以下内容依据 STC 官方提供的源码以及 HID 协议规范对程序设计进行讲解，本次程序设计将实现 HID 键盘设备。

15.4.1 设计分层

此次程序设计由于涉及不同的协议规范，因此需要对设计进行分层，以便日后扩展和修改。此次程序设计依据不同协议规范分为以下几层，由于厂商没有定义特定的请求故厂商请求层部分将略去。

（1）USB 层

（2）USB 标准请求层

（3）HID 特定类请求层

（4）定时器层

1. USB 层

该层对应程序设计中的 usb.c 文件，该层存放 USB 初始化函数以及 USB 中断处理函数，USB 中断处理函数中提供了 USB 所有相关中断的的处理函数的入口，其中代码相同部分的分析详见 USB2.0 程序设计实现。

下面讲解 USB 层的不同处。

1）USB HID 键盘的 IN 中断原理

在中断传输前，USB 设备已完成了 SETUP 阶段的内容。在中断传输中，主机会定期轮询一次设备的报告，从而触发设备端点 1IN 的中断。如果触发的是端点 1IN 的中断，则程序会跳转至 usb_in_ep1()函数中，而 usb_in_ep1()就是端点 1IN 的中断处理函数。该中断用于传输键盘码值。

端点 1IN 中断函数的具体内容为：

① 硬件状态设置

将 INDEX 寄存器定位到端点 1；读取 INCSR1 寄存器的内容，依据 INCSR1 寄存器的 INSTSTL 位决定是否复位数据切换位到“0”，如果 INSTSTL 位为 1 表示 STALL 信号发送完成，可以复位数据切换位到“0”；如果对主机的轮询回复的是 NACK 数据包，则清零 INCSR1 寄存器的所有位。

② 软件状态设置

设置 UsbInBusy 为 0（IN 端点空闲，可以传输报告数据）

具体写法如代码清单 15-15 所示。

代码清单 15-15　USB 端点 1IN 中断处理函数的 C 语言代码

```
/************************USB 端点 1IN 中断处理函数*********************/
void usb_in_ep1()
{
    BYTE csr;
    usb_write_reg(INDEX, 1);
```

```
    csr = usb_read_reg(INCSR1);
    if (csr & INSTSTL)
    {
        usb_write_reg(INCSR1, INCLRDT);
    }
    if (csr & INUNDRUN)
    {
        usb_write_reg(INCSR1, 0);
    }
    UsbInBusy = 0;
}
```

2）USB HID 键盘的 OUT 中断原理

与键盘的 IN 中断原理一致，如果触发的是端点 1OUT 的中断，则程序会跳转至 usb_out_ep1()函数中，而 usb_out_ep1()就是端点 1OUT 的中断处理函数。不同的是此中断用于设置键盘状态，例如：当按下 Capslk 按键后，主机会发一个 OUT 包，当 OUT 包接收完毕后，端点 1OUT 中断函数会调用 usb_class_out()函数对开发板上对应的 LED 灯进行设置。

具体写法如代码清单 15-16 所示。

代码清单 15-16　USB 端点 1OUT 中断处理函数的 C 语言代码

```
/*************************USB 端点 1OUT 中断处理函数*********************/
void usb_out_ep1()
{
    BYTE csr;

    usb_write_reg(INDEX, 1);
    csr = usb_read_reg(OUTCSR1);
    if (csr & OUTSTSTL)
    {
        usb_write_reg(OUTCSR1, OUTCLRDT);
    }
    if (csr & OUTOPRDY)
    {
        usb_class_out();          //调用 class_out()函数设置灯的状态
    }
}
```

2. USB 标准请求层

该层对应程序设计中的 usb_req_std.c 文件，该层存放设备对 USB 主机 11 个标准请求的响应函数，具体详见第 3 章。

不同点，若设备接收主机的 Get Descriptor 请求，设备除了需要依据 wVlaueH 状态字发送标准描述符外，还需要发送 HID 类设备特有的描述符。

3. HID 特定类请求层

该层对应程序设计中的 usb_req_class.c 文件，该层除了存放 HID 特定类请求处理函数外，其还存放了发送键盘报告的函数 usb_class_in()，读取主机报告函数 usb_class_out()，以及键盘码值读取函数 scan_key()。

