第 5 章 软件程序的设计

系统启动后，单片机首先对系统进行初始化；然后单片机进入循环程序，通过外部中断产生的事件标志来判断数据采集卡是联机工作还是脱机工作，并执行相应的固件程序；如果在设定时间内既没有上位机连接也没有 U 盘接入，则程序结束，系统停止工作。

5.1 D12 固件的编写

在设计中，采用 C51 语言对单片机固件部分进行编写，使用德国 Keil Software 公司的 Keil uVision2 编译器进行编译并产生下载的配置文件（.hex 文件）。固件的主要功能是控制 D12 芯片接收并处理 USB 设备驱动程序的请求，通过 D12 实时将 FPGA 处理完成送过来的数据上传给主机或者将主机下载的数据传给 FPGA 处理[14]。

固件设计的目标就是使 D12 在 USB 上达到最大的传输速率，这样可以将固件设计成中断形式，CPU 处理前台任务，USB 的传输在后台中断服务程序（ISR）进行，后台服务程序和前台主程序循环之间的数据交换通过事件标志和数据缓冲区来实现。当 USB 芯片接收到一个数据包的时候，其对单片机产生一个中断，CPU 立即响应中断，在 ISR 中，固件将数据包从 USB 芯片内部缓冲区移到循环数据缓冲区并在随后清零 USB 芯片内部缓冲区以使能接收新的数据包。CPU 可以继续它当前的任务直到完成，然后返回到主循环检查循环缓冲区内是否有新的数据并开始其他的前台任务。

由于这种结构，主循环不关心数据是来自 USB、串口还是并口。它只检查循环缓冲区内需要处理的新数据。这样主程序只需专注数据处理而 ISR 能够以最大的可能的速度进行数据的传输[15]。ISR 与前台主循环通过事件标志 EPPFLAGS 和数据缓冲区 CONROL_XFER 进行通信。EPPFLAGS 的定义如下：

```
typedef union_epp_flags
{struct_flags
{unsigned char bus_reset: 1;          //总线复位标志
unsigned char suspend: 1;             //总线挂起标志
unsigned char setup_packet: 1;            //包建立标志
unsigned char remote_wakeup: 1;       //远程唤醒标志
unsigned char in_isr: 1;              //D12 中断处理标志
unsigned char control_state: 2;       //控制状态标志
unsigned char configuration: 1;       //配置状态标志
unsigned char ep1_rxdone: 1;          //端点 1 收到数据标志
```

```
unsigned char main_rxdone: 1;          //主端点收到数据标志
}bits;
unsigned short value;
}EPPFLAGS;
```

固件的层次结构如图 5-1 所示：

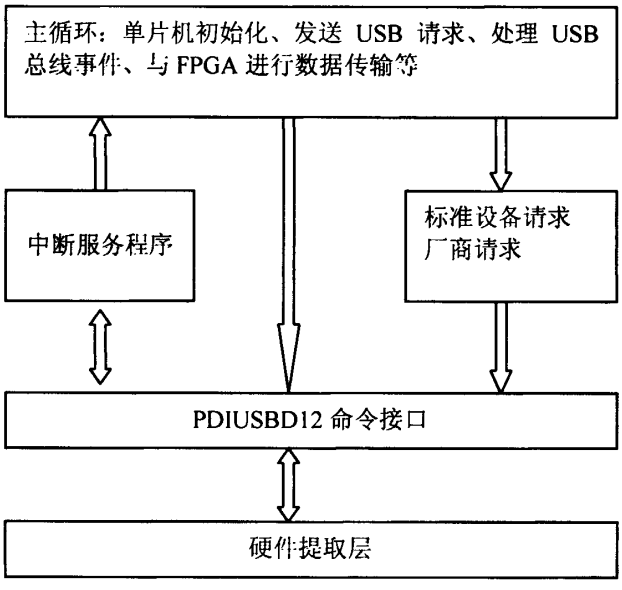

图 5-1 固件的层次结构图

在系统中，USB 数据下载模块和上传模块的固件设计结构是一样的。其中，在 USB 下载数据模块部分。PC 主机通过主端点向 D12 发送数据，D12 收完数据向单片机产生中断。单片机判断中断类型，如果是主端点输入中断，则读取 D12 内部缓冲区的数据到循环缓冲区中，同时清零 D12 内部缓冲区。这些操作完成后，单片机将数据发给 FPGA 处理。

在 USB 数据上传模块部分。单片机开启了两个外部中断，一个是 D12 的中断，一个 FPGA 模块上传数据时的中断。在 FPGA 中设计了一个 FIFO，当数据满的时候触发单片机中断。单片机在 FPGA 中断处理程序中发出读脉冲信号并读取数据。但如果在这个过程 D12 产生中断，则单片机就会进入 D12 的中断处理中，这有可能引起 FPGA 模块发送过来的数据丢失，因此在这里设置 FPGA 模块的中断优先级为最高级别。在向 FPGA 模块发送读脉冲信号读取数据之后，单片机将数据包放入内部的循环缓冲区并设立标志位。当单片机查询到 PC 主机通过端点 1 发送一个上传数据命令时，设立相应的标志位。在主循环中，当单片机同时检测到这两个标志位时，就将数据主通过端点发给 D12 的内部缓冲区，然后由 PC 主机读取[16]。