# LabWindows/CVI 在半实物分布式仿真系统中的应用

李海 武小栋

(北京理工大学电子工程系，北京，100081)

*本文已经发表在《电子技术应用》2005 年第 4 期*

**摘要**：基于 HLA 的分布式系统仿真在半实物仿真中的比重日益增加。将 LabWindows/CVI 应用于半实物分布式仿真非常有利于系统的开发。本文以一个通信仿真系统的开发为例，针对传统的 LabWindows/CVI 调用外部 DLL 的方法不能应用于 HLA 系统的问题，提出创建 LabWindows/CVI 的 DLL 和利用外部编译器两种解决方案将 LabWindows/CVI 和 HLA 仿真系统结合在一起。

**关键词**：LabWindows/CVI；虚拟仪器；半实物分布式仿真系统；HLA；RTI

**中图分类号**：TN97

**Application of LabWindows/CVI in Hardware-in-the-loop Distributed Simulation System**

LI Hai, WU Xiao-dong

（Department of Electronics Engineering, Beijing Institute of Technology, Beijing, 100081）

**Abstract**：The proportion between hardware-in-the-loop simulation and distributed simulation system based on HLA is increasing. Application of LabWindows/CVI is propitious to the development of hardware-in-the-loop distributed simulation system. Two methods, creating DLLs in LabWindows/CVI and using external compiler, are introduced in this paper by an example of communication simulation system, to solve the problem that calling external DLLs in LabWindows/CVI can't be used in HLA system. Both methods can combine the power of LabWindows/CVI and HLA simulation system.

**Keywords**：LabWindows/CVI；Virtual Instrument；Hardware-in-the-loop Distributed Simulation System；HLA；RTI

半实物仿真又称为硬件在回路中的仿真（Hardware-in-the-loop Simulation），是一种将实物接入仿真回路中的仿真试验。半实物仿真已经成为航空航天、武器系统等研究领域中不可缺少的重要手段。随着计算机仿真技术的迅猛发展，仿真系统变得越来复杂，许多复杂的仿真要牵涉到一些不同类型系统的仿真的联合，而这些系统各自的仿真环境组成了整个仿真环境。为了解决不同类型的仿真间的互操作和仿真部件的重用，美国国防部公布了建模与仿真领域里的高层体系结构(HLA：High Level Architecture)。HLA正日益得到重视和广泛的应用，基于HLA的分布式系统仿真在半实物仿真中的比重日益增加[1]。而另一方面，NI公司的LabWindows/CVI集成了GPIB、VXI、PXI、RS232/485 和插入式（PCI、USB）数据采集设备等进行通信的功能，支持DataSocket和TCP/UDP等技术与远程应用程序通信，作为虚拟仪器开发工具在数据采集和界面控制方面具有明显的优势。所以，将LabWindows/CVI应用于半实物分布式仿真非常有利于系统的快速开发。

本文以一个通信对抗仿真系统的开发为例，分析了 LabWindows/CVI 应用于基于 HLA 的半实物仿真系统中所存在的问题，并提出两种解决方案将 LabWindows/CVI 和 HLA 仿真系统完美结合。

## 1. 半实物仿真系统结构

我们所开发的半实物通信对抗仿真系统中的干扰机和通信设备均为硬件实物，而干扰机和通信设备之间的电磁环境是使用软硬件模拟产生的。系统的简化框图如图 1 所示。

该仿真系统共有 7 个 HLA 成员，HLA 成员之间通过 RTI 发送和接收数据。各成员的主要功能如下：

- 指挥控制成员负责仿真场景与系统参数的设置，通过 HLA 进行时间推进，并不断发送控制参数给各成员进行多种侦察和干扰实验。
- 干扰器成员、模拟器成员和地面站成员将从 HLA 系统接收到的仿真命令发送给硬件实物并根据从硬件采集到的数据进行分析，将捕获时间、误码率等信息发送给 HLA 中的其他成员。
- 环境模拟成员随着仿真推进实时计算各设备的位置、姿态、速度、距离衰减和多普勒频移，并控制连接各设备的微波网络，以模拟通信设备之间的链路。
- 效能评估成员根据从实物采集到的数据计算干扰效能。
- 视景仿真成员显示各硬件实物在仿真场景中的位置、姿态和其他信息。

