文章编号:1671-4598(2015)05-1660-04 DOI:10.16526/j.cnki.11—4762/tp.2015.05.064 中图分类号:TP911—34 文献标识码:A

基于NI—DAQ的高采样实时数据绘制方法与实现

王 鹏,叶 冰
(中国飞行试验研究院,西安 710089)

摘要:飞行试验实时监控中,颤振等高采样数据通常采用时间历程曲线绘制的方式进行分析与故障判断,这对数据绘制的实时性、准确性以及曲线绘制长度等都提出了较高的要求;文章在对NI PCI—6713板卡的模拟输出功能进行详细讨论和比较验证的基础上,对基于NI—DAQ的曲线绘制进行了算法研究和软件设计,实现了在监控客户端的16路高采样数据实时曲线绘制与打印;经试验验证,采用该方法绘制的高采样数据曲线准确可靠,满足监控要求,能有效保障飞行试验的正常进行。

关键词:高采样;NI—DAQ;曲线绘制

Curve Plotting Method of High Sampling Real—time Data Based on NI—DAQ

Wang Peng, Ye Bing
(CFTE, Xi'an 710089, China)

Abstract: In real—time monitoring of flight test, the high sampling data, such as flutter data, is generally analyzed and evaluated by means of time—history curve plotting, which must have the characteristic of real—time, high precision and enough length. In this paper, based on the detailed discussion, comparison and verification of NI PCI—6713 board in terms of the analog output, the curve plotting method of high sampling data based on NI—DAQ is researched and implemented, as a result, 16 channels high sampling data are plotted and printed on the client side. It shows that, the curves plotted by this method are accurate and reliable, meet the requirement in real—time monitoring.

Keywords: high sampling; NI—DAQ; curve plotting

0 引言

随着试飞测试技术的不断发展,机载数据的采集数量和采样率都有了大幅提高,为试飞工程师了解飞机和机载设备性能提供了更详实的数据,这对于及时发现信号瞬态变化,保障飞行特别是高风险科目的安全高效进行有着重要的现实意义。

以颤振课题监控[1]为例,新型飞机及重大改型飞机均要通过颤振飞行试验来证实飞机在使用包线范围内会不会发生颤振现象。与其他试验科目相比,对数据的实时性、可靠性及精确性都要求较高,机载数据采样率为512点/s。对于关键参数的监控,多采用曲线形式,便于观察参数趋势,发现不正常现象。由于参数更新速率较高,使用常用方法进行绘制很难保证实时性和精确性,而且由于屏幕长度有限,仅能显示有限时间内的数据,不能满足用户对数据量的要求。

现有颤振课题监控,采用了定制的遥测监控系统,通过NI PCI—6713多功能数据采集卡外接DASH 18X暂态波形记录器,对高采样数据进行曲线绘制和打印输出,满足了实时监控和数据分析的要求。由于开发时的网络传输限制,将数据绘制功能与服务器端实时数据处理系统进行了集成,数据解调、DA输出、实时绘图所涉及的硬件资源绑定在了一起,使得服务器端的处理能力和稳定性受到了限制,已不能满足监控系统分布式、网络化的发展要求。针对现有系统的不足以及新机试飞中的用户需求,本文在现有硬件体系基础上,设计了基于NI—DAQ的高采样实时数据绘制软件,实现了客户端的数据监控与绘制。

收稿日期:2014-08-18; **修回日期**:2014-09-30。

作者简介:王 鹏(1985-),男,硕士,工程师,主要从事遥测数据实时处理技术方向的研究。

1 系统结构

系统结构分为硬件和软件两部分,其中硬件在现有硬件基础上进行改进,重点进行软件设计与实现。

1.1 硬件结构

实现高采样数据绘制的硬件结构如图1所示。飞机下发的PCM流数据经遥测天线接收、解调后,送入前端服务器进行采集、位帧同步、参数提取、工程单位转换等处理,客户端通过与服务器端通信获取工程数据进行监控和后处理。本文讨论重点为虚线框中的部分。其中,客户端除具备一般客户端功能外,通过增加NI PCI—6713多功能数据采集板卡进行参数的DA转换,由输出的电压驱动DASH 18X暂态波形记录器进行曲线绘制和打印输出。NI PCI—6713多功能数据采集卡,具有8条电压输出通道,范围±10 V,12位精度,可完成最高1 MS/s采样速率的模拟输出。DASH 18X暂态波形记录器(以下简称条图仪)共有18条模拟通用输入通道,支持连续不间断采集和实时波形打印。实际应用中,使用两块PCI—6713板卡和一台条图仪进行16路通道的波形输出。

