第 33 卷 第 11 期 计 算 机 工 程 2007 年 6 月
Vol.33 No.11 Computer Engineering June 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)11—0254—02 文献标识码：A 中图分类号：TP334

Qt/Embedded 环境下嵌入式键盘驱动的实现

张 萍，徐 晶
(华中科技大学电子信息与工程系，武汉 430074)

摘 要：描述了基于嵌入式 Linux 和 Qt/Embedded 的手持终端的键盘驱动的设计与实现。依据嵌入式手持设备键盘的工作特点及其驱动实现难点，采用了一种将键盘驱动分解成底层驱动模块和上层文本输入模块的双层设计方案，且在实际设备中予以实现。
关键词：嵌入式终端；Linux；Qt/Embedded；键盘驱动；文本输入

Implementation of Embedded Keyboard Driver in Qt/Embedded Environment

ZHANG Ping, XU Jing
(Electronic and Information Department, Huazhong University of Science and Technology, Wuhan 430074)

【Abstract】 This article discusses the design and implementation of keyboard device driver based on embedded Linux and Qt/Embedded. Based on the working process and keyboard driver characteristics, the article divides the keyboard driver into two modules: bottom device driver module and upper text input module.
【Key words】 Embedded device; Linux; Qt/embedded; Keyboard driver; Text input

嵌入式系统(Embedded System)无疑是当今最热门的概念之一。在嵌入式手持设备中，高性能、高可靠性 GUI 的支持显得越来越重要。就现有的发展趋势看，越来越多的大型企业和组织开始选用 Qt/Embedded，包括 Motorola、Sharp 和 IBM 等。在中国，作为 TD-SCDMA 领头企业之一的大唐，也已经选择了 Qt/Embedded 作为开发基于 Linux 的移动电话的开发平台和图形用户界面。

作为一种和图形用户界面的交互手段，同时，也作为一种重要的数据输入方式，嵌入式键盘也起着举足轻重的作用。由于嵌入式系统具有功耗低、体积小、专用性强等特点，因此要求嵌入式键盘具有特殊的工作方式和特定的驱动设计。

本文通过系统地研究键盘工作原理，以及Linux下在不同体系之间键盘驱动的接口和Qt/Embedded下键盘驱动的接口[1~5]，最终完成了嵌入式手持设备的键盘驱动的双层设计与实现。这种双层模块的设计方案，不仅取消了扫描码到键盘码、符号码的转换，同时也消除了对键盘模式的判断。

1 普通键盘驱动的不足

普通键盘一般工作在 UNICODE 模式，并且应用程序也都接收符号码。这样一来，一个按键动作的产生，将要经过至少 2 次模式判断和 2 次码值变换，这对于一个嵌入式键盘而言是极其无效。因为用于表示一个基本按键的键盘扫描码由 1 个字节的接通扫描码和 2 个字节的断开扫描码组成，单单一个键的按下与断开，键盘最多要产生一系列多达 6 个字节的扫描码序列。再经过 2 次的模式判断和码值变换，不仅会降低整个系统的响应速度，也会占用更多的系统资源。资源有限性是嵌入式系统最基本的特征，因此这样的键盘及驱动是无法在嵌入式手持设备上使用的。

嵌入式系统的资源有限性也表现在对物理体积的最大压缩上。为了限制键盘体积，26 个英文字母按字母表顺序 3 个或 4 个一组依次排列在 2～9 这 8 个数字键上，并与阿拉伯数字进行复用。这样每个按键同时表示了 4～5 个包括英文字母和数字在内的不同的键值。那么，原来的键盘驱动的处理过程，尤其是扫描码到键盘码、再到符号码的转换过程会变得极其复杂，因为这时出现了同一个扫描码对应多个键盘码和符号码的情况。这样势必会造成大量不必要的系统开销。

