AN4053

在 EZ-USB® FX2™和 FX2LP™上测试同步或批量端点的数据流

作者： Rama Sai Krishna Vakkantula

相关项目：有

相关器件系列：CY7C68013A

软件版本：无

相关应用笔记：AN65209、AN61345

要想获得本应用笔记的最新版本或相关项目文件，请访问 http://www.cypress.com/go/AN4053。

使用音频、视频等数据类型的 USB 应用程序需要一个高速的连续数据流。由于该类数据无中断被传输，因此将其定义为数据流。本应用笔记介绍了支持流数据的 USB 高带宽传输机制，并且含有用于实现和运用 FX2LP 高带宽端点的代码。它还提供了一个 PC 端应用程序，用于选择不同的传输类型和测量传输速率。本应用笔记还提供了用于演示如何使用赛普拉斯 USB 框架实现备用 USB 设置以允许主机选择不同的传输率的示例代码。要想获取 USB Hi-Speed 代码示例的完整列表，请访问 http://www.cypress.com/?rID=101782。

目录

简介 1
相关项目文件 2
数据流应用 2
高带宽传输 3
　数据 PID 序列（高速模式） 3
Cypress SuiteUSB（赛普拉斯 SuiteUSB） 4
流固件示例 4
使用赛普拉斯 USB 框架 5
　fw.c 6
　dscr.a51 6
　CyStream.c 7
运行 CyStreamer.hex 8
性能分析 9
将 FX2LP 连接到图像传感器上 10
参考 10
总结 11
全球销售和设计支持 13

简介

USB 为流数据提供了被命名为“同步”（ISO）的传输类型。在拉丁语中，*iso* 意为相等，*chron* 意为时间：传送流数据的数据包时每个 USB 时间帧需要花费相同的时间。

USB 同步传输类型具有两个重要特点：

1. 与控制（CONTROL）、BULK（批量）或 INTERRUPT（中断）传输类型不同，同步传输类型不为受损或丢失的数据提供握手或重新尝试服务。此开销会影响到每个时间帧架中的核心 ISO 特性，即保证数据传输。
2. 虽然 USB ISO 传输机制提供了保证基础上的数据块，但是数据提供者和消费者需要快速填充和清空该块，保证不间断的数据流。

虽然不会重新发送受损或丢失 ISO 数据包，但接收方会*检测*它们的存在，从而采取纠正操作。例如，可以通过内插法或重复以前的数据来替换丢失的数据块。然而一个设计良好的应用程序应保证数据提供方（例如，照相机）和数据接收方（例如，PC）给数据流设计提供更高的优先级，以便不会错过加载或卸载数据包。

INTERRUPT（中断）端点也可以是 Streamer。

USB 规范，版本 2.0 为高速度模式（480 Mbits/sec）操作定义了高带宽端点。一个高带宽端点在一个微帧内可以提供或接受至少一个、最多 3 个数据包。因此，INTERRUPT 传输模式也可以使用于高速度设备中的流数据。在全速度器件中，INTERRUPT 传输并不适用于高带宽流应用，因为 INTERRUPT 端点的最大数据包大小为 64 字节，而 ISO 的最大数据包尺寸却为 1023 字节。

本应用笔记说明了如何编程 FX2LP，以用于高速 USB ISO 流数据。它还执行全速度传输示例进行对比。还可以使用更早期的 FX2（没有 LP）器件，其差别较小（在本文档中进行了介绍）。另外，还执行 BULK 传输操作，以便进行带宽比较。

相关项目文件

本应用手册所附带的 zip 文件（*AN4053.zip*）包括以下文件夹：

1. FX2LP Streamer 源代码

Keil 项目文件‘CYStream.Uv2’位于 `Firmware\CyStreamer` 文件夹内。双击该文件，启动通过执行 FX2LP 固件提供并使用连续数据流的 Keil 项目。Keil IDE 将输出 *CyStream.hex* 文件，该文件适合被加载到 FX2LP 开发板上。

2. VS_Control_Center

该 Microsoft Visual Studio C#应用程序用于将 FX2LP 十六进制代码（*CyStream.hex*）加载到 FX2LP 开发板上。可执行文件位于`\bin` 目录中，或您可以使用 Microsoft Visual Studio 加载*.sln*（solution）文件来检查或修改源代码。

3. VS_Streamer

该 Microsoft Visual Studio C++项目是主机方应用程序，它通过运行 CyStream 代码将 PC 内的数据加载到 FX2LP 开发板上。该应用程序用于测试 FX2LP 流固件中的各种带宽设置。可执行文件位于`\bin` 目录中，或者您可以加载*.sln* 文件来检查或修改源代码。

也可以通过安装 Cypress SuiteUSB 获得该应用程序。该应用程序位于：`Cypress Suite USB 3.4.7\CyAPI\examples\Streamer`

通过安装 Cypress SuiteUSB，您也可以得到基于 C# streamer 的应用程序。C#版本位于下面路径：`Cypress Suite USB 3.4.7\CyUSB.NET\examples\Streamer`

下面各部分详细介绍了 ISO 流以及通过使用 FX2LP 执行 ISO 数据流时所需要的信息。

数据流应用

当您将 USB 器件插入一个 PC 上时，PC 将通过运行被称为枚举的流程来获取器件的基本特性以及要求。在枚举过程中，主机将通过询问该器件来获知器件详细信息（例如，器件所使用的驱动程序、器件编号、器件端点基本信息（数据源和数据库）等内容）。对于 ISO 端点，主机将读取描述符，从而确定器件在 ISO 传输过程中预期得到的内容。要求带宽是重要的 ISO 参数，它是 ISO 端点描述符中 *wMaxPacketSize* 字段。ISO 带宽限制如下：

