独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。具我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：王凤碧　　日期：2002年　3月　5日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解秘后应遵守此规定）

签名：王凤碧　　导师签名：

日期：2002 年 3 月 5 日

摘 要

本文从数据域测试对仪器的需要入手，首先论述了逻辑分析仪的发展状况及体系结构，并从虚拟仪器的角度简要阐述了 ES4541 虚拟逻辑分析仪的原理；然后介绍了本系统软件设计中所应用的面向对象程序设计的思想与 Windows 编程方法；在阐述 ES4541 逻辑分析仪系统软件组成结构的基础上详细论述了逻辑分析仪软件设计中的面板控制、存储限定字功能实现、GPIB 通讯程序设计，最后，就系统软件的调试进行了论述。

在第一章，简要论述了数字系统信号特征及其对检测的要求，论述了逻辑分析仪在数据域测试中的应用、逻辑分析仪发展状况和体系结构，阐述了 ES4541 逻辑分析仪的原理以及对系统软件的设计要求。在第二章，论述了 ES4541 逻辑分析仪系统软件设计中运用的面向对象程序设计方法、windows 编程的知识和人机界面设计原则，同时对编程工具 Borland C++Builder 作了介绍。在第三章，全面论述了逻辑分析仪系统程序设计的几个方面：程序总体结构、功能、自定义类，逻辑分析仪系统软件的设计说明，系统软件的变量传递，键盘操作的实现，波形保存、打开功能，存储限定字功能的实现。在第四章，介绍了 ES4541 逻辑分析仪实现 GPIB 程控的硬件基础，TMS9914 内部结构，GPIB 基本函数及其功能，逻辑分析仪系统的程控指令。第五章 介绍了本系统软件设计中应用的 C++Builder 集成调试环境、调试方法，以及具体的系统调试过程。

关键词： 虚拟仪器，逻辑分析仪，GPIB，系统软件，调试

Abstract

In this dissertation, first, based on the aspects of the request to the instrument in data domain test logic analyzer's development and it's architecture are introduced. And the principle of virtual logic analyzer is discussed. Second, object-oriented programming and windows programming is introduced. The design of the control function and the GPIB communication of system software. At last the dissertation deals with the debugging of the system program.

Chapter one discusses the signal characteristic and detecting requirement of digital system, and introduces the application in data domain test, the development of logic analyzer and the system architecture. The principle of the ES4541 logic analyzer is also introduced in this chapter. Chapter two discusses the object-oriented programming method 、the knowledge in windows programming which are applied in system software design of logic analyzer and the interface between the users and the computers. Then it introduces the used programming software—Borland C++ Builder. Chapter three discusses aspects of software design of ES4541 logic analyzer including the system structure, the system function, the user-defined class, the programming of waveform analysis and displaying. It elaborates the realization of the storing and the opening of waveform, the realization of the pre-defined word trigging and the document of system software. Chapter four detailly discusses the GPIB's control programming. Chapter five introduces the debugging environment and method of C++ Builder and introduces the system software's debugging and error correcting.

Key words: virtual instrument, logic analyzer, GPIB, system software,debugging

第一章 绪论

在数字系统，特别是在数字计算机系统的研制、调试和故障诊断过程中，由模拟系统的时域和频域分析发展起来的传统测试方法与测试仪器往往难以实现。随着大规模集成电路、DSP 技术和微型计算机的发展，现代数字系统已逐步微机化。一方面使系统的能力大为提高，能够完成许多复杂的任务；另一方面传统的检测设备(示波器)已不能有效地检测和分析数字系统，特别是微机系统。这是因为数字系统的数据传输是按空间分布多码位的方式进行的，这些码位组成一定格式的数据，传输的数据流是离散时间为自变量的波形。重点是考察信号高于或低于某一门限电平值，以及这些数字信号与系统时间之间的相对关系。20 世纪 60 年代后期，随着数据域测试领域的开拓，逻辑分析仪作为数据域测试仪器中最有用、最有代表性的一种仪器，近十几年来，品种日新月异，性能与功能日益完善。因此，结合我国国情，利用虚拟个人仪器的概念，我们研制了以微型计算机为基础的，性价比较高的 ES4541 型逻辑分析仪。

1.1 逻辑分析仪与数据域测试

1.1.1 数字系统信号特征及其对检测的要求

典型的微机系统可划分为三个主要部分：CPU、I/O 和外部设备。CPU 总线包括地址总线、数据总线和控制总线。地址总线和数据总线与系统时钟是同步的，常见的故障有算法错误、丢失程序、执行时间问题、效率问题及数据相关错误。控制总线是异步的，常见的故障是时序出错，它对毛刺、干扰是很敏感的。I/O 总线进行异步或同步的多路数据传输。由于各种原因，I/O 总线可能较多地遇到诸如竞争条件、噪声尖峰脉冲、毛刺等问题。至于外设部分，主要是数据传输错误、并/串问题和 A/D、D/A 转换问题。

因此，数字系统信号的特征归纳如下：

(1) 数字信息几乎都是多位传输的。

(2) 许多信号仅发生一次。有些信号虽然重复发生，但是非周期性的；

(3) 数字信息常伴有竞争和冒险现象发生；

(4) 数字系统常由硬件和软件构成，其数字信息互相穿插，互相影响，难以区分；

(5) 数字信息工作速率变化范围大(如高速运行的主机和低速的外围设备)。

上述这些特点决定了对检测的基本要求：

(1) 跟踪与分析状态数据流。

(2) 在总线的数据流上，设置一个观察参考点，它是一个布尔表达式所对应的唯一数据字即触发字。

(3) 为了分析异步总线，要求能分析信号状态之间的时间关系。

(4) 来自系统内部和外界的干扰及毛刺常引起硬件出错，这样就需要捕捉干扰或毛

刺，并把它们显示出来。

1.1.2 逻辑分析仪在数据域测试中的应用

逻辑分析仪是一种主要的通用数据域测试仪器。逻辑分析仪在计算机、自动测试系统、智能仪器、数字通讯以及自动控制等数字系统中，用于硬件逻辑和程序软件的研究、分析、测试和故障诊断，为各种微机系统、数字系统等智能化设备的新产品开发提供测试手段，是新产品开发和系统维护必不可少的工具。逻辑分析仪已成为目前国际上最通用的电子测量仪器之一。

为了满足数据域的检测要求，逻辑分析仪一般有以下特点：

(1)足够多的输入通道。为了适应微机总线结构而需要多通道。通道数越多，所能检测的数据信息量越大，逻辑分析仪的功能就越强。

(2)多种触发方式。逻辑分析仪应该具有灵活准确的触发能力，它可以在很长的数据流中，对所观察的那部分信息作出准确定位，从而捕获对分析有意义的信息。在硬件分析中，它可以有效地检测和显示系统的运行状态；在软件分析中，它可以跟踪系统运行中的任意一段程序。

(3)具有足够的存储深度。逻辑分析仪内部具有高速存储器，因此它能快速地将采集数据进行存储。存储器决定了获取数据的多少，存储深度越深，采集数据就越多，使逻辑分析仪能够观察单次和随机性故障。

(4)具有负的延迟能力。逻辑分析仪的内部存储器可存储触发前的信息，可显示出相对于触发点为负延迟的数据。

(5)灵活而直观的显示方式和可靠的毛刺检测能力。

1.2 ES4541 逻辑分析仪体系结构

从功能上分，逻辑分析仪可分为逻辑状态分析仪(Logic State Analyzer)和逻辑定时分析仪(Logic Timing Analyzer)。它们的结构基本类似，多数情况下是做到一起的。

ES4541 逻辑分析仪主要由数据捕获和数据处理两部分组成。数据捕获部分用来捕获、存储要观察的数据，其中数据输入部分将各通道的输入变换成相应的数据流；而触发产生部分则根据数据捕获方式，在数据流中搜索特定的数据字，当搜索到特定的数据字时，就产生触发信号去控制数据存储器开始存储有效数据或停止存储数据，以便将数据流进行分块(数据窗口)。数据显示部分则将存储在存储器里的有效数据以多种显示方式显示出来，以便对捕获的数据进行分析。整个系统的运行，都是在外时钟(同步时钟)或内时钟(异步时钟)的作用下实现的。

从 ES4541 逻辑分析仪的系统结构中，可以清楚看出，整个仪器的控制和管理，数据处理以及数据显示都可方便地由微机来完成。因此，其系统硬件的设计主要集中在高速数据捕获及其与微机的接口；而软件设计主要在系统管理、数据的后处理及数据显示。

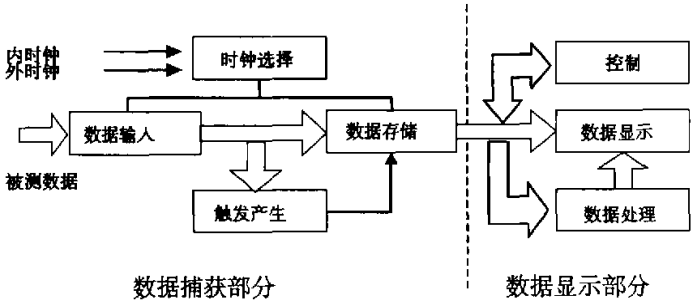


图 1-1. ES4541 逻辑分析仪基本结构图

1.3 ES4541 逻辑分析仪的实现原理

1.3.1 虚拟逻辑分析仪

逻辑分析仪与个人计算机相结合，成为基于个人计算机的智能仪器，这便是虚拟逻辑分析仪。逻辑分析仪与个人计算机结合是一个新的发展方向。两者的结合扩展了逻辑分析仪的分析能力与计算能力，降低了成本，而且使仪器的通用性增强。在逻辑分析仪中占很大比重的控制电路、显示电路、指示电路等功能都由计算机完成。逻辑分析仪与微机相结合的优点为：

★性能提高

一般情况下，仪器取数后，以适当形式把数据显示出来，供用户分析。当逻辑分析仪与微机结合之后，利用微机丰富的软件，这些分析工作可由微机完成，包括通过数据滤波提取对分析有用的数据及软件性能分析等。

★成本降低

逻辑分析仪的功能增强，特别是加强分析能力，常常是通过微机软件来实现的。在这种情况下，功能的提高并没有增加硬件的成本。相反，因部分硬件功能软化而使硬件本身大为简化。故这种逻辑分析仪的价格较低而功能全。

★使用简便

微机灵活的编程能力及键盘与显示终端的交互操作方式，简化了逻辑分析仪的使用。用户可利用 Windows 的联机帮助，学会逻辑分析仪各种功能的操作。

★功能易于扩展

逻辑分析仪通常都有多块功能插板，增加模块插板，对于增加数据通道和实现多

种分析方式很方便。

基于上述优点，特别是高性价比，决定了逻辑分析仪与微机结合的方案有广阔的前途，ES4541 逻辑分析仪的研制就采用了此技术方案。

1.3.2 ES4541 逻辑分析仪原理

图 1-2 示出了 PC 环境下的虚拟逻辑分析仪设计与实现原理。

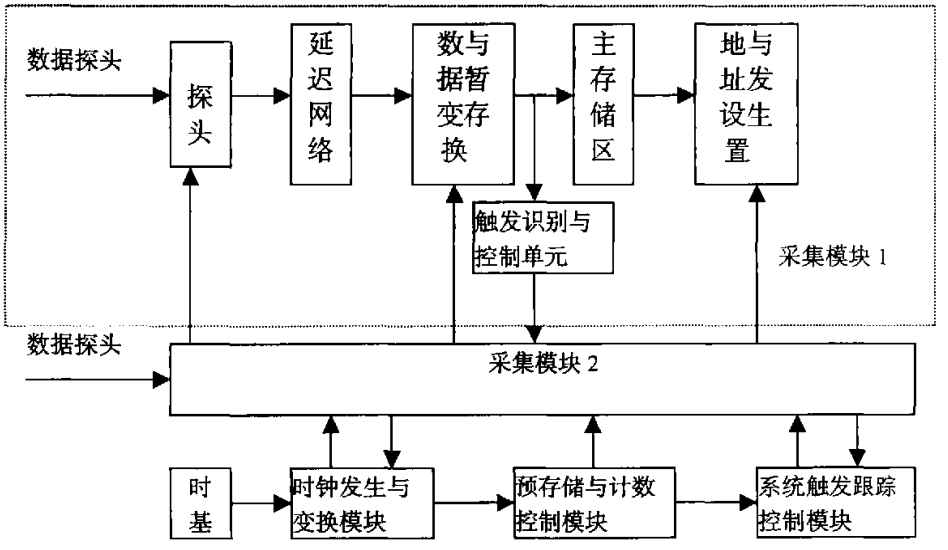


图1-2．系统构成原理图

由图 1-2 所示的系统总体构成原理框图知，虚拟逻辑分析仪主要包括数据采集、探头、触发跟踪、时序变换与生成等部分。

该系统输入、存储及控制部分有 2 个模块构成，其中 1-2 为输入采集模块，采用完全相同的功能结构，每个模块有 32 个数据通道，第 2 个模块附加了时钟输入与输出功能。采用该结构的主要原因，一是避免主采集板过大，元件过密造成散热方面的困难，二是系统结构灵活，可以根据需要选 32，64 路组态方式。

系统数据探头与被测对象测试点连接，引入被测信号，经延迟网络后进入数据变换与暂存模块，延迟网络的作用是用来满足高速状态分析时，时钟通道的相对零延迟。

时钟探头则通过外时钟的电平变换引入时钟发生电路，产生逻辑状态用的各种时钟信号实现逻辑状态分析的各种触发和跟踪方式。触发控制与识别部分则是完成对外部数据流的实时监测，确定数据跟踪窗口和限定存储条件。

1.3.3 ES4541 逻辑分析仪系统软件设计要求

根据前述，ES4541 逻辑分析仪的软件设计主要在系统管理、数据的后处理及数据显

示。由于 ES4541 虚拟逻辑分析仪采用了与个人计算机相结合的方式，这使得控制软件的设计可以充分利用个人计算机丰富的软件资源。在个人计算机上利用图像界面操作系统 Windows 以及以 Windows 操作系统为基础的可视化程序设计平台，如 C++Builder，可以快速地开发出界面美观，操作方便的用户程序，同时也便于将面向对象的优秀编程方法应用到程序当中。

第二章 ES4541 逻辑分析仪系统软件的基本设计思想

面向对象程序设计方法是目前最流行的程序设计方法，它是在Windows编程基础上发展起来的。

2.1 面向对象编程思想

结构化程序设计把现实对象模型化，它通过定义数据类型并且在松散相关程序层中使用它们来实现，而这些程序又是用来操作数据类型中的信息的。数据类型和相关程序间的相当松散的关系使得编程效率很低，尤其是当用于开发大型软件项目时，用于操作数据类型的程序编写是永无止境的。

面向对象程序设计提供了一种新的看问题的方法。面向对象编程的核心是对象，它是面向对象的支柱之一。每个对象都属于一个对象类，这就是面向对象的第二大支柱—类。换句话说，一个类就是一组非常相似的对象。面向对象编程的第三大和第四大支柱是继承性和多态性。

类：对象家族值

面向对象程序设计把对象分类升华为类。每一个类描述其对象的特性与操作。

一个类就是能共享相同特性与操作的一类对象。

对象

对象是类的一个实例。每一对象都有自己的状态，此状态也许和另一对象的状态相合，也可能不合。

对象是类的实实在在的例子。

方法和消息

所谓面向对象程序设计就是人们和对象联系并且通过给其发送消息来控制其状态。消息会告诉对象该做些什么。例如，当我们改变电视频道时，通过使用频道选择钮来给电视对象发送要换台的消息。我们的电视机作为一个对象，通过选择和执行一个方法来给消息以响应。就电视机这个例子来说，换台使其电路接收到一个不同的代表我们所选频道的信号。

就面向对象程序设计来说，方法会告诉对象怎样对消息作出反应。图 2-4 描述了我们发送给频道旋钮的消息以及旋钮是如何选择合适的方法的。

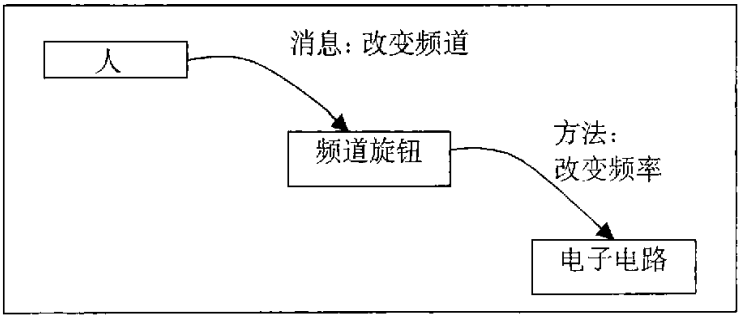


图 2-1 给频道旋钮发送一个换台消息

我们可以把消息作为相对抽象的命令或请求，消息也许包含我们可以看作是附加指令的参数。对象通过选择和执行正确的方法来对消息作出反应。被执行的方法完成所有困难的细节工作。

