嵌入式浏览器 Konqueror/ embedded 的技术分析

■北京航空航天大学 蒋文军 张晓林 崔迎炜

摘 要 分析嵌入式浏览器 Konqueror/ embedded 实现的关键技术;讨论 Kparts 组件技术、I/ O - Slave 机制以及 DOM 技术在 Konqueror/ embedded 中的运用;结合 Konqueror/ embedded 在以 Motorola i. MX1 为核心的嵌入式系统中移植和开发的问题,讨论 Konqueror/ embedded 的移植和汉化技术。

关键词 Kparts I/ O - Slave DOM 移植 汉化

1 Konqueror/ embedded 概述

目前,嵌入式浏览器已经逐渐成为高端手机和 PDA 的标准配置。已经面市的大多数嵌入式浏览器是商业版本的,像 opera 和 MS explorer 等,而 Konqueror/ embedded 是符合 GNU 条款的自由软件。Konqueror/ embedded 是针对嵌入式 Linux,由著名的桌面操作环境 KDE 下的浏览器 Konqueror 派生出来的。Konqueror/ embedded 将 Konqueror 中关于 KHTML、SSL、Javascript 等内容继承了下来,同时简化了 Konqueror 中很多类的定义,剔除了依赖于 KDElib 部分,以适应在不同的嵌入式平台上移植和运行。两者都是基于 Qt 的,因此 Konqueror/ embedded 也可以运行在 Qt/ X11 环境下。Konqueror/ embedded 完整地支持 HTML4 和 css1 (部分支持 css2)、JavaScript (ECMAScript 262)、cookies、SSL、IPv6;支持和管理兼容 XBEL 的书签,并且能够很好地支持中文网页浏览;可以将 Konqueror/ embedded 作为一个 flashplayer、pppdialer 或文件管理器使用。

2 Konqueror/ embedded 的构成

Konqueror/ embedded 的层次结构如图 1 所示。

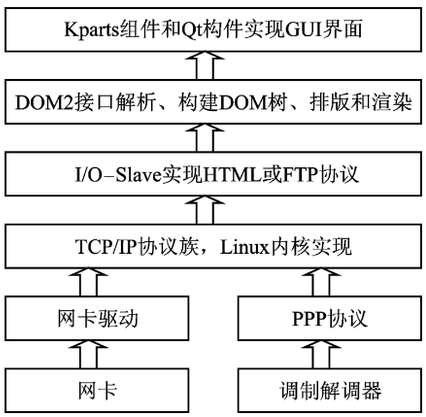

图 1 Konqueror/ embedded 层次结构

Konqueror/ embedded 是由底层网路连接、图形化用户界面和处理 HTML 绘制的引擎——KHTML 构成的。

在图 4 的基础上,就可以根据具体的 RTOS 比较方便地进行软件的开发了。

结 语

基于 RTOS 编程虽然增加了系统硬件开销,但随着现代半导体工艺的迅速发展,使得硬件的性能在不断地提升而成本在不断地下降。这就意味着作为一个嵌入式系统工程师,需要关注更多的是软件开发。面向对象开发具有减少费用、开发产品所需时间短的优势,并在开发过程中具有灵活性。

参考文献

1 陈明计,周立功. 嵌入式实时操作系统 Small RTOS51 原理及应用. 北京:北京航空航天大学出版社,2004
2 Labrosse Jean J. 嵌入式实时操作系统 μC/ OS - II. 北京:北京航空航天大学出版社,2003
3 Richard C. Lee William M. Tepfenhart C + + 面向对象开发. 北京:机械工业出版社,2002

(收稿日期:2004-11-23)

底层的通信协议实现是基于 KIO/ Slave 机制来实现的;GUI 界面采用了 Kparts 组件技术和 Qt 中的基本构件;而作为 Konqueror/ embedded 的核心——KHTML,则运用了文档对象模型(DOM)所提供的 API 接口,并在 DOM 树上挂接 javascript 引擎、CSS 解析器以及渲染引擎。