- 对于全速传输，每 1 毫秒一帧中，主机可以处理一个 ISO 数据包（数据包大小最大为 1023 字节）。如果每毫秒传输 1023 字节，最大传输速率为每秒 1.023 Mb。
- 对于高速传输，在每 125 微秒一帧中，主机可以处理最多三个 ISO 数据包（数据包大小最大为 1024 字节）。如果每 1/8 毫秒传输 3072 字节，最大传输速率为每秒 24.576 Mb。

只有提供方和接收方将数据快速从端点 FIFO 内移入和移出以得到所需传输速率，才能无中断流动数据。为了满足这种要求，将名称为通用可编程接口（GPIF）的硬件数据移动器配备在 FX2/FX2LP 中。

USB 也提供了被称为备用设置的机制（Altenate Settings），用于允许主机选择流带宽。例如，USB 照相机可以提供备用设置，如表 1 所示。

表 1. 三个用于 USB 照相机的带宽设置

备用设置	wMaxPacketSize
1	1023
2	512
3	256

由于该 USB 相机不一定是总线上唯一的器件，因此在存在连接器件的情况下，PC 必须为它分配有限的 USB 带宽资源。

要注意，在该带宽协商过程中，主机会确定在保证的基础上为给 USB 照相机提供每帧传输 700 字节数据的带宽。如果照相机 ISO 端点描述符只包含一个 1023 字节的设置，则主机不会配置（使用）照相机，从而引起应用程序运行失败。然而，由于照相机还提供了其它设置选项，因此主机可以选择 2 号备用设置。主机（通过使用 SET_INTERFACE）向照相机发送一条信息（该信息包含主机要求的备用设置信息）。该信息告诉照相机要降低其带宽要求（可通过降低帧速率满足降低带宽要求）。

ISO 带宽备用设置以 1 开始，这是因为默认设置（即为‘0’）绝不允许占用任何 ISO 带宽。这些数据包大小为最大值；器件可能会通过所有数据包传输较少的字节数。零长度的 ISO 数据包也有含义。器件会发送该类型的数据包，以表明它不能提供数据来满足‘just-in-time’（实时）ISO IN 请求。

外设设计人员可能选择传接更大的数据包大小（比如一个 1024 字节数据包而不是四个 256 字节数据包），以获得不同的效率。通常，需要结合考虑外设中数据缓冲区的数量和大小来做出该决定。为了满足灵活性要求，USB 为高带宽端点定义了 *bInterval* 字段。通过该值，主机可以确定各传输请求之间跳过了多少微帧。对于大多高带宽应用（包括本应用笔记中所描述的应用）*bInterval* 值被置为 1，以表示‘每个微帧安排执行一个传输操作’。

虽然 BULK 端点不是高带宽端点，但如果两方拥有足够的缓冲空间用于处理 BULK 端点的突发（不定期）特性，仍可以使用 BULK 端点流动数据。不建议这样使用，是因为您的照相机可能在轻载 USB 系统上工作，但如果多个器件（如 USB 磁盘驱动程序）被插入，向照相机的 BULK 传输速率会变慢。

设计 USB 数据流应用时要注意下面几点：

- 传输吞吐量：每秒需要的字节数
- FX2LP 的速度是否足以转移使用已编程（MCU 参与）传输的数据，或是否要使用 GPIF 来传输 FIFO 数据？
- 系统的缓冲要求：按照规范，ISO 端点为双缓冲区，但 IC（如 FX2LP）最多可提供 4 个缓冲区（如果需要）
- 需要满足所需带宽和数据速率的合适端点类型
- 每微帧需要的数据包数量
- PC 驱动程序支持所选高带宽传输

高带宽传输

USB 将总线时间分为各个命名为帧的长度固定段。对于全速器件，主机每一毫秒将发送一个帧标志（SOF — 帧的开始）。对于高速器件，主机会每 125 微秒发送一个微帧。

每微帧需要多于 1024 个字节的高速端点被称为高带宽端点。高带宽端点每个微帧最多可传输三个含 1024 字节的数据包。每微帧传输次数由端点描述符的 *wMaxPacketSize* 字段定义。由于它们使用微帧，因此高带宽端点只适用于高速度模式操作。

根据 USB 2.0 规范，定期端点（ISO 或 Interrupt）必须指定它所需要的总线访问周期。该操作通过设置端点描述符的 *bInterval* 字段来完成。端点描述符的 *bInterval* 字段定义了主机对端点进行轮询的最大速率。这样可提供一个机制，用于降低主机服务端点的速率。

为了在同一个微帧中跟踪数据包传输，高带宽同步传输使用了一个名为 Packet ID（数据包 ID — PID）排序的机制。要想设计高带宽 I*SO* 端点，需要了解数据 PID 排序，如下面各部分的介绍。

数据 PID 序列（高速模式）

全速 USB 器件采用了两个数据 PID：DATA0 和 DATA1。USB 2.0 规范为高速度操作添加了另外两个 PID：MDATA 和 DATA2。PID 排序会检测一个微帧期间所发生的丢失或受损数据包。表 2 针对不同大小加载数据的 IN 和 OUT ISO 传输列出 PID 的顺序。