消息告诉对象该作什么。方法告诉对象怎样对消息作出反应。

还是拿逻辑分析仪系统软件来说吧，当我们使用逻辑分析仪时，我们给逻辑分析仪对象发送不同种类的消息。我们可以通过按键盘上不同的键来生成和发送消息，也可以通过单击鼠标按钮来实现。

逻辑分析仪软件把不同的消息译成方法，这些方法对菜单选择命令、键盘输入以及鼠标按钮击打作出反应。举例说明，当我们借助于设置限定字命令发一个限定字设置消息时，逻辑分析仪软件会援引“存储限定字设置”的那段代码。其结果是逻辑分析仪软件显示“存储限定字设置”窗口，此窗口允许指定有无限定字、限定字个数、预先设置每一个限定字等。

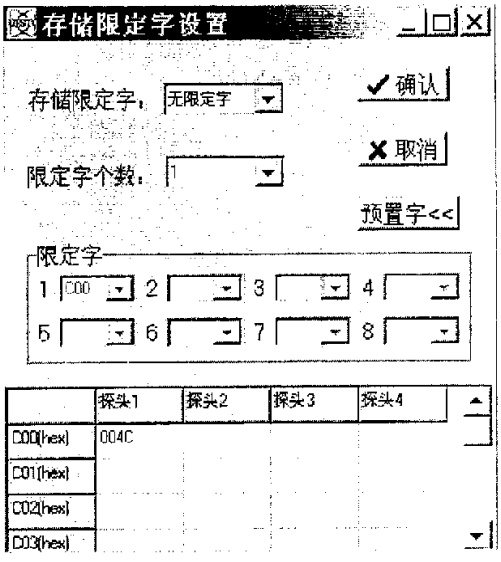


图 2-2 存储限定字设置窗口

继承性

继承性是面向对象程序设计的精华部分，因为它提供了类的再使用。面向对象程序设计允许你声明一个新类，它是现存类的后代。后代类继承其父类的属性和操作。后代类也定义新的属性和操作并且忽略那些不符合要求的遗传操作。每一后代类都代表其父类精练的产物。

面向对象程序设计允许你生成对象层。对象层的根被称之为基类。父类的双类被称为祖先类。

多态性

多态性是 OOP 的一个难点，它意味着不同的形态。多态性是面向对象程序设计的一个非常重要的特性，它支持同一层中类之间的统一响应。这并不是说不同的类必须统一行动，并且提供统一(也可以叫作持续)的响应。恰恰相反，层次中的类可以通过执行稍微不同的任务来自由支持同一响应。

举电视机这个例子。TV-01，TV-02 以及 TV-03 模型各自有一个音量旋钮，并且提供了改变音量的方式。假定你正在本地的一个电器店选购这些模型，电视模型的音量调节应该是平滑的。因此不管你检测哪一个电视模型，电视模型都提供统一的响应。每个电视模型最好使用不同电路来控制电视音量，可只要打开音量旋钮，上述三个电视模型都会产生同样的结果这就是多态行为。因此，多态性确保类层中的类实例对所支持的类层次全局的消息提供统一的响应。

目前，支持面向对象编程的语言主要有 smalltalk、C++、Java 等。Java 语言采用虚拟机机制，效率较低而 smalltalk 应用面比较窄，C++继承了 C 语言的效率高及应用面广的优点，故我们选择了 C++语言。

2.2 WINDOWS编程特点

Windows是微软公司80年代推出的一个操作系统，具有以下几个方面的特点：

·多任务操作系统

Windows操作系统是一个多任务的操作系统。在同一时刻，计算机操作系统中可以有多个应用程序在协同地运行。

·图形界面

在多任务的WINDOWS操作系统中用户通过计算机屏幕使用鼠标和键盘来与应用程序进行信息交换。为了实现用户和多个应用程序进行信息交换，在计算机的屏幕上，每一个应用程序均需要一个窗口来标识。窗口是计算机屏幕上的一个矩形区域，包含有主菜单、控件、滚动条。

·事件驱动

在同一时刻，计算机操作系统中可以有多个应用程序在协同运行，因此计算机系统的硬件和软件资源不可能由某一个应用程序独占，必须是所有应用程序来共享整个计算

机的资源。外部产生的输入事件如键盘事件、鼠标事件和定时器事件都送到操作系统的消息队列中去，操作系统再把这些消息发送到各个应用程序的消息队列中去，各个应用程序在自己的消息队列中获得这些消息，并对此产生响应和处理。

WINDOWS应用程序除完成自己特定的功能外，为了能与用户进行交互，还必须完成下面两方面的工作：

- WINDOWS应用程序需要从操作系统的消息队列中获取消息。
- 在窗口过程中对获取的消息进行响应和处理。

由此可见，基于WINDOWS的高级编程工具(如C++ Bulider)为编程者完成了建立消息循环和从操作系统消息队列中获取消息的编程。这样就大大简化了WINDOWS程序设计，使编程者可以集中精力完成程序的特定功能。

Borland C++ Builder是Borland公司最新推出的功能强大、并已彻底解决了“千年虫”问题的应用程序开发工具。该开发工具具有像VB一样易使用的优点，同时又是C++语言，适用于众多使用过C++语言的程序员。

2.3 Borland C++ Bulider 简介

2.3.1 Borland C++ Builder的产生

Borland C++ Builder是Borland公司新一代面向对象、可视化的快速应用程序开发环境(RAD: Rapid Application Development)，它运行在Windows95或Windows NT操作系统上。使用C++ Builder可以开发通用的或基于客户/服务器模式的32位Windows应用程序。

Borland公司的Borland C++Builder是传统的C++开发工具的自然发展，它是第三代C++应用程序集成开发环境。它不仅继承了传统C++应用程序开发工具高效和低层硬件控制能力的特点，同时通过可视化构件类库(VCL)所提供的构件，使得此工具有快速和真正可视化的特点。

在Borland C++ Builder中，应用程序设计和实现的基本单元称为构件，构件是一个可视化的软件单元，它可以直接放置到开发环境中去，多个这样的构件协同完成应用程序的各项功能。在Builder C++ Borland集成开发环境中，集成了130个各种功能的构件，这些构件基本上覆盖了应用程序开发的各个方面，如：基本应用程序主窗口、菜单、菜单项、工具栏、状态栏、通用对话框、数据库、Internet、定时器等。以此种方式进行应用程序的开发具有快速和可视化的特点，减少应用程序开发的时间，提高程序开发人员的工作效率，适应了当前软件市场快速软件系统开发的需求。

Borland C++ Builder同时也是一个完整的软件开发环境，它提供了完整的用于软件设计、编程、测试、调试跟踪的工具。特别的，为了适应团体协同进行大型客户/服务器模式软件的开发，在Borland C++ Builder集成开发环境中也附带了应用程序版本

控制软件(PVCS)，用于在团体应用程序开发过程中，跟踪和管理软件系统的开发。

总之，采用构件的思想进行Windows应用程序编程彻底改变了传统Windows应用程序编程的方法。应用程序编程人员无需深入了解操作系统的运行机制，就可以进行编程。在应用程序编程时，编程人员只需要把各种功能的构件放置在一起，通过非常少的手动编程就可以完成应用程序的整体功能。

2.3.2 Borland C++ Builder的特点

Borland C++ Builder集成开发环境具有下面几个方面的特点：

- 真正面向程序设计全过程的可视化程序开发环境，改变了程序开发的编程方式，开发人员通过非常少的手工编程，即可实现和完成复杂的功能；
- 采用符合ANSI标准的C++编译器；
- 集成开发环境中，内置了功能强大的软件开发工具软件，如对象浏览器、表单编辑器、构件调色板、工程项目管理器、对象存储器等；
- 集成开发环境通过工程项目管理器对应用程序开发过程中的各个方面进行组织和管理；
- 通过集成开发环境中的对象存储器，最大限度地实现应用程序开发过程中不同层次的重用；
- 集成开发环境内置130个各种功能构件，帮助用户进行应用程序的开发；
- 集成开发环境提供数据库支持，帮助用户开发一般的或多层客户/服务器模式的数据库应用程序；
- 在集成开发环境中通过WebBroker、WebDispatcher、WebBridge等25个Internet构件来帮助用户开发Internet应用程序。

正是由于Borland C++ Builder具有以上一些特点，才在逻辑分析仪系统软件的设计中选择它作为开发工具。

2.3.3 Borland C++ Builder的编程模式

在Borland C++ Builder集成开发环境中，设计和实现应用程序的基本单元是构件，它是一个具有特定功能的软件模块。和编程人员密切相关的构件有Applicatuin构件、主窗口表单构件和可以放置在表单构件中的各种构件如按钮、编辑控件、列表控件、对话框、菜单等。

应用程序开始运行时，首先创建Application对象。Application对象负责创建应用程序窗口表单对象，接收操作系统的消息，并将其消息发送到应用程序主窗口表单对象中。应用程序主窗口表单构件中，一般包含了多个可视化或不可视化的构件。从Application对象接收到消息后，表单对象根据消息类型确定由表单中的哪一个构件对象处理此消息，并将此消息发送给相应构件对象进行处理。放置在表单中的构件，是一

个相对独立的且具有特定功能的软件模块，它可以对接收到的消息进行缺省响应和处理。如果编程人员希望对某些事件进行处理，可以通过集成开发环境的对象浏览器来选择此事件，并在一个单元文件中编写出对此事件的相应和处理代码。

总之，和传统的Windows应用程序开发模式完全不同，开发Borland C++ Builder应用程序需要下面两个阶段的工作：

第一阶段：根据应用程序的要求，以可视化的方式设计Borland C++ Builder应用程序的窗口界面。此阶段不需要任何编程代码。具体步骤：

- 创建应用程序主窗口表单构件，设置表单构件的属性参数；
- 在应用程序表单构件中，放置可视化或非可视化的构件，并设置构件的属性参数。

第二阶段：根据应用程序功能的要求，对表单构件及其包含的构件的某些特定事件进行相应和处理。在此部分中，需要一定的手动编程。

2.3.4 Borland C++ Builder的构件

构件是 Borland C++ Builder 进行程序设计和实现的基本建筑单元。利用 Borland C++ Builder 的构件进行程序设计，就如同从计算机市场上购买计算机主板、处理器、内存条、显示器、显示卡、CDROM、软驱、声卡来自己组装计算机一样。构件就如同硬件系统中的处理器、内存条、显示器等一样，它是一个完成特定功能的软件单元，可以以可视化的形式放置到 Borland C++ Builder 的表单编辑器中去，通过简单的编程，就可以和此表单中的其他构件协同完成应用程序的功能。

构件就其本质而言是一个类。和传统 C++语言中的类相比，它具有如下特点： C++ Builder 中的构件是可视化构件库(VCL)中 Tcomponent 的一个派生类。在 Tcomponent 类中定义了构件的基本功能，如构件具有放置到构件栏中的能力，通过对象浏览器可以设置它的属性参数，因此 Borland C++ Builder 中所有的构件都继承此种能力。

构件由下面三个部分组成：

- 属性；
- 事件；
- 方法；

构件的属性表明了构件的形状或性质。在应用程序运行时，构件的属性确定了构件对象的外在形式。如表单构件的 Caption 属性决定了主窗口的标题栏中所显示的标题；构件的属性根据其工作方式的不同分为 3 类：设计属性是在应用程序设计阶段就可以显示和发挥作用的属性；运行时属性是在应用程序运行时才得以发挥作用的属性；构件的只读属性不能被更改。就其本质而言，构件是类，而属性是类中的数据成员。构件的事件响应是指构件在接收到激励后所有执行的动作。激励可以是外部的，如使用鼠标单击按钮构件，亦可以是内部的，如定时器时间事件。就其本质而言，构件的事件响应是类中的成员函数。构件的方法就是构件所具有的功能。通过调用构件的方法，可以使构件

完成特定的功能。

构件的事件响应是指构件在接收到激励后所有执行的动作。激励可以是外部的，如使用鼠标单击按钮构件；也可以是内部的，如定时器时间到事件。就其本质而言，构件的事件响应是类中的成员函数。

构件的方法就是构件所具有的功能。通过调用构件的方法，可以使构件完成特定的功能。

2.4 ES4541 逻辑分析仪人机界面风格

2.4.1 人机界面设计过程

人机界面的设计过程可分为下面几个步骤：

(1)创建系统功能的外部模型；

(2)确定为完成此系统功能人和计算机应分别完成的任务；

(3)考虑界面设计中的典型问题；

(4)借助 CASE 工具构造界面原型；

(5)真正实现设计模型；

(6)评估界面质量。

任务分析与建模：

逐步求精和面向对象分析等技术同样适用于任务分析。逐步求精技术可把任务不断划分为子任务，直至对每个任务的要求都十分清楚。而采用面向对象分析技术可识别出与应用有关的所有客观的对象以及与对象关联的动作。

一旦每个任务或动作定义清楚，界面设计即可开始。界面设计首先要完成下列工作：

(1)确定任务的目标和含义；

(2)将每个目标/含义映射为一系列特定动作；

(3)说明这些动作将来在界面执行的顺序；

(4)指明各个系统状态，即上述各动作序列中每个动作在界面上执行时，界面呈现的形式；

(5)定义状态机制，即便于用户修改系统状态的一些设置和操作；

(6)说明控制机制怎样作用于系统状态；

(7)指明用户应怎样根据界面上反映出的信息解释系统的状态。

2.4.2 界面设计的一般问题

设计任何一个人机界面，一般必须考虑系统响应时间、用户求助机制、错误信息处理和命令方式四个方面。

系统响应时间指当用户执行了某个控制动作后(例如，按回车键，点击鼠标器等)，系统作出反应的时间(指输出所期望的信息或执行对应的动作)。系统响应时间过长是交互式系统中用户抱怨最多的问题，当几个应用系统分时运行时尤甚。除了响应时间的绝对长短外，用户对不同命令在响应时间上的差别亦很在意，若过于悬殊，用户将难以接受。

任何错误和警告信息对用户不啻是“坏消息”，若此类信息不是自明的，用户接到后只能图增烦恼。试想，当用户看到如下一行显示：

SEVERE SYSTEM FAILURE------14A

一定会满腹牢骚。原因是尽管能从其他什么地方查出 14A 的含义，可设计者为什么不在此指明呢？一般来说，出错信息应选用用户明了、含义准确的术语描述，同时还应尽可能提供一些错误恢复的建议，此外，显示出错信息时，若辅以听觉、视觉刺激，则效果更佳。

键盘命令曾经一度是用户与软件系统之间最通用的交互方式，随着面向窗口的点选界面的出现，键盘命令虽不再是唯一交互形式，但许多有经验的熟练的软件人员仍喜爱这一方式，更多的情形是菜单与键盘命令并存，供用户选用。

2.4.3 ES4541 逻辑分析仪人机界面实现的原则考虑

人机界面设计得好坏与设计者的经验有直接的关系，本节从一般可交互性、信息显示和数据输入三个方面简单介绍一些界面设计的经验。

1.一般可交互性

提高可交互性的措施有：

(1)在同一用户界面中，所有的菜单选择、命令输入、数据显示和其他功能应始终保持同一种形式和风格。

(2)通过向用户提供视觉和听觉上的反馈，保持用户与界面间的双向通信。

(3)所有可能造成损害的动作，坚持要求用户确认，例如，提问“你肯定••？”

