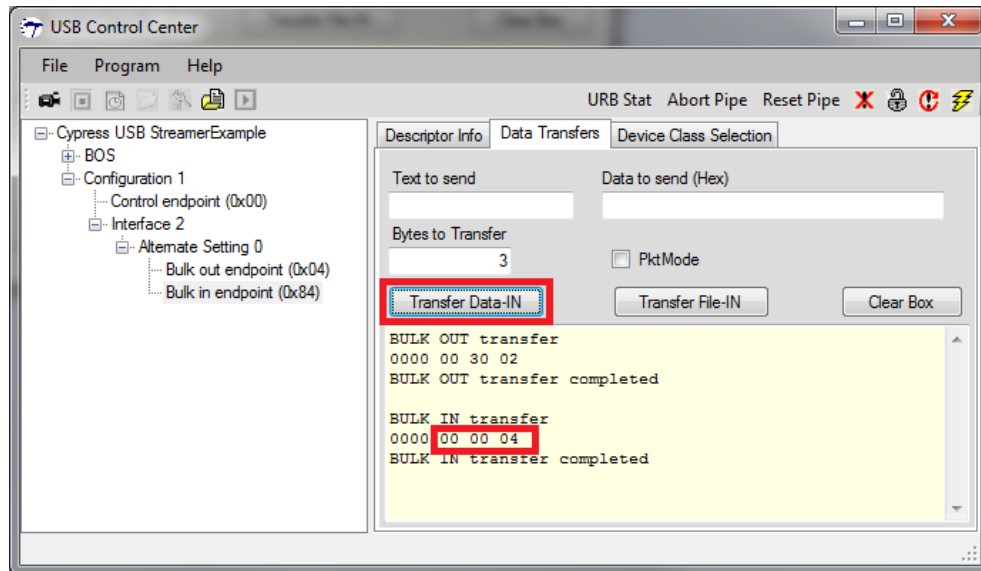
2. 点击“Transfer Data-Out”来发送指令。



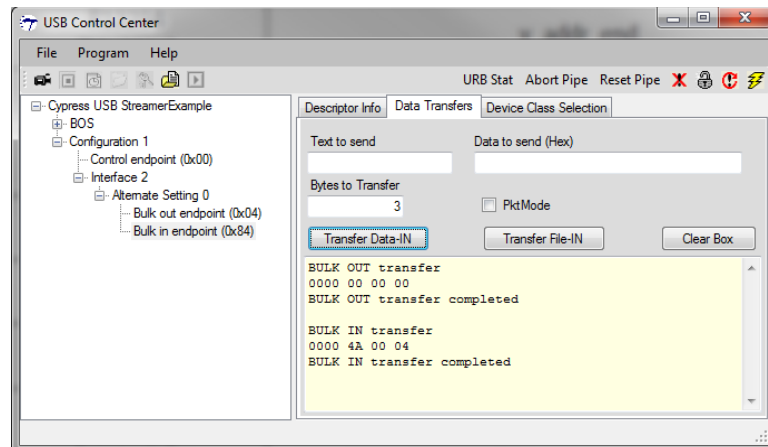
3. 选中响应端点并将“Bytes to Transfer”字段设置为“3”，这样可以读取单一读取指令的响应。



4. 点击“Transfer Data-IN”，接收响应

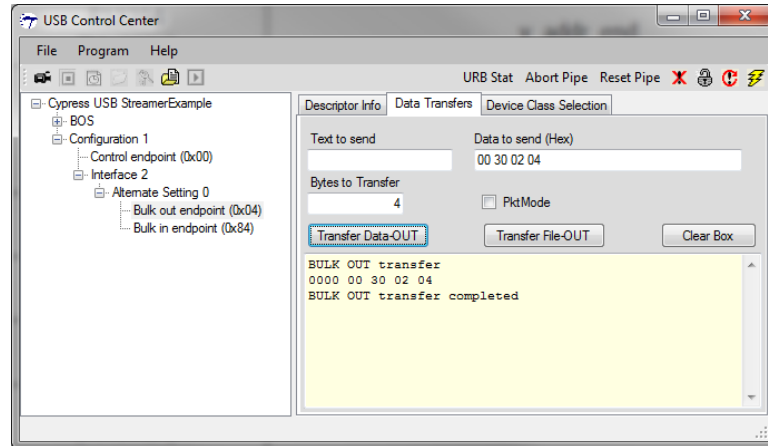


5. 响应的第一字节表示状态。状态为 0 时，表示指令已经通过；如果为其它值，则表示指令失败。后面各字节表示读回的寄存器值为 0x0004。该示例显示的是一个失败的传输，其中，状态字节为非 0 值，其余字节的值是前一个传输的过期值。

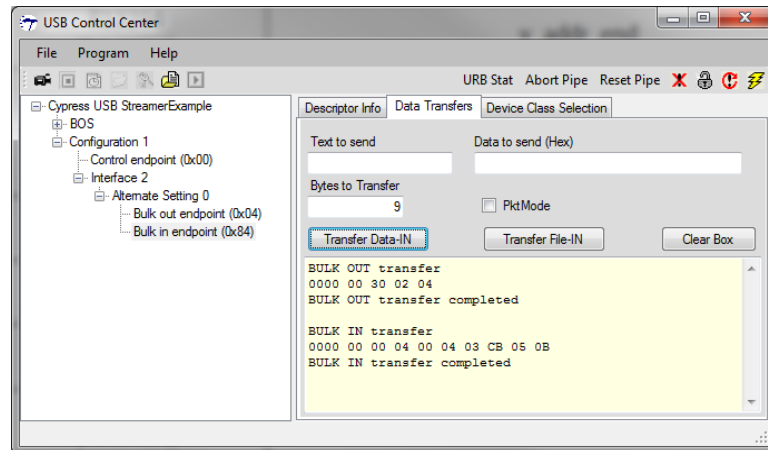


连续读取：

- 在“Data transfers”选项卡下，选中指令端点并以十六进制的格式输入指令。连续读取的指令格式为 0x00 <寄存器地址高字节> <寄存器地址低字节> <N>。下图显示的是对开始地址为 0x3002 的四个（N=4）寄存器所进行的读指令。

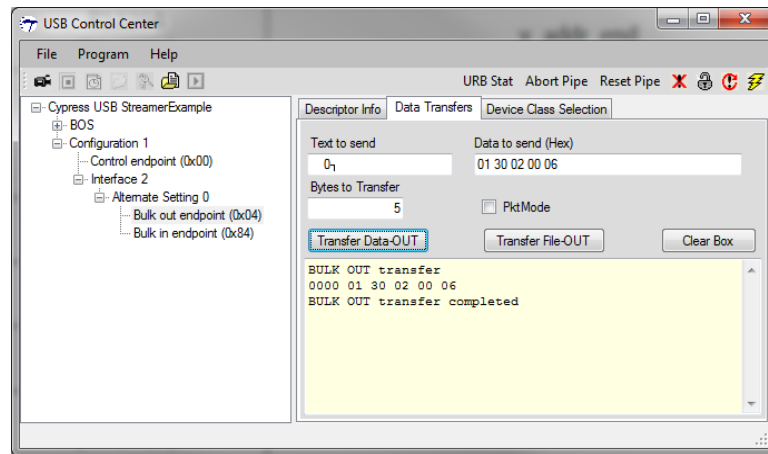


- 在这种情况下，响应的“传输字节”为 $(N*2+1) = 9$ 。下图显示的是 FX3 读取的值。

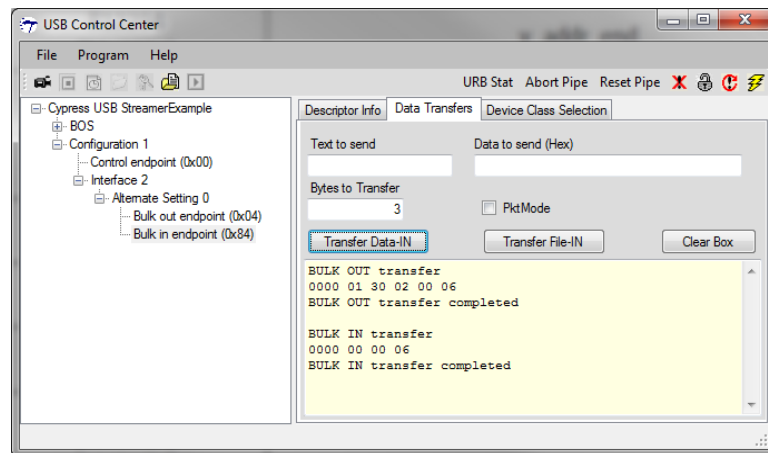


单一写入：

- 在“Data transfers”选项卡中，选中指令端点并以十六进制的格式输入指令。单一写入的指令格式为 0x01 <寄存器地址高字节> <寄存器地址低字节> <寄存器值高字节> <寄存器值低字节>。下图显示的是向寄存器地址为 0x3002 的空间内写入 0x0006 值的写指令。