表 2. 高带宽 ISO 传输的 PID 序列

可用数据包	方向	数据 PID 序列		
		Pkt 1	Pkt 2	Pkt 3
3	IN	DATA2	DATA1	DATA0
2	IN	DATA1	DATA0	–
1	IN	DATA0	–	–
3	OUT	MDATA	MDATA	DATA2
2	OUT	MDATA	DATA1	–
1	OUT	DATA0	–	–

在 IN 传输中，进行枚举期间，主机预期收到的每微帧数据包多少是器件 *wMaxPacketSize* 字段所指定的每微帧数据包数量。器件向主机发送的第一个 DATA PID 通知器件在微帧发送的 IN 令牌数量：DATA2 表示两个以上令牌、DATA1 表示一个以上令牌，DATA0 表示没有令牌。如果器件没有数据，它将使用 DATA0 PID 发送零长度的数据包（ZLP）。

在 OUT 传输中，除了最后的数据包，主机对全部数据包使用 MDATA（More Data — 更多资料）PID；最后一个的数据包是 DATA PID，它表示主机所发送的 MDATA PID 数量。

FX2（非 LP）ISO IN 传输的注意事项

在枚举过程中，高速 ISO 器件将通知主机它在一个微帧中预期使用的数据包额外数量。器件通过端点描述符中的 *wMaxPacketSize* 字段来表示。位 12:11 表示在每个微帧中含有 0 个，1 个或 2 个额外数据包（或者 additional transaction）。在 USB 2.0 规范中，这些位被称为数据传输条件，这是因为它们表示器件在所有微帧内不需要使用的数据包最大数量。

FX2 负责发送正确的 DATA PID，具体情况取决于第一个主机 IN 令牌在一个微帧到达后它填充并且提交给的 USB 传输的实际 IN FIFO 缓冲区数量。代码示例通过 *wMaxPacketSize* 字段报告一个微帧中存在三个 IN 数据包（最大值）。然而，如果它在一个微帧只提交 2 个数据包，则必须使用 DATA1 PID（表 2）送第一个数据包。

为了自动执行正确的 ISO-IN PID 排序，FX2LP 提供了四个 EPnISOINPKTS 寄存器；其中，n 为 2、4、6 或 8，分别对应于大型、高带宽端点。

位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
AADJ	0	0	0	0	0	INPPF1	INPPF0
0	0	0	0	0	0	0	1

“IN Packets per Frame”（INPPF — 每个帧的 IN 数据包数量）默认为 1，可将它设置为 1、2 或 3。应将“Auto Adjust”（AADJ — 自动调整）设置为 1，以便使能自动 PID 排序。如果 AADJ = 0，FX2LP 会通过使用 INPFF[1:0] 指示的 PID 无条件启动一个 ISO IN 传输。

由于更早的 FX2 器件不具有这种自动机制（无 AADJ 位），所以编程员必须采取相应措施来确保在 ISO IN 传输过程中发送正确的 Data PID 排序。请按照下面任一方法执行该操作：

1. 确保在设计方面在每个微帧已加载和准备好所有 IN FIFO。例如，如果 INPPF 字段被设置为 2，则需要确保两个缓冲器被加载和提交给 USB 以便执行传输操作。
2. 如果在下一个微帧数据是较短的，请使能任何未用的 FIFO 来发送一个零长度的数据包（ZLP）。该操作很简单和快速；只需要将是‘0’值加载到 FIFO 的字节计数寄存器内即可。
3. 如果一个外部源加载 USB FIFO，则通过选择 FIFOADR[1:0]指定的 FIFO 并激活 PKTEND 引脚，可准备好任何未用的 IN FIFO 来发送一个 ZLP。使用 PKTEND 将部分填充（或空白）FIFO 提交给 USB 传输。

主机驱动程序可能会也可能不会正常处理 USB PID 排序错误。赛普拉斯提供了一个主机驱动程序，用以在器件发送一个错误 PID 排序的情况下通过将该指示传播到客户端应用解决问题这样，永远不会使用 FX2LP，因为当 AADJ = 1 时，将自动发送正确的 Data PID 排序。

通过同步 OUT 端点进行传输时，不需要设置任何寄存器。器件固件必须报告在端点描述符中每个微帧需要的数据包数量。

Cypress SuiteUSB（赛普拉斯 SuiteUSB）

用于写入和测试本应用笔记的代码的文件夹附加在 zip 文件中。为了提供更多的支持，赛普拉斯提供了 Cypress SuiteUSB，它包括用于 Visual Studio 的 USB 开发工具集以及示例 Keil 固件项目。Cypress SuiteUSB 可用于创建 .NET Windows 应用程序，以用于所有赛普拉斯 USB 2.0 系列。该套件包含 *cyusb.sys*，它是与开发工具相兼容的赛普拉斯 Windows 驱动程序。

可以通过下面网站下载 Cypress SuiteUSB：http://www.cypress.com/?rID=34870。Cypress SuiteUSB 的默认安装路径为：`C:\Cypress\Cypress Suite USB 3.4.7`。

流固件示例

本笔记所包含的应用（CyStream）通过使用 EZ-USB FX2/FX2LP 开发板演示了 USB 同步流和 BULK 传输。该代码示例执行了一个测试器件，该器件可为 Windows Streamer 应用提供并使用数据流。

CyStream 项目创建了一个单独的接口（编号 0）（该接口含有两个不同的配置描述符：一个用于高速（HS）操作，另一个用于全速（FS）操作）。表 3 显示了 7 个高速备用设置，表 4 显示了 4 个全速备用设置。

