

# 基于 Pocket PC 的 IDE 设计与实现<sup>①</sup>

## Design and Implement of IDE Based on Pocket PC

张 胜 洪 明 (南昌航空大学 计算机学院 江西 南昌 330063)

**摘 要:** 开发基于 Pocket PC 的集成开发环境(IDE)使用户不必依赖台式 PC,直接在掌上电脑上编辑、编译、连接生成可执行程序,让编程爱好者真正做到随时随地开发程序。在介绍嵌入式集成开发环境相应功能的基础上,重点阐述了部分关键技术的设计思想和实现方法,并给出部分运行效果截图。该软件成功地应用到多普达 830 智能手机上。

**关键词:** 智能移动设备 Pocket PC 集成开发环境(IDE) 智能编辑器

### 1 引言

随着计算机软硬件技术的发展及用户个性化需求的不断增长,计算机也从台式机(Desktop PC),发展到笔记本电脑(Notebook PC),更进一步发展到掌上电脑(Palm size PC),使计算机更加轻型化。近几年掌上电脑的发展更是突飞猛进,在全球销售量不断增长的同时,其功能也继承并扩展了 PDA(Personal Digital Assistant)的功能,还不断与移动通信技术、网络互连技术、多媒体技术,甚至 GPS、GIS 技术相结合<sup>[1]</sup>,形成品种繁多的便携式智能移动设备,逐渐成为人们生活中不可缺少的一部分。

目前,应用于掌上电脑的操作系统主要有: Palm 公司的 Palm OS,微软公司的基于 Windows CE 3.0 内核的 Pocket PC, Poision 公司的 EPOC OS,还有 Linux 和 AgentVR。除了 Nokia 将 EPOC OS 作为其智能手机的操作系统外, Linux 和 AgentVR 更是不成气候。市面上真正有着广泛应用的是 Palm OS 和 Pocket PC。尽管 Palm OS 占尽先机,据统计到 2000 年为止,使用 Palm OS 的掌上电脑占有了全世界掌上电脑市场 75% 以上的市场份额<sup>[2]</sup>。但 Pocket PC 有后来居上之势。Pocket PC 具有多任务处理功能,同时附带了如: Pocket word, Pocket Excel、Pocket Outlook、Pocket Internet Explorer、Pocket Money(财务)、Reader(电子书阅读)、计算器等个人信息管理软件,甚至具有从桌面版本的 Windows 带过来的 Windows Media Player(多媒体播

放程序),这些内建的功能极强的软件大大增强了 Pocket PC 的竞争能力。另外它的多线程处理功能,以及跟台式机的无缝结合功能使其具有广阔的发展前景。微软还专门发布了中文版本的 Pocket PC,足见其对中国市场的重视。由此可见基于 Pocket PC 的嵌入式程序的开发是未来掌上电脑应用程序开发的主流。

随着 Pocket PC 用户的逐渐增多,开发基于 Pocket PC 的应用软件也越来越受到重视。然而,目前这些应用软件的开发模式都是在台式 PC 上进行的,在 Pocket PC 模拟器中调上试成功后再安装到掌上电脑。开发者无法脱离台式 PC 而直接在掌上电脑上开发程序生成本地代码。本文在借鉴国外的一些研究成果的基础上,开发了基于 Pocket PC 的智能移动设备集成开发环境(IDE)。通过该集成开发环境,用户可以直接在掌上电脑或智能手机上编辑、编译、连接生成可执行的程序,真正做到随时随地进行程序的开发。

### 2 系统主要功能模块

掌上电脑与台式机比较,具有体积小、重量轻,携带方便的特点。但另一方面,由于其 CPU 的处理能力、内存容量都有限,且屏幕小、只有有限的几个按键等固有的不足,给开发掌上电脑的软件带来了很大限制。针对掌上电脑这些固有的特点,采取特别的处理,

① 基金项目:江西省教育厅科技计划资助项目(GJJ08223);南昌航空大学学院基金资助项目(EA200606198)

