基于 ARM 的 DCS 专用工业键盘研究与实现

蒙 艳,孙旭华,姜铁梅
(上海自动化仪表股份有限公司技术中心,上海 200072)

摘 要:工业控制现场中的分散控制系统(DCS)需要特定的 DCS 操作员键盘对其进行控制。研究了基于 ARM 微处理器的 DCS 专用工业键盘,设计了具有 USB2.0 通信协议的标准 HID 类键盘接口程序以及具有标准 PC 键盘和 DCS 功能扩展键盘功能的键盘任务处理程序。实验结果表明,设计合理、系统稳定可靠,能满足工业现场环境的需要。

关键词:DCS;键盘;USB;ARM;STM32
中图分类号:TP334 **文献标识码**: **文章编号**:1674-7720(2012)09-0078-04

Research and realization of the special keyboard in DCS system based on ARM

Meng Yan, Sun Xuhua, Jiang Tiemei
(Shanghai Automation Instrumentation Co., Ltd., Shanghai 200072, China)

Abstract: With the development of DCS system in the industry control field, the special keyboard for DCS is required. This papar focuses on the research of the special keyboard based on ARM MCU. It illustrates the design of USB control interface as the standard HID keyboard, and how to realize the keyboard function of standard PC and the special function espended by DCS. Experimental results show that the design is reasonable and the system is stable and reliable. It can meet the industry field environment.

Key words: DCS; keyboard; USB; ARM; STM32

随着工业控制现场中分散控制系统(DCS)的发展,需要特定的 DCS 操作员键盘对系统进行控制。该键盘使操作员能对系统的操作更加直观、操作方法更加简捷,以降低对操作员的操作技能要求,进一步减少误操作的可能,从而提高整个生产线的自动化能力。DCS 操作员键盘正是基于这个目的产生的。该键盘在功能上除了兼容标准 PC 键盘的功能外,增加了 DCS 功能扩展区和用户自定义扩展区。在该键盘按键壳体上印刷按键功能,如逻辑开、逻辑关、手动、自动、报警、输出步长等等,一旦有按键动作,键盘立刻响应并将按键编码发给主机,主机根据接收到的按键编码调用相应的子函数来完成响应功能。同时该键盘支持目前广泛流行的 USB 接口的标准 PC 键盘通信协议。USB 协议专门为人机交互设备提供了接口描述,即 HID 设备类。用户可以按照 HID 设备类的协议设计通用键盘,也可以根据自己的需要设计特殊的键盘,以满足不同的应用场合。USB 通信协议的设备一旦接入主机 USB 接口,主机调用底层驱动自动完成 USB 设备的枚举,实现方便快捷的即插即用。另外该键盘可记挂标准 PC 键盘,即便在 DCS 操作员键盘故障或在高级工程师操作管理模式下,可使用标准 PC 键盘做进一步的操作。在基本功能上与标准 PC 键盘保持一致,可相互控制 Capslock 状态。本文描述的正是这样一个基于 ARM 的 DCS 专用工业键盘的研究与实现。

1 系统硬件实现

该键盘硬件主要由 ARM MCU、USB HUB 等控制电路以及矩阵式键盘组成。

1.1 ARM MCU 控制电路和矩阵式键盘

ARM MCU 控制电路主要负责与主机的通信,完成 USB 通信建立、按键扫描、键码确认以及发送。ARM 其他部分电路包括 JTAG 调试电路接口、复位模块、晶振介入电路等常规电路。另外还有一个 GPIO 端口做指示灯,用于指示键盘 CAPS 键。

本设计微处理器选用 ST 公司的 STM32F103R6T6。

STM32F103R6T6 是 ARM 公司具有突破性的 Cortex-M3 内核的 STM32 系列 32 bit 闪存微控制器的增强型产品。具有高性能、低功耗、实时应用且具有竞争力价格等多项优点，其工作频率为 72 MHz，1.25 DMIPS/MHz；片上集成了 32 KB 的 Flash 代码存储器和 10 KB 的 SRAM 用户数据存储器；通过 APB 总线连接丰富和增强的外设和 I/O；集成 3 个定时器、51 个快速 I/O 端口以及 SPI、I^2C、USART、CAN 等多种标准通信接口[1]；嵌入了一个支持全速 USB2.0 总线的 USB 外设，实现了全速(12 Mb/s)功能接口；可通过软件配置端点，也可以通过软件控制挂起/恢复；时钟来自内部 PLL 产生的 48 MHz 专用时钟源。

