

第一章 窗口双列表框应用

1.1 什么是双列表框

在 Windows 作业环境下, 常会遇到一种显示格式 (Form), 如图 1.1 所示, 可以让用户做多重选择, 如安装、条件选择, 记录选择等, 这样的显示格式 (Form), 是以对象 (Object) 为主的显示方式, 并在右边有滚动条 (Scroll Bar)。这样可以让列表框中的记录在移动时立即发出相关信息, 也可以设计成让用户按下 ENTER 键, 或双击鼠标左键 (Double Click) 再送出相关信息; 所以采用列表框 (Double List) 设计方式, 已经成为 Windows 屏幕输出格式的一种标准设计, 在本书利用 FoxPro 所提供的工具及特性, 模拟出相同的屏幕输出方式, 并提出其设计思想及应用方式。

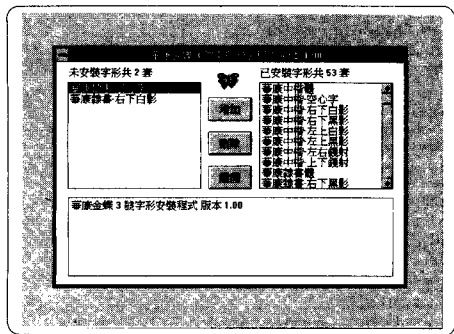


图 1.1 Double List 双列表框范例

1.2 定义列表框

1.2.1 指令方式

我们可以使用指令方式直接写出列表框 (List) 功能, 但并不鼓励读者采用这样的方式设计, 因为 FoxPro for Windows 中的任何对象 (Object) 均可定义字形、大小、类型等; 使得编辑语法非常长, 常使设计者在设计过程中, 感到非常麻烦, 也造成编码动作过多, 常常造成错误产生, 从而增加不必要的除错时间, 所以并不建议读者采用编码方式设计。

1.2.1.1 指令说明

```
DEFINE POPUP <popup name>
PROMPT FIELD <fieldname1+fieldname2...>
[SCROLL]
```

DEFINE POPUP 是定义列表框内所提示的内容类型, 并设计成滚动方式 (Scroll), 可上下移动。

```
@. (row, column) GET (memory) | (field)
FROM (array) [RANGE (expN1) [, (expN2)]] | POPUP (popup name)
[FUNCTION (expC1)] | [PICTURE (expC2)]
[FONT (expC3) [, (expN3)]]
[STYLE (expC4)]
[DEFAULT (expr)]
[SIZE (expN4), (expN5)]
[ENABLE|DISABLE]
[MESSAGE (expC5)]
[VALID (expL1) | (expN6)]
[WHEN (expL2)]
[COLOR SCHEME (expN7) | COLOR (color pair list)]
```

@... GET... 指令定义列表框 (List) 时必须加上 PROMPT 参数或 FROM 参数, 其差别在于 POPUP 来自 Define Popup, 而 From 的内容来自一个数组 (ARRAY)。

范例:

```
USE CUSTOMER &&. 打开数据库
```

• 定义列表框内容

```
DEFINE POPUP _qcd0unjit PROMPT FIELD CUSTOMER.cname1
```

• 定义 List 对象

```
@ 2.435, 1.625 GET selname PICTURE " @&N";
POPUP _qcd0unjit SIZE 8.727, 20.875 DEFAULT"";
FONT " Time New Roman", 14 STYLE""
```

```
READ CYCLE &&. 读取对象
```

1.2.2 利用 4GL 工具-Screen Builder 方式建立

1.2.2.1 步骤 1——建立新屏幕文件

于 FoxPro 系统 Menu 中选择 File 内 New... 项目, 如图 1.2 所示, 在 File Type 选择 Screen 按下 New 按钮即可进入 Screen Builder 屏幕格式生成器中进行编辑。

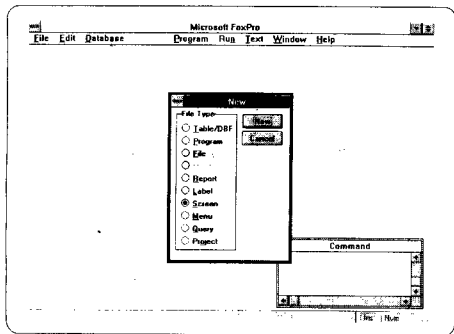


图 1.2 执行 Screen Builder 屏幕格式生成器

1.2.2.2 步骤 2——设置 List 对象

在 Screen Builder 中选择右边工具箱内列表框(List)对象后, 再到编辑框中的相对位置, 以鼠标左键定位 (如图 1.3) 随即进入 List 编辑对话框中。