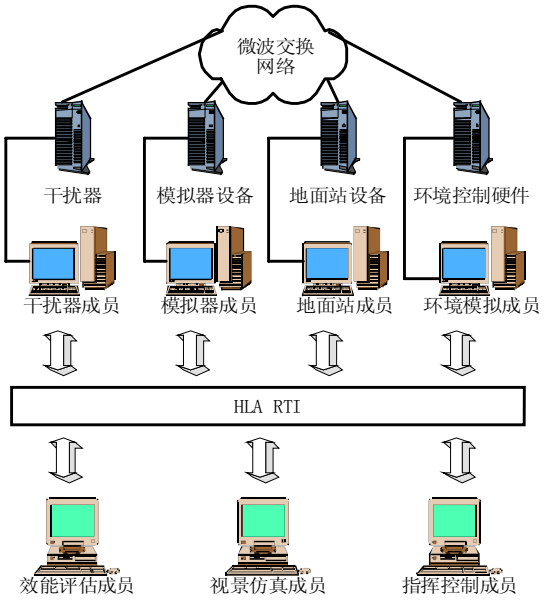


图 1 半实物分布式仿真系统结构

由于设备的用途、使用环境各异，考察的指标各有侧重，设备实物和工控计算机之间的连接类型也各不相同（见表 1）。借助 LabWindows/CVI 的数据采集和处理能力，可以使每台工控机变成一台虚拟仪器，完成控制数据的发送、遥测数据的采集和实时显示功能。这些工控机程序同时又是 HLA 的仿真成员，在 HLA 的协调下进行工作。工控机的双重使命使我们有必要研究如何将虚拟仪器技术和分布式仿真平台 HLA 结合起来。

表 1 仿真成员和实物之间的硬件接口关系

| 仿真成员 | 接口类型 | 用途 |
|---|---|---|
| 干扰器成员 | RS485 | 遥控指令和回传状态。 |
| | RS422 | 回传中频采集数据。 |
| 模拟器成员 | CAN 总线 | 遥控指令和回传误码率。 |
| | 7 路 CMOS 开关信号 | 备份加电控制接口。 |
| | 40 路 CMOS 开关信号和 10 路 0~5V 模拟信号 | 模拟器实物的状态监测。 |
| | RS422 和 LVDS | 回传 1 组中频采样数传、3 组解调数据数传和 3 组返向数传数据 |
| 地面站成员 | 100M 以太网 | 遥控指令和回传状态。 |
| 环境模拟成员 | USB 2.0 高速模式 | 指令控制和修正数据。 |

2. LabWindows/CVI 应用于 HLA 系统的两种解决方案

LabWindows/CVI提供了多种和其他开发工具的编程接口，其中最常用的方法是以

LabWindows /CVI编写主程序，而使用其他开发工具编写DLL（Dynamic Link Library），再将DLL加入到LabWindows/CVI系统中使用[2]。这种方法对于一般的硬件采集和处理来说是非常简便的，但是对于本文所讨论的HLA系统无法适用。所有HLA应用都是通过RTI接口库来调用HLA所提供的功能，RTI库一般是以C++或Java类库的形式提供的，而LabWindows/CVI只能调用使用C语言接口的DLL库文件。曾经有人尝试过将HLA 的服务封装成MEX(Matlab规定的C语言接口的DLL)[3]，但一直没有进入实用阶段，因为RTI提供 100 多种仿真服务，要将这些复杂的服务都以C语言接口的形式封装成DLL，不但费时费力，而且必然削弱HLA作为面向对象的分布式仿真平台的功能。所以，必须寻找其他的方法来解决这个问题。在实际开发中，我们先后使用两种解决方案，都可以解决前面的问题。下面分别加以介绍。

2.1. 基于 LabWindows/CVI 的 DLL 的方法

通常的 LabWindows/CVI 程序开发是在其集成环境中编写 C 程序，最终编译生成可执行文件(.exe)。采用基于 LabWindows/CVI 的 DLL 的方法与此不同，该方法将 LabWindows/CVI 程序编译成为 DLL 库，然后在 Visual C++编写的主程序中加以调用。软件的主程序使用 Visual C++编写，可以调用 RTI 库加入 HLA 仿真系统，并且调用 LabWindows/CVI 编写的 DLL 所提供的输出函数进行界面显示、数据采集与控制。而当用户在界面上进行操作或者数据采集完成时，LabWindows 的 DLL 程序可以借助 Visual C++主程序提供的回调(callback)函数来通知主程序，并将数据发送给主程序。软件各模块的调用关系可以用图 2 表示。