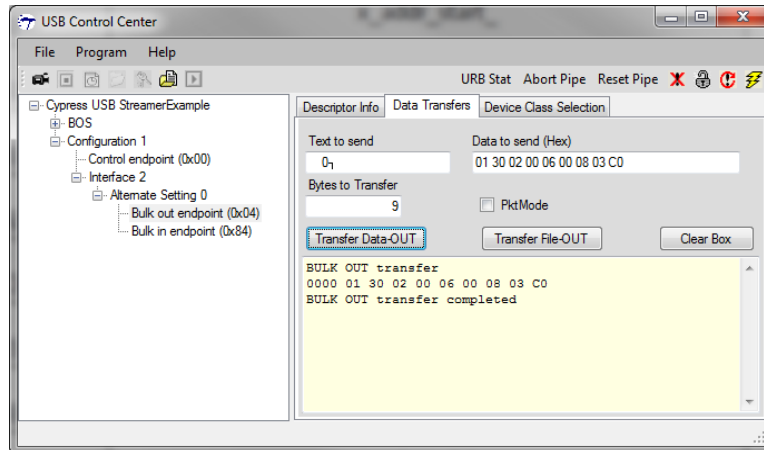


2. 单一写入的响应包含三个字节：<状态> <寄存器值高字节> <寄存器值低字节>。执行写入后将读取这些寄存器的值，您可在所发送的指令看到同样的值。

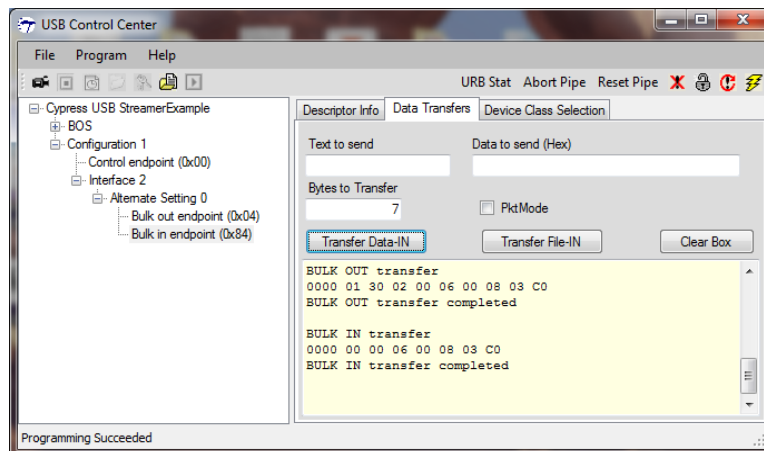


连续写入：

1. 在“Data transfers”选项卡下，选中指令端点并以十六进制的格式输入指令。连续写入的指令格式为 0x01 <寄存器地址高字节> <寄存器地址低字节> （（<寄存器值高字节> <寄存器值低字节>）* N 次数），以对寄存器进行 N 写操作次数。下图显示的是分别将 0x0006、0x0008 和 0x03C0 值连续写入寄存器 0x3002、0x3004 和 0x3006 （N=3）内。



2. 响应格式为<状态>（<寄存器值高字节> <寄存器值低字节>）* N 值。
例如，传输的总字节数为 $(2*N+1) = 7$ 。



6 硬件设置

已经在一组设备上测试了当前项目。该组设备包含了 SuperSpeed Explorer 套件 (CYUSB3KIT-003)、安森半导体图像感应互联电路板 (CYUSB3ACC-004A) 以及一个安森半导体图像感应电路板 (MT9M114EBLSTCH3-GEVB / MT9M114_55CSP_HB_DEMO3_REV0)。有关如何获取这些组件, 请参阅下面总结:

6.1 硬件要求

1. 安森半导体 MT9M114 图像感应电路板 (MT9M114EBLSTCH3-GEVB/MT9M114_55CSP_HB_DEMO3_REV0) (从安森半导体分销商购买)
2. SuperSpeed Explorer 套件 (CYUSB3KIT-003)¹。
3. 用于 SuperSpeed Explorer 套件的 CYUSB3ACC-004A² (用于安森半导体图像感应的互联电路板)。
4. 使用 USB 3.0 主机使能的电脑进行评估 SuperSpeed 的性能。

6.2 SuperSpeed Explorer 套件电路板安装包

按照下面各步骤准备用于视频应用测试的电路板:

1. 在 SuperSpeed Explorer 套件 (CYUSB3KIT-003) 和安森半导体图像感应电路板 (MT9M114_55CSP_HB_DEMO3_REV0) 上进行跳线设定, 如表 9 和表 10 所示。

表 9. 跳线设定— 安森半导体图像感应电路板 (MT9M114_55CSP_HB_DEMO3_REV0)

跳线/插头编号	跳线/插头名称	引脚设置	说明
P1	OE_L	1~2	使能并行接口
P2	CONFIG	2-3	设置为一般模式
P4	闪存	开放	连接至外部闪存
P5	TRST	1~2	设置为一般模式
P6	SADDR	1~2	从设备地址: 0x90
P7	+VDDIO	2~3	感应工作电压为 2.8 V
P8	UART	开放	关闭 UART (三态)
P11	EEPROM	关闭 (A1 列)	低电平有效 (A1)
P15	I2C 连接器	1~2 和 3~4	连接了主设备和感应器 I2C
P32	时钟选择	3~5 和 1~2	多镜头、主模式; 板载振荡器

表 10. SuperSpeed Explorer 套件 (CYUSB3KIT-003)。

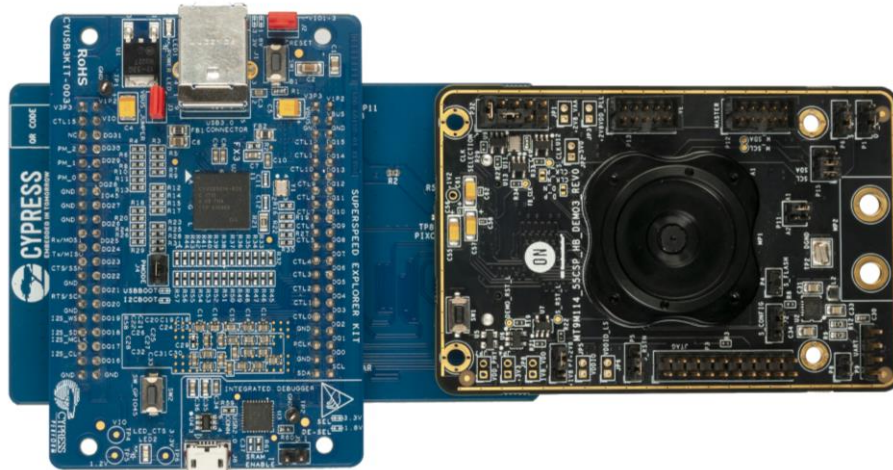
跳线/插头编号	跳线/插头名称	引脚设置	说明
J2	VIO1_3	关闭	VDDIO 电压选择 (2.8 V)
J3	VBUS_JUMPER	关闭	总线供电
J4	PMODE	关闭	USB 引导
J5	SRAM_ENABLE	开放	禁用板载 SRAM