表 3. 用于接口 0 的 FX2LP 高速配置描述符备用设置

备用设置	端点数	端点数方向（类型）	数据包最大大小（字节）
0	1	2 IN（批量）	512
1	1	2 OUT（批量）	512
2	2	2 IN（批量）	512
		6 OUT（批量）	512
3	1	2 IN（同步）	3x1024
4	1	2 OUT（同步）	3x1024
5	1	2 IN（同步）	1x1024
6	2	2 IN（同步）	1x1024
		6 OUT（同步）	1x1024

表 4. 用于接口 0 的 FX2LP 全速配置描述符备用设置

备用设置	端点数	端点数方向（类型）	数据包最大大小（字节）
0	1	2 IN（批量）	64
1	1	2 OUT（批量）	64
2	1	2 IN（同步）	1023
3	1	2 OUT（同步）	1023

主机应用（Streamer.exe）会选择不同的传输参数（备用设置）来测量其吞吐量速率。要想测量可获得的最大吞吐量速率，请勿在 FX2LP 端发生任何延迟事件，例如将字节转入和转出端点 FIFO 所引起的延迟。因此，本应用笔记中的 FX2LP 代码将以最小开销处理 IN 和 OUT 传输：

- IN 传输 — 当一个 IN FIFO 可用时，8051 仅要重新待命 IN FIFO 来执行 1024 字节高速传输或 1023 字节全速传输。初始化期间将加载 FIFO 数据。
- OUT 传输 — 8051 不会将任何字节从 OUT FIFO 中输出；它只需要将任意字节数量加载到字节计数低（EPxBCL）寄存器内来重新准备执行下一个 OUT 传输。

注意：由于实时应用*确实*管理着 FIFO 数据，因此 FX2LP 提供了两个硬件机制用于最小化其传输开销。

1. 数据传输是由外部控制器处理的，该控制器直接与 FX2LP 端点 FIFO 相连。这样会使 8051 不在数据路径范围内。FIFO 连接包括一个 8 位或 16 位数据总线、读和写选通以及 FIFO 状态标志。
2. 可将 FX2LP FIFO 配置为自动运行模式。既为：端点 FIFO 中的字节数与寄存器中所设置的值相匹配后，数据包会自动准备进行 USB 传输。

本应用笔记中所介绍的示例使用了端点 FIFO 的手动模式设置，其中，8051 负责检查 FIFO 状态并提交端点数据。

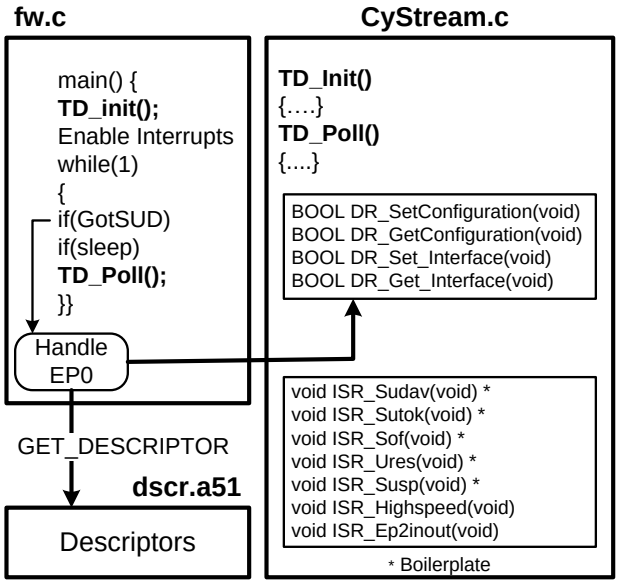
使用赛普拉斯 USB 框架

本应用笔记中包括的代码示例说明了如何设置 FX2LP 端点以用于流数据。本部分详细介绍了代码的主要程序。该示例包含以下源文件:

1. 赛普拉斯框架文件，*fw.c*
2. 描述符文件，*dscr.a51*
3. CyStream.c 文件

这些代码模块均是合适的，如下图所示。这种结构适用于所有基于赛普拉斯固件框架的 FX2LP USB 应用。

图 1. CyStream 代码模块



fw.c

这个赛普拉斯编写的固件（即 Frameworks）用于处理低级 USB 详细信息。该文件无需修改。*Fw.c* 包含了项目的 main()函数，启动时它将调用 **TD_Init()**，然后在操作过程中重复调用 **TD_Poll()**。该无限循环也会处理 CONTROL 端点（EP0）SETUP 数据包。对于 GET_DESCRIPTOR 请求，它使用 dscr.a51 中提供的描述符数据。对于其它主机请求，将调用您的应用来执行各种操作，例如修改某个接口的备用设置。该无限循环还会处理 USB 暂停和恢复事件（睡眠）。

dscr.a51

它是一个包含用于特定 USB 器件的描述符数据的 8051 汇编语言模块。该文件含有‘.db’（define byte — 定义字节）语句，按下面的顺序列出了描述符表格数据：

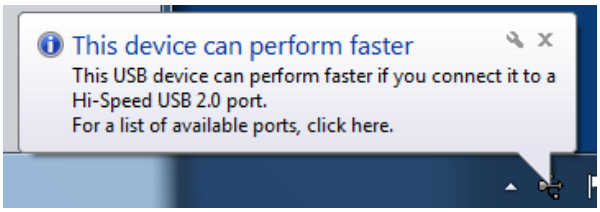
1. 器件描述符
   a. USB 规范版本（2.0）
   b. 器件标准类别（无）
   c. EP0 MaxPacketSize（64）
   d. 供应商 ID = 0x04B4 = 赛普拉斯
   e. 产品 ID = 0x1003 = “EZ-USB 示例 Streamer 器件”。
   f. 制造商和产品字符串索引
   g. 配置数量 = 1
2. 器件限定符：对于能够在全速或高速两种模式下运行的器件，该选项为‘其它’（非当前）速度提供设置。
3. 高速配置
   a. 接口数量 = 1
   b. 选择该配置的索引 = 1
   c. 七个接口-端点描述符配对，同表 3 中的七个备用设置参数相对应。
4. 全速配置
   a. 接口数量 = 1
   b. 选择该配置的索引 = 1
   c. 四个接口-端点描述符配对，同 表 4 中的四个备用设置参数相对应。
5. 两个字符串描述符。