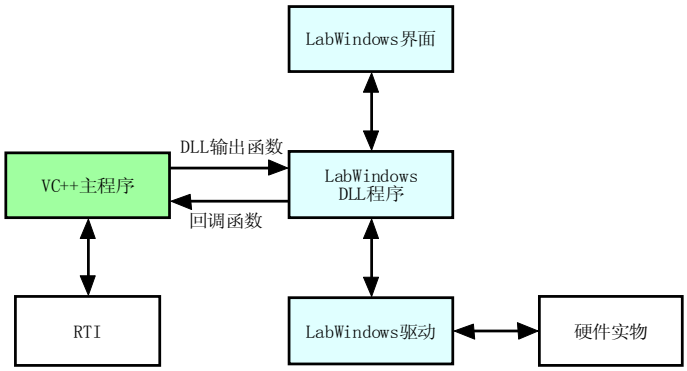


图 2 基于 LabWindows/CVI 的 DLL 的方法的调用关系

要生成 DLL 库，需要在 LabWindows/CVI 中选择菜单“Build | Target Type | Dynamic Link Library”。选择菜单“Edit | Insert Construct | DllMain”向.c 文件中加入 DLL 所需要的 DllMain 函数。

LabWindows/CVI 中最主要的工作是编写 DLL 输出函数。例如下面的 CallCVI 函数显示 LabWindows 的用户面板 TestUI.uir：

```
int __stdcall CallCVI ()
{
   if ((panelHandle = LoadPanelEx (0, "TestUI.uir", PANEL,  __CVIUserHInst)) < 0)
        return -1;
   DisplayPanel (panelHandle);
   RunUserInterface ();
   DiscardPanel (panelHandle);
   return 0;
}
```

这里使用的是 LoadPanelEx 函数，而不是通常使用的 LoadPanel 函数。如果使用 LoadPanel

函数，在 Visual C++中运行时会因为找不到.uir 文件而报告错误。在 LabWindows 中要实现 DLL 输出函数，需要将函数的声明写到一个头文件(.h)中。不要直接将 DLL 输出函数的声明写到面板对应的.h 文件中，因为修改面板时，LabWindows/CVI 会重新生成.h 文件，从而丢失手工添加的定义。建立一个 export.h 加入到工程中，在 export.h 中加入 CallCVI 函数的定义。再选择菜单“Build | Target Settings”，点击对话框中 Exports 框架的 Change 按钮。在“DLL Export Options”对话框中，设置“Export What”为“Include File Symbols”，并选中 export.h。最后选择菜单“Build | Create Debuggable Dynamic Link Library”就可以创建.dll 库文件和.lib 文件。将.dll、.lib、.uir 文件和 export.h 文件拷贝到 Visual C++的工程目录下，从 Visual C++菜单中执行“Project | Add To Project | Files”，将.lib 和 export.h 添加到工程中。在程序中调用 DLL 中的函数 CallCVI()即可显示 LabWindows/CVI 的面板。借助 DLL 输出函数的参数，可以实现从 Visual C++程序向 LabWindows/CVI 程序中发送数据。

反过来，如果希望将 LabWindows/CVI 程序中的用户操作或外部输入数据传递给 Visual C++，可以通过由 Visual C++提供回调函数来实现。可以修改前面的 CallCVI 的定义为：

```
typedef int (CVICALLBACK *VCPROC)(void *callbackData);
static VCPROC pCallbackFunc = 0;

int __stdcall CallCVI (void* pFunc)
{
    pCallbackFunc = (VCPROC)pFunc;
     ……
}
```

这里使用 typedef 定义了一个回调函数类型 VCPROC。VCPROC 实际上是一个函数指针，函数的参数为 void 类型。函数指针 VCPROC 的返回值和参数可以根据实际应用的情况灵活修改。Visual C++调用 CallCVI 时，需要提供一个回调函数的地址作为参数，如：

```
CallCVI((void*)HLAProc);
```

这里的 HLAProc 是 Visual C++程序中的一个用户函数，它的参数和返回值都要和 VCPROC 中的定义一致。CallCVI 中，LabWindows/CVI 程序将回调函数的地址保存在全局变量 pCallbackFunc 中。当 LabWindows/CVI 中某个事件发生时，可以调用 pCallbackFunc 变量中所保存的函数指针通知 Visual C++程序。下面的例子中，用户点击 LabWindows/CVI 程序面板上的按钮后，程序调用回调函数发送一个字符串给 Visual C++，执行加入 HLA 联邦的操作：

```
int CVICALLBACK ClickCallback (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
        {
        case EVENT_COMMIT:
            if(pCallbackFunc)
            {    // 加入 HLA 联邦
                (*pCallbackFunc)((void*)"Join Fed");
            }
            break;
        }
    return 0;
```