该电路主要用于检测键盘矩阵上的按键情况，并处理包括特殊功能键在内的数据传输，具体功能由软件编程实现。本设计中键盘需要支持 86 个按键，接口部分为矩阵式键盘，列线通过电阻接正电源，用 MCU 的 11 个 I/O 口做通用输出口，作为键盘扫描信号的输出口；行线用 8 个 I/O 口做通用输入口，作为键盘反馈信号的输入口。这样总共使用 19 个 I/O 端口即可控制多达 11×8=88 个按键，减少了 I/O 口的占用，满足了 86 个按键的需要。将全部列线置低电平输出，然后读行线有无低电平出现。当没有按键按下时，所有的输入端都是高电平，代表无键按下；一旦有键按下，则输入线就会被拉低，这样通过读入输入线的状态就可得知是否有键按下。

此外本项目采用 IAR System 公司为 ARM 微处理器开发的一个集成开发环境 IAR EWARM，需要配套的 IAR J-LINK 仿真器。J-LINK 一端通过 PC 机 USB 口与 PC 连接，另一端通过标准 20 芯 JTAG 插头与目标板连接，并将目标板的电源接上，即可进行应用程序的在线调试。

单片机控制设计电路如图 1 所示。

1.2 USB HUB 部分

USB HUB 控制电路主要是为该键盘作冗余，将 USB 数据分成两路，一路接收和发送键盘的数据，另一路作为独立的 USB 口，可与其他的标准 USB 2.0 设备通信(如标准 PC 键盘)，以防工业现场恶劣环境下键盘故障。USB HUB 芯片采用赛普拉斯的 CYC765621，主要接口是 1 路和上位机通信的 USB 接口(D-、D+)，另 2 路为 HUB 分出来的 2 路 USB 接口，其中一路作为外接 USB 口，另一路则接到 ARM 作为键盘通信的数据传输口(DD1-、DD1+、DD2-、DD2+)。另有 MIC2026-2YM 的电源

