第5章 虚拟仪器开发与车辆底盘系统测试案例

5.1 虚拟仪器及开发软件

5.1.1 虚拟仪器的基本概念

虚拟仪器(Virtual Instrument,VI)是智能仪器之后的新一代测量仪器,是在电子仪器与计算机技术更深层次结合的基础上产生的一种新的仪器模式。所谓虚拟仪器实际上是一种基于计算机的自动化测试仪器系统,是现代计算机技术和仪器技术结合的产物,是当今计算机辅助测试(CAT)领域的一项重要技术。虚拟仪器通过相应的软件与仪器模块相连接,将计算机硬件资源与仪器硬件有机地融合为一体,从而把计算机强大的计算处理能力和仪器硬件的测量、控制能力结合在一起,降低了仪器硬件的成本,缩小了仪器体积,并通过较为友好的图形界面和强大的数据处理能力完成对测量数据的显示和分析。

虚拟仪器主要由硬件系统和软件系统两部分构成。硬件系统一般分为计算机硬件平台和测控功能硬件,计算机硬件平台可以是各种类型的计算机,如普通台式机、便携式计算机、嵌入式计算机、工作站等,它为虚拟仪器的软件提供运行平台。硬件部分一般包括各种形式的数据采集设备,将采集到的各种形式的信号转换成相应的电信号后输入到计算机内。按照测控功能硬件的不同,可将虚拟仪器分为 PC-DAQ、GPIB、VXI、PXI 和 LXI 总线五种标准体系结构。

基本硬件确定之后,要使虚拟仪器能按用户要求自行定义,必须有功能强大的应用软件,然而相应的软件开发环境长期以来并不理想,用户花在编制测试软件上的工时与费用相当高,即便使用 C、C++、C♯、Java 等高级语言,也不能满足缩短开发周期的要求。因此,世界各大公司都在改进编程及人机交互方面做了大量的工作,其中基于图形的用户接口和开发环境是软件工作中最流行的发展趋势。典型的软件产品有 NI 公司的 LabVIEW 和 LabWindows/CVI,安捷伦公司的 VEE,微软公司的 Measurement Studio for VB 等。

与传统仪器相比,虚拟仪器有以下优点:

(1) 融合计算机强大的硬件资源,突破了传统仪器在数据处理、显示、存储等方面的限制,大大增强了传统仪器的功能。

(2) 利用了计算机丰富的软件资源,实现了部分仪器硬件的软件化,增加了系统灵活性。通过软件技术和相应的数值算法,可以实时、直接地对测试数据进行各种分析与处理。同时,图形用户界面技术使得虚拟仪器界面友好、人机交互方便。

(3) 基于计算机总线和模块化仪器总线,硬件实现了模块化、系列化,提高了系统的可靠性和可维护性。

(4) 基于计算机网络技术和接口技术,具有方便、灵活的互联能力,广泛支持各种工业总线标准。利用虚拟仪器技术可方便地构建自动测试系统,实现测量、控制过程的智能化、网络化。

(5) 具有计算机的开放式标准体系结构。虚拟仪器的硬件、软件都具有开放性、可重复使用及互换性等特点。用户可根据自己的需要选用不同厂家的产品,使仪器系统的开发更为灵活、效率更高,缩短系统构建时间。

虚拟仪器系统现已成为快速构建测试系统的一个基本方案,它是科学技术发展的必然结果。虚拟仪器技术十分符合国际上流行的"硬件软件化"的发展趋势,被广泛地称为"软件仪器"。虚拟仪器以计算机技术为基础,随着计算机技术的高速发展,虚拟仪器将向智能化、网络化的方向发展。虚拟仪器的技术优势使其应用非常广泛,尤其在科研开发、测量、检测、控制等领域。随着时间的推移,其必将对科学技术的发展和国防、工业、农业、航天等领域的进步产生巨大影响。

5.1.2 LabWindows/CVI 软件

1. 简介

一旦提及虚拟仪器开发软件,可能最先联想到的是 NI 公司推出的 LabVIEW 软件,其实该公司还有一款非常优秀的虚拟仪器开发软件 LabWindows/CVI。与 LabVIEW 相比,LabWindows/CVI 主要应用在各种测试、控制、故障分析及信息处理软件的开发中,其更适合中、大型复杂测试软件的开发,是工程技术人员开发建立监测系统、自动测量环境、数据采集系统、过程监测系统的首选工具。这也是本书选中其作为教学软件的原因。

LabWindows/CVI 是一种面向计算机测控领域的虚拟仪器软件开发平台,可以在多种操作系统(Windows 10/Windows 7/Windows XP、Mac OS 和 UNIX)下运行。LabWindows/CVI 为 C 语言程序员提供了一整套集成开发环境(IDE),在此开发环境中可以利用 C 语言及其提供的库函数来实现程序的设计、编辑、编译、链接、调试。使用 LabWindows/CVI 可以完成以下工作,当然不局限于以下工作:

(1) 交互式程序开发。

(2) 具有功能强大的函数库,用来创建数据采集和仪器控制的应用程序。

(3) 充分利用完备的软件工具进行数据采集、分析和显示。

(4) 利用向导开发 IVI 仪器驱动程序和创建 ActiveX 服务器。

(5) 为其他程序开发 C 目标模块、动态链接库(DLL)、C 语言库。

LabWindows/CVI 软件提供了丰富的函数库,利用这些库函数除可实现常规的程序设计外,还可实现更加复杂的数据采集和仪器控制系统的开发,例如:

数据采集库,包括 IVI 库、GPIB/GPIB 488.2 库、NI-DAQmx 库、传统的 NI-DAQ 库、RS-232 库、VISA 库、VXI 库以及 NI-CAN 库。

数据分析库,包括格式化 IO 库、分析库以及可选的高级分析库。

GUI 库,使用 LabWindows/CVI 的用户界面编辑器可以创建并编辑图形用户界面(GUI),而使用 LabWindows/CVI 的用户界面库函数可以在程序中创建并控制 GUI。此外,LabWindows/CVI 为 GUI 面板的设计准备了许多专业控件,如曲线图控件、带状图

控件、表头、旋钮和指示灯等，以适应测控系统软件开发的需求，利用这些控件可以设计出专业的测控程序界面。

网络和进程间通信库，包括动态数据交换（DDE）库、TCP 库、ActiveX 库、Internet 库、DIAdem 连接库、DataSocket 库等。

除此之外，用户可以在 CVI 中使用 ANSI C 库中的全部标准函数。

2. LabWindows/CVI 安装和开发环境

要安装 CVI，只需要根据 CVI 安装包里的“Release Notes. pdf”和“说明. txt”一步步完成安装即可。

安装完毕后，安装程序会在计算机的磁盘中 LabWindows/CVI 的目录安装一系列文件夹，每个文件夹中所涉及的主要内容如表 5-1 所示。

表 5-1　LabWindows/CVI 主要文件夹及其内容

目　录　名	说　　明
\bin	LabWindows/CVI 的库文件
\extlib	外部编译器使用的 CVI 库文件（只在 Windows 95/NT 中使用）
\fonts	字体文件
\include	头文件
\instr	仪器模块
\samples	CVI 开发例程
\sdk	SDK 库文件（只在 Windows 95/NT 中使用）
\toolslib	开发工具包和库文件
\tutorial	使用手册
\vxd	VXD 实例开发模板
\wizard	CVI 开发环境中的向导程序

其中，samples 文件夹中的例程可以使初学者迅速掌握 CVI 编程开发基本步骤，bin 文件夹下的 cvi. chm（也可以在 CVI 开发环境中按 F1 键打开）是学习 CVI 必不可少的参考文档。

安装完毕后，单击桌面上快捷方式图标，即可启动 CVI 软件，进入 LabWindows/CVI 的编程环境，其编程环境有四个主要的界面窗口（见图 5-1）：

- 工程文件编辑窗口（Project Window），简称工程窗口；
- 用户界面编辑窗口（User Interface Editor Window）；
- 源代码文件编辑窗口（Source Window），简称源代码窗口；
- 函数面板窗口（Function Panel）。

其中，工程窗口完成对 *. prj 文件的创建与编辑，用户界面编辑窗口完成对 *. uir 文件的创建与编辑，源代码窗口完成对 *. c 文件的创建与编辑。

因此，选择 LabWindows/CVI 作为检测系统软件设计的主要编程语言，采用面向对象的编程方法，借助其丰富的库函数，可以大大加快软件系统的开发。

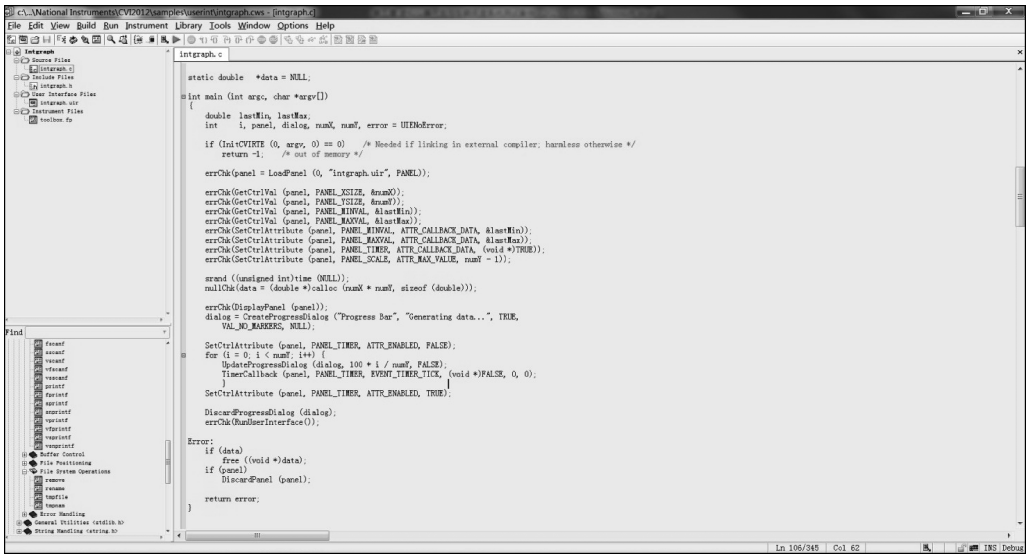


图 5-1 LabWindows/CVI 软件编程环境

例如,在其他软件中,绘制正弦函数图形往往需要编制一大段程序,而借助 CVI 强大的 analysis 库函数,可以轻松地绘制图 5-2 所示的正弦函数波形。事实上,其核心文件中只需要包含 analysis.h 文件,在 main 函数中写下 3 行核心代码,其他由系统自动生成即可。

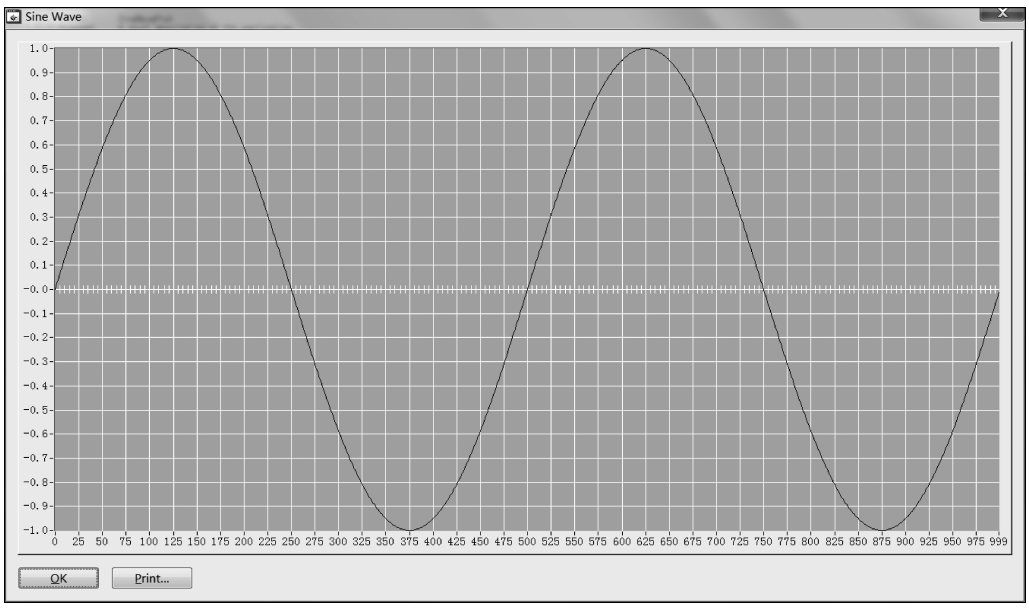


图 5-2 CVI 程序绘制的正弦函数波形

```
#include <analysis.h>
int main (int argc, char *argv[])
{
    double sine[1000];                                    //核心代码
    SinePattern(1000,1,0,2,sine);                         //核心代码
    YGraphPopup("Sine Wave", sine, 1000, VAL_DOUBLE);     //核心代码
    return 0;
}
```

5.1.3 调试简单程序

不管是学习一门新的语言,还是学习一个新的开发工具,第一个程序往往都是HelloWorld。下面就利用CVI软件一步一步来实现HelloWorld程序。

1. 建立/保存工程

运行LabWindows/CVI,初始状态的CVI会自动为用户建好一个新的工作空间Untitled.cws以及新的工程。Untitled.cws文件是CVI工作空间文件(CVI WorkSpace),而.prj(project)是CVI的工程文件。单击菜单File→New→Source(*.c),新建一个C文件,如图5-3所示。

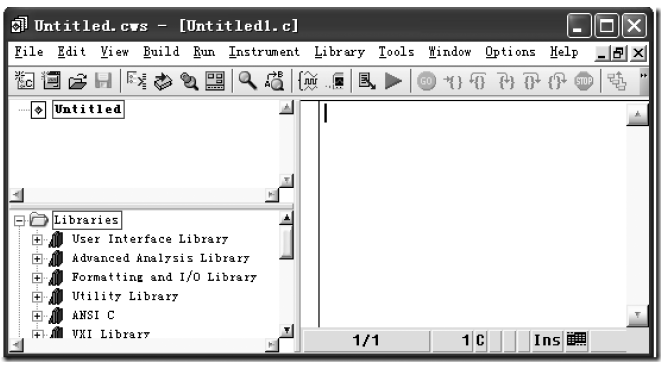


图5-3 打开CVI并新建C文件

2. 输入代码保存代码文件

在新建的C文件中输入下列代码(图5-4),单击菜单File→Save Untitled1.c(或者按Ctrl+S或者单击工具栏中的"保存"按钮),将新建的C文件保存在自己想要保存的位置中。

```
int main (int argc, char *argv[])
{
    printf("Hello,world!\n");
    getchar();
    return 0;
}
```

图5-4 在新建的C文件中输入代码

3. 编译运行

此时若单击菜单Run→Debug Project(或者单击工具栏中的绿色三角形按钮),则CVI会弹出如图5-5

所示提示，说明刚刚保存的C语言文件必须添加到一个工程中才能继续编译过程。此时单击Yes按钮会自动将C文件添加到工程中，若单击Cancel按钮也可以右击Untitled工程之后选择Add File…添加C文件到工程中。

