2006 年第 8 期

电脑编程技巧与维护 月刊

每月3日出版

社　　长：孙茹萍

副 社 长：毕研元　田　真

总　　编：王路敬

编　　辑：袁　伟　姬振伟　程　芳

李晓鸿　管逸群　叶　永

公关部主任：苏加友

出版发行部：韦玉发

编 辑 出 版：《电脑编程技巧与维护》杂志社

主 管 部 门：中华人民共和国信息产业部

主 办 单 位：中国信息产业商会

社　　址：北京海淀区长春桥路 5 号

6 号楼 1209 室

投 稿 信 箱：gaojian@comprg.sina.net

gaojian@comprg.com.cn

编辑部信箱：gaojian@comprg.sina.net

发行部信箱：zzsfx@vip.sina.com

网　　址：http //www.comprg.com.cn

邮　　编：100089

电　　话： 010 82561037

传　　真： 010 82561614

照　　排：《电脑编程技巧与维护》

杂志社电脑排版部

印 刷 厂：世界知识印刷厂

订 阅 处：全国各地邮电局

国内总发行：北京报刊发行局

邮 发 代 号：82—715

国外发行代号：M6232

刊　　号：ISSN 1006 - 4052 / CN11 - 3411/TP

广告许可证：京海工商广字 0257

全 年 定 价：93.60 元

每 期 定 价：7.80 元

《电脑编程技巧与维护》书友会

书　　目	出版单位	原　价	折　扣	现　价
合订本系列				
1995 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	35.00	5 折	17.50
1997 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	60.00	5 折	30.00
1999 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	80.00	5 折	40.00
2001 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	98.00	5 折	49.00
2002 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	58.00	8 折	46.60
2003 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	59.00	8 折	47.20
2004 年《电脑编程技巧与维护》合订本	《电脑编程技巧与维护》杂志社	58.00	9 折	52.20
2003 年合订本 + 2004 年合订本	《电脑编程技巧与维护》杂志社	117.00	优惠	60.00
2001 - 2004 年合订本	《电脑编程技巧与维护》杂志社	273.00	优惠	100.00
2005 年合订本 (送 2005 年增刊)	《电脑编程技巧与维护》杂志社	94.00	优惠	58.00
编程技巧典型案例集锦系列				
Visual Basic 编程技巧典型案例解析	中国电力出版社 《电脑编程技巧与维护》	42.00		42.00
Delphi 编程技巧典型案例解析	中国电力出版社 《电脑编程技巧与维护》	42.00		42.00
C#编程技巧典型案例解析	中国电力出版社 《电脑编程技巧与维护》	42.00		42.00
Java 编程技巧典型案例解析	中国电力出版社 《电脑编程技巧与维护》	35.00		35.00
PowerBuilder 管理信息系统编程技巧典型解析	中国电力出版社 《电脑编程技巧与维护》	45.00		45.00
Visual C + + 编程技巧典型案例解析——基础与应用篇 (上)	中国电力出版社 《电脑编程技巧与维护》	39.00		39.00
Visual C + + 编程技巧典型案例解析——基础与应用篇 (下)	中国电力出版社 《电脑编程技巧与维护》	35.00		35.00
Visual C + + 编程技巧典型案例解析——图形图像处理与数据库篇	中国电力出版社 《电脑编程技巧与维护》	35.00		35.00
Visual C + + 编程技巧典型案例解析——网络与通信及计算机安全与维护篇	中国电力出版社 《电脑编程技巧与维护》	39.00		39.00
杂志月刊系列				
《电脑编程维护与技巧》月刊	《电脑编程技巧与维护》杂志社	7.80		7.80
《电脑编程维护与技巧》半年	《电脑编程技巧与维护》杂志社	46.80	9 折	42.12
《电脑编程维护与技巧》全年	《电脑编程技巧与维护》杂志社	93.60	8.5 折	79.60
增刊				
2004 年《计算机系统安全与维护编程实例》	《电脑编程技巧与维护》杂志社	36.00	8 折	28.80
2005 年《网络与通讯编程实例》	《电脑编程技巧与维护》杂志社	36.00	8 折	28.80

喜讯

亲爱的读者,你们好!

《电脑编程技巧与维护书友会》推出的所有图书都是有关电脑编程类的精品图书,欢迎广大读者选购。凡在杂志社《书友会》购买图书的读者,我们都造册登记,记录在案,自然成为《书友会》的会员,凡《书友会》的会员以后购买杂志社有关图书可以原有的折扣基础上再享受 9.5 折优惠。我们《书友会》以后准备定期举办电脑编程类精品图书的各种优惠活动,诚邀广大读者积极参与。

网上订购

www.dangdang.com　www.baotoo.com　www.shubaokan.com　www.dearbook.com　www.gotoread.com　www.huachu.com.cn

订阅方式:1) 北京四环以内的读者可以直接打电话到杂志社订购,杂志社免费送书上门。

2) 可通过邮局汇款方式购买,汇款时请将收件人姓名、通讯地址、邮编、金额等填写清楚,并在汇款附言中注明订阅的书名和份数。

3) 银行卡汇款:工商银行:4580 6010 2189 4853　持卡人:田真　交通银行:405512 8091 8032206　持卡人:田真

请在银行汇款后,及时通知杂志社您所购买的图书名称、数量、收件人姓名、地址。

通讯地址:北京市海淀区长春桥路 5 号 6 号楼 1209 室　邮编:100089

收款人:《电脑编程技巧与维护》杂志社发行部

电话/传真:010 82561037　010 82561614　订刊邮箱:zzsfx@vip.sina.com　网址:http://www.comprg.com.cn

微软为旧式电脑提供精简版 Windows 操作系统

微软正式宣布停止提供 Windows 98 和 Windows ME 系统之后，又于近日宣布，将为旧式电脑提供一款精简版的 Windows Fundamentals 操作系统。此操作系统是基于 Windows XP Service Pack 2 系统技术。据计算机网络的报告称，Windows Fundamentals 操作系统只能运行少数的几款程序，包括 IE 浏览器、Windows 媒体播放器及杀毒软件等，此操作系统可以在原始的 Pentium 处理器和仅有 64MB 的内存中运行。

据微软方面表示，Windows Fundamentals 操作系统主要是为那些仍使用旧式电脑的用户提供服务，其硬件将无法升级，但仍需要使用安全更新功能。

Skype 破解团队欲推新平台

“我们的兴趣不在于破解、复制或者盗版 Skype 软件，我们只是想研发一个更好的软件。”近日，接近破解 Skype 协议中国团队的核心相关人士给记者转发了这一团队对“破解”事件的表态和声明。

对于这一团队的动机，他们做了如上解释。声明同时称，我们通过向 Skype 学习，并从 Skype 中获得灵感，最终目的是创造一个社会公众都可以享受到高效、安全和免费通信的 P2P 平台，而这一平台将会比现在的 Skype 更加优秀。

其实自 Skype 出现开始，就有不少软件爱好者开始了对这一软件的研究和分析，其中也不乏一些技术天才在这方面小有成就，但如果真的有一款软件可以借鉴 Skype 的技术优势并且超越 Skype，这将对 Skype 形成巨大挑战。

不过，TOM 在线副总裁冯钰认为，这不会对 Skype 产生任何影响，她表示，其实国内外本来就有很多 VoIP 的软件，但它们都不具有 Skype 庞大的用户基础，也没有形成用户规模效应，此外，Skype 在技术上的优势仍然非常明显，“只有运营大规模的用户群才能检验技术的稳定性”。

这一团队表示，虽然外界对于他们这一做法有很多谣言，其中还有一些是负面的，但他们已经得到了投资者的巨大支持，所以他们希望能专心研究，而不是被外界的电话和邮件所打扰，因为他们是做研发的，而并不是明星。

VoIP 系统加重网络支持负担 外包服务是趋势

“VoIP 并不像灌篮那样简短有力，存在着不可预料的支持问题”，位于密苏里州 Saint Luke 堪萨斯市的 Saint Luke 的 Health System Inc. 的首席信息官 John Wade 这样说，该公司在三年前就开始部署超过 750 部 VoIP 电话，依靠 AT& T 公司提供维护和支持，据 Wade 说，每年的维护和支持成本为 7.5 万美元。

