

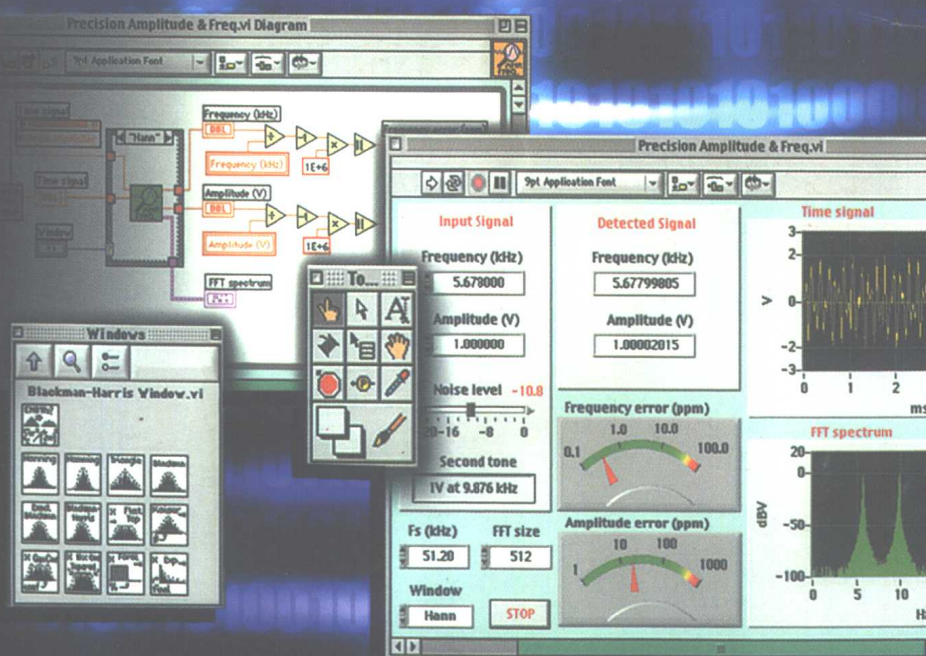


LabVIEW Graphical programming

LabVIEW

图形编程

[美] Gary W. Johnson 著
Richard Jennings 译
武嘉澍 陆劲昆 译



第一版 第一次

内附全部示例光盘

Mc
Graw
Hill

北京大学出版社
<http://cbs.pku.edu.cn>

Mc
Graw
Hill Education
<http://www.mheducation.com>

LabVIEW 图形编程

[美] Gary W. Johnson 著
Richard Jennings
武嘉澍 陆劲昆 译

北京大学出版社

• 北 京 •

内 容 简 介

本书由两位顶尖资深的仪器开发专家——Gary W. Johnson 和 Richard Jennings 编写,旨在为致力于数据采集与控制、数据分析,以及数据表达等方面的科学家和工程师们提供一种全新的程序编写方法,即对称之为“虚拟仪器”(Virtual Instruments, VIS)的软件对象进行图形化的组合操作。书中讲解了如何通过交互式的图形化前面板来控制系统,并显示所得的结果;如何利用 LabVIEW 中功能强大的数据分析程序将原始数据转换成有意义的结果;以及如何利用多种设备进行数据采集等方面。本书内容全面,结构清晰,是一本 LabVIEW 用户的首选参考书。

著作权登记号: 01-2001-1929

Gary W. Johnson Richard Jennings : LabVIEW Graphical Programming
ISBN: 0-07-L3700L-3

Copyright © 2001 by the McGraw-Hill Companies, Inc.

Authorized translation from the English language edition published by McGraw-Hill, Inc.

All rights reserved. For sale in the People's Republic of China only.

本书中文简体字版由北京大学出版社和美国麦格劳-希尔国际公司合作出版。未经出版者书面许可,不得以任何方式复制或抄袭本书的任何部分。

版权所有,翻印必究。

图书在版编目(CIP)数据

LabVIEW 图形编程/(美)约汉逊(Johnson,G.W.), (美)詹宁斯(Jennings,R.)著;武嘉澍,陆劲昆译. —北京:北京大学出版社, 2002.1

ISBN 7-301-05512-9

I. L... II. ①约... ②詹... ③武... ④陆... III. 图形软件, LabVIEW—程序设计 IV. TP317.4

中国版本图书馆 CIP 数据核字(2002)第 010503 号

书 名: LabVIEW 图形编程

著作责任者: [美] Gary W. Johnson Richard Jennings

译 者: 武嘉澍 陆劲昆

责任编辑: 徐丽萍

标准书号: ISBN 7-301-05512-9/TP · 0652

出 版 者: 北京大学出版社

地 址: 北京市海淀区中关村北京大学校内 100871

网 址: <http://cbs.pku.edu.cn>

电 话: 发行部 62754140 62765127 编辑室 62765126 邮购部 62752015

电 子 信 箱: macrowin@263.net.cn

排 版 者: 北京东方人华科技有限公司

印 刷 者: 河北省滦县滦兴书刊印刷厂

发 行 者: 北京大学出版社

经 销 者: 787 毫米×1092 毫米 16 开本 39.5 印张 821 千字

2002 年 4 月第 1 版 2002 年 4 月第 1 次印刷

定 价: 78.00 元(含光盘)