由于只保存了C文件，并未保存工程.prj文件，所以右击Untitled工程之后选择Save，将工程文件保存，如图5-6所示。

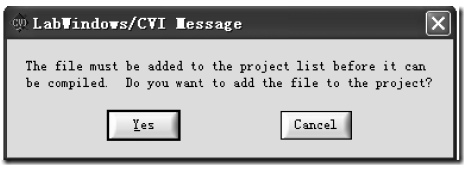


图5-5 CVI提示信息框

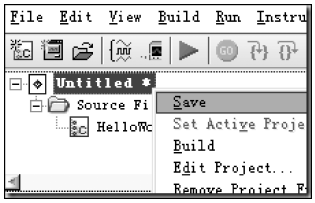


图5-6 保存工程弹出信息框

此时若再单击菜单Run→Debug Project(或者单击工具栏中的绿色三角形按钮)，“HelloWorld!”成功运行，如图5-7所示。

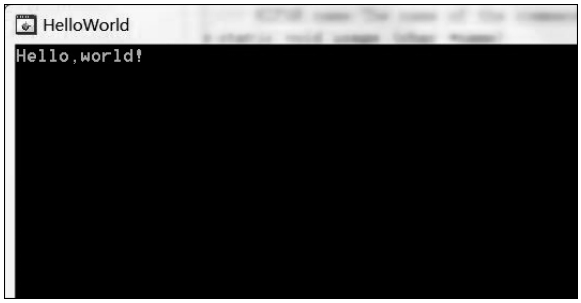


图5-7 程序运行结果

总结起来，完成一个HelloWorld程序只需要执行“建立/保存工程”→“输入代码保存代码文件”→“编译运行”简单的三步。

若感兴趣，读者也可以将以前在VC中写的C语言程序代码复制到CVI中，看看在CVI中是否也能正确地运行。

其实，CVI是支持ANSI(American National Standards Institute，美国国家标准协会)C的，只要是ANSI C的代码，在CVI中一样可以运行。

初次接触CVI可能对CVI的“工作空间”与“工程”并不熟悉。一个工作空间中可能存在一个或多个工程。

CVI每次编译时一般只对“当前”工程进行编译。需要注意的是，当前工程不是指当前打开的文件所在的工程，而是被设置为“Active Project”的工程。设置一个工程为当前工程，可以通过右击“工程→Set Active Project”来完成，被设置为当前工程的工程名会被加粗显示，如图5-8所示。

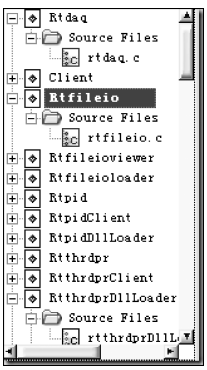


图5-8 设置当前工作工程Rtfileio

CVI 也可以批量编译，即同时对一个工作空间下的多个工程进行编译。批量编译可以通过菜单 Build→Batch Build…来实现。

学习或提升一门编程语言的水平，最快速有效的是阅读大量优质的代码。NI 在 CVI 安装目录的 samples 下提供了大量的参考例程。大家可以将 samples 例程中的.cws 文件拖动到 CVI 中打开.cws 文件，运行并查看官方的代码。

5.1.4 实践与探索

1. 安装 CVI 软件，并实现类似图 5-2 CVI 程序运行结果所示的应用程序，要求产生高斯噪声波形。

2. 通过查看 NI 帮助文档或查找资料，探究 LabWindows/CVI 的各种程序模板及其应用。

5.2 案例 1 数据采集虚拟仪器构建

5.2.1 实践目的

(1) 掌握利用 LabWindows/CVI 软件编程控制 NI 采集卡的基本原理和方法。

(2) 通过案例，学会利用软件控制 PCI-6233 数据采集卡，设计信号采集和分析程序。

5.2.2 虚拟仪器驱动软件

1. 虚拟仪器软件结构

由 5.1 节可知，功能灵活且强大的软件是虚拟仪器系统的核心。根据 VPP(VXI plug&play)系统规范的定义，虚拟仪器系统的软件结构应包含应用程序开发环境、仪器驱动程序、输入/输出(I/O)接口软件三部分，如图 5-9 所示。

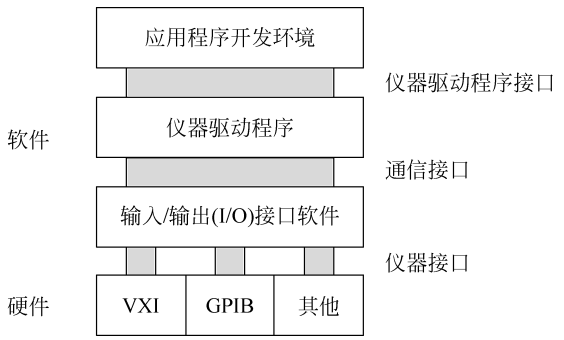


图 5-9 虚拟仪器软件结构

(1) 输入/输出(I/O)接口软件：存在于仪器与仪器驱动程序之间，是一个完成对仪器内部寄存器单元进行直接存取数据操作、对总线背板与器件做测试与控制、并为仪器与仪器驱动程序提供信息传递的底层软件层，是实现开放的、统一的虚拟仪器系统的基础与核心。在 VPP 系统规范中，详细规定了虚拟仪器系统输入/输出(I/O)接口软件的特点、组成、内部结构与实现规范，并将符合 VPP 规范的虚拟仪器系统输入/输出(I/O)接口软件定义为 VISA(Virtual Instrument Software Architecture)软件。

(2) 仪器驱动程序：每个仪器模块均有自己的仪器驱动程序。仪器驱动程序的实质是为用户提供用于仪器操作的较抽象的操作函数集。对于应用程序来说，它对仪器的操作是通过仪器驱动程序实现的；仪器驱动程序对于仪器的操作与管理，又是通过输入/输出(I/O)软件所提供的统一基础与格式的函数库(VISA 库)的调用实现的。对于应用程序设计人员来说，一旦有了仪器驱动程序，在不是十分了解仪器内部操作过程的情况下，也可以进行虚拟仪器系统的设计工作。仪器驱动程序是连接上层应用软件与底层输入/输出(I/O)软件的纽带和桥梁。过去，仪器供应厂家在提供仪器模块的同时提供的仪器驱动程序的形式，都类似于一个“黑匣子”，用户只能见到仪器驱动程序的引出函数原型，而将源程序“神秘”地隐藏起来。用户即使发现供应厂家提供的仪器驱动程序不能完全符合使用要求，也无法对其作出修改，仪器的功能由供应厂家而不是由用户本身来规定。而 VPP 规范明确地定义了仪器驱动程序的组成结构与实现，明确规定仪器生产厂家在提供仪器模块的同时，必须提供仪器驱动程序的源程序文件与动态链接库(DLL)文件，并且由于仪器驱动程序的编写是在 VISA 软件的共同基础上，因此仪器驱动程序之间有很大的互参考性，仪器驱动程序源程序也容易理解，从而提供给用户修改仪器驱动程序的权利和能力，使用户可以对仪器功能进行扩展，将仪器使用的主动权真正交给了用户。

(3) 应用程序开发环境：常见的应用程序开发环境，包括 LabVIEW、LabWindows/CVI 和 Measurement Studio(Visual Studio 编程语言)等软件。具体开发环境的选择，可因开发人员的喜好不同而不同，但最终都必须提供给用户界面友好、功能强大的应用程序。

从图 5-9 可知，仪器驱动程序和输入/输出(I/O)接口软件对于虚拟仪器的开发异常重要，它的质量直接决定了虚拟仪器软件的质量。

2. NI-DAQmx 驱动软件

本节案例控制对象为 NI 公司的 PCI-6233 采集卡。作为 NI 测量设备，均附带 NI-DAQmx 驱动软件，下面简要介绍该软件。NI-DAQmx 驱动软件是一个用途广泛的库，可从 LabVIEW 或 LabWindows/CVI 中调用其库函数，对 NI 设备进行编程控制。NI 的测量设备包括各种 DAQ 设备，如 E 系列多功能 I/O(MIO)设备、SCXI 信号调理模块、开关模块等。驱动软件有一个应用程序编程接口(API)，包括了创建某特定设备的相关测量应用所需的 VI、函数、类及属性。需要注意的是，NI 公司的 DAQ 函数库是伴随着 NI MAX 软件一起安装的。也就是说，要利用 DAQ 函数库，一定要先安装 NI MAX 软件。

下面介绍 NI-DAQmx 驱动软件中的几个重要概念。

1）虚拟通道和任务

虚拟通道，有时简称为通道，是将物理通道和通道相关信息（范围、接线端配置、自定义换算等格式化数据信息）组合在一起的软件实体。任务是具有定时、触发等属性的一个或多个虚拟通道。

与虚拟通道相对，物理通道是测量和发生模拟信号或数字信号的接线端或引脚。信号物理通道可包括一个以上接线端，例如，差分模拟输入通道或 8 线数字端口。设备上的每个物理通道都有唯一的符合 NI-DAQmx 实体通道命名规范的名称（例如，SC1Mod4/ai0、Dev2/ao5、Dev6/ctr3）。

虚拟通道是将物理通道和通道相关信息（范围、接线端配置、自定义换算等格式化数据信息）组合在一起的软件实体。使用“DAQmx 创建虚拟通道”函数/VI 或 DAQ 助手创建虚拟通道。

通过“DAQmx 创建虚拟通道”函数/VI 创建的虚拟通道是局部虚拟通道，只能在任务中使用。使用该函数，可选择虚拟通道的名称。该名称将用于 NI-DAQmx 的其他位置，用于指代该虚拟通道。

如使用 DAQ 助手创建虚拟通道，可在其他任务中使用这些虚拟通道，并在任务之外引用虚拟通道。因为这些虚拟通道是全局虚拟通道，可用于多个任务。可使用 NI-DAQmx API 或 DAQ 助手选择全局虚拟通道，并将其加入任务。如将一条全局虚拟通道添加若干个任务，然后使用 DAQ 助手修改这个全局虚拟通道，改动将应用于所有使用该全局虚拟通道的任务。全局虚拟通道的改动生效前必须先保存改动。

2）虚拟通道的类型

根据信号的类型（模拟、数字、计数器）和方向（输入、输出），可创建不同类型的虚拟通道。虚拟通道可以是全局虚拟通道或局部虚拟通道。

模拟输入通道——模拟输入通道使用各种传感器测量不同的物理量。创建的通道类型取决于传感器以及测量物理量的类型。例如，可创建热电偶测量温度的通道、测量电流电压的通道、测量带激励电压的通道等。

模拟输出通道——NI-DAQmx 支持两种类型的信号，即电流信号和电压信号。如设备测量的是其他信号，可将测得的信号进行转换得到电压或电流信号。

数字输入/输出通道——对于数字通道，可创建基于线和基于端口的数字通道。基于线的通道可包含设备一个或多个端口的一条或多条数字线。读取这些基于数字线的通道不会影响硬件上的其他数字线。

3）物理通道

物理通道的名称由设备标识符、斜杠（/）和通道标识符组成。例如，物理通道是 Dev1/ai1，设备标识符是 Dev1，通道标识符是 ai1。MAX 根据设备在系统中安装顺序的前后为设备分配标识符，如 Dev0、Dev1 等。当然也可在 MAX 中修改设备分配设备标识符。

对于模拟 I/O 和计数器 I/O，通道标识符由通道类型（模拟输入 ai、模拟输出 ao、计

数器 ctr)和通道编号组成,如 ai1、ctr0。对于数字 I/O,通道标识符指定了一个端口,包括了端口中的所有线,如 port0。通道标识符可指定数字端口中的线,例如,port0/line1 指端口 0 的数字线 1。

如要指定一个物理通道的范围,在两个通道编号或物理通道名称的编号之间使用冒号分隔。例如,Dev1/ai0:4、Dev1/ai0:Dev1/ai4 均表示设备 Dev1 的模拟输入通道 0～4。

对于数字 I/O,在两个端口编号之间用冒号分隔,指定一个端口范围。例如,Dev1/port0:1 表示 Dev1 的数字端口 0。也可指定一个数字线的范围,例如,Dev1/port0/line0:4 表示设备 Dev1 端口 0 的数字线 0～4,Dev1/line0:31 表示设备 Dev1 的数字线 0～31。

4) 任务、虚拟通道、物理通道相互间的关系

对于虚拟通道,根据其建立是否从属于一个任务而分为全局虚拟通道和局部虚拟通道。在一个任务中建立的虚拟通道称为局部虚拟通道;在一个任务以上建立的虚拟通道称为全局虚拟通道。全局虚拟通道可用于任何应用程序,或添加到多个不同的任务中。一旦全局虚拟通道发生改变,则所有引用该全局虚拟通道的任务都将受到影响。多数情况下,使用局部虚拟通道更简便。物理通道是测量和发生模拟信号或数字信号的硬件设备的接线端或管脚。三者间的关系见图 5-10。

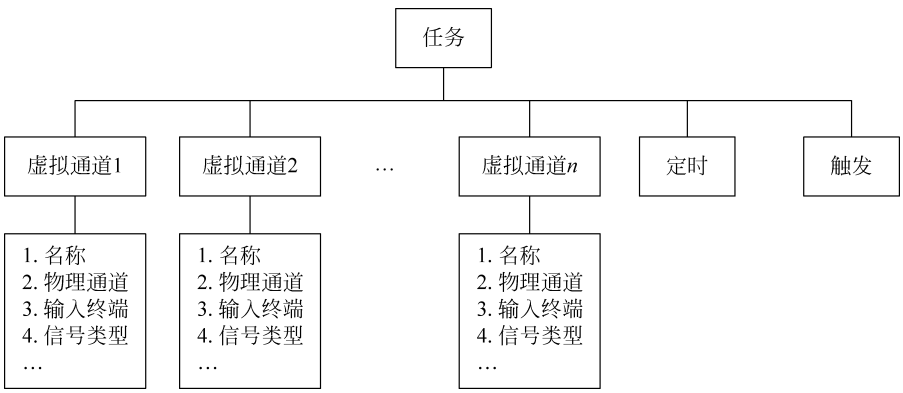


图 5-10 任务、虚拟通道、物理通道关系图

5) 虚拟通道的配置

可以通过 3 种方式配置 1 个数据采集编程的虚拟通道:利用 DAQ 助手、DAQmx 函数、DAQmx VI 函数。DAQ 助手可以从 NI MAX 中调用,如第 4 章 4.8 节实验;DAQmx 函数只能从 LabWindows/CVI 或者 VC++等编程环境中调用;DAQmx VI 函数则可以通过 LabVIEW 调用。这种关系如图 5-11 所示。

3. DAQmx 函数库

当 NI MAX 正确地完成安装后,在其安装目录中,如 C:\Program Files (x86)\National Instruments\Shared\CVI\toolslib\custctrl,就会出现 daqmxioctrl.fp 文件。当在 CVI 开发界面中编辑 DAQ 相关工程时,在 Libraries 会出现 NI-DAQmx Library,展