近期采访的其他六位 IT 经理都大致同意 Wade 的说法，VoIP 系统会带来更高的复杂度，这来自于终端用户对扩展功能的需求以及保证语音传输维护服务质量的需求。

据波士顿的 Yankee Group Research 公司分析师 Zeus Kerravala 估计，有 80% 的公司尝试自行管理其 VoIP 系统，不过他指出，一个新出现的趋势是有很多公司希望寻求外部帮助。

Gartner 公司分析师 Eric Goodness 指出，管理 IP 电话系统“堪称大象”，尤其是要在多个地点安装超过 500 个终端用户。他说，随着 VoIP 等应用的增长，管理的复杂度还将增加。

存储技术革命性突破　万能存储芯片 MRAM 问世

从摩托罗拉分拆出来的飞思卡尔公司最近宣布了一种新的磁性存储芯片。该项发明可能会改变移动设备的设计。上个月初，飞思卡尔发售了 4Mbit 的磁阻 RAM 芯片（MRAM），该产品有目前市面产品的优点，却没有他们的不足。 和以往芯片使用电荷存储消息不同，MRAM 使用磁场存储消息。

普通的 RAM 速度快，不但耗电而且一掉电就会丢失数据。一般数码相机和手机在闪存掉电后仍能保存数据，不过速度慢。而新型的 MRAM，速度快，掉电后不会丢失数据。因此部分支持者称它为“万能芯片”。如果真如所说的话，MRAM 可以取代电脑里面的内存，甚至是移动设备里的微硬盘。用到电脑上的话，电脑就可以马上开机了。

技术调查公司 Current Analysis 的分析师 Nicole d Onofrio：“毫无疑问，这是存储市场的一个革命性突破。它加速了静态 RAM 和非易失性闪存。”她指出，刚进入市场的 MRAM 还没有充分显示它的潜力。不过，MRAM 最终会在性能上超过目前的芯片，成为电脑和移动设备市场的热门选择。这可是一个大市场。单单闪存第一季度的销售额就达到 50 亿美元。

飞思卡尔表示，它和其他公司已经研究该项技术超过十年了。接下来的十八个月内，该公司打算将 MRAM 打进消费电子、汽车和更多市场。

韩国推出基于 TD 标准无线平台　成功通过测试

近日，据韩国媒体报道，一款由韩国商家开发的无线互联网平台预计将会被 TD - SCDMA 手机所采用，并将在中国市场上获得大范围应用。

韩国 XCE 公司致力于无线互联网解决方案的开发，本周二该公司表示，他们专门针对 TD - SCDMA 服务所开发的 XVM 平台已成功通过了中国移动、中国电信以及中国网通的商业化测试。

客户端就诊信息上传模块与业务外层的实现

杨 莉 汤嘉立

摘 要 简要介绍了案例的体系结构设计与就诊信息上传模块实现的功能、采用的技术及相关数据表，并详细讲解了就诊信息上传模块的实现过程。

关键词 .NET，WebServices，WinForm，DataGridq

一、引言

随着基于 .NET 的应用开发迅速普及，越来越多的开发人员及爱好者渴望学习 .NET 的开发技术。然而 .NET 下的技术众多，许多初学者开始学习时往往不知从何入手。为此，笔者写了基于 .NET 平台下的医疗保险系统开发系列，旨在通过案例向读者演示 .NET 平台下开发一个分布式应用程序项目使用到的技术及编程技巧帮助读者掌握 .NET 平台下开发分布式应用需使用到的体系结构设计方法，关键技术及一些编程方法。该案例从功能上分为三个模块，本篇主要讲解就诊信息上传模块的实现过程。

二、体系结构

案例采用分布式体系，分为客户端、中间逻辑层、后台数据库系统，其中间逻辑层设计成：业务外观层、业务规则层、数据访问层三层架构。具体如图 1 所示。

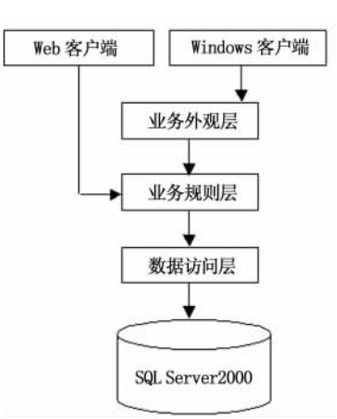


图 1 案例体系结构图

数据访问层为业务规则层提供数据访问服务。业务规则层包含各种商业逻辑的实现，如：计算药品报销金额，检查药品报销总额是否超过个人帐户上剩余总额等。业务外观层负责处理客户端数据处理和查询的请求，并通过调用业务规则层来完成这些请求。业务外观层用作隔离层，它将客户端与各种业务功能的实现隔离开来。客户端提供对应用程序的访问 负责与最终用户交互。

案例中基础数据维护与就诊信息上传模块均采用 Windows 客户端，通过创建 Windows 应用程序实现，统计查询模块采用 Web 客户端，通过创建 ASP. NET Web 应用程序实现，业务外观层通过创建 WebServices 项目实现，业务规则层与数据访问层通过创建类库项目实现，整个案例的实现放在一个解决方案文件 MSS 中。

三、模块功能技术及相关数据表

1. 实现的功能及采用的技术

就诊信息上传模块功能：根据药品编码查询药品的报销费用、将病人门诊医疗的详细信息实时上传至服务器。

该功能模块客户端使用 WinForm 技术开发 业务逻辑通过 WebServices 封装，供客户端调用，数据表操作采用 A-DO. NET 技术实现，封装在数据访问层 （一个类库项目）中，供业务逻辑层调用。

2. 相关数据表

该功能模块的实现，实际涉及到 6 张表，表 1 至表 4 在上一篇中已给出，故不再赘述。

四、模块的实现过程

以下主要详细讲解就诊信息上传模块 Windows 客户端、业务外观层的实现过程，有关业务规则层与数据访问层的实现将在后续篇中介绍。

1. 创建 MssService 项目 WebServices

表 5 DiagnosisSummary 就诊概况

字段名称	字段描述	主键	类型	长度	说明
DiagId	就诊编号	是	Int	非空	
PatID	病人编号		Int		非空,外键
HspId	医院编号		Int		非空,外键
ExpenseAmount	费用总额		Decimal	(18,4)	非空,费用总额=报销部分+自费部分
SubsidyAmount	报销部分		Decimal	(18,4)	非空
PaymentAmount	自费部分		Decimal	(18,4)	非空
DiagDate	就诊日期		DateTime		非空

表 6 DiagnosisDetail 就诊明细

字段名称	字段描述	主键	类型	长度	说明
DiagId	就诊编号	是	Int		非空,外键
DiagSubId	明细编号	是	Int	2	非空
MedName	药品名称		Nvarchar	4	非空
MedCode	药品编码		Char	4	非空,外键
MedUnitPrice	药品单价		Decimal	(18,2)	非空
MedQuantity	药品数量		Int	非空	
ExpenseSum	费用金额		Decimal	(18,4)	非空,费用金额=报销金额+自费金额
SubsidySum	报销金额		Decimal	(18,4)	非空
PaymentSum	自费金额		Decimal	(18,4)	非空

1) 启动 Visual Studio. NET 2003。

2) 选择菜单[文件][打开解决方案...]后，弹出[打开解决方案]对话框，[查找范围]选择“D hProjects”，双击显示栏内“MSS”文件夹，则显示该 MSS 文件夹内所有文件，[文件类型]选择“解决方案文件”，[文件名]文本框中输入“MSS”，则 Visual Studio. NET 打开“D hProjects hMSS”目录中 MSS 解决方案文件 (MSS 为上篇中所述步骤创建的解决方案文件)。

3) 为了让 MssService 项目创建到指定的目录里，这里需在 MMS 文件夹中新建文件夹“MssService”，然后创建 IIS 虚拟目录，选[开始]菜单栏->[控制面板]->[管理工具]->[Internet 信息服务]，在默认网站下创建虚拟目录“MssService”，该虚拟目录指向实际目录“D hProjects hMSS h MssService” 为方便记忆，这里使虚拟目录名与指向的实际目录同名)。