3 Konqueror/ embedded 中的关键技术

3.1 底层通信协议的实现——I/O - Slave 机制

在 KDE 中采用 I/O - Slaves 系统来访问各种数据。Konqueror/ embedded 沿用这种方式,通过 I/O - Slaves 和进程间管道通信机制实现完整的浏览器功能。在基于 Qt/ X11 的 Konqueror 中,同样采用 I/O - Slaves 机制。简单说来,I/O - Slaves 就是那些从网络上获取文件或目录的进程,它们依赖桌面通信协议 DCOP(Desktop Communications Protocol)与其他进程进行通信;但是,DCOP 的实现又依赖于 X11 ICE(Inter Client Exchange)库。在嵌入式平台上移植体积庞大的 X11 lib 是不现实的。Konqueror/ embedded 采用了另外一种进程间的通信机制:通过管道(pipe)实现主进程和其他 I/O - Slaves 子进程之间的通信。

在 KDE1.X 之后,KDE 的文件管理器和 Konqueror 等应用程序具有网络透明的特征,Konqueror/ embedded 也继承了这个特性。Konqueror/ embedded 不管对本地文件还是远程文件都采用 URL(统一资源定位符)进行标识。网络透明性允许应用程序的用户使用与处理本地文件相同的方法来处理远程文件。在 KDE 的文件选择器能够通过诸如 FTP、SMB 甚至 Webdav(在 KDE3 中)等方式列出远程目录。网络透明性和 I/O - Slaves 机制是靠 KIO 类来实现的。KIO 类提供了几乎所有的文件管理功能。其中,KIO::NetAccess 提供文件下载、上传以及临时文件的创建或删除等简单的同步访问功能。这是一种阻塞调用方式。如果用 Konqueror/ embedded 下载网络上的数据,在数据没有完全加载之前,当前进程将会被阻塞。数据流完全加载之后,网页才可以再次渲染或刷新。KIO::Job 提供较为复杂的异步功能,包括打开、创建、复制、删除以及重命名等与文件或目录相关的操作。Konqueror/ embedded 正是依靠 KIO::Job 来实现网页访问这样的异步工作的。一旦某个 job 被启动之后,它将运行在后台,并且不会阻塞父进程。Konqueror/ embedded 中各种协议,如 HTTP、FTP 等是由一些独立的进程来实现的。这些独立的进程被称为 Slaves。Slave 是 KIO::Slavebase 的子类,KIO::Slavebase 中定义了一些虚函数,不同的 Slave 必须重载这些接口。Slave 以函数库的形式存在于系统中,提供给相应的 job 调用。Slaveinterface 运行在应用端(job),Slavebase 运行在 Slave 端。一般情况下,创建的 job 会处在队列中,当事件循环处理到该 job 时,KIO 为队列中的 job 分配适当的 Slave。Slave 的管理和调度是由调度器 scheduler 来执行的,scheduler 将 job 队列加入适当的协议(Slave)中。当该 job 结束之后,Slave 进程不会立即停止,会在空闲区域中等待一定的时间。这种机制的优点在于,如果有几个 job 访问的是同一个主机,那么就不需要重新启动新的进程,并且不需要再次进行协议握手。在同一时间,面向同一种网络协议,系统中最多只能启动 3 个 Slave 进程。如果引用该协议的 job 超过这个数目,那么多余的 job 将被添加到队列中,直到又有空闲的 Slave 进程可用。图 2 描述了 Konqueror/ embedded 中关于底层协议(Slave)和任务(job)的实现过程。