前言

LabVIEW 是一个划时代的图形化编程系统，应用于数据采集与控制、数据分析，以及数据表达等方面，它提供了一种全新的程序编写方法，即对称之为“虚拟仪器”(Virtual Instruments, VIS)的软件对象进行图形化的组合操作。据独立的市场调查分析，LabVIEW 是科学家和工程师们进行仪器应用开发的首选工具。

LabVIEW 让您充分享受功能强大的编程语言才具备的灵活性，而且简单易用。如果您想对仪器系统进行编程，而又不希望降低其性能，那么就需要选择这种用于测试与测量、数据采集与控制，以及过程监控的革命性软件开发一套完整的仪器系统。

利用 LabVIEW，您通过交互式的图形化前面板来控制系统，并显示所得的结果。您可以利用几千种设备进行数据采集，这些设备包括：GPIB、VXI、串口设备、PLC，以及插入式数据采集卡等。您也可以通过网络、交互应用通讯和结构化查询语言(SQL)等方式与其他的数据源相联。数据采集完之后，您可以利用 LabVIEW 中功能强大的数据分析程序，将原始数据转换成有意义的结果。

本书是专门为那些用过 LabVIEW 软件的用户编写的。我们建议您参加一些关于 LabVIEW 课程的培训。如果您是刚开始学习，则可以利用其他内容更浅显的书作为参考。在本书的第 2 章中，我们提到了一些基本参考和建议。要跟上现代软件的增强功能是一件极具挑战的事情，特别是类似于印刷卷的书本，本书第 3 版是建立在 LabVIEW 6 的基础上，但我们与 National Instrument 公司保持着紧密的联系，所以可以保证本书在 LabVIEW 以后的版本中长期可用。您将会发现在本书中讨论的一些原理和技术都是在使用 LabVIEW 过程中最根本的东西，这将使得您从本书中掌握的知识不会随着软件的发展很快被淘汰。

本书尽可能采用独立的平台来讲述，就像 LabVIEW 本身是独立的一样，只有在必须从其他一台或两台计算机上获得功能时，才偶尔会出现多平台的情况。LabVIEW 的用户手册包含了易查询的指南，当您在跨平台开发应用程序时遇到问题，可以在该手册中查找相应的指南。

本书还附带了光盘，光盘中包括一些应用程序记录、许多实用的 VI，以及一些工作的应用程序示例，其中许多 VI 都在书中详细讨论了。光盘中还包含了在与 Linux 相关章节中

用到的所有源代码和可执行图像。其他一些重要资源还可以通过互联网查找，为了您使用方便，我们已在相应的章节中列出了一些网址，以及一些重要提供商的主页和邮件地址。

最后，希望我们对 LabVIEW 的热情能够感染您！

作者

2001 年 4 月

目 录

第 1 章 起源.....	1	1.2.19 LabVIEW 6.....	26
1.1 LabVIEW 和自动化.....	1	1.3 预言未来.....	26
1.1.1 虚拟仪器: LabVIEW 的基础 ...	3	1.4 LabVIEW 参与的大项目.....	28
1.1.2 为什么要使用 LabVIEW	5	第 2 章 开始	31
1.2 LabVIEW 的起源.....	6	2.1 如何装配一个系统	31
1.2.1 简介	7	2.1.1 使用熟悉的设备	32
1.2.2 可视化编程的出现.....	7	2.1.2 选择 I/O 系统.....	32
1.2.3 整个世界就是一台仪器.....	9	2.1.3 选择一台计算机	37
1.2.4 一个顽固的 UNIX 拥护者		2.1.4 其他要考虑的软件	40
被 Macintosh 征服.....	10	2.2 帮助和问答的资源	41
1.2.5 用图片拼成一个整体	10	2.2.1 接受培训	42
1.2.6 关注系统设计失败的平台	12	2.2.2 使用实例 VIs	42
1.2.7 疯狂的开发过程.....	13	2.2.3 当所有方法失败时,	
1.2.8 突破工具和机器的限制.....	14	阅读目录	43
1.2.9 在估计开发时间方面		2.2.4 网络上的 LabVIEW.....	43
面对现实.....	15	2.2.5 因特网邮件组	44
1.2.10 发售第一版.....	16	2.2.6 来自 National Instruments	
1.2.11 LabVIEW 提供的潜力		公司的邮件帮助	45
成就了 Apple.....	16	2.2.7 频繁问及的问题(FAQ)文档.....	45
1.2.12 LabVIEW2.....	19	2.2.8 LabVIEW 技术资源(LTR).....	46
1.2.13 向 Windows 和 Sun		2.2.9 用户组和 VI 用户网	46
上移植系统.....	20	2.2.10 联盟表: 顾问及其他	47
1.2.14 LabVIEW 3.....	21	2.2.11 教育的应用软件	47
1.2.15 LabVIEW 4.....	22	2.2.12 (800)IEEE 488.....	47
1.2.16 LabVIEW 分支到		2.3 参考文献.....	48
Bridge VIEW	23	第 3 章 输入和输出	49
1.2.17 LabVIEW 5.....	23	3.1 信号的起源	49
1.2.18 LabVIEW RT 分支	25	3.1.1 转换器和传感器	49