1.2 软件结构

软件结构如图2所示,在实现原有客户端软件功能基础上,增加了高采样数据的曲线绘制。其中,网络通信模块、数据显示与保存模块、画面调度与数据分发模块的设计请参考文

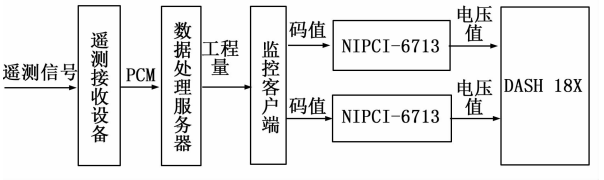


图1 系统硬件结构框图

献［2］，本文重点对曲线绘制模块进行讨论。

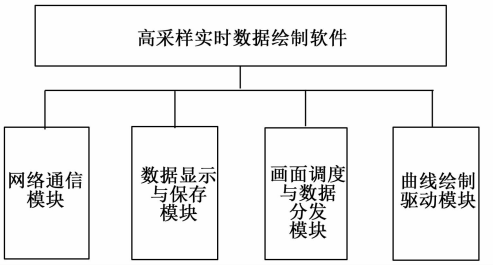


图 2　软件结构

2　关键技术

NI 为采集卡提供了完备的驱动软件和硬件开发工具包 NI－DAQ[3]，支持 NI LabVIEW，C++、ANSI C 等开发语言，可以帮助用户快速高效开发应用程序。本文使用 Borland C++ Builder 编程工具[4]结合 NI－DAQ 的 C++函数库，实现高采样数据的实时绘制。

2.1　模拟输出

NI－DAQ 中有两组输出函数用于模拟量输出。

模拟输出（AO）函数组：用于单点数模转换；

波形（WFM）函数组：用于有缓冲区的数模转换。

使用 AO 函数组进行模拟输出的流程如图 3 所示。

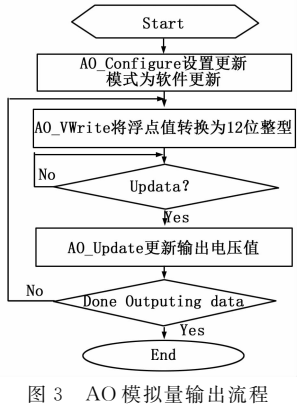


图 3　AO 模拟量输出流程

其中，AO _ Configure 对 6713 板卡各模拟输出通道的极性、参考电压，更新模式等进行配置；AO _ VWrite 将输入的浮点型电压值，转换为［0，4095］之间的整型并写入模拟通道；AO _ Update 更新模拟通道输出当前值，通道输出前一电压值直到该函数调用。

使用该组函数的优点是结构简单，对单点直接输出，无系统延迟，但一点输出后，在下一点输出之前，电压值会持续前一点大小，实际输出波形变成了逐点延迟的阶梯信号（如图 4 所示）。