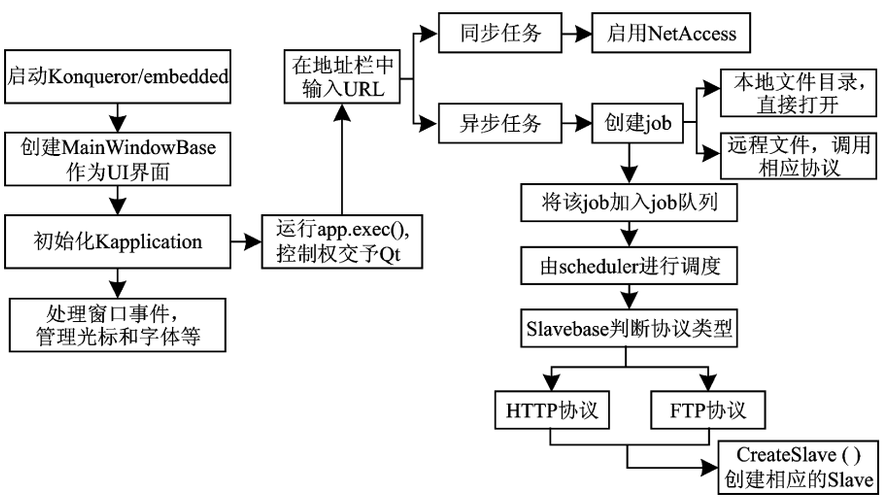

图 2 I/O - Slaves 实现流程

3.2 GUI 系统的实现——Kparts 组件技术

(1) Konqueror/ embedded 中的组件

虽然,Konqueror/ embedded 的 UI 界面是直接继承 QMainWindow 的,但是,UI 界面的功能和布局是由组件来实现的,QMainWindow 只是这些组件的宿主。在 Konqueror/ embedded 启动初始化过程中,首先加载一个特殊的组件——KHTMLPART。KHTMLPART 负责其他组件(如 Kflashpart、Kplaintextpart 等)的加载和管理。采用 Kparts 组件编程技术能够使得 Konqueror/ embedded 的功能得到扩展,而不需要重新修改底层代码,同时增强了 Konqueror/ embedded 的可定

制性。Konqueror/embedded 将 Web 浏览器、flsahpalyer、文本编辑器和简单音频播放器都作为 Kparts 组件嵌入到主窗口中。Kparts 组件编程技术能够通过将图形组件嵌入应用程序的窗口使需要同一功能的应用程序共享一个组件。Kparts 组件分只读组件和读写组件。只读组件 ReadOnlyPart 类为实现任何类型的查看器提供了一个公共框架。如果提供了一个文件的 URL,那么所有这些查看器都可以显示该文件,并阻止对该文件的任何修改。在 Konqueror/embedded 中,像 Web 浏览器、flashpalyer、简单音频播放器都属于只读组件。另外一种读写组件 ReadWritePart 类是 ReadOnlyPart 的扩展,它添加了修改和保存文档的可能性,像 Konqueror/embedded 中嵌入的文本编辑器,属于读写组件。

(2) Konqueror/embedded 中组件实现技术

以 Konqueror/embedded 中文本编辑器组件(Kplaintextpart)为例,组件必须要由三个元素组成,包括窗口构件、组件功能和用户界面。窗口构件必须是 QWidget 的子代,Konqueror/embedded 中的文本编辑器继承于 QmultiLineEdit 类;除了窗口构件外,还需要组件提供的功能,Konqueror/embedded 的中文本编辑器提供前进、后退、剪切、复制和全选等附加功能;当然还要提供访问那些功能的用户界面(操作以及 XML 文件)。Konqueror/embedded 中的文本编辑器只提供了窗口构件功能的菜单项。在 XML 格式文件 Kplaintextpart.rc 中定义其用户界面的布局,它和应用程序代码是分开的。当该文本编辑器被嵌入到 Konqueror/embedded 时,采用称为 XML-GUI 的技术将组件菜单和原来的用户界面合并。

Konqueror/embedded 中的文本编辑器组件构成如图 3 所示。

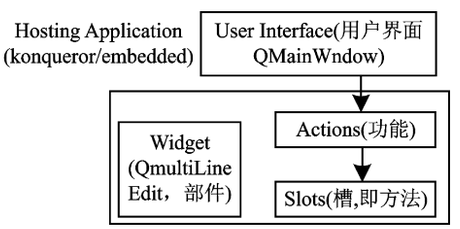

图 3 文本编辑器组件结构

Kparts 组件的最大特征在于它的可重用性。要做到这一点,就必须将组件编译到共享库中。希望动态地打开共享库的应用程序使用 KLibLoader 类。KLibLoader 处理库的定位、打开以及调用初始化函数。初始化函数是库的入口点。这个初始化函数创建一个工厂,由组件的工厂来创建组件。具体的实现方法如图 4 所示。

3.3 KHTML 绘制引擎的实现

KHTML 引擎作为 Konqueror/embedded 的核心,包含了 dom、xml、html、css、rendering、misc、ecma 七个子目录。这几个子目录的功能分别如图 5 所示。

从这个功能结构图可以看出,Konqueror/embedded 的 KHTML 引擎是基于 XML 的 DOM 技术来构建的,与现在的大部分浏览器的架构是相似的。DOM 是以层次结构组织的节点或信息片断的集合。在这种层次结构中可以利用导航仪搜寻特定的信息。同时,DOM 还提供了一套 API,可以用 JAVA,C++ 或 C 来实现这些 API。

