第33卷 第19期 计 算 机 工 程 2007年10月
Vol.33 No.19 Computer Engineering October 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)19—0259—03 文献标识码：A 中图分类号：TP311.52

嵌入式 Linux 图形系统实时采样曲线绘制的实现

吴建飞，程明霄
(南京工业大学自动化学院，南京 210009)

摘 要：在嵌入式 Linux 环境下，以 PC/104 以及相关采样扩展板为平台，介绍了图形嵌入式应用软件中，实时采样曲线的绘制设计与实现，描述了基于 MiniGUI 的底层绘图机制，提出了一种新的数据结构和接口，解决了实时绘图中闪屏以及动态坐标技术难题。
关键词：嵌入式 Linux；MiniGUI；实时曲线；闪屏；动态坐标

Implementation of Real-time Curve Plotting Under Embedded Linux Graphic System

WU Jian-fei, CHENG Ming-xiao
(College of Automation, Nanjing University of Technology, Nanjing 210009)

【Abstract】This paper solves real-time curve plotting problem by creating a new plotting data interface which has large expansibility and transplant behavior under an embedded GUI application. This GUI application is applied under the embedded Linux environment based on the PC/104 platform. A new low level plotting mechanism and method is created in order to solve the dynamic scale and glitter scream problem. The experiment result shows that they can solve the real-time plotting problem well under embedded application.
【Key words】embedded Linux; MiniGUI; real-time curve; glitter scream; dynamic coordinate

近几年来，兴起了基于嵌入式 Linux 的应用研究，嵌入式 Linux 系统保留了大部分原 Linux 操作系统的各种优点：开放的源码；功能强大的内核，性能高效、稳定，多任务；支持多种体系结构，如 X86、ARM、MIPS、ALPHA、SPARC 等；完善的网络通信、图形、文件管理机制；良好的开发环境，不断发展的开发工具集。

在嵌入式 Linux 上面已经存在多种图形用户界面，如 Qt/Embedded、Microwindow、OpenGUI 以及 MiniGUI。目前国内基于图形界面嵌入式应用开发一般选择 MiniGUI，它是一款比较优秀的图形用户界面开发环境，它利用多线程技术提供了相对完备的多窗口机制，实现了类 Win32 的消息传递机制，可以支持常见的图像文件等。MiniGUI 体积小巧、可配置、移植性好，且支持中文字符集，同时 MiniGUI 是免费的。所以选择 MiniGUI 作为本科研项目的图形开发工具环境。

在基本的嵌入式应用中，往往涉及到数据的采集，以及动态实时地显示数据曲线，由于嵌入式系统硬件资源有限，嵌入式 Linux 系统一般很少携带 X Windows 系统。目前较流行的做法是使用 framebuffer 驱动，直接操作显存缓冲来实现无 X server 下的图形界面。而实时曲线的绘制对于底层绘图的处理要求较高，要求高分辨率、高效率、无闪烁、动态缩放等。本文基于 MiniGUI 工具提出解决嵌入式应用中实时曲线的绘制方法以及实现。

1 背景与需求分析

本技术的背景是对工业色谱仪的软件系统的开发，工业色谱仪采用 PC/104 以及相应的输入输出模块作为硬件平台，采用经过优化的嵌入式 Linux 操作系统作为软件系统，并且为数据采集模块开发相关驱动模块，加载入内核，提供实时采集或中断采集系统服务。在工业色谱仪系统应用软件当中，对色谱数据提供谱图的在线实时显示。系统实时数据显示部分的系统解剖图如图 1 所示。

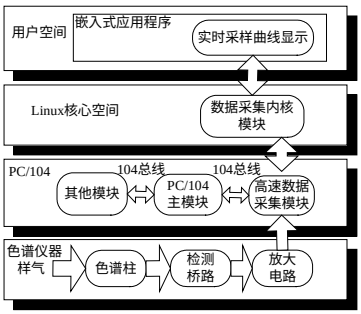

图 1 色谱曲线显示系统解剖图

在工业色谱仪中，待分析的样气经过色谱柱分离，色谱分析检测器将其不同组分含量的信息转换为不同保留时间的组分信号，经过程控放大器放大以后，利用 A/D 采样把该信号转换为数据流，从而形成谱峰数据。色谱仪采样通常采用 16~24 位 A/D 采样，数据精度高，谱峰数据带有一定的毛刺；同时在分析检测过程中，由于检测器以及程控放大器的影响，数据存在一定的波动；样气化学组分变化大，谱峰数据的变化范围也会非常巨大，出现谱峰的曲线也会高低相差很大。在本系统中，嵌入式 Linux 系统驱动服务负责硬件的配置以及 A/D 转化模式等设置后，将采集到的实时数据最终传送到用户程序空间，用于分析各种化学成分以及组成含量。

