AN78446

EZ-USB® FX2LP™中的中断处理

作者：Prajith Cheerakkoda
相关项目：有
相关器件系列：CY7C68013A/CY7C68014A/CY7C68015A/CY7V68016A
软件版本：无

更多代码例程？

获取更对关于 FX2LP 的例程请访问 USB High-Speed Code Examples webpage.
你在寻找 USB3.0 外设控制器吗？
USB3.0 系列请访问 USB 3.0 Product Family webpage.

本应用笔记说明了 EZ-USB® FX2LP™中的三种 USB 特殊中断和外部中断的处理情况。另外，还提供了每个中断的示例代码。

目录

1 简介.....1
2 端点中断.....2
3 In Bulk NAK 中断.....2
4 PING NAK 中断.....3
5 外部中断.....4
5.1 中断服务子程序 (ISR).....4
6 固件.....5
6.1 端点中断.....8
6.2 In Bulk NAK 中断.....9
6.3 Ping NAK 中断.....11
6.4 外部中断.....11
7 硬件连接.....12
8 测试项目.....12
9 总结.....14
文档修订记录.....15
销售、解决方案以及法律信息.....16

1 简介

EZ-USB® FX2LP™将其中断架构中的 13 个中断源、5 个标准 8051 中断和 8 个其他 EZ-USB 中断相结合。

标准的 8051 中断：

- IE0(INT0)：外部中断 0
- IE1(INT1)：外部中断 1
- RI_0 与 TI_0：UART 0 中断
- TF0：定时器 0 溢出
- TF1：定时器 1 溢出

其他 EZ-USB 中断：

- TF2：定时器 2 溢出
- PF1：唤醒引脚 (WU2)
- RI_1 与 TI_1：UART 1 发送与接收
- USBINT(INT2)：USB 特殊中断
- I2CINT(INT3)：I2C 总线中断
- IE4(INT4)：外部中断 4
- IE5(INT5)：外部中断 5
- IE6(INT6)：外部中断 6

27 个不同的 USB 特定自动向量中断共享了 USBINT (INT2) 中断。EZ-USB FX2LP 提供了可用向量中断的高级版本，并被命名为自动向量中断。自动向量是用于 EZ-USB FX2LP 的机制，在发生相应中断时，该机制允许自动调用中断服务子程序 (ISR)。更多有关自动向量概念和 USB 特定中断的详细内容，请查看 EZ-USB 技术参考手册的 “第 4.5 章 USB 中断自动向量”。

本应用笔记介绍了下面各种 USB 特定中断：

- 端点中断
- In-Bulk-NAK 中断
- Ping-NAK 中断

除了标准的 8051INT0 和 INT1 外部中断外，EZ-USB FX2LP 还集成了三种新的外部中断：INT4、INT5 和 INT6。本应用笔记说明了三种 USB 中断和所有外部中断的使用情况。假定您已经熟悉了技术参考手册 “第 4 章中断”，并对 8051 中断有了基本了解。

2 端点中断

表 1 显示的是所有端点中断以及它们的优先级和 INT2VEC 值。

对于某个 OUT 端点，中断请求表示 OUT 数据从主机中发出，EZ-USB FX2LP 已经进行了验证，但仍处于端点缓冲区存储器内。

对于 IN 端点，中断请求表示 EZ-USB 已经将数据加载到 IN 端点缓冲区内。主机读取并验证了该缓冲区，从而使它准备好接收新数据。

表 1. EZ-USB 端点中断

中断名称	优先级	INT2VEC 值	注释
EP0IN	9	20	EP0-IN 准备好加载数据
EP0-OUT	10	24	EP0-OUT 包含 USB 数据
EP1IN	11	28	EP1-IN 准备好加载数据
EP1-OUT	12	2C	EP1-OUT 包含 USB 数据
EP2	13	30	IN：缓冲区可用。OUT：缓冲区包含数据
EP4	14	34	IN：缓冲区可用。OUT：缓冲区包含数据
EP6	15	38	IN：缓冲区可用。OUT：缓冲区包含数据
EP8	16	3C	IN：缓冲区可用。OUT：缓冲区包含数据

3 In Bulk NAK 中断

当主机向 EZ-USB FX2LP Bulk 端点请求一个 IN 数据包时，端点会 NAK (没有应答，即返回 NAK 数据包)，直到填充端点缓冲区并将其准备传输为止，此时 EZ-USB FX2LP 可使用数据包来应答 IN。

直到确认端点，大量 IN-NAK 仍可以连接总线带宽。因此，如果 IN 端点并非始终保持已满和就绪状态，那么应该了解主机 “敲门” 的时间，以请求 IN 数据。In-Bulk-NAK (IBN) 中断会发出该提示。每当批量端点没有应答 (NAK) IN 请求时，均会生成一个 IBN 中断。