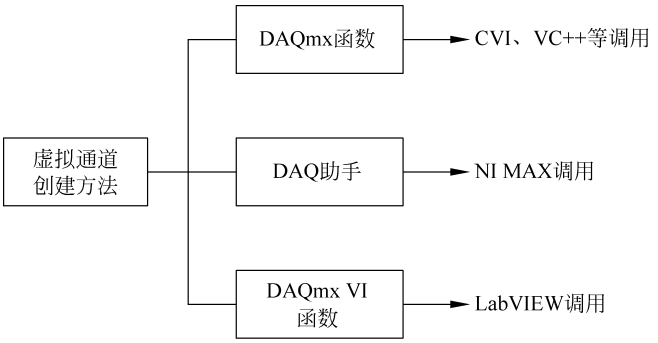


图 5-11　虚拟通道创建的方法

开后，可以看到包括任务配置、通道配置、时基配置、触发设置、读、写、错误处理等函数，可以根据需要选择相应的函数。

4. NI PCI-6233 数据采集卡的控制方法

本节以 CVI 控制 NI PCI-6233 采集卡为例说明，其主要步骤如下：

(1) 创建一个任务。

(2) 生成一个模拟输入电压通道。

(3) 设置采样速率，定义采集模式，选择连续采集或单次采集。

(4) 调用采集函数以启动采集。

(5) 在 EveryNCallback 回调函数中读数据直到停止按钮按下或者错误发生。

(6) 调用 Clear Task 函数清除任务。

(7) 如果发生错误则显示错误。

5.2.3　数据采集程序分析

利用 CVI 打开本节的例子程序“ContAcq-IntClk”，分析其 DAQ 控制代码。此程序较好地诠释了如何控制 NI PCI-6233 采集卡进行连续 A/D 采集、数据显示的过程，具有很高的参考价值，下面对该程序的各个函数中相关的代码进行分析。

1. main 函数

```
int main(int argc, char *argv[])
{
    …
    //设置采集卡的物理通道
    NIDAQmx_NewPhysChanAICtrl(panelHandle,PANEL_CHANNEL,1);
    …
    //采集数据工作结束后将所建立的采集任务清除
    if( gTaskHandle )
```

```
        DAQmxClearTask(gTaskHandle);
    …
}
```

2. Start 按钮回调函数

```
    /*********************************************/
    // DAQmx Configure Code
    /*********************************************/
    //创建数据采集任务
DAQmxErrChk (DAQmxCreateTask("",&gTaskHandle));
//设置模拟电压输入通道,单端有参考地输入,输入范围由变量 min 和 max 确定,输入信号单位为伏特
DAQmxErrChk(DAQmxCreateAIVoltageChan(gTaskHandle,chan,"",DAQmx_Val_RSE,min,max,DAQmx_
Val_Volts,NULL));
//设置模拟输入通道的采样时钟为板上时钟,电压采样率根据 rate 变量设置,上升沿触发采样,
//连续采样模式,输入范围由变量 min 和 max 确定,采样点数由变量 sampsPerChan 确定
DAQmxErrChk(DAQmxCfgSampClkTiming(gTaskHandle,"",rate,DAQmx_Val_Rising,DAQmx_Val_
ContSamps,sampsPerChan));
DAQmxErrChk(DAQmxGetTaskAttribute(gTaskHandle,DAQmx_Task_NumChans,&gNumChannels));
//根据采样通道数和采样点数之积申请存储数据内存,如果内存不足,报错
    if( (gData=(float64*)malloc(sampsPerChan*gNumChannels*sizeof(float64)))==NULL ) {
            MessagePopup("Error","Not enough memory");
            goto Error;
        }
//每完成一次采集,调用 EveryNCallback 回调函数一次
DAQmxErrChk(DAQmxRegisterEveryNSamplesEvent(gTaskHandle,DAQmx_Val_Acquired_Into_Buffer,
sampsPerChan,0,EveryNCallback,NULL));
//停止采集后,调用 DoneCallback 回调函数一次
DAQmxErrChk (DAQmxRegisterDoneEvent(gTaskHandle,0,DoneCallback,NULL));
```

3. EveryNCallback 回调函数

```
/*********************************************/
// DAQmx Read Code
/*********************************************/
//从采集卡的 FIFO 中读出数据,nSamples 变量设置读出数,等待读出时间为 10s,数据按照采样数
//来编组,函数的输出为每通道采样数,读出数据数组
DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,nSamples,10.0,DAQmx_Val_GroupByScanNumber,
gData,nSamples*gNumChannels,&numRead,NULL));
```

4. DoneCallback 回调函数

```
//释放所申请的内存
if( gData ) {
        free(gData);
        gData = NULL;
```

```
    }
//将采集任务句柄清零,停止数据采集任务
    gTaskHandle = 0;
```

5. 错误处理相关代码

在程序运行过程中,由于种种原因,可能会出现各种错误,因此,相关的错误处理代码必不可少,否则就会出现程序异常中断,甚至导致系统崩溃。

```
Error:
    if( DAQmxFailed(error) ) {
        //如果发生错误,将错误内容写入 errBuff 数组中
DAQmxGetExtendedErrorInfo(errBuff,2048);
        /*********************************************/
        /* DAQmx Stop Code
        /*********************************************/
        //停止 DAQ 采集任务
DAQmxStopTask(taskHandle);
        //清除 DAQ 采集任务
DAQmxClearTask(taskHandle);
        //将采集任务句柄清零,停止数据采集任务
gTaskHandle = 0;
//释放所申请的内存
        if( gData ) {
            free(gData);
            gData = NULL;
        }
        MessagePopup("DAQmx Error",errBuff);
        SetCtrlAttribute(panelHandle,PANEL_START,ATTR_DIMMED,0);
```

5.2.4 数据采集实践

1. 所用设备

序　　号	名　　称	型　　号	数　　量
1	西门子工控计算机	IPC3000 SMART	1
2	数据采集卡	NI PCI-6233	1
3	数据采集卡附件	NI CB-37F-HVD	1
4	信号发生器	DG1022U	1
5	数字示波器	DS1072U	1

2. 单通道采集信号发生器输出信号

启动 DG1022U 信号发生器,设置通道 1 为输出通道,信号类型为正弦波,频率为

100Hz,峰-峰值为 5V,偏移量为 0V。将信号发生器输出正弦信号接入模拟输入通道 1。启动采集程序,单击 Start 按钮,得到图 5-12 所示不断更新的波形。根据波形可知,所采集数据正确。至此,改变输出信号设置、采样频率、采用通道,检查所采集波形是否正确。单击 Stop 按钮,数据采集工作停止,波形停止更新。

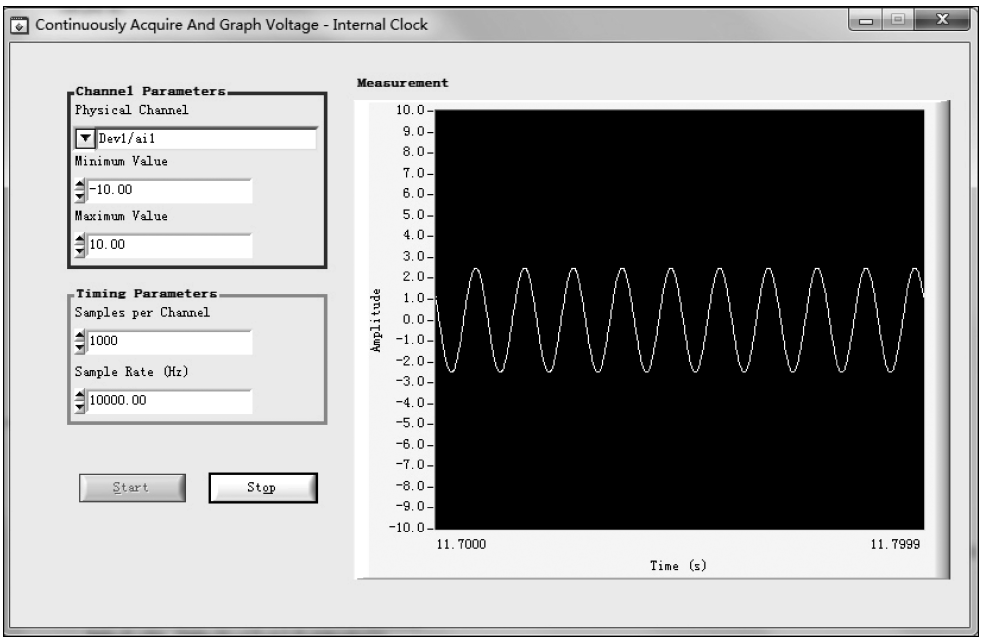


图 5-12　采集 100Hz 正弦信号的波形

3. 多通道采集信号发生器输出信号

利用该程序,也可以实现多通道采集。单击 Physical Channel 控件,再单击 Browse…,进入图 5-13 所示界面,选择 Dev1/ai1 和 Dev1/ai2 通道,单击 OK 按钮后,Physical Channel 控件的内容显示 Dev1/ai1:2。

利用 DG1022U 信号发生器输出二路信号,其中通道 1 输出信号依然为频率为 100Hz、峰-峰值为 5V、偏移量为 0V 的正弦波;通道 2 输出信号设定为频率为 100Hz、峰-峰值为 3V、偏移量为 0V 的三角波。启动采集程序,单击 Start 按钮,得到如图 5-14 所示的采集界面,其中采集所得的正弦波、三角波与信号发生器的设置一致。

思考题

1. 利用本节所提供的数据采集程序,通过编程分别改变通道输入电压范围、采集频率、采集点数,显示所得的信号波形。

2. 编程实现单次数据采集数据,采集点数为 1000,采集频率为 20kS/s,并将所采集数据存入指定的数据文件中。

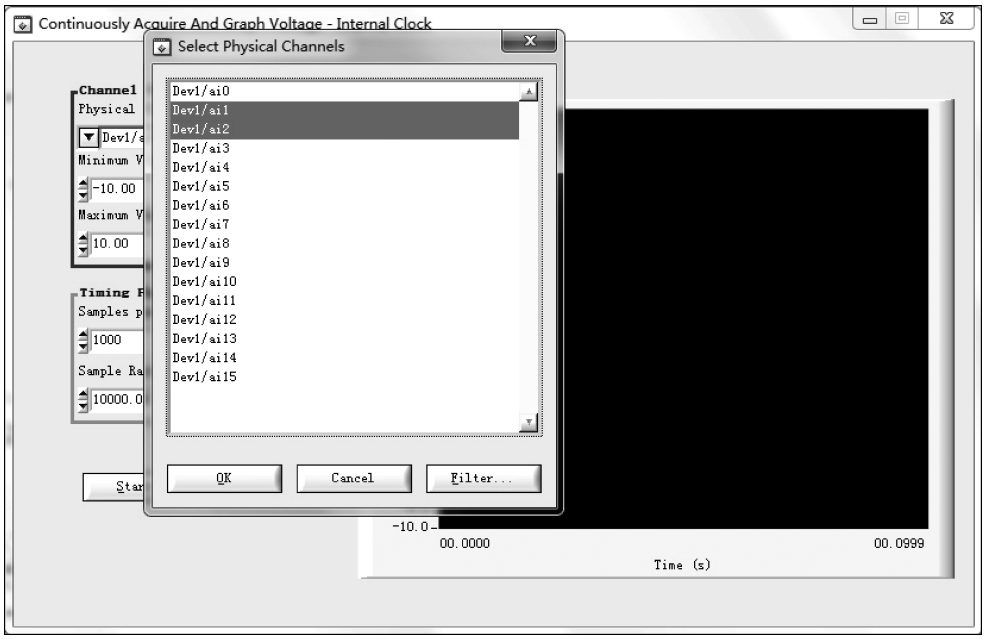


图 5-13　采集通道选择设置界面

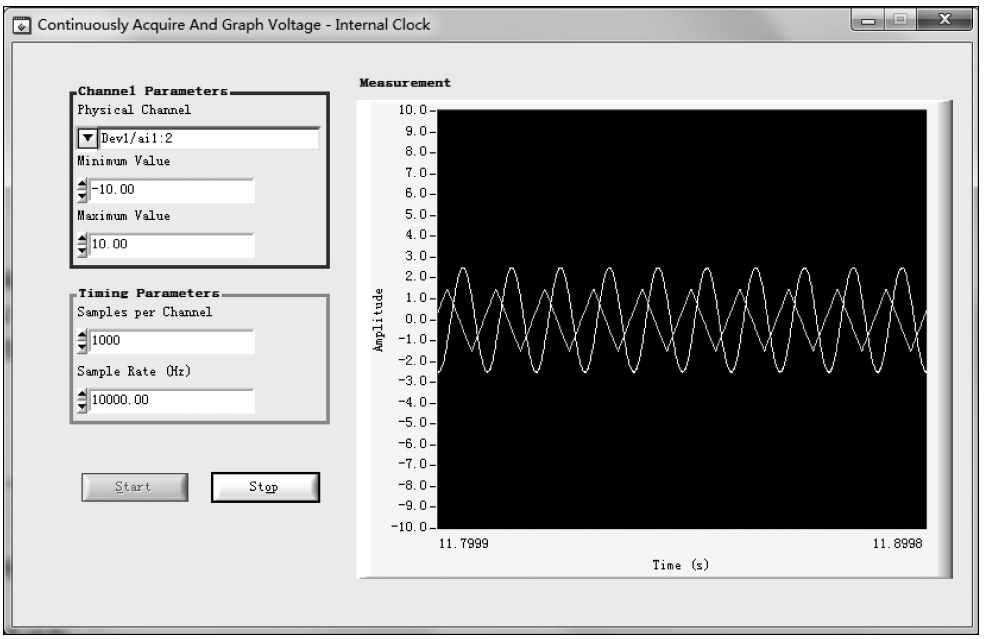


图 5-14　二路信号同步采集波形

5.3 案例 2 网络化虚拟仪器构建

5.3.1 实践目的

(1) 通过案例学习,掌握利用 CVI 软件编程实现基于以太网的虚拟仪器构建的基本原理。

(2) 针对给定的局域网和数据采集卡,能利用 CVI 软件设计远程信号采集和分析程序。

5.3.2 基于以太网的虚拟仪器构建

与传统的检测仪器相比,基于以太网的虚拟仪器检测平台性价比高。它一方面能够同时对多个参数进行实时检测;另一方面,由于信号传送和数据处理都是靠软件来实现,大大降低了环境干扰和系统误差的影响。此外,用户还可以根据需要随时调整软件以调整仪器的功能,从而大大缩短了仪器在改变测量对象时的更新周期。基于以太网的虚拟仪器检测平台具有良好的人机界面,测量结果通过计算机屏幕上的面板来显示,综合各类仪器的特色,突破了传统仪器及仪表在测量数据以及处理数据等方面的限制,可以方便地进行维护和功能扩展,并且很容易实现升级,提高检测仪器的机动性及通用性,从而可以显著提高设备的检测技术水平,避免低层次的重复开发,减少浪费。

1. 基于 C/S 模式的虚拟仪器

本节拟构建基于 C/S 模式的网络化虚拟仪器(模型见图 5-15),其流程为:

(1) 服务器初始化,启动数据采集和服务器进程,进入等待的循环;此时客户机就可以与服务器建立连接。

(2) 连接建立以后,客户机和服务器之间就可进行数据交互与传送;直到客户机关闭连接,服务器才关闭与客户机间的连接。一个服务器可同时为多个客户机提供服务,同样,单个客户机也可以同时接受不同服务器提供的服务。