¹ 更多有关使用 FX3 DVK (CYUSB3KIT-001) 的详细信息, 请参阅附录 A

² 更多有关使用带有 Aptina 图像感应的 Aptina™ 图像感应互联电路板 (CYUSB3ACC-004) 的详细信息, 请参阅附录 B

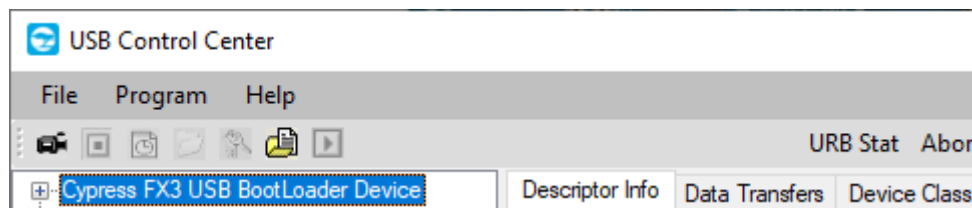
2. 组装 SuperSpeed Explorer 套件、互联板以及安森半导体图像感应电路板，如图 47 所示。

图 47. SuperSpeed Explorer 套件、安森半导体图像感应互联与感应电路板的组装



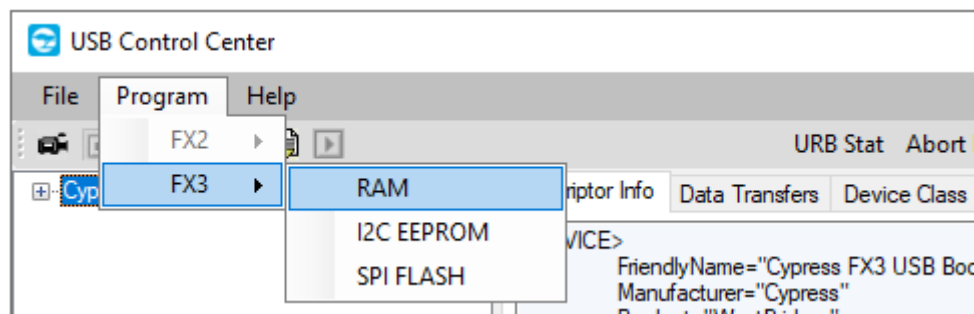
3. 使用 DVK 所提供的 USB 3.0 线缆将 FX3 DVK 电路板插到 USB 电脑上。
4. 使用 FX3 SDK 提供的控制中心应用将固件加载到电路板内。[AN75779.zip](#) 中提供了固件源和预建图像。详细内容，请查阅 [AN75705 — EZ-USB FX3 入门](#)。简要说明如下：
 - a. 启动 Control Center（控制中心）应用。当插到 FX3 Explorer 电路板时，它将被识别为 Cypress FX3 USB Bootloader 器件（图 48）。

图 48. FX3 枚举为 Bootloader



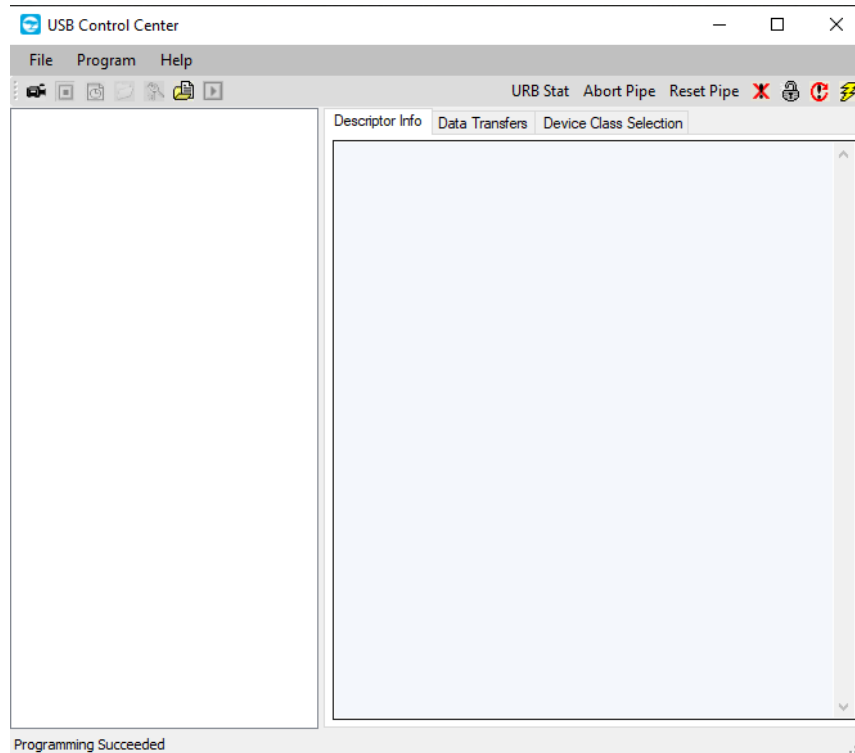
- b. 依次选择 **Program>FX3>RAM**，然后导航到本应用笔记所附带的 `cyfx_ufv_an75779.img` 文件（图 49）。请注意，如果在编程后进行设备关开机，该设备将丢失所加载的固件，并重新枚举为赛普拉斯 FX3 USB Bootloader 设备。

图 49. 将代码加载到 FX3 RAM 内



Control Center 应用将在状态栏上显示编程状态。成功编程设备后，将从 Control Center 设备列表中移除该设备，因为它将被枚举为一个 UVC 类设备。可以在 **Cameras or Imaging Devices**（镜头或图像设备）类型的 Windows Device Manager（Windows 设备管理器）下方查看该设备（图 50）。

图 50. 成功编程信息将显示在状态栏上



7 基于 UVC 的主机应用

各种主机应用允许您使用 UVC 器件来显示和捕捉视频。VLC 媒体播放器很受欢迎。另一种被广泛使用的 Windows 应用程序就是 [VirtualDub](#)（开源应用程序）。其他 Windows 应用分别为 [MPC-HCPlayer](#)（开源应用）和 [AMCap](#)（版本 8.0）和 [Debut Video Capture](#) 软件。

Linux 系统可以使用 V4L2 驱动程序和 VLC 媒体播放器进行视频流操作。可以在网上下载 VLC 媒体播放器。

Mac 平台可以使用 FaceTime、iChat、Photo Booth 和 Debut Video Capture 软件创建与 UVC 器件相连的接口，以便执行视频流操作。

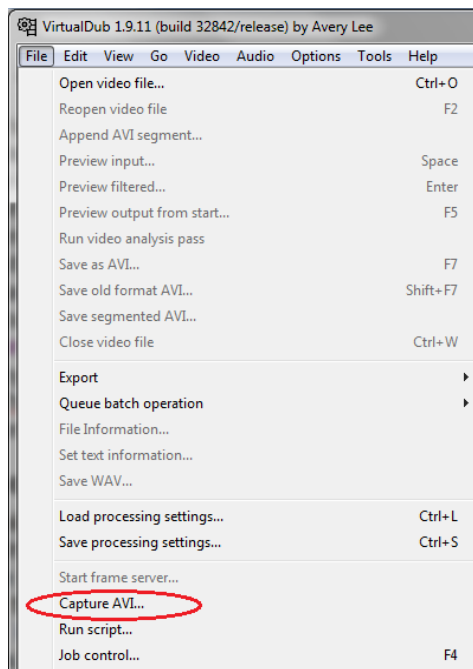
注意：使用 [VirtualDub](#) 和 [MPC-HC](#) 类似的开源应用程序来设计您的主机应用程序。[AmCap 示例](#)也在 Windows SDK 上可用。