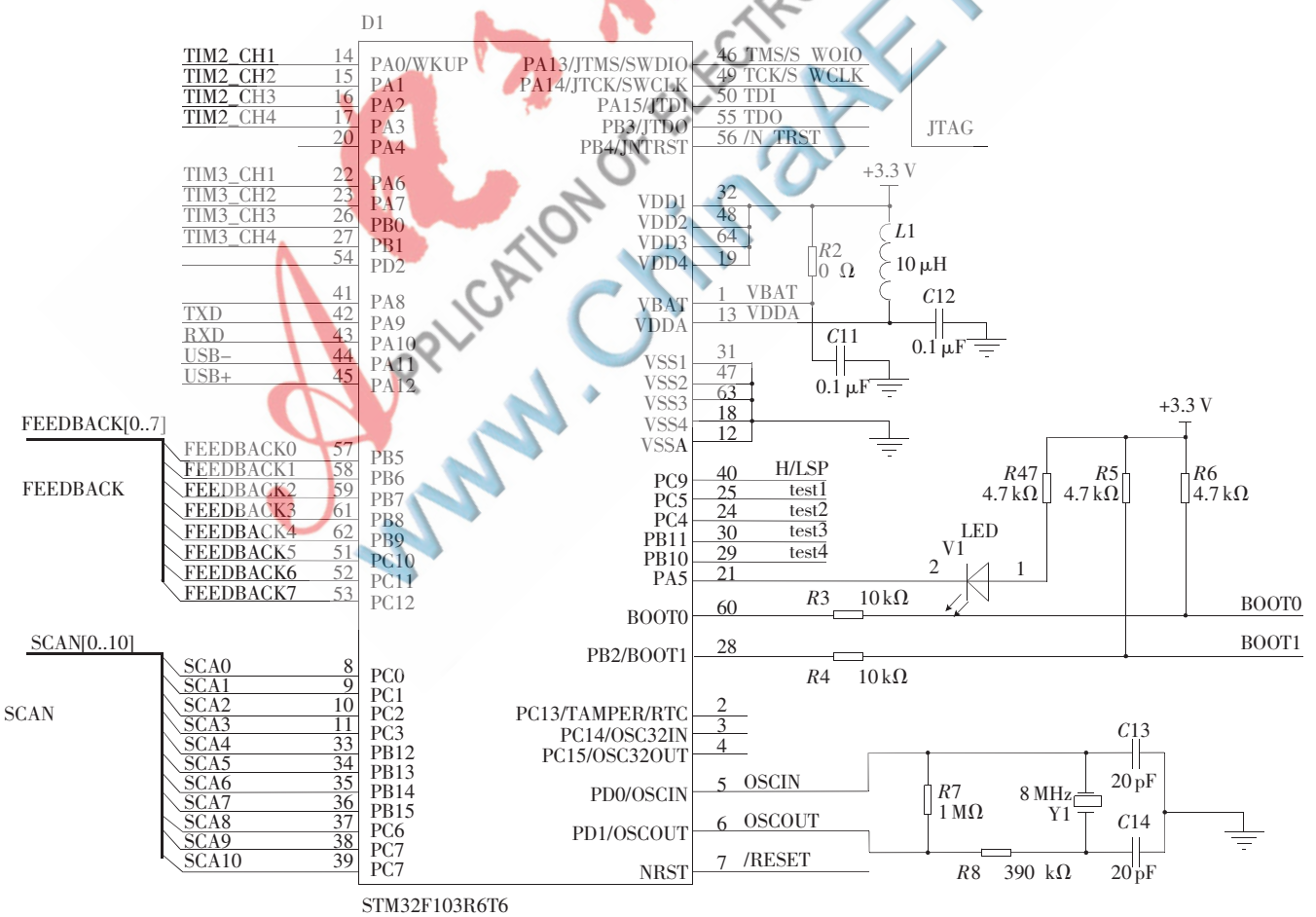

图 1 单片机控制电路

保护关断芯片，作为保护电路与 CYC765621 相连。

1.3 电源部分电路

电源部分电路采用 USB 供电。USB 为 5 V 供电，通过 LM1117MPX-3.3 的线性电源芯片转换为 3.3 V 电源，为 USB HUB 和 ARM 芯片供电。

2 系统软件设计

本设计中软件主要是对硬件电路的驱动，实现与 USB 口的通信以及键盘的响应功能。首先对 ARM MCU 进行系统初始化，包括系统时钟设置、GPIO 口的初始化设置；其次对 USB 模块中断配置与使能，中断包括 USB 低优先级中断、USB 唤醒中断、按键控制中断等，并对 USB 模块时钟设置和使能，然后初始化。对定时器 TIM2 进行设置，之后进入循环任务函数。循环任务函数有两个：一是主机枚举响应；另一个是使用定时器中断方式实现键盘任务。键盘主程序流程图如图 2 所示。

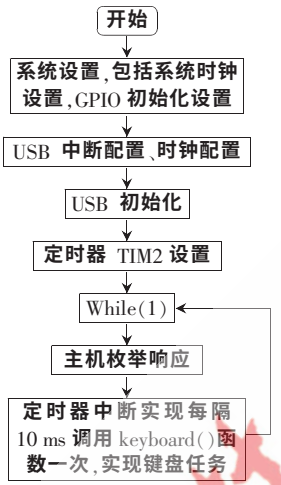

图 2 键盘主程序流程图

2.1 USB 任务处理设计

USB 键盘是 HID 类设备的一种，遵循着 USB 设备启动过程。即每次设备接入总线，先进入设备检测状态，总线对设备进行总线复位；其次是设备枚举过程，在这里 USB 设备将被枚举为标准的 HID 类键盘。主机通过默认端点 0 发送 SETUP 包，得到设备描述符，然后给设备分配新的地址，分配成功后，主机将通过新的设备地址向设备依次发送标准 USB 请求来获取设备的配置信息，即获得设备描述符、配置描述符、接口描述符、端点描述符、HID 描述符和报表描述符。通过设备的这些描述符，主机就知道了设备的详细信息，并根据设备的配置情况对设备的端点进行操作。这些操作包括：初始化端点数目，分配各端点所需使用的 Packet Buffer；初始化所使用的端点，配置端点的传输类型、传输方向、Packet Buffer 地址和初始状态。在需要发送数据或接收数据的时候，使能端点；并在该端点的中断回调函数中处理数据，如果需要则使能下一次传输。以上便是实现 USB 键盘设备的步骤。