增强独特的功能(如:函数列表、函数自动生成、代码格式化等功能),以方便用户使用。

基于 Pocket PC 的集成开发环境的基本功能包括四大模块:参数设置模块、工程管理模块、智能编辑器模块和编译生成模块。

#### (1) 参数设置模块

参数设置模块主要包括编译路径设置、智能编辑器参数设置、界面设置等。其中通过编译路径设置来设定编译器的路径;智能编辑器参数设置进行字体、字号、TAB 和代码风格(ANSI、LINUX、JAVA)的设置;界面设置用来完成是否显示工具栏的设定。

#### (2) 工程管理模块

工程管理模块包括工程管理和文件管理。工程管理包括新建工程、打开工程、存储工程、关闭工程等功能。新建的工程有两种模式:① Pocket PC Console Application (Pocket PC 控制台程序);② Pocket PC Application (Pocket PC 标准 Windows 程序)。前者用于控制台命令执行的环境,后者用于标准 Windows 环境。文件管理包括 C++ 源文件(C++ Source file)、头文件(C++ header file)、RC 资源文件(Resource Script)的管理,如:文件的建立、打开、添加、保存、关闭等功能。

#### (3) 智能编辑器模块

智能编辑器模块包括基本编辑功能、函数自动生成功能、函数列表功能和代码格式化功能。基本编辑功能主要有全选、撤销、剪切、复制、粘贴、删除、查找和替换等功能,完成基本的程序代码编辑。系统中增加了函数自动生成功能、函数列表功能和代码格式化功能是为了尽可能减少代码的输入量,以提高编写代码的效率。

函数自动生成功能是指当用户输入某个函数的前几个字母时,系统能够自动识别并与函数库进行匹配,列出以这些字母开头的函数供用户选择。这样用户不需要记忆大量的 API 函数,也能够非常方便快捷的编写程序。

函数列表功能是指系统自动分析代码并以快捷菜单的形式列出代码中所涉及到的全部函数供用户选择。当用户选择指定函数之后,系统将会立即跳转到该函数的函数体当中,以方便用户更好的阅读代码和把握代码结构。

代码格式化功能是指将用户编写的凌乱的代码格式化成符合 ANSI、LINUX、JAVA 等风格的结构清晰的代码,便于用户阅读。本系统借鉴一个开源软件 Artistic Style<sup>[3]</sup>提供的代码格式化软件。

#### (4) 编译生成模块

编译生成模块完成工程的编译、连接、生成可执行的代码。如果编译出错系统自动报告出错原因以及错误代码位置。用户双击错误信息提示,光标立即跳转至源程序错误代码的位置,以方便用户快速修改代码。

## 3 关键技术

由于绝大多数掌上电脑没有键盘,只有少数几个按钮,字符输入主要通过点击触摸屏上的软键盘完成,远没有台式机输入方便。另外, Pocket PC 毕竟是嵌入式操作系统,没有桌面 Windows 操作系统功能强大。很多功能在桌面 Windows 中很容易用 MFC 来实现,但在 Pocket PC 中无法实现。为了弥补这些不足,本系统一方面专门增加了函数自动生成功能、函数列表功能、代码格式化功能,来减少用户代码的输入量,以提高编程效率;另一方面,跳出桌面 Windows 的编程思路寻找解决问题的途径。下面简单介绍一下系统开发中的部分关键技术。

### 3.1 如何实现多窗口切换及动态添加菜单项

在编辑、编译、调式程序时免不了打开多个窗口,还能随时进行不同窗口间的切换。这些功能在台式机上实现起来很方便,但在 Pocket PC 上实现比较困难。由于 Pocket PC 不支持多文档界面类型,只支持单文档、对话框和带列表的单文档三种类型。为了实现多窗口操作,本文采用“黑板机制”,根据用户要求将不同的文档分时装载到单文档窗口中,实现不同时段打开不同的文档窗口的功能。具体实现步骤为:

第一步,自定义 CCodeViewEditContent 类,用于存放已经打开的文档内容和相关信息。类的结构如下:

```
class CCodeViewEditContent
{
public:
    CString m_FileDir; //文档的路径
    CFileName m_FileName; //文档的文件名
    BOOL m_NeedSave; //是否需要保存
    CPosition m_Position; //光标位置
```

```

CFind m_Find; //查找的信息
CString m_Content; //文档内容
BOOL m_Show; //文档的显示状态
CCodeViewEditContent();
void Save(CString p_FileDir, CFileName
p_FileName, BOOL p_NeedSave, CPosition p_
Position, CFind
p_Find, CString p_Content, BOOL p_Show);
//保存当前窗口显示的文档信息
void Reset(); //重置文档信息
~CCodeViewEditContent(); //析构函数
};

```

第二步,建立一个以该类为元素的数组,数组中的每个元素存放一个用户打开的文档信息。若要实现不同文档间的切换,先保存当前窗口中的文档信息到数组中,再将待切换的文档信息从数组中取出并装载到当前窗口。

第三步,如何实现窗口切换操作。借用 Windows 的思想,在打开一个新文档时,在“窗口”菜单下动态添加一个菜单项,供用户切换窗口时使用。如图 5 所示。

在台式 PC 环境下动态添加菜单项很容易实现,实现的代码如下:

```

AfxGetMainWnd() -> GetMenu() -> GetSub-
Menu(0) -> InsertMenu(1, MF_BYPOSITION, y, "new-
menu");

```

但是在 Windows CE 环境下无法仅仅使用上面一条语句实现动态添加菜单项。由于在 Windows CE 里,菜单不是标准顶层窗口的一部分,而是被绑定到窗口创建的命令条(或者菜单条)控件上。利用 `AfxGetMainWnd() -> GetMenu()` 只能得到一个空对象,无法得到菜单对象。因此需要另辟蹊径找到解决该问题的方法。

首先,通过 Windows CE 提供的 API 函数 `SHFindMenuBar()` 获取当前菜单条窗口句柄。那么如何获取当前菜单条窗口子菜单句柄呢?由于 `SHFindMenuBar()` 执行后返回的是 `HWND` 类型的句柄,而 `GetSubMenu()` 的第一个参数需要的是 `HMENU` 类型句柄,因此无法使用函数 `GetSubMenu()` 获取当前菜单条窗口子菜单句柄。庆幸的是 Windows CE 提供了一个 `SHC-`

`MBM_GETSUBMENU` 消息,通过发送该消息到当前菜单条窗口来获得子菜单的句柄。实现语句如下:

```

HMENU hWindowMenu = (HMENU) :: SendMes-
sage((HWND) SHFindMenuBar(GetParent() -> m_
hWnd), SHCMBM_GETSUBMENU, 0, (LPARAM) IDM_
MAIN_WINDOW);

```

其中, `IDM_MAIN_WINDOW` 为“窗口”子菜单 ID。然后,利用获得的子菜单句柄动态添加一个菜单项:

```

AppendMenuW(hWindowMenu, MF_STRING, IDM_
MAIN_WINDOW_ADD, TEXT("test"));

```

至此,当打开新的窗口时,系统自动在“窗口”菜单下添加新的菜单项,通过选择不同的菜单项实现多窗口间的切换。

### 3.2 如何实现 IDE 与编译器通信

实现 IDE 与编译器通信是 IDE 最基本的功能,这也是摆脱命令行方式编译程序的关键。在台式机桌面系统中的 IDE 与编译器之间的通信是通过建立管道通信来实现的,但 Windows CE 不支持管道,无法实现 IDE 与编译器的通信。

要实现编译器与 IDE 的通信,就必须获知:(1)如何获取编译出错信息;(2)编译器何时编译完毕。

DOS 重定向功能给了我们提示。利用 DOS 中重定向功能可以将 DOS 命令的输出重定向到指定的文件中,如:使用 `dir >1.txt` 命令将目录列表重定向到 `1.txt` 文本文件中。由此,使用这种方式可以将编译错误信息重定向到指定文件,然后 IDE 通过读取该文件就可以获知编译出错信息。