全速/快速操作概况

当 USB 外设连接至主机时，主机上的内部 USB 集线器（或外部集线器，如果存在）将与外设进行协商，以确定其操作速度。实际上，主机从未*选择*一个工作速度，而它将使用集线器报道的速度。只要器件仍保持连接状态，集线器报道的工作速度便不发生变化。仅在器件断开并重新连接的情况下，才重新开始确定速度的过程。

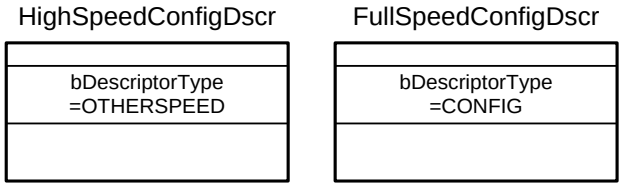
对于可在全速或高速模式下运行的器件，集线器报道的速度为‘当前’速度，而它*可*运行的速度是其他速度。由于拥有两个配置描述符（每一个用于每一工作速度），主机可以在器件插入不同端口的情况下确定器件支持其它速度。因此，如果您将高速器件插入到全速端口上，Windows 将显示图 2 中所示的信息。

图 2. Windows 检测到高速器件已插入到某个全速端口上



FX2LP 固件负责检测当前工作速度，并指定适当的配置描述符（FS 或 HS）作为当前正在使用的速度，另一个作为与非选定（其它）速度相应的。通过将正确的 *bDescriptorType* 值（CONFIG 或 OTHERSPEED）插入每个配置描述符的第二个字节，固件可以执行上述操作。

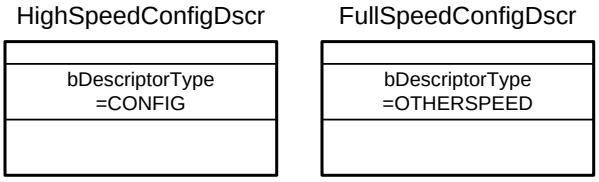
图 3. 复位后的描述符类型



USB 总线 RESET（复位）后，器件将恢复 FS 操作。因此，在 USB 复位中断服务子程序中，固件标出了描述符（图 3）。它也设置指向每个描述符的指针（*pConfigDscr* 和 *pOtherConfigDscr*），供在枚举过程中使用。如果器件被插入全速端口上，将保持此标号。

每当 USB 内核为高速度模式成功协商时，FX2LP 将接收一个中断（ISR_Highspeed）。在该 ISR 中，固件将交换标号，如图 4 所示，并且要设置相应的描述符指针。

图 4. HS 检测后的描述符类型



wMaxPacketSize 的概况

用于高带宽端点的 *wMaxPacketSize* 入口的格式不是很明显。很有吸引力的一点是一个端点在每个微帧能够处理三个 1024 字节的数据包，该端点具有一个最大大小为 3*1024 或 3072 字节的数据包。然而，这个内容不被定义在 USB 2.0 规范中，因为一个物理数据包的最大大小为 1024 字节。规范中的内容定义了位 0-位 10 为 *maxPacketSize*，最大大小为 1024；位 11-位 12 为每个微帧中的*额外数据包*：00 表示没有额外的数据包、01 表示一个额外数据包、10 表示两个额外数据包。因此，按低字节优先的原则，使用表 5 中的值填入高带宽端点描述符。

表 5. 高带宽端点的有效 maxPacketSize 值

每个微帧中的数据包数量	wMaxPacketSize
1	0x0400
2	0x0C00
3	0x1400

CyStream.c

这是数据流程序。该模块中的应用代码是通过 USB Framework（USB 框架）的帮助编写的：

1. 根据应用的要求编写 TD_Init()和 TD_Poll 函数。
2. 由于 Fw.c 将多个器件请求传到端点 0，它将调用您的代码中的特定命名函数。赛普拉斯代码模板（peripheral.c）通过提供包含了所有函数存根的代码帧架保存创建函数的工作。只要对应用中需要使用的函数进行编码。
3. 您需要提供中断服务子程序来处理您的应用所使用的中断。大多数这些程序（用星号表示）是简单的确认内部工作；若不需要采取其他行为，不要修改它们。

本部分的剩下内容描述了 CyStream.c 代码。

TD_Init()

初始化程序执行以下操作：

1. 将 IFCONFIG 寄存器设置为 0x40，以便选择 8051 48 MHz 时钟频率，并将一组 I/O 引脚配置为端口（GPIO）引脚（这些引脚可以作为从设备 FIFO，也可作为 GPIF 接口引脚）。
2. 关闭四个开发板上的 LED。注意，通过读取预定的存储器位置可以打开和关闭开发板上的 LED。
3. 使能 EP2，会禁用所有其他 LED。然后，如果主机将备用设置更改为使用 EP6-OUT 的设置，代码将在设置更改的同时使能该端点。
4. 使能帧开始（SOF）中断。SOF ISR 处理 LED 闪烁时序。
5. 将 1024 数据字节填充到 EP2-IN FIFO 内，并准备执行第一个 IN 传输。在 FIFO 中带有特定数据将便于使用 USB 总线分析仪查看数据包，但是此数据与该应用无关。因此，该步骤是可选的。