4) 在解决方案资源管理器中，右击“解决方案‘MSS’”，选择[添加][新建项目]后，弹出[添加新项目]对话框 将[项目类型]设置为“Visual C#项目”，[模板]设置为“ASP. NET Web 服务”，[位置]文本框中输入“http //localhost / MssService” (为上述步骤 3 中创建的 IIS 虚拟目录，[名称]文本框中默认值为“MssService”，然后单击[确定]按钮，则 Visual Studio. NET 在“D hProjects hMSS hMssService”目录中创建 MssService 项目文件。具体如图 2 所示。

图 2 创建 MssService 的[添加新项目]对话框

5 在解决方案资源管理器中，右击 MssService 项目下的“Service1. asmx”文件，选删除。右击“MssService”，选择[添加][添加新项]后 弹出[添加新项]对话框，将[模板]设置为“Web 服务”，[名称]文本框中输入“DiagnosisService. asmx” 然后单击[打开]按钮，则 Visual Studio. NET 在“D hProjects hMSS hMssService”目录中创建 DiagnosisService 文件。

6) 在解决方案资源管理器中，右击“MssService”，选择[添加引用] 弹出[添加引用]对话框，按“浏览...”按钮，弹出[选择组件]对话框，选择“D hProjects hMSS hBusinessRules h bin h Debug”目录下的“BussinessRules. dll 和“DataAccess. dll”组件 有关这两个组件的创建过程将在后续篇章三中详细讲解，本篇中读者只需从案例源代码中相关路径下将两个组件文件拷贝至项目文件夹下即可，在[添加引用]对话框中按“确定”按钮，即将两个组件添加至“MssService”项目下的“引用”中。

7 在解决方案资源管理器中，右击“DiagnosisService. asmx”文件，选择“查看代码”。

在文件中添加如下代码：

```
. . . . . .
//添加命名空间(DataAccess 和 BusinessRuels 两个组件)
using MSS. DataAccess;
using MSS. BusinessRules;
```

```
. . . . . .
//就诊规则属性
private DiagnosisRules DiagnosisRules
{
    get
    {
        return ( new DiagnosisRules( ) );
    }
}
/// <summary>
/// 计算药品报销金额
/// </summary>
/// <param name = "medCode"> 药品编码 </param>
/// <param name = "expenseSum"> 费用金额 </param>
/// <returns> </returns>
    [WebMethod]
public decimal CalcSubsidySum(string medCode, decimal ex-
penseSum)
{
        //调用就诊规则类中 CalcSubsidySum 方法, 并返回
        //药品报销金额
        return DiagnosisRules. CalcSubsidySum( medCode,
        expenseSum );
}
    /// <summary>
    /// 就诊信息上传
    /// </summary>
    /// <param name = "dsDiagInfo"> 就诊信息数据集 </
    //param>
    /// <returns> </returns>
[WebMethod]
public DataSet DiagnosisUpload(DataSet dsDiagInfo)
{
        bool flag = false;
        int patId;
        decimal subsidyAmount;
        //获取病人编号
        patId = (int) dsDiagInfo. Tables[" DiagnosisSumma-
ry"]. Rows[0]["PatId"];
        //获取报销总额
        subsidyAmount = (decimal) dsDiagInfo. Tables[" Di-
agnosisSummary"]. Rows[0]["SubsidyAmount"];
//调用就诊规则类中 ValidateSubsidyAmount 方法, 检查药品
//报销总额是否超过个人帐户上剩余总额
        flag = DiagnosisRules. ValidateSubsidyAmount(
        patId, subsidyAmount );
        if( flag )
        {
            try
            {
//调用就诊规则类中 AddDiagnosis 方法, 上传就诊信息至数
//据库中就诊概况表和就诊明细表
dsDiagInfo = DiagnosisRules. AddDiagnosis( dsDiagInfo );
                }
                catch(Exception ex)
                {//抛出异常
                    throw( ex );
                }
            }
            else
            {
throw( new Exception( "报销总额超出个人账户余额!" ) );
            }
            return dsDiagInfo;
        }
. . . . . .
```

8 从[文件]菜单中，选择[全部保存]以保存项目。 从[生成]中选择[生成解决方案]，Visual Studio. NET 会自动完成编译，并在“D hProjects hMSS hMssService hbin” 目录下生成一个名称为“MssService. dll”组件。

有关 WebServices 的部署过程系列一文章中已详细给出，不再赘述。要注意的是，MssService 中实际包含两个 Web 服务文件：MaintenanceServcie 文件供基础数据维护模块客户端调用，DiagnosisService 文件供就诊信息上传模块客户端调用。因此，在部署 WebServices 时需将案例源代码中“MSS hMssService” 目录下的 MaintenanceService. asmx、DiagnosisService. asmx、global. ascx 和 web. config 文件拷贝至创建的 IIS 虚拟目录中 注： MaintenanceService 文件的创建方法与 DiagnosisService 相似，详见案例源代码，不再赘述 。

2. 创建 WinUI 项目 (Windows 应用程序)

由于就诊信息上传模块与基础数据维护模块客户端皆为 Windows 客户端应用程序 故此将两个模块的客户端放在同一个 WinUI 项目中实现。在 WinUI 项目下创建两个文件夹“Maintenance”和“Diagnosis”，分别存放基础数据维护与就诊信息上传模块的窗体文件。就诊信息上传模块客户端由一个窗体文件构成。 注意：在创建之前，需先创建 MSS 数据库及部署 MssService，并假定读者已按系列一文章中所给步骤实现了基础数据维护模块的客户端 。

1) 在解决方案资源管理器中，右击“WinUI”，选择[添加 Web 引用] 弹出[添加引用]对话框，在[URL]栏中输入“http //localhost/MssService/DiagnosisService. asmx”，[Web 引用名]栏中输入“MssService. DiagnosisService”，然后单击[添加引用]按钮，添加完成对上述 WebServices 的引用。 这里假定 MssService 已创建并部署在本地机器上

2) 在解决方案资源管理器中，右击“WinUI”，选择[添加][新建文件夹]后，生成一新文件夹 NewFolder1，将其重命名为“Diagnosis” 该文件夹内存放就诊信息上传模块的窗体文件 。右击“Diagnosis”文件夹，选择[添加][添加 Windows 窗体]后 弹出[添加新项]对话框。在[名称]文本框中输入

“DiagnosisForm. cs” 单击[打开]按钮 则在“Diagnosis 文件夹下生成一窗体文件“DiagnosisForm. cs”。

3 在解决方案资源管理器中，双击“DiagnosisForm. cs”窗体文件，则该窗体出现在 Windows 窗体设计器中。在 Windows 窗体设计器中 右击该窗体，选择[属性] 则属性框显示。设置“FormBorderStyle”属性为“FixedDialog”、“MaximizeBox”属性为“False”、“MinimizeBox”属性为“False”、“Text”属性为“[就诊信息上传]”、“StartPosition”属性为“CenterScreen”。依次向 DiagnosisForm 窗体上添加相关控件，具体控件属性设置过程与系列一文章中所述步骤相似，详细情况可参看案例源代码，不再赘述。设置 DiagnosisForm 窗体，具体如图 3 所示。

