文章编号：2095-4980(2016)03-0432-06

基于 VC++6.0 的泰克数字示波器应用类设计

林立杰，耿　涛，程　刚，胡志英

(中国工程物理研究院 电子工程研究所，四川 绵阳 621999)

摘　要：针对泰克 DPO4000 系列示波器特性，采用 C/C++语言编程设计示波器应用类。基于某工程项目进行分析研究，设计了适合于该示波器的应用类，此类方法主要包括示波器资源查找、示波器打开和关闭、示波器设置以及示波器数据回读等。经过实验测试，证实该类设计方法可行，具有较好的应用价值。

关键词：示波器；软件编程；类

中图分类号：TN791　　**文献标识码：**A　　**doi：**10.11805/TKYDA201603.0432

Class design for tektronix oscilloscope's application based on Visual C++ 6.0

LIN Lijie, GENG Tao, CHEN Gang, HU Zhiying

(Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang Sichuan 621999, China)

Abstract: Aiming for the features of Digital Phosphor Oscilloscope(DPO) 4 000 series, C/C++ language is adopted to program and design oscilloscope application class. Then based on the analysis of some certain project, an application class suitable to this kind of oscilloscope is designed. The class methods mainly include oscilloscope resources search, oscilloscope opened and closed, set and read oscilloscope data back etc. The design is proved to be feasible and applicable by the experimental results.

Key words: oscilloscope; soft program; class

以提高自动测试系统通用性和自动化测试程度为指导思想，通过计算机软件编程设计，运用设备互连技术把具备互连接口的现代通信仪器仪表和计算机连接起来形成的测试系统，可以实现自动测试功能，极大地提高测试效率和测试准确性。数字示波器是现代测试仪器的代表，具有波形触发、采集和波形数据分析处理等独特优点，而泰克 DPO4000 系列示波器更是以其先进的设计方法和优异的性能为广大设计开发者使用，该类型示波器具有自动测量、FFT 分析、测量统计、波形直方图和高级波形数学运算等功能[1]，能够简化被测设备的分析工作，方便查看快速变化信号细节，分析波形的基本特性；该系列示波器还提供了基于 USB(Universal Serial Bus)通用串口总线接口和基于 TCP/IP(Transmission Control Protocol/Internet Protocol)传输控制协议的以太网总线接口，这 2 种接口具有接插方便，传输速率高的特点，逐渐开始取代 VXI(VME bus Extension for Instrumentation)和 GPIB(General-Purpose Interface Bus)通用接口总线成为自动测试系统间连接的标准总线。通过这 2 种总线连接，既方便了计算机对示波器进行查询和控制操作，也方便将示波器采集的波形和分析的数据读回计算机界面进行显示。

在某工程项目中，需要在测试软件中对泰克 DPO4000 系列示波器进行设置操作，以便于采集波形数据并进行波形分析。但经过调研，发现很多资料仅仅提到如何打开示波器，或者局限于 LabView 图形化编程操作，没有专门针对这种数字示波器应用的 C/C++类，因此在该工程项目开发的基础上，在熟悉示波器编程操作函数后，将泰克 DPO4000 系列数字示波器的查询、设置、采集、分析等功能函数调用进行封装打包，编译成专供示波器应用的 C/C++类，为软件开发统一化提供共享资源。

1　物理连接

在该工程项目中，波形采集及分析功能涉及的组件主要包括 1 台工控机、2 台数字示波器和多个测试对象，整个测试系统物理连接如图 1 所示，其中工控机通过 2 个以太网端口分别与 2 台示波器的以太网端口进行连接，

收稿日期：2014-11-14；修回日期：2015-04-20

每台示波器可通过 4 个测试通道与 4 个测试对象进行连接，采集测试波形数据并进行分析。

项目中采用性能比较稳定的 Windows XP(SP3)操作系统作为系统平台，编程语言为 C/C++，Visual C++6.0(Service Pack 6)作为软件开发工具编写项目测试软件，操作者通过测试软件控制示波器的查询、设置、采集和分析工作，随后从示波器读回测试波形和数据分析结果，并将它们显示在测试软件界面，最终完成项目的波形采集和测试工作。