(4)对大多数动作应允许恢复(UNDO)；

(5)尽量减少用户记忆上的负担；

(6)提高对话、移动和思考的效率，即最大可能地减少击键次数，缩短鼠标移动的距离，避免使用户产生无所适从的感觉；

(7)错时采取宽容的态度；

(8)按功能分类组织界面上的活动；

(9)提供上下文敏感的求助系统；

(10)用简短的动词和动词短语提示命令。

2.信息显示

若在人机界面上给出的信息不完全、有二义性或难以理解，用户肯定不满意。信息显示的形式和方式可以有多种多样，下面是一些带有普遍指导意义的原则：

(1) 仅显示与当前上下文有关的信息；

(2) 避免因数据过于费解造成用户烦恼；

(3) 采用统一的标号、约定俗成的缩写和预先定义好的颜色；

(4) 允许用户对可视环境进行维护，如放大、缩小图象；

(5) 只显示有意义的出错信息；

(6) 用大、小写，缩进和按意群分组等方法提高可理解性；

(7) 用窗口(在适合的情况下)分隔不同种类的信息；

(8) 用“类比”手法，生动形象地表示信息；

(9) 合理划分并高效使用显示屏。

3.数据输入

用户与系统交互的大部分时间用于键入命令，提供数据或系统要求的其他输入信息。目前，键盘仍为最常用的输入设备，但鼠标、数字仪、甚至语言识别系统正迅速成为替代品。关于数据输入，应注意：

(1) 尽量减少用户输入的动作；

(2) 保证信息显示方式与数据输入方式的协调一致；

(3) 许用户定做输入格式；

(4) 用灵活多样的交互方式，允许用户自选输入方式；

(5) 隐藏当前状态下不可选用的命令；

(6) 为所有输入动作提供帮助信息；

(7) 允许用户控制交互过程；

(8) 删除所有无实现意义的输入。

第三章 ES4541 逻辑分析仪系统本控软件的设计

3.1 概述

3.1.1 系统软件组成及功能

逻辑分析仪程序设计采用事件驱动机制，程序共可分为五大功能模块：自检初始化模块，主控制台，功能设置模块，数据采集模块以及数据列表、波形显示/定时分析和反汇编模块。如图 3-1. 所示

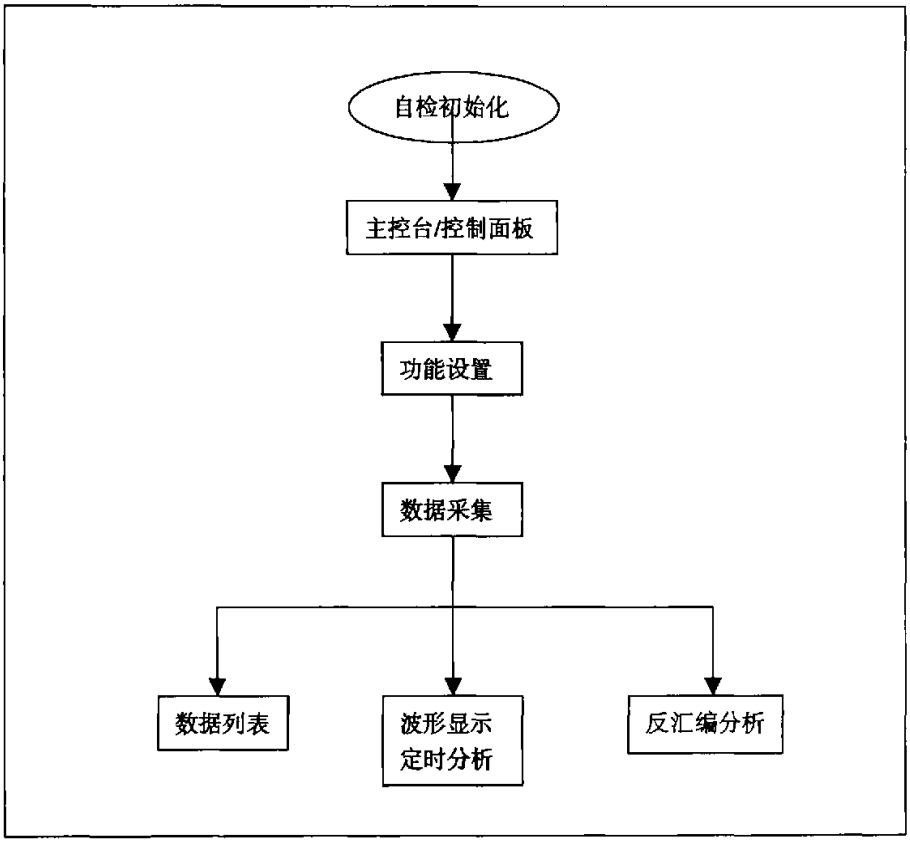


图 3-1. 逻辑分析仪程序结构

★运行启动

双击或打开逻辑分析仪可执行文件，程序显示如图 3-2.所示，同时进行逻辑分析仪自检，如果发现错误将显示错误提示信息。

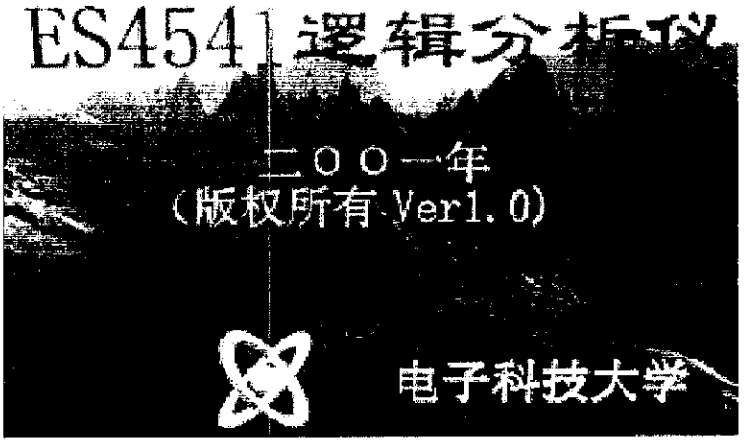


图 3-2. 逻辑分析仪初始化界面

★设置

逻辑分析仪自检通过，并成功进行变量初始化后，自动进入逻辑分析仪控制面板，如图 3-3.所示。

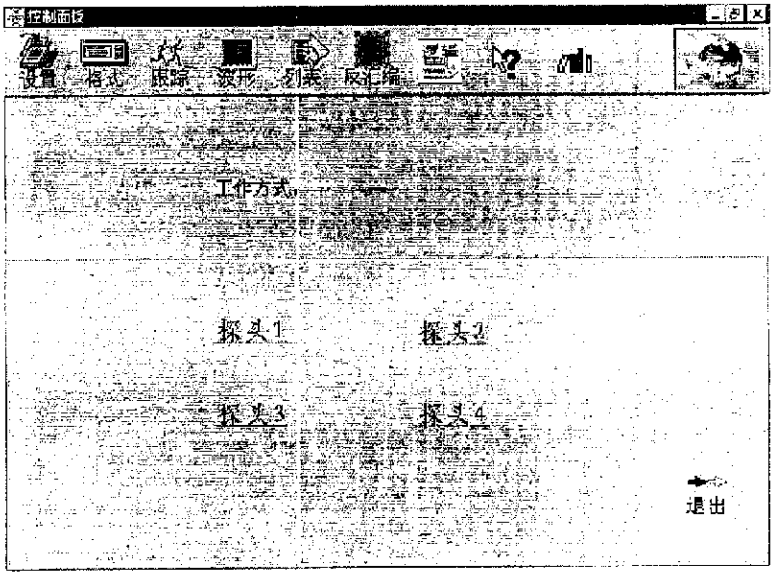


图 3-3. 逻辑分析仪控制面板

逻辑分析仪控制面板是逻辑分析仪的主控中心，通过控制面板来设置逻辑分析仪的各种参数，并控制逻辑分析仪的运行。退出控制面板将退出逻辑分析仪程序。

控制面板工具栏有七个按钮，将光标停留在按钮的上面将会提示其功能。从左至此

右分别是：探头格式设置，跟踪方式设置，波形显示，状态列表，反汇编代码，状态设置，关于逻辑分析仪。

在进行测试之前，首先点击［设置］按钮进行逻辑分析仪工作方式和使用探头的设置。如图 3-4.所示。

选择定时分析仪或状态分析仪，如选择了定时分析仪工作方式，还应在组合框下拉列表中选择分析仪采样频率，同时选定需要使用的探头号(选定探头号后，该探头的十六通道默认为完全使用，但可在探头格式设置中修改)。按确认保存选择退回控制面板，退回控制面板后，将看到刚才所做的选择，在控制面板上有标示。按取消将不作保存而回到控制面板。

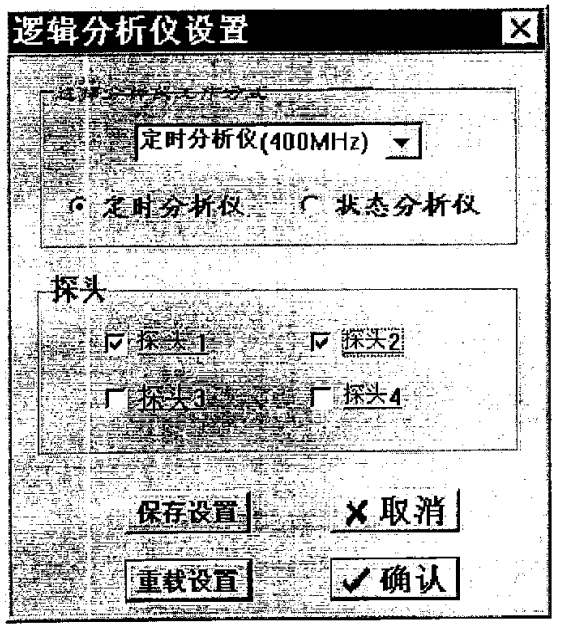


图 3-4. 逻辑分析仪模式设置

然后，点击［格式］按钮进行探头格式设置，如图 3-5 所示。可以分别对各探头设置其别名、有效通道、时钟极性、时钟触发沿。别名有助于区分各通道数据的性质，此处设定的别名会在波形显示窗口中标示各探头；通道被标记为 0～F，其中有效通道以“＊”表示，无效通道以“－”表示。被标示为无效的通道在波形显示时将不被显示。在设置好所有探头后，按［确认］按钮，保存设置并返回控制面板。

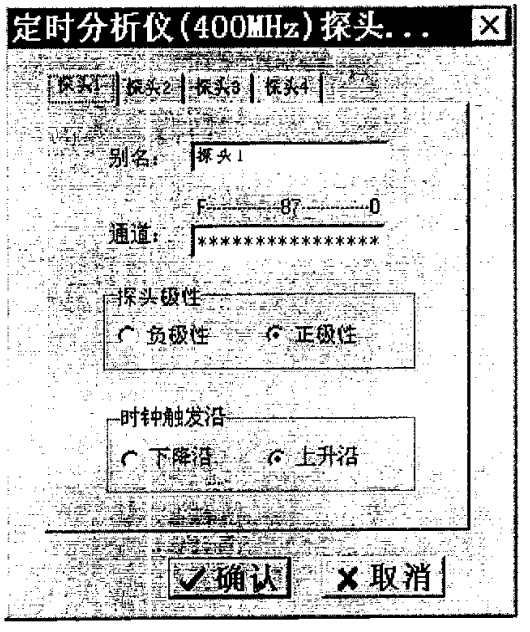


图 3-5. 逻辑分析仪探头格式设置

其次，单击［跟踪］按钮进行跟踪方式设置。如图 3-6.所示，可设置跟踪方式为起始/终止/随机触发方式之一，随机触发时还可设置为单次或重复采样。如果需要设定存储限定字或触发限定字，则应先点击“触发字”或“限定字”按钮，可弹出触发字设置窗口或限定字设置窗口以设置触发字或限定字(如图 3-7、3-8 所示)。最多可以设置二十个预置字。预置字的设定可以选择十六进制、十进制、八进制、二进制和 ASIC 码的形式，通过《码制》菜单或右键菜单进行选择。

这里特别强调一点：延时数的范围是“0-32000”事件计数的范围是“0-32000”如果超出这个范围在按确定按钮时将会弹出提示信息。

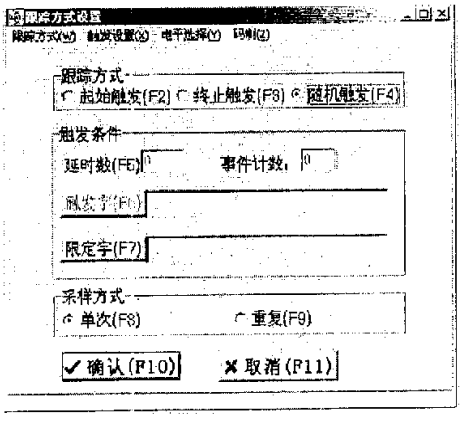


图 3-6. 逻辑分析仪跟踪方式设置

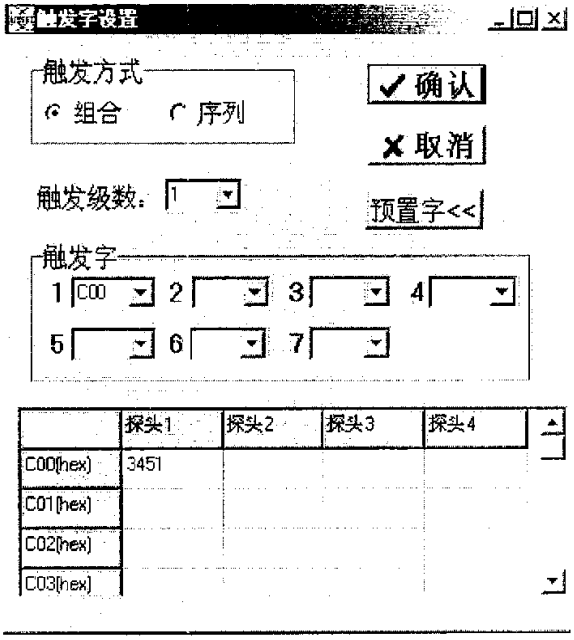


图 3-7. 逻辑分析仪跟踪方式触发字设置

如果需要设定触发字，点击［触发字］按钮设定触发方式和触发字。根据要求设定触发方式为组合或序列方式，选定触发级数和各级触发字后按［确认］便保存设置返回上级窗体。如图 3-7 所示。

这里也特别强调一点：当用户选择了触发字后，如果忘了设置触发字将提示“触发字还未设置”的提示信息

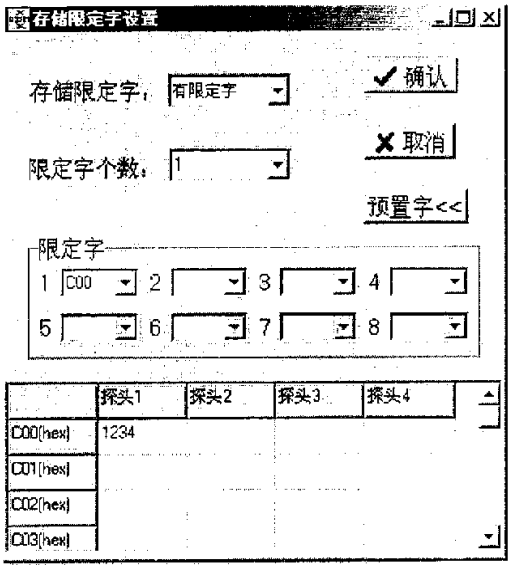


图 3-8. 逻辑分析仪跟踪方式存储限定字设置

如果需要设定存储限定字［限定字］按钮设定限定字方式和限定字。根据要求设定

限定方式为组合或序列方式，选定限定级数和各级限定字后按［确认］便保存设置返回上级窗体。如图 3-8 所示。

同样，当用户选择了限定字后，如果忘了设置限定字将提示“限定字还未设置”的提示信息

★采集数据

设置好各项逻辑分析仪参数后，点击控制面板工具栏上的[波形]按钮，弹出波形分析／显示窗口，通过波形分析／显示窗口进行数据采集，并将所采集到的数据以波形的形式显示出来。然后，就可以对波形数据进行分析。如图 5-9 所示。点击工具栏左侧第一个按钮，逻辑分析仪就开始采集数据。