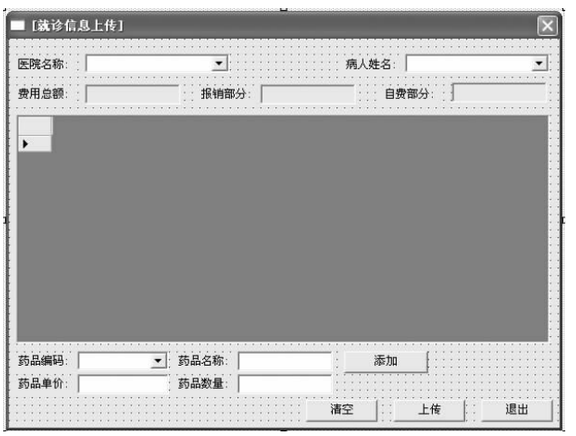


图 3 DiagnosisForm 窗体

4 在 Windows 窗体设计器中，分别双击 btnAdd 按钮、btnClear 按钮、btnUpload 按钮、btnEdit 按钮，添加按钮的 Click 事件处理程序。在解决方案资源管理器中，右击“DiagnosisForm. cs”文件，选择[查看代码]。在文件中添加如下代码：

```
. . . . . .
private MssService. MaintenanceService. MaintenanceService
maintenanceService;
private MssService. DiagnosisService. DiagnosisService diag-
nosisService;
private System. Data. DataSet dsMedicines;
private System. Data. DataSet dsHospitals;
private System. Data. DataSet dsPatients;
. . . . . .
public DiagnosisForm()
    {
        . . . . . .
        //实例化就诊服务代理类
        diagnosisService = new MssService.
        DiagnosisService. DiagnosisService();
        //实例化维护服务代理类
        maintenanceService = new MssService.
        MaintenanceService. MaintenanceService();
        //初始化就诊信息数据集
        InitDiagInfoDataSet();
        //获取数据表中记录
        GetDataSet();
    }
. . . . . .
/// <summary>
///设置 DiagnosisForm 为应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application. EnableVisualStyles();
    Application. Run(new DiagnosisForm());
}
//初始化就诊信息数据集
private void InitDiagInfoDataSet()
{
//构造就诊概况数据表
DataTable summary = new DataTable( "DiagnosisSummary" );
summary. Columns. Add( "DiagId", typeof(int) );
//就诊编号
summary. Columns. Add( "HspId", typeof(int) );
//医院编号
summary. Columns. Add( "PatId", typeof(int) ); //病人编号
summary. Columns. Add( "ExpenseAmount", typeof(decimal)
); //费用总额
summary. Columns. Add( " SubsidyAmount", typeof(decimal)
); //报销部分
summary. Columns. Add( " PaymentAmount", typeof
(decimal) ); //自费部分
summary. Columns. Add( " DiagDate", typeof(DateTime)
); //就诊日期
//就诊日期取当前系统日期
summary. Rows. Add( new object[] {0, 0, 0, 0, 0, 0,
System. DateTime. Now} );
//构造就诊明细数据表
DataTable detail = new DataTable("DiagnosisDetail");
detail. Columns. Add("DiagId", typeof(int)); //就诊编号
detail. Columns. Add("DiagSubId", typeof(int)); //明细编号
detail. Columns. Add( "MedCode", typeof(string) );
//药品编码
detail. Columns. Add( "MedName", typeof(string) );
//药品名称
detail. Columns. Add( "MedUnitPrice", typeof(decimal) );
//药品单价
detail. Columns. Add( "MedQuantity", typeof(int) );
//药品数量
```

```
    detail. Columns. Add( ″ExpenseSum″, typeof(decimal) );
    //费用金额
    detail. Columns. Add( ″SubsidySum″, typeof(decimal) );
    //报销金额
    detail. Columns. Add( ″PaymentSum″, typeof(decimal) );
    //自费金额
    //向就诊信息数据集中添加就诊概况数据表
    dsDiagInfo. Tables. Add(summary);
    //向就诊信息数据集中添加就诊明细数据表
    dsDiagInfo. Tables. Add(detail);
    }
    //设置 gridDetail 数据源
    private void SetGridDetail()
    {
        //设置 view 为就诊明细表的自定义视图
        DataView view = dsDiagInfo. Tables[″Diagnosis
        Detail″]. DefaultView;
        view. AllowDelete = false;
        view. AllowEdit = false;
        view. AllowNew = false;
        //设置数据源为视图
        gridDetail. DataSource = view;
    }
    //获取数据表中记录
    private void GetDataSet()
    {
    //调用维护服务代理类中 Web 方法 GetAllHospitals,
    //获取参保医院信息表中所有记录
    dsHospitals = maintenanceService. GetAllHospitals();
    //调用维护服务代理类中 Web 方法 GetAllPatients,
    //获取参保病人信息表中所有记录
    dsPatients = maintenanceService. GetAllPatients();
    //调用维护服务代理类中 Web 方法 GetAllMedicines,
    //获取报销药品目录表中所有记录
    dsMedicines = maintenanceService. GetAllMedicines();
    }
/// <summary>
/// 更新费用总额、报销部分、自费部分
/// </summary>
/// <param name = ″expenseSum″> 费用金额 </param>
/// <param name = ″subsidySum″> 报销金额 </param>
/// <param name = ″paymentSum″> 自费金额 </param>
private void UpdateAmount(decimal expenseSum, decimal
subsidySum, decimal paymentSum)
{
    DataRow row = dsDiagInfo. Tables[ ″ DiagnosisSumma-
ry″]. Rows[0];
    row[″ExpenseAmount″] = (Decimal)row
[″ExpenseAmount″] + expenseSum;
    row[″SubsidyAmount″] = (Decimal)row
[″SubsidyAmount″] + subsidySum;
    row[″PaymentAmount″] = (Decimal)row
[″PaymentAmount″] + paymentSum;
txtExpenseAmount. Text = row[″ExpenseAmount″]
. ToString();
    txtSubsidyAmount. Text = row[″SubsidyAmount″]
. ToString();
    txtPaymentAmount. Text = row[″PaymentAmount″]
. ToString();
}
private void DiagnosisForm_Load(object sender, System.
EventArgs e)
{
    //设置 gridDetail 数据源
    SetGridDetail();
    //设置数据源为参保医院信息表
    cbHspId. DataSource = dsHospitals. Tables[0];
    //设置显示内容的数据源属性为医院名称
    cbHspId. DisplayMember = ″HspName″;
    //设置取值的数据源属性为医院编号
    cbHspId. ValueMember = ″HspId″;
    //设置数据源为参保病人信息表
    cbPatId. DataSource = dsPatients. Tables[0];
    //设置显示内容的数据源属性为病人姓名
    cbPatId. DisplayMember = ″PatName″;
    //设置取值的数据源属性为病人编号
    cbPatId. ValueMember = ″PatId″;
    //设置数据源为报销药品目录表
    cbMedCode. DataSource = dsMedicines. Tables[0];
    //设置显示内容的数据源属性为药品编码
    cbMedCode. DisplayMember = ″MedCode″;
    //设置取值的数据源属性为药品编码
    cbMedCode. ValueMember = ″MedCode″;
}
private void btnAdd_Click(object sender, System.
EventArgs e)
{
//获取药品编码
string medCode = cbMedCode. SelectedValue. ToString();
//获取药品名称
string medName = txtMedName. Text;
//获取药品单价
decimal medUnitPrice = Decimal. Parse( txtMedUnitPrice.
Text );
//获取药品数量
int medQuantity = Int32. Parse( txtMedQuantity. Text );
//计算费用金额
decimal expenseSum = medUnitPrice * medQuantity;
//调用就诊服务代理类中 Web 方法 CalcSubsidySum,
//计算报销金额
decimal subsidySum = diagnosisService. CalcSubsidySum
( medCode, expenseSum );
//计算自费金额
decimal paymentSum = expenseSum – subsidySum;
//向 DataRowCollection 中添加一行记录
dsDiagInfo. Tables[″DiagnosisDetail″]. Rows. Add(new
```

```
object[] {0, 0, medCode, medName, medUnitPrice,
medQuantity, expenseSum, subsidySum, paymentSum }
);
//更新费用总额、报销部分、自费部分 TextBox 控件
UpdateAmount(expenseSum, subsidySum, paymentSum);
}
private void btnClear_Click(object sender, System. EventArgs e)
{
//设置控件的 Text 属性为空
txtExpenseAmount. Text = "";
txtSubsidyAmount. Text = "";
txtPaymentAmount. Text = "";
//清除该表所有行的集合
dsDiagInfo. Tables["DiagnosisSummary"]. Rows. Clear();
//向该表中添加一行记录
dsDiagInfo. Tables[" DiagnosisSummary"] . Rows. Add( new
object[] {0, 0, 0, 0, 0, 0, System. DateTime. Now } );
//清除该表所有行的集合
dsDiagInfo. Tables["DiagnosisDetail"]. Rows. Clear();
//重新设置 gridDetail 数据源
SetGridDetail();
}
    private void btnUpload_Click(object sender, System.
EventArgs e)
{
    DataRow row = dsDiagInfo. Tables[" DiagnosisSumma-
ry"]. Rows[0];
    //获取并设置医院编号
    row["HspId"] = Int32. Parse(cbHspId. SelectedValue.
ToString());
    //获取并设置病人编号
    row["PatId"] = Int32. Parse(cbPatId. SelectedValue.
ToString());
    try
    {
    //调用就诊服务代理类中 Web 方法 DiagnosisUpload,
    //将就诊信息上传至就诊概况表和就诊明细表
 dsDiagInfo = diagnosisService. DiagnosisUpload(dsDiagInfo);
    //获取就诊编号
    int diagId = (int) dsDiagInfo. Tables[" DiagnosisSumma-
ry"]. Rows[0]["DiagId"];
    //显示就诊编号
    MessageBox. Show( String. Format(" 就诊信息上传成功,
就诊编号为: {0}! ", diagId) );
    }
    catch(Exception ex) //捕获异常
    {
        //显示异常描述信息
        MessageBox. Show( ex. Message. ToString());
    }
}
    private void btnExit_Click(object sender, System.
EventArgs e)
{
    //关闭窗体
    Close();
}
```