(3) 客户机将接收到的数据保存到数据库。

(4) 客户端工控计算机对接收的数据进行分析,并生成报表。

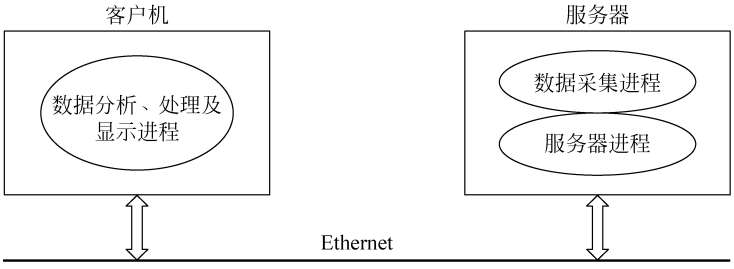


图 5-15 基于 C/S 模式的网络化虚拟仪器模型图

现场端和远程端均运行 DataSocket 服务器程序，由 DataSocket 服务器完成底层的数据通信任务。对现场端而言，发送数据程序(Write)负责将测试数据发送到本地的 DataSocket 服务器，接收数据程序(Read)通过网络从远程端的 DataSocket 服务器读取控制信号。对远程端而言，发送数据程序负责将控制信号发送到本地的 DataSocket 服务器，接收数据程序通过网络从现场端的 DataSocket 服务器读取测试数据。测试数据和控制信号均采用 DataSocket 技术中的 DSTP 传输，以提高数据传输的效率。DSTP 是一种应用于远程数据传输的专用网络协议，数据发送前的打包和接收端的解包比较简单，额外的冗余数据也少，此协议比在程序中直接利用 TCP/IP 来传递数据具有更好的实时性。

基于 DataSocket 技术的测控系统基本结构如图 5-16 所示。

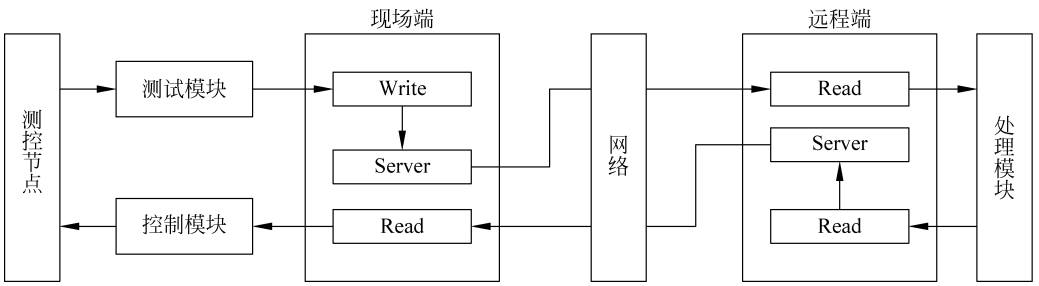


图 5-16　基于 DataSocket 技术的测控系统基本结构

2. DataSocket 技术

1) 简介

DataSocket 是一种基于 TCP/IP 的网络新技术，支持本地文件 I/O 操作、FTP 和 HTTP 文件传输、实时数据共享，并提供统一的 API 编程接口，具有方便使用、高效编程、不需了解底层操作过程等优点，适合于远程数据采集、监控和数据共享等应用程序的开发。从结构上看，DataSocket 包括 DataSocketAPI 和 DataSocketServer 两部分。DataSocketAPI 提供了简单的应用接口，作为客户，可以在多种编程环境下与多种数据类型通信。DataSocketAPI 包含四个基本动作：Open、Read、Write 和 Close。除了从 DataSocketServer 上获取数据外，DataSocket 还可以获得 HTTP Server、FTP Server 和 OPC Server 的数据。DataSocketServer 是一个独立运行的程序，是提供数据交换的场所，作为服务器，负责存储数据源发布的数据，然后提供给请求的计算机。利用 DataSocket 接口开发通信程序，通常应用的是面向连接的 DataSocket 系统调用，该调用首先在客户机和服务器间创建一个连接并建立一条通信链路，以后的网络通信操作完全在这一对进程之间进行，通信完毕后关闭此连接过程。

2) DataSocket 技术的应用

使用 DataSocket 进行网络通信程序的流程为：

(1) 调用 DS_Open 创建 DataSocket 对象，使用 URL 进行数据源定位的连接。使用不同 URL(HTTP、FTP、DSTP 等)，该函数创建相应的数据对象及属性对象，确定读取

数据方式，当数据、数据属性、连接状态发生改变时回调函数，连接成功，返回句柄。

（2）调用 DS_GetDataValue()读取数据。不同的读取方式，读回的数据的时效性有很大的差别。读方式为 ReadAutoUpdate，客户端 DataSocket 对象中所包含的数据是数据源。对象数据、属性一旦发生改变，客户端 DataSocket 对象包含的数据是在程序中调用 DS_Update 后的数据。

（3）调用 DS_DiscardObjHandle()断开连接。

图 5-17 是 DataSocket 程序流程图。

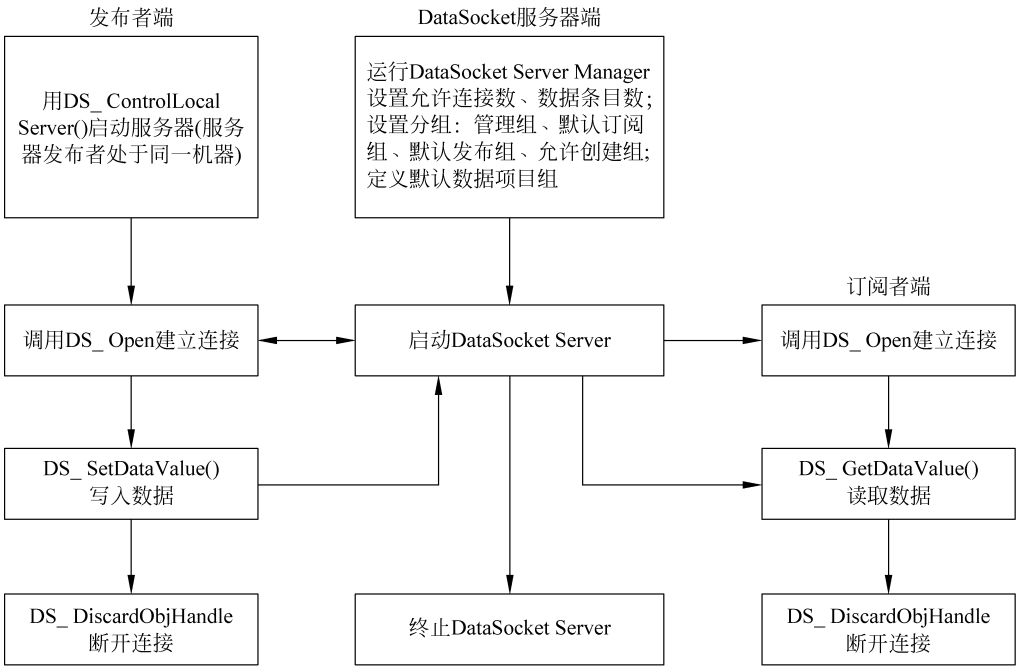


图 5-17　DataSocket 程序流程图

DataSocket Server 是免编程的，只需通过 DataSocket Server Management 的配置（图 5-18），就能完成安全管理、权限管理、进程管理等。首先设置允许连接数和允许创建的数据项目数，系统允许的最大值是 1000 个，然后设置分组，包括管理组、默认订阅组、默认发布组、允许创建组。这样，DataSocket 服务器可由一个或多个主机管理，一个数据项目可对应多个发布者和订阅者。图 5-19 是 DataSocket 服务器工作界面。

5.3.3　数据发送与接收程序设计

1. DataSocket 数据发送程序

第一步：设计程序界面（图 5-20），生成程序框架。

第二步：设计主程序，启动本地 DataSocket 服务器程序。

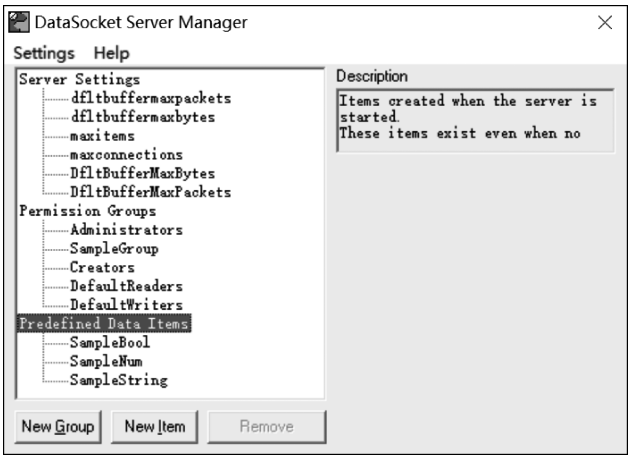


图 5-18　DataSocket Server Manager 界面示意图

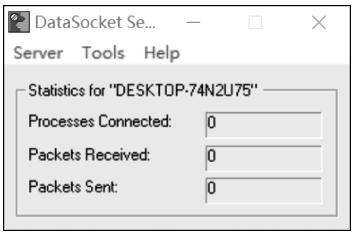


图 5-19　DataSocket Server 界面示意图

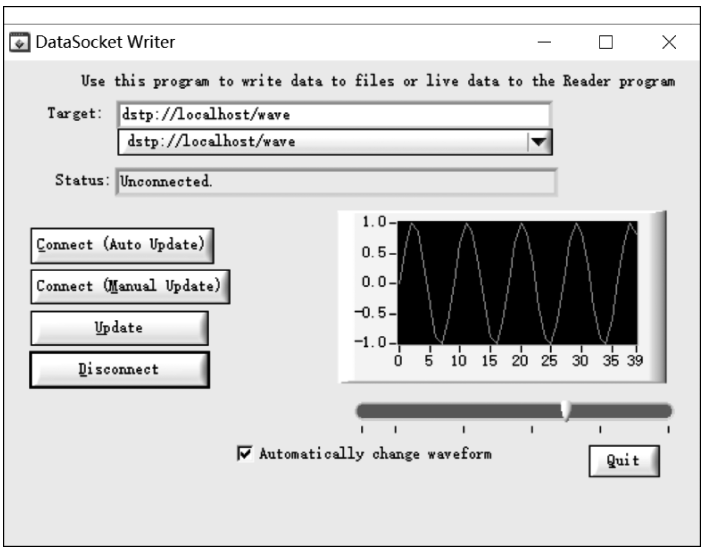


图 5-20　数据发送程序界面

```
int main (int argc, char * argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
    if ((panelHandle = LoadPanel (0, "Writer.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    DS_ControlLocalServer (DSConst_ServerLaunch);
RunUserInterface ();
DS_ControlLocalServer (DSConst_ServerClose);
    DiscardPanel (panelHandle);
    return 0;
}
```

第三步：创建 DataSocket 对象并连接到数据源。

```
int CVICALLBACK OnConnectManual (int panel, int control, int event,
        void * callbackData, int eventData1, int eventData2)
{
    char URL[500];
    BOOL bValue;
    switch (event)
{
        case EVENT_COMMIT:
SetCtrlAttribute (panelHandle, PANEL_TIMER, ATTR_ENABLED, FALSE);
            if (dsHandle)
{
                DS_DiscardObjHandle(dsHandle);
                dsHandle = 0;
            }
GetCtrlVal (panelHandle, PANEL_SOURCE, URL); DS_Open (URL, DSConst_Write, DSCallback,
NULL, &dsHandle);
        OnSlideChanged (panelHandle, PANEL_NUMERICSLIDE, EVENT_COMMIT, NULL, 0, 0);
        DS_Update (dsHandle);
        GetCtrlVal (panelHandle, PANEL_CHECKBOX, &bValue);
SetCtrlAttribute (panelHandle, PANEL_TIMER, ATTR_ENABLED, bValue);
            break;
}
    return 0;
}
```

第四步：设计写 DataSocket 对象数据程序。

```
int CVICALLBACK OnSlideChanged (int panel, int control, int event,
        void * callbackData, int eventData1, int eventData2)
{
    double value;
    int i;
    HRESULT hr = S_OK;
    DummyType dummy;
```

```
    char str[2];
    str[1] = 0;
    switch (event)
{
        case EVENT_COMMIT:
        GetCtrlVal (panelHandle, PANEL_NUMERICSLIDE, &value);
            for (i = 0; i < 40; i++)
{
                dummy.waveform[i] = sin(i * value);
            }
        GraphData(dummy.waveform, 40);
        dummy.item1 = (unsigned char)((int)(value * 100) % 26) + 'A';
        dummy.item2 = value;
if (dsHandle)
{
    hr = DS_SetDataValue (dsHandle, CAVT_FLOAT|CAVT_ARRAY, dummy.waveform, 40, 0);
            }
            break;
    }
    return 0;
}
```

第五步：断开连接，释放 DataSocket 对象。

```
int CVICALLBACK OnDisconnect (int panel, int control, int event,
        void * callbackData, int eventData1, int eventData2)
{
    switch (event)
{
        case EVENT_COMMIT:
            SetCtrlAttribute (panelHandle, PANEL_TIMER, ATTR_ENABLED, FALSE);
            if (dsHandle) {
                DS_DiscardObjHandle(dsHandle);
                dsHandle = 0;
            }
            SetCtrlVal (panelHandle, PANEL_STATUS, "Unconnected.");
            break;}
    return 0;}
```

第六步：程序结束，停止 DataSocket 服务器程序。

2. DataSocket 数据接收程序

第一步：设计程序界面(图 5-21)，生成程序框架。

第二步：设计主程序，创建 DataSocket 对象并连接到数据源。

```
int CVICALLBACK OnConnectAuto (int panel, int control, int event, void * callbackData, int
eventData1, int eventData2)
{
```

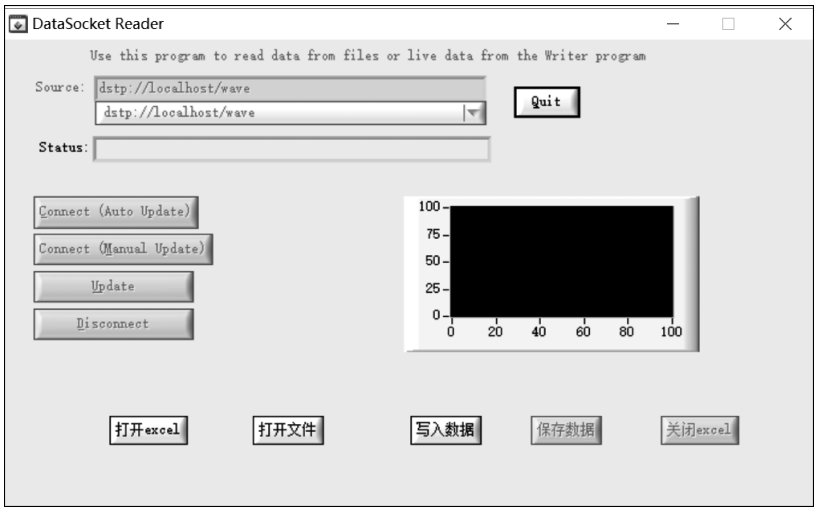


图 5-21　数据接收程序界面

```
HRESULT error;
char URL[500];
switch (event)
{
case EVENT_COMMIT:
/* 如果句柄已存在,则在打开前先关闭 */
if (dsHandle)
{
DS_DiscardObjHandle(dsHandle);
dsHandle = 0;
}
singleX = 0;
ClearGraph();
/* 先获取 URL,再创建 DataSocket 对象,设置为自动刷新模式 */
GetCtrlVal (panelHandle, PANEL_SOURCE, URL);
error = DS_Open (URL, DSConst_ReadAutoUpdate, DSCallback, NULL, &dsHandle);
break;
    }
return 0;
}
```