7.1 运行演示

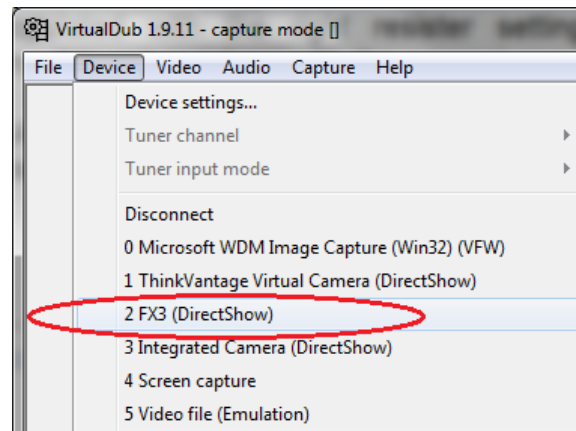
在 [AN75779.zip](#) 中提供了一个用于 FX3 Explorer 套件上演示目的的预编译代码图像文件。

按照下面各步骤运行该代码：

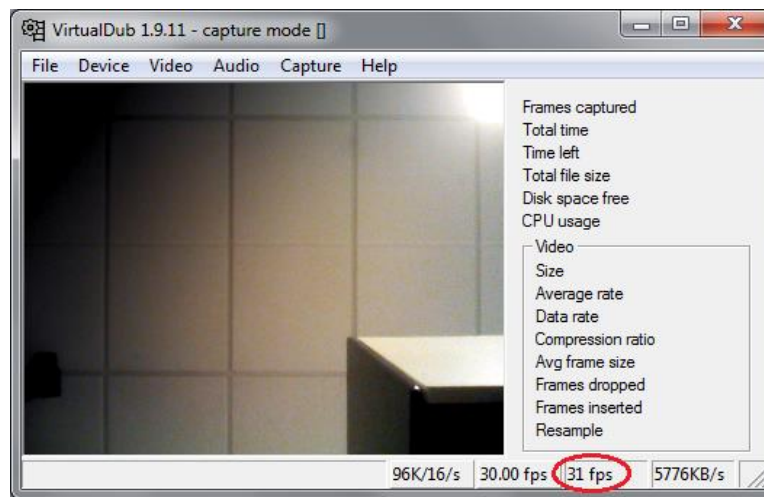
1. 将预编译固件图像加载到 FX3 UVC 安装包内，如第 6.2 节所述。
2. 这时，安装包将被重新枚举为 UVC 器件。操作系统将安装 UVC 驱动程序；此外不要求任何其它驱动程序。
3. 打开主机应用（例如，VirtualDub）。
4. 选择 **File > Capture AVI**。



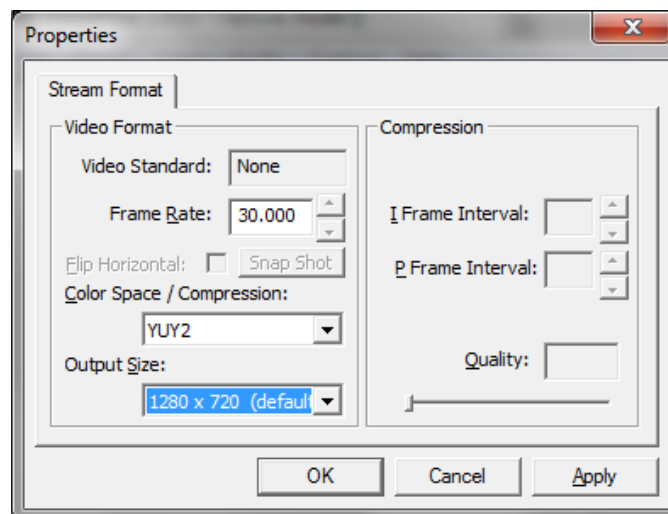
5. 选择 **Device > FX3 (Direct Show)**，这样会开始串流图像。



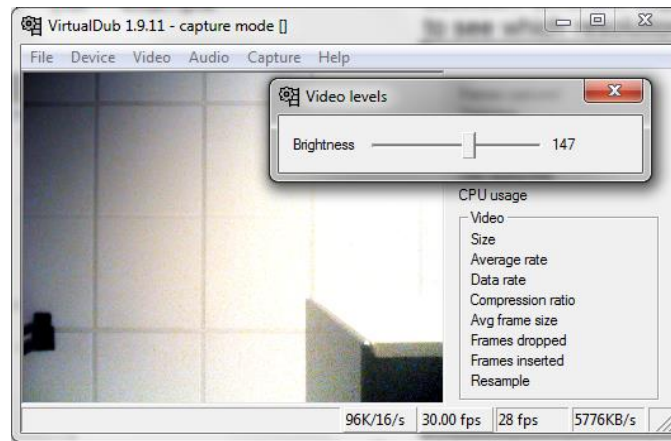
6. 右下方显示的是实际的帧率



7. 您也可以选择 **Video > Capture Pin**，可以在受支持的分辨率中选择当前有效的分辨率。



8. 您也可以选择 **Video > Levels** 可修改亮度（通过修改滑条位置来修改数值）或其它受支持的控制指令。选择 **Video > Capture Filter**，可找到其它控制指令。



注意：要想启用视频控制功能，如曝光度、清晰度，等等，需要与安森半导体签订保密协议（NDA）。要想获取帮助，以从 FX3 固件启用这些附加控制，请带上已执行的 NDA 并联系[赛普拉斯技术支持](#)。

8 故障排除

如果在主机应用中出现黑屏，请按照下面各步骤进行调试：

1. 确保使用固件的 **Release** 构建，因为它的性能是优化的。
2. 在 `uvc.h` 文件中存在“`DEBUG_PRINT_FRAME_COUNT`” switch 语句。使能开关，以查看 FX3 是否有串流图像。该语句用于使能 UART 打印帧数量。在 **SuperSpeed Explorer** 套件上，通过板上集成调试器可以始终使用 UART。打开能够访问 PC 上 COM 端口的 **Hyperterminal**、**Tera Term** 或其它工具。
启动传输前，请按照下面各项内容对 UART 进行配置：**115200 波特、无奇偶校验位、1 个停止位、无流控以及 8 个数据位**。这样应足以捕获调试打印。如果您在 PC 终端程序中没找到递增帧计数器，则 FX3 和图像传感器（GPIF 或传感器控制/初始化）之间的接口可能出现问题。
3. 如果您在 PC 端编程中看到了递增帧计数器的打印，需要验证被发送出去的图像数据。USB 走线可以显示每帧传输的数据量。
4. 为了检查每帧所传输的数据总量，需要查找具有标头上设置的结束帧位的数据包。（对于结束帧传输，标头的第二字节为 `0x8E` 或 `0x8F`）。一个帧上所传输的图像数据（不包含 UVC 标头）的总量应为：**宽度 × 高度 × 2**。
5. 如果当前不是 USB 走线所传输的数据量，则图像传感器的设置或 GPIF 接口可能存在问题。
6. 如果图像数据的总量正确，并且主机应用程序仍未显示任何图像，请更换电脑主机。
7. 如果问题仍然发生，请参考[赛普拉斯开发工程师社区](#)中相同的案例，或联系[赛普拉斯技术支持](#)。

9 连接两个图像传感器

主机应用程序（如：3-D 图像或移动跟踪）需要 FX3 同时传输两个图像传感器中的视频数据。如果传感器是不同的，则必须在图像传感器模块和 FX3 之间插入 FPGA，以便调整格式，将其并合成单一的视频数据通道。具有两个不同的图像传感器的设置超出了本应用手册的范围。

一种常用的方法是使用相同的图像传感器。在本节中详细介绍了这种方法。

图 51 显示的是连接的详细信息。绿色模块在 GPIF II 里面，红色模块属于 FX3 低带宽外设（I²C/GPIO）的一部分。需要同步化两个传感器，以便获得相同的帧时序，并使 GPIF II 在 16 位数据总线上同步输入每个 8 位数据流。