1）类请求函数的参考写法

如表 15.80 类特定请求依据 USB 标准设备请求的数据结构的 bRequest 值来进入具体的请求函数。

表 15.80　类特定请求

请求	bRequest 值	功能
GET_REPORT	0x01	请求允许主机通过控制管道接收报告
SET_REPORT	0x09	请求允许主机向设备发送报告，以设置输入、输出或功能控件的状态
GET_IDLE	0x02	请求读取特定输入报告的当前空闲率
SET_IDLE	0x0A	请求屏蔽中断输入管道上的特定报告，直到新事件发生或经过指定的时间间隔
GET_PROTOCOL	0x03	请求读取当前已激活的协议（例如引导协议或报告协议）
SET_PROTOCOL	0x0B	切换引导协议或报告协议

为了标准起见，本书的类请求函数同样分为三个步骤：1. 特殊情况处理 2. 依据状态发包 3. 结束状态，即进行数据操作（如发送数据，或者接收数据，或者不进行数据操作）。此部分的代码请参考工程 HID 协议范例文件 usb_req_class.c 内容。

① GET_REPORT

表 15.81　GET_REPORT 请求内容

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0xA1	GET_REPORT	报告类型和报告 ID	接口号	报告的长度	报告

第 1 步：特殊情况的处理，与 GET_STATUS 请求的设计思路一致，依旧是特殊情况处理。如果 USB 标准设备请求数据结构不满足表 15.81 所有条件，则挂起。

第 2 步：UsbBuffer 的数据装入端点 0 的数据区，并设置大小为 UsbBuffer 的大小为 wLength 大小。UsbBuffer 的数据用于后续 HID 键盘设备的状态设置。

第 3 步：进入标准请求结束状态（进行数据过程），例如：这时在步骤 2 中软件已经依据 USB 协议内容将 UsbBuffer 数据包装入了数据缓冲区，然后软件再通过调用 usb_setup_in()，完成数据包的发送。

注意：usb_setup_in()的内容是：先将控制端点 0 设置为 DATAIN 状态，控制寄存器 CSR0 的 SPORDY 写 1 以清零 OPRDY，随后调用 usb_ctrl_in()进行数据发送，即使用控制 IN 类型向主机发送数据包。

② SET_REPORT(见表 15.82)

表 15.82 SET_REPORT 请求内容

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x21	SET_REPORT	报告类型和报告 ID	接口号	报告的长度	报告

第 1 步:特殊情况的处理,与 GET_STATUS 请求的设计思路一致,依旧是特殊情况处理。如果 USB 标准设备请求数据结构不满足表 15.82 所有条件,则挂起。

第 2 步:将 UsbBuffer 数据装入端点 0 的数据区,并设置大小为 UsbBuffer 的大小为 wLength 大小。

第 3 步:进入标准请求结束状态(进行数据过程),端点 0 状态设置为 OUT 状态,最后将 CSR0 寄存器的 SOPRRDY 置 1,表示处理完成从端点 0 接收到的数据包。

③ GET_IDLE(见表 15.83)

表 15.83 GET_IDLE 请求内容

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0xA1	GET_IDLE	0 和报告 ID	接口号	1	bHidIdle

第 1 步:特殊情况的处理,与 GET_STATUS 请求的设计思路一致,依旧是特殊情况处理。如果 USB 标准设备请求数据结构不满足表 15.83 所有条件,则挂起。

第 2 步:将数据为 bHidIdle 所指向的数据区域装入端点 0 的数据缓冲区,并设置大小为数据缓冲区的大小为 1 字节大小。UsbBuffer 的数据用于后续 HID 键盘设备的状态设置。

第 3 步:进入标准请求结束状态(进行数据过程),例如:这时在步骤 2 中软件已经依据 USB 协议内容将 bHidIdle 所指向的数据装入了端点 0 的数据缓冲区,然后软件再通过调用 usb_setup_in(),完成数据包的发送。

注意:usb_setup_in()的内容是:先将控制端点 0 设置为 DATAIN 状态,控制寄存器 CSR0 的 SPORDY 写 1 以清零 OPRDY,随后调用 usb_ctrl_in()进行数据发送,即使用控制 IN 类型向主机发送数据包。

④ SET_IDLE(见表 15.84)

表 15.84 SET_IDLE 请求内容

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x21	SET_IDLE	持续时间和报告 ID	接口号	0	无

第 1 步:特殊情况的处理,与 GET_STATUS 请求的设计思路一致,依旧是特殊情况处理。如果 USB 标准设备请求数据结构不满足表 15.84 所有条件,则挂起。