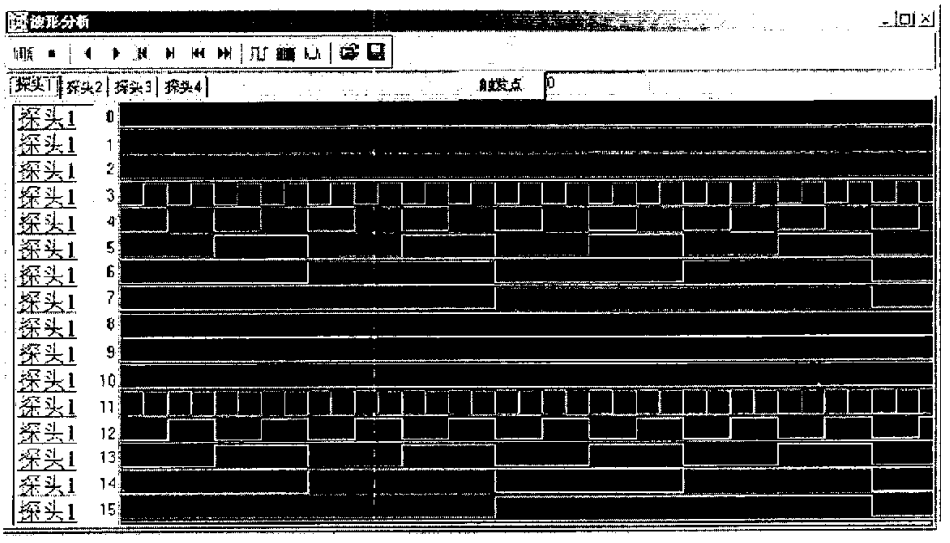


图 3-9．逻辑分析仪波形分析／显示窗口

★分析数据

波形分析／显示窗口工具栏一共有十个按钮，从左至右功能依次是：运行、终止、前移一位、后移一位、前翻一页、后翻一页、前翻到起始位置、后翻到结束位置、波形展宽和波形缩窄。在波形显示窗口的左侧可以对各通道分别编辑其标记名称，对于未使用的通道将没有波形显示。波形分析窗口一个重要的功能是可以通过左右鼠标控制的光标来确定分析数据的位置，而且只要在相应位置轻轻点击鼠标左键就可以马上得到该位置处的波形所对应的十进制、十六进制数据。如图 3-10 所示。

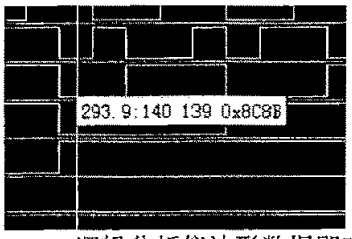
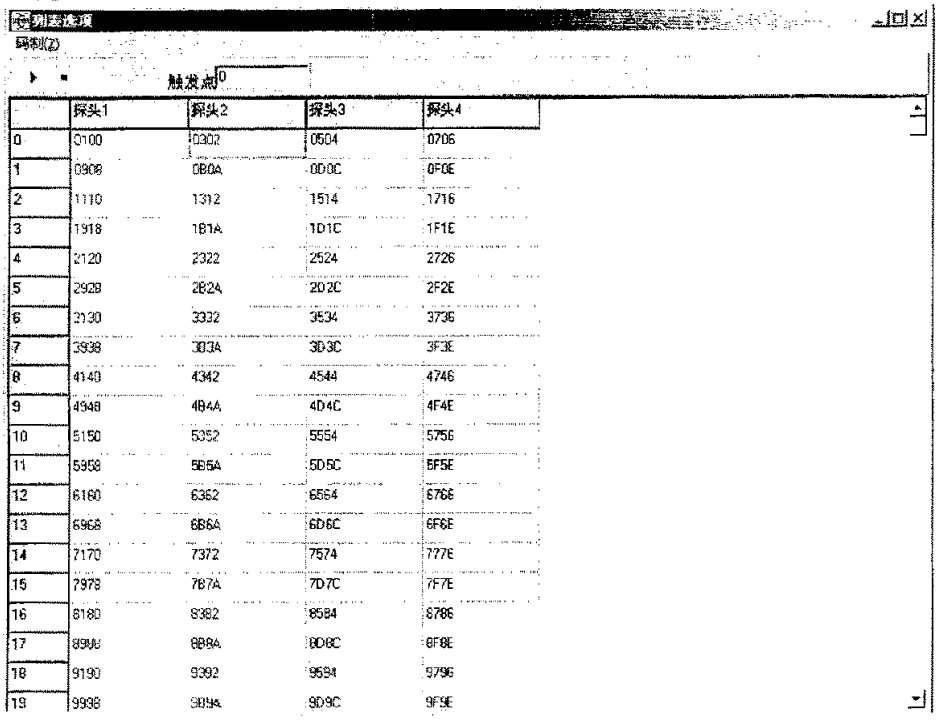


图 3-10. 逻辑分析仪波形数据即时显示

采集数据除了可以波形显示以外，还能以数据列表的形式显示。退出波形分析 / 显示窗口回到控制面板，点击［列表］按钮，就会显示列表窗口，如图 3-11 所示。点击［运行］按钮，就可以列出各通道所采集的数据。



	探头1	探头2	探头3	探头4
0	0100	0302	0504	0706
1	0908	0B0A	0D0C	0F0E
2	1110	1312	1514	1716
3	1918	1B1A	1D1C	1F1E
4	2120	2322	2524	2726
5	2928	2B2A	2D2C	2F2E
6	3130	3332	3534	3736
7	3938	3B3A	3D3C	3F3E
8	4140	4342	4544	4746
9	4948	4B4A	4D4C	4F4E
10	5150	5352	5554	5756
11	5958	5B5A	5D5C	5F5E
12	6160	6362	6564	6766
13	6968	6B6A	6D6C	6F6E
14	7170	7372	7574	7776
15	7978	7B7A	7D7C	7F7E
16	8180	8382	8584	8786
17	8988	8B8A	8D8C	8F8E
18	9190	9392	9594	9796
19	9998	9B9A	9D9C	9F9E

图 3-11. 逻辑分析仪列表数据显示

如果在状态分析仪模式，还可以对采集数据进行反汇编。在控制面板单击[反汇编]按钮，便进入反汇编窗口。如图 3-12 所示。本程序提供了对多种指令集的反汇编支持，其中包括：8086、80386、8085、Z80、6800、6502、8048、8051、8096 等 CPU 指令集。另外，程序还提供了反汇编结果存盘、打开以前存盘结果、改变读写模式等功能。

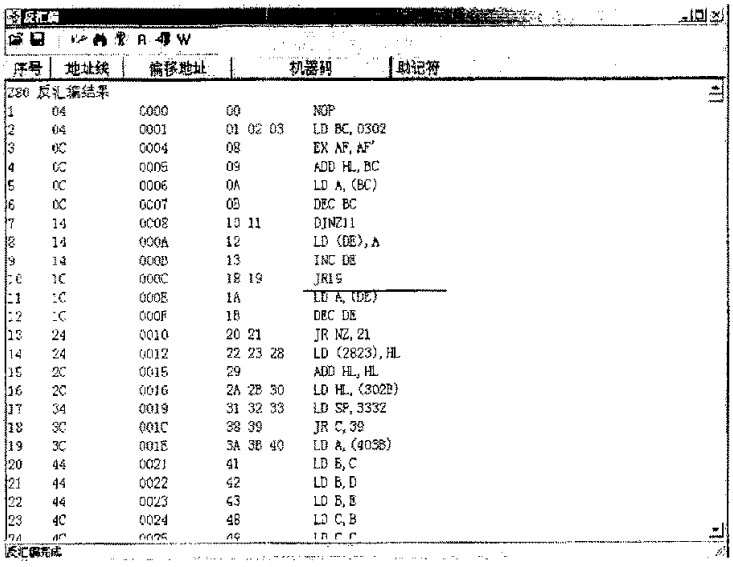


图 3-12 逻辑分析仪反汇编功能

3.2 逻辑分析仪系统软件的设计

3.2.1 软件的结构

逻辑分析仪程序总共由 17 个窗体加 5 个单元文件组成，各主要窗体之间的关系可由图 3－13 所示；

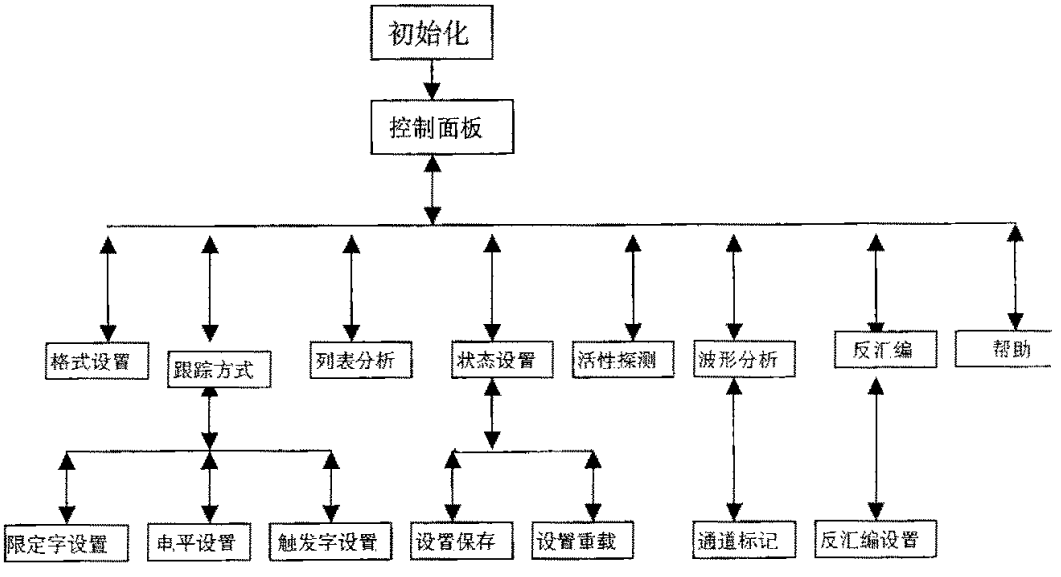


图 3－13 各窗体之间的关系

3.2.2 关于自定义类 LAnalyzer

根据 2.1 节所述，类通常称为具有同样性质和功能的事物所构成的集合，在本程序设计中，根据逻辑分析仪的特性，我们归纳出了适合于编程应用的类 ▮ ▮LAnalyzer。其定义如下：

```
class LAnalyser
{
private:
public:
    bool LAMode;              //逻辑分析仪模式  1=定时分析仪
    bool BSample;             //起始触发
    bool ESample;             //终止触发
    bool RSample;             //随机触发
    bool SampleRpt;           //重复采样
    bool TrgMode;             //触发模式，组合 0/序列 1
    bool StrgMode;           //存储限定模式 组合 0/序列 1
    int StrgSerise;          //存储限定级数
    int TrgSerise;       //触发级数  随机触发-->0,起始/终止触发-->"1...7"
    int RptNumber;              //重复次数
    int DelayTime;              // 延时周期数
    byte *DataBuffer;           //数据存储区首地址
    byte *Qglitch;           //毛刺位存储区首地址
    UINT datalen;               //+++数据长度
    POD POD1;                   //+++++++++
    POD POD2;                   //+++++++++
    POD POD3;                   //+++++++++
    POD POD4;                   //+++++++++
    unsigned short int cw[5][21];           //触发限定字
    unsigned short int cx[5][21];           //触发限定字任意位
    unsigned short int sw[5][21];           //存储限定字
    unsigned short int sx[5][21];            //存储限定字任意位
    int strgwn[8];       //第一到第八级存储限定字号
    int trgwn[8];                //第一到八级触发字号
                  //trgwn[0]存放第一级触发字在 cw,cx 中的序号
                 //trgwn[1]存放第二级触发字在 cw,cx 中的序号....
```

```
int resnum;              //存储限定字在 cw,cx 中的序号++++++  0~~无限定,
float volt;
freq speed;          // 0~10 = 400,200,100,50,10,5,2,1,0.5,0.2,0.1MHz
int stateclk;                    // J-0 / K-1
int site;
_fastcall LAnalyser() ;
_fastcall ~LAnalyser();
void SampleMode(int mode);
     };
```

其中，枚举形变量 freq 代表定时分析仪的采样频率；结构体 POD 代表逻辑分析仪的探头参数,其定义如下:

```
struct  POD
{
 unsigned short int        ipod;        //探头使用位
 bool       polarity;                  //探头极性(+)-1
 bool       clkup;                     //时钟上升沿-1
};
```

在类的定义中有两个成员函数特别值得说明，它们是 LAnalyzer()和~Lanalyzer()，两者分别代表了此类的构造函数和析构函数。当声明一个类的对象时，编译需要为对象分配存储空间，进行许多必要的初始化，这部分工作就有构造函数来完成。在此构造函数中，提供了对逻辑分析仪的缺省参数的设置，以及数据存储空间的分配。与构造函数相对应的是析构函数，当类对象超出作用域时，析构函数删除对象，回收存储释放内存空间。

此外，成员函数 SampleMode(int mode)用来设置逻辑分析仪的触发方式。它将触发方式排他地设置为起始触发、终止触发或随机触发三种方式之一。

3.3 系统软件中的窗体及功能设计实现

3.3.1 逻辑分析仪系统软件组成

单元文件	表单文件	说明
ACTIVITY.CPP	FRMACT.DFM	探头活性
LAABOUT.CPP	FRMABOUT.DFM	初始化表单
LACTRL.CPP	FRMCTRL.DFM	控制面板
LAFORMAT.CPP	FRMFORMAT.DFM	探头格式设置
LALIST.CPP	FRMLIST.DFM	列表分析
LASET.CPP	FRMSET.DFM	状态设置

LASTRGD.CPP	FRMSTRGD.DFM	存储限定字设置
LATRACE.CPP	FRMTRACE.DFM	
LAUASM.CPP	FRMUASM.DFM	反汇编
PREUASM.CPP	FRMPREUASM.DFM	反汇编设置
SETVOLT.CPP	FRMVOLT.DFM	中心电压设置
STCLK.CPP	FRMCLK.DFM	状态时钟设置
WAVES.CPP	FRMWAVES.DFM	波形分析窗口
WNOTE.CPP	FRMNOTE.DFM	探头标记
WTRACE.CPP	FRMWTRAC.DFM	跟踪方式设置
WTRIG.CPP	FRMWTRIG.DFM	触发字设置
TRIGD.CPP	FRMTRIGD.DFM	定义预置字(二进制)
WABOUTBOX.CPP	FRMWABOUTBOX.DFM	关于对话框
ASMIO.CPP		端口输入输出
CONVDATA.CPP		进制转换
FIFOTEST.CPP		Fifo自检
LASAVE.CPP		自定义类
WAVEDRAW.CPP		波形采集子线程
MAINCALL.CPP		底层采集数据
WUASM.CPP		反汇编函数

表 3-1 逻辑分析仪系统软件组成

3.3.2 功能设计

3.3.2.1 activity.cpp 单元设计

1．函数及全局变量

名称	功能
Timer1Timer(Tobject *Sender);	定时器时间的函数
Void __fastcall SensePod1();	探头 1 活性
Void __fastcall SensePod2();	探头 2 活性
Void __fastcall SensePod3();	探头 3 活性
Void __fastcall SensePod4();	探头 4 活性
DrawArrow(TpaintBox *pb,int pos);	画活性箭头
Lanalyser * latemp;	存逻辑分析仪参数

2．窗体功能及部分函数的使用

此单元所完成的功能是检测各探头是否连接正确，当在设置窗口选定接入的探头并且输入信号后，在探头活性窗口将会看到相应探头 PaintBox 控件会呈现出箭头状态，否则相应探头则只显示出短横线。

在此窗体中我们选定了一个时钟控件、四个 Label 控件、四个 PaintBox 控件和一个 Button 按钮。四个 Label 控件用于标示各探头的名称，四个 PaintBox 控件用于画标

示探头活性的箭头。

Timer1Timer(Tobject *Sender)分别检测探头的接入情况；

Void __fastcall SensePod1()是探头 1 的活性检测函数；

Void __fastcall SensePod2()是探头 2 的活性检测函数；

Void __fastcall SensePod3()是探头 3 的活性检测函数；

Void __fastcall sensePod4()是探头 4 的活性检测函数；

DrawArrow(TpaintBox *pb,int pos)函数是画图函数。其中探头活性检测函数中，当有信号输入时，在相应探头名称正下方的 PaintBox 控件中画箭头，否则将以短横线表示。

3.3.2.2 单元 lactrl.cpp 的设计

1 函数及全局变量

名称	功能
SpeedButton6Click(Tobject *Sender);	状态设置
SpBtnFormatClick(Tobject *Sender);	格式设置
SpeedButton3Click(Tobject *Sender);	波形分析
SpBtnTraceClick(Tobject *Sender);	跟踪方式设置
SpBtnListClick(Tobject *Sender);	列表分析
SpBtnUasmClick(Tobject *Sender);	反汇编
SpeedButton2Click(Tobject *Sender);	关于逻辑分析仪
SpeedButton1Click(Tobject *Sender);	探头活性
SpeedButton7Click(Tobject *Sender);	帮助