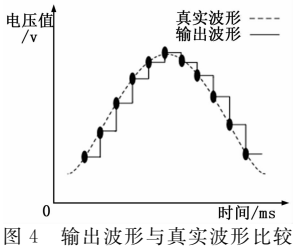


图 4　输出波形与真实波形比较

虽然随着采样率的提高这种延迟会减小，但条图仪输出波形仍不够光滑，不能满足监控要求。

使用 WFM 函数组进行模拟输出的流程如图 5 所示。

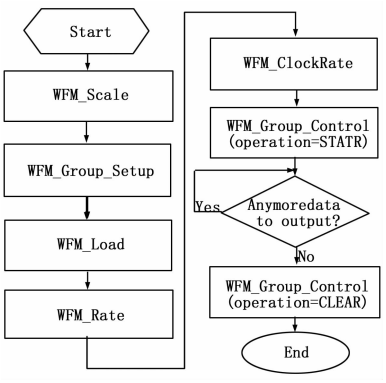


图 5　WFM 模拟量输出流程

具体步骤如下：

1）WFM _ Scale 将浮点型电压值数组转化为二进制数组；

2）WFM _ Group _ Setup 将一个或多个通道加入一组；

3）WFM _ Load 载入波形缓冲区到指定输出通道；

4）WFM _ Rate 将数据输出速率转化为时间基准和更新间隔；

5）WFM _ ClockRate 使用时间基准和更新间隔设置板卡更新速率；

6）WFM _ Group _ Control 开始或结束波形生成。

在有缓冲区波形生成过程中，数据以 DMA 或者中断方式先从主机内存中的缓冲区被移动到 DAQ 设备的 FIFO 缓冲区，然后由数模转换器在每个采样周期逐点读出来产生波形。输出方式遵循先入先出的原则，写入 FIFO 的点只有在当前缓冲区内的点输出后才会被数模转换器读取。与单点输出模式相比，因为数据以块的形式传输，所以该模式允许更快的数据速率。同时由于先入先出的原则，数据点从写入缓存到模拟输出会消耗一定时间，造成波形产生有所延迟，这与板卡更新速率以及数据块大小有关。通过合理的安排数据加入的时间，可以减小甚至消除这个延迟。

对比两组函数，WFM 函数在高采样数据条件下，能够更好的还原数据信息。

2.2　码值转换

由于 PCI－6713 输出通道输出精度为 12 位，电压值范围为（10 V，即［0，4 095］的整形数对应［－10，10］的浮点数，最低有效位 $LSB = 20/2^{12}$。实际输入给 FIFO 缓冲区的应该是［0，4 095］的整形数，而工程量数据是经过服务器端对精度为 16 位的原始码值进行校线校准计算得到的浮点数，因此必须经过坐标变换后方能输出。坐标变化过程如图 6 所示。

对于工程量数据 wx，按照公式（1）进行坐标转换。首先通过校线信息计算其原始码值 $vy \in [0,65\ 535]$，然后对码值进行 1/16 缩放，将码值范围调整为 $vy1 \in [0,4\ 095]$，由于输出为双极性，2 048 对应的电压值为 0，因此通过 $vy1 - 2\ 048$ 得到 da_val 值即为实际的 FIFO 输入值。

$$da\_val = \frac{1}{16} \times \left(\frac{c_{max} - c_{min})(wx - v_{min})}{v_{max} - v_{min}} + c_{min} \right) - 2048 \quad (1)$$

2.3　实时绘制

由于本文设计软件运行在客户端，需要完成网络通信、参

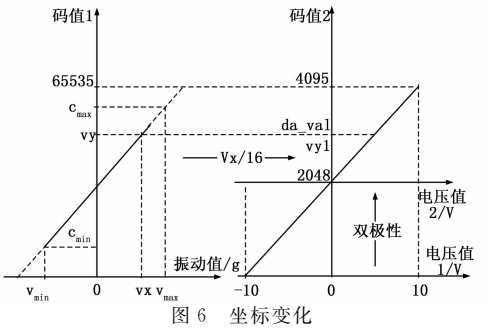


图 6 坐标变化

数显示、数据保存、画面调度、数据分发和曲线绘图等功能，在高采样数据条件下，需要在极短的时间内完成多项任务，对系统硬件特别是 CPU 造成很大压力，特别是在单线程程序中，一项任务只有在前一项任务完成后才会被执行，费时的操作可能会造成数据延迟和丢失。同时，软件界面响应迟滞，影响操作，甚至造成系统假死。这些因素在飞行试验实时监控中都是应该避免和消除的。为了解决这个问题，实现数据实时绘制，本文使用了多线程。

多线程技术将程序的行为组织为功能独立的并行进程，通过同步执行每个并行进程中的单段代码，提供了更好的程序响应速度和编程效率的折衷。BCB 的 VCL 类库中提供了用于线程编程的 TThread 类。在该类中封装了 Windows 中关于线程机制的 API，可在应用程序中使用线程对象来表示执行线程，简化了多线程程序的编写。

本文通过创建 TThread 的派生类 DAThread 来实现曲线绘制。步骤如下：

1）定义 DAThread 类：

```
class DAThread : public TThread
{
private:
protected:
void __fastcall Execute();
public:
int da_val[256];
__fastcall DAThread();
};
```

其中 *da_val*[256] 用于存储待绘制数据；DAThread (bool CreateSuspended) 用于对线程进行初始化；Execute () 函数负责实现曲线绘制。

2）实现 DAThread 类：

```
…
//初始化 PCI−6713
int iChanV[8]={0,1,2,3,4,5,6,7};
WFM_Group_Setup(1,8,iChanVect,1);
WFM_Rate(512,0,updateTB,updateINT);
WFM_ClockRate(1,1,0,updateTB,updateIN
…
//开始绘制
WFM_Group_Control(1,iGroup,0);
WFM_Load(1,8,iChanV,da_val,256,1,1);
WFM_Group_Control(1,iGroup,1);
…
```