5 在解决方案资源管理器中，右击“WinUI” 选择[设为启动项目]，选[生成][重新生成解决方案]，选[调试][启动]，弹出一窗体，在“药品编码”栏选择“1002”，“药品名称”栏输入“白加黑”，“药品单价”栏输入“50. 5”，“药品数量”栏输入“3”，单击“添加”按钮，运行如图 4 所示 单击“上传”按钮，则弹出一信息提示框，如图 5 所示，提示就诊信息上传成功并显示病人就诊编号 根据就诊编号可查询个人就诊明细，该功能实现将在后续篇中介绍 。

图 4 DiagnosisForm 运行窗体

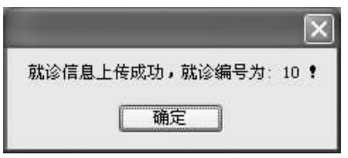


图 5 信息提示对话框

五、结束语

本篇详细讲解了就诊信息上传模块的实现过程及采用的技术。下一篇中将讲解业务规则层与数据访问层的实现。

参考文献

1. David Richard Kalkstein DeLoveh. William Sempf. Visual Studio . NET 高效编程[M]. 北京清华大学出版社 2002

2. Jeffrey Richter. . NET 框架程序设计[M]. 北京清华大学出版社，2003

收稿日期：2006 年 4 月 15 日

在 . NET 环境中使用 C#调用非托管 DLL

黄 猛 唐 琳

摘 要 讲述了在 C#程序中调用 Win32 API 函数以及使用 Win32 编写的 DLL，并给出了实例。

关键词 C#，Win32，API，DLL，托管和非托管

一、. NET 技术和 C#语言

Microsoft. NET 作为一种全新的开发平台，已经渐渐成为主流技术。. NET 是一种面向网络、支持各种用户终端的开发平台，它提供了统一的命令集支持任何编程语言，使得应用开发与语言无关。

C#是一种现代的面向对象的程序开发语言，它使程序员能够在 . NET 平台上快速开发丰富的应用程序。C#是真正的面向对象的语言，而且它还能为C + + 程序员提供快捷的开发方式。

1. 公共语言运行库 common language runtime

托管代码执行核心中的引擎。运行库为托管代码提供各种服务，如跨语言集成、代码访问安全性、对象生存期管理、调试和分析支持。

2. 托管代码 managed code

由公共语言运行库环境（而不是直接由操作系统）执行的代码。托管代码应用程序可以获得公共语言运行库服务，例如自动垃圾回收、运行库类型检查和安全支持等。这些服务提供独立于平台和语言的、统一的托管代码应用程序行为。

3. 非托管代码 unmanaged code

在公共语言运行库环境的外部，由操作系统直接执行的代码（例如 Win32 API）。非托管代码必须提供自己的垃圾回收、类型检查、安全支持等服务；它与托管代码不同，后者从公共语言运行库中获得这些服务。

二、调用非托管的 DLL

使用了 C#语言，如何使用 Win32 API 函数和非 . NET 环境下编写的代码？. NET 的一个最主要的优势是它的语言无关特性，可以使用 Visual Basic、C + + 、C#等语言来编写类，然后在其它语言中使用，甚至可以用不同的语言来派生类。本文将介绍如何从 C#代码中调用非托管 DLL。

如何调用 Win32 环境下开发的非托管 DLL 呢？方法是必须将 . NET 对象转化成结构、char * 以及 C 语言的指针。也就是参数必须被列集（marshal）。

若要声明一个方法使其具有来自 DLL 导出的实现，执行下列操作：

- 使用 C# 关键字 static 和 extern 声明方法。
- 将 DllImport 属性附加到该方法。DllImport 属性允许您指定包含该方法的 DLL 的名称。通常的做法是用与导出的方法相同的名称命名 C# 方法，但也可以对 C# 方法使用不同的名称。
- 还可以为方法的参数和返回值指定自定义封送处理信息，这将重写 . NET Framework 的默认封送处理。

1. 直接从 C# 调用 DLL 导出

为了从 C# 中调用 DLL 函数，首先必须要有一个声明，就象使用 C 或C + + 的程序一样，只不过在 C#中使用的是 DllImport 关键字，下面的代码显示用 DllImport 属性通过调用 msvcrt. dll 中的 puts 输出消息。

```
using System;
using System. Runtime. InteropServices;
class PlatformInvokeTest
{
    [DllImport(″msvcrt. dll″)]
    public static extern int puts(string c);
    [DllImport(″msvcrt. dll″)]
    internal static extern int _flushall();
    public static void Main()
    {
      puts(″Test″);
      _flushall();
    }
}
```

上面这段代码显示了声明在非托管 DLL 中实现的 C# 方

法的最低要求。PlatformInvokeTest. puts 方法用 static 和 extern 修饰符声明并且具有 DllImport 属性，该属性使用默认名称 puts 通知编译器此实现来自 msvcrt. dll。在 C#中，DllImport 关键字作用是告诉编译器入口点在哪里，若要对 C# 方法使用不同的名称 （如 putstring），则必须在 DllImport 属性中使用 EntryPoint 选项，代码如下：

```
[DllImport(″msvcrt. dll″, EntryPoint = ″puts″)]
```

2. 默认封送处理和为非托管方法的参数指定自定义封送处理

当从 C# 代码中调用非托管函数时，公共语言运行库必封送参数和返回值。

对于每个 .NET Framework 类型均有一个默认非托管类型，公共语言运行库 （CLR） 将使用此非托管类型在托管到非托管的函数调用中封送数据。例如，C# 字符串值的默认封送处理是封送为 LPTSTR （指向 TCHAR 字符缓冲区的指针） 类型，例如下面的代码调用了 USER32. DLL 中的 SetWindowText 和 GetWindowText 函数：

```
using System;
using System. Drawing;
using System. Text;
namespace Win32API {
 public class Win32 {
[DllImport(″User32. Dll″)]
   public static extern int GetWindowText(int hwnd,
    StringBuilder buf, int nMaxCount);
[DllImport(″User32. Dll″)]
   public static extern bool SetWindowText(int h, String s);
    . . . . . .
 }
}
```

编译器知道在 user32. dll 中找到 SetWindowText，并在调用前自动将 Strnig 对象转换为 LPTSTR TCHAR * 。. NET 是如何实现的呢？其实，每一个 C#类型都有一个缺省的列集类型。对于 Stirng 来说，它的列集类型就是 LPTSTR。但如果调用的是 GetWindowText，它的 String 参数是一个输出参数，而非输入参数，因为 String 对象是不可改变的，再象上面这样处理就行不通了。每次使用 String 对象时，都要在内存中创建一个新的字符串对象，这就需要为该新对象分配新的空间。在需要对字符串执行重复修改的情况下，与创建新的 String 对象相关的系统开销可能会非常昂贵。如果要修改字符串而不创建新的对象，则必须使用 System. Text . StringBuilder 类，例如上面声明 GetWindowText 的代码。

StringBuilder 缺省的列集类型是 LPTSTR，但是 GetWindowText 现在可以修改实际的 String 对象：

```
    int hwnd = // get it. . .
    StringBuilder sb = new StringBuilder(256);
    Win32. GetWindowText(hwnd, sb, sb. Capacity);
```