首先，考虑的谱图的谱峰大小差距非常大的可能情况，

作者简介：吴建飞(1982－)，男，硕士研究生，主研方向：嵌入式 Linux 应用；程明霄，副教授
收稿日期：2006-10-17 **E-mail：**rainfly_365@163.com

显示部分需要动态坐标支持，即当谱峰很小时，需要系统坐标范围减小来放大谱峰，而对于出现的高谱峰，系统将坐标范围放大，使其能够显示整个谱峰。其次，色谱图曲线显示中，要解决绘图的连续显示问题，即曲线占用整个显示用横轴宽度之后，谱图曲线将开始滚动显示。另外，对于核心模块传送来的实时数据，需要提供高效的接口，既方便数据收集，也方便数据的显示。

2 设计与实现

2.1 数据接口设计

为了很好地解决上述的绘图需求，采用绘画与数据分离的技术，即把一般直接将数据用于绘图的思想摒弃，而根据解耦的思想来创建绘图专用的数据结构，将需要绘图的数据与绘图操作分离。这样做的优点是，绘图只需要面向该数据结构，解决绘图中的图形处理问题，关于图形的移动、坐标变换以及数据存储和更新则交给专门的程序段负责，维和该数据结构。数据接口在整个实时曲线绘制部分的关系如图 2。

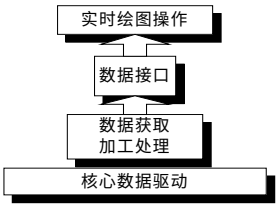

图 2 实时曲线绘图结构

数据接口根据实时曲线的需求，进行如下定义：

```
 #define MAXLENGTH  450      //定义最大的实时曲线屏幕
//宽度， 该宽度可以通过宏定义修改
  #define MINIAXIS      1
  //定义最小坐标量程电压值 当前定义为 1V
  typedef struct
   { float WaveLine[MAXLENGTH];
    //绘图数组，用于映射到屏幕曲线，存储采集数据经过转换后
//的实际电压值
     float Max; //  数组中的最大电压值,用此变量来动态变换坐标
     int length;    //  数组中有效电压值个数
     int  current;   //  绘图数组元素指针，指向绘图数组中电压存
//储单元
     char realtime_vol[10];    //  字符串，记录当前的实时电压值
     char current_vol[10];     //  相对应于 current 指针指到的电压
//值字符数据
   } DrawData;
```

该接口可以完全胜任实时曲线的数据交换需求，使用过程中，只需要定义并先初始化：

```
  DrawData Wave_1;
  Wave_1.Max=MINIAXIS;
  Wave_1.length=0;
  …
```

2.2 绘图机制

利用上节定义的数据接口，将绘图过程分成两个步骤：(1)从驱动程序中获取实时数据，对该接口进行数据更新以及相应的绘图参数更新；(2)绘图处理程序块通过读取该数据接口，按照指定的参数进行底层绘图操作。当新数据到来时，首先更新数据接口中 WaveLine 数组中最后一个数据，以及将该数组中的数据顺次左移，然后判断最大电压值，更新 Max；在绘图处理部分，将根据最新的绘图参数以及 WaveLine 数组中的数据将整个曲线重新绘出，该绘图部分的功能是数据采集和处理解耦。部分代码如下：

```
if(Wave_1.length<MAXLENGTH)
{   // 程序运行初始阶段，接口数据初始化
     Wave_1.WaveLine[Wave_1.length++]=NewData;
     if(NewData>Max)Max=NewData;
     …    }
else{ //  数据已填满绘图数组
    for(int i=1,Max=1;j<MAXLENGTH;j++)
     {          …
//数据接口的采样数据和绘图参数更新      }
     … }
```