TD_Poll()

USB Frameworks 在无限循环中所调用的 TD_Poll()是很简单的。它只需通过检查 FIFO 标志来确定端点 FIFO 需要重新待命的时间点。该操作发生在两个相同的代码段内，其中一个用于高速操作，另一个用于全速操作。每个代码段将执行以下操作：

1. **检查 EP2-IN FIFO 的未满（not FULL）状态。** 该状态表明主机已成功读取了 FIFO IN 数据，并且 FX2LP 逻辑将 FIFO 控制权转给 8051。如果它未满：

   a. 打开 IN LED，并且设置 SOF 中断服务子程序（ISR）的时间常量（inblink）以便关闭它。

   通过加载与当前备用设置相匹配的尺寸加载到 EP2-IN 字节计数寄存器内来重新准备执行下一个 IN 传输。这些值与表 3（HS）和表 4（FS）中最右侧一列相应。

2. **检查 EP2-OUT FIFO 的非空（not EMPTY）状态。**该状态表明主机已经成功将 OUT 数据发送到 EP2-OUT FIFO 内，并且 FX2LP 逻辑将 FIFO 控制权转交给了 8051。如果它为非空：

   a. 打开 OUT LED，并设置 SOF ISR 的时间常量（outblink），以关闭它。

   b. 通过将字节计数加载到 EP2BCL 寄存器内来重新准备执行下一个 OUT 传输。实际字节计数值并不重要，但必须将位 7 置位，以设置 SKIP 位。SKIP 位允许忽略刚收到的数据包，并且它的缓冲器立即可用于下一个 OUT 数据包。

3. **（适用于 HS）检查 EP6-OUT FIFO 的非空状态**。该状态表明主机已经成功将 OUT 数据发送到 EP6-OUT FIFO，并且 FX2LP 逻辑将 FIFO 控制权转交给 8051。如果它为非空，需要执行步骤 2a 和 2b，但是使用 EP6BCL 寄存器。

处理器件请求和 IRQ

大多数简单的器件请求是很明显的，如设置本机变量为配置索引的 DR_SetConfiguration 以及返回索引的 DR_GetConfiguration。简单的 ISR 只清除 IRQ 标志。

CyStream.c 函数

CyStream.c 执行其操作时要使用下面四个函数：

- ISR_Ures
- ISR_Highspeed
- DR_Set_Interface
- DR_SOF

ISR_Ures

在 USB 总线 RESET 事件完成后触发该 ISR。高速器件从全速模式中退出复位状态，所以 ISR 将 FS 配置描述符设置为主描述符，并将 HS 配置描述符设置为 OTHERSPEED 描述符（图 3）。然后，ISR 将关闭 HS LED（EVKIT 板上的 D2），并使用‘blinkmask’为 LED D5 设置两秒周期。

ISR_Highspeed

FX2LP 进入高速模式时将触发该 ISR。配置描述符被替换（图 4），高速 LED2 被打开，另外默认的 EP2-IN 被初始化。设置变量‘blinkmask’以用于在高速模式下的 1 秒 D2 闪烁周期。

DR_SetInterface

USB 主机使用 Set_Interface 请求来更改某个接口的备用设置。由于该设计只使用一个接口，所以无必要检查接口索引。该代码需要检索 SETUP 数据包中表示所请求的备用设置的第三个字节（SETUPDAT[2]）。

两个‘switch’语句按照备用设置并使用表 3（高速模式）和表 4（全速模式）中的参数来配置端点。‘case’语句将执行以下操作：

- 只使能设置项需要使用的端点
- 对于 IN 端点，复位 FIFO 以便刷新任意过时数据。对于 BULK IN 端点，将端点的数据切换复位为 0
- 对于 OUT 端点，准备执行所需的 FIFO 数量
- 更新 7 段显示屏，以便显示新的备用设置 0-6

DR_SOF

SOF 中断对时序事件（如 LED 闪烁频率）非常有用。SOF 中断在每 1 mSec（全速模式）和每 125 微秒（高速模式）发生一次。SOF ISR 将计数关闭时间，并在合适场合下关闭活动 LED D5、IN LED D4 以及 OUT LED D3。

运行 CyStreamer.hex

与所有赛普拉斯 EZ-USB 芯片相同，FX2LP 使用 RAM 来存储程序。因此，它可作为 USB 装载机，将各种来源的程序代码转移到 RAM 内。可以使用附带的 EEPROM 或当作能够从 PC 加载代码的 USB 器件。本部分中的各个步骤介绍了如何通过 USB 下载和运行 *CyStream.hex* 文件。

1. 按照表 6 准备 FX2LP 电路板的跳线器。

表 6. EZ-USB FX2LP 电路板跳线器设置

跳线器	状态	用途
6、7	OUT	用于开发的存储器配置
2	IN	通过 USB 连接器供电给开发板
1、5、10	IN	本地 3.3 V 电压源
模块 3	IN	所有 4 个跳线器为 IN — 激活 4 个 LED D2-D5
8	OUT 或 IN	未使用（用于远程唤醒测试）