现在最常用的解析 XML 文件的方法有四种:文档对象模型(DOM,Document Object Model)、用于 XML 的简单 API(SAX,Simple API for XML)、JDOM 和用于 XML 解析的 Java API(JAXP,Java API for XML Parsing)。它们有各自的优点和弱点,因而适用在不同的场合。Konqueror/embedded 中采用的是 DOM 技术。它所提供的接口和方法可以对构建的 DOM 树的节点进行添加、删除,甚至可以删除树的几个部分,还可以重新排列树和添加新的分支;但是,由于 DOM 构建整个文档驻留内存的树,如果文档很大,就会要求有极大的内存,这对于一般内存不大的嵌入式设备是个挑战。另外,DOM 构建整个文档的每个节点和元素,如果用户关心的只是其中的一部分,那么

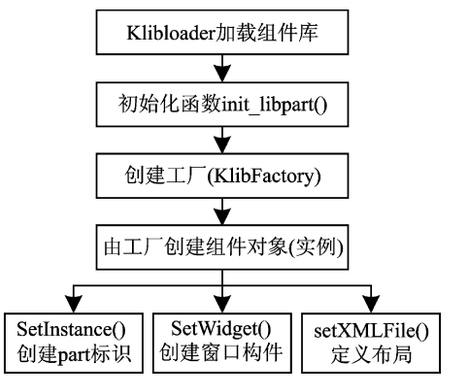

图 4 组件创建步骤

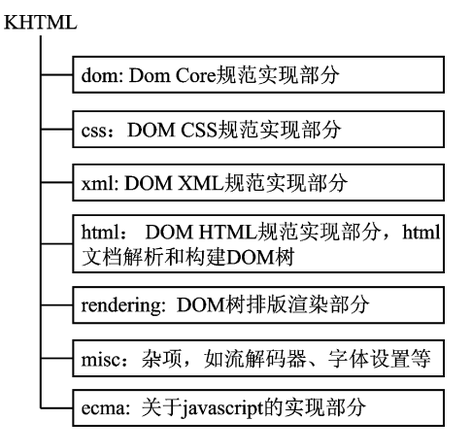

图 5 Konqueror/embedded 的构成

将会引起资源的浪费;同时,DOM 是在用户获取控制权之前加载整个文档的,如果文档很大,将产生明显的延迟。Konqueror/ embedded 中,采用 DOM 技术来解析 HTML/ XML 的最大优点在于,DOM 会自动地保存已经解析过的文档,而不必要在用户希望浏览历史的时候再重新解析文档。

Konqueror/ embedded 中采用的是 DOM2 级规范,分为 DOM2 Core 和 DOM2 HTML。相对于 DOM Level1, DOM2 增加了对 XML 文档处理的一些接口和方法:

◆视图(view),提供视图与文档的联系;

◆样式表(style sheet),提供访问和修改样式表的方法;

◆层叠样式表(CSS2),提供 CSS2 兼容的方法;

◆事件(events),提供各种事件的接口;

◆文档遍历(document traversal),提供遍历文档层次的接口;

◆文档范围(document range),提供分割文档范围的接口。

4 在 MC9328MX1 上的应用

4.1 Konqueror/ embedded 的移植

笔者在自行设计的基于 Motorola Dragonball MC9328MX1 芯片的开发板上成功移植了 Konqueror/ embedded。该开发板上运行嵌入式 Linux,并以 Qt/ embedded2.3.7 作为图形引擎库,上层运行 Qtopia1.6。Konqueror/ embedded 是基于 Qt 的,并将依赖于 Kde 部分进行了简化和包含,因此,移植 Konqueror/ embedded 到该开发板是完全可能的。

Konqueror/ embedded 提供了几十种编译选项,可以通过增减编译选项来定制适合不同平台,具有不同功能的 Konqueror/ embedded。

Konqueror/ embedded 的非核心组件的实现是在 add-ons 目录下,例如如果需要播放 Flash 支持,必须在编译选项中加入 - - enable - add - ons = kflashpart。在该部件中,默认是将声音关闭的,可以通过修改代码将声音开关打开。