图 51. 连接到 FX3 的两个相同的图像传感器

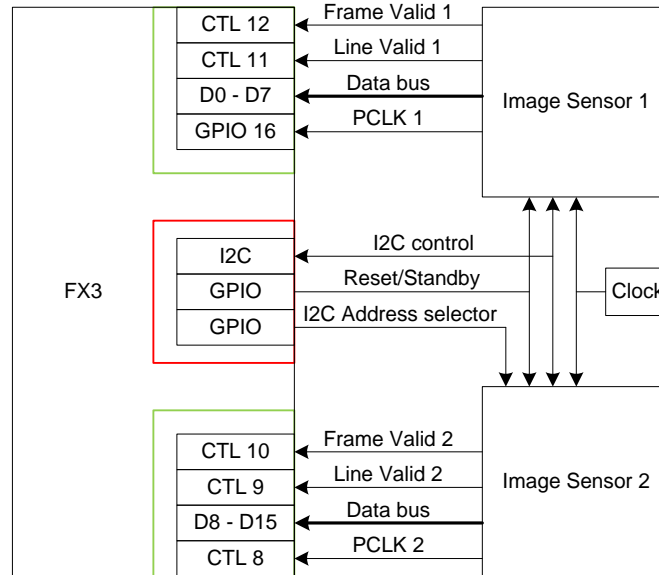


图 51 中假设两个图像传感器的情况如下：

- 每个图像传感器的总线宽度均为 8 位，因此 GPIF II 总线宽度为 16 位。
- 两个图像传感器是同步的。因此，这两个图像传感器将使用相同的时钟、LV 与 FV 转换和像素时序。也就是说可以对这两个图像传感器的帧进行准确的覆盖。某些图像传感器具有外部触发输入，可以用来同步两个视频流。其他图像传感器可以使用不同的同步方法。可应用的传感器数据手册对其进行了详细介绍。
- 这两个图像传感器都使用 I²C 进行配置。应用手册中使用的图像传感器需要 I²C 控制寄存器在完全相同的时间内进行写操作，以便实现各图像传感器模块之间的同步。通过使用 FX3 GPIO 引脚来控制一个传感器中的 I²C 地址可以完成该操作。FX3 使用 I²C 写操作来同时配置两个图像传感器。通过使用可配置的地址引脚切换到图像传感器上的不同 I²C 地址，FX3 可以读取各个传感器。
- 应通过相同的 FX3 GPIO 输出引脚来驱动每个传感器的复位信号。同样，每个传感器的待机引脚（如果存在的话）应共享另一个 FX3 GPIO 输出引脚。
- 图像传感器的自动配置被禁用。例如，在两种传感器中自动曝光，自动增益和自动白平衡等特性都被关闭。关闭它们可以确保在图像传感器上进行集成和处理任何图像同步进行。这样可使两个传感器的帧同时从图像传感器输出。

如连接图所示，将帧有效 2、行有效 2 和 PCLK 2 信号连接到 FX3，但是 GPIF II 模块不使用它们，因为图像传感器被假设是同步的。这些信号被连接到 FX3，因此在调试和开发过程中，可以通过监控这些信号来检查各个图像传感器间的同步精度。

9.1 使用 UVC 传输交错图像

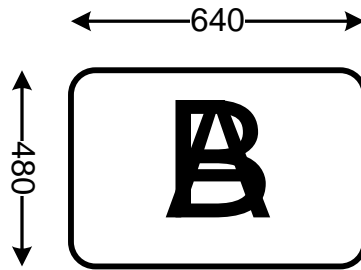
UVC 规范不能辨别所发送的图像中使用了多少图像传感器。它只能辨别一个图像参数组：帧的宽度和高度，以及每帧中总字节的数量。为了符合多个图像传感器的要求，UVC 驱动程序必须读取已修改的描述符，以便执行并通过内部一致检查。如果这些一致性检查失败，UVC 驱动程序不能将图像数据传送到主机应用，导致应用失败。需要修改描述符，以便 UVC 驱动程序将额外帧识别为单一图像传感器。

下面两个示例说明了如何实现该操作。

9.1.1 示例 1：两个 640x480 的单色传感器。

两个图像传感器可提供 640 x 480 单色（每像素 1 字节）数据。FX3 在每帧中同时接收到两个完整的图像，如图 52 所示。

图 52. FX3 从两个图像传感器所接收的数据



描述符继续报告 640 x 480 图像的尺寸。通过将 A 图像放置在 Y 数据中，并将 B 图像放置在 U 和 V 数据中，这样可以调节双倍数据的尺寸。

可通过下面公式计算每帧的字节数：

$$\text{每帧的字节数} = \text{每像素的字节数} \times \text{图像传感器数} \times \text{分辨率}$$

其中：

$$\text{分辨率} = \text{宽度 (像素)} \times \text{高度 (像素)}$$

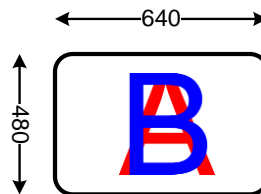
例如：

$$\text{每帧的字节数} = 1 \times 2 \times 640 \times 480 = 614,400 \text{ 字节}$$

9.1.2 示例 2：两个 640 x 480 的彩色传感器

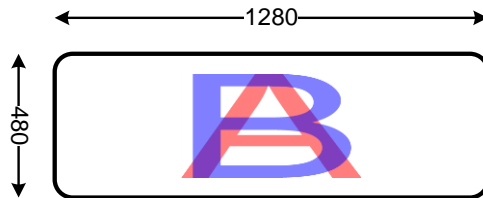
采取两个彩色传感器发送分辨率为 640 x 480 的 YUY2 数据为例。FX3 在每帧中接收到两个完整的彩色图像（图 53）。

图 53. FX3 接收到两个彩色图像



与示例 1 唯一不同点是 YUY2 格式中的每像素使用了两个字节，而不是一个字节。为了适应双倍像素，报告的图像尺寸也为双倍（图 54）。

图 54. 报告的图像尺寸



由于现在每个像素占用了两个字节，因此每帧的数据长度为：

每帧的字节数 = $2 \times 2 \times 640 \times 480 = 1,228,800$ 字节

请注意，只要反映的是双倍像素，便可以报告任意宽度和高度的帧。只双倍一个尺寸将简化下行计算。与高度相比，双倍宽度可以覆盖所有垂直线，从而简化应用的图像处理。

这些描述符的修改允许以交错方式将图像从图像传感器传送到主机应用：任何两个连续字节均来自不同的图像传感器。

上述修改已经通过 UVC 驱动程序的一致性检查，因此允许驱动程序将视频数据传送到主机应用。当直接查看它们时，按照这种方式进行的数据流并不清晰。您可以使用标准的 UVC 主机应用执行完整性检查。但是，由于图像以非标准的方式进行串流，因此应用将不能正确地显示它们。需要编译一个自定义应用，以便区分这些图像，并查看和计算交错视频的有益信息。

9.2 将新的视频分辨率添加到当前项目的固件修改检查表

总之，当将新的分辨率添加到给定示例的固件时，请按照以下步骤进行操作，以确保所有必需的更改已经完成：

1. 通过将新的帧描述符添加到可用的格式描述符内来更新高速和超高速 USB 配置描述符。新的帧描述符应反映新的视频分辨率的以下内容：帧索引，分辨率宽度，分辨率高度，帧率，最大比特率，最小比特率和宽高比。更新高速和超高速 USB 配置描述符长度和视频流输入标头描述符长度，以包含支持的所有视频帧描述符。
2. 为添加的视频帧分辨率添加 PROBE/COMMIT 控制结构。处理新添加的视频帧分辨率的 SET_CUR 和 GET_CUR 请求。
3. 添加传感器控制命令来设置新添加的视频帧分辨率。

9.3 支持新传感器的固件修改检查表

总之，当修改给定示例的固件以支持新的传感器时，请按照以下步骤进行操作，以确保所有必需的更改已经完成：

1. 图像传感器控制：示例将参考 I2C 代码作为控制接口，以将各命令传送到图像传感器。可能需要修改或要求另一种新图像传感器支持的接口。
2. 其它代码，用于控制作为控制接口的图像传感器选择的 GPIO。当 FX3 对本应用笔记中所使用的传感器进行写操作时，该代码将两个图像传感器作为多播信息。但是，当 FX3 读取图像传感器中 I2C 寄存器时，需要单独访问。
3. 其它代码，用于监控控制图像传感器的待机模式或低功耗模式的 GPIO。
4. 更改用于帧和格式的 UVC 特定的高速和超高速 USB 配置描述符。该操作用于报告受支持的所有视频和帧分辨率的索引格式，分辨率宽度，分辨率高度，帧率，最大比特率，最小比特率和宽高比。
5. 更新高速和超高速 USB 配置描述符长度和视频流输入标头描述符长度，以包含所添加的所有视频帧描述符。
6. 为受支持的每个视频帧分辨率添加 PROBE/COMMIT 控制结构。处理所有视频帧分辨率的 SET_CUR 和 GET_CUR 请求。
7. 修改 GPIF II 描述符，以便增大总线宽度，并根据总线宽度更新相应的计数器限制。通过 GPIF II Designer 实现上述修改，并生成头文件。需要将新生成的文件复制到您的项目中，以确保这些修改生效。