第 2 步:bHiIdle 的状态设置为 wValue 的值。

第 3 步:进入请求结束状态:端点 0 设置成空闲状态,寄存器 CSR0 的 SPORDY 位以及 DATEND 位置位,表示处理完成从端点 0 接收到的数据包,且数据结束。

⑤ GET_PROTOCOL

此次实验不涉及该请求,故不对内容进行编写,若进入该函数,则调用 USB 挂起函数以替代。

⑥ SET_PROTOCOL

此次实验不涉及该请求,故不对内容进行编写,若进入该函数,则调用 USB 挂起函数以替代。

2) 发送键盘报告的函数

先判断 IN 端点是否空闲且矩阵按键是否已完成读入,若已完成则进行译码;但需要注意的是键盘上传的报告数据并不是 ASCII 码,而是键盘的扫描码即 HID 码,因此我们需要将矩阵按键译码为 HID 码,这样主机才正确识别到设备上传的报告值,此处我们仅使用报告的第三个字节即 key[2]来上传键盘 HID 码。

发送阶段:先设置 UsbInBusy 状态字为 1(IN 端点忙碌,正在传输报告数据)。我们需要操作 INDEX 寄存器指向端点 1,后我们将待发送的报告数据以字节为单位发送出去即可。

3) 读取主机报告函数

该函数使用 UsbBuffer 缓冲区从 FIFO0 中提取出主机向设备发送的控制数据,然后使用该数据对实验箱上的 LED 灯进行设置。

4) 键盘码值读取函数

此处详细见矩阵按键程序设计部分,此处我们只需设计一个可以识别矩阵按键的函数即可,在我们调用过该函数后可以通过 bKeyCode 变量读出按键值。

4. 定时器层

该层对应于程序设计中的 timer.c 文件,定时器层中存放了定时器初始化函数,定时器中断处理函数。

定时器层的作用是提供一个固定的时间间隔,STC32 使用该固定时间间隔调用 scan_key()函数对实验箱上的按键进行扫描,并使用 bKeyCode 变量读出按键值。

15.4.2 描述符数据结构

1. 设备描述符

设备描述符 bDeviceClass 字段,bDeviceSubClass 以及 bDeviceProtocol 字段值均固定为 00H,其余字段值依据 USB 规范以及厂商规范进行定义。

2. 配置描述符

依据 USB 规范,USB 主机发送 Get Descriptor 请求获取设备的描述符数据结构,其中配置描述符、接口描述符、HID 描述符、端点描述符是合并为一个数据结构发送给主机的。

1) 配置描述符

配置描述符依据 USB 规范定义方式进行定义。由于仅实现键盘一个功能故 bNumInterfaces 为 1。

2) 接口描述符

依据 HID 协议,接口描述符的 bInterfaceClass 字段值定义为 03H(HID),bInterfaceSubClass 字段值定义为 01H(Boot),bInterfaceProtocol 字段值定义 01H(Keyboard)。

3) HID 描述符

各字段值请参照 HID 描述符一节进行定义。

4) 端点描述符

由于端点 1 既要使用 IN 端点又要使用 OUT 端点,因此需要用两个数据结构分别定义端

点 1。端点 1 的数据传输模式依据协议需要定义为中断类型的端点。因此端点 1IN 的 bEndpointAddress 字段值定义为 81H,端点 1OUT 的 bEndpointAddress 字段值定义为 01H,两个端点描述符结构中 bmAttributes 字段值都为 03H,其余字段依据 USB 规范定义方式进行定义。

3. HID 报告描述符

HID 报告描述符中所使用的项均为短项目。此部分建议读者配合 HID 报告描述符一节对其内容进行理解:

例如 USAGE_PAGE(Generic Desktop)项,该项属于短项目中的全局项,通过全局项的一字节前缀的表格可以知道,该项的第 0 个字节为 0000 01nn(b)其中 nn 为后面跟随的数据字节的数量,故 nn 为 01,该项的第 1 个字节则是确定 USAGE_PAGE 项的类型,该部分需要对照 USB 官方文档 HID usage table1 进行查看。

4. 厂商描述符

此处依据厂商标准对该描述符进行定义。

15.4.3 例程使用说明

下载程序至实验箱,在主机上打开文件编辑器,定位好光标,分别按下实验箱上的矩阵按键,光标输出内容;按下主机的大写锁定键,发现实验箱上对应的指示灯亮起。

拓展:请依据 HID 协议内容,使用实验箱模拟一个 HID 鼠标设备。