4 Ping NAK 中断

当器件运行于全速模式时，即使端点没有应答 (未就绪)，每个主机 OUT 传输仍包含 OUT PID 和端点数据。端点未就绪时，主机会重复发送所有 OUT 数据；如果

端点重复没有应答，便会浪费总线带宽。USB 2.0 规范介绍了一种新的机制 (又称 PING)， 在批量输出端点未就绪时，通过该机制可够更加合理地使用总线带宽。

在高速模式下，主机可以‘ping’ (询问) 一个批量输出端点，以确定它是否准备好接收数据，主机可以拖延 OUT 数据传输，直到端点实际能够接收数据为止。主机发送 PING 令牌，EZ-USB FX2LP 将通过以下操作做出响应：

- 发出 ACK (应答) 信号，表示 OUT 端点缓冲区中有可用空间。
- NAK (没有应答) 信号表示‘未就绪，稍后重试’。

PING 中断表示某个 EZ-USB FX2LP 批量 OUT 端点返回了一个 NAK (没有应答) 信号，作为对 PING 的响应。

注意：PING 仅用于高速模式 (480 Mbps)。

表 2 显示的是 “Ping” 中断以及它们的优先级和 INT2VEC 值。各个中断的中断使能位均位于 NAKIE 寄存器内，并且中断请求标志位于 NAKIRQ 寄存器内。

表 2. EZ-USB PING 中断

中断名称	优先级	INT2VEC 值	注释
EP0 PING	19	48	询问了 (PING) EP0，但它被没有应答
EP1 PING	20	4C	询问了 (PING) EP1，但它被没有应答
EP2 PING	21	50	询问了 (PING) EP2，但它被没有应答
EP4 PING	22	54	询问了 (PING) EP4，但它被没有应答
EP6 PING	23	58	询问了 (PING) EP6，但它被没有应答
EP8 PING	24	5C	询问了 (PING) EP8，但它被没有应答

5 外部中断

该表列出了 EZ-USB FX2LP 芯片中的所有外部中断。

图 1. 外部中断

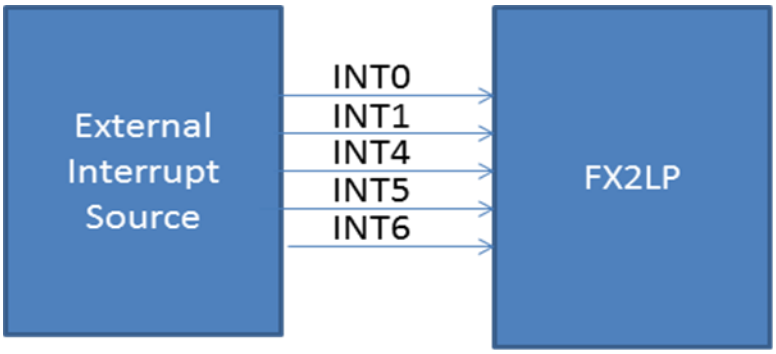


表 3. 外部中断

中断	中断使能	中断引脚	优先级控制	默认优先级	中断请求标志	中断类型	中断类型控制位
INT0	IE.0	PA.0	IP.0	1	TCON.1	电平敏感或边沿敏感，低电平有效	[TCON.0]
INT1	IE.2	PA.1	IP.2	3	TCON.3	电平敏感或边沿敏感，低电平有效	[TCON.2]
INT4	EIE.0	请参考注释 1	EIP.2	8	EXIF.4	边沿敏感，高电平有效	--
INT5	EIE.1	请参考注释 1	EIP.3	9	EXIF.5	边沿敏感，低电平有效	--
INT6	EIE.4	PE.5	EIP.4	12	EICON.3	边沿敏感，高电平有效	--

注释：

- 仅在 100 和 128 封装中，INT4 和 INT5 才有专用引脚 (CY7C68014A-128AXC、CY7C68013A-128AXI、CY7C68013A-100AXC 和 CY7C68013A-100AXI)。GPIF/FIFO/INT4 中断共享了 INT4 的引脚。将 INTSETUP 寄存器 INTSETUP.1 设置为‘0’，可使能 INT4 操作。默认的 USBJmpTb.a51 具有用于 INT4 的自动向量选项。要想禁用该选项，必须使汇编源文件 USBJmpTb.a51 中中断向量表内的各种代码行无效：

```
CSEG AT 53H
USB_Int4AutoVector equ $ + 2
ljmp USB_Jump_Table
```