10 总结

本应用笔记描述了如何使用赛普拉斯的 EZ-USB FX3 实现符合 USB 视频类别的图像传感器。尤其重要的是，它显示了：

- 主机应用和驱动程序如何与 UVC 器件交互作用
- UVC 器件如何管理 UVC 特定的请求
- 如何使用 GPIF II Designer 编程 FX3 接口，以便接收到典型的图像传感器的数据
- 如何显示视频流并修改主机应用中的摄像机属性
- 如何向 UVC 器件增加用于调试的 USB 接口
- 如何查找各种平台上的主机应用，包括开源主机应用项目
- 如何使用 UVC 连接多个图像传感器、外部同步它们，并同步流操作
- 如何排除故障并调试 FX3 固件（若需要）。

关于作者

名称: Karnik Shah
职务: 应用工程师

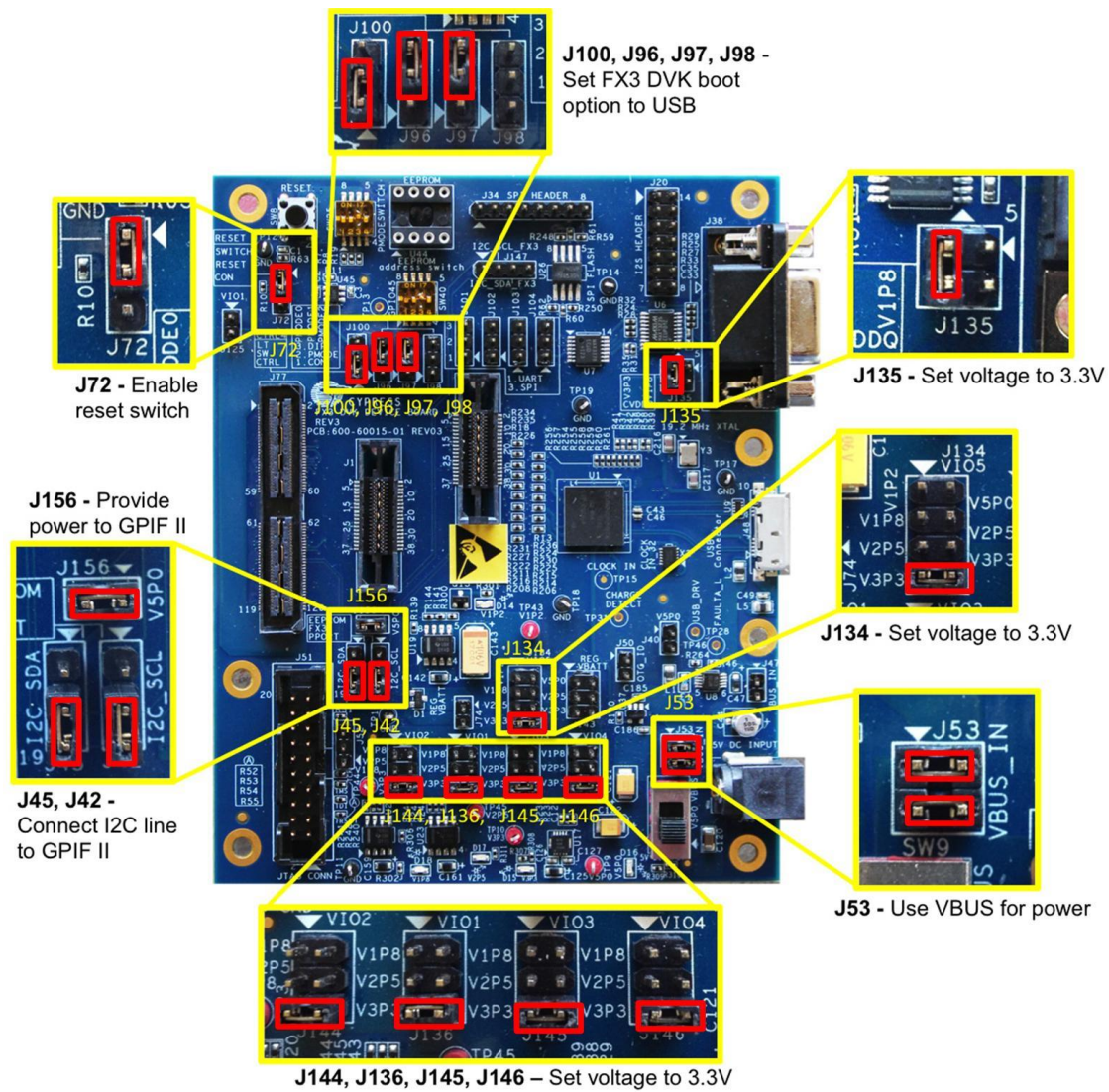
附录A. FX3 DVK (CYUSB3KIT-001)的硬件安装详细内容

A.1 FX3 DVK 板安装

按照下面各步骤准备用于视频应用测试的电路板：

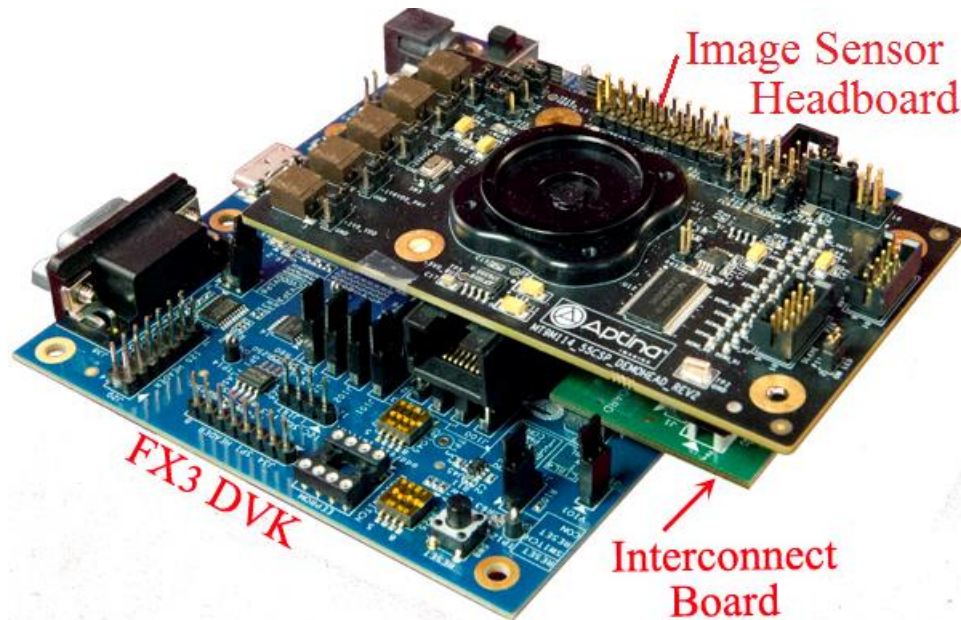
1. 按照图 55 中所示的内容配置 FX3 DVK 板上的跳线器。请勿加载图中高亮显示以外的跳线器。