3.1.2 激励器.....	52	5.5.2 局部变量.....	105
3.1.3 信号的分类.....	52	第 6 章 LabVIEW 数据类型.....	109
3.2 连接.....	56	6.1 数值类型.....	109
3.2.1 接地和屏蔽.....	57	6.2 字符串.....	111
3.2.2 为什么使用放大器或者 其他信号调节.....	62	6.2.1 创建字符串.....	111
3.2.3 选择正确的 I/O 子系统.....	69	6.2.2 分析字符串.....	112
3.2.4 用网络连接一切.....	73	6.2.3 处理不宜打印的字符.....	115
第 4 章 抽样信号.....	74	6.2.4 电子数据表格、字符串 以及数组.....	115
4.1 抽样法则.....	74	6.3 数组.....	118
4.2 滤波与取平均值.....	76	6.3.1 初始化数组.....	122
4.3 有关 ADCs、DACs 以及 多路转换器.....	77	6.3.2 数组内存的使用和性能.....	123
4.3.1 数模转换器.....	81	6.4 群集.....	126
4.3.2 数字码.....	82	6.5 波形.....	128
4.4 触发和定时.....	83	6.6 数据类型转化.....	131
4.5 少量噪音可以成为好事.....	84	6.6.1 转化和强迫.....	131
4.6 吞吐量.....	87	6.6.2 复杂的转化和类型构造.....	133
4.7 参考文献.....	88	6.6.3 Flatten To String.....	136
第 5 章 控制程序流.....	90	6.6.4 枚举类型(enums).....	137
5.1 有关这本书的图表.....	90	6.6.5 Get Carried Away Department.....	138
5.2 定序和数据流.....	91	第 7 章 定时.....	139
5.2.1 Sequence 结构.....	92	7.1 小型定时器的来历.....	139
5.2.2 数据从属.....	93	7.2 使用内置定时功能.....	140
5.2.3 增加公共线程.....	94	7.2.1 时间间隔.....	141
5.3 循环.....	95	7.2.2 绝对定时函数.....	142
5.3.1 For 循环的微妙.....	95	7.3 将定时数据传送至其他应用程序.....	143
5.3.2 奇妙的 While.....	96	7.4 高精度和高准确定时.....	145
5.4 移位寄存器.....	98	第 8 章 同步化.....	147
5.5 全局变量和局部变量.....	100	8.1 轮询.....	147
5.5.1 内置全局变量以及 它们的危险.....	102	8.2 事件.....	150
		8.3 通告程序.....	153

8.4	信号量	155	10.4.6	通过抄袭进行程序设计	201
8.5	会合机制	157	10.4.7	请关注自己的数据	202
第 9 章	文件	159	10.4.8	勾画出程序结构	204
9.1	存取文件	159	10.4.9	编写伪代码	205
9.2	文件类型	161	10.4.10	语言翻译	207
9.3	写文本文件	161	10.4.11	范围、强制转换以及 默认值	208
9.4	读文本文件	165	10.4.12	处理错误	209
9.5	格式化为文本文件	166	10.4.13	将它们都放在一起	212
9.6	二进制文件	167	10.5	测试和调整自己编写的程序	213
9.6.1	写二进制文件	168	10.5.1	搞清楚子 VIs 所胜任 的任务	213
9.6.2	读二进制文件	170	10.5.2	偷看数据	214
9.7	写数据流文件	172	10.5.3	每次一步	214
9.8	读数据记录文件	173	10.5.4	Execution highlighting	215
9.9	数据记录文件应用	174	10.5.5	设置断点	216
第 10 章	Gary 创建一个应用 程序的方法	176	10.5.6	调试全局变量	217
10.1	定义问题	177	10.5.7	调试局部变量和属性节点	218
10.1.1	分析用户需求	177	10.5.8	追踪执行	218
10.1.2	收集规格	177	10.5.9	检查性能	220
10.1.3	画一个方框图	179	10.6	最后一击	221
10.2	指定 I/O 硬件	181	10.7	参考文献	222
10.3	用户界面的原型	182	第 11 章	文件编制	223
10.3.1	用户界面设计	183	11.1	VI 描述	223
10.3.2	可能的面板	183	11.2	控件描述	224
10.4	设计, 然后再编写程序	186	11.3	自定义在线帮助	225
10.4.1	请求范例	187	11.4	文件编制图表	225
10.4.2	自顶而下还是自底而上	187	11.5	VI 历史记录	226
10.4.3	模块化	189	11.6	其他文件编制的方法	226
10.4.4	选择一个体系结构: 规范的 VIs	190	11.7	打印 LabVIEW 面板和图表	227
10.4.5	作为设计工具的 VI 分层 体系结构	200	11.8	编写规范的文件	229
			11.8.1	文件概述	230
			11.8.2	连接器窗体图像	230