如果缺省的列集类型不是想要的类型，如何处理？例如需要传递的参数是 LPSTR char * ，甚至是 UNICODE 串。如果传递一个串，缺省情况下公共语言运行时 （CLR） 将把它转换成 TCHARs，解决办法是使用 MarshalAs 属性重写函数参数的默认封送处理。

```
using System;
using System. Runtime. InteropServices;
class PlatformInvokeTest
{
  [ DllImport(″msvcrt. dll″)]
   public static extern int puts([MarshalAs( UnmanagedType. LPStr)]string m);
  [DllImport(″msvcrt. dll″)]
  internal static extern int _flushall();
  public static void Main()
  {
     puts(″Hello World! ″);
     _flushall();
  }
}
```

在上面的代码中，puts 函数的参数的默认封送处理已从默认值 LPTSTR 重写为 LPSTR。MarshalAs 属性可以放置在方法参数、方法返回值以及结构和类的字段上。若要设置方法返回值的封送处理，请将 MarshalAs 属性与返回属性位置重写一起放置在方法上的属性块中。例如，若要显式设置 puts 方法返回值的封送处理：

```
. . . . . .
[ DllImport(″msvcrt. dll″)]
[ return: MarshalAs(UnmanagedType. I4)]
public static extern int puts(
. . . . . .
```

3. 为用户定义的结构指定自定义封送处理

可以为传递到非托管函数或从非托管函数返回的结构和类的字段指定自定义封送处理属性。通过向结构或类的字段中添加 MarshalAs 属性可以做到这一点。还必须使用 StructLayout 属性设置结构的布局，还可以控制字符串成员的默认封送处理，并设置默认封装大小。

下面是 MFC 中的结构：

```
typedef struct tagLOGFONT
{
 LONG lfHeight;
 LONG lfWidth;
 LONG lfEscapement;
 LONG lfOrientation;
 LONG lfWeight;
 BYTE lfItalic;
```

```
  BYTE lfUnderline;
  BYTE lfStrikeOut;
  BYTE lfCharSet;
  BYTE lfOutPrecision;
  BYTE lfClipPrecision;
  BYTE lfQuality;
  BYTE lfPitchAndFamily;
  TCHAR lfFaceName[LF_FACESIZE];
} LOGFONT;
```

那么，如何在 C#中使用它呢，方法是使用 StructLayout 和 MarshalAs 属性重新描述前面的结构，如下代码：

```
using System;
using System. Runtime. InteropServices;
[StructLayout(LayoutKind. Sequential)]
public class LOGFONT
{
  public const int LF_FACESIZE  = 32;
  public int lfHeight;
  public int lfWidth;
  public int lfEscapement;
  public int lfOrientation;
  public int lfWeight;
  public byte lfItalic;
  public byte lfUnderline;
  public byte lfStrikeOut;
  public byte lfCharSet;
  public byte lfOutPrecision;
  public byte lfClipPrecision;
  public byte lfQuality;
  public byte lfPitchAndFamily;
  [MarshalAs(UnmanagedType. ByValTStr, SizeConst =
LF_FACESIZE)]
  public string lfFaceName;
}
```

现在，就可以将此结构用于 C#中了，添加如下代码：

```
using System;
using System. Runtime. InteropServices;
class PlatformInvokeTest
{
  [DllImport("gdi32. dll", CharSet = CharSet. Auto)]
  public static extern IntPtr CreateFontIndirect(
  [In, MarshalAs(UnmanagedType. LPStruct)]
  LOGFONT lplf  // characteristics);
  [DllImport("gdi32. dll")]
  public static extern bool DeleteObject(IntPtr handle);
  public static void Main()
  {
    LOGFONT lf  = new LOGFONT();
    lf. lfHeight  = 9;
    lf. lfFaceName  = "Arial";
    IntPtr handle  = CreateFontIndirect(lf);
    if (IntPtr. Zero  = = handle){
      Console. WriteLine("Can't creates a logical font. ");
    }
    else{
     if (IntPtr. Size  = = 4)
       Console. WriteLine("{0: X}", handle. ToInt32());
     else
       Console. WriteLine("{0: X}", handle. ToInt64());
     // Delete the logical font created.
     if (!DeleteObject(handle))
     Console. WriteLine("Can't delete the logical font");
    }
  }
}
```

在上面的代码中，CreateFontIndirect 方法使用了一个 LOGFONT 类型的参数。MarshalAs 和 In 属性用于限定此参数。程序将由此方法返回的数值显示为十六进制大写字符串。

4. 注册回调方法

如何将回调从 C#传递到非受管代码呢？只要用委托 delegate 即可，以 Win32 API 函数 EnumWindows 为例，它的作用是枚举系统中的所有现有窗口，并调用回调函数来对每个窗口执行一项任务，首先声明委托 /回调类型：

```
delegate bool EnumWindowsCB(int hwnd, int lparam);
```

一旦声明了委托 /回调类型，就可以象下面这样打包：

```
iDllImport("user32")]
public static extern int
EnumWindows(EnumWindowsCB cb, int lparam);
```

上面的 delegate 仅仅是声明了一个委托类型，还必须在提供一个实际的委托实现：

```
public static bool MyEWP(int hwnd, int lparam) {
  // do something
  return true;
}
```

然后对它进行打包处理：

```
EnumWindowsCB cb  = new EnumWindows
CB(MyEWP);
Win32. EnumWindows(cb, 0);
```

注意 这里掩饰了 lparam 的问题，在 C 中，如果你给 EnumWindows 一个 LPARAM，则 Windows 会用它通知回调函数。一般典型的 lparam 是一个结构或类指针，其中包含着所需要的上下文信息。在 .NET 中绝对不能提到“指针”！可以将 lparam 声明为 IntPtr 并用 GCHandle 对它进行打包：

```
// 现在 lparam 是 IntPtr
delegate bool EnumWindowsCB(int hwnd, IntPtr lparam);
// 在 GCHandle 中打包对象
MyClass obj  = new MyClass();
GCHandle gch  = GCHandle. Alloc(obj);
```

(下转第 23 页)

家庭防盗报警控制器设计

王 阳

摘 要 基于单片微处理器 PIC16F84 研制开发的简易型家庭防盗报警智能控制器，给出控制系统的硬件电路和软件程序，并在硬件设计上，阐述了控制器工作原理。

关键词 PIC 单片机，A/D 转换，中断，撤防

随着人们生活水平的不断提高，家庭安全服务正在逐渐引起人们的重视，在家庭住宅处安装设置防盗报警系统，使住宅处能对住房的特发警情进行及时处理，正是人们所期盼的。在家庭安装简易防盗报警系统，系统以微控制器为核心，对住宅进行安全监视和服务，具有可靠性高，成本低、易普及等优点。

一、系统简介

1. 组成：系统主要由 CPU、遥控发射、接收、解码、报警传感输入、输出控制等电路组成。CPU 采用 PIC16F84，为了减少工艺，遥控发射和接收均采用成品板，并采用遥控发射自带编码型的遥控器。解码电路采用与遥控器编码 PT2262 对应的 PT2272 芯片，报警传感输入可分为门磁开关，紧急按钮，红外热释防护栅栏等，本系统共设有二路开关型输入口（可随意扩展）和三路传感器输入口。LED1 用作布防指示灯，LED2 用于电源工作指示灯。LED3 用于警讯输入指示灯。

2. 工作原理：(1)加电后系统复位，将进入等待遥控器工作指令。遥控器可以实现布防、撤防、报警三种功能。当遥控器发出布防指令，解码片 PT2272 第 11（D2）脚输出高电平，CPU 检测到 RB2 高电平后，程序进入布控状态，此时，CPU 将不循环检测 RB3—RB7 报警传感输入口的电平状态，当检测到某一路输入口为低电平时，并通过软件抗干扰确认为警情信号后，将发出一分钟的警声和警灯信号，同时 LED3 指示灯亮；一分钟后警情未解除，将继续发出警报，直至警情解除或撤防。(2)当遥控器发出撤防指令后，解码片 PT2272 第 13（D0）脚输出高电平，CPU 的 RB0 检测到高电平后将解除布防状态，并可解除正在报警状态。撤防后，即使各传感输入口有警情发生，都不会发出警报。(3)当遥控器发出报警指令后，PT2272 第 12（D1）脚输出高电平，CPU 的 RB1 检测到高电平后将发出报警和警灯，一分钟后自动停止。