- IE、EIE、IP、EIP、TCON、EXIF 和 EICON 都是 SFR (特殊功能寄存器)。更多有关这些 SFR 的说明，请参考 EZ-USB 技术参考手册中介绍的内容。
- 低电平有效中断在下降沿上被触发，高电平有效中断在上升沿上被触发。

5.1 中断服务子程序 (ISR)

在该示例中，所有外部中断的 ISR 均被定义在‘C’文件‘***isr.c***’中。例如，为 INT0 定义的 ISR 的格式为：

```
void ISR_EXTR0 (void) interrupt 0
{
//The code to execute within the ISR;
}
```

该函数作为 INT0 的 ISR，C 或汇编函数不能调用它。当发生 INT0 时，将自动执行该函数。

“interrupt 0” 通知编译器在地址 0x0003 上查找 ISR。同样，INT5 的 ISR 如下：

```
void ISR_EXTR5 (void) interrupt 11
{
// The code to execute within the ISR;
}
```

6 固件

本应用笔记所提供的相关项目说明了如何使能和处理这些中断。编写固件，从而使它根据 USB 特定中断执行数据回送操作。批量端点根据端点中断、In-Bulk-NAK 中断和 Ping NAK 中断分别将 EP1OUT-EP1IN、EP2OUT-EP6IN 和 EP4OUT-EP8IN 移出回送操作配对。对于外部中断，固件切换 DVK LED 和端口 C 引脚的频率是引脚上中断请求频率的一半。

本应用笔记中的相关项目使用 Bulkloop 操作演示了上述所有 USB 特定中断的工作原理。Bulkloop 是一种数据回送操作，它使用 EZ-USB FX2LP 端点上的 Bulk OUT 和 Bulk IN 进行传输。在 Bulk OUT 传输期间，主机将数据发送到 OUT 端点 (如果它为空) 上。固件将 OUT 端点中的数据复制到空的 IN 端点内，然后提交给 IN 端点缓冲区内。在 Bulk IN 传输期间，主机从已提交的 IN 端点上读取数据。总之，Bulkloop 操作从主机读取数据并将读到的数据发送回主机。

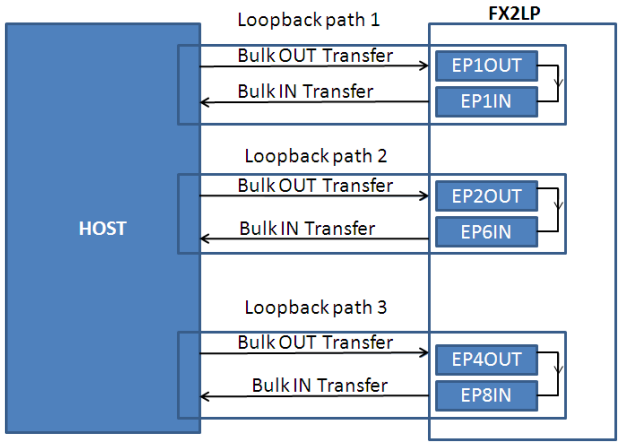
IN-OUT 端点对为 Bulkloop 操作提供数据回送路径。在该示例项目中，Bulkloop 操作具有三种数据路径 (如图 2 所示)，因此意味着固件具有三对 IN-OUT 端点。数据流根据 USB 特定中断通过这些路径发生。通过路径 1 进行的数据传输取决于端点中断。与此相似，通过路径 2 和路径 3 进行的数据传输则分别取决于 In-Bulk-NAK 和 Ping NAK 中断。环回操作包括以下步骤。

1. 激活 OUT 端点
2. 将 OUT 中的数据复制到 IN 端点上
3. 提交 IN 端点中的数据
4. 重新激活 OUT 端点

注意：

- 激活 OUT 端点意味着为串行接口引擎 (SIE) 提供缓冲区空间，用于接收主机中的数据。
- 通过提交 IN 端点，主机可以对 FIFO 缓冲区进行访问以读取端点中的数据。

图 2. 外部中断



在所有 USB 中断服务子程序 (ISR) 中，清除单独 USB 中断请求锁存器前，需要先清除主 USB 中断。该要求非常重要。这是因为清除单独的 USB 中断后，所有挂起的 USB 中断会立即尝试生成另一个主 USB 中断。如果没有提前清除主 USB IRQ 位，挂起的中断将被丢失。典型的 USB ISR 结构如下所示：

```
USB interrupt_ISR
{
; FIRST clear the USB (INT2) interrupt request
; Clear the USB interrupt request
; Service the interrupt here
}
```