2 窗体功能及部分函数的使用

逻辑分析仪控制面板是逻辑分析仪的主控中心，通过控制面板来设置逻辑分析仪的各种参数，并控制逻辑分析仪的运行。

为此，我们选定了九个 SpeedButton 控件，这九个控件分别完成“状态设置”、“探头格式设置”、“跟踪方式设置”、“波形显示”、“状态列表”、“反汇编代码”、“探头活性”、“帮助”功能，在每次点击这九个控件时，会分别弹出相应的窗口。以及六个 Label 控件。其中四个 Label 控件用于表示探头名称。一个用于标示逻辑分析仪工作方式以及显示所选择的逻辑分析仪工作方式。

SpeedButton6Click(Tobject *Sender)函数用于“弹出逻辑分析仪设置窗体”，进行逻辑分析仪工作方式和逻辑分析仪探头的设置；

SpBtnFormatClick(Tobject *Sender)函数用于“弹出逻辑分析仪的格式设置窗体”，进行逻辑分析仪探头极性和时钟触发沿的设置；

SpeedButton3Click(Tobject *Sender)函数“弹出逻辑分析仪的波形显示窗口”，

对逻辑分析仪采集到的数据进行波形分析；

SpBtnTraceClick(Tobject *Sender)函数“弹出逻辑分析仪跟踪方式设置窗体”，进行跟踪方式设置；

SpBtnListClick(Tobject *Sender)函数“弹出逻辑分析仪列表窗口”，对逻辑分析仪所采集到的数据以列表的形式显示出来；

SpBtnUasmClick(Tobject *Sender)函数“弹出反汇编窗体”，对逻辑分析仪所采集到的二进制数据进行反汇编，并以汇编语言的形式显示出来；

SpeedButton2Click(Tobject *Sender) 函数用于“弹出一个对话框”，显示该逻辑分析仪的版权，以及研制单位；

SpeedButton1Click(Tobject *Sender) 函数用于“弹出显示探头活性的窗口”；

SpeedButton7Click(Tobject *Sender) 函数用于弹出“显示整个逻辑分析仪的帮助的窗体”；

Label1~ label4 用于表示该逻辑分析仪所用的探头，当我们选定所要用的探头，并且保存设置后将在控制面板上以蓝颜色显示出来。同理，当我们选定逻辑分析仪的工作方式，以及逻辑分析仪的工作

3.3.2.3 单元 laFormat.cpp 的设计

1 函数及全局变量

名称	功能
bitBtnOkClick(TObject*Sender);	保存设置
Pos1Click(TObject*Sender);	设置探头 1 极性
Pos2Click(TObject*Sender);	设置探头 2 极性
Pos3Click(Tobject*Sender);	设置探头 3 极性
Pos4Click(TObject*Sender);	设置探头 4 极性
Clkup1Click(TObject*Sender);	设置探头 1 时钟触发沿
Clkup2Click(TObject*Sender);	设置探头 2 时钟触发沿
Clkup3Click(Tobject*Sender);	设置探头 3 时钟触发沿
Clkup4Click(TObject*Sender);	设置探头 4 时钟触发沿

2 窗体功能及部分函数的使用

此窗体完成逻辑分析仪探头格式的设置，所谓“探头格式”也就是探头的极性以及探头的触发沿，探头的正极性还是负极性，触发沿是上升沿还是下降沿。在此窗体中，总共有九个函数，分别进行探头极性设置和时钟触发沿设置并且保存设置。

bitBtnOkClick(TObject*Sender)函数用于对所设置的参数值进行保存，并且将所设置的参数值传递到类 Lanalyser 的变量

3.3.2.4 单元 laList.cpp 的设计

1 函数及全局变量

名称	功能
BitBtn1Click(TObject*Sender);	开始数据采集
N7Click(TObject*Sender);	十六进制转换
N8Click(TObject*Sender);	十进制转换
N9Click(Tobject*Sender);	二进制转换
N10Click(Tobject*Sender);	八进制转换
SpeedButton4Click(TObject *Sender);	停止数据采集
void__fastcallSpeedClick();	填充数据到列表框，供线程调用
thrdlist*listTh;	列表采集线程对象
TEvent*datardy;	开始采集事件对象
boolruned;	已采集数据

2 窗体功能及部分函数的使用

此窗体的功能是完成对数据的采集，并以列表的形式显示各通道所采集到的数据。数据的码制可以是十六进制、二进制、八进制或十进制。

BitBtn1Click(TObject*Sender)实现数据采集功能，当按下按钮时开始数据采集，当采集到数据并以列表的形式显示出来时，按钮由灰变亮；

N7Click(TObject*Sender)函数对采集到的数据以十六进制的形式显示出来，或者是将所显示的数据转换成八进制数据；

N8Click(TObject*Sender) 函数对采集到的数据以十进制的形式显示出来，或者是将所显示的数据转换成八进制数据；

N9Click(Tobject*Sender) 函数对采集到的数据以二进制的形式显示出来，或者是将所显示的数据转换成八进制数据；

N10Click(Tobject*Sender) 函数对采集到的数据以八进制的形式显示出来，或者是将所显示的数据转换成八进制数据；

SpeedButton4Click(Tobject *Sender)当采集数据时，由于某种原因不能停止数据采集，这时我们必须强行停止数据采集。

void__fastcallSpeedClick()该函数将所采集到的数据填充到列表框，供线程调用。

3.3.2.5 单元 laSet.cpp 的设计

1 函数及全局变量

名称	功能
bitBtnOkClick(TObject*Sender);	保存设置

RadioButton1Click(TObject *Sender);	选择定时分析仪
BitBtn1Click(TObject*Sender);	重载设置
BitBtn2Click(TObject*Sender);	另存设置
RadioButton2DblClick(TObject *Sender);	选择状态分析时钟
RadioButton2Click(Tobject *Sender);	选择状态分析仪
ComboBox1Change(TObject *Sender);	改变定时分析采样频率

2 部分函数的使用

bitBtnOkClick(TObject*Sender)；用于保存所设置的参数；

RadioButton1Click(TObject *Sender)；选择逻辑分析仪的工作方式为定时分析仪；

BitBtn1Click(TObject*Sender)；打开以前用过的设置；

BitBtn2Click(TObject*Sender)；保存现在的设置；

ComboBox1Change(TObject *Sender)；改变定时分析采样频率。

3.3.2.6 单元 laUasm.cpp 的设计

1 函数及全局变量

名称	功能
ToolButton1Click(Tobject*Sender);	打开文件
ToolButton2Click(Tobject*Sender);	保存文件
ToolButton6Click(Tobject*Sender);	行反汇编
ToolButton10Click(Tobject*Sender);	停止数据采集
ToolButton9MouseDown(Tobject *Sender, TmouseButton Button, TshiftState,int X,int Y);	改变 Button9 图标
ToolButton9Click(Tobject *Sender);	设置只读模式
ToolButton7Click(Tobject *Sender);	查找
ToolButton1Click(Tobject *Sender);	标记找到的文字
ToolButton8Click(Tobject *Sender);	反汇编设置
Int uasmindex;	反汇编种类
BYTE *pdata;	数据指针
BYTE *paddr;	地址指针
Int datalength;	反汇编数据长度
Int dataserise[12];	数据线排序顺序
Int addrserise[12];	地址线排序顺序
Bool showaddr;	显示地址线数据
Int acnt;	地址线位宽(字节)

Int dcnt; 数据线位宽(字节)

2 窗体功能及部分函数的使用

反汇编是汇编的逆过程，即将二进制目标程序反向翻译成汇编语言程序的过程。该窗体完成对反汇编的相关操作。

ToolButton1Click(Tobject*Sender)；该函数用于打开被保存的反汇编后的文件；

ToolButton2Click(Tobject*Sender)；用于保存反汇后的文件；

ToolButton10Click(Tobject*Sender)；用于退出反汇编窗体，返回到主控窗体；

ToolButton9MouseDown(Tobject *Sender,TmouseButton Button,TshiftState,int X,int Y)；用于改变 Button9 图标(“读/写模式：可写”；“读/写模式：只读”)；

ToolButton7Click(Tobject *Sender);用于在反汇编代码中查找特定字符串；

ToolButton1Click(Tobject *Sender)；标记找到的文字；

ToolButton8Click(Tobject *Sender);反汇编设置(数据位，地址位以及选择反汇编指令集)。

3.3.2.7 单元 preuasm.cpp 的设计

1 函数及全局变量

名称	功能
Button1Click(Tobject*Sender);	添加到数据位
Button2Click(Tobject*Sender);	数据位返回到可用位
Button3Click(Tobject*Sender);	数据位全部返回到可用位
Button4Click(Tobject*Sender);	全部添加到数据位
Button5Click(Tobject*Sender);	全部添加到地址位
Button6Click(Tobject*Sender);	地址位全部返回到可用位
Button7Click(Tobject*Sender);	地址位返回到可用位
Button8Click(Tobject*Sender);	添加到地址位
bitBtnOkClick(Tobject*Sender);	确认设置退出

2 窗体功能及部分函数的使用

该窗体的作用是完成反汇编前的设置功能，这包括地址线的添加与选择、数据线的添加与选择、反汇编指令集的选择。其中地址线、数据线的选择与添加与整个逻辑分析仪的设置有关。Button1Click(Tobject*Sender)；表示将所选的可用位添加到数据位；

Button3Click(Tobject*Sender);表示将所有的数据位全部返回到可用位；

Button4Click(Tobject*Sender);表示将所有的可用位全部添加到数据位；

Button5Click(Tobject*Sender)；表示将所有的可用位全部添加到地址位；

Button6Click(Tobject*Sender);表示将所有的地址位全部返回到可用位；

Button7Click(Tobject*Sender)；表示将所选的地址位返回到可用位；

Button8Click(Tobject*Sender);表示将所选的可用位添加到地址位;

bitBtnOkClick(Tobject*Sender);表示保存设置的参数并退出窗口。

3.3.2.8 单元 WaveS.cpp 的设计

1 函数及全局变量

名称	功能
PaintBoxlPaint(Tobject*Sender);	探头 1 波形重画
PaintBox2Paint(TObject*Sender);	探头 2 波形重画
PaintBox3Paint(TObject*Sender);	探头 3 波形重画
PaintBox4Paint(TObject*Sender);	探头 4 波形重画
PaintBoxlMouseDown(Tobject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y);	PaintBoxl 上绘制光标
ToolButtonlClick(TObject*Sender);	启动数据采集
ToolButtonllClick(Tobject*Sender);	波形展宽
ToolButtonl2Click(Tobject*Sender);	波形缩窄
ToolButtonl0Click(TObject*Sender);	显示最后一屏波形
ToolButton8Click(Tobject*Sender);	显示第一屏波形
ToolButton7Click(TObject*Sender);	波形后移一屏
ToolButton6Click(TObject*Sender);	波形前移一屏
ToolButton3Click(TObject*Sender);	终止数据采集
ListBoxlDblClick(TObject*Sender);	设置探头 1 通道标记
IistBox2DblClick(TObject*Sender);	设置探头 2 通道标记
ListBox3DblClick(Tobject*Sender);	设置探头 3 通道标记
ListBox4DblClick(Tobject*Sender);	设置探头 4 通道标记
N1Click(TObject*Sender);	设置探头通道标记
PaintBox2MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y);	PaintBox2 上绘制光标
PaintBox3MouseDown(TObject *Sende, TMouseButton Button, TShifstate Shift, int X, int Y);	PaintBox3 上绘制光标
PaintBox4MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y);	PaintBox4 上绘制光标

```
ToolButton4Click(Tobject*Sender);          波形前移一周期
ToolButton5Click(TObject*Sender);          波形后移一周期
FormKeyDown(TObject    *Sender,            按键移动光标
WORD  &Key,     TshiftState  Shift);
ToolButtonl 3Click(Tobject*Sender);        开启毛刺
DrawGlitch( unsigned char *, wTPaintBox*); 画毛刺
DrawGlitch8( unsigned char *, wTPaintBox *); 画 8 位毛刺
ToolButtonl5Click(TObject *Sender);        打开保存的波形文件
ToolButtonl6Click(TObject *Sender);        以文件形式存储波形
wavedraw  *drwTh;                          线程对象
TEvent*datardy;                            事件对象
bool stop;                                 停止数据采集
bool showbit[161;                          显示通道波形
byte  *wdatab;                             指向数据的指针
int  opos;                                 基准光标位置
int  xpos;                                 光标位置
int  dist;                                 光标偏移值
int  weight;                               单位脉冲宽度
bool  runed;                               已采集数据
bool  cleft;                               左光标被击活
bool  cright;                              基准光标被击活
int base;                                  通道宽度
int yh;                                    波形高度
int  bnum;                                 8 个字节
int num;                                   一屏所能画的采样点数
float  f;                                  采样频率
int px[16];                                画笔位置横坐标
int py[161;                                画笔位置纵坐标
BYTE pb;                                   通道屏蔽标志
unsigned short pw;                         通道屏蔽标志
bool showglitch;                           显示毛刺
```

2 窗体功能及部分函数的使用

该窗体实现的功能是采集数据并以波形的形式显示出来。在采集的过程中，可能由于某种原因，(如未遇到触发字或重复采样方式)不能自行退出采集，就要求人为终结采

集，这就必须通过多线程来实现。为此，创建一线程对象 drawthread 和事件对象 datardy，事件对象用于通知子线程开始进行数据采集；子线程 drawthread 不断检测事件 datardy,一旦发现事件被设置，就开始执行数据采集。为了能中断数据采集过程，设置变量 stop,VCL 主线程可以在任意时刻设置它的值，在采集循环过程中，当 stop 被设置为 true 时则终止数据采集。

PaintBox1MouseDown(Tobject *Sender,TMouseButton Button,TShiftState Shift, int X, int Y)；在 PaintBox1 上按下鼠标左键时，在 PaintBox1 上按下鼠标处绘制光标；

Tool Button1Click(TObject*Sender)；运行,启动数据采集函数。

Tool Button3Click(TObject*Sender)；用于终止数据采集。

PaintBox2MouseDown(TObject*Sender, TMouseButton Button, TShiftState Shift, int X, int Y)；在 PaintBox2 上按下鼠标左键时，在 PaintBox2 上按下鼠标处绘制光标； PaintBox3MouseDown(Tobject *Sender, TMouseButton Button, TShifstate Shift, int X, int Y)；在 PaintBox3 上按下鼠标左键时，在 PaintBox3 上按下鼠标处绘制光标；PaintBox4MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)；在 PaintBox4 上按下鼠标左键时，在 PaintBox4 上按下鼠标处绘制光标；

FormKeyDown(TObject *Sender, WORD &Key, TshiftState Shift)；当按下“←”或“→”按键时移动光标一个象素；

3.3.2.9 单元 wtrace.cpp 的设计

1 函数及全局变量：

名称	功能
N11Click(TObject*Sender);	起始触发
N12Click(TObject*Sender);	终止触发
N13Click(Tobject*Sender);	随机触发
N14Click(TObject*Sender);	单次采样
N15Click(TObject*Sender);	重复采样
N21Click(Tobject*Sender);	设置触发字
N22Click(Tobject*Sender);	设置延时
N31Click(TObject*Sender);	十六进制
N32Click(Tobject*Sender);	十进制
N33Click(TObject*Sender);	八进制
N34Click(TObject*Sender);	二进制
N35Click(Tobject*Sender);	ASIC 码

Button2Click(TObject *Sender);	设置限定字
Button1Click(TObject*Sender);	设置触发字
bitBtnokClick(Tobject*Sender);	确认
N5Click(TObject*Sender);	任意电平设置
TTL1Click(Tobject*Sender);	TTL 电平
ECL1Click(Tobject*Sender);	ECL 电平
Edit2Exit(Tobject*Sender);	判断延时范围

2 窗体功能及部分函数的使用

该窗体的作用是进行跟踪方式设置。可设置跟踪方式为起始/跟踪/随机触发方式之一，随机触发时还可设置为单次或重复采样和存储限定字方式。如果需要设定限定字，则点击限定字按扭，将弹出限定字设置窗体，进行相应项的设定。起始/跟踪触发时可设置触发字方式。N31Click(TObject*Sender)用于实现触发字或限定字以十六进制表示；

