5.1.1 底层函数

最底层为两个面向硬件电路的函数：outportb(unsigned char Addr，unsigned char Data)，inportb(unsigned char Addr)，用以完成CPU和D12之间的通信，具体代码如下：

```
void outportb(unsigned char Addr, unsigned char Data)
{
*((unsigned char xdata*)Addr)=Data;
}
unsigned char inportb(unsigned char Addr)
{
return*((unsigned char xdata*)Addr);
}
```

5.1.2 命令接口

为了进一步简化D12的编程固件，Philips公司定义了一套压缩了所有访问D12功能的命令接口。主要组成函数如下：

```
void SetAddressEnable(unsigned char Address, unsigned char Enable);
//设置地址使能
void SetEndpointEnable(unsigned char Enable);                    //设置
端点使能
void SetMode(unsigned char Config, unsigned char ClkDiv);  //设置
模式
unsigned short ReadInterruptRegister(void);                //读中断
寄存器
unsigned char SelectEndpoint(unsigned char Endp);              //选择
端点
unsigned char ReadLastTransactionStatus(unsigned char Endp); //读
最后传输状态
void SetEndpointStatus(unsigned char Endp, unsigned char Stalled);
//设置端点状态
void SendResume(void);                                         //发送续
传
unsigned char ReadEndpoint(unsigned char endp, unsigned char*buf,
unsigned char len);                                        //读取端点
unsigned char WriteEndpoint(unsigned char endp, unsigned char*buf,
```

```
unsigned char len);                                    //写端点
```

以上所有命令都是通过调用底层函数实现的。

5.1.3 中断服务程序

5.1.3.1 FPGA 模块中断服务程序

这部分程序在 USB 上传数据模块中实现，它对 FPGA 模块上传数据时产生的中断进行处理。在中断处理程序中，单片机循环向 FPGA 模块发送读脉冲信号，然后接收 64 字节的数据到内部缓冲区并设立标志位。在主程序中，通过主端点将接收到的数据发送到 D12 的数据缓冲区里面。在数据发送完成后，主程序清除该标志位以便接收新的数据包[17]。在这段代码中，采用了汇编语言编写以节省时间，保证接收效率。流程图如图 5-2 所示：

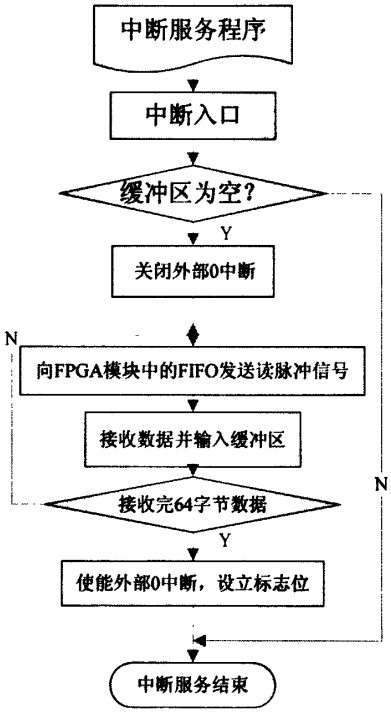

图 5-2 FPGA 模块中断服务程序流程图

5.1.3.2 D12 中断服务程序

这部分程序对 D12 产生的中断进行处理。它将数据从 D12 的内部 FIFO 取回到 CPU 存储器，并建立正确的事件标志以通知主循环程序进行处理[18]。在 ISR 函数的入口使用 Read Interrupt Register()来决定中断源，根据中断源进入相应的子程序进行处理。ISR 从 D12 收集数据，而主循环对数据进行处理。当 ISR 收集了足够的数据时，它只通知主循环已经准备好等待处理，这将减少主循环不必要的服务时间并且简化了主循环的编程。中断服务程序流程如图 5-3 所示：