在该框架上构建的相关项目被作为 CY3684 套件的一部分。汇编源文件 dscr.a51 为三个数据回送路径定义了主机的三对端点。端点 2 和端点 4 为 OUT 端点。端点 6 和端点 8 是 IN 端点。另外还定义了 EP1 OUT 和 EP1 IN 端点。所有端点均被配置为 Bulk 端点。

代码执行从文件 *fw.c* 中的主函数开始。文件 *intr.c,*中定义的函数 TD_Init()在主函数中被调用。该函数会初始化所有端点并使能上述四种中断。

汇编源文件 dscr.a51 中定义的六个端点通过该函数被配置。通过下列语句实现该操作：

```
EP1OUTCFG = 0xA0;
EP1INCFG = 0xA0;
SYNCDELAY;
EP2CFG = 0xA2;
SYNCDELAY;
EP4CFG = 0xA0;
SYNCDELAY;
EP6CFG = 0xE2;
SYNCDELAY;
EP8CFG = 0xE0;
SYNCDELAY;
```

每个寄存器的定义在技术参考手册的第 15 章中介绍。每个端点的关键特性如下所示：

- 端点 1 — IN, Bulk
- 端点 1 — OUT, Bulk
- 端点 2 — OUT、Bulk、双缓冲
- 端点 4 — OUT、Bulk、双缓冲
- 端点 6 — IN、Bulk、双缓冲
- 端点 8 — IN、Bulk、双缓冲

```
// out endpoints do not come up armed.  Arm EP1, EP2 and EP4 OUT endpoints
EP1OUTBC = 0x40;
// arm the EP1 OUT endpoint by writing to the byte count
// since the defaults are double buffered we must write dummy byte counts twice
SYNCDELAY;
EP2BCL = 0x80;
// arm EP2OUT by writing byte count w/skip.
SYNCDELAY;
EP2BCL = 0x80;
SYNCDELAY;
EP4BCL = 0x80;
// arm EP4OUT by writing byte count w/skip.
SYNCDELAY;
EP4BCL = 0x80;
```

下列两行使能了为该项目定义的 EP1 端点上的中断。

```
EPIE |= bmBIT3 ; // Enable EP1 OUT Endpoint interrupts
EPIE |= bmBIT2; // Enable  EP1 IN  Endpoint interrupts
```

IBNIE 寄存器在每个端点上都有一个单独的中断使能位：EP0、EP1、EP2、EP4、EP6 和 EP8。只有端点被配置为批量或中断端点时，这些位才有效。IBNIRQ 寄存器在 6 个端点上均包含了各个单独的中断请求位。通过设置 IBNIE 寄存器中相应的位，可以使能 EP6 In-Bulk-NAK 中断。每个 NAKIE/NAKIRQ 寄存器均包含一个单独的位 “IBN”，它是 IBNIE/IBNIRQ 中各单独位进过 OR 组合的结果。更多有关这些寄存器的信息，请参考技术参考手册的第 8.6.3.1 章节。固件通过将使能位设置为高电平来使能中断，也可以通过向该位写入‘1’来清除中断请求。可通过下列代码执行这些功能：

```
// clear the global IBN IRQ
NAKIRQ = bmBIT0;
// enable the global IBN IRQ
NAKIE |= bmBIT0;
// clear any pending IBN IRQ
IBNIRQ = 0xFF;
// enable the IBN interrupt for EP6
IBNIE |= bmEP6IBN;
```

TD_Init 函数中的下列代码将清除所有挂起的 PING-NAK 中断并使能 EP4 的 PING-NAK 中断。

```
NAKIRQ |= ~bmIBN; // clear any pending PINGNAK IRQ
NAKIE |= bmEP4PING; // enable the PING-NAK interrupt for EP2 and EP4
```

在相关项目中，下列寄存器配置将在‘intr.C’中实现，用于将外部中断和端口 C 设置为输出端口：

```
//INT0 and INT1
PORTACFG = 0x03;
// PA0 and PA1 are pins for INT0 and INT1 respectively.
TCON |= 0x05;
// INT0 and INT1 are configured as Edge triggered interrupts.
//INT4
INTSETUP &= ~0x02;   // If INTSETUP.1=0, then INT4 is supplied by the pin. Else, the
// interrupt is supplied internally  by FIFO/GPIF sources.
//INT5  is a dedicated  pin , available in the 100 and 128 pin packages.
//INT6
PORTECFG = 0x20;// PE5  is  INT6
OEE &= ~0x20;
//Enable External Interrupts
EIE |= 0x1C; // Enable
External Interrupts 4, 5 and 6
IE |= 0x05;//Enable External Interrupts 0 and 1
//Clear  Flags
EXIF &= 0xBF;  // Clear INT4 EXIF.6  Flag
EXIF &= 0x7F;  // Clear INT5 EXIF.7  Flag
EICON &= 0xF7; // Clear INT6 EICON.3  Flag
EA   = 1; // Enable Global Interrupt
PORTCCFG = 0x00; // PORTC is is configured as an I/O, alternately
//it can output the lower address of enabled GPIF address pins
OEC  = 0xFF;       // PORTC is an output
IOC = 0xFF;  // Initialize PORTC to all LOW
```