3. 硬件电路设计，如图 1 所示。

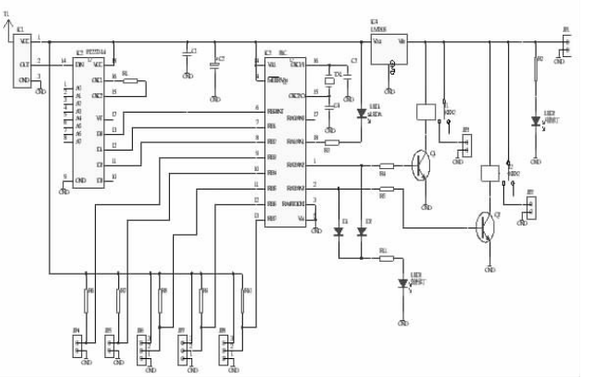


图 1 硬件电路设计

二、系统程序

1. 系统程序流程图，如图 2 所示。

2. 系统程序清单

```
; * * * * * * * * * * * * * * * * * * * * * * * * *
;简易型家庭防盗报警器 YY－808. ASM
; * * * * * * * * * * * * * * * * * * * * * * * * *
    LIST  P＝PIC16F84
    #INCLUDE  ＜P16F84. INC＞
    ;
tmr0  equ  01h  ;
  pcl  equ  02h  ;
  status  equ  03h  ;
  fsr  equ  04h  ;
```

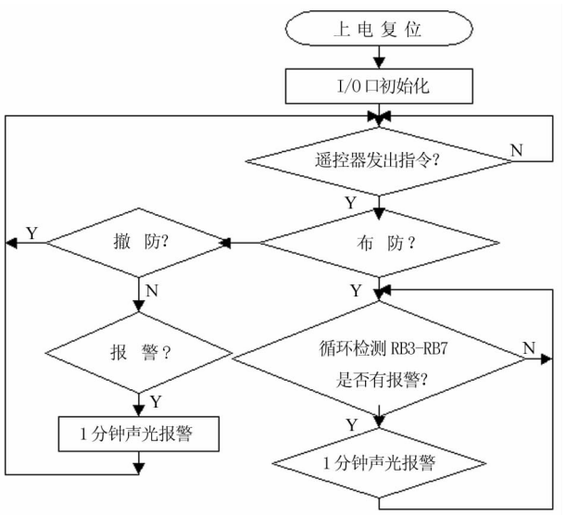


图 2 系统程序流程图

```
        porta   equ   05h    ;
        portb   equ   06h    ;
        ;             ;
        trisa   equ   85h    ;
        trisb   equ   86h    ;
        ;             ;
        data0   equ   10h    ; 05s 外层循环次数寄存器
        data1   equ   11h    ; 05s 内层循环次数寄存器
        data2   equ   12h    ; 警声时间寄存器
        data3   equ   13h    ; 1s 外层循环次数寄存器
        data4   equ   14h    ; 1s 内层循环次数寄存器
        data5   equ   15h    ; 03s 外层循环次数寄存器
        data6   equ   16h    ; 03s 内层循环次数寄存器
;* * * * * * * * * * * * * * * * * * * * * * * *
        org   0          ;复位入口
        goto  main       ;
;.....................................................
 main  bsf   status, 5    ; 选体 1
        movlw  b´00000000´    ; 置 RA0 – RA4 为输出
        movwf  trisa       ;
        movlw  b´11111111´    ; 置 RB0 – RB7 为输入
        movwf  trisb       ;
        bcf   status, 5    ; 恢复体 0
        clrf   porta       ;
        clrf   portb       ;
        bsf   5, 1         ;
 loop  btfss  6, 2         ; 设防
        goto  l1           ;
        call  delay350     ;
        btfss  6, 2        ;
        goto  l1           ;
        call  plsp         ;
        goto  ce           ;
 l1    btfss  6, 1         ; 警声
        goto  l2           ;
        call  delay350     ;
        btfss  6, 1        ;
        goto  l2           ;
        call  ps           ;
 l2    btfss  6, 0         ; 撤防
        goto  loop         ;
        call  delay350     ;
        btfss  6, 0        ;
        goto  loop         ;
 l3    call  plsp         ;
        call  delay05      ;
        call  plsp         ;
        bsf   5, 1         ;
        goto  loop         ;
;.....................................................


 ps    ; 1 分钟警声子程序
        movlw  . 140       ;
        movwf  data2       ; 警声时间寄存器
        bsf   5, 2         ;
        bsf   5, 3         ;
 pl1   call  delay1s       ;
        decfsz  data2, 1    ; 递减, 为 0 则跳
        goto  pl1          ;
        bcf   5, 2         ;
        bcf   5, 3         ;
        return             ;
;* * * * * * * * * * * * * * * * * * * * * * * *
 ce    ; 设防子程序
        bcf   5, 1         ;
 ce1   btfss  6, 3        ;
        goto  c1           ;
        call  delay350     ;
        btfss  6, 3        ;
        goto  c1           ;
        call  ps           ;
 c1    btfss  6, 4        ;
        goto  c2           ;
        call  delay350     ;
        btfss  6, 4        ;
        goto  c2           ;
        call  ps           ;
 c2    btfss  6, 5        ;
```

```
        goto   c3          ;
        call   delay350    ;
        btfss  6, 5        ;
        goto   c3          ;
        call   ps          ;
  c3    btfss  6, 6        ;
        goto   c4          ;
        call   delay350    ;
        btfss  6, 6        ;
        goto   c4          ;
        call   ps          ;
  c4    btfss  6, 7        ;
        goto   c5          ;
        call   delay350    ;
        btfss  6, 7        ;
        goto   c5          ;
        call   ps          ;
  c5    btfss  6, 1        ; 警声
        goto   c6          ;
        call   delay350    ;
        btfss  6, 1        ;
        goto   c6          ;
        call   ps          ;
  c6    btfss  6, 0        ; 是否撤防
        call   delay350    ;
        btfss  6, 0        ;
        goto   ce1         ;
        goto   l3          ;
;.....................................
  plsp  ; 提示音子程序
        bsf    5, 2        ;
        bsf    5, 3        ;
        call   delay1s     ;
        bcf    5, 2        ;
        bcf    5, 3        ;
        return             ;
; = = = = = = = = = = = = = = = = = = =
; 300uS 延时子程序

 delay350  movlw   .30     ; 外层循环次数
        movwf   data6
 P2     movlw   .10        ; 里层循环次数
        movwf   data5       ; 循环总次数 =30  * 10 =300us
 P1     decfsz  data5, 1    ; 递减, 为 0 则跳
        goto    P1
        decfsz  data6, 1    ; 递减, 为 0 则跳
        goto    P2
        return
;.....................................
  ; 05S 延时子程序
  delay05 movlw  .155       ; 外层循环次数
        movwf   data3       ;
  d1    movlw   .255        ; 内层循环次数
        movwf   data4       ; 循环总次数 =255 * 255us =65S
  d2    decfsz  data4, 1    ; 递减, 为 0 则跳
        goto    d3          ;
        decfsz  data3, 1    ; 递减, 为 0 则跳
        goto    d1          ;
  d4    ;
        return
  d3    btfss   6, 0        ; 是否撤防
        goto    d2          ;
        call    delay350    ;
        btfss   6, 0        ;
        goto    d2          ;
        goto    d4          ;
;.....................................
  ; 1S 延时子程序
  delay1s movlw  .255       ; 外层循环次数
        movwf  data0        ;
  lop   movlw  .255         ; 内层循环次数
        movwf  data1        ; 循环总次数 =255 * 255us =65S
  lop1  decfsz  data1, 1    ; 递减, 为 0 则跳
        goto  lop2          ;
        decfsz data0, 1     ; 递减, 为 0 则跳
        goto  lop           ;
  lop3  ;
        return
  lop2  btfss  6, 0         ; 是否撤防
        goto  lop1          ;
        call  delay350      ;
        btfss  6, 0         ;
        goto  lop1          ;
        goto  lop3          ;
        ;               ;
        END
```