仔细分析 DOS 命令中标准文件的输入输出功能得知:文件描述符为“0”代表一个程序的标准输入;文件描述符为“1”代表标准输出;而文件描述符为“2”代表标准错误输出。因此,要获得编译器编译时的出错信息,就可以利用“标准错误输出”的功能输出错误信息,即利用“2 >”重定向符号将错误信息输出到指定文件。在调用编译器时附加编译参数的同时在其最后添加“2 > error.txt”这个参数,使得编译出错信息输出到指定文件 `error.txt` 中。具体的获取编译出错的语句如下:

```

command.Format(TEXT("/c \"%s\\cpp0.exe\"
"%s\\%s.%s" -o "%s\\%s.rc.p" -l "%s\\in-
clude" -l "%s" -DRC_INVOKED -include "%s\\fix-

```

```
incl. h" 2 > \temp\ error. txt" ), pgcc, src, filename. m_
Title, filename. m_Ext, out, filename. m_ Title, pgcc,
out, pgcc);
```

那么, IDE 如何得知编译器何时完成编译呢? 我们采用安装一个 2000ms 的定时器的策略, 通过定时器来监测编译器进程是否结束, 从而判断编译是否完成。Windows CE 有一个 API 函数 `GetExitCodeProcess()`, 其第一个参数为指定检测进程的句柄, 第二个为回调值返回该进程状态。如果返回 `STILL_ACTIVE` 则说明该进程仍然存在, 表示编译过程仍在进行; 否则, 说明编译过程结束。

### 3.3 如何实现错误代码定位

编译时出错代码定位是指在编译错误信息中双击“错误信息”立即跳转到出错的代码行, 实现快速查错的功能。在 3.2 节中利用 DOS 重定向功能获得了错误信息文件 `error. txt`。文件中的错误信息有其固定的格式。编译器出错信息格式如下:

文件绝对路径: 错误语句在编辑框的索引: 错误类型 ( `error` 或 `warning` ): 具体错误信息

利用错误信息的固定格式很容易实现用户双击错误信息而快速跳出错代码位置的功能。首先, 提取编译出错信息文件, 将其载入到当前代码窗口。然后, 根据错误语句在编辑框的索引将光标跳转至指定位置即可。

### 3.4 如何实现函数列表

函数列表顾名思义, 即系统自动分析代码并以快捷菜单的形式列出代码中所涉及到的全部函数供用户选择。当用户选择指定函数之后, 系统立即跳转到该函数的函数体当中, 以方便用户更好的阅读代码和把握代码结构。本文使用编译原理中词法分析的知识 and C++ 函数定义规则实现该功能。具体实现过程如下:

(1) 通过词法分析提取关键字汇成表格, 该表格采用双向链表结构表示。其好处是只需一步就能定位某一表项的前驱和后继, 便于快速提取函数名。词法分析的相关函数归类到 `CLex` 类中。 `CLex` 类的声明如下:

```
class CLex
{
public:
    CString& m_strContent; //待分析的文本
    long m_nTotal;
```

```
    long m_nCharIndex; //当前分析字符位置
    long m_nLineIndex; //当前分析所在行号
    CLex ( CString& p_strContent );
    inline bool GetChar ( TCHAR& p_ch ); //得到一个字符
    inline TCHAR PeekChar ( ); //向后探测一个字符
    inline void PutBack ( TCHAR p_ch ); //回退一字符
    inline bool SkipChar ( ); //跳过一字符
    inline long CurLineNo ( ) const; //获得当前行号
    bool FilterSpace ( ); //过滤空格
    bool FilterComment ( ); //过滤注释语句
    bool CLex::isKeyword ( CString& p_strWord ); //判断是否为关键字
    CString GetDigitalString ( ); //获取数字串
    CString GetHexString ( ); //获取十六进制字符串
    CString GetString ( TCHAR p_delimiter ); //获取字符串
    CLexStatus CLex::GetToken ( CToken& p_token );
    //分析生成以 CToken 为结构的双向链表中的一个元素
    ~CLex ( ); //析构函数
};
```