通过不同方式处理 8051 中断和 USB 中断。 USB 中断向量的生成由 USB 跳转表 (USBJmpTb.obj) 处理，而 8051 中断向量的生成则由 Keil 编译器处理。intr.c 中的 “#pragma noiv” 语句会通知 Keil 编译器使用 USB 跳转表生成 USB 中断的中断向量，而不是其自身的中断向量方案 (该方案随后传统的 8051 方案)。

6.1 端点中断

端点 ISR，即 ISR_Ep1in 和 ISR_Ep1out 分别与端点 EP1IN 和 EP1OUT 相关联。执行这两个 ISR 能够使数据流通过该路径。Out 端点 ISR 将数据从 EP1OUT 传输到 EP1IN (若可用)，并允许主机使用 EP1IN 缓冲区中的数据。EP1IN ISR 重新激活 EP1OUT 端点，以便使其能够接收主机中的新数据。主机的新传输按该周期继续执行。中断辅助操作 (如清除中断请求) 也对 ISR 的内部产生影响。

下面代码段是中断服务子程序。每当发生 EP1OUT 传输时，都将处理该程序。如果端点 EP1OUT 带有数据 (从主机发送)，则将检查端点 EP1IN 是否能够接收数据。该操作是通过读取端点状态寄存器 EP1INCS 中的 EP1IN 繁忙位实现的。如果端点 EP1IN 未满，将传输数据。

可通过以下语句执行：

```
if(!(EP1INCS & bmBIT1))
{// Checks EP1IN availability
```

可以从字节计数寄存器 EP1OUTBC 中读取需要传输的字节数量。EP1OUTBC 寄存器包含了主机写入 FIFO 缓冲区中的字节数量。

```
count = EP1OUTBC;// The count value is loaded from the byte count register
```

通过执行以下循环可以实现数据传输：

```
for (i=0;i<count; i++)
{
       EP1INBUF[i]=EP1OUTBUF[i];
}
```

下列代码可赋予 IN 端点功能。它允许主机使用该缓冲区并清除中断请求。

```
EP1INBC =count;
EZUSB_IRQ_CLEAR();//Clears the USB interrupt
EPIRQ = bmBIT3;//Clears EP1 OUT interrupt request
```

数据传输完成后，将重新激活端点 EP1OUT 以接收主机发来的新数据包。通过执行 IN 端点 ISR 中的下列代码，可实现该操作：

```
EP1OUTBC = 64;
EZUSB_IRQ_CLEAR();//Clears the USB interrupt
EPIRQ = bmBIT2;//Clears EP1 IN interrupt request
```

注意：更多有关中断延迟的信息，请参考 EZ-USB 技术参考手册中 “4.3.3 章中断延迟” 的内容

6.2 In Bulk NAK 中断

显示在图 2 中的数据路径 2 取决于 In-Bulk-NAK 中断。与端点中断不同，只有一个 ISR 与 IBN 中断相关联。例如，所有 Bulk IN 端点只有一个 ISR。通过使用 IBNIRQ 寄存器可以找到没有应答的端点。在 IBNIRQ 寄存器中，每个端点均具有单独的请求位。在该项目中，只要 EP6 没有应答，便会执行 IBN ISR。这是因为它是唯一一个带有使能 IBN 的端点。

IBN ISR 按下列顺序进行操作：

1. 通过将‘0’写入到刚才 NAK (没有应答) 的 IN 端点清除 USB (INT2) 中断请求 并禁用所有端点的 In-Bulk-NAK 中断以便 ISR 在执行期间不被中断。

   ```
   IBNIE = 0x00;
   // clear the global USB IRQ
   EZUSB_IRQ_CLEAR();
   ```

2. 检查 IBNIRQ 中的端点位，以确定 刚才 NAK (没有应答) 的是哪个 IN 端点。为多个端点使能 IBN 时，该步骤的作用非常重要。由于仅为 EP6IN 使能了 IBN，因此不需要执行该步骤。