第三步：设计写 DataSocket 回调函数读取 DataSocket 对象数据。

```
void CVICALLBACK DSCallback (DSHandledsHandle, int event, void * pUserData)
{
    HRESULT hr = S_OK;
    float singleFloat;
    char message[1000];
    unsigned int sz;
    unsigned type;
```

```
    switch (event)
    {
         case DS_EVENT_DATAUPDATED:
            ClearGraph();
            hr = DS_GetDataType (dsHandle, &type, NULL, NULL);
            if (type & CAVT_ARRAY)
            {
            hr = DS_GetDataValue (dsHandle, CAVT_FLOAT|CAVT_ARRAY, fdata, 5000 * sizeof
(float), &sz, NULL);
            }
            else
            {
                hr = DS_GetDataValue (dsHandle, CAVT_FLOAT, &singleFloat, sizeof(float),
NULL, NULL);
                if (singleX >= 5000)singleX = 0;
                fdata[singleX] = singleFloat;
                singleX++;
                sz = singleX;
            }
            if (SUCCEEDED(hr)) GraphData(fdata, sz);
            return;
            break;
        case DS_EVENT_STATUSUPDATED:
            hr = DS_GetLastMessage (dsHandle, message, 1000);
            if (SUCCEEDED(hr))
                SetCtrlVal (panelHandle, PANEL_STATUS, message);
            break;
    }
    return;
}
```

第四步：断开连接，释放 DataSocket 对象。

```
int CVICALLBACK OnDisconnect (int panel, int control, int event,
        void * callbackData, int eventData1, int eventData2)
{
    switch (event)
{
        case EVENT_COMMIT:
            SetCtrlVal (panelHandle, PANEL_STATUS, "Unconnected.");
            if (dsHandle) {
                DS_DiscardObjHandle(dsHandle);
                dsHandle = 0;
            }
            singleX = 0;
            break;
    }
    return 0;
}
```

思考题

(1) 利用本节所提供的例子程序,通过编程实现利用网络远程改变通道输入电压范围、采集频率、采集点数,远程接收所测得数据,并显示其信号波形。

(2) 阅读相关文献,探索利用 CVI 软件中的 TCP 和 UCP 控件,代替 DataSocket 控件实现第(1)题中的任务。

5.4 案例 3 柴油机振动信号采集与分析

5.4.1 实践目的

(1) 掌握柴油发动机振动分析原理和振动信号采集方法。

(2) 通过案例学习,学习缸盖振动信号的角域平均处理方法。

5.4.2 柴油机振动信号分析

1. 概述

柴油机作为大型机械设备的动力源,其运行状态直接影响着设备效能的发挥。通常,大功率柴油机结构复杂,工作环境恶劣,故障率高。据统计,柴油机产生的故障占装备全部故障的 20%以上,是大型机械设备的主要故障源,而且常常因故障不能及时发现而造成事故。常用的柴油机"定期维修"方案对装备使用过程中柴油机运行状态的实时监测能力不足,容易因"维修滞后"而造成潜在故障不能及时发现和排除,从而导致故障恶化。同时,由于该维修方式忽视了设备的个体差异和具体状态,可能会因"维修不足"而导致设备维修不到位,从而造成严重事故;或者因"过剩维修"而造成资源浪费,提高设备维护成本。因此,综合利用设备状态监测与故障诊断手段对柴油机的运行状态进行准确识别,从而实现柴油机的"视情维修",是提高柴油机维护效率的有效途径。

在传统的柴油机状态监测手段中,主要依赖机油压力、机油温度和冷却水温等参数,但这些参数属于缓变参数,其精度和可靠性不足。柴油机振动信号,特别是缸盖振动信号,含有大量柴油机状态信息,易于在线测试与处理,实现对柴油机运行状态的在线监测。

2. 柴油机缸盖振动信号特点

柴油机缸盖系统结构复杂,承受缸内气体压力、气门落座瞬时冲击力、活塞不平衡往复惯性力、曲轴不平衡惯性力以及随机激励等多种激励。气体压力、气门落座冲击力使

缸盖产生相对机身的振动；而不平衡惯性力、沿活塞连杆传递的气体压力则通过机身传递到缸盖上，使机身和缸盖一起振动。

以 F3L912 型柴油机为例，根据配气机构配气定时规律，分析如图 5-22 所示的 F3L912 型柴油机第 1 缸盖表面的振动响应组成，这些瞬态信号分别表示：A 为 1 缸燃烧气体压力冲击响应和喷油器针阀落座冲击响应，B 为 2 缸燃烧气体压力冲击响应，C 为 3 缸燃烧气体压力冲击响应，D 为 1 缸排气门开启冲击响应，E 为 1 缸排气门落座冲击响应，F 为 1 缸进气门落座冲击响应（进气门开启时的振动响应不大，在图中没有标出此时刻）。

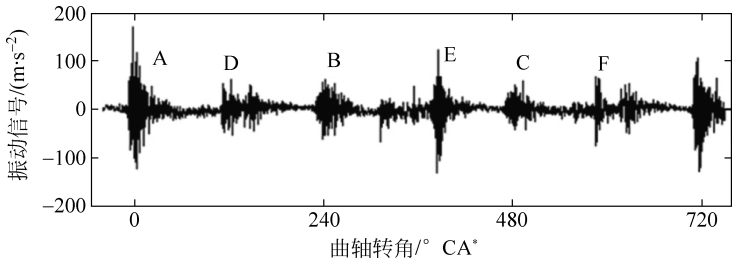


图 5-22　F3L912 型柴油机第 1 缸盖振动响应

从图中看到，对缸盖振动影响较大的是气体压力、气门关闭冲击，而机身振动对缸盖振动影响较小。缸内气体压力、排气门关闭冲击和开启以及进气门关闭所产生的响应各自按照一定规律作用于缸盖，根据缸盖表面测得的振动响应可以推断各个激励源的性质，从而对柴油机各部件技术状态进行判断。

3. 缸盖振动信号预处理

由于柴油机工作环境恶劣，缸盖振动信号不仅包含配气相位角、燃爆时间等有用信息，还包含很多的噪声干扰。如果背景噪声很强，不但信号的时间历程显示不出规律性，而且在频谱图上这些周期分量都可能被淹没在背景噪声中。

时域多段平均分析是从混有白噪声干扰的振动信号中提取周期信号的有效方法，对时域信号，按一个周期为间隔截取信号，然后将得到的每段信号叠加平均，这样可以消除信号中的非周期分量和随机干扰，保留稳定的周期分量，使设备可以在噪声环境下工作，提高分析信噪比。此外，时域同步平均也可作为一种重要的信号预处理过程，其平均结果可再进行频谱分析或作其他处理，如时序分析、小波分析等，均可得到比直接分析处理高的信噪比。

设信号 $x(t)$ 由周期信号 $f(t)$ 和白噪声 $n(t)$ 组成，即

$$x(t)=f(t)+n(t) \tag{5-1}$$

以 $x(t)$ 的周期 M 去截取信号 $x(t)$，共截得 N 段，然后将各段对应点相加，由于白噪

* CA 实际上是 CA 循环，也叫“有限时间热力学循环”。因为 Curzon 和 Ahlborn 较早开始研究，常用二位学者的名字来称此循环为“CA”循环。

声的不相关性，可以得到

$$x(t_i)=Nf(t_i)+\sqrt{N}n(t) \tag{5-2}$$

对 $x(t_i)$求平均，得到输出信号 $y(t_i)$：

$$y(t_i)=\frac{x(t_i)}{N}=f(t_i)+\frac{n(t)}{\sqrt{N}} \tag{5-3}$$

此时，输出的白噪声是原来输入信号 $x(t)$中白噪声的 $1/\sqrt{N}$，因此信噪比将提高 $\sqrt{N}$倍。

由柴油机缸盖振动情况可知，缸盖上的振动信号是有规律的、周期性出现的。如果根据常用的时域多段平均的方法，以时间信号作为每个周期所选择的同步信号来做多段平均，由于瞬时转速的波动性，每个周期所经历的时间是不同的，因此，时域平均在此条件下并不完全适用，可以考虑用角域同步平均技术来解决这个问题。

根据发动机缸盖振动信号产生的机理，气体燃爆及气门开启/关闭所引起的振动信号总是出现在曲轴的同一转角位置，也就是说，对于同一种工况，配气相位角是固定的。无论转速的高低，如果以角度来表示振动发生的位置，就可以消除转速波动的影响。对于四冲程发动机来说，一个工作循环即一个周期是曲轴转两圈，即 720°CA。选取曲轴转角作为判断的同步信号，即以飞轮每个轮齿转过的角度作为同步触发信号，进行角域平均。

5.4.3 缸盖振动信号采集与分析

1. 所用设备

序　号	名　称	型　号	数　量
1	西门子工控计算机	IPC3000 SMART	1
2	数据采集卡	NI PCI-6233	1
3	柴油发动机	F3L912 型	1
4	振动传感器	QSY8611	1
5	光电传感器	GD320	1
6	信号调理装置	自研	1

2. 实验系统搭建

整个测试系统结构如图 5-23 所示。

实验在 F3L912 型三缸内燃机实验台架上进行，F3L912 为 3 缸四冲程柴油发动机，发火顺序为 1 缸—2 缸—3 缸，缸径 100mm，行程 120mm，额定功率和额定转速分别为 30kW 和 3000r/min，飞轮齿数 $Z=129$。其中实验台架如图 5-24 所示。

振动传感器 QSY8611 的安装位置如图 5-25 所示。

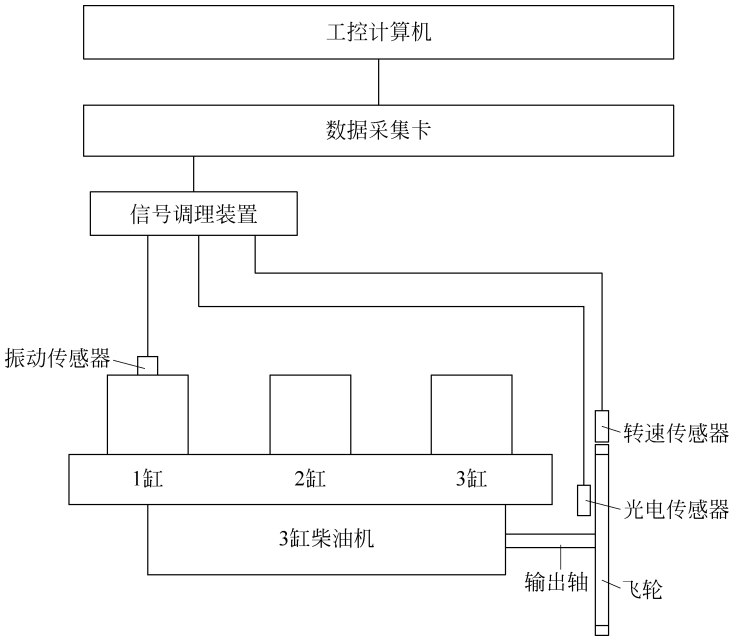


图 5-23　缸盖振动信号测试系统图

图 5-24　F3L912 型柴油机发动机实验台架

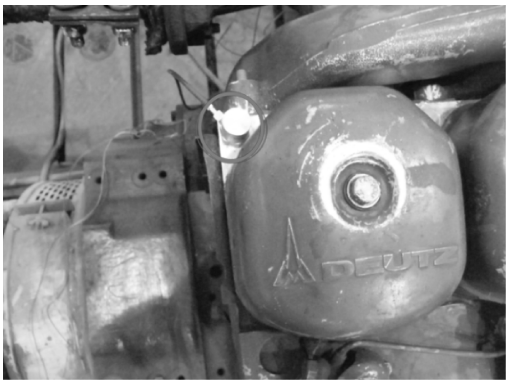


图 5-25　振动加速度传感器安装位置

3.　测试步骤及结果分析

测试过程中柴油机为空载工况，将振动传感器装在第 1 缸的缸盖上，光电传感器用来测取曲轴上止点的位置，磁电式转速传感器安装在机壳上，用来测飞轮的转速，信号的采样频率为 32.768kS/s。

其中以光电传感器测取的第 1 缸压缩上止点信号作为定位发动机曲轴转角的零位置即发动机的一个工作循环的开始，转速传感器测取的发动机飞轮转速信号用来提供飞轮每个轮齿的转角。一个完整工作循环的上止点、振动及转速信号如图 5-26 所示。

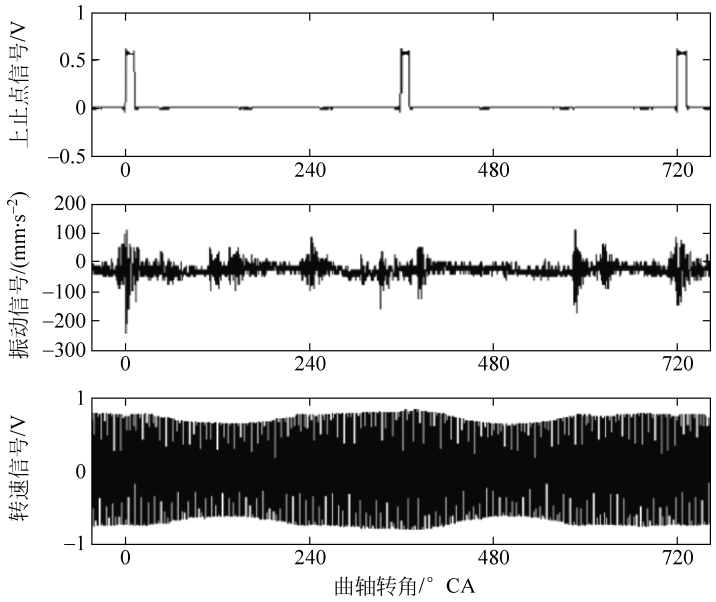


图 5-26　原始测试信号

由磁电式转速传感器直接测得的飞轮位移信号为近似正弦波信号，每个正弦波对应飞轮上的一个轮齿。因此可以用正弦波过零点作为角域平均的同步信号进行采样。由于转速的波动，导致每个齿内采到的数据点不一样多，可以采用插值的方法对采样数据进行插值处理，使每个齿内的采样点数相同，然后进行角域平均。为了更好地反映振动特性，可以滤去与振动无关的低频趋势项。图 5-27 为进行了 7 次角域平均之后得到的振动信号。

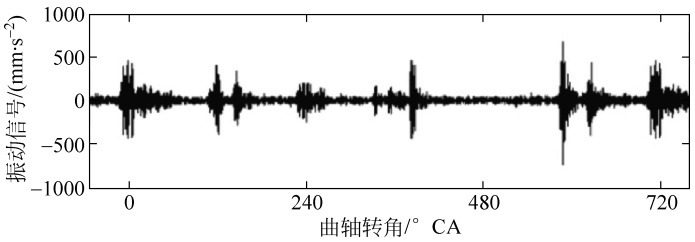


图 5-27　角域同步平均后的振动响应波形

由此可见，对于工作环境恶劣的发动机来说，由于其缸盖上的振动信号中有很多的噪声干扰，同时缸盖振动信号又是有规律的、周期性出现的，为了消除信号中的非周期分量和随机干扰，保留稳定的周期分量，提高分析信噪比，采用角域同步平均技术可对其进行有效预处理。与时域平均相比，角域同步平均技术以飞轮每个轮齿转过的角度作为同步触发信号，克服了转速波动导致的一个周期经历的时间不等的问题。