11.8.3 VI 描述	231	第 14 章 DAQ 库的使用	284
11.8.4 终端描述	231	14.1 NI_DAQ 驱动程序	284
11.8.5 编程实例	232	14.2 NI_DAQ 通道模板	286
11.9 分配文件	232	14.3 硬件选择	286
第 12 章 设备驱动程序的基本知识	234	14.3.1 选择一个平台和总线	286
12.1 有关设备库	234	14.3.2 多功能输入输出板	287
12.2 可相互交换的虚拟设备(IVI)	235	14.3.3 专用模拟输入输出板	287
12.3 驱动程序基础	236	14.3.4 数字和输入输出定时板	288
12.3.1 通信标准	236	14.3.5 SCSI	288
12.3.2 了解自己的设备	247	14.3.6 便携式 DAQ	289
12.3.3 决定要编程的函数	248	14.3.7 直接存储器访问(DMA)	289
12.3.4 建立通信	249	14.4 DAQ 库概况	290
第 13 章 设备驱动程序的开发技术	255	14.4.1 DAQ 库的层级	290
13.1 驱动程序体系结构	255	14.4.2 DAQ 库中的规范	292
13.2 通过功能的分组进行模块化	256	14.4.3 DAQ 例子和模板	298
13.3 错误 I/O 流程控件	258	第 15 章 模拟 DAQ	300
13.4 从一个模板开始	263	15.1 模拟输入	300
13.5 打开一个连接器	264	15.1.1 配置、启动和停止	300
13.6 使用一个通信子 VI	265	15.1.2 低速采集	302
13.6.1 GPIB 通信子 VI	265	15.1.3 中速采集和处理	306
13.6.2 串行通信子 VI	267	15.1.4 高速磁盘流	309
13.6.3 VXI 通信子 VI	271	15.1.5 特殊采样	311
13.7 在同一子 VI 中的读和写	271	15.2 模拟输出	321
13.8 在一个驱动程序 VI 中的 多个函数	273	15.2.1 设置和停止	321
13.9 定时、信息交换以及触发	275	15.2.2 简单更新	322
13.9.1 GPIB 信息交换	275	15.2.3 波形生成	323
13.9.2 串行信息交换	277	15.2.4 同步模拟输入和输出	327
13.10 范围检查	278	15.3 参考文献	328
13.11 设置管理	280	第 16 章 数字数据采集	329
13.11.1 文件编制	282	16.1 数据 I/O	329
13.12 参考文献	283	16.1.1 简单的位 I/O	329
		16.1.2 支持信息交换的数字 I/O	331

16.2 计数器与计时器.....	334	18.2.2 其他的智能 I/O 子系统	415
16.2.1 Am9513 系统定时		18.3 人机界面	415
控件的说明	335	18.3.1 显示层次	420
16.2.2 事件计数器	336	18.3.2 其他有趣的显示技术	424
16.2.3 间隔时间、持续时间和		18.3.3 处理所有前面板的项目	425
周期计时器	338	18.4 数据分配	426
16.2.4 脉冲产生	341	18.4.1 用作服务器的输入扫描器	426
16.2.5 频率计数	342	18.4.2 处理输出数据	427
第 17 章 编写数据采集程序	344	18.4.3 使用网络连接	432
17.1 数据分析与储存	345	18.4.4 实时过程控制数据库	434
17.1.1 运行后分析	346	18.4.5 通过模拟得到验证	435
17.1.2 实时分析与显示	356	18.5 顺序控制	435
17.2 取样与吞吐量	363	18.5.1 逻辑和表格互锁	436
17.2.1 信号带宽	364	18.5.2 状态机(state machines)	437
17.2.2 过度抽样和数字滤波	364	18.5.3 初始化问题	438
17.2.3 定时技术	369	18.5.4 GrafcetVIEW	439
17.3 配置管理	370	18.6 连续控制	440
17.3.1 配置内容	370	18.7 趋势显示	446
17.3.2 配置编辑器	372	18.7.1 实时趋势	446
17.3.3 配置编译器	382	18.7.2 历史趋势	449
17.3.4 保存和再调用设置	384	18.7.3 统计过程控制	452
17.3.5 一个低速数据采集		18.8 警报	453
的例子	388	18.8.1 使用一个警报处理器	454
17.4 参考文献	391	18.8.2 通知操作者的技术	457
第 18 章 过程控制应用程序	392	18.9 商用软件	458
18.1 过程控制基本原理	393	18.10 参考文献	460
18.1.1 工业标准	393	第 19 章 物理学应用	461
18.1.2 控制 = 操纵输出	398	19.1 专用的硬件	461
18.1.3 过程信号	400	19.1.1 信号调节	462
18.1.4 控制系统的体系结构	402	19.1.2 CAMAC	465
18.2 使用智能控制器	407	19.1.3 其他的 I/O 硬件	465
18.2.1 可编程逻辑控制器(PLC)	407	19.2 场和等离子体诊断	466
		19.2.1 步进测量实验	467