2　VISA 技术

经过调研发现对示波器功能的调用需要用到 VPP(VXI Plug&Play)系统联盟统一制定的 I/O 函数库 Visa32.dll 和 Visa.lib 库,该库简称为 VISA (Virtual Instrument Software Architecture)库[2]。

原来为了确保不同厂商、不同接口标准的仪器能相互兼容、相互通信,VPP 联盟提出了 VISA 虚拟仪器软件结构这一自底向上的 I/O (Input/Output)接口软件模型，VISA 的内部结构是一个先进的面向对象的结构，这一结构使得 VISA 与之前的 I/O 控制软件相比，接口无关性有很大提高。VISA 的可扩展性使它远远超出了一般的 I/O 控制软件范畴，而且由于 VISA 内部结构的灵活性，VISA 在功能和灵活性上也超过了其他 I/O 控制库。尽管 VISA 的 API 函数与其他 I/O 控制库具有类似功能,但是函数却少得多，因此 VISA 很容易被初学者掌握。另外 VISA 高度的可访问性和可配置性又使得熟练的用户可以利用 VISA 的许多独有特性，使得 VISA 的应用范围大大超过了传统的 I/O 软件。VISA 不仅为将来的仪器编程提供了许多新特性，而且兼容过去已有的仪器软件。所以 VISA 具有与仪器硬件接口无关的特性，是理想的仪器 I/O 软件。

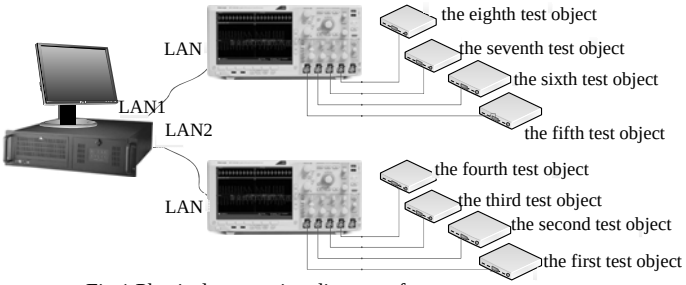

Fig.1 Physical connection diagram of test system
图 1　测试系统物理连接图

由于 VISA 采用了面向对象编程思想，且集成了当前所有仪器接口类型功能函数,使标准函数与仪器的 I/O 接口类型无关，方便程序移植，使应用开发者方便调用函数进行测试程序开发，所以本文采用 VISA 库的接口函数进行软件开发。下面以泰克公司 4054 数字示波器的应用操作为例，说明运用 VISA 资源模板对示波器的应用。工作开始前需要预先获得链接的库文件 Visa32.dll，头文件“Visa.h”和“Visatype.h”。

3　示波器应用流程

通过应用探索，发现可以将 VISA 库中的接口函数重新用 C/C++语言进行封装，形成单独的示波器查询、设置、数据采集和波形分析等功能函数[3]，然后将各功能函数按照测试步骤进行串联，设计为自动测试流程，减少应用操作，提高了测试效率。其中对示波器的操作流程如图 2 所示。