3. 执行所需操作，然后通过向 IBNIRQ 中的单独 IBN 位写入‘1’来为需要操作的端点清除该位。数据传输即为需要进行的操作。如果端点 2 中存在 (从主机发送的) 数据，那么可以通过读取端点状态寄存器中的空位进行检查。可通过以下语句执行：

   ```
     if (!(EP2468STAT & bmEP2EMPTY))
     {
   // check EP2 EMPTY (busy) bit in EP2468STAT (SFR), core set's this bit when FIFO is
   empty
   ```

   数据指针被初始化以指向相应的缓冲区。第一个自动指针被初始化以指向端点 2 FIFO 缓冲区中的第一个字节。第二个自动指针被初始化以指向端点 6 FIFO 缓冲区中的第一个字节。可以从端点 2 的字节计数寄存器中读取需要传输的字节数量。寄存器 EP2BCL 和 EP2BCH 包含了主机写入 FIFO 缓冲区内的字节数量。这两个寄存器提供了传输到

FIFO IN 和 OUT 缓冲区的数据字节的数量，直到数据停止传输到外设为止。通过以下语句初始化数据指针和加载该计数：

```
APTR1H = MSB( &EP2FIFOBUF );
// initializing the first data pointer
APTR1L = LSB( &EP2FIFOBUF );

AUTOPTRH2 = MSB( &EP6FIFOBUF );
// initializing the second data pointer
AUTOPTRL2 = LSB( &EP6FIFOBUF );

count = (EP2BCH << 8) + EP2BCL;
// The count value is loaded from the byte
count registers
```

这里的 Bulkloop 操作与其他两个 USB 特定中断有点不同。在 ISR_Ibn 中，接收到的数据的第一个字节递增，然后它被写入到 IN 端点缓冲区内。通过以下代码可实现该操作。

```
EXTAUTODAT2 = EXTAUTODAT1+1;
```

剩下的数据字节被保持原样复制到 IN 端点缓冲区内。通过执行以下循环可以实现数据传输：

```
for ( i = 0x0001; i < count; i++ )
   {
// setup to transfer EP2OUT buffer to EP6IN buffer using AUTOPOINTER(s)
       EXTAUTODAT2 = EXTAUTODAT1;
   }
```

语句

```
EXTAUTODAT2 = EXTAUTODAT1;
```

将数据从端点 2 传输到端点 6。每次执行上面语句，自动指针都将自动递增。重复执行该语句，从而将各字节从端点 2 传输到端点 6。

通过以下代码可以发送 IN 端点，即允许主机使用缓冲区。

```
EP6BCH = EP2BCH;
SYNCDELAY;
EP6BCL = EP2BCL; // arm EP6IN

IBNIRQ = bmEP6IBN; // clear the IBN IRQ
IBNIE |= bmEP6IBN; // enable the IBN IRQ
SYNCDELAY;
```

数据传输后，将重新激活端点 2，以接收来自主机的新数据包。

```
EP2BCL = 0x80; // re(arm) EP2OUT
```

4. 将‘1’写入到 NAKIRQ 寄存器中的 IBN 位，从而清除它并使能 IBN 中断。

```
NAKIRQ = bmBIT0;// clear the global IBN IRQ
IBNIE = bmEP6IBN; // Enable IBN for EP6
```

6.3 Ping NAK 中断

EZ-USB FX2LP 将 PING-NAK 中断作为 EP0PING、EP1PING 等实现，一个端点使用一个。当主机询问 (PING) 端点，并且 EZ-USB FX2LP 返回 NAK 信号 (因为不可用特殊端点缓冲区存储器) 时，将确认 EPxPING 中断。EZ-USB FX2LP 固件框架为其实现的所有中断提供更改性能。该示例项目使用 ISR_Ep4pingnak 中断服务子程序来处理 EP4PING 中断。以下代码适用于 EP4 ISR。

```
void ISR_Ep4pingnak(void) interrupt 0
{
SYNCDELAY; // Re-arm endpoint 4
EP4BCL = 0x80;
EZUSB_IRQ_CLEAR();
//clear the 4interrupt
NAKIRQ = bmEP4PING;
}
```

ISR_Ep4pingnak 通过取消激活该端点 (即 EP4BCL = 0x80) 去除先前存储在端点 4 中一个缓冲区内的数据。因此，EP4 现在可以接收主机正在发送的数据。这是因为其中一个缓冲区内还有剩余空间。

然后，EP4 通过设置 NAKIRQ 中的特殊位来清除该中断，因为它已经被执行。

6.4 外部中断

在相关项目中，端口 C 被配置作为控制 DVK LED D2 的切换。当发生某个 INT0 中断时，PC.0 与 PC.1, D3/ PC.4 和 D4/ PC.5 和 D5/ PC.6 进行切换。在这些中断的中断服务子程序内编写好了用于切换 LED 和端口 C 引脚的代码