19.2.2 等离子体电势实验.....	472	21.3.1 检测性能.....	531
19.3 处理快速脉冲.....	477	21.3.2 共享资源.....	535
19.3.1 瞬时数字化仪.....	477	21.3.3 多线程和多任务处理.....	536
19.3.2 数字存储示波器(DSO).....	480	21.3.4 组织 VI 以获得最佳的 实时性能.....	538
19.3.3 定时和触发.....	481	21.3.5 前后关系转换会增加 时间花费.....	540
19.3.4 捕捉多个脉冲.....	483	21.4 时序安排(调度).....	541
19.3.5 从同步实验复现信号.....	486	21.4.1 循环调度程序.....	541
19.4 处理巨量数据集.....	488	21.4.2 静态调度程序.....	542
19.4.1 减少数据的数量.....	489	21.5 通讯.....	543
19.4.2 为内存占用率优化 VI.....	489	21.6 构建应用程序.....	545
19.5 参考文献.....	495	21.7 参考文献.....	546
第 20 章 数据可视化、成像和声音.....	496	第 22 章 在虚拟机上嵌入 LabVIEW for Linux.....	547
20.1 成像.....	497	22.1 为什么是 Linux.....	548
20.1.1 显示波形和笛卡儿数据.....	498	22.1.1 Linux 资源.....	549
20.1.2 二元数据.....	503	22.1.2 本书附带 CD-ROM 上的 Linux.....	549
20.1.3 多元数据.....	504	22.2 使用 PeeWeeLinux 构建 嵌入式系统.....	550
20.2 三维图形.....	508	22.3 VMware Workstation 建立虚拟机.....	551
20.3 图形采集和处理.....	511	22.4 在 Linux 中建立一个新的 VMware Workstation 虚拟机.....	553
20.3.1 成像的系统需求.....	512	22.5 构建嵌入式 Linux 系统 的 6 个步骤.....	556
20.3.2 使用 IMAQ Vision.....	514	22.5.1 虚拟硬盘驱动器分区.....	556
20.3.3 IMAQ 组件.....	514	22.5.2 格式化虚拟硬盘驱动器.....	557
20.4 声音输入/输出.....	522	22.5.3 装入目标分区.....	558
20.4.1 DAQ 实现声音输入/输出.....	522	22.5.4 从 CD-ROM 传送 操作系统.....	558
20.4.2 声音输入/输出功能.....	523	22.5.5 安装引导装入过程.....	559
20.4.3 声音输入.....	523	22.5.6 启动新的嵌入式系统.....	559
20.4.4 声音输出.....	523		
20.4.5 声音文件.....	525		
20.5 参考文献.....	525		
第 21 章 LabVIEW RT 处理实时任务.....	526		
21.1 实时不意味着快速.....	526		
21.2 RT 硬件.....	528		
21.3 设计软件满足实时要求.....	530		

22.6 示例嵌入式 Linux 的详述	559	23.7.7 启动新的嵌入式系统	590
22.6.1 目录结构	560	23.8 检测和故障检修	591
22.6.2 引导过程	561	23.8.1 调试器	591
22.6.3 配置文件	562	23.8.2 窗口管理器	591
22.6.4 通过 NFS 装入传送 OS	565	23.9 在嵌入式领域中使用 LabVIEW	592
22.7 VMware-mount 工具和 Linux	566	23.10 为自定义启动屏幕而 修改 Linux 内核	592
22.8 帮助性的提示	567	23.11 参考文献	596
22.9 进行 LabVIEW 开发	568		
22.10 转到真实的硬件	568	第 24 章 LabVIEW 与实时 Linux	
22.11 参考文献	568	的接口技术	597
第 23 章 嵌入式系统中的 LabVIEW		24.1 综述	598
和 Linux	569	24.1.1 什么是实时 Linux	598
23.1 为什么使用桌面操作系统 构建嵌入式系统	570	24.1.2 更多的信息来源	600
23.2 Linux 能带来的帮助	571	24.2 安装实时 Linux	601
23.3 平台	571	24.3 在实时 Linux 中编程	602
23.4 在快速存储器里存储 嵌入式系统	572	24.4 实时 FIFO	602
23.5 I/O 硬件驱动程序	574	24.4.1 Rtf_open 和 Rtf_close VI	604
23.5.1 硬件驱动程序以及何处 可以获得硬件驱动	575	24.4.2 使用 Rtf_get 和 Rtf_read VI 读取数据	604
23.5.2 把驱动程序整合到 系统中	576	24.4.3 使用 Rtf_put 和 Rtf_write VI 写数据	605
23.6 构建一个真实的嵌入式系统	578	24.5 共享内存	605
23.7 成功的 7 个步骤	581	24.5.1 Mbuff_open 和 Mbuff_close VI	606
23.7.1 硬盘驱动器分区	582	24.5.2 使用 Mbuff_get 和 Mbuff_read VI 读数据	607
23.7.2 格式化硬盘驱动器	583	24.5.3 使用 Mbuff_put 和 Mbuff_write 写数据	608
23.7.3 装入硬盘驱动器	584	24.6 处理结构数据	608
23.7.4 传送操作系统和嵌入式 应用软件	584	24.7 Lvrtd 软件包	609
23.7.5 为新的硬件修改安装	584	24.8 实时 FIFO 的 POSIX 方法	610
23.7.6 安装引导装入过程	589	24.9 参考文献	613