由于 MC9328MX1 芯片是基于 ARM 架构的,所以移植的第一步是构建合适的交叉编译环境。笔者采用的是 arm-linux-g++ 编译器。编译过程与一般的 Linux 软件编译的过程相似;但是,将 Konqueror/ embedded 编译成功之后,单独下载 Konqueror/ embedded 的可执行文件 Konqueror 到开发板上运行将会出现致命的错误。因为 Konqueror/ embedded 的运行需要两个主要文件——html4.css 和 charset,是分别关于样式表解析和字体设置的。

4.2 Konqueror/ embedded 的汉化

Konqueror/ embedded 的汉化可分为两个部分:一个是 Konqueror/ embedded 本身控制界面的汉化;另一个是 Konqueror/ embedded 对中文网站的访问支持。和其他基于 Qt 的应用程序一样,Konqueror/ embedded 中对所有用户可见的文本使用;对所有文字形式的文本使用 tr()。tr()将文本标识出来,这样利用 Qt 提供的翻译工具将很容易把这些文本转化成所需要的语言;同时,Qt 提供了两个宏:QT_TR_NOOP() 和 QT_TRANSLATE_NOOP()。用它们标示出文本,以便于被 lupdate 工具提取。具体操作步骤如下:

◆在需要翻译源码的 XXX.pro 中加入一项:TRANSLATIONS = XXX.ts 。

◆运行 lupdate XXX.pro,生成 XXX.ts 文件。该工具识别出 tr() 结构和上面描述的 QT_ *_NOOP 宏,产生.ts 文件(通常每种语言一个)。

◆运行 lrelease XXX.pro 将生成 XXX.qm。这是一个没有翻译成其他语言的.qm 文件(也可以用 Qt Linguist 生成),可以把它改名为 XXX_en.qm。

◆启动 Qt Linguist 将 XXX.ts 文件导入,将需要翻译的内容翻译成中文。翻译完成之后,点击 File →Release,将文件保存为 XXX_zh.qm。

◆翻译文件保存之后,在程序源码中需要构建 Qtranslator 实例,利用 Qtranslator 将翻译文件加载到图形界面上。

在 Konqueror/ embedded 中可以通过修改 main.c 中的这段代码来实现 UI 的汉化:

```
QString qmFile = "XXX_zh.qm"
QTranslator *translator = new QTranslator (&app);
if (translator->load (qmFile))
    app.installTranslator (translator);
else
    delete translator;
```

为了让 Konqueror/ embedded 能够浏览中文网页,需要理解 Qt 对字符编码的处理。在 Qt 中采用 Unicode 编码的方式来存储、描述和运用字符串。让 Konqueror/ embedded 支持中文页面浏览实际上与让 Qt 支持中文字体的概念是一样的。Qt/ embedded/ Qtopia 中能够识别以下四种字体格式,并且必须是 Unicode 编码的:

◇TrueType(TTF) ——Scalable

◇PostScript Type1(PFA/ PFB) ——Scalable

◇Bitmap Distribution Format fonts(BDF) ——non-Scalable

◇Qt prendered Font (QPF) ——non-Scalable

其中,QPF 格式是 Qt 为了减小字符集体积和减小内存消耗而定义的一种字符存储格式。在 Qt/ embedded/ Qtopia 中,采用这种格式的字符集。

在 Qt/ embedded 中,提供转换 QPF 字体的工具——makeqpf。它是一个基于 Qt/ embedded 的程序,编译之后运行在 QVFB 中。我们采用的是 Qt/ embedded2.3.7 版本的 makeqpf,运行时需要先启动 QVFB,程序 makeqpf 会自动查找文件 $QTDIR/ lib/ fonts/ fontdir;所以,在此之前需要设置好 QTDIR,告诉 makeqpf 所需要转换的字符集的各种属性。将 MS Windows 下的字符集 simsun .ttc 复制到 $QTDIR/ lib/ fonts 目录下,改名为 simsun .ttf,在文件 fontdir 中添加如下的一行:

simsun	simsun .ttf	n	50	120	aus
字符集名称	字符集文件名	n 表示不是斜体, y 表示斜体(italic)	Weight 50 表示 nomal 75 表示 bold(粗体)	Size 120 表示 12pt	a 表示反走样(anti-aliased) u 表示unicode range when saving s 表示ascii range when saving