思考题

1. 独立构建柴油机振动信号测试系统，编写采集程序，同步采集振动、转速与光电传感器输出信号，显示其波形。

2. 利用 MATLAB 软件进行编程实现角域平均算法，并对所采集到的信号进行角域平均处理。

3. 阅读相关文献，利用小波变换方法处理所采集的信号。

5.5 案例 4 瞬时转速信号采集与分析

5.5.1 实践目的

(1) 掌握柴油发动机瞬时转速信号测试原理和检测方法。

(2) 通过案例学习瞬时转速信号的域波形分析法和阶次谱分析法。

5.5.2 柴油机瞬时转速信号分析原理

1. 概述

在柴油机工作过程中，当柴油机进入压缩冲程时，在压缩气体阻扭矩的作用下，曲轴的旋转速度变慢；当柴油机的某一气缸着火燃烧进入膨胀做功冲程时，气缸膨胀产生的扭矩大于曲轴的阻扭矩，曲轴的旋转速度加快。在柴油机的循环工作中，各气缸依次进入压缩冲程和膨胀冲程，从而使曲轴的转速在某一平均转速附近上下波动，此即为曲轴转速的波动特性。转速的波动波形包含着丰富的气缸压力和输出扭矩的信息，利用转速波动波形可以对柴油机进行状态监测和故障诊断。对 N 缸柴油机来说，在一个工作循环内其转速有 N 次波动，当各缸工作情况完全相同时，瞬时转速曲线的 N 个波形应完全一致。而实际上，各缸工作情况不可能完全相同，因而，实测的瞬时转速曲线上各个波形必然出现差异。

2. 基于瞬时转速分析的发动机工况检测机理

对于多缸发动机，有扭矩平衡方程

$$J\ddot{\theta} = T_e(\theta) = T_p(\theta) - T_r(\theta) - T_L \tag{5-4}$$

式中，假设曲轴飞轮系统是刚性轴；J——整个轴系旋转运动部分有效集总转动惯量(包括自由端、飞轮、负载机构和所有各缸曲柄连杆机构旋转运动部分转动惯量)；θ——曲轴转角；$\ddot{\theta}$——曲轴转角加速度；$T_e(\theta)$——整个轴系旋转惯性扭矩；T_L——负载力矩，视作常数；$T_p(\theta)$——气体力扭矩；$T_r(\theta)$——往复输出扭矩。

$$T_p(\theta)=A_p r\sum_{k=1}^{N}[f_p^{(k)}(\theta)f(\theta-\phi_k)] \tag{5-5}$$

$$T_r(\theta)=m^2r^2\sum_{k=1}^{N}\{\ddot{\theta}f(\theta-\phi_k)+\theta^2 g(\theta-\phi_k)f(\theta-\phi_k)\} \tag{5-6}$$

式中，ϕ_k——第 k 缸相对于第 1 缸发火相位；N——气缸数；r——曲柄半径；f_p——气缸压力；A_p——活塞面积；m——整个机构中，做往复运动和旋转运动的集中换算质量。

对四冲程内燃机有

$$\phi_k=\frac{4\pi}{N}(k-1) \tag{5-7}$$

$$f(\theta)=\sin\theta+\frac{\lambda\sin2\theta}{2\sqrt{1-\lambda^2\sin^2\theta}} \tag{5-8}$$

$$g(\theta)=\cos\theta+\frac{\lambda\cos2\theta}{\sqrt{1-\lambda^2\sin^2\theta}}+\frac{\lambda^3(\sin2\theta)^2}{4\sqrt{(1-\lambda^2\sin^2 2\theta)^3}} \tag{5-9}$$

$$\ddot{\theta}=\frac{1}{2}\frac{\mathrm{d}[\omega(\theta)]^2}{\mathrm{d}\theta} \tag{5-10}$$

其中，λ——曲径连杆比，$\lambda=r/L$；L 为连杆长度；$\omega(\theta)=\dot{\theta}$。

从上式可知，当扭矩波动时，转速 ω 必然发生波动；反过来，ω 的波动则反映了扭矩的波动，而扭矩的波动又与气缸工作状态，如喷油、进气、燃烧等因素有关，因此可以利用速度传感器测取瞬时转速信号，以诊断发动机各缸工况。

3. 瞬时转速信号检测方法

目前，瞬时转速的测量主要是借助于光电、电涡流和磁电式传感器来实现的。根据计算方法的不同，转速测量可分为频率法和周期法两种，其基本原理是通过测量转速传感器发出信号的频率或周期来获取转速值。频率法是在规定的检测时间内，检测转速传感器所产生的脉冲信号的个数来确定转速。周期法是测量相邻两个转速脉冲信号的时间来确定转速。

本节采用的是测周期法，其基本方法是通过测量飞轮齿圈上的两个相邻齿之间所经历的时间间隔，间接地计算出瞬时平均转速。由传感器直接测得的飞轮转速为近似正弦波信号，每个正弦波对应飞轮上的一个轮齿。由此可以得到发动机的瞬时转速的计算公式：

$$n=\frac{60}{Zt}=\frac{60f_s}{ZK} \tag{5-11}$$

式中，n——瞬时转速(r/min)；Z——飞轮的齿数；t——飞轮每个齿转过所需的时间(s)；K——一个正弦波内的采样点数；f_s——采样频率(Hz)。K 的计算累计误差最大为 2，则瞬时转速的相对误差为

$$\delta_t=\frac{nZ}{30f_s} \tag{5-12}$$

为了提高瞬时转速的测量精度，很多学者做了大量的工作，其中包括用高频计数器制作成高精度的数字式转速测量仪、将单片机引入发动机瞬时转速测量中等，可以将这

些方法统称为硬件法，其特点是通过提高计时精度从而提高测量精度。还有一种方法就是软件法，核心是采用插值方法，对用较低采样频率采集到的原始信号进行插值处理，以提高采样频率或精确地求得过零点时刻，从而显著提高发动机瞬时转速测量精度。

本节利用硬件法进行瞬时转速的测试，瞬时转速测试电路选用 20MHz 的高频晶振作为脉冲发生器，设计了基于外触发高频时钟计数的瞬时转速信号测试电路，其基本组成与工作原理如图 5-28 所示。该测试电路主要包括信号放大整形电路、高频计数电路、存储器、微控制器电路等。

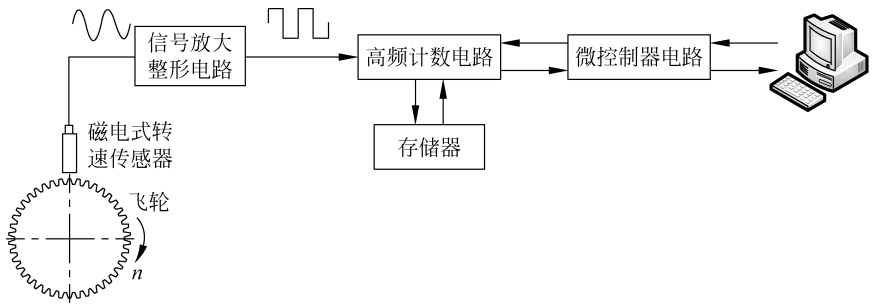


图 5-28　瞬时转速信号测试电路结构图

4. 瞬时转速信号分析方法

对于瞬时转速信号序列一般采用角域波形分析法和阶次谱分析法。

波形分析法需要观察瞬时转速信号的波动细节特征，一般截取柴油机 4 个工作循环内的瞬时转速波动曲线，观察在瞬时转速整体波动趋势曲线及其叠加的高频波动分量。通过瞬时转速的整体变化趋势，其波动周期和强度反映了电子调速系统对转速的调节能力，在多个工作循环内的波动分量则应具有明显的规律性和周期性，而且在每个工作循环内均出现与气缸数相同次数的不均匀波动，这是由于柴油机曲轴扭振激振力矩做功产生的瞬时转速波动，各波动波形依次对应柴油机各缸工作相位，其波动不均匀性反映了各缸做功能力的差异。

阶次谱分析是通过对等角度采样信号进行傅里叶变换得到其阶次谱，在阶次域内研究信号中各分量的分布特性，进而分离出信号中的不同阶次分量。因为与柴油机气缸工作状态相关的信息包含在瞬时转速信号激振力矩做功分量中，而与电调分量无关。因此，需要从原始瞬时转速信号中去除其电调分量，并分离出反映气缸动力性能的激振力矩做功分量，进而提取该分量的特征参数对柴油机故障进行诊断与定位。由于两个分量的波动频率明显不同，可以通过滤波的方法进行分离。

设柴油机曲轴飞轮齿数为 z，则柴油机每转内的信号采样点数，即采样阶次 O_s，采样间隔角 $\Delta\theta$ 之间存在如下关系：

$$O_s = z = \frac{2\pi}{\Delta\theta} \tag{5-13}$$

以 O_s 为采样阶次，L 为采样长度，采集得到的角域内瞬时转速信号表示为 $n_l(\theta)(l=0,1,\cdots,L-1)$，对其进行 L 点离散傅里叶变换，可得瞬时转速信号的阶次谱如下：

$$N_m(O)=\sum_{l=0}^{L-1}n_l(\theta)\mathrm{e}^{-\mathrm{j}\frac{2\pi ml}{L}},\quad m=0,1,\cdots,L-1 \tag{5-14}$$

式中，$N_m(O)$表示第 m 个阶次谱。

柴油机信号阶次谱分析中，以曲轴旋转一周为基准进行阶次计数。因此，定义曲轴旋转阶次为 1，曲轴旋转阶次的 r 倍称为 r 阶。在阶次域中，瞬时转速信号电调分量为低阶趋势分量，激振力矩做功分量为高阶细节分量。在不同激振力矩作用下，瞬时转速信号激振力矩做功分量中包含不同的阶次成分。在各激振力矩中，气缸内燃烧气体压力力矩作用最强，其产生的阶次成分幅值最大，该成分即为柴油机发火阶次。若四冲程柴油机气缸数为 g，则曲轴每转内气缸发火次数为 $g/2$，即发火阶次为 $g/2$。对于四冲程柴油机而言，由于其曲轴旋转一圈完成半个工作循环，使得瞬时转速信号激振力矩中出现半次谐波分量，各次分量的阶数分别为 $O=0.5,1.0,1.5,2.0,\cdots$。柴油机各缸工作均匀时，由曲轴扭振激振力矩产生的瞬时转速波动信号阶次谱中幅值较大的阶次成分主要为发火阶次及其谐阶次/谐波阶次，称为主阶次。

5.5.3 柴油机瞬时转速信号采集与分析

1. 所用设备

序　　号	名　　称	型　　号	数　　量
1	西门子工控计算机	IPC3000 SMART	1
2	8 缸柴油发动机	—	1
3	瞬时转速分析仪	自研	1
4	光电传感器	GD320	1
5	夹持式油压传感器	KG7	1

2. 实验系统搭建

柴油机瞬时转速信号测试系统如图 5-29 所示。该系统主要包括转速传感器、外卡油压传感器、瞬时转速分析仪、工控计算机。本次实验对象为某型 8 缸大功率电控柴油机。为了方便确定故障缸，以柴油机左 1 缸高压油管上的外卡油压传感器产生的喷油压力信号作为触发信号，对转速传感器输出的瞬时转速信号实施触发采样。

该型号柴油机的实验台架如图 5-30 所示，其输出轴飞轮齿数为 160。

转速传感器选用 M16-85 型磁电式转速传感器，将其正对柴油机飞轮轮齿安装于飞轮壳上，如图 5-31(a)所示。喷油压力信号测试时选用 KG7H 型外卡油压传感器，将其安装于左 1 缸高压油管管路上采集该气缸喷油压力信号，如图 5-31(b)所示。

瞬时转速分析仪外形如图 5-32 所示，其内部主要由电源电路、转速信号调理电路、电荷放大器和瞬时转速测试电路组成。外卡油压传感器输出信号首先输入电荷放大器，然后整形调理后作为时标信号输入瞬时转速测试电路，以实现瞬时转速信号的触发采集。

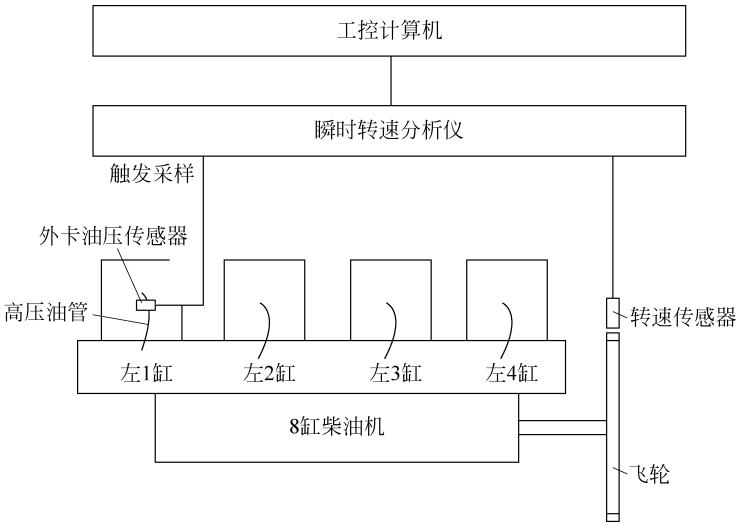


图 5-29　柴油机瞬时转速信号测试系统

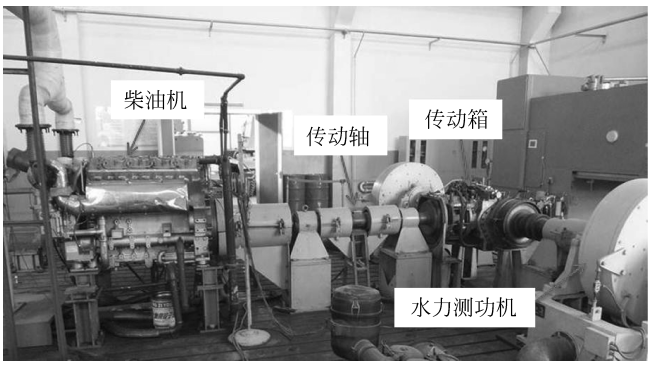
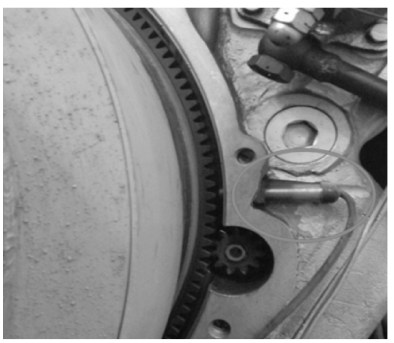
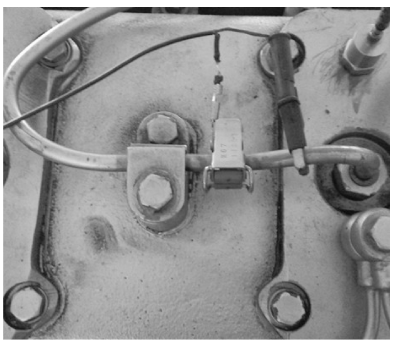