在该示例中，所有外部中断的 ISR 均被定义在‘C’文件‘***isr.c***’中。例如，为 INT0 定义的 ISR 的格式为：

```
void ISR_EXTR0 (void) interrupt 0
{
//The code to execute within the ISR;
}
```

该函数作为 INT0 的 ISR，C 或汇编函数不能调用它。当发生 INT0 时，将自动执行该函数。

“interrupt 0” 通知编译器在地址 0x0003 上查找 ISR。同样，INT5 的 ISR 如下：

```
void ISR_EXTR5 (void) interrupt 11
{
// The code to execute within the ISR;
}
```

在 ISR.C 中，INT4 的示例 ISR 代码为

```
void ISR_EXTR4(void) interrupt 10
{
EXIF &= 0xBF;// Clear INT4 EXIF.6  Flag
IOC ^= 0x10;// Toggle pin 4 of PortC
}
```

7 硬件连接

更多有关 CY3684 开发套件的默认跳线器设置的信息，请参考 FX2LP™入门手册文档的表 3。表 4 显示的是用于所需要引脚的各种端口/跳线器名称 (这些引脚用于测试 CY3684 上的外部中断)。

表 4. EZ-USB 跳转指令

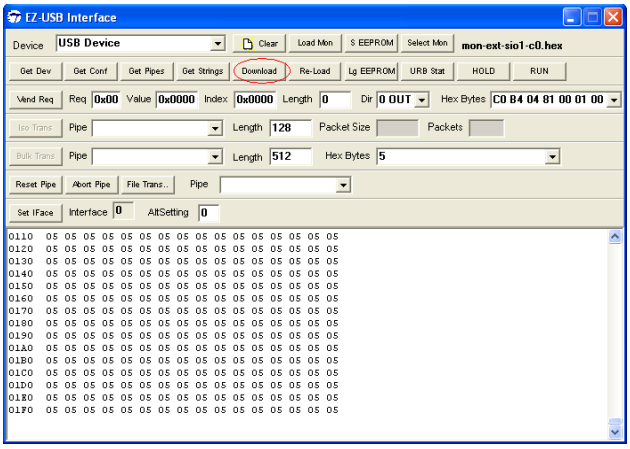
引脚名称	CY3684 上的端口/跳线器名称
端口 C	P3
LED	JP3
INT0	P2.19
INT1	P2.18
INT4	P6.5
INT5	P6.4
INT6	PE.5

8 测试项目

本节介绍了如何测试三条数据回送路径的功能，并介绍了端口 C 引脚和 DVK LED 如何根据外部中断进行切换。要想进行该测试，请按照以下步骤进行操作：

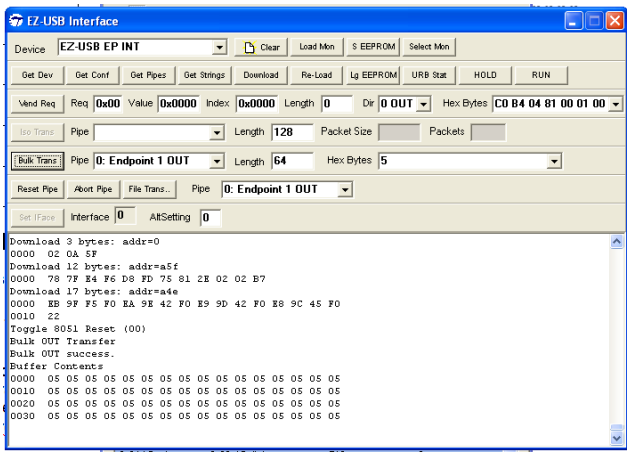
1. 下载并安装 SuiteUSB 3.4，然后安装 CyConsole 工具。
2. 将 CY3684 电路板连接至 PC (将 EEPROM ENABLE 开关放置在 “No EEPROM” 的位置) 上。现在，该电路板会枚举为默认的内部描述符。使用 Associated_project\driver 文件夹中的 *CyUSB.inf* 来绑定器件。有关绑定驱动器的详细信息，请参考驱动器文件夹中的 MatchingDriverToUSBDevice.htm 或 “Matching Devices to the Driver”。
3. 打开 CyConsole 工具。依次选择 **Start** > **All Programs** > **Cypress** > **USB** > **CyConsole EZ-USB**。
4. 通过点击 **Download** 按键，将已编译的十六进制文件 ‘*extr_intr.hex*’ 下载到 RAM 内，然后选择用于放置该文件的路径。下载完成后，将提示驱动程序。
5. 将新的 VID/PID (一个用于 dscr.asm 文件内：PID-04B4 VID-1004) 添加到 CyUSB.inf 然后将其绑定到器件。