为便于理解，结合图 2，把数字示波器的编程操作分成 4 个阶段进行描述。

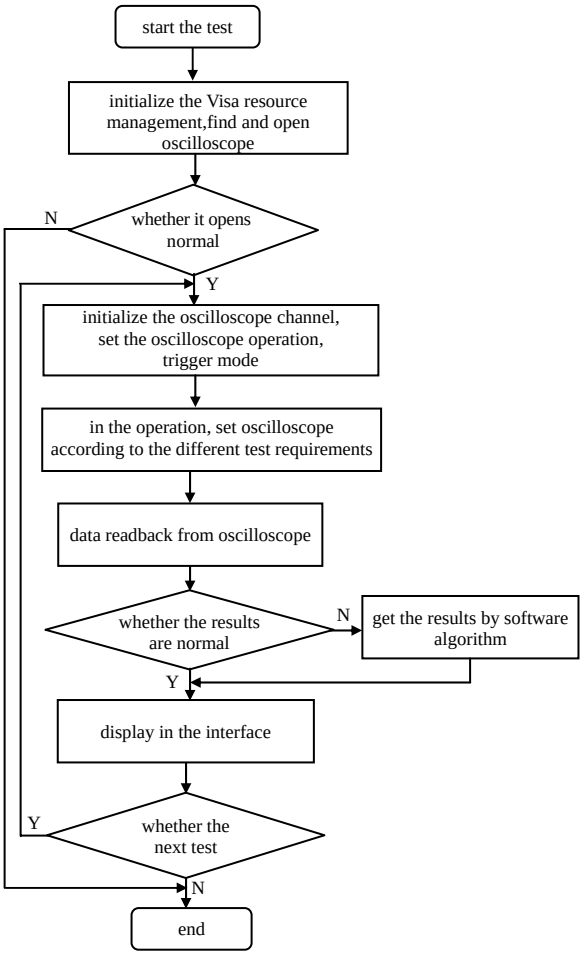

Fig.2 Flow chart of oscilloscope operation
图 2 示波器操作流程图

3.1 打开设备

在打开设备阶段，包括了寻找设备资源、打开设备的工作。通过 C/C++编程[4]，在这个阶段主要调用的函数包括 viOpenDefaultRM(初始化 VISA 资源管理)、viFindRsrc(查询 VISA 设备，进行资源定位)、viFindNext(寻找下一个资源)、ViOpen(打开特定资源)、viClose(如果没有找到所有 VISA 设备，则关闭 VISA 设备资源)，寻找和打开设备函数如下：

```
int CDPOManager::Find_GetDPOVI()
    {
    ViStatus status1 = VI_NULL;
    ViStatus status2 = VI_NULL;
    m_DPO1ST.Device_VI = VI_NULL;
    m_DPO2ST.Device_VI = VI_NULL;
    numInstrs = 0;
    if (m_DPO_rm != VI_NULL)
    {
        viClose (m_DPO_rm);
    }
    status1 = viOpenDefaultRM(&m_DPO_rm);
    if (status1 < VI_SUCCESS)
    {
        viStatusDesc(m_DPO1ST.Device_VI, status1,viStatusbuffer);
        if (m_DPO_rm != VI_NULL)
        {
            viClose(m_DPO_rm);
            m_DPO_rm = VI_NULL;
            return m_bReturnValue|0x04;
        }
    }
    else
    {
        ZeroMemory(instrDescriptor,sizeof(instrDescriptor));
        status1 = viFindRsrc(m_DPO_rm,"?*INSTR",&findList,&numInstrs,instrDescriptor);
            if (status1 < VI_SUCCESS)
        {
            viClose(m_DPO_rm);
            m_DPO_rm = VI_NULL;
            return m_bReturnValue|0x04;
        }
        else
        {
            CString strTemp = "";
            strTemp = instrDescriptor;
memcpy(&m_DPO1ST.chDeviceName_Buf,&instrDescriptor,sizeof(instrDescriptor));
            m_bGetDPOID[0] = true;
        }
        while (--numInstrs)
        {
            status1 = viFindNext (findList, instrDescriptor);
            if (status1 < VI_SUCCESS)
            {
                viClose(m_DPO_rm);
                m_DPO_rm = VI_NULL;
                return m_bReturnValue;
            }
            else
            {
                CString strTemp = "";
                strTemp = instrDescriptor;
                    memcpy(&m_DPO2ST.chDeviceName_Buf,&instrDescriptor,sizeof(instrDescriptor));
                m_bGetDPOID[1] = true;
            }
```

```
            }
            if (m_bGetDPOID[0])       //OpenFirst Device
            {
                status1=viOpen(m_DPO_rm,m_DPO1ST.chDeviceName_Buf,VI_NULL,
VI_NULL,&m_DPO1ST.Device_VI);
                if (status1< VI_SUCCESS)
                {
                    m_bReturnValue |= 0x01;
                }
            }
        if (m_bGetDPOID[1]) //Open Second Device
        {
            status2=viOpen(m_DPO_rm,m_DPO2ST.chDeviceName_Buf,VI_NULL,
VI_NULL,&m_DPO2ST.Device_VI);
            if (status2< VI_SUCCESS)
            {
               m_bReturnValue |= 0x02;
            }
        }
      }
      return m_bReturnValue;
      }
```