图 5-30　大功率柴油机实验台架



(a) 磁电式转速传感器



(b) 外卡油压传感器

图 5-31　传感器安装位置

图 5-32　瞬时转速分析仪

3. 测试步骤与结果分析

瞬时转速信号测试实验在柴油机平均转速为 1000r/min，匀速空载状态下进行。瞬时转速分析仪的数据采样长度设定为 16000 点。柴油机正常工况下的瞬时转速信号波形如图 5-33 所示。由于触发采样信号为左 1 缸喷油压力信号，所以瞬时转速信号的初始采样点处于左 1 缸压缩冲程后期，进而根据各气缸发火顺序即可确定瞬时转速波形中各波峰对应的气缸序号。已知本实验系统中柴油机飞轮齿数为 160，则柴油机每个工作循环的瞬时转速采样点为 320，根据采样点数即可确定单工作循环内的瞬时转速波形。

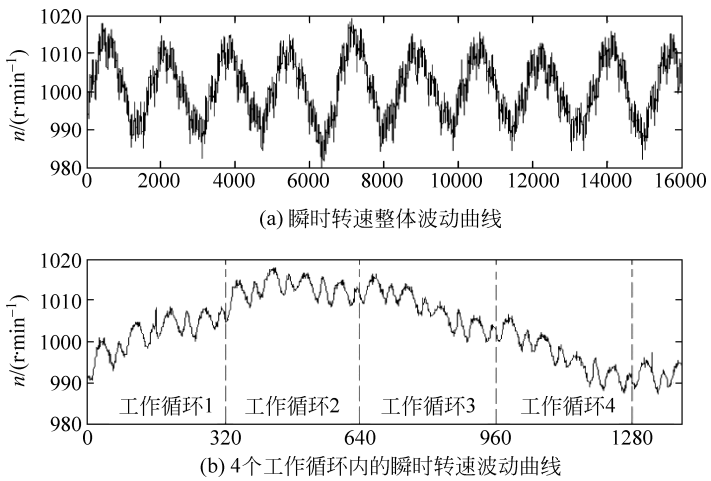


图 5-33　柴油机正常工况下的瞬时转速信号波形

电控柴油机的瞬时转速在电子调速系统与曲轴扭振激振力矩的共同作用下产生周期性波动。图 5-33(a)为瞬时转速信号的整体波形，由图可见瞬时转速信号的整体波动规律表现为在平均转速 1000r/min 上下周期性均匀波动，波动范围为 980～1020r/min。这是由于电子调速系统对平均转速的调节机制产生的，反映了瞬时转速的整体变化趋势，其波动周期和强度反映了电子调速系统对转速的调节能力。为观察瞬时转速信号的波动细节特征，截取柴油机 4 个工作循环内的瞬时转速波动曲线，如图 5-33(b)所示，可

见在瞬时转速整体波动趋势曲线上叠加有高频波动分量。该波动分量在多个工作循环内具有明显的规律性和周期性，而且在每个工作循环内均出现与气缸数相同的 8 次不均匀波动，这是由于柴油机曲轴扭振激振力矩做功产生的瞬时转速波动，各波动波形依次对应柴油机各缸工作相位，其波动不均匀性反映了各缸做功能力的差异。由此可知，电控柴油机的瞬时转速信号中包含电子调速系统产生的低频变化趋势分量与曲轴扭振激振力矩做功产生的高频波动细节分量，前者称为瞬时转速信号电调分量，后者称为瞬时转速信号激振力矩做功分量。

根据式(5-14)得到的瞬时转速信号阶次谱如图 5-34 所示。瞬时转速信号阶次谱中幅值较大的主要阶次成分为 0.1 阶、2.0 阶、4.0 阶、6.0 阶。0.1 阶信号为低阶趋势分量，幅值远远大于其他阶次，对应瞬时转速信号电调分量。2.0 阶、4.0 阶和 6.0 阶分别为瞬时转速信号激振力矩做功分量中的不同阶次成分，该分量中的其他阶次成分幅值较小。4.0 阶次为 8 缸柴油机的发火阶次，2.0 阶次与 6.0 阶次分别为发火阶次的 0.5 倍和 1.5 倍谐(波)阶次成分。

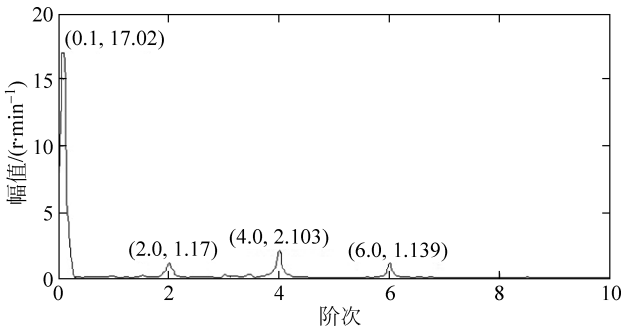


图 5-34　瞬时转速信号阶次谱

思考题

1. 构建柴油机转速信号测试系统，编写采集程序，同步采集转速与油压传感器输出信号，显示其波形。

2. 利用 MATLAB 软件进行瞬时转速信号波形分析和谱分析。

5.6　案例 5　燃油压力信号采集与分析

5.6.1　实践目的

(1) 掌握柴油发动机燃油压力信号测试原理和采集方法。

(2) 通过案例学习，学习燃油信号的时域波形分析法。

5.6.2 柴油机燃油压力信号测试原理

1. 柴油机供油系统

柴油机供油系统主要由燃油泵、出油阀、高压油管和喷油器组成，它直接影响燃烧过程和柴油机的工作性能，统计资料表明柴油机的故障30%以上是发生在供油系统。通过检测柴油发动机高压柴油泵供油压力，可发现下列故障：喷油器弹簧弹性减弱、喷雾针与座不密闭、喷油孔磨损或部分阻塞、高压柴油泵出油活门不密闭、高压泵柱塞偶件严重磨损等。

供油系统工作不正常的结果是直接降低功率和热效率，功率下降后，必须增加供油量以满足功率的需要。热效率降低不但使损失的热量增加，还会引发一些重大故障，如可引起活塞过热，排气门烧蚀，润滑油结焦，水温、油温不正常升高等。喷油器的喷射能力改变或喷油器针阀运动受阻，都会影响喷油雾化质量，导致燃烧不良引发故障。这类故障刚发生时并无明显的异常现象，但随着劣化的积累，活塞环局部黏结，排气门的密封性也被破坏，进一步恶化了燃烧过程，使柴油机运转不正常。可见，检测供油系统的工作状态，对保证柴油机可靠安全的运行很重要。

在供油系统在不解体情况下，若想通过检测雾化质量，如平均油滴直径、靠近喷孔的油束锥角等，来完成供油系统故障诊断是不可能的。但获取供油系统的燃油压力波形相对容易，以燃油压力分析为基础，可以完成供油系统的状态检测，辨别其典型故障和异常喷射等。

2. 燃油压力检测方法

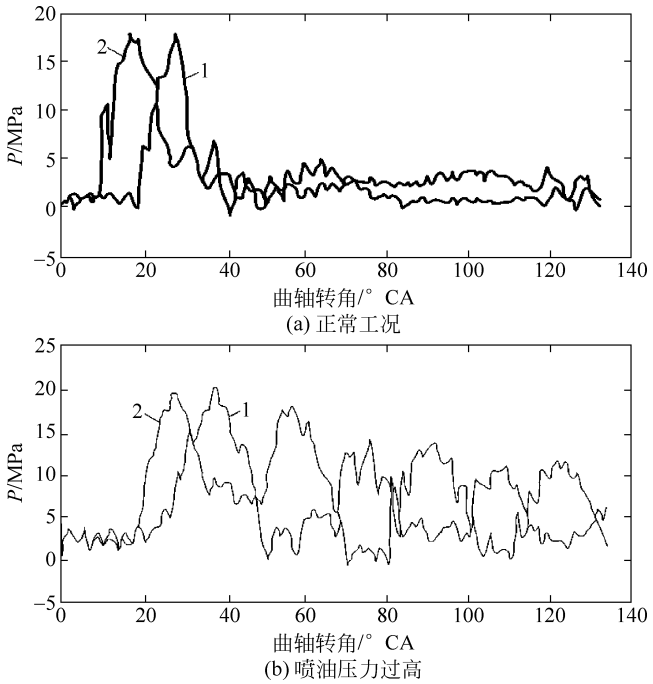
燃油压力检测常采用供油压力传感器。检测某缸供油压力时，松开该缸高压油管接头，串接供油压力传感器，通过传感器可测量该缸供油最高压力值、油管中残余压力值和供油压力波形等。

燃油压力检测也可采用不解体的外卡式压力传感器，不需拆卸油管，只需要将外卡式压力传感器分别卡在各缸高压油管靠近喷油器处，再分别检出供油压力波形，将所测波形与标准波形比较，或各缸相互比较，便可判别某缸的供油故障。

图5-35中，图(a)为某发动机正常工况下的燃油压力波形，图(b)为喷油压力过高时燃油压力波形。图中，1为喷油压力曲线，2为供油压力曲线。与正常喷射压力波形相比，喷油压力过高时的压力波形不但出现二次喷射，三次压力峰值也较大，而且供油提前角与喷油提前角均有所减小。

喷油泵端与喷油器端的油压波形差别较大是由于在压力大幅度变化的作用下，燃油存在可压缩性，高压油管也有弹性，使高压系统形成一个弹性系统。在供油过程中，当出油阀开启时，高压油管中，喷油泵端燃油产生的压力波向喷油器端传播，如果不足以升起针阀，则压力波全部被反射，向喷油泵端传播，与该处新产生的压力波叠加起来，又被反

射而向喷油器端传播。当压力传播使喷油器端燃油压力升高到大于针阀开启压力时，针阀即打开，喷油开始，此时传至喷油器端的压力波仍有部分被反射回去，形成弹性振荡。所以在整个供油过程中，由于压力波动现象存在，使实际喷油过程与柱塞的供油过程不一致。



(a) 正常工况

(b) 喷油压力过高

图 5-35　燃油压力波动分析

3. 基于外卡压力传感器检测机理

柴油机高压油管一般是用厚壁无缝钢管制成，可视为厚壁圆筒，在内部压力 P_1 和外部压力 P_2 的作用下，根据材料力学厚壁圆筒受力变形理论，得到油管外表面的径向位移分量 u 为

$$u=\frac{2}{E}\frac{a^2b}{b^2-a^2}P_1-\left[\frac{b}{b^2-a^2}\left(\frac{a^2+b^2}{E}-\frac{a^2-b^2}{E}\mu\right)\right]P_2 \tag{5-15}$$

式中，a——油管的内径；b——油管的外径；E——油管材料弹性模量；μ——泊松比。

实际高压油管承受的内部压力为燃油压力 P_1，外部压力为大气压的力和传感器夹持力之和 P_2，因为 $P_2 \ll P_1$，故 P_2 的影响可以忽略不计，则在静态条件下，高压油管的径向膨胀变形与其内部油压呈线性关系，因此为利用外卡式压力传感器实现油压波形的测量提供了理论基础。

针对柴油机高压油管中燃油的喷射过程，忽略油道中压力波传递的影响，可得靠近喷油端某点供油压力的微分方程为

$$\frac{\mathrm{d}P_D}{\mathrm{d}t}=\frac{E_r}{6nv_D}\left[\frac{F_e}{c_y\rho_r}(2P_{vD}-P_D+P_0)-\frac{\mathrm{d}h}{\mathrm{d}t}\cdot 6n\cdot F_n-\mu s6n\sqrt{\frac{2}{\rho_r}(P_D-f_p)}\right] \tag{5-16}$$

式中，P_D——供油压力；t——供油时间；E_r——燃油弹性模数；μ——喷孔流量系数；n——柴油发动机转速；v_D——油嘴中空腔容积；c_y——油液中的声速；ρ_r——燃油密度；P_{vD}——前进波的压力；P_0——针阀开启压力；F_n——针阀截面积；F_e——阀腔入口截面积；f_p——气缸中燃气压力；s——出油阀座面处通道截面积；h——柱塞行程。根据式(5-16)，在转速 n 一定的情况下，供油压力 P_D 与 t、v_D、P_0 和 f_p 有关，但主要取决于 P_0。

图 5-36 为教科书上柴油机某缸燃爆过程中的燃油压力信号波形。图中 a 点是高压油泵开始供油点，当油泵开始关闭进油孔时，高压泵内的燃油被压缩，当压力超过剩余压力时，燃油进入高压油管，但喷油器并未马上喷油，必须使喷油器内的压力超过针阀开启压力时，针阀才打开并开始喷油(b 点)，在针阀打开过程中，由于针阀上升让开容积以及燃油开始喷入气缸，油管中燃油压力暂时下降(c 点)，由于喷油泵压油的速度增加，所以油管中的油压继续上升直到最大值(d 点)。当油泵压油的速度降低时，油管内压力下降，当油压低于喷油器针阀关闭压力时，针阀关闭(e 点)。针阀关闭后，油泵继续将一部分燃油压入油管，油压有所回升(f 点)，然后油泵出油阀关闭，油压下降到剩余压力。

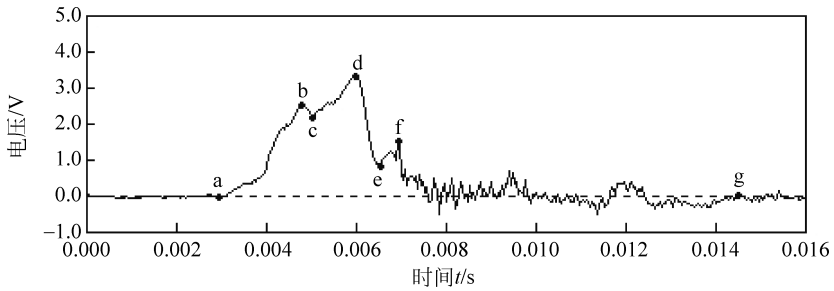


图 5-36　燃油压力信号波形

图 5-37 是采用 AVL 公司的夹持式压力传感器在左 3 缸喷油器端实测获得的燃油压力波形，图中 $P_{\max}$ 是最大燃油压力，α 是喷油提前角。比较图 5-36 与图 5-37 左 3 缸喷油前后的实测压力波动曲线与理论波形相符。