2 嵌入式键盘驱动的双层设计思想

本文描述的嵌入式键盘驱动采用了底层驱动模块和上层文本输入模块的双层模块设计方案。与普通键盘不同的是，嵌入式系统硬件平台上并没有键盘扫描芯片，本文采用了复杂可编程逻辑器件(CPLD)。CPLD 键盘逻辑主要提供扫描、消抖和编码的硬件功能。在双层模块设计方案中，底层驱动模块只负责简单的读取 CPLD 板级寄存器的扫描码，而不经过键盘码、符号码的转换，直接传递给上层应用程序。并且由 CPLD 板级寄存器提供的经过编码的扫描码只有 5 位有效值，远远少于原来的 3 个字节，节省了系统资源。同时取消键盘模式的判断，提高键盘驱动的处理效率。

而作为双层模块设计方案中的上层文本输入模块将实现扫描码到图形用户接口 Qt/Embedded 中 Qt 键值的直接转化和文本输入的过程。这一直接转化过程将实现 10 个数字按键可以输出 10 个阿拉伯数字，26 个大写英文字母，26 个小写英文字母及常用标点符号等 70 多个不同符号。

3 嵌入式键盘驱动的双层设计实现

3.1 底层驱动模块设计实现

键盘的初始化是由 TTY 设备的初始化开始的。由于 TTY 设备在初始化时已经注册了相应的设备文件，因此，键盘的

基金项目：国家“863”计划基金资助项目(2005AA420050-05)
作者简介：张 萍(1982－)，女，硕士生，主研方向：嵌入式系统和图形用户界面；徐 晶，博士生
收稿日期：2006-06-25 **E-mail：**lyneausten@yahoo.com.cn

初始化不需要注册键盘设备，而只要进行中断请求号的申请。

CPLD 通过行和列信号的序列识别按键行为，并对按键进行 5 位有效值的编码，然后存放到 CPLD 板级寄存器中。键盘底层驱动程序通过板级寄存器的地址操作读取扫描码，把所获取的扫描码放入 tty_flip_buffer，等候 tasklet 的底半部处理。

键盘底层驱动程序读入扫描码后的中断处理过程如图 2 所示。整个过程并没有扫描码到键盘码、符号码的转换，同时也没有键盘模式的判断。

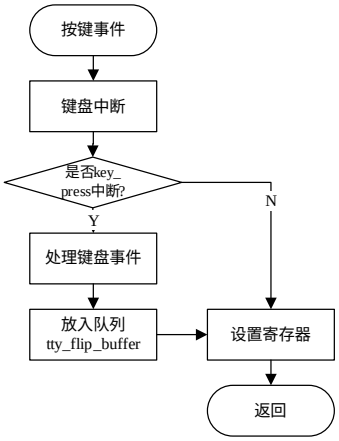

图 1 键盘驱动程序的中断处理

3.2 上层文本输入模块设计实现

对于上层文本输入模块，通过从 QWSPC101KeyboardHandler 中继承出子类 QWSWlitKeyboardHandler，并在 QWSServer::newKeyboardHandler()中给出实例的过程来实现键盘扫描码到图形用户接口 Qt/Embedded 中 Qt 键值的直接转换。同时利用 Qt 的信号与插槽机制将键盘设备文件信号与 QWSWlitKeyboardHandler 类中的插槽 readKeyboardData()连接起来。

为了实现文本输入，上层文本输入模块定义了输入状态及按键次数。用于表示输入状态的变量 input_state 有 4 个不同的取值：数字输入状态；小写英文输入状态；大写英文输入状态；标点符号输入状态。利用功能键星号键(*)实现这 4 种状态的切换。计算连续按同一个键的次数是由成员函数 HandleSameKey()实现的，如果连续 2 次按键时间的间隔在设定的域值以内，就判断为连续按键 2 下。

图 2 示意了函数 HandleSameKey 的实现流程。该函数首先判断是否是同一个键被按下，如果不是，则该键的按键次数为 1 次；如果是同一个键，还需要判断第 2 次按键与第 1 次按键的时间间隔是否小于设定的域值，如果不是，则该键的按键次数也为 1 次，但是输入了 2 次。换言之，只有在设定的域值以内按同一个按键，按键次数才自增 1。当按键次数自增到比复用的英文字母个数还多的时候，就循环归置按键 1 次。需要注意的是数字键 7 和数字键 9，在这两个键上都复用了 4 个英文字母，而在其它数字键上只复用了 3 个英文字母。