USB 设备描述符如下：

```
Keyboard_DeviceDescriptor[]=
  {
    0x12,                                   /*bLength*/
    0x01,                                   /*bDescriptorType*/
    0x00,0x02                               /*bcdUSB*/
    0x00,                                   /*bDeviceClass*/
    0x00,                                   /*bDeviceSubClass*/
    0x00,                                   /*bDeviceProtocol*/
    0x40,                                   /*bMaxPacketSize 64*/
    0x88,0x88                               /*idVendor*/
    0x16,0x10                               /*idProduct=0x1016*/
    0x00,0x02                               /*bcdDevice rel. 2.00*/
    1,          /*Index of string descriptor describing
                                            manufacturer*/
    2,  /*Index of string descriptor describing product*/
    3,          /*Index of string descriptor describing the
                                            device serial number*/
    0x01                                    /*bNumConfigurations*/
  }
```

在实际设计与开发中，由于 STM32 提供丰富的 USB 标准函数库，充分使用该函数库会加快开发进程。下面是一段对 IN 端点的初始化和使能以及 IN 传输的关键代码：

```
/*Initialize Endpoint In 1*/
SetEPType(ENDP1,EP_INTERRUPT);
SetEPTxAddr(ENDP1,ENDP1_TXADDR);
SetEPTxCount(ENDP1,8);
SetEPTxStatus(ENDP1,EP_TX_NAK);

/*Copy Keyboard value in ENDP1 Tx Packet Memory Area*/
USB_SIL_Write(EP1_IN,Keyboard_Buffer,8);
/*Enable Endpoint In 1 for transmission*/
SetEPTxValid(ENDP1);

/*EPx_IN_Callback*/
EPx_IN_Callback(void);
USB_SIL_Write(EP1_IN,Keyboard_Buffer,8);
SetEPTxStatus(ENDP1,EP_TX_VALID);
```

2.2 键盘任务处理设计

键盘任务处理函数 keyboard () 流程图如图 3 所示。先对按键进行扫描，若无按键按下，则每隔 10 ms 扫描一次。若有按键按下，记录下按键所处键盘的位置代码，转向消抖处理。所按下按键若通过消抖处理，则确认其位置代码，进入到按键处理阶段，将位置代码转换为按键 HID 码并发送。否则释放其位置代码，转到按键扫描处重新扫描。发送完其 HID 码的按键再进行消抖处理判断其释放否。

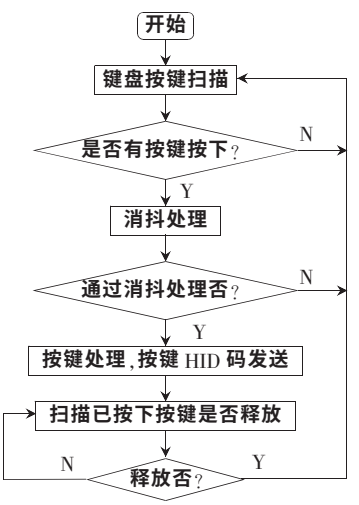

图 3 键盘任务处理函数 keyboard()流程图

若按键未释放则继续等待，若已释放则转向按键扫描处重新扫描。

在具体设计中引入状态转移分析法和定时器中断。将按键的所有状态分为 4 种：状态 0 为按键扫描，状态 1 为按键消抖处理并确认，状态 2 为按键键码转换并发送，状态 3 为等待按键释放状态。以上的函数流程实际上是在这 4 个状态中转移，如图 4 所示。设置一个状态标志位 key_state 来表示按键所处的不同状态，采用多分支结构 swich-case，可以很方便地实现。