打开 VISA 资源后，可以调用 viWrite 函数写入“SELECT:CH1 ON”等命令打开对应示波器通道。

3.2 配置示波器通道属性

3.2.1 初始化配置

示波器设备打开正常后，需要对示波器通道进行初始化配置。单台数字示波器有 4 个通道，每个通道可以测试一个不同对象，有时针对不同测试对象需要进行不同的通道配置，需要初始化配置的内容主要包括：测试带宽、信号输入的耦合方式、波形采样位置、波形缩放比例、波形触发方式、触发范围、波形记录长度等。

3.2.2 运行时配置

针对不同的测量项目，在过程的不同阶段，也可以调用预先模块化设计的设置函数对示波器进行设置。这个阶段的设置需要对示波器的应用操作比较熟悉，需要有专业的测试基础。它的内容具有独特性，不同的测试内容要求测试项目不一样，涉及测试项目配置(如幅值测试、脉宽测试以及时间测试等)。在工程项目中，为了得到测量项值例如时间宽度、周期等，通常需要在这个过程对示波器进行光标设置、下降沿设置操作。

3.3 数据采集

3.3.1 波形数据采集

在每次数据采集和波形测试前设置“ACQuire:STATE RUN”和“TRIGger:FORCe”等命令，使示波器处于运行和触发状态；同时编程控制示波器采集信号波形；采集后将数据从 ASCII 码转换成符点数；得到测试结果；测试完毕后对示波器写入“ACQuire:STATE STOP”命令，停止示波器采集；等待下一次测试再重复设置运行。其中示波器采集数据函数代码如下：

```
bool CDPOManager::Recv_DPOData(ViSession&vi,char*chAcqDataBuf,ViUInt32 nAcqDataLen,BYTE
ubCurveID)
{
    ViStatus status = VI_NULL;
    bool bRet = false;
    ViBuf chBuf[50];
    ZeroMemory(chBuf,sizeof(chBuf));
    CString strOscChannel = _T("");
    strOscChannel.Format(_T("%d"), ubCurveID + 1);
    if (!ExecCmd(vi,_T("DATa:SOURCE CH") + strOscChannel))
    return false;
    if (!ExecCmd(vi,_T("DATa:STARt 0")))
    {
        return bRet;
    }
```

```
        if (!ExecCmd(vi,_T("DATa:STOP 9999")))
        {
            return bRet;
        }
        if (!ExecCmd(vi,_T(":WFMOutpre:ENCdg ASCII")))
        {
            return bRet;
        }
        if (!ExecCmd(vi,_T("curve?")))
        {
            return bRet;
        }
        viLock(vi, VI_EXCLUSIVE_LOCK, VI_TMO_INFINITE, NULL,NULL);
        status = viRead(vi, (ViBuf)chAcqDataBuf,sizeof(chAcqDataBuf),
    (ViPUInt32)&nAcqDataLen);
        viUnlock(vi);
        if (status < VI_SUCCESS)
        {
            viStatusDesc(vi, status, viStatusbuffer);
            fprintf(stderr, "failure: %s\n", viStatusbuffer);
            if (m_DPO_rm != VI_NULL)
            {
              viClose(m_DPO_rm);
              m_DPO_rm = VI_NULL;
            }
        }
        else
        {
            bRet = TRUE;
             Convert_CHAN_ASCII_2_floatVal(chAcqDataBuf,nAcqDataLen,ubCurveID); //将数据转换为符点数
      }
    return bRet ;
    }
```