文本输入模块的实现还需要考虑一个符号分割的问题。当后续输入的符号和前一个已经输入的符号在同一个按键上，比如单词“ON”，字母 O 和 N 都在数字键 6 上。为了正确的输入单词“ON”，可以定义一个时限。连续按数字 6 键 3 次输入字母“O”，其中连续按键的时间间隔不超过定义的时限；然后系统等待该时限超时，再连续按数字 6 键 2 次输入字母“N”。另一个解决符号分割的方法是定义一个专门的按键来跳过时限限制，这样就可以不用等待时限超时直接在同一个按键上输入下一个符号。本文实现的键盘驱动的文本输入模块同时实现了这两种方法。不仅定义了符号分割的时限，还使用了光标右移键来跳过时限限制。这样用户可以按照自己的习惯选择使用。

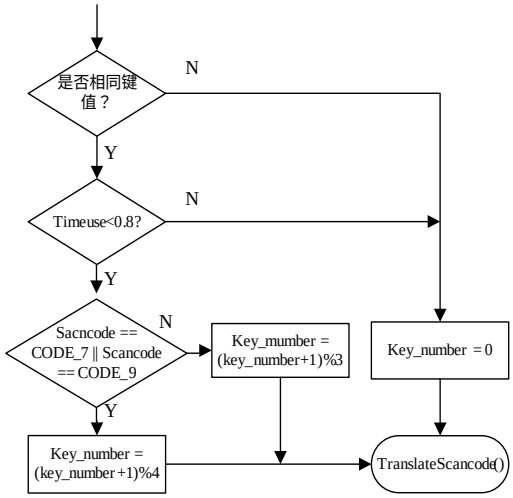

图 2 HandlleSameKey 实现流程

在得到了输入状态和连续按键次数后，根据这 2 个变量的组合将扫描码直接转化为 Qt 键值。这一过程是由成员函数 TranslationScancode 来实现的。该函数的实现流程如图 3 所示。星号键用来切换输入状态，所以该函数将星号键与其它键分开处理。然后根据输入状态和连续按键次数的 10 种不同组合，将扫描码转换成对应的 QWSKeyEvent 并传递给静态函数 QWSServer::processKeyEvent()进行下一步的处理。

而处理的过程会根据标志位 first_press 的值有所不同。标志位 first_press 表示按键是否首次被按下，首次按下是相对于循环归置而言的。对于首次被按下的键，直接调用 processKeyEvent()函数输出对应的符号，将在光标处输出对应符号，并使光标右移；但当按键不是首次被按下时，先调用 processKeyEvent()函数输出 Qt::backspace，再调用 processKeyEvent()函数输出对应的符号，这样才能在原光标处输出对应符号，并使光标右移。

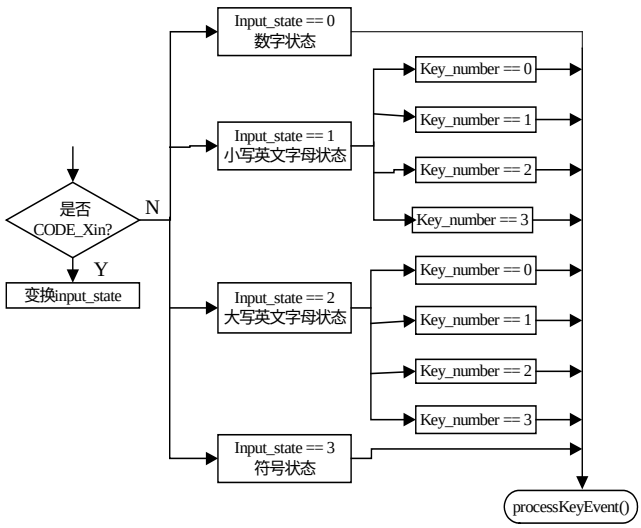

图 3 函数TranslateScancode实现流程

（下转第 258 页）