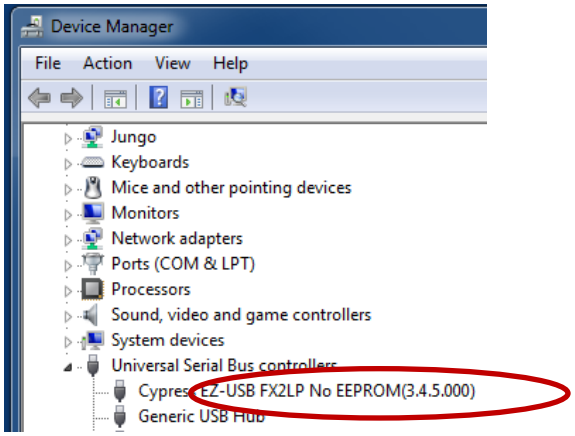
2. 在电路板上的左下角，将 EEPROM ENABLE 滑动开关（SW2）移动到 **NO EEPROM**（下面）的位置。这样会允许 FX2LP 枚举为加载 USB 器件的代码。其他滑动开关（SW1，EEPROM SELECT）可放置在任何位置。

   将 FX2LP 板插入到 PC USB 端口上。如果是第一次进行操作，您会看到弹出消息，提示您安装 USB 驱动程序。请导航到：
   `C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.0\Drivers\cyusbfx1_fx2lp`

3. 选择与您 Windows 操作系统相应的文件夹。

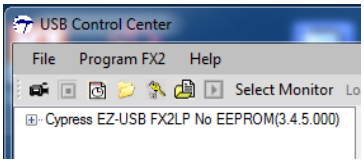
   您可以通过查看 **Device Manager**（器件管理程序）（见下图）来确认成功安装驱动程序。

图 5. 已经正确安装了驱动程序

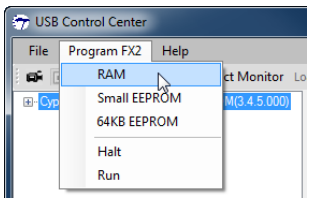


4. 启动在 AN4053 文件夹中的 USB Control Center（USB 控制中心）。可执行文件位于二进制文件夹中。
5. FX2LP 电路板被罗列在左侧面板中，如图 6 所示。

图 6. 器件在 USB Control Center 页面中出现

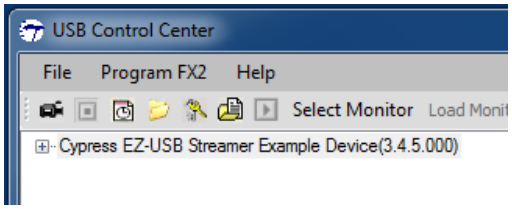


6. 点击赛普拉斯器件条目以高亮显示该项，然后依次选择 **Program FX2 >RAM**



7. 导航到 `AN4053\FX2LP streamer source code\firmwareCyStreamer`，并双击 *CyStream.hex* 文件。
8. 查看 USB Control Center 的左侧面板。如果 PC 的声音器件被使能，您会听到‘USB 连接’声音，随后是‘USB 断开’声音，然后出现新的 USB 器件：

图 7. 新 USB 器件



这是赛普拉斯的重新枚举过程。图 6 中所显示的初始器件是 USB 装载机，它被硬连接到 FX2LP 器件上。将您的代码加载到 FX2LP RAM 内后，FX2LP 将断电并作为图 7 中显示的新 USB 器件（由加载的代码所创建）重新被连接。

注意：当您创建新版本的 CyStream.hex，并想将其加载到 FX2LP 电路板上时，必须先按下电路板上的 RESET 按键。这样会使电路板与 USB 断开连接。当您释放 RESET 按键时，FX2LP 利用 USB loader 重新连接至 PC。建议按下 RESET 按键两秒钟，以便为 Windows 提供足够长的时间来记录断开-重新连接事件。

Streamer 示例器件开始运行，可通过下面标示得到确认：

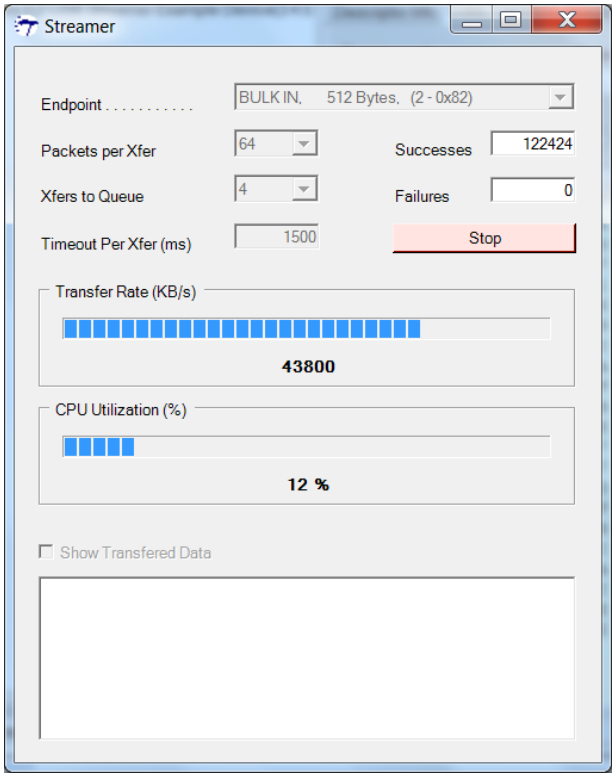
- 7 段数码管显示当前备用设置。默认设置（打开）为 0。
- LED D5 每秒（在高速连接中）或每 2 秒（在全速连接中）闪烁一次；
- 当一个 IN FIFO 准备就绪时，LED D4 将打开。
- 当一个 OUT FIFO 准备就绪时，LED D3 将打开。
- 在高速模式中，LED D2 被打开；在全速模式中，LED D2 被关闭。