该过程的机制原理如图 3 所示。

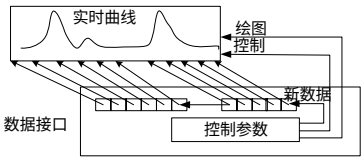

图 3 绘图机制原理

2.3 实时绘图技术

在MiniGUI开发环境下，实时绘图采用GDI，即图形设备接口(graphics device interface)。通过 GDI, GUI 程序就可以在计算机屏幕上，或者其他的显示设备上进行图形输出，包括基本绘图和文本输出。绘图所操作的是图形设备上下文，这点类似于Windows 和 X Window 中普遍采用的图形设备概念。每个图形设备定义了计算机显示屏幕上的一个矩形输出区域。在调用图形输出函数进行实时曲线绘制时，要指定经过处理后的图形设备上下文(device context, DC)。使用之前，首先要进行设备上下文的初始化，一个经过初始化的图形设备上下文确定了其后进行图形输出的一些基本属性，包括输出的线条颜色、填充颜色、字体颜色、字体形状等[1]。

基于MiniGUI的应用程序基于消息循环与处理机制，程序的默认绘图操作发生在窗口失效或者发送窗口更新消息之后。在消息处理函数中的MSG_PAINT部分进行窗口的更新操作。要对窗口进行操作，首先要获得设备上下文，在该消息处理部分，一般使用BeginPaint(HWND hWnd)函数返回整个窗口的设备上下文，之后可以通过调用设备上下文处理函数进行相应的处理[2]。示例代码如下：

```
 case MSG_PAINT:
    hdc = BeginPaint (hWnd);
   …
 EndPaint(hWnd,hdc);
 break;
```

但是实时绘图如果采用MSG_PAINT消息处理的话，会带来很多问题，比如，响应时间慢，增加系统的负担，因为实时绘图一般 1ms~10ms就需要更新一次。并且，经过试验，在该消息下绘图，并且使用BeginPaint获得的设备上下文，会带来曲线显示的闪烁问题，闪烁的原因是由于在绘制曲线的过程中，频繁操作显示缓冲区，并且该区域的刷新图像刷新操作封装在MiniGUI内部，内部刷新操作缓慢。由于上述原因，实时曲线绘制将不采用直接操作显示缓冲区的设备上下文，改用内存显示设备上下文[3]。

MiniGUI当中有丰富的相关操作API函数，笔者经过多次试验和调试，选用GetClientDC(HWND hWnd)来获得窗口客户区域的设备上下文，用CreateCompatibleDC(HDC hdc)来创建内存设备上下文，绘图操作针对内存设备上下文，待绘图结束，将内存设备上下文覆盖原先的客户区域设备上下文，

该过程调用BitBlt (HDC hsdc, int sx, int sy, int sw, int sh, HDC hddc, int dx, int dy, DWORD dwRop)来实现[3]。此外，由于这种方式的绘图，系统将不会自动刷新屏幕，因此需要自己刷新绘图区域，刷新操作在这里采用SetBrushColor(hdc,PIXEL_lightwhite)和FillBox(HDC hdc, int x, int y, int w, int h)函数，前者用来指定刷屏的颜色，后者是矩形区域填充函数用来实现刷屏。在该过程中需要注意的是绘图的坐标，如果绘图区域指定不当将造成不可预料的图像或者花屏。具体的曲线绘制过程：首先根据绘图参数绘制坐标，之后调用LineTo(HDC hdc, int x, int y)和MoveTo (HDC hdc, int x, int y)来对内存设备上下文进行绘图。核心代码如下：

```
  hdc_old=GetClientDC(hWnd);
//获取窗口客户区域设备上下文
  hdc=CreateCompatibleDC(hdc_old);     //创建内存设备上下文
  SetBrushColor(hdc,PIXEL_lightwhite); //设置刷屏颜色
  FillBox(hdc,50,50,600,400);          //在指定区域刷新
  TextOut(hdc,150,50,dis_vol);         //输出电压值
   …
//根据参数绘制坐标
  for(int i=0;i<MAXLENGTH;i++)
   { if(i<Wave_1.length)LineTo(…);
      …
      //曲线绘图操作
      BitBlt(hdc,50,50,600,400,hdc_old,50,50,0);
      //更新窗口客户区域设备上下文
      DeleteCompatibleDC(hdc);
      ReleaseDC(hdc_old);
      //释放创建的设备上下文 }
```

2.4 动态坐标及其他

由于谱峰的变化范围非常大，小谱峰与大谱峰之间有时候相差巨大，这时就需要根据不同的情况，采取动态坐标的方法。动态坐标的算法是：首先，选取屏幕绘图区域范围，本系统中设定屏幕区域为 400×600 像素，设定最低量程范围，比如 1V，如果在当前绘图数组中没有出现大于 1V 的，则屏幕的量程为 1V，而如果绘图过程中出现大于 1V 的，设为 A，则取 A 屏幕显示的最高量程为 A，这样则会由于大谱峰的出现而使整个谱图纵向压缩，当大谱峰退出时，谱峰图像会纵向放大。具体原理如图 4 所示。