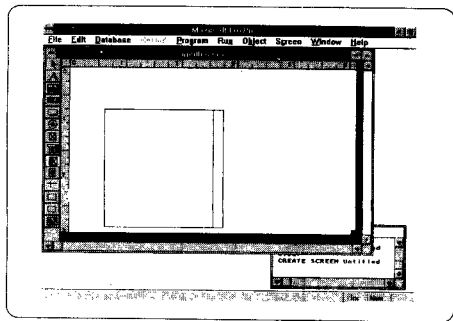


图 1.3 Screen Builder List 对象

1.2.2.3 步骤 3 ---定义 List 对象属性

在 List 对话框中 List Items 条件内选择 From Field 显示项目，并在 Variable 设置变量名（即对象名称）即可，如图 1.4 所示。

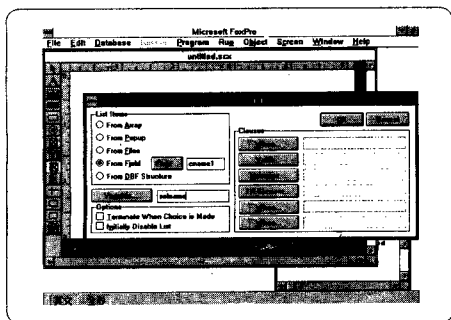


图 1-1 Screen Builder List 对话框口设置

1.2.2.4 步骤 4——设置列表框显示字形及属性

在设置好 List 对象对话框的各种设置后，可再选择 Menu 中的 Object 项目进行字形、颜色等其他属性定义，起到美化作用，如图 1.5。

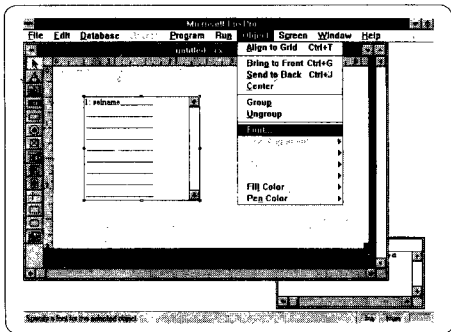


图 1.5 Screen Builder List 特殊属性设置

注意：作者建议，当将 FoxPro 4GL Tools 及 Power Tools 作为程序开发工具时，应尽量利用，才可以使程序开发过程中节省更多的时间；另外更重要的一点就是：将来可使程序在跨平台（Cross Platform）操作中，移植顺利。

1.3 范例介绍——邮寄标签范例

1.3.1 设计思想

双列表框（Double List）的设计思想在于模拟真实世界（Visual World），让用户在操作过程中，利用鼠标或键盘在左边的原始列表框（Source List）做选择，就能将记录或内容转移至右边目标列表框（Target List）中。当然，亦可让记录或内容从左边列表框转移至右边列表框中，这样将记录或内容做转移的输出设计方式，就是列表框设计思想。

可以利用 FoxPro 事件驱动（Event Driven）程序设计特性，模拟出相同的输出格式，作者已将此特性调到应用系统的开发中，其中几个范例如下（如图 1.6）。

其实双列表框中所显示的内容或记录，应当可以视为数据库或数组值的显示，而列表框之间的增减显示，更可有作两边列表框数据库或数组元表之增减显示，再配合窗口更新显示（Refresh），让用户感受到记录或内容真实转移的过程，以显示事件驱动程序中模拟真实世界的效果。