第 1 章 起 源

毫无疑问，LabVIEW 让工程师 Gary 的日子过得比以前轻松多了。我至今对 80 年代早期工作的情况记忆犹新。那时候工作量很大，因为要写长得可怕的程序才能完成看上去很简单的测量和控制作业。除非绝对必要，科学工作者和工程师们才允许让他们的作业自动运行。由于软件的复杂程度高，非专业级的用户和操作人员都不敢摆弄。肯定地讲，在基于计算机的仪器制造事业里，与所能获得的乐趣相比，更多的是工作的艰辛。但是到了 1987 年年中，情况一切都变了。当时我参加了 National Instruments 公司的一个产品展示会，会上展出了一种在苹果机上运行的新软件，据说这个软件可以做与仪器相关的事情，这个消息引起我的兴趣。当我亲眼看到软件所能做到的事情，也就是 LabVIEW 所能做到的事情时，我感到异常兴奋！卷动图标来编写程序？图形控制？这一切真使我感到不可思议！我打定主意搞到这个软件，准备亲自动手试一试。

到了当年的年底，我参加完了 LabVIEW 的课程班后，开始着手我的第一个项目(一个简易的数据采集系统)。当时的感受就好像看一轮红日升起那样美好。有了这种使用简便并且用起来很好玩的编程语言，真是可以做很多事情。我开始在实验室附近找些事情，设法用 LabVIEW 来做一下。(请相信我，我真的找到了)。就这样，我彻底地改变旧有的工作方式。不到一年的光景，LabVIEW 成了我的仪器和控制仪器事业中不可缺少的工具。目前，我的实验室不仅可以说是计算机化了，简直可以称得上自动化了。一个用计算机处理实验或处理主要还是靠操作人员完成——尽管计算机通过承担测量或类似的简单作业能够将任务完成得简单一些，但是还远没有达到完全脱离人工处理的地步。然而，一个自动化处理的实验的做法就迥然不同了。用户只需装配好实验器材，在 LabVIEW 屏幕上按下开始(Start)按钮，然后就可以看着计算机编辑事件序列、进行测量、提交试验结果了。这正是用户所希望的系统工作方式，这也是 LabVIEW 的长处，它能节约大量的工作时间。让我们出发去看看自动化的世界吧。

1.1 LabVIEW 和自动化

计算机被认为可以把事情做得更简单、更迅速或更自动化。也就是说，计算机可以为它的主人人类分担工作。LabVIEW 是一种很独特的编程系统，它可以帮助从事多年实验室研究、工业控制和数据分析的科学工作者或工程师们轻松地实现计算机自动化。假设有一

项工作(某人付钱给你让你完成某项任务)要做,只要正确地应用 LabVIEW,它就可以帮助你将工作搞定。但是围绕着计算机及其对我们生产率的影响这一问题,在整个业界引起了广泛的争论。例如在我最近阅读过的一篇文章中报道说,只要有一名打字员,人类就有可能写出比以往更长的提议书和报告(当然,还能更工整)。现代化的文字处理软件使得人类进行全面而有效的思想沟通变得容易。然而,这种现代化手段总能提高生产效率和质量吗?有时候,我们实际上把更多的时间花在做重复的事情上。我们成为了计算机的奴隶,整天为设置、安装新的软件升级版本(是否考虑过这么做有没有必要呢?)等琐事操劳。做这些事通常就是浪费时光。

用户必须避免掉入这样的陷阱。避免掉入陷阱的关键是分析所面对的问题,考虑如何利用 LabVIEW 和专门的计算机硬件来使它们发挥最大的功效,然后有效地运用现有的 LabVIEW 解决方案。正如你将看到的,许多实验室自动化问题已经为用户解决掉了,已经有现成的程序和仪器可供使用。这其实也没有什么神秘的,只不过是用户要做出一些关于计算机自动化利弊的简单工程决策。让我们从实际角度出发考虑这种解决方案。现实中存在着很多需要自动化的操作,其中包括:

- 周期长、速度低的操作。例如,环境监视和控制。
- 高速操作。例如,在短时间内要收集大量数据的脉冲功率诊断。
- 重复性的操作。例如,自动化的测试和校正,以及需要执行多次的实验。
- 远距离操作(远程控制)或具有危险性的操作。让操作人员在现场执行这种操作是不切实际或不可能的,或者操作人员在现场会有危险。
- 超过人类自身能力的高精度操作。
- 需要大量输入和输出的复杂操作。

请注意,在上述例子中,计算机自动化系统使这些操作或实验变得实际可行,而要做这些操作或实验,除了采取自动化系统以外,的确不可能再做其他选择。另外,自动化还具有如下优势:

- 可以减少数据转录错误。老式的“数据采集系统”(一杆铅笔)比起计算机数据采集系统更容易产生错误。的确,更具可靠性的数据通常有利于更好地控制产品质量,并能在实验时导致新的发现。
- 可以在处理或数据采集过程中消除因操作人员失误引起的偏差。由于计算机工作起来不知疲倦,并且做事情时总是保持同样方式不变,因而极大地提高了做重复工作的效率。
- 增加数据的吞吐量。因为可以按照计算机的速度而不是人的速度操作系统。

然而,在这种计算机自动化处理中也隐藏着一些弊端:

- 自动化可能带来新类型的错误。例如,由于不正确地使用了传感器、信号条件、

数据转化导致的错误,偶尔是由于计算错误(例如四舍五入)导致的错误。

- 对硬件或软件系统的误用也有可能招致麻烦。例如,试图用过高的速率收集数据会导致数据记录错误。
- 可靠性也是计算机系统的一个重要问题。系统错误(系统崩溃)和软件错误(bug)给每种高科技产品的安装都带来了麻烦,它们同样也会给系统带来麻烦。

永远要考虑每种潜在的自动化解决方案的性价比。当今这个社会,好像没有一件事不是受金钱驱动的。如果想把某件事办得又省钱,又省时间,还得更漂亮些,看来只能征得业主、股东或主顾的同意才行。但是,计算机能保证给用户省钱或节约时间吗?如果在一个一次性的实验中,我有充分的时间用一只带状记录纸记录器,一台示波器,或者用支铅笔来记录数据的话,花两天的时间去写一段专用程序就没有任何意义了。

一个自动处理(或者至少用计算机处理)简单的、一次性的试验的方法是构造一辆 LabVIEW 紧急救护车,这种车非常类似于急诊室里医生的救护车。当某个人需要在短时间内解决一个测量问题时,我可以带着我的仪器行李架前去工作。行李架里包括装着 LabVIEW 的 Macintosh 机、模拟界面硬件,以及一些可编程的工具。我可以迅速地配置通用的数据采集程序,记录数据并进行分析,完成所有这一切不超过几个小时。如果用户在需要多种数据采集的地区工作,可能要考虑这个概念:使用任何用户所有的多余仪器,重复利用经过试验运行正确的程序,将它们堆在一个车上。使用什么样的计算机都不要紧,因为 LabVIEW 可以在各种版本的 Windows、Power Macintosh、Sun SPARC 工作站、Linux、以及 HP 工作站上运行。紧急救护车的概念简单并且极为有效。

自动化的费用是很大的,其中包括传感器、计算机和软件的费用,以及随着程序员的工作量而迅速增加的程序编制费用。但是最终,不可思议的新能力将会出现,研究人员会一下子从记录和解释数据等烦琐的工作中解放出来。操作人员再也不必编制这么多的重要调整,数据质量和产品质量也会提高。如果用户的情况适合自动化的基本条件,请一定考虑 LabVIEW 作为解决方案。

1.1.1 虚拟仪器: LabVIEW 的基础

LabVIEW 使得虚拟仪器(VI)这一概念成为实际可行的现实。在虚拟仪器中的对象是使用通用计算机和专用的控制器和显示器来模拟实际的仪器,但是由于软件的作用,使虚拟仪器表现出一些新的功能(如图 1.1)。用户无需再购买带状记录纸记录器、示波器和频谱分析仪这些旧仪器,而仅需购买一台性能优异的数模转换器,然后使用一台运行着 LabVIEW 的计算机,这样就可以模拟所有这些仪器甚至更多的仪器。

虚拟仪器与实际仪器的关系

类似于 LabVIEW 这样的虚拟仪器系统不可避免地引起人们要把它们同实际仪器进行比较。使用个人电脑来执行虚拟仪器的主要缺陷是：计算机只有一个中央微处理器。一个使用多台仪器的应用程序很容易使微处理器负载过重。然而一台独立的仪器可以包括任意多个处理器，每个处理器可以承担专门的作业。