由于 da_val 数组在主线程赋值后由 DAThread 线程读取进行曲线绘制，为了防止两个线程同时访问，创建全局的 TEvent 类对象 DAout 作为所有线程可检测的标志。当主线程赋值完成后，调用 DAout 的 SetEvent () 方法通知 DAThread 线程开始绘制，绘制完成后调用 ResetEvent () 方法取消标志，等待下次赋值结束。

3 软件实现

3.1 软件流程

软件实现流程如图 7 所示，关键步骤如下。

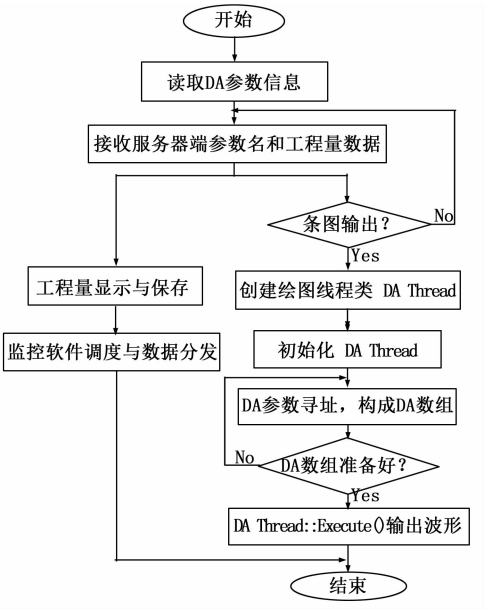


图 7 软件流程图

1）加载 Outpar.par 文件。Outpar.par 文件保存了进行曲线绘制的参数信息，格式为：“参数名 最大码值 最小码值 最大物理量 最小物理量”（其中的码值与物理量根据校准曲线相对应），读取后的信息存放在结构体 da_par 中：

```
struct da_par //DA 输出数据结构
{
char par[8][32]; //参数名
int cmax[8]; //最大码值
int cmin[8]; //最小码值
float vmax[8]; //最大物理量
float vmin[8]; //最小物理量
}da;
```

2）接收服务器端参数列表与物理量，定位绘制参数，将物理量存入浮点型数组 *data*[8]；

3）创建 DA 线程并进行使用 WFM 函数组对板卡进行初始化；

4）在一个 PCM 帧时间内，按照公式（2）对 *data*[8] 进行坐标转化，并循环存入整型数组 *val*[256]。

$$val[i+p*8]=\frac{1}{16}\times\left(\frac{(da.cmax[i]-da.cmin[i])(data[i]-da.vmin[i])}{da.vmax[i]-da.vmin[i]}+da.cmin[i]\right)-2048 \tag{2}$$

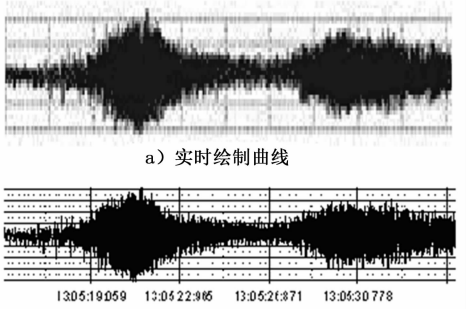
其中，i 为数组 *data*[8] 的索引，*p* 为一帧内 DA 参数的采样率。

5）通知 DAThread 线程可以进行绘制，DAThread:: Execute () 调用 WFM 函数组，将 *da_val* 数组送入 FIFO 开始

绘制。绘制结束后调用 daout－＞ResetEvent（），返回第 4 步。

3.2　软件应用实例

在服务器端对某型机颤振试验实时记录数据以 512 点/秒速率进行回放，在客户端运行曲线绘制软件进行曲线绘制与打印；将实时记录数据进行事后数据处理，选择同一参数绘制时间历程曲线，选取同一时间段内曲线进行对比，如图 8 所示，



a）实时绘制曲线

b）事后处理绘制曲线

图 8　实时绘制曲线与事后处理绘制曲线比较

可见实时绘制的曲线与事后绘制的一致，准确反映了参数变化情况。

4　结论

经多次试验证明，基于 NI－DAQ 的高采样曲线绘制软件在客户端准确实现了监控数据的实时绘制，与原有软件相比具有配置简单、使用灵活等特点，更加符合监控资源网络化、分布式的发展要求。在颤振等风险科目的安全监控中，将发挥重要的辅助作用。其稳定性与准确性，也将在飞行试验中得到进一步的验证和优化。

参考文献：

[1] 祁　春，段宝元．遥测数据实时处理软件系统及其应用［J］．科学技术与工程，2010，10（28）：7047－7050

[2] 覃　燕，段宝元．飞行试验颤振数据实时监控系统［J］．现代电子技术，2011，11（34）：63－65

[3] NI 公司．传统 NI－DAQ 用户手册［Z］.