(2) 根据 C++ 函数定义规则, 系统筛选出函数名并记录函数定义的起始位置。然后以快捷菜单的方式列出来供用户选择。

C++ 函数定义规则:

```
返回值类型 函数名 ( 形参类型 形参名, ... )
{
}
```

值得注意的是在筛选函数时可能会误把 `if`、`for`、`while`、`switch`、`catch` 等保留字也当成函数。因为它们都符合函数定义的规则, 需要将它们过滤掉。其方法很简单, 只需在检测到关键字为 `if`、`for`、`while`、`switch`、`catch` 时跳过即可。

### 3.5 如何实现函数自动生成

函数自动生成功能省去了用户记忆大量 API 函数麻烦。本系统规定: 当用户输入“.”符号时, 将被认定用户欲使用函数自动生成功能。此时, 系统自动记录光标位置并获取后续用户输入字符 ( 称为“当前字符”), 将“当前字符”与函数库比较, 找出以“当前字

符”开头的函数并以列表的方式供用户选择。具体实现过程如下:

(1) 系统在启动时自动载入已经建立好的函数库 `function.dat` 到一列表框 `ListBox` 当中, 并将该列表隐藏。

(2) 建立代码编辑框类 `CCodeViewEdit` 继承于 `CEdit` 类。

(3) 重载 `CCodeViewEdit` 类中 `OnChar` 事件检测用户是否输入“.”。若检测到用户输入了“.”, 则将隐藏的函数列表框 `ListBox` 置于显示状态, 同时记录用户编

辑框中光标起始位置 (`m_iStart`), 并对 `m_iEnd` (光标结束位置) 和 `m_Input` (`m_iStart` 与 `m_iEnd` 之间的内容) 进行初始化。

(4) 重载函数 `PreTranslateMessage()`, 检测按键消息 `WM_KEYDOWN`, 实现函数自动生成功能。

需要检测的按键有左右按键 (`VK_LEFT, VK_RIGHT`)、回车与 `TAB` 按键 (`VK_RETURN, VK_TAB`)、空格按键 (`VK_SPACE`)、退格按键 (`VK_BACK`)、向上向下按键 (`VK_UP, VK_DOWN`), 规定其功能如表 1 所示。

表 1 函数自动生成中相关按键的功能规定

按 键	功 能
左右按键	由于按下左右按键会影响到编辑框光标位置。规定如果用户使用左右按键导致其光标位置超出 <code>m_iStart</code> 与 <code>m_iEnd</code> 范围之外, 那么立即结束使用函数自动生成功能并将函数列表框隐藏。
回车与 <code>TAB</code> 按键	当用户在使用函数自动生成功能时按下回车或者 <code>TAB</code> 时, 意味着用户已经在函数列表框中选定好函数并要求系统将原来输入的内容替换成完整函数并结束使用该功能。
空格按键	规定无条件结束使用函数自动生成功能并将函数列表框隐藏。
退格按键	当用户使用退格按键时, 我们不仅要检测光标位置是否小于 <code>m_iStart</code> 以便结束该功能的使用, 而且我们还要根据用户使用退格导致的 <code>m_Input</code> 的变化及时反映在函数列表框中, 即用户删除输入的内容时, 函数列表框也必须相应的变化。
向上、向下按键	规定用户使用向上或者向下按键选择函数列表框中的内容, 以代替使用笔点击屏幕的繁琐动作。

#### 4 系统开发环境及部分效果截图

基于 Pocket PC 的 IDE 的开发环境如下:

(1) 开发平台

- ① 硬件环境: P4 2.8G/512M DDR/80G。
- ② 操作系统: Windows 2000 Server 及其以上版本。
- ③ 开发工具: Visual Studio 2005 (VC)。
- ④ 通讯软件: PC 机与 PDA 同步通讯软件 (ActiveSync v4.5)。

(2) 应用平台

- ① 硬件环境: 多普达 830 智能手机。
- ② 操作系统: Windows Mobile 6.0。

系统运行时的部分截图如下。其中, 系统参数设

置如图 1 所示。图 2 显示工程和文件管理菜单。图 3 和图 4 显示代码格式化前后的效果。图 5 显示打开编辑多个窗口时, 在“窗口”菜单中动态地添加相应的菜单项。选择不同的菜单项来完成不同窗口间的切换。图 6 显示编辑菜单的各项功能。