此外，如果需要增强整体的性能，这些多处理器可以按照并行的方式运行。但是这种整体功能的增强不仅降低灵活性，还要为添置许多专用仪器而支付高昂的费用。

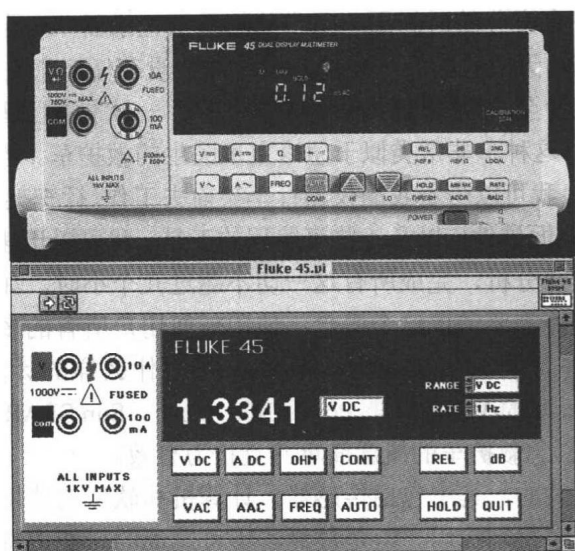


图1.1 这台虚拟仪器(底下的图)是实际仪器的定制版本，
只具备用户所需要的一种特征

然而，来自 National Instruments 公司的先进技术解决了这些问题。首先，LabVIEW 的新版本——LabVIEW RT 允许用户在多处理器、专用处理器和实时处理器上独立运行程序，这样就可以降低桌面主机的工作负担。其次，LabVIEW 允许用户通过各种网络系统联接各种计算平台以及其他软件，从而进一步将主机的工作量分配出去。再次，目前插入式主板里都设置了自身的处理器，而且价钱非常合适。数字信号处理器就是专门用途的处理器中设置在插入式主板上一个范例。许多插入式的数据采集主板同时还设置了技术先进的直接存储器存取、定时以及兼容于多种主板的触发功能，增强了主板之间的同步性和信号耦合性。这些技术上的进步，再加上操作系统功能和计算机体系结构方面的发展，已经使得个人电脑具有了并行处理的能力，成为更加先进的仪器和数据采集应用程序的平台。然而，技术的进步是以复杂性的提高为代价的。这么说的原因是：比起使用具有类似功能的一台

独立仪器，用户必须具备更多有关的硬件组成与硬件之间相互关系的知识。虚拟仪器软件对于将这些技术尖端的硬件组件变成实际可用的仪器系统是必不可缺的。

虚拟仪器在性价比、灵活性以及用户化方面，具有实际的仪器不可比拟的优势。以一台高性能仪器的价钱，用户可以装配一台装有基于基本硬件和软件部分的个人计算机系统，为专门的应用软件设计虚拟仪器。硬件部分可以是插入式主板、外围仪器，或者两者的结合。在任何一种情况下，软件界面可按照应用软件的需要或复杂或简单。用户可以通过虚拟仪器来简化一台复杂的独立仪器的操作，因为虚拟仪器只是集中控制仪器全部功能的一个子集。我曾经被许多现代化仪器面板上的按钮菜单等搞得晕头转向，现在终于找到了一个简单的用户界面，这实在值得欣慰。

尽管 LabVIEW 从 1986 年起就问世了，但其设计中所体现的虚拟仪器和方块图的概念至今仍然处于仪器和计算机科学的领先地位。随着被测试的装置和测试这些装置所需的仪器复杂性不断增加，开发测试程序的成本持续上升。软件的模块性、可维护性、可重复利用性，以及 LabVIEW 的分层和同质结构的关键作用，对于减轻每个软件开发人员的负担是至关重要的。重新利用已经写好的程序并与其他人员分享可以节约大量的时间，并使得程序更具可靠性。

在仪器控制领域，虚拟仪器变得日益重要了。仪器的 VMEbus 扩展(VXI)——仪器领域一个主要的发展方向——是定义物理电子参数的标准，也是用来实现卡式仪器测试系统的软件协议。VXI 是执行插卡仪器测试系统的标准，定义了物理和电子参数以及软件的协议。一台 VXI 仪器是插入包含若干插片的底座的插卡。由于这种仪器是插入式插卡而非独立的箱子，单个的 VXI 仪器就没有前端用户界面。用户不能通过按按钮或阅读前面板的显示信息来同 VXI 仪器进行交互。相反，VXI 系统必须由一台计算机或其他用处理器驱动的设备管理，这样就对控制仪器的计算机软件提出了新的要求。

VXI 仪器是虚拟仪器实施的当然候选者。在用户交互方面，软件前面板提供了一种控制不露面的 VXI 仪器的可视方式。而且，插入式模块和 VXIbus 高性能的定时和通信功能的结合使得 VXI 测试系统的配置比通用接口总线(GPIB)测试系统的配置复杂得多。LabVIEW 提供的流程图将虚拟仪器连接到方块图的方法加快了 VXI 测试系统配置和编程的速度。

1.1.2 为什么要使用 LabVIEW

我使用 LabVIEW 是因为 LabVIEW 比起传统的语言和其他控制及数据采集程序包具有更大的优势。

与使用传统的编程语言相比，LabVIEW 大大地提高了我的生产效率。我曾经用 C 语言在小的项目上作过比较测试，得出结论是：用 C 语言的效率要比用 LabVIEW 的效率低 4/5。