N32Click(Tobject*Sender) 用于实现触发字或限定字以十进制形式表示；

N33Click(TObject*Sender) 用于实现触发字或限定字以八进制形式表示；

N34Click(TObject*Sender) 用于实现触发字或限定字以二进制形式表示；

N35Click(Tobject*Sender) 用于实现触发字或限定字以 ASIC 形式表示；

bitBtnokClick(Tobject*Sender)用于实现窗体参数的保存；

TTL1Click(Tobject*Sender)用于实现 TTL 电平设置；

ECL1Click(Tobject*Sender)用于实现 ECL 电平的设置。

3.3.2.10 单元 Wuasm.cpp 的设计

1 函数及全局变量

名称	功能
18086uasm(BYTE “pdata, int length, TStringList&Lines);	8086 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果
z80uasm(BYTE “ pdata, int length, TStringList&Lines);	z80 反汇编 Pdata：机器码首地址 Length：机器码长度 Lines：反汇编结果
18048uasm(BYTE “pdata, int length, TStringList&Lines);	8048 反汇编 pdata：机器码首地址 length：机器码长度

	Lines：反汇编结果
18051uasm(BYTE" pdata, int length, TStringList&Lines);	8051 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果
16502uasm(BYTE 2k pdata, int length, TStringList&Lines);	6502 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果
18096uasm(BYTE 4c pdata, int length, TStringList&Lines);	8096 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果
180386uasm(BYTE " pdata, int length, TStringList&Lines);	80386 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果
M6800uasm(BYTE dc pdata, int length, TStringList&Lines);	6805 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果
18085uasm(BYTE "pdata, int length, TStringList&Lines);	8080 / 8085 反汇编 pdata：机器码首地址 length：机器码长度 Lines：反汇编结果

3.3.2.11 preUasm.cpp 单元的设计

1 函数及全局变量

函数名称	功能
BitOkClick(TObject *Sender);	确认功能
BitBtn3Click(TObject *Sender);	预置字功能
BitCanclClick(TObject *Sender);	取消功能
ComboBox10Change(TObject *Sender);	触发方式选择功能
N1Click(TObject *Sender);	十六进制
N3Click(TObject *Sender);	十进制

N2Click(TObject *Sender);	八进制
N4Click(TObject *Sender);	二进制
N5Click(TObject *Sender);	ASIC 码

2 窗体功能及部分函数的使用

本窗体的功能是实现触发字的设置。

BitOkClick(TObject *Sender)；保存参数，并关闭窗体；

BitCanclClick(TObject *Sender)；不保存参数，直接关闭窗体；

ComboBox10Change(TObject *Sender)；实现触发方式选择功能，即触发字是序列触发还是组合触发；

N1Click(TObject *Sender)；触发字以十六进制表示；

N3Click(TObject *Sender)；触发字以十进制形式表示；

N2Click(TObject *Sender)；触发字以八进制形式表示；

N4Click(TObject *Sender)；触发字以二进制形式表示；

N5Click(TObject *Sender)；触发字以 ASIC 码形式表示。

3.3.2.12laStrgD.cpp 单元的设计

1 函数及全局变量

名称	功能
BitOkClick(TObject *Sender);	确认功能
BitBtn3Click(TObject *Sender);	预置限定字功能
BitCanclClick(TObject *Sender);	取消功能
ComboBox10Change(TObject *Sender);	限定字级数
N1Click(TObject *Sender);	十六进制
N3Click(TObject *Sender);	十进制
N2Click(TObject *Sender);	八进制
N4Click(TObject *Sender);	二进制
N5Click(TObject *Sender);	ASIC 码

2 窗体功能及部分函数的使用

本窗体的功能是实现限定字的设置。

BitOkClick(TObject *Sender)；保存参数，并关闭窗体功能；

BitCanclClick(TObject *Sender)；不保存参数，直接关闭窗体；

ComboBox10Change(TObject *Sender)；实现有、无限定字选择功能；

N1Click(TObject *Sender)； 限定字以十六进制表示；

N3Click(TObject *Sender)；限定字以十进制形式表示；

N2Click(TObject *Sender)； 限定字以八进制形式表示；

N4Click(TObject *Sender)； 限定字以二进制形式表示；

N5Click(TObject *Sender)；限定字以 ASIC 码形式表示。

3.3.2.13 单元 asmio.cpp 的设计

1 函数及全局变量

名称	功能
unsigned char __fastcall inportb(int portid)；	读端口 portid：端口号 返回值：端口数据
Void __fastcall outportb(int portid, unsigned char Value)；	写端口； portid： 端口号 value：输出值

3.3.2.14 单元 convdata.cpp 的设计

1 函数及全局变量

名称	功能
AnsiString UsiToDec(unsigned short)；	无符号短整形转换为十进制字符串
AnsiString UsiToHex(unsigned short)；	符号短整形转换为十六进制字符串
AnsiString UsiToOct(unsigned short)；	无符号短整形转换为八进制字符串
AnsiString UsiToBin(unsigned short)；	无符号短整形转换为二进制字符串
AnsiString UsiToAsc(unsigned short)；	无符号短整形转换为 ASIC 字符串
unsigned short DecToUsi(AnsiString)；	十进制字符串转换为无符号短整形
unsigned short HexToUsi(AnsiString)；	十六进制字符串转换为无符号短整形
unsigned short OctToUsi(AnsiString)；	八进制字符串转换为无符号短整形
unsigned short BinToUsi(AnsiString)；	二进制字符串转换为无符号短整形
unsigned short AscToUsi(AnsiString)；	ASIC 字符转换为无符号短整形数
Void getdec(AnsiString, unsigned ShOrt&, unsigned short&)；	保存十进制预置字/触发字
Void gethex(AnsiString, unsigned Short&, unsigned short&)；	保存十六进制预置字/触发字
Void getoct(AnsiString, unsigned Short&, unsigned short&)；	保存八进制预置字/触发字
Void getbin(AnsiString, unsigned Short&, unsigned short&)；	保存二进制预置字/触发字
Void getasc(AnsiString, unsigned	保存 ASIC 字符预置字/触发字

```
    Short&, unsigned short&);
AnsiStrinng  putdec( unsigned  short,        显示十进制预置字/触发字
    unsigne short);
AnsiString  puthex( unsigned  short,         显示十六进制预置字/触发字
   unsigned short);
AnsiString  putoct( unsigned  short,         显示八进制预置字/触发字
    unsigned short);
AnsiString  putbin( unsigned  short,         显示二进制预置字/触发字
    unsigned short);
AnsiString  putasc( unsigned  short,         显示 ASIC 字符预置字/触发字
    unsigned short);
```

3.4 系统软件的变量传递设计

1 控制面板

```
pAnalyser->POD1.ipod ──────────► frmCtrl-> Label1->Visible
pAnalyser->POD2.ipod ──────────► frmCtrl->Label2->Visible
pAnalyser->POD3.ipod ──────────► frmCtrl->Label3->Visible
pAnalyser->POD4.ipod ──────────► frmCtrl->Label4->Visibl
pAnalyser->LAMode ─────────────► frmCtrl->Edit1->Text
```

2 逻辑分析仪设置

```
frmSet->RadioButton1->Checked ──► pAnalyser->LAMode=1
frmSet->RadioButton2->Checked ──► pAnalyser->LAMode=0
frmSet->comboBox1->ItemIndex ──► pAnalyser->speed
frmSet->CheckBox1=Checked ─────► pAnalyser->POD1.ipod
frmSet->CheckBox2=Checked ─────► pAnalyser->POD2.ipod
frmSet->CheckBox3=Checked ─────► pAnalyser->POD3.ipod
frmSet->CheckBox4=Checked ─────► pAnalyser->POD4.ipod
```

3 探头格式设置

```
frmFormat->alias1->Text ──────► frmWaveS->POD1->Caption
frmFormat->alias2->Text ──────► frmWaveS->POD2->Caption
frmFormat->alias3->Text ──────► frmWaveS->POD3->Caption
frmFormat->alias4->Text ──────► frmWaveS->POD4->Caption
```

frmFormat->pos1->ItemIndex ——→ pAnalyser->POD1.polarity

frmFormat->pos2->ItemIndex ——→ pAnalyser->POD2.polarity

frmFormat->pos3->ItemIndex ——→ pAnalyser->POD3.polarity

frmFormat->pos4->ItemIndex ——→ pAnalyser->POD4.polarity

frmFormat->clkup1->ItemIndex ——→ frmWaveS->POD1.clkup

frmFormat->clkup2->ItemIndex ——→ frmWaveS->POD2.clkup

frmFormat->clkup3->ItemIndex ——→ frmWaveS->POD3.clkup

frmFormat->clkup4->ItemIndex ——→ frmWaveS->POD4.clkup

4 反汇编

frmpreuasm->ListBox3->Items--String[ ]=="探头 1: 7------0"
——→ frmUasm->addrserise[ ]=0

frmpreuasm->ListBox3->Items--String[ ]=="探头 1:15------8"
——→ frmUasm->addrserise[ ]=1

frmpreuasm->ListBox3->Items--String[ ]=="探头 2: 7------0"
——→ frmUasm->addrserise[ ]=2

frmpreuasm->ListBox3->Items--String[ ]=="探头 2: 15------8"
——→ frmUasm->addrserise[ ]=3

frmpreuasm->ListBox3->Items--String[ ]=="探头 3: 7------0"
——→ frmUasm->addrserise[ ]=4

frmpreuasm->ListBox3->Items--String[ ]=="探头 3: 15------8"
——→ frmUasm->addrserise[ ]=5

frmpreuasm->ListBox3->Items--String[ ]=="探头 4: 7------0"
——→ frmUasm->addrserise[ ]=6

frmpreuasm->ListBox3->Items--String[ ]=="探头 4: 15------8"
——→ frmUasm->addrserise[ ]=7

frmpreuasm->ListBox2->Items--String[ ]=="探头 1: 7------0"
——→ frmUasm->dataserise[ ]=0

frmpreuasm->ListBox2->Items--String[ ]=="探头 1: 15------8"
——→ frmUasm->dataserise[ ]=1

frmpreuasm->ListBox2->Items--String[ ]=="探头 2: 7------0"
——→ frmUasm->dataserise[ ]=2

frmpreuasm->ListBox2->Items--String[ ]=="探头 2: 15------8"
——→ frmUasm->dataserise[ ]=3

frmpreuasm->ListBox2->Items--String[ ]=="探头 3：7------0"
→ frmUasm->dataserise[ ]=4

frmpreuasm->ListBox2->Items--String[ ]=="探头 3：15------8"
→ frmUasm->dataserise[ ]=5

frmpreuasm->ListBox2->Items--String[ ]=="探头 4：7------0"
→ frmUasm->dataserise[ ]=6

frmpreuasm->ListBox2->Items--String[ ]=="探头 4：15------8"
→ frmUasm->dataserise[ ]=7

frmpreuasm->CheckBox1->Checked=true → frmUasm->Showaddr=true

5 跟踪方式设置

frmWtrac->RG1->ItemIndex=0 → pAnalyser->BSample=1

frmWtrac->RG1->ItemIndex=1 → pAnalyser->ESample=1

frmWtrac->RG1->ItemIndex=2 → pAnalyser->RSample=1

frmWtrac->Edit2->Text → pAnalyser->DelayTime

frmWtrac->Edit1->Text → pAnalyser->RptNumber

frmWtrac->RG2->ItemIndex=0 → pAnalyser->SampleRpt=1

frmWtrac->RG2->ItemIndex=1 → pAnalyser->SampleRpt=0

6 限定字设置

frmStrgD->ComboBox1->ItemIndex-1 → pAnalyser->StrgMode

frmStrgD->Combobox1->ItemIndex → pAnalyser->StrgSerise

frmStrgD->SB[ ]->ItemIndex → pAnalyser->Strgwn[ ]

frmStrgD->StringGrid1->Cells[ ][ ] → pAnalyser->sw[ ][ ]

frmStrgD->StringGrid1->Cells[ ][ ] → pAnalyser->sx[ ][ ]

7 触发字设置

frmTrgD->ComboBox1->ItemIndex-1 → pAnalyser->TrgMode

frmTrgD->Combobox1->ItemIndex → pAnalyser->TrgSerise

frmTrgD->SB[ ]->ItemIndex → pAnalyser->Trgwn[ ]

frmTrgD->StringGrid1->Cells[ ][ ] → pAnalyser->cw[ ][ ]

frmTrgD->StringGrid1->Cells[ ][ ] → pAnalyser->cx[ ][ ]

8 列表分析

frmFormat->alias1->Text ——→ frmList->SG1->Cells[1][0]

frmFormat->alias2->Text ——→ frmList->SG1->Cells[2][0]

frmFormat->alias3->Text ——→ frmList->SG1->Cells[3][0]

frmformat->alias4->Text ——→ frmList->SG1->Cells[4][0]

frmFormat->SG1->Cells[ ][ ] ——→ pAnalyser->DataBuffer[ ]

9 波形分析

frmWaveS->wdatab[ ] ——→ pAnalyser->DataBuffer[ ]

由于 ES4541 逻辑分析仪系统软件是一个比较复杂的程序，需要实现的功能比较多，下面将着重阐述以下几个方面的设计：

3.5 控制面板的实现

任何一种仪器都有控制面板，如果由于某种原因鼠标不能用，这时就必须有另一种操作方式来代替它。故对于仪器的操作功能应同时具备鼠标、面板操作。对于逻辑分析仪系统软件的面板控制分别实现了逻辑分析仪控制面板、列表选项、逻辑分析仪波形设置的面板控制功能。

在逻辑分析仪的控制面板中，实现了功能键操作方式。分别按下功能键 F2、F3、F4、F5、F6、F7、F8、F9 功能键时，将弹出逻辑分析仪设置窗口、探头格式设置窗口、跟踪方式设置窗口、波形分析、列表选项、反汇编、探头活性、关于逻辑分析仪、帮助九个窗口。

在列表选项窗口中，实现了数据采集/停止面板控制。当按下数据采集/停止按钮时，进行数据采集，并以列表的形式显示出来。而再次按下按钮时，退出数据采集。在显示时则代之以自检数据。

同理，在波形分析窗口中也相似，在波形分析中采用的是多线程同步方式，要特别注意指针的使用，以免出现内存冲突。在此，本人排除了一些错误，增强了程序的健壮性和可读性。

3.6 波形存档、打开功能的设计

由于逻辑分析仪是对数据进行分析，当遇到有意义的数据需要保存，以及需要对保存的数据进行分析、比较。我们必须具有将波形以文件的形式保存和打开的功能。这是对逻辑分析仪的一种基本要求。

3.7 存储限定字功能的实现

在逻辑分析仪的跟踪方式设置中，如果跟踪方式设置为随机触发，可以对采集到的数据进行有选择的存储。这就是限定存储功能的实现。所谓“限定存储”就是对数据流进行限定存储，当找到限定字后，fifo 才开始存储有效数据，直到存储器满。

为此，专门设计了一个窗口进行限定字的设置。在进行限定字设置时，我们设置了一个下拉框 ComboBox1，进行“有限定字”、“无限定字”的选择，用变量 **ComboBox1->ItemIndex-1------pAnalyser->StrgMode** 来进行变量的传递。当选择“无限定字”时，所有的限定字设置项无效，变量 **pAnalyser->StrgMode=0**。当选择“有限定字时”，相应的所有限定字的设置项将是活动的，变量 **pAnalyser->StrgMode=1** 并进行相应的设置。这包括限定字个数(变量为 **pAnalyser→StrgSerise**)、限定字(变量为 **pAnalyser→Strgwn[ ]**)、限定字设置(变量为 **pAnalyser→sw[ ][ ]**和 **pAnalyser→sx[ ][ ]**)的设置。限定字个数有 0—8 个，也就是限定字个数最多可为 8 个。在选定了限定字个数后，在限定字一栏的 8 个下拉列表框中，选择限定字所在的行号。设置了行号后，点击预置字按钮，将扩展限定字设置窗口，在相应的行号及所选择的探头进行限定字的设置。为了适应用户的习惯，还设置了相应的码制变换，当在字符列表框中点击鼠标右键时，将弹出快捷菜单，分别实现十六进制、十进制、八进制、二进制、ASIC 码的相互转换。另外，也可以在跟踪方式设置窗口的菜单中的码制菜单进行码制的设置。整个界面如图所示简洁，便于操作。