图 3. 下载 Endpoint_Interrupts.hex



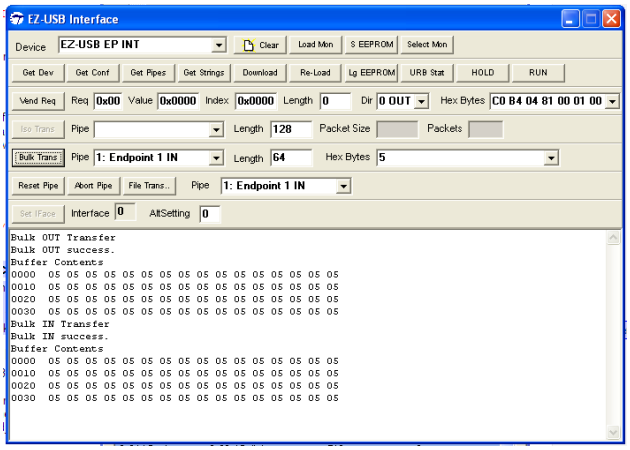
6. 为测试数据路径 1，需要使用 CyConsole 工具将 64 字节的用户定义数据从主机发送到端点 EP1OUT 上。例如，在管道中选择 ‘**0: Endpoint 1 OUT**’，将长度设置为 64，并将 HexBytes 设置为 5，然后再点击 ‘Bulk Trans’ 按键。

图 4. Bulk OUT 传输



7. 通过使用 CyConsole 工具可以从端点 EP1IN 读取该数据。例如，在管道中选择‘**1: Endpoint 1 IN**’，将长度设置为 64，然后点击 **Bulk Trans** 按键。

图 5. Bulk IN 传输



测试其他中断的流程将在下面相应部分进行介绍

- **In-Bulk-NAK 中断：**对于 OUT 和 IN 数据，通过数据返回路径 2 进行的传输会分别选择端点 EP4OUT 和 EP8IN，而不是 EP1OUT 和 EP1IN。这些端点具有的数据包最大为 512 个字节。根据所述内容执行数据回送路径 1，并验证所接收到数据的第一个字节已经递增，并且剩下的字节保持不变。
- **Ping NAK 中断：**在这里，数据包最大为 512 个字节的端点 EP4OUT-EP8IN 端点将执行数据传输。分别为 OUT 和 IN 数据选择端点 EP4OUT 和 EP8IN。进行数据回送并验证接收到的数据是否和已传输的数据相同。通过连续将数据发送至 EP4 可以检测该代码，而不需要读取 EP8 中的数据。由于 PING-NAK ISR 重新激活了该端点，因此您可以连续传送数据 EP4 并且始终能够完成该传输。缓冲区内当前的数据即为主机最近发送的两个数据包。
- **外部中断：**这些中断类型使用了外部源来触发中断，例如，使用功能生成器生成中断。通过设置功能生成器可以生成已知频率方波 (使用低频。例如通过 100Hz 的信号来查看 LED 的切换)。当相应中断被触发时，会出现 LED 切换。当 INT0 中断发生时，PC.0 和 D2 将被切换。类似的，在 INT1/ INT4/ INT5/ INT6 上，PC.1 和 D3/ PC.4 和 D4/ PC.5 和 D5/ PC.6 进行切换。通过将这些引脚连接至 DSO 可以检查端口 C 引脚的切换情况。

9 总结

本应用笔记作为快速指南提供给想使用 EZ-USB FX2LP 的 USB 特定中断和外部中断来开发应用的客户。示例代码演示了 EZ-USB FX2LP 的三个 USB 特定中断和处理所有外部中断的过程。通过使用本文档和技术参考手册中的第 4 节 “中断”，客户很容易便能够开发 USB 特定中断代码。

关于作者

名称：Prajith Cheerakkoda

职务：应用工程师

文档修订记录

文档标题：AN78446 — EZ-USB® FX2LP™中的中断处理

文档编号：001-98027

版本	ECN	变更者	提交日期	变更说明
**	4802510	LYAO	07/14/2015	本文档版本号为 Rev**，译自英文版 001-78446 Rev**。
*A	5822665	AESATMP9	07/18/2017	更新徽标和版权。
*B	6263793	SSAS	07/27/2018	本文档版本号为 Rev*B，译自英文版 001-78446 Rev*C。

销售、解决方案以及法律信息

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、原厂代表和经销商组成的全球性网络。如欲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 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，2012-2018 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。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。