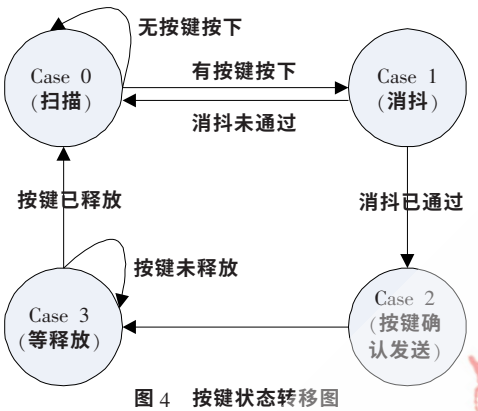

图 4 按键状态转移图

定时器中断是使 MCU 的 TIM2 定时器产生 10 ms 的定时中断。主程序每隔 10 ms 中断调用 keyboard()函数一次。当无按键按下时，key_state=0，执行 case0，即实现了每 10 ms 执行一次键盘扫描。若有按键按下，则定位其按下按键的位置代码，并使 key_state=1。当 10 ms 后再次调用 keyboard()函数，这时 key_state=1 而进入 case1，在 case1 中对前面所定位的按键位置再次进行确认，若还在，则再次确定其位置代码，并使 key_state=2。如此时按键已经释放，说明为假按键，则使 key_state=0，返回按键扫描。这里巧妙利用了两次定时中断的 10 ms 间隔实现按键的消抖处理。若 key_state=2，10 ms 后再次调用 keyboard()函数时则进入 case2，这里将已确认的按键位置代码转化为按键代码以及最终的 HID 码，建立键盘报表并发送给主机，然后使 key_state=3。当下一个 10 ms，key_state=3 则进入 csae3，等待按键释放状态，此时再次扫描前面已确定按键的位置，若按键已释放，则 key_state=0，下一个 10 ms 来临则进入 case0 重新按键扫描；否则仍然 key_state=3，继续等待释放。这里也巧妙地利用了这 10 ms 进行按键释放时的去抖动处理。

另外在多键（含双键）、特殊功能键和复合键的实现中，该软件也设计了比较好的实现方法。譬如多键，设置内部缓存器，在逐行扫描中将每个按下按键在矩阵键盘中的位置代码存入其中。之后的消抖处理等操作的对象便是缓存器中的按键位置代码值。系统处理的普通键数最多为 6 个，超过则为溢出。对于特殊功能键，其形式上是单键，实际实现的是多键的功能。只要将单键在发送前转换成需要的多键 HID 码，即可方便实现。对于复合键，理论上是两个以上按键同时按下所完成的功能，实际情况很难实现真正的“同时按下”，它们的时间差别可能长达 50 ms。譬如对引导键 SHIFT 键的设计，需设置专门标志位 Shiftkey_flag，有 SHIFT 按下则 Shiftkey_flag=1，否则为 0。在第一次检测出 SHIFT 单按键时，改变状态标志位为 1。随后的按键扫描中，若再无其他按键按下，则 key_state=0，直到有其他按键按下，key_state=1。这在软件中实际上是对状态 0 的细化。同理，在各个状态中，因为该键的特殊性，也有相应的细化过程。

从以上分析可见，这样的软件设计不仅结构清晰，代码简洁、实现便捷，而且使得当无按键按下，键盘每 10 ms 扫描一次；当有确定非复合按键按下（即通过消抖处理），则键盘响应速度在 30 ms 以内，如此的响应速度大大提高了键盘的灵敏度。

DCS 系统是目前工业控制领域的核心系统，其专用键盘是提高整个生产线自动化能力的关键一环。该专用键盘的设计避免了现有键盘电路特殊按键实现复杂、软硬件成本高的情况。本文研究开发的基于 ARM 的 DCS 专用工业键盘，实现了对多达 86 个按键的控制。采用 STM32 芯片及有效率的软件开发大大提高了产品开发的速度。经过实验结果与实际应用证明，该专用键盘易用性、可靠性达到了工业要求，并可产生很大的经济效益。

参考文献

[1] 王永虹，徐炜，郝立平.STM32 系列 ARM Cortex-M3 微控制器原理与实践[M].北京：北京航天航空大学出版社，2008.

[2] 马潮.AVR 单片机嵌入式系统原理与应用实践 [M].北京：北京航天航空大学出版社，2007.

[3] 刘荣.圈圈教你玩 USB[M].北京：北京航天航空大学出版社，2009.

[4] 廖济林.USB2.0 应用系统开发实例精讲[M].北京：电子工业出版社，2006.

（收稿日期：2011-12-08）

作者简介：

蒙艳，女，1976 年生，工程师，主要研究方向：控制系统，嵌入式系统。

孙旭华，男，1984 年生，工程师，主要研究方向：FPGA，控制系统。

姜铁梅，女，1973 年生，高级工程师，主要研究方向：控制系统，现场总线。