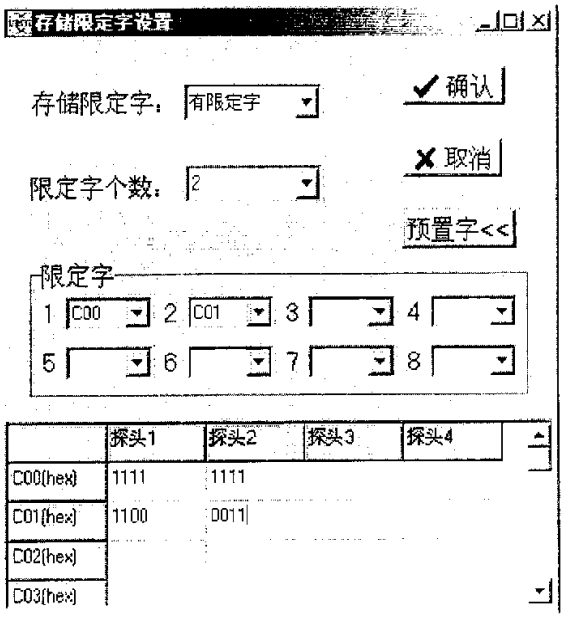


图 3-15 存储限定字设置窗口

第四章 ES4541 逻辑分析仪的程控功能设计

4．1 程控功能实现的硬件基础

程控指令的硬件基础是 GPIB 接口卡，接口芯片为 TMS9914，驱动芯片为 SN75160、SN75162，接口卡电路组成框图如图 4-1 所示：

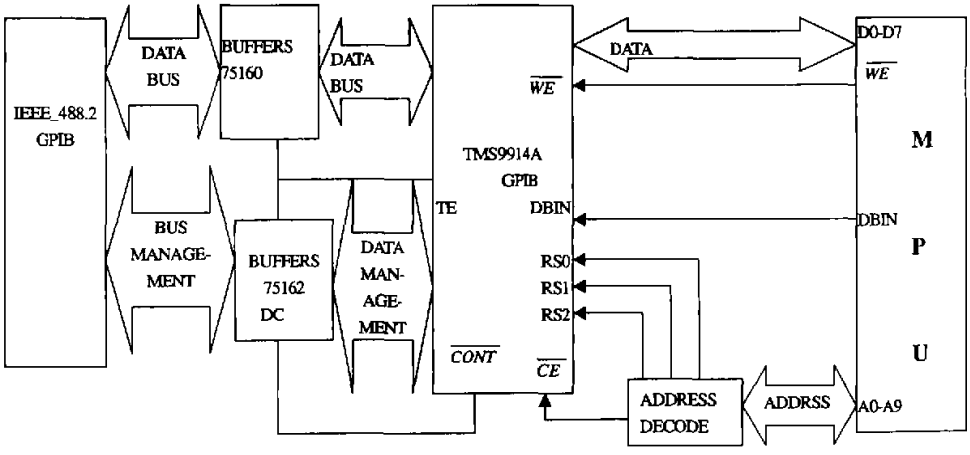


图 4-1 GPIB 电路原理框图

4.1.1 TMS9914 内部结构

本设计采用 TMS9914 作为接口母线适配器，它具有包括 C 功能在内的十种 标准接口功能，收发门使用的是 Texas Instrument 公司所生产的 S/N75160/1/2。

TMS9914A 是用 N—沟道硅栅工艺制造的，四十引脚，所有的输入/输出端，包括供电（ +5 伏 ）端都完全与 TTL 相容。它需要外接时钟(0.5 兆 ~ 5 兆赫兹)，该时钟可与微处理器系统的时钟无关。9914A 的内部结构如方框图 4 ▮2 所示。

从图中可以看出，TMS9914 的信号线分为两类，一类面向微处理器，共 19 条，其中：

片选线/CE，低电平有效；

写入线/WE，低电平有效；

读出线 DBIN，高电平有效；

中断请求线/INT，低电平有效；

DMA 请求线/ACCRQ 和 DMA 允许线/ACCGR，低电平有效；

复位线/RESET，低电平有效；

时钟线/CLOCK，时钟频率 500KHz ~ 5MHz；

3 条寄存器选择线 RS0 ~ RS2，配合读、写操作线用以选择内部 13 个寄存器；8 条双向数据线 D0 ~ D7，与微机数据总线相连；

以上这类信号线采用正逻辑。

另一类信号线面向 GPIB 母线，也是 19 条：

1 条讲允许线 TE，用作母线收发器的控制线，控制母线上数据传递的方向；

1 条系统控制线/CONT，表示该仪器是控者器件，能够控制 SRQ 线和 ATN 线；

1 条触发线 TR；

16 条信号线，与接口母线上的信号线相互对应；

这一类信号线则采用的是负逻辑，即：高于 2V 称为逻辑“0”态或“真态”，与 GPIB 母线所采用的逻辑关系一致，因此，不必倒向就可以直接与 GPIB 母线相连。

TMS9914 的内部寄存器 13 个，其中 6 个可读，7 个可写。这 13 个寄存器对母线和微处理器对母线和微处理器传来的消息进行处理。附表 1 和附表 2 列出了所有寄存器的名称、访问代码及内容。附录 1 为 7 个可写型寄存器，附录 2 为 6 个可读型寄存器。

其中，各个寄存器的作用如下：

(1) 6 个可读寄存器

中断状态寄存器 SRR0 和 SRR1 中共包括 14 种中断因素，除 INT0 和 INT1 这两位外，其它每一位都分别对应于中断屏蔽寄存器 SRW0 和 SRW1 的每一位。这些位在 MPU 读过后就被清除了。

寻址状态寄存器 SRR2 用于传达讲者/听者的寻址状态以及远控/本控和本地封锁的状态，这些信息由 TMS9914 的内部逻辑状态产生，可在读该寄存器时得到。

母线状态寄存器 SRR3 则显示 IEEE ▮488 母线中各管理线的状态。

命令通过寄存器 SRR6 用于把 TMS9914 不能自动处理的任何命令或副地址传到微处理器去。

数据输入寄存器 SRR7 用于暂存由 GPIB 母线传入 TMS9914 的数据。

(2) 7 个可写寄存器

中断屏蔽寄存器 SRW0 和 SRW1，只有先把中断屏蔽寄存器的某一位为 1 后，相应的中断状态才可能产生。

辅助命令寄存器 SRW3 能控制片内若干附加的能力，并提供了一种手段把某些本地消息输入到接口功能去，使能或不使能 TMS9914 的大多数特性，或者是对设备进行初始化。这些辅助命令共 23 条，当 Bit c/s =1 时，该能力使能；若 C/S=0；则该能力不使能；若 C/S=NA，那该位就应发送 0。

地址寄存器 SRW4 用来设置 TMS9914 的地址。

串行查询寄存器 SRW5 用来建立状态拜特，当控制器执行串行查询时，它被输出。

辅助命令寄存器 SRW3 能控制片内若干附加的能力，并提供了一种手段把某些本地

消息输入到接口功能去，使能或不使能 TMS9914 的大多数特性，或者是对设备进行初始化。这些辅助命令共 23 条，当 Bit c/s =1 时，该能力使能；若 C/S=0；则该能力不使能；若 C/S=NA，那该位就应发送 0。

地址寄存器 SRW4 用来设置 TMS9914 的地址。

串行查询寄存器 SRW5 用来建立状态拜特，当控制器执行串行查询时，SRW5 输出状态拜特。

并行查询寄存器 SRW6 存放当控制器执行并行查询时应输出的状态拜特。

数据输出寄存器 SRW7 用来暂存从 TMS9914 到母线的数据。

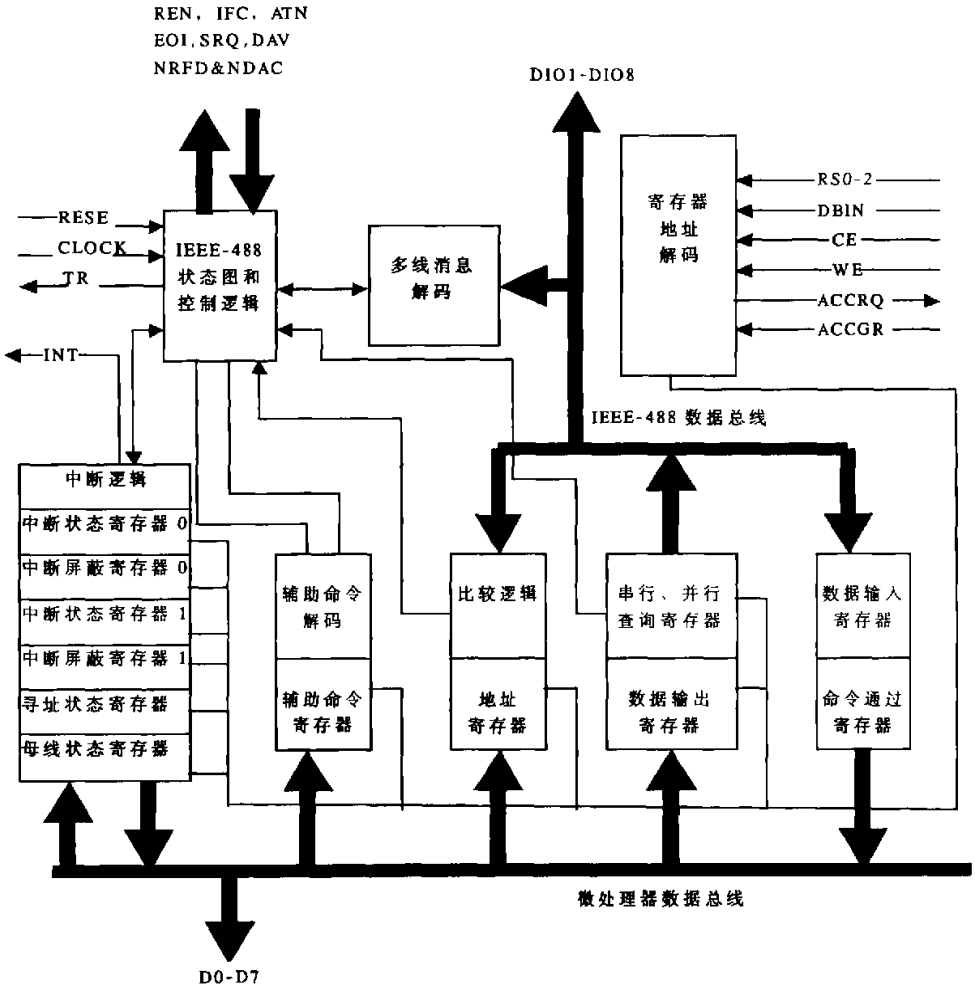


图 4-2　TMS9914 内部结构方框图

4.2 GPIB 基本函数及其功能实现

1. Igpib( )函数

功能：这个函数用于对 TMS9914 芯片进行初始化，初始化工作包括让其回复到已知的空闲状态，定义该仪器的地址，接口清除，中断屏蔽寄存器清零以及该仪器存在的标志_iGpib_Flag。

2. Output (int a)

功能：向指定的仪器以拜特单位发送信息。当我们向指定的仪器发送的信息被收到时，芯片就会收到返回信息。不断检测零号寄存器的值，当其 B0 位为 1 时，就表明该消息已成功发送。不断检测。

3. Enter ( )

功能：该函数负责接收讲者传送来的信息并且在接收信息时以一个拜特为单位进行接收。在接收数据时，如果超时则接收不到数据并且返回到空。

当接收到的是 CR、LF 或 EOI 时将停止接收数据。

4. Void lad (int d)

功能：发送”器件听地址”。d 为控制器的听地址。

5. Void tad (int d)

功能：发送”器件讲地址”，参数 d 为控制器的讲地址。

6. Void atnt ( )

功能：使 ATN 线变真，该函数无参数。

7. Void atnf ( )

功能：使 ATN 线变假，该函数无参数。

8. Void rent ( )

功能：使 REN 线变真，该函数无参数。

9. Void renf ( )

功能：使 REN 线变假，该函数无参数。

10。 Void ifct ( )

功能：使 IFC 线为真，该函数无参数。

11. Void ifcf ( )

功能：使 IFC 线为假，该函数无参数。

12. Int gpibremote (int cd) 接口远控函数

功能：cd 为 7 则使 REN 线为真。

Cd 为 7XX，其中 XX 为仪器的地址，则使地址为 XX 的仪器进入远控。

例如：

```
main ( )
 {
   gpibremote( 722 );
```

```
}
```

13. Unl ( )

功能：发送“不听(?)”接口消息，该函数无参数。

14. Unt ( )

功能：发送“不讲(--)”接口消息，该函数无参数。

15. Int gpibopen (int selcode)

功能：用于打开该仪器。如果 GPIB 卡是插在 ISA 总线的则将该仪器打开并初始化。

16. Void sendout (char *string)

功能：用于向指定的仪器发送消息。

17. Int inioutput(int devAdd)

功能：用于对控制器准备输出信息的初始化工作。参数 devAdd 为听者地址。函数返回值为 1，则成功初始化，为 0 则相反。

例如：

```
main ( )
       {inioutput (712);
        }
```

18. Int gpiboutput(int devAdd,char *outstr)

功能：用于仪器发送信息，参数 devAdd 为听者地址，*outstr 为需要发送的信息。返回值如果是 0，则未结束发送，为 1 则发送完毕。

例如：

```
main ( )
       {
          gpiboutput (712, " *IDN?);
          }
```

19. Int inigpibenter(int devAdd)

功能：用于控制器接收信息前的准备工作。参数 devAdd 为讲者地址。返回值如果为 1，则成功进行了初始化工作，为 0 则相反。

举例：

```
main ( )
{
   inigpibenter ( 712 );
   }
```

以上只是 GPIB 接口根据 C 语言的特点编制的对硬件直接操作的函数。在实际运用中往往是多个函数组合起来才能完成一定的功能。

例如：以单个拜特为单位接收一串数据为例

```
main ( )
       { int m;                                  /*初始化计数参数*/
         char data[100];                          /*存放收到的数据的变量*/
          igpib();                                /*初始化*/
          for(;;)
          { time_t enterafterstarttime1,enterafterendtime1;
            data[i]=enter();                      /*以拜特为单位接收数据*/
             if(recf==0x01)                       /*当所接收到的数据为 0x01
                                                    时，加结束符“\0”*/
            {
             data[i]='\0';
              break;
                }
          else if(recf==0x02)                     /*在接收到的数据为 0x02 时，加结
                                                  束符“\0”*/
            {
               data[i+1]='\0';
                 break;
               }
          else if(recf==0x04)                     /*在接收到的数据为 0x02 时，加
                                                  结束符“\0”*/
             {
               data[i+2]='\0';
               break;
              }
              i++;
     delay (300);                                 /*延时以接收完数据*/
    }
}
    }
```

以上这些代码是为了完成听功能。

4.3 ES4541 逻辑分析仪系统程控指令实现

程控消息是“控制器到器件”的消息。<程控消息>是一系列<程控消息单元>，代表一个“程控命令”。它们也可能是一个<询问消息单元>，代表一个“询问命令”。程控命令表示测量设备应实现的一个远控操作。为了实现远地操作，器件设计者必须为 GPIB 测量仪器设置一组程控命令（或说程控指令、程控数据）。在命令题头后紧跟随“？”号表示一个询问命令。在收听询问命令之后，仪器应该分析询问要求的功能，并作出“回答”。当控制器读取回答时，仪器讲出<响应消息>。

4.3.1 ES4541 逻辑分析仪程控消息译码

在 GPIB 接口中，接收完外部信息之后还应识别该信息的含义。为此，我们采取了查表的方式。把所有的命令列成一张表，如果该命令在此表内，我们就执行相应的命令。否则返回非法命令。

4.3.2 ES4541 逻辑分析仪程控指令集设计

1 系统数据命令组

系统数据命令组，系统数据命令给用户提供了从系统内器件获取更多信息的能力，在逻辑分析仪中即询问器件识别码。

IDN?

是器件必须响应的询问命令。该询问命令要求器件经总线以一个<ASCII 响应数据>元素传送器件的识别码。识别码用“,”分成三个场：

1 场：生产单位；要求的

2 场：产品型号；要求的

3 场：软件版本号；

例如：ES4541 的响应串是：

“UESTC，ES4541,Ver1.0”<NL>

2 内部操作命令

这些命令控制器件的内部操作。例如，测试、复位、读器件设置等。

REST

这是命令必须执行的命令，会使器件复位，它将使器件：

① 置器件功能到一个已知的初始状态

该命令不影响：

①GPIB 接口功能状态

②器件 GPIB 总线地址

③消息交换控制接口的“输出序列”

TST?