对于燃油压力信号，一般采用时域波形分析法，根据时序关系，选取三个特征参数，即最大油压 $P_{\max}$、喷油提前角 α(开始喷油点到上止点的曲轴转角)，脉冲因子 I_f，其计算式为

$$I_f=\frac{\max(P_n)-\min(P_n)}{\frac{1}{N}\sum_{n=1}^{N}|P_n|} \tag{5-17}$$

式中，P_n——油压信号($n=1,2,\cdots,N$)；N——采样点数。

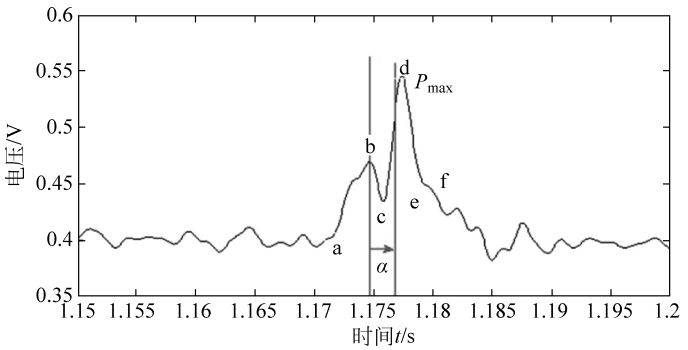


图 5-37　外卡压力传感器所测燃油压力波形

5.6.3　柴油机燃油压力信号采集与分析

1. 所用设备

序　　号	名　　称	型　　号	数　　量
1	西门子工控计算机	IPC3000 SMART	1
2	数据采集卡	NI PCI-6233	1
3	8 缸柴油发动机	—	1
4	光电传感器	GD320	1
5	夹持式油压传感器	KG7	1

2. 实验系统搭建

图 5-38 为燃油压力信号采集系统。喷油压力信号测试时选用 KG7H 型夹持式油压传感器，将其安装于与 5.5 节实验相同的 8 缸柴油机左 1 缸高压油管管路上采集该气缸喷油压力信号(图 5-39)。光电传感器安装在发动机的输出轴旁，在输出轴上贴一张白纸用于反射光电传感器发出的光信号(图 5-40)。调理装置主要由稳压电源、电荷放大器等组成，对油压信号、上止点信号进行放大调理后输出至 A/D 数据采集卡。

3. 测试步骤及结果分析

测试工作在 8 缸柴油机试验台上进行，采集左 1 缸油压信号和上止点信号。上止点信号通过贴在曲轴上的反光纸和光电传感器进行采集，信号采集主要设备为 PCI-6233 型数据采集卡，采样率设置为 12.5kS/s。

在平均转速保持 1000r/min，柴油机正常工作的工况下采集到的左 1 缸油压信号及其压缩行程上止点信号如图 5-41 所示。

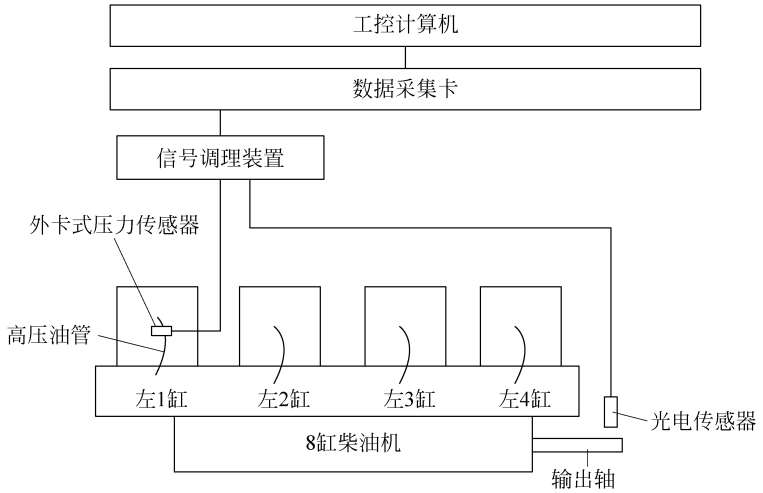


图 5-38　柴油机燃油压力信号采集系统

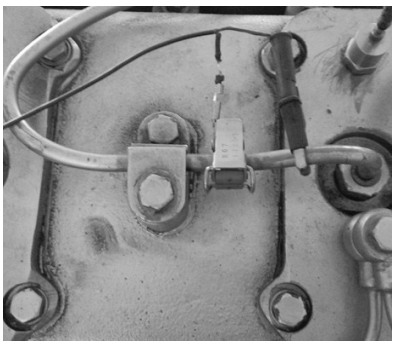


图 5-39　夹持式传感器安装位置

图 5-40　光电传感器安装位置

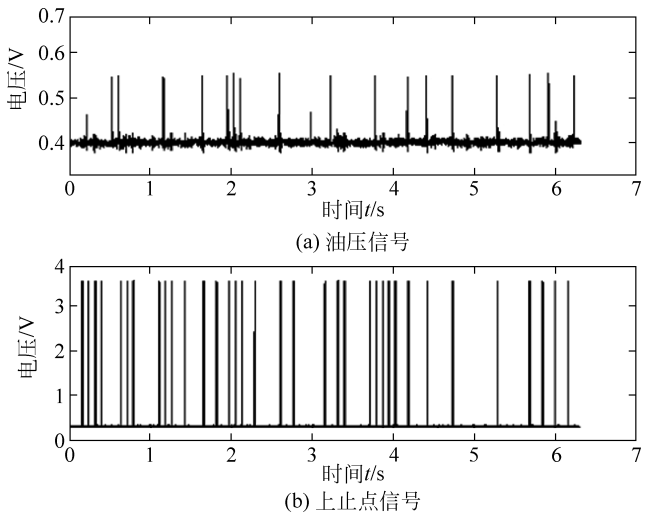


图 5-41　左 1 缸油压信号与上止点信号

图 5-42 为两信号对应关系的细节放大图。从图中可知，上止点信号的上升沿对应于燃油压力最大值出现的时刻。

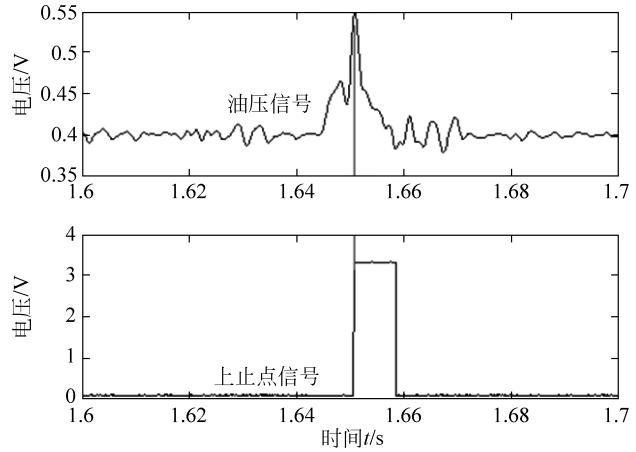


图 5-42　油压信号与上止点信号对应关系细节

思考题

1. 独立构建柴油机燃油压力信号测试系统，编写采集程序，同步采集上止点与油压传感器输出信号，显示其波形。

2. 依据本节所提出的 3 个燃油压力信号波形参数，利用 MATLAB 软件对所采集到的数据进行分析。

5.7　案例 6　车辆底盘总线系统信息采集

5.7.1　实践目的

(1) 掌握车辆底盘系统 CAN 总线的通信协议以及总线信息采集方法。

(2) 通过案例学习，学习 CAN 总线信息采集程序的设计方法。

5.7.2　底盘系统总线简介

车辆底盘电气系统采用了 CAN 总线技术后，发动机、综合传动、电气设备等部件之间可以实现控制信息点对点、多点、组播式通信，可以有效地减少控制线缆数目，提高系统的可靠性。以某型履带车辆底盘为例，底盘综合电子系统和驾驶员综合操控装置、发动机电控系统、综合传动电控盒、数字微压差测量仪等设备都具有 CAN 通信接口，它们均为 CAN 网络上的节点，这些节点共同构成了整车底盘的 CAN 总线通信网络。

该型履带车辆底盘系统 CAN 总线系统采用双冗余 CAN 总线结构。

1. 协议标准

总线结构采用双冗余的 CAN 总线，其物理层、数据链路层和应用层遵循 CAN2.0B 协议标准，数据传输速率为 250kb/s，总线的应用层在 SAE J1939 协议标准上进行扩充。

2. 标识符(ID)格式要求

采用扩展数据帧格式，使用 29 位 ID 标识符进行总线通信：

11 位标识符			SRR	IDE	18 位标识符			RTR
PRIORITY 3-1	R	PF 8-3			PF 2-1	PS 8-1	SA 8-1	
28-26	25-24	23-18			17-16	15-8	7-0	

PRIORITY——简写 P，3 位优先级位，可以有 8 个优先级，000 具有最高优先级；
R——保留位，目前固定为 00；
PF——8 位报文类型代码；
PS——8 位目标地址或报文类型扩展码；
SA——8 位发送报文源地址。

3. 数据帧要求

要发送的 CAN 数据信息分为短帧信息(数据长度小于等于 8 字节)和长帧信息。

当车辆工作时，在 CAN 总线上汇集了很多各部件的工作状态信息，因此，研究如何基于车辆底盘 CAN 总线提取各部件的信息，对实现车辆底盘系统状态评估和故障快速定位很关键。

5.7.3 传动系统 CAN 总线信号采集

1. 所用设备

序　号	名　称	型　号	数　量
1	便携式笔记本电脑	TY-YN800 型	1
2	智能 USB CAN 接口卡	USBCAN-2A	1
3	检测接口电缆		1

2. 实验系统搭建

所用的 CAN 总线检测设备结构如图 5-43 所示。其中包括便携式计算机、USB CAN 卡以及检测接口电缆。

图 5-43　CAN 总线检测设备结构

3. 核心代码分析

本节的主要工作是如何通过编程控制 USB CAN 接口卡，核心问题是要将 CAN 卡生产厂商所提供 DLL 中的函数打开，程序中采用了显式调用的方法来获取相关函数的地址，核心代码如下：

```
  GetProjectDir (ParameterPathName);
  strcat (ParameterPathName, "\\ControlCAN.dll");          //dll 文件
  m_hDLL = LoadLibrary(ParameterPathName);                 //打开动态库
  //取得函数地址
  VCI_OpenDevice = (LPVCI_OpenDevice)GetProcAddress(m_hDLL,"VCI_OpenDevice");
  VCI_CloseDevice = (LPVCI_CloseDevice)GetProcAddress(m_hDLL,"VCI_CloseDevice");
  VCI_InitCAN = (LPVCI_InitCan)GetProcAddress(m_hDLL,"VCI_InitCAN");
  VCI_ReadBoardInfo = (LPVCI_ReadBoardInfo)GetProcAddress(m_hDLL,"VCI_ReadBoardInfo");
  VCI_ReadErrInfo = (LPVCI_ReadErrInfo)GetProcAddress(m_hDLL,"VCI_ReadErrInfo");
  VCI_ReadCanStatus = (LPVCI_ReadCanStatus)GetProcAddress(m_hDLL,"VCI_ReadCANStatus");
  VCI_GetReference = (LPVCI_GetReference)GetProcAddress(m_hDLL,"VCI_GetReference");
  VCI_SetReference = (LPVCI_SetReference)GetProcAddress(m_hDLL,"VCI_SetReference");
  VCI_GetReceiveNum = (LPVCI_GetReceiveNum)GetProcAddress(m_hDLL,"VCI_GetReceiveNum");
              VCI _ ClearBuffer = (LPVCI _ ClearBuffer) GetProcAddress (m _ hDLL," VCI _
ClearBuffer");
            VCI_StartCAN = (LPVCI_StartCAN)GetProcAddress(m_hDLL,"VCI_StartCAN");
            VCI_ResetCAN = (LPVCI_ResetCAN)GetProcAddress(m_hDLL,"VCI_ResetCAN");
            VCI_Transmit = (LPVCI_Transmit)GetProcAddress(m_hDLL,"VCI_Transmit");
            VCI_Receive = (LPVCI_Receive)GetProcAddress(m_hDLL,"VCI_Receive");
  if( VCI_OpenDevice(m_devtype, m_devind, 1) == STATUS_OK )
  {
      if( VCI_InitCAN(m_devtype, m_devind, m_connect, &initconfig) == STATUS_OK)
      {
            if( VCI_StartCAN(m_devtype, m_devind, m_connect) == STATUS_OK)
            {
                  readok = 1; // MessagePopup("Warn","readok = 1!");
                  SetCtrlVal(panel,CDPANEL_LED,1);
                  SetCtrlAttribute(panel,CDPANEL_COMMZERO,ATTR_DIMMED,1);
            }
      else
            MessagePopup("Warn","启动 CAN 失败!");
      }
  else
       MessagePopup("Warn","初始化 CAN 错误!");
        }
else
  MessagePopup("Warn","打开端口错误!");
```

```
        break;
    }
        return 0;
    }
```

4. 传动系统信息采集

利用检测电缆将计算机 CAN 口与底盘工况监测记录仪 CAN 数据输出插座相连。打开工况监测盒的开关，在检测项目中选择综合传动系统检测，然后单击“开始检测”按钮，进入综合传动系统的检测，如图 5-44 所示。

图 5-44　综合传动系统检测界面

单击“打开设备”按钮，打开 CAN 设备，指示灯为绿色。单击“修改波特率”按钮，将底盘上的 CAN 网络的传输波特率转化为传输文件时的高速状态，修改成功后指示灯显示为绿色，否则将弹出相应的提示窗口。

若上述步骤操作成功，单击“下载数据”按钮，自动下载工况监控盒内的数据到界面上，数据将以监测时间为顺序依次显示在如图 5-45 所示的表格中，相应的诊断结果也会在提示框中给出。

思考题

独立编写 CAN 总线信息采集程序，完成底盘 CAN 总线上的综合传动系统的技术状态数据的采集、显示和存储。

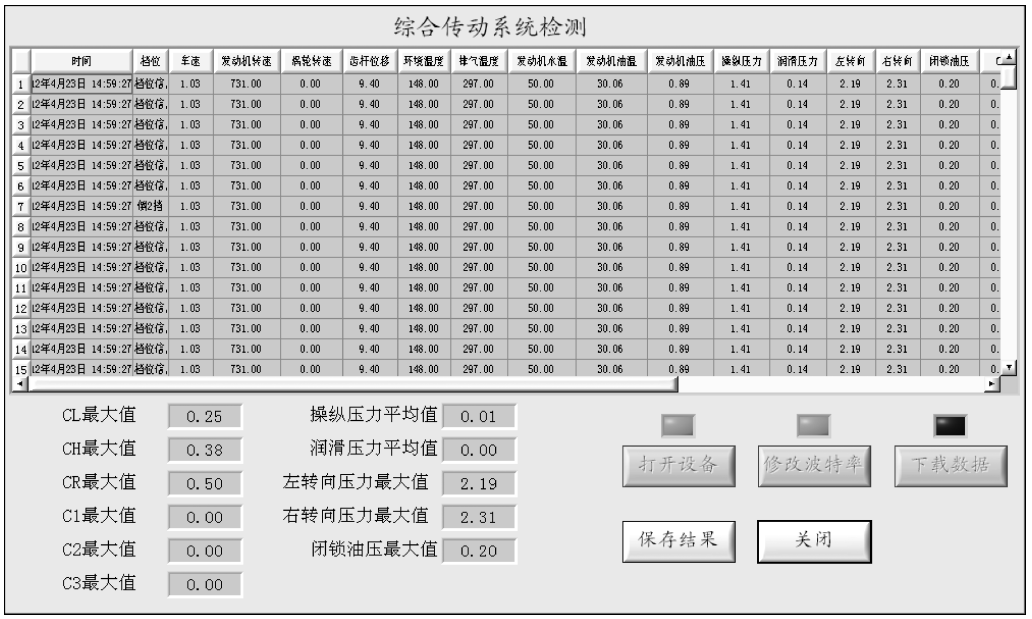


图 5-45　综合传动系统数据下载完成界面