字符集转换完成之后,Konqueror/ embedded 中用来管理属性的 preference 类会在重启之后自动搜索系统可用字符类型,新的字体名称将会出现在选项栏中,以供选择。

5 总 结

作为一款全功能的嵌入式浏览器,Konqueror/ embedded 运用了很多 KDE 程序设计的方法和思路。Kparts 组件技术使得 Konqueror/ embedded 具有良好的可扩展性,以适应不同用户或不同场合的应用;I/ O - Slave 机制让 Konqueror/ embedded 能够通过各种网络协议透明地访问网络文件;而作为 KHTML 引擎的核心——DOM,使得 Konqueror/ embedded 能够正确地解析和渲染 HTML/ XML 文件,并在 DOM 结构树上绑定 ECAMScript 引擎和 CSS 解析器。可见在结构上,Konqueror/ embedded 和其他现代浏览器具有相似之处。Konqueror/ embedded 是基于 Qt 工具套件的,因此,Konqueror/ embedded 只能运行在以 Qt/ embedded 为基础的嵌入式设备上或运行 Qt/ X11 的 PC 机上,这在一定程度上限制了它的广泛应用。但是,由于它实现了完整的浏览器功能,并且是完全免费的,所以 Konqueror/ embedded 仍然具有很强的吸引力,同时对于其他的嵌入式浏览器设计,它所包含的很多设计思想是值得研究和借鉴的。

参考文献

1 David Faure. Kparts 教程. IBM:developerWorks
2 Nicholas Chase. DOM 教程. IBM:developerWorks
3 Doug Tidwell. XML 教程. IBM:developerWorks
4 孙勇. Konqueror/ embedded 之结构分析,2001
5 Trolltech Inc. Qt Reference Documentation 3.3.2.,2004
6 Cary R. Wright W. Richard Stevens. TCP/ IP 详解卷 1:协议. 北京:机械工业出版社,2002

蒋文军:硕士研究生,主要研究方向为基于嵌入式系统的网络解决方案,嵌入式 GUI 设计。张晓林:教授、博士生导师,主要研究方向为通信与信息系统,嵌入式系统,无人飞行器遥控遥测,集成电路设计。

(收稿日期:2004-12-27)

飞利浦首推基于 ARM9 的 90 nm 微控制器系列

2005 年 3 月 28 日,皇家飞利浦电子公司在 2005 年飞利浦半导体微控制器研讨会上宣布,利用其一直以来对 90 nm 技术的承诺及技术专长,推出基于 ARM9 系列的 90 nm 32 位微控制器系列。LPC3000 系列基于飞利浦的 Nexperia 平台,采用飞利浦、飞思卡尔半导体和意法半导体在法国建立的 Crolles2 联盟先进的 300 mm 试验设备开发的 90 nm 工艺技术制造。通过采用 90 nm 工艺技术和 ARM926EJ - S 内核,飞利浦可以降低生产成本,减少功耗,并提高其先进的 32 位微控制器的运行速度。对行业、外设控制、电动机、网络和安全应用领域的生产商而言,高性能、低功耗和低成本得到了最佳的结合。

系统设计师在不断努力降低功耗,尤其是对于手持消费设备而言,优化功耗可以延长电池寿命。飞利浦基于 ARM9 系列的 LPC3000 微控制器系列是上述应用的理想选择,因为 90 nm 技术支持 1 V 运行,与 3 V 器件相比,功耗仅及其 1/ 9。ARM9 系列还提供其他功率管理功能,包括可保持低功耗状态,直至产生中断或调试需求。集成诸如 USB On-the-Go(OTG)和全 USB 开放主机控制器接口(OHCI)等外设的主机功能,避免了对外部控制器的需求,进一步降低了功耗和成本。此外,消费电子生产商还可为新设备附加更多先进的功能。LPC3000 系列采用多级 NAND 闪存接口,使客户可以采用市场上密度最高、最经济的闪存。LPC3000 系列产品以 200MHz 的频率运行,采用大量标准的通信外设以减少系统逻辑,从而降低功耗和成本。其中包括多达 7 个 UART、SPI、I^2C、USB、实时时钟、NAND 闪存接口以及随后会推出的以太网接口等等。该系列还采用了一个矢量浮点协处理器,完全支持以 CPU 时钟速度进行的单精度和双精度运算。这对于电动机控制等信号处理类应用是至关重要的。