邮寄标签输出屏幕结果如图 1.7 所示。

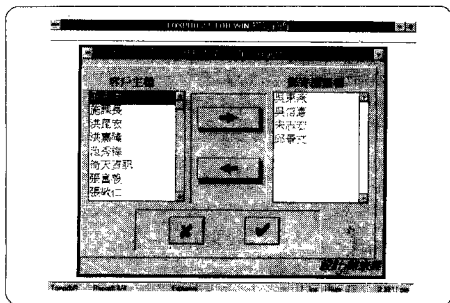


图 1.6 双列表框应用：邮寄标签范例

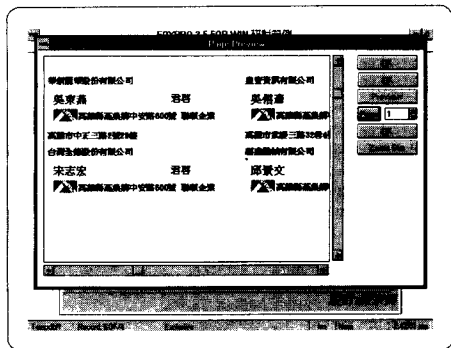


图 1.7 邮寄标签范例

1.3.2 基本指令与函数

1.3.2.1 DEFINE WINDOW

语法

```

DEFINE WINDOW <window name1> FROM <row1, column1>
    To <row2, column2> {AT <row3, column3>
SIZE <row4, column4>
    [IN [WINDOW <window name2>
IN SCREEN [IN DESKTOP]
[FONT <expC1> [. <expN1>]]]
[STYLE <expC2>]
[FOOTER <expC3>]
[TITLE <expC4>]
[HALFHEIGHT]
[DOUBLE|PANEL|NONE
|SYSTEM] <border string>]
[CLOSE|NOCLOSE] [FLOAT|NOFLOAT]
[GROW|NOGROW]
[MDI|NOMDI] [MINIMIZE]
[SHADOW] [ZOOM|NOZOOM]
[ICON FILE <expC5>]
[FILL <expC6> {FILL FILE <bmp file>}
[COLOR SCHEME <expN2>]
[COLOR <color pair list>]

```

说明：定义窗口及其属性。

什么是窗口

在 FoxPro 2.5 中，窗口即是在屏幕中划分出来直角矩形，其中包含有名称、边框、标题、以及颜色、尺寸大小、坐标等，并可作输出导向等功能；而一个单一的窗口可以说是一个窗口对象。

在每一个窗口定义上，都必须赋予一个窗口名称（Window Name），及窗口大小（尺寸）。所以窗口名称及大小是构成窗口必要条件。在启动窗口对象之前必须先定义窗口。

范例：

```

DEFINE WINDOW demo1 AT 0.000, 0.000;
SIZE 16.435, 66.125 TITLE " 定义窗口";
FONT " Time New Roman", 14;
STYLE " B" FLOAT NOCLOSE MINIMIZE SYSTEM;
COLOR RGB (,, 192, 192, 192)

```

定义窗口 demo1, 0.000, 0.000, 尺寸大小为宽 16.435, 长为 66.125; 字形为 Time New Roman, 字形大小为 14, 类型为粗黑体" B"。

.FLOAT ——窗口可移动

- . NOCLOSE ——窗口不可关闭
- . MINIMIZE ——窗口可最小化 (ICON)
- . SYSTEM ——Border 边框类型为 Windows 系统型
- . COLOR RGB () ——定义窗口颜色

窗口定义参数共有二十几个,除了窗口名称及大小外,其余参数均为任选项,可根据需要定义,其余参数定义请参阅相关书籍,此处不再赘述。

1.3.2.2 MOVE WINDOW

语法

```
MOVE WINDOW <window name> TO <row, column> [BY
<expN1>, <expN2>]
[<CENTER>]
```

说明: 移动窗口。

MOVE WINDOW <window name> 可将已经定义好的窗口移动到指定位置或相关位置,在移动过程中,是把窗口作为一个对象移动,所以在连续移动过程中,并不会在屏幕上产生残影,所以该指令可作出窗口动画效果。

MOVE WINDOW demo1 CENTER

将窗口 demo1 移至屏幕中央或是其父窗口 (Parent Window) 中央。

注意: 窗口移到中央位置,并不代表该窗口已经显示在屏幕上,显示与否要视该窗口是否已经启动而定。MOVE WINDOW 指令可以在启动窗口之前执行,做为预先定义窗口的输出位置,然后再启动窗口到相关的位置上。这种为定义好的窗口先做 MOVE WINDOW,通常是为了避免将窗口输出在屏幕相对位置上时,将窗口先启动在其他位置上,再移动窗口至屏幕中央。这是一种整体性输出设计方式,这样的输出方式在 Screen Builder 所产生的源程序中处处可见。