这种方式将整个记录长度的数据全部读回，例如示波器记录长度预先设置为 10 000，那么波形数据采集将一次性读回 10 000 个点数据。

3.3.2 测量项数据采集

在运行过程中，还可以设置单独的测量项目，例如测量幅值、脉宽、时间周期等，测量项数据采集函数如下所示。

```
bool CDPOManager::GetDPOChannelData(ViSession &vi,int nNum, float &fData)
{
ViUInt32    retCnt;
bool bRet = false;
char chMeasureActName[128] ="";
char chReadDataBuf[128] = "";
ViUInt32 nReadDataLength = 0;
CString strTemp = "MEASUREment:MEAS:Value?",strChannel = "";
strChannel.Format("%d",nNum);
strTemp.Insert(16,strChannel);
sprintf(chMeasureActName,"%s",strTemp);
if (viWrite (vi, (ViBuf)chMeasureActName,strlen(chMeasureActName),&retCnt)< VI_SUCCESS)
{
    return bRet;
}
if (( viRead(vi, (ViBuf)chReadDataBuf,sizeof(chReadDataBuf), &nReadDataLength)) < VI_SUCCESS)
{
    return bRet;
}
else
{
    Convert_Single_ASCII_2_floatVal(chReadDataBuf,fData);
}
```

```
return bRet = TRUE;
}
```

3.4 关闭示波器通道和关闭示波器资源管理

当应用程序退出时，需要关闭示波器通道，关闭示波器资源管理，这些操作通常可以放在类析构函数中，退出时自动调用析构函数完成处理动作。

4　示波器应用参数接口

为方便调用示波器类完成资源查询、数据采集、数据存储的操作，需要在应用程序中预定义参数接口，该参数接口用一个结构体[5]变量表示，参数接口内容如下所示：

```
typedef struct _Oscillscope_Device
{
    ViSession Device_VI;                              //资源 VI
    char    chDeviceName_Buf[150];                    //缓冲区存放示波器名称
    char    chReadBuf[AcqDataLength * 4];             //缓冲区存放从示波器读回数据
    float fAcqDataBuf[DevceNum][AcqDataLength]; //缓冲区存放单次从示波器读取数据
    float fDPOSampleRate;                             //采样率
    BYTE    ubDPOChanNum;                             //测试通道数
}
```

通过打开设备资源和数据采集，可以获得示波器的 VI 资源、示波器编号、示波器采集数据以及设置的通道采样率等内容，方便数据的分析和存储。

5　结论

对示波器的应用操作都是在封装 VISA 接口库函数基础上进行的，通过 VC++6.0 设计数字示波器应用类时，在封装前需要先做好软件设计架构，考虑输入输出参数接口，方便软件设计统一化，促进软件设计可靠性。

封装并获得示波器应用类的.h 文件和.cpp 文件后，即可在应用程序中调用相应函数对示波器进行控制和数据采集等操作。经过多个项目的测试，应用结果良好，能够满足要求。表明通过该示波器类的应用，能够采用面向对象的编程方式对泰克 DPO4000 系列示波器进行操作，极大促进了相似软件的功能开发。

参考文献：

[1] Tektronix,Inc.. MSO4000B AND DPO4000B Series Digital Phosphor Oscilloscopes Programmer Manual[Z]. 2011.

[2] National Instruments Corporation. NI-DAQmx C Reference Help[Z]. 2014.

[3] LRINRVKRT R C. Visual C++6 Bible[M]. Beijing:Beijing Publishing House of Electronics Industry, 1999.

[4] XU Xiaogang,GAO Zaofa,WANG Xiujuan. Visual c 6.0 Entry and Improvement[M]. Beijing:Tsinghua University Press, 1999.

[5] MindView,Inc,. ECKEL Bruce,ALLISON Chuck. THINKING IN C++[M]. New Jersey:Prentice Hall, 2000.

作者简介：

林立杰(1972–)，男，安徽省潜山县人，硕士，高级工程师，主要研究方向为计算机测控技术及软件开发应用.email:easyname_llj@sina.com.

耿　涛(1976–)，男，江苏省徐州市人，学士，高级工程师，主要研究方向为高压技术测试专业.

程　刚(1972–)，男，重庆市人，硕士，高级工程师，高压技术测试专业.

胡志英(1974–)，男，重庆市人，硕士，工程师，主要研究方向为复杂系统分析及综合与仿真专业.