图 55. FX3 DVK 跳线器



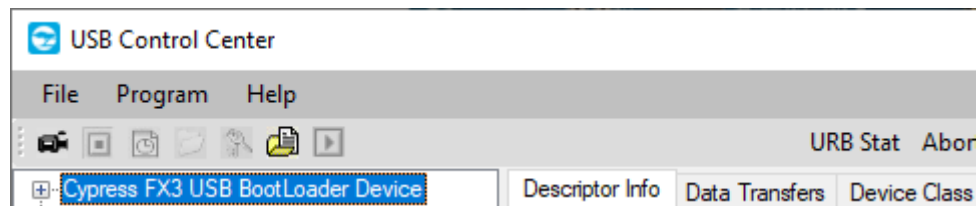
2. 将 Aptina 图像感应互联电路板 (CYUSB3ACC-004) 插入 FX3 DVK J77。互联板上的连接器类型是唯一的，另外套接字是有方向性的，所以只有正确地放置了它们的方向才算连接成功。
3. 将 Aptina 图像传感器模块连接到互联板。图 56 显示的三个电路板的装配。

图 56. 三个电路板的 720P 照相机装配



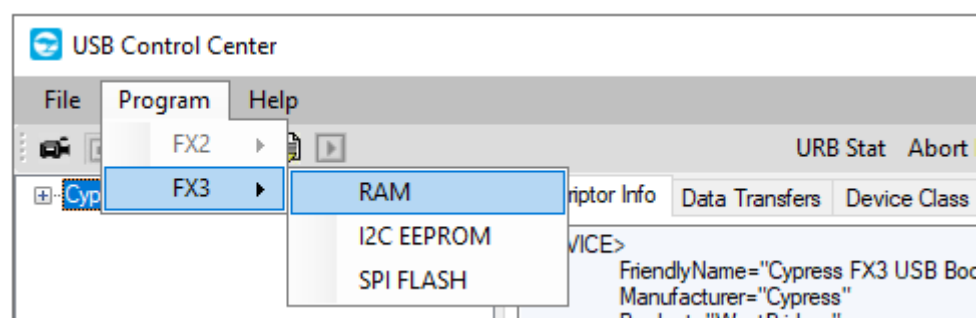
4. 使用了 DVK 提供的 USB 3.0 线缆将 FX3 DVK 电路板插到 USB 端口电脑上。
5. 使用 FX3 SDK 提供的控制中心应用将固件加载到电路板内。AN75779.zip 中提供了固件源和预建图像。详细内容，请查阅 AN75705 — EZ-USB FX3 入门。简要说明如下：

图 57. FX3 枚举为 Bootloader



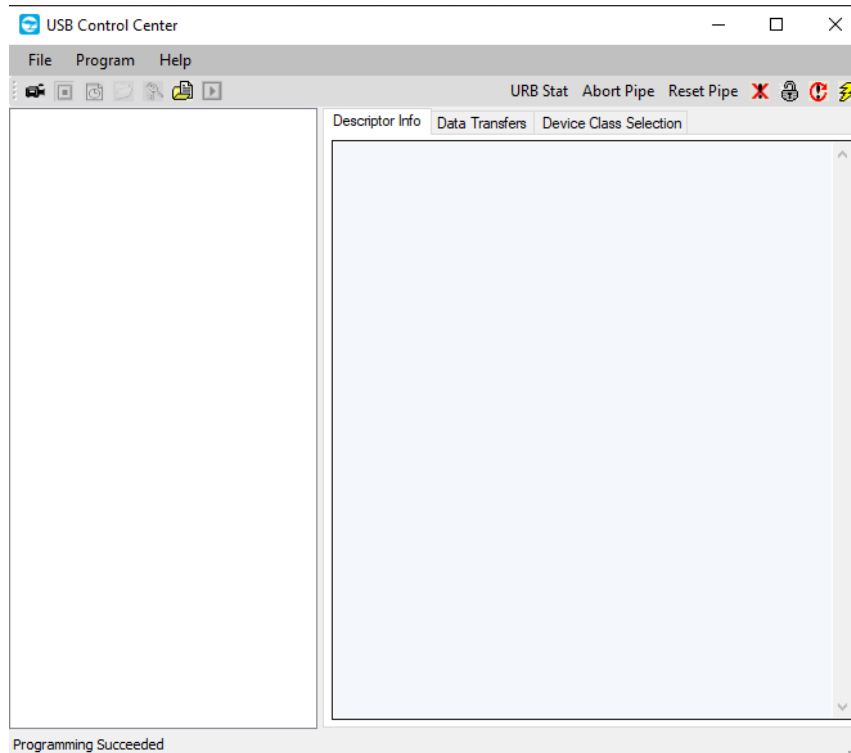
- a. 启动 Control Center（控制中心）应用。当插到 FX3 DVK 时，它将被识别为 Cypress FX3 USB Bootloader 器件（图 57）。
- b. 依次选择 **Program>FX3>RAM**，然后导航到本应用笔记所附带的 `cyfx_uvc_an75779.img` 文件（图 58）。请注意，如果在编程后进行设备关开机，该设备将丢失所加载的固件，并重新枚举为赛普拉斯 FX3 USB Bootloader 设备。

图 58. 将代码加载到 FX3 RAM 内



Control Center 应用将在状态栏上显示编程状态。成功编程设备后，将从 Control Center 设备列表中移除该设备，因为它将被枚举为一个 UVC 类设备。可以在 **Cameras or Imaging Devices**（镜头或图像设备）类型的 Windows Device Manager（Windows 设备管理器）下方查看该设备（图 59）。

图 59. 成功编程消息将显示在状态栏上

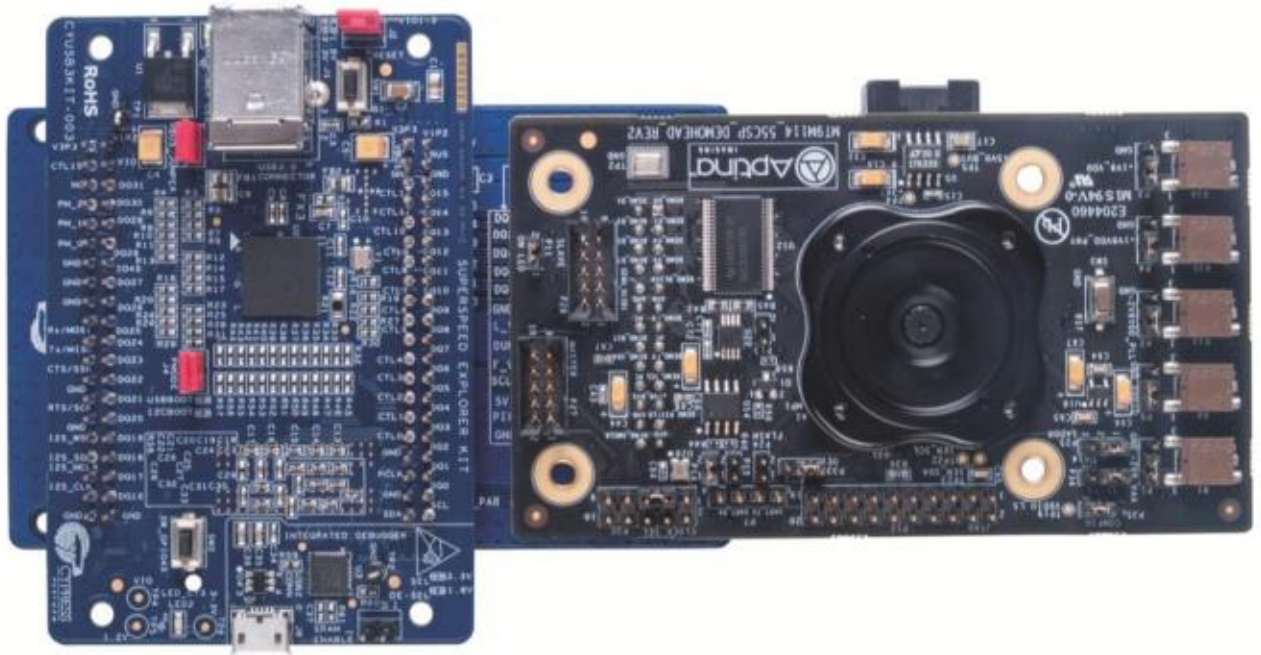


附录B. FX3 Explorer 套件和 Aptina 图像感应互联电路板（CYUSB3ACC-004）的硬件设置详细信息

B.1 硬件设置

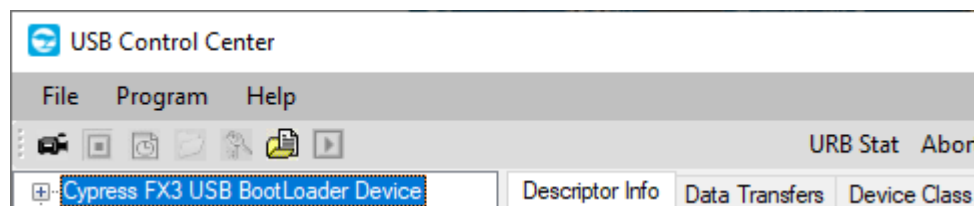
1. 在 FX3 Explorer 套件上关闭跳线器 J2、J2 和 J4，并打开跳线器 J5。
2. 根据 Aptina 互联板快速入门指南中的说明，将 SuperSpeed Explorer 套件、互联板和 Aptina 图像传感器板进行组装。
图 60 显示了组装说明。