参考文献

1. 王有绪 . PIC 系列单片机接口技术及应用系统设计 . 北京航天大学出版社

2. 余永权 . 模糊控制技术与模糊家用电器 . 北京航天大学出版社

3. 施庆隆 . PIC16F87X 单片机原理与专题应用 . 电子工业出版社

收稿日期：2006 年 4 月 26 日

闪存盘数据备份软件

王昊鹏

摘　要　介绍在 Delphi7.0 上利用“WM_DEVICECHANGE”消息编写闪存盘数据备份软件的实现方法。

关键词　消息，闪存盘，备份

一、前言

现在闪存盘（即 U 盘）的使用已经十分普及，逐渐淘汰了软件等移动存储介质。闪存盘是 USB 技术和闪存技术结合的产物，从它的产生到现在这短短几年中给我们带来了极大的便捷，计算机移动数据的应用也由此得到了极大的体现。但是，在欣喜于闪存盘的方便、快捷时，一些问题也随着滋生，烦恼着广大的闪存盘使用者。目前，闪存盘的品牌五花八门，其中的质量也良莠不齐。很多质量不高的闪存盘很不稳定，在使用的时候经常会给用户带来丢失数据等不便。

计算机系统的硬件发生变化时，可以通过拦截 Windows 的 WM_DEVICECHANGE 消息得知。本文讨论了 WM_DEVICECHANGE 消息的定义和结构，在 Borland Delphi 7.0 平台上可以利用 WM_DEVICECHANGE 编写软件，软件后台运行，实时拦截 WM_DEVICECHANGE 消息，从而实现对计算机系统中 USB 等硬件设备的监测。在判断 USB 设备装载到计算机中时自动对闪存盘内的全部数据进行备份。下面就介绍利用 WM_DEVICECHANGE 消息编写该软件的方法。

二、WM_DEVICECHANGE 消息的定义和结构

WM_DEVICECHANGE 消息通报注册在计算机系统中的硬件的变化情况。WM_DEVICECHANGE 消息中包含 2 个参数 wParam 和 lParam。其中，参数 wParam 用于指定硬件变化事件的发生；参数 lParam 是一个指针，指针指向一个结构。WM_DEVICECHANGE 消息的定义方式如下：

```
Event = (UINT) wParam; //事件
dwData = (DWORD) lParam; //指针
```

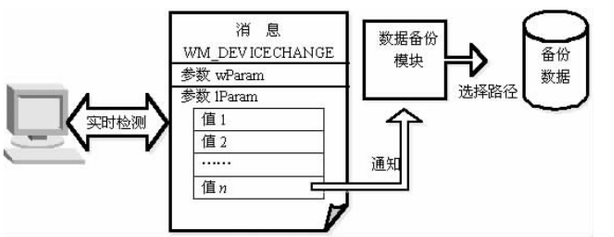
WM_DEVICECHANGE 消息中参数 wParam 取值如右表所示。

参数 wParam 取值表

取值	值域说明
DBT-CONFIGCHANGECANCELED	请求改变当前被取消的配置
DBT-CONFIGCHANGED	当前配置已经改变
DBT-DEVICEARRIVAL	设备已经装入并且可以使用
DBT-DEVICEQUERYREMOVE	请求移除硬件设备的许可
DBT-DEVICEQUERYREMOVEFAILED	请求移除被取消的设备
DBT-DEVICEREMOVECOMPLETE	被移除的设备
DBT-DEVICEREMOVEPENDING	等待被移除的设备
DBT-DEVICETYPESPECIFIC	已发生的设备事件
DBT-QUERYCHANGECONFIG	请求改变当前配置的许可
DBT-USERDEFINED	自定义消息的含义

三、编写过程 WM_DEVICECHANGE

监视系统中 USB 设备的变化情况来源于 Windows 的消息 WM_DEVICECHANGE，通过拦截消息 WM_DEVICECHANGE 获得系统的硬件设备变化情况，从而判断 USB 设备是否装载到系统中。当得知系统中已有 USB 设备状态后，准备数据存放路径，启动数据备份功能，实现闪存盘的自动数据备份。消息 WM_DEVICECHANGE 的工作方式如下图所示。



消息 WM_DEVICECHANGE 的工作方式

根据消息 WM_DEVICECHANGE 编写 WMDeviceChange 过程的代码如下：

```
procedure TFrm_Main.WMDeviceChange(var Msg: TMessage);
```

```
var lpdb : PDEV_BROADCAST_HDR;
 lpdbv : PDEV_BROADCAST_VOLUME;
 unitmask: DWORD;
 i: integer;
begin
 lpdb : = PDEV_BROADCAST_HDR(Msg. LParam);
 case Msg. WParam of
  DBT_DEVICEARRIVAL : //新的设备装载完毕
  if lpdb. dbch_devicetype = DBT_DEVTYP_VOLUME then
  begin
   lpdbv : = PDEV_BROADCAST_VOLUME(lpdb);
   unitmask: = lpdbv. dbcv_unitmask; //获取设备的盘符
   for i: = 0 to 25 do //遍历本地硬盘
   begin
    if Boolean(unitmask and $1) then
     //查看该驱动器的状态是否发生变化
     break;
    unitmask : = unitmask shr 1;
   end;
   //char(i + 65); //变化的盘符
   CopyDirAll(Char(i + 65) + ':\', ExtractFileDir
(Application. Exename) + '\备份的文件\');
 //拷贝闪存盘的数据
  end;
 end;
end;
```

四、编写数据备份函数 CopyDirAll

闪存盘中数据的自动备份是该软件的主要功能，当获得闪存盘已经装载到计算机系统中后，立刻调用数据备份函数 CopyDirAll。函数 CopyDirAll 首先根据 WM_DEVICECHANGE 过程获得的闪存盘的相关信息（如盘符等）将闪存盘下所有目录进行拷贝，然后将所有目录（即数据）保存到当前目录中。由于软件的主界面的可视（Visible）属性设为 False，其运行、工作不干扰系统当前运行状态，从而实现闪存盘数据的自动备份。

编写函数 CopyDirAll 的代码如下：

```
function TFrm_Main. CopyDirAll(sDirName: String;
sToDirName: String): Boolean;
var
 hFindFile: Cardinal; //拷贝整个目录(包括子目录)
 t, tfile: String;
 sCurDir: String[255];
 FindFileData: WIN32_FIND_DATA;
begin //先保存软件当前目录
 sCurDir: = GetCurrentDir;
 ChDir(sDirName);
 hFindFile: = FindFirstFile('*. *', FindFileData);
 if hFindFile <> INVALID_HANDLE_VALUE then
 begin
  if not DirectoryExists(sToDirName) then ForceDirectories
(sToDirName);
  repeat
   tfile: = FindFileData. cFileName;
   if (tfile = '. ') or (tfile = '. . ') then Continue;
   if FindFileData. dwFileAttributes =
   FILE_ATTRIBUTE_DIRECTORY then
   begin
    t: = sToDirName + '\' + tfile;
    if not DirectoryExists(t) then ForceDirectories(t);
    if sDirName[Length(sDirName)] <> '\' then
     CopyDirAll(sDirName + '\' + tfile, t)
    else CopyDirAll(sDirName + tfile, sToDirName + tfile);
   end
   else
   begin
    t: = sToDirName + '\' + tFile;
    CopyFile(PChar(tfile), PChar(t), True);
   end;
  until FindNextFile(hFindFile, FindFileData) = false;
  Windows. FindClose(hFindFile);
 end
 else
 begin
  ChDir(sCurDir);
  result: = false;
  exit;
 end;
 //回到原来的目录下
 ChDir(sCurDir);
 result: = true;
end;
```

由于软件的运行不可见，为方便对软件的控制操作，可以为软件设置热键，通过热键即可实现结束软件等操作。

五、结束语

本文分析了 WM_DEVICECHANGE 消息的定义、结构和工作方式，介绍了通过 Windows 中的 WM_DEVICECHANGE 编写闪存盘的自动数据备份软件的实现方法。软件功能明确、实用，解决了 USB 存储设备在使用过程中不稳定性的实际问题。

参考文献

1. John Ayres. The Tomes of Delphi tm Win32 Core API Windows 2000 Edition[M]. Wordware Publishing Inc. 2001.

2. 谭燕 赵磊 李之明. Delphi 高级辅助工具精解[M]. 中国铁道出版社，2003.

收稿日期：2006 年 4 月 26 日