这是 GPIB 器件必须响应的询问命令，该命令要求器件进行一次内部自检，并报告是

否出错。在响应消息中也可有选择地说明出错原因。内部自检测试项目由器件设计者规定。

根据逻辑分析仪的特点，设计了以下的程控命令。现分别说明如下：

(1) :laset:latmod:latimng

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪工作在定时分析仪工作方式。

(2):laset:latmod:lastate

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪工作在状态分析仪工作方式。

(3):lafmt:polrty:positiv

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的探头极性为正极性

(4):lafmt:polrty:negativ

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的探头极性为负极性

(5):lafmt:clckup:upclock

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的时钟是上升沿有效。

(6):lafmt:clckup:downclk

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的时钟是下降沿有效。

(7):latrc:trcmod:Bsample

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的跟踪方式是起始触发。

(8):latrc:trcmod:Rsample

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的跟踪方式是随机触发。

(9):latrc:trcmod:Esample

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的跟踪方式是终止触发。

(10):trgwd:trgmod:lasries

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发方式为序列触发方式。

(11):trgwd:trgmod:lagroup

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发方式为组合触发方式。

(12):lalst:codmod:hexcode

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发字和限定字的码制为十六进制。

(13):lalst:codmod:bincode

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发字和限定字的码制为二进制。

(14):lalst:codmod:octcode

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发字和限定字的码制为八进制。

(15):lalst:codmod:strmode

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发字和限定字的码制为八进制。

(16):trgwd:trgsrs:n

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的触发级数，n 为不大于 8 的正整数。

(17):wvfrm:datrun

是器件必须响应的程控设置命令，无返回信息，其功能为使逻辑分析仪的波形窗口开始采集数据

(18):fmlst:datrun

是器件必须响应的程控设置命令，无返回信息，其功能为使逻辑分析仪的列表窗口开始采集数据

(19):latrc:trgvot:n

是器件必须响应的程控设置命令，无返回信息，其功能为设置逻辑分析仪的探头采集数据的触发电平。其中 n 为不大于 5 的任何正整数

(20):latmg:lafreq?

是器件必须响应的程控询问命令，其功能为当逻辑分析仪工作在定时分析仪的工作方式时，询问逻辑分析仪的工作频率。其返回值就为逻辑分析仪的频率值(:latmg:lafreq:n 其中 n 为 400M、200M、100M、50M、10M、5M、2M、1M、500K、200K、100K)。

(21):latmg:trgwd:tgwsit?

是器件必须响应的程控询问命令，其功能为当逻辑分析仪工作在定时分析仪的工作方式时，询问逻辑分析仪的触发字的位置。其返回值就为逻辑分析仪触发字的位置

(:latmg:trgwd:n 其中 n 为 0 ~ 32000 之间的任意正整数)。

(22):laset:latmod?

是逻辑分析仪必须响应的程控询问命令，其功能为询问当前逻辑分析仪的工作方式，工作在定时分析仪工作方式还是状态分析仪工作方式。其返回值就是定时分析(:laset:latmod:latimng)或状态分析仪(:laset:latmod:lastate)。

(23):latrc:trcmod?

是逻辑分析仪必须响应的程控询问命令，其功能为询问当前逻辑分析仪的跟踪方式，其返回值为随机触发(:latrc:trcmod:Rsample)、起始触发(:latrc:trcmod:Bsample)或终止触发(:latrc:trcmod:Esample)

第五章 ES4541 逻辑分析仪系统软件的调试

软件调试是软件设计过程中重要而又不可或缺的一步，它占据了软件设计的大部分工作。

在软件设计过程中我们可以借助 Borland C++ Builder 提供的集成调试环境对程序进行调试。

5.1 Borland C++ Builder 集成调试环境

无论编程人员多么认真和仔细，应用程序开发过程中不可避免地包含着错误，通过调试和跟踪就可以找出这些错误。在 Borland C++Builder 集成开发环境中继承了调试和跟踪工具，用于系统软件的调试，它具有下面功能：

- 控制系统软件的执行；
- 监视系统软件执行过程中变量和数据结构的变化；
- 通过调试器更改变量的值。

5.1.1 集成调试环境的设置

为了使用 Borland C++ Builder 的集成调试工具，调试系统软件要求在系统软件连接生成可执行文件过程中，必须产生应用程序的调试信息。其设置方法如下：

1 在主菜单中选择 Project|Option 菜单项，弹出 Project Options 对话框，选择 Linker 选项卡，在 Linking 选项卡组中，选中 Include debug information 复选框。

2 在主菜单中选择 Project|Make 菜单项，对工程文件进行编译和连接操作。

3 运行应用程序，如果在运行过程中发现应用程序包含有逻辑和运行时错误，可以通过设置断点来调试应用程序。

5.1.2 断点

在系统软件运行过程中，为了便于观察系统软件的运行情况，我们希望系统软件在某些代码处暂停执行，以便于检查系统软件运行时变量值的变化情况，以判定系统软件是否出现错误。在这种情况下需要使用断点。

在文本编辑器中，可以使用下面两种方法设置断点：

- 在文本编辑器中，用鼠标单击需要设置断点的代码行左侧空白区域，Borland C++ Builder 集成开发环境自动以红颜色点亮此代码行。
- 在文本编辑器中，将光标放到选定的代码行上，按下 F5 键。

在使用断点对应用程序进行调试的过程中，我们希望在一定条件下，断点才产生作用，因此应给断点设置一定的条件，使用条件断点。

5.1.3 系统软件错误类型

在编写系统软件代码时，产生了下面几种类型的错误：

1 编译时错误；

2 逻辑错误；

3 运行时错误。

编译时错误是由于系统软件代码违反了编程语言的语法规则。下面几种典型的情形都会产生此种错误：在书写代码时，缺少标点符号；函数调用时实参和形参类型不匹配，使用未声明的变量等。此种类型的错误在应用程序编译阶段就可以由编译器检查出来。

逻辑错误是这样一种类型的错误，从语法角度是完全合法的，但是其运行产生的结果不符合编程人员的设计思想。这种类型的错误一般是因为应用程序算法不完善而产生的。这种类型的错误一般难以检查。在 Borland C++ Builder 集成开发环境中，在运行时通过集成调试工具可以分析和检查应用程序的数据结构和变量值的变化，来找出逻辑错误发生的位置。

运行时错误和运行时的系统环境有密切的关系，如在应用程序运行过程中打开一个并不存在的文件时，就会产生此种类型的错误；又如在进行除法计算时除数为 0，也会出现此种类型的错误。此时计算机操作系统会自动终止应用程序的执行。通过使用集成调试工具运行至可能的错误处，然后单步运行应用程序，直到产生错误，通过此种方法可以发现产生错误的语句。

5.2 调试与纠错

在所有的错误类型中，比较难处理的是逻辑错误。在调试软件中其典型错误如下：

1 在实现波形的存储、打开功能后，反汇编的功能不能实现，后来经过测试发现在保存波形、打开文件时，改变了路径。

由于逻辑分析仪系统软件本身是一个比较完整的系统。当增加或减少功能时，有时会引起原来很好用的功能不能用。

以此为例，当将原来存储的文件打开，并以波形方式显示。再进行相应的反汇编功能时会出现内存读写错误这样的提示。要解决这个问题，首先应该了解反汇编的流程图(如图 5-1)。在进行实际的反汇编以前，我们预先将 CPU 指令按不同种类以不同的方式制成不同的*.Drv 的二进制文件，并加载进相应的反汇编程序中，每读进一个指令第一操作码，将其左移四位之值作为在反汇编指令表中的偏移地址，查得该指令的分类号，根据分类号的不同进行反汇编；特别值得一提的是关于指令前缀的处理，当分类号为一时，判断是否是操作数长度前缀或有效地址长度前缀，若是，则置位相应的前缀标志，在反汇编下一条指令的时候，如果发现操作数长度前缀标志或有效地址长度前缀标志被置位，则按照 32 位寻址方式进行反汇编，否则按照 16 位寻址方式进行反汇编，在完成该条指令的反汇编后将前缀标志复位。每完成一条指令的反汇编后，判断指令流是否结

束，如果指令流未结束则继续反汇编，否则输出反汇编结果，然后结束。其流程图如下：

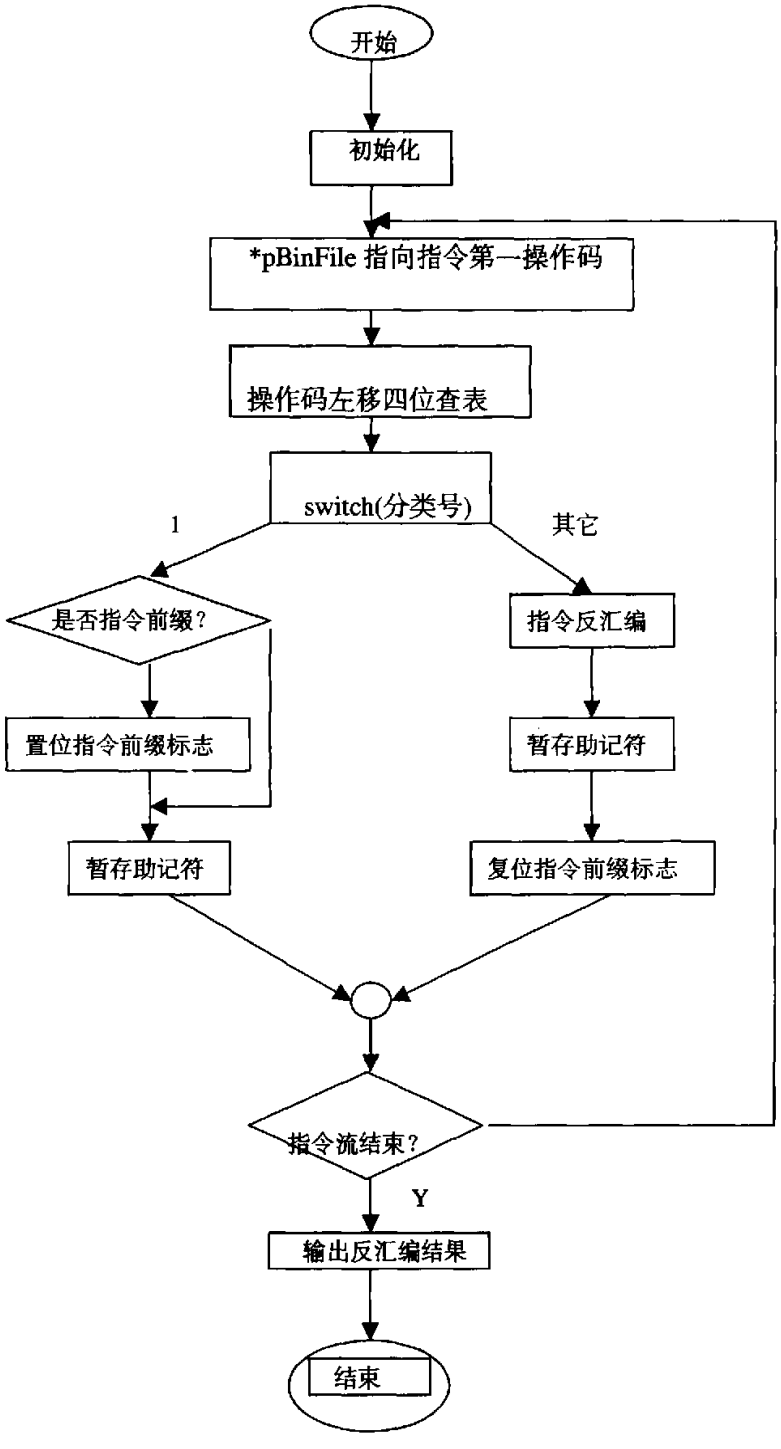


图 5-1 反汇编流程图

而驱动程序*.drv 可以保存在任何目录。因此，如果在原程序中指定驱动程序路径，当文件保存目录改变或访问路径改变时，势必将引起内存冲突、读写错误之类的提示。故为了增强程序的健壮性，我们定义了一个获取绝对路径的函数。当路径改变时，能正确地进行文件的读、写操作。

2 在实现逻辑分析仪波形分析的面板操作时，会出现这样一种情况，单步运行时，不会出现任何问题，而当整个工程进行编译时，就会出现错误，经过仔细检查发现可能是由于进行频繁的指针移动造成的。经过修改后问题得到了比较好的解决。

第六章 结束语

经过一年多的分析、论证、编程及调试，终于完成了本课题的设计任务。本课题是国防科工委“九五”重点科技型号项目“ES4541 逻辑分析仪”的软件设计部分，由于该逻辑分析仪采样率高，通道数目多，其系统软件实现的难度大，加之是“九五”型号项目，要求 2001 年底验收鉴定，因此时间比较紧迫。我从 2000 年底开始进入逻辑分析仪课题组，进行逻辑分析仪系统软件设计，在查阅了大量逻辑分析仪相关资料的基础上，归纳逻辑分析仪的特征，定义了适合于编程应用的对象模型，然后在此对象模型的基础上进行编程，完成了界面设计、在线帮助、存储限定字、逻辑分析仪面板控制、逻辑分析仪系统软件程控指令的设计以及系统软件的文档。逻辑分析仪软件不仅提供了布局合理、美观实用的用户操作界面，而且该软件与逻辑分析仪硬件组合为一体，构成了功能全面，操作简便、性能优越的高速逻辑分析仪系统。

2002 年 1 月 22 日，由信息产业部主持并组织专家在望江宾馆对该项目进行了技术鉴定。认为该成果与国外先进国家同类产品比较，达到了九十年代中期水平，并与当前国际市场主流产品水平相当，在国内属领先水平。

参考文献

1．张世箕,杨安禄,陈长龄．自动测试系统 .电子科技大学出版社，1990.48.59

2．张世箕，陈长龄，杨安禄．可程控电子设备接口技术.电子科技大学出版社，1990.20.45

3．李广军，王厚军．实用接口技术．成都：电子科技大学出版社．1997.22.35

4．齐治昌，谭庆平，宁洪．软件工程.高等教育出版社， 1997.209.215

5．何丕雁．CAT工作站硬件平台及驱动软件的设计：[硕士论文].成都：电子科技大学自动化学院 1997

6．顾乃　孙续．逻辑分析仪原理与应用．人民邮电出版社．1989．1.11

致 谢

首先，要感谢导师师奕兵副教授在毕业设计中给予的悉心指导和无私帮助，他的博学和睿智为我课题的最终完成提供了巨大的支持

其次，要感谢王厚军教授、田书林副教授及童玲副教授对我耐心细致的教导；

另外，还要感谢陈光 禹教授、顾亚平教授、徐建南高工和教研室的其他老师及同学给予的支持与帮助。

寄存器名	D0	D1	D2	D3	D4	D5	D6	D7
中断状态寄存器 0	INT0	INT1	BI	BO	END	SPAS	RLC	MAC
中断状态寄存器 1	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC
地址状态寄存器	REM	LLO	ATN	LPAS	TPAS	TDAS	LADS	uipa
母线状态寄存器	ATN	DAV	NDAC	NRFD	EOI	SRQ	IFC	REN
*								
*								
命令通过寄存器	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
数据输入寄存器	DIO8	DIO7	DIO6	DIO5		DIO3	DIO2	DIO1

附录1 TMS9914写寄存器

寄存器名	D0	D1	D2	D3	D4	D5	D6	D7
中断屏蔽寄存器 0	INT0	INT1	BI	BO	END	SPAS	RLC	MAC
中断屏蔽寄存器 1	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC
*								
辅助命令寄存器	cs	*	*	F4	EOI	SRQ	IFC	REN
地址寄存器								
串行查询寄存器								
并行查询寄存器	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
数据输出寄存器	DIO8	DIO7	DIO6	DIO5		DIO3	DIO2	DIO1

附录2　TMS9914读 寄 存 器

个人简历

本人生于 1974 年 7 月 25 日，于 1997 年在四川师范大学获物理学理学学士学位。

在电子科技大学读研究生期间，本人参加了国防科工委“九五”重点科技型号项目“ES4541 逻辑分析仪”的研究、开发工作，该项目已于 2002 年 1 月通过了由信息产业部技术鉴定。