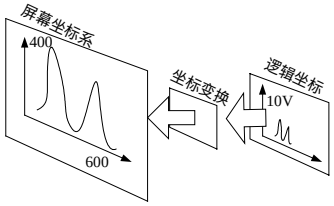

图 4　动态坐标原理

实现动态坐标的代码嵌入到底层画图中，核心代码如下：

```
#define ORIGIN_LEFT 150
          //坐标原点像素坐标
#define SCALE_Y       400
          //屏幕坐标 Y 方向量程
   …
   for(int i=0;i<MAXLENGTH;i++){
     if(i<Wave_1.length)LineTo(hdc,ORIGIN_LEFT+i,(ORIGIN-Wave_1.WaveLine[i]*SCALE_Y/Max));  //在动态坐标下绘制实时曲线
      …}
```

该算法能够解决采样曲线的动态放大和缩小功能，经过实际的运行动态效果良好。

此外，由于画图没有在 MSG_PAINT 消息处理下执行，因此需要触发画图，根据工业色谱仪的实际运行需求，20ms 之内就需要采样一次，所以，可以将画图放在定时器消息处理中，即 MSG_TIMER 下执行采样数据的获取和画图。

经过 gcc 编译并连接到工业色谱仪主程序中，实际运行动态响应迅速，动态坐标表现出色且没有闪屏和抖动。程序运行截图如图 5 和图 6，显示了运行中动态坐标的变换效果。

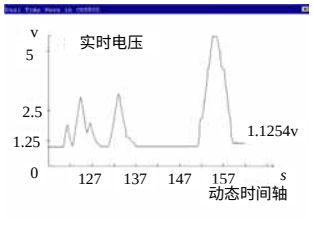

图 5　实时曲线截图(1)

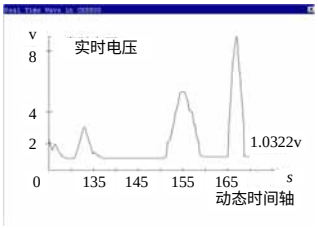

图 6　实时曲线截图(2)

3　总结

本文主要阐述了在嵌入式 Linux 下，结合 MiniGUI 工具，进行实时曲线绘图的基本设计原理和实现方法。主要设计了通用的数据接口，实现了绘图和数据处理的解耦，方便了绘图数据的处理以及绘图操作的优化。并且这种数据接口具有很好的扩展和移植性，比如对于多曲线显示的扩展，只需要再增加若干个同样的接口就可以简单实现。在此基础上，通过内存设备上下文绘图方法的实现，很好地解决了实时绘制过程中容易出现的闪屏问题。对于大范围动态曲线的动态坐标显示，通过动态坐标方法，实现了动态改变屏幕量程来达到最佳显示效果，并且该方法已成功用于国产智能工业色谱仪中，在线分析下谱图的实时显示，收到了良好的效果。

参考文献

1 魏永明. 基于 Linux 和 MiniGUI 的嵌入式系统软件开发指南(四)使用 GDI 函数[EB/OL]. [2006-05-02]. http://www.minigui.com/techdoc/guide-4/.

2 Beijing Feynman Software Technology. MiniGUI API Reference Documentation[EB/OL]. (2003-09-15). http://www.minigui.com/api_ref/1.3.x/index.html.

3 Beijing Feynman Software Technology. MiniGUI Programming Guide[EB/OL]. (2003-09-01). http://www.minigui.org/docs/mpg-1.3-contents.shtml.

4 刘文俊, 杜　旭, 杨宗凯. 基于嵌入式 Linux 和 MiniGUI 的 E-mail 客户端软件的实现[J]. 计算机工程, 2004, 30(11).

（上接第 258 页）

参考文献

1 Veritas Software Corporation. NetBackup_AdminGuideI[Z]. (2004-04-26). www.veritas.com, 2004.

2 Veritas Software Corporation.NetBackup_AdminGuideII[Z]. (2004-04-26). www.veritas.com, 2004.

3 Veritas Software Corporation. 企业重生——信息系统的灾难恢复[Z]. (2004-06-01). www.veritas.com, 2004.