性能分析

`AN4053\VS_Streamer`路径下的Streamer.exe被用于评估USB数据流流应用的性能。该代码使用*CyUsb.sys*（AN4053\Driver）驱动程序连接至运行CYStream固件的FX2LP开发板（CY3684）。

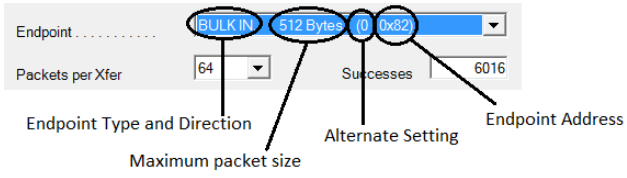
下面是Streamer主机应用窗口的屏幕截图，下面将对图8中所显示的字段进行说明。

图 8. Streamer.exe 应用窗口显示屏



Endpoint（端点）：该下拉列表显示的是备用设置：0-6（适用于高速模式）和 0-3（适用于全速模式）。通过它，可以选择各种传输类型和端点缓冲器数量。

图 9. Endpoint 下拉列表的说明



本部分中显示了 Endpoint 下拉列表中您所看到的文本说明。下拉列表中的第一项（请查阅图 9）为：BULK IN、512Bytes、0-0x82。

BULK IN — 该字段表示端点类型和方向。在这个示例中为 BULK 端点，您可以对它进行 IN 传输操作。

512Bytes — 表示端点的数据包最大大小。在该实例中，数据包最大大小为 512 字节。

0 — 该字段包含备用设置编号。在该实例中，备用设置编号为 0。

0x82 — 端点地址。在该实例中，端点编号为 0x02，方向为 IN。

Packets per Xfer（每传输操作的数据包数量）：一个传输操作指的是一组数据的数据包集合。每次传输中含有的数据包越多，则开销越少，取得的数据速率也会越高。

Xfers to Queue（队列的传输操作）：该设置有助于启动多个传输，并把它们添加到任务队列中。该项减小了主机应用方上的连续传输间的延迟。因此，排列的传输次数越多，数据速率也越高。

Successes（成功）：增加以表示在数据流测试期间已经成功传输的数据包总数。

Failures（失败）：随缓冲区传输的错误报告而增加。一个可能错误机制为不适当处理 ISO IN PID 序列的 FX2（无 LP）应用。

Transfer Rate（传输速率）：提供通过有关所选端点上的 USB 总线和 EZ-USB FX2LP 当前吞吐量性能的实时更新。

CPU Utilization（CPU 的使用）：提供电脑 CPU 在 USB 流式传输期间利用率的视觉指示。

该应用在配备 Intel IHC9 USB 2.0 主机控制器并运行 Windows®7 的的系统上运行。该结果表明，通过 BULK 传输所获得的平均吞吐量为 43.8 MBps，以及通过 ISO 传输获得的最大吞吐量为 24 MBps。注意，上述性能数字是在外部外设未被连接到 FX2LP 上的情况下测得的。在 FX2LP 中内部生成数据。如果您通过外部外设生成数据，这些性能数字也取决于外部外设生成数据的速度。

在某些系统上，您可能看到 BULK 传输速率可以超过 ISO 传输速率。这是因为在轻载总线上，streamer 器件使用大多数或所有可用的 USB BULK 带宽资源。然而，当其它 USB 器件启动时，例如正在播放视频的 USB 磁盘驱动器，需要保证 ISO 端点的协商带宽，所以缩小了可用的 BULK 带宽。

将 FX2LP 连接到图像传感器上

FX2LP 可以连接到图像传感器上，以通过供应商类别驱动程序传送图像。请参考 KBA95736 — 将 FX2LP™连接到图像传感器上。

参考

USB 2.0 规范 —（http://www.usb.org/developers/docs/usb_20_070113.zip）

FX2LP 技术参考手册（TRM）

总结

本应用笔记演示了支持流数据（如：音频和视频流等应用）的 USB 高带宽传输机制。相关固件项目“CYStream.Uv2”说明了如何编程 FX2LP 以用于高速 USB ISO 传输。为了进行带宽对比，还执行了一个 BULK 传输集。它还演示了如何使用赛普拉斯 USB 框架来实现备用 USB 设置，允许主机选择不同的传输速率。还提供了名称为“Streamer.exe”的 PC 应用程序伴侣用以选择各种传输类型和测量传输速率。

关于作者

姓名：	Rama Sai Krishna Vakkantula.
职务：	应用工程师

文档修订记录

文档标题：AN4053 — 在 EZ-USB® FX2™和 FX2LP™上测试同步或批量端点的数据流

文档编号：001-75393

版本	ECN	变更者	提交日期	变更说明
**	3476700	NITA	12/27/2011	本文档译自英文版 001-15289。
*A	4643789	GAYA	01/28/2015	发布前检查。
*B	4880029	LYAO	08/12/2015	本文档版本号为 Rev*B，译自英文版 001-15289 Rev*H。
*C	5789157	AESATMP9	06/28/2017	更新徽标和版权。

全球销售和设计支持

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/pmic
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

赛普拉斯开发者社区

论坛 | WICED IoT 论坛| 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

此处引用的所有其他商标或注册商标归其各自所有者所有。

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

赛普拉斯半导体公司，2007-2017 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由赛普拉斯所有。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。如果软件并不附随有一份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属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，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。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

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。