[4] 李幼仪，甘　志．C++ Builder 高级应用开发指南［M］．北京：清华大学出版社，2002

（上接第 1659 页）

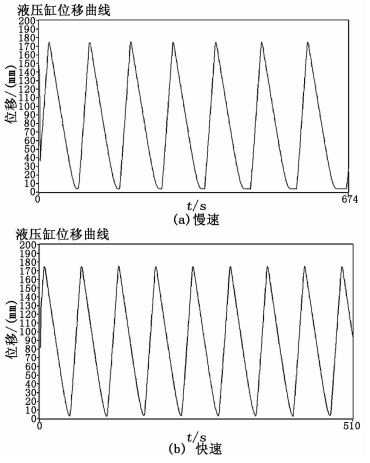


图 6　液压缸实时位移曲线图

积。通过对比可以看出，数据采集模块能够以较高的精度获取传感器数据，并及时地通过 CAN 总线发送至主控计算机。

在离线的情况下，还可以通过 SD 卡获取系统运行时的数据，导入 MATLAB 也可以得到位移曲线，还可以进一步分析处理。图 7 为 SD 卡存储的液压缸位移数据。

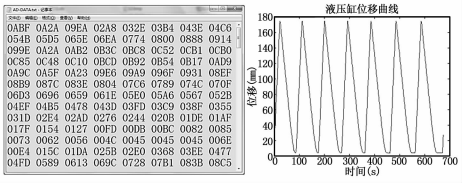


图 7　SD 卡存储的数据

5　结论

本文提供了一种数据采集模块的设计方案，并完整介绍了该模块的硬件设计和软件实现。选用的 STM32 微控制器具有丰富的外设接口，易于功能扩展。嵌入式实时操作系统 μC/OS－II 简化了程序的编写和维护，数字滤波算法，提高了采样精度。实验结果表明，该数据采集模块具有体积小、功耗低、实时性好、可靠性高等特点，有较强的实用性，适合在车载装备上使用。

参考文献：

[1] 强　宁. 基于 TMS320F2812 的航空发动机转速信号采集研究［J］. 电子测量技术，2008，31（11）：76－79.

[2] 谢拴勤，王文换. 基于 μC/OS－II 和 CAN 总线的网络化测控系统设计［J］. 计算机测量与控制，2008，16（7）：926－928.

[3] 毕　盛，闵华清，李　淳，黄夔金，陈必强. 姿态传感器采集测试系统的设计与实现［J］. 计算机测量与控制，2011，19（7）：1562－1564.

[4] Jean J. Labrosse. 嵌入式实时操作系统 μC/OS－II（第 2 版）［M］. 邵贝贝，等译. 北京：北京航空航天大学出版社，2003.

[5] 刘辉邦，褚金奎，支　伟，李晓雨. 基于 STM32 的无人机姿态测量控制系统设计［J］. 传感器与微系统，2013，32（8）：108－110.

[6] 吴华玉. 基于 STM32 单片机的嵌入式仪器 A/D 接口设计［J］. 长春工业大学学报（自然科学版），2013，34（4）：448－451.

[7] Zhang H F，Kang W. Design of the Data Acquisition System Based on STM32［J］. Procedia Computer Science，2013，17：222－228.

[8] 闵　建，程　明. 嵌入式实时数据采集系统的设计与实现［J］. 微计算机信息，2009，25（7－2）：98－100.

[9] 郑　朋. 在 Cortex－M3 上实现基于 μC/OS－II 和 CAN 总线的实时数据采集系统［D］. 青岛：青岛大学，2011.

[10] 李世奇，董浩斌，李荣生. 基于 FatFs 文件系统的 SD 卡存储器设计［J］. 测控技术，2011，30（12）：79－81.

[11] 刘波文，孙　岩. 嵌入式实时操作系统 μC/OS－II 经典实例：基于 STM32 处理器［M］. 北京：北京航空航天大学出版社，2012.

[12] 周慈航. 嵌入式系统软件设计中的常用算法［M］. 北京：北京航空航天大学出版社，2012.