}

在 Visual C++函数 HLAProc 函数中，可以实现对 RTI 的 CreateFederation、JoinFederation 等服务的调用。

LabWindows/CVI 编写的 DLL 也可以进行源代码级调试，但需要做一些额外的设置。先在 Visual C++中编写好调用 DLL 的程序并编译为 EXE 文件，然后在 LabWindows/CVI 中打开工程，选择菜单“Run | Specify External Process”，指定 Visual C++编写的 EXE 文件作为外部运行程序。在 LabWindows/CVI 程序中设置断点后，选择菜单“Run | Debug Project”进行程序调试。LabWindows/CVI 将先启动 Visual C++编写的程序，当执行到 LabWindows/CVI 的 DLL 内部的代码时就可以进行源代码级跟踪调试了。

2.2. 基于外部编译器的方法

另一个解决方案是利用Visual C++作为外部编译器来实现HLA和LabWindows/CVI程序之间的相互调用。LabWindows/CVI自身的编译器的核心是David R. Hanson开发的lcc，这个编译器以容错和调试为擅长，但是优化性能较弱[4]。使用外部编译器还可以生成优化的代码，从而提高程序运行效率。

使用这种方法，先借助 LabWindows/CVI 生成显示面板和硬件操作所需要的 C 语言代码，然后把代码和 LabWindows/CVI 的头文件和库文件添加到 Visual C++的工程中进行编译。由于 LabWindows/CVI 生成的代码和 Visual C++生成的代码是在同一个工程中的，可以直接相互调用，不需要象上一种方法那样设计 DLL 输出函数和 Visual C++回调函数。

要使用外部编译器，最主要的工作是为 LabWindows 的面板生成对象表。在 LabWindows/CVI 环境中开发程序时，LabWindows/CVI 会自动为所有面板上所有控件的回调函数建立一个对象表并链接到程序中，运行时利用这个对象表将.uir 文件中的控件对象和控件的回调函数对应起来的。而使用 Visual C++等外部编译器，并不能自动建立这样的对象表，也就不能自动识别控件对象的回调函数。因此，必须首先在 LabWindows/CVI 中手工生成对象表，才能在外部编译器中使用 LabWindows/CVI 的控件对象。基本操作步骤是在设计好 LabWindows/CVI 面板后，从菜单中执行“Build | External Compiler Support...”，在“External Compiler Support”对话框中设置“UIR CallBacks”为“Object File”，并输入文件路径和文件名 callback.obj，点击 Create 按钮即可创建 callback.obj 文件。然后将.uir 文件拷贝到与 Visual C++编译生成的可执行文件目录下（如 Debug 或 Release 目录）。这里需要特别注意的是.uir 所在目录和前一种方法是不同的。再将生成的.obj、.c 和.h 文件都添加到 Visual C++工程中，同时还需要将 cvi70/extlib/msvc 目录下的文件 cvirt.lib、cvisupp.lib 添加到工程中。在 Visual C++中调用 LabWindows/CVI 程序时，需要在#include 部分添加对 cvirte.h 和 userint.h 的包含。在调用 LabWindows/CVI 生成的代码之前，还需要调用 InitCVIRTE 函数进行初始化，如：

```
if (InitCVIRTE (AfxGetInstanceHandle(), 0, 0) == 0)
    return;        /* 内存不够 */
```

3. 总结

本文所介绍的两种方法都可以将 LabWindows/CVI 和 HLA 系统结合起来，而且两种方法各有特点：基于 LabWindows/CVI 的 DLL 的方法比较独立于外部编译环境，对于一些不使用 C++接口的 RTI（如使用 Java 语言接口的 pRTI）也是适用的，比较通用，应用范围较广；而基于外部编译器的方法可以直接调用 Visual C++所提供的各种功能，从而突破了 LabWindows/CVI 自身的 ANSI C 编译器的种种局限，通信模式简单，易于编程，但这种方法不易于借助 LabWindows/CVI 的环境对硬件进行调试。硬件设备开发阶段推荐使用前一种方案，而如果硬件设备已经定型或采用现成商用硬件，采用后一种方法更简便。

**参考文献**

[1] 周彦、戴剑伟，HLA 仿真程序设计[M]，北京：电子工业出版社，2002
[2] 刘君华、白鹏、汤晓君、郭会军，基于 LabWindows/CVI 的虚拟仪器设计[M]，北京:电子工业出版社，2003
[3] Pawletta, S.; Drewelow, W.; Pawletta, T.， HLA-based simulation within an interactive engineering environment[J]，Distributed Simulation and Real-Time Applications, 2000. (DS-RT 2000). Proceedings. Fourth IEEE International Workshop on , 24-26 Aug. 2000: 97-102
[4] National Instruments Corp.，LabWindows/LVI Programmer Reference Manual[M]，2003