图 7 显示函数自动生成功能。当用户输入函数的部分字符时 (如: `createw`), 系统列表提示有关函数供用户选择, 用户找到要输入的函数 (如: `CreateWindow()`) 按回车键即可将该函数原型输入到光标当前位置。这样不仅避免了用户准确记忆大量的库函数, 还可以减少用户代码的输入量和输入的准确性。图 8 显示函数列表功能。

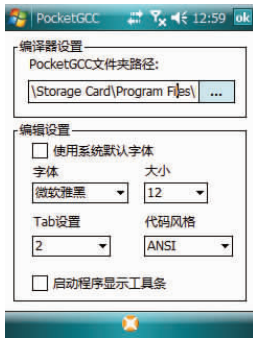


图 1 系统参数设置

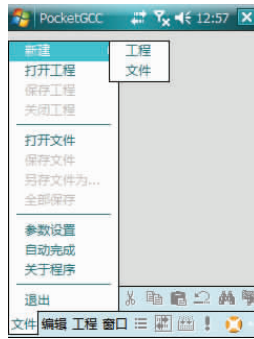


图 2 工程和文件管理

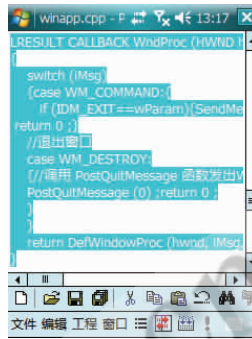


图 3 代码格式化前

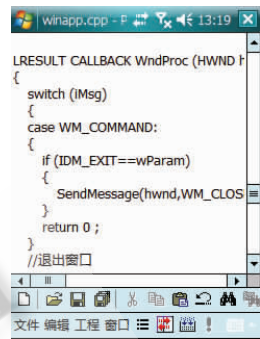


图 4 代码格式化后

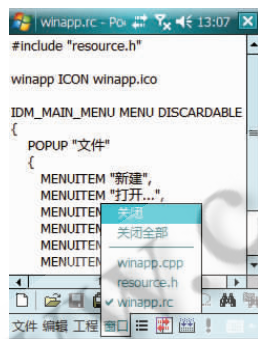


图 5 多窗口切换



图 6 编辑菜单

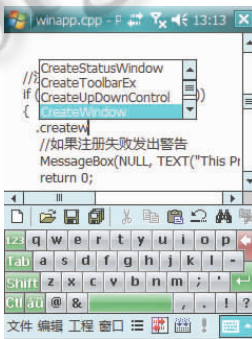


图 7 函数自动生成



图 8 函数列表

### 5 小结

在 Visual Studio 2005 环境下开发了基于智能移动设备的集成开发环境,实现了嵌入式 IDE 的基本功能。用户能够完全脱离 PC 的情况下编写、调试运行 Pocket PC 程序。根据智能移动设备的特点,为减少用户编辑输入代码的工作量,特别设计了函数列表、函数自动生成、代码格式化等功能,提高了代码输入的效率 and 准确度。更值得一提的是,编译器所生成的程序为标准的应用程序,而非类似于 Pocket C 或 JAVA 等生成中间代码,无需附加平台即可直接运行。

### 参考文献

- 1 田根,张锦,童小华,刘妙龙. 基于掌上电脑的“3S”技术集成研究与实现. 计算机工程,2006,32(4): 21-23,35.
- 2 杰逊,吴加录. 谁能挽救 Palm OS. <http://www.ciw.com.cn/News/deepstory/2006-05-08/4405.shtml>.
- 3 Jim Pattee. Artistic Style. <http://astyle.sourceforge.net/>.
- 4 Microsoft Corporation. MSDN Library. Microsoft Press, 2001. 10.
- 5 Douglas Boling. Microsoft Windows CE Programming. Microsoft Press, 1999. 8.