图 60. SuperSpeed Explorer 套件、Aptina 互联板和 Aptina 传感器板的组装



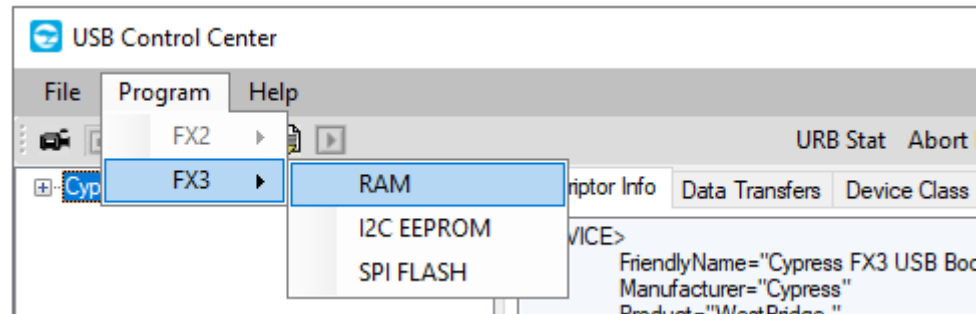
3. 使用了该套件提供的 USB 3.0 线缆将 FX3 Explorer 套件电路板插到 USB 端口电脑上。
4. 使用 FX3 SDK 提供的控制中心应用将固件加载到电路板上。[AN75779.zip](#) 中提供了固件源和预建图像。详细内容，请查阅 [AN75705 — EZ-USB FX3 入门](#)。简要说明如下：
 - a. 启动 Control Center（控制中心）应用。当插到 FX3 Explorer 电路板时，它将被识别为 Cypress FX3 USB Bootloader 器件（图 61）。

图 61. FX3 枚举为 Bootloader



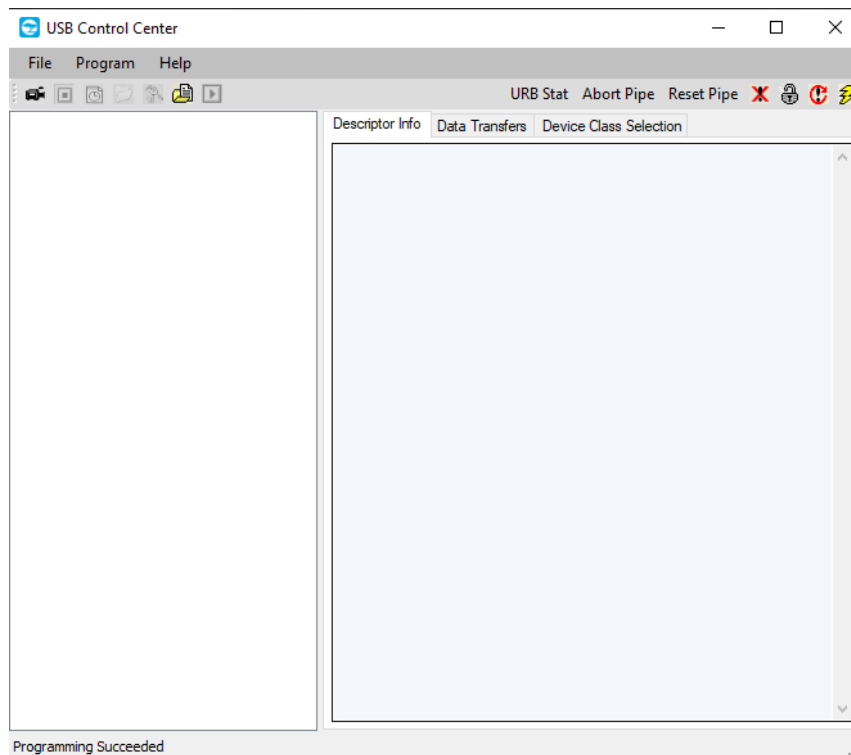
- b. 依次选择 **Program>FX3>RAM**，然后导航到本应用笔记所附带的 `cyfx_ufv_an75779.img` 文件（图 62）。请留意，如果在编程后进行设备关开机，该设备将丢失加载的固件，并重新枚举为赛普拉斯 FX3 USB Bootloader 设备。

图 62. 将代码加载到 FX3 RAM 内



Control Center 应用将在状态栏上显示编程状态。成功编程设备后，将从 Control Center 设备列表中移除该设备，因为它将被枚举为一个 UVC 类设备。可以在 **Cameras or Imaging Devices**（镜头或图像设备）类型的 Windows Device Manager（Windows 设备管理器）项下查看该设备（图 63）。

图 63. 成功编程消息将显示在状态栏上



文档修订记录

文档标题：AN75779 — 如何使用 EZ-USB® FX3™ 将图像传感器连接到 USB 视频类别（UVC）框架内

文档编号：001-92220

修订版	ECN	提交日期	变更说明
**	4354623	04/21/2014	本文档版本号为 Rev**, 译自英文版 001-75779 Rev*D。
*A	5708781	04/28/2017	更新公司标志和版权
*B	6002443	12/27/2017	本文档版本号为 Rev*B, 译自英文版 001-75779 Rev*J。
*C	6743952	12/05/2019	本文档版本号为 Rev*C, 译自英文版 001-75779 Rev*K。

全球销售和 design 支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要想查找离您最近的办事处，请访问 [赛普拉斯所在地](#)。

产品

Arm® Cortex® 微控制器	cypress.com/arm
汽车级产品	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
物联网	cypress.com/iot
存储器	cypress.com/memory
微控制器	cypress.com/mcu
PSoC	cypress.com/psoc
电源管理 IC	cypress.com/pmuc
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线连接	cypress.com/wireless

PSoC® 解决方案

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

赛普拉斯开发者社区

[社区](#) | [项目](#) | [视频](#) | [博客](#) | [培训](#) | [组件](#)

技术支持

cypress.com/support

此处引用的所有其它商标或注册商标都归其各自所有者所有。



赛普拉斯半导体公司
198 Champion Court
San Jose, CA 95134-1709

© 赛普拉斯半导体公司，2012-2019 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约归赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件没有附带许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方适用于个人的、非独占性、不可转让的许可（无转授许可权）（1）在版权保护下的软件（a）以源代码形式提供的软件，只能是在组织内部为了使用赛普拉斯的硬件去修改和复制。（b）以二进制代码形式从外部发到终端用户（直接或间接通过经销商和分销商），仅用于赛普拉斯硬件产品单元。（2）在软件（由赛普拉斯公司提供，且未经修改）侵犯赛普拉斯专利的权利主张下，仅许可在赛普拉斯硬件产品上制造、使用、提供和导入软件。禁止对软件的任何其他使用、复制、修改、翻译或编译。

赛普拉斯不对此材料提供任何类型的明示或暗示保证，包括但不限于针对特定用途的适销性和适用性的暗示保证。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权的访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的范围内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿并保护赛普拉斯免受所有索赔的损害，包括因人身伤害或死亡引起的索赔、费用、损失和其它责任。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。