MOVE WINDOW demo1 TO 10.000, 20.000

TO 参数所定义的坐标代表窗口左上角坐标的新位置。

MOVE WINDOW demo1 BY 10.000, 20.000

BY 参数所定义的坐标代表相对于原先的坐标位置。

注意: 若新坐标位置超过输出范围时,在执行 MOVE WINDOW 时并无任何错误信息,而在启动窗口后,窗口将无法显示在屏幕上或显示不齐全:有些部分超出屏幕范围。

1.3.2.3 ACTIVATE WINDOW

语法

```
ACTIVATE WINDOW [ <window name1> [, <window name2> ... ] [ALL]
[IN [WINDOW] <window name3>] [SCREEN]
[BOTTOM|TOP|SAME]
[NOSHOW]
```

说明: 启动窗口。

可将先前定义好的窗口对象,从存储器中启动,并将输出指向此窗口。

如果 <window-name-list> 超过一个窗口时,那么参数 ALL 将可启动所有的窗口,而最后一个定义的窗口,就是输出窗口,如图 1.8。

ACTIVATE WINDOW ALL

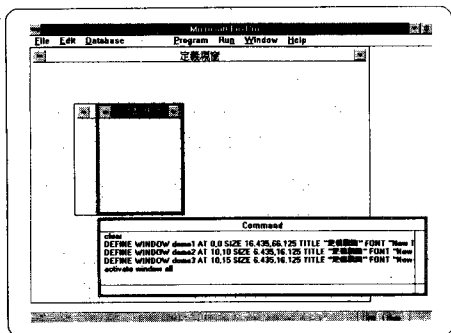


图 1.8 启动所有窗口

可以单独启动一个已经定义好的窗口，以 ACTIVATE WINDOW 直接启动一个窗口名称，如图 1.9。

ACTIVATE WINDOW demo1

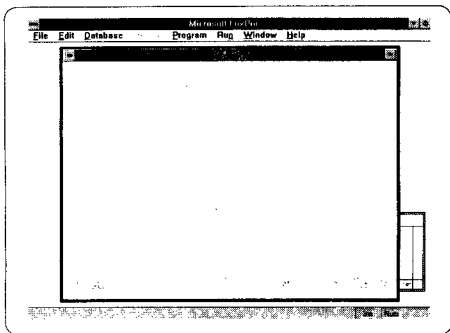


图 1.9 启动 demof 窗口

1.3.2.1 CREATE CURSOR

语法

```
CREATE CURSOR (dbf name) ( (fname1) (type)
    [ (precision [, scale, j])
    [, (fname2) ... ])
FROM ARRAY (array)
```

说明：CREATE CURSOR 属于 SQL 命令，可以利用存储器建立一个临时文件，其数据库结构均与真正数据库一样，并可建立单一索引文件，或复合式索引文件。优点在于存取速度快，且不占磁盘空间，在处理完后关掉该数据库或离开系统，该临时数据库随即从内存删除。如下面程序码说明：

```
CREATE CURSOR (dbf name) ( (fieldname-list))
```

在建立临时数据库时必须指定数据库名称 (dbf name) 及其字段名称、类型、长度等属性 (fieldname-list)。

数据库名称无需符合 MS-DOS 文件规范 (8 个字符以内)，但必须符合 FoxPro 变量长度限制，在 10 个字符 (含) 长度内。

(fieldname-list) 字段设置可以以逗号“,” 隔开，建立二个以上之字段。

范例：

```
CREATE CURSOR tempdbf1 (cname C (8),
add2 C (30), company C (30), memo M)
```

建立一个 tempdbf1 存储器临时数据库 (CURSORS)，其字段为：

字段名称	字段类型	长度	小数位
ename1	C	8	
add2	C	30	
company	C	30	
memo	M	10	

注意：备注字段 (M)、通用字段 (G)、逻辑字段 (L) 建立时，不必定义字段长度，因为这三种字段类型均内定长度 10，所以在 'fieldname list' 只要写入数据类型即可；浮点运算字段 (F) 则必须定义长度。如图 1-10 所示。

若为数值字段类型，则设置小数点位数时，以 N (n1, n2) 方式设置，n1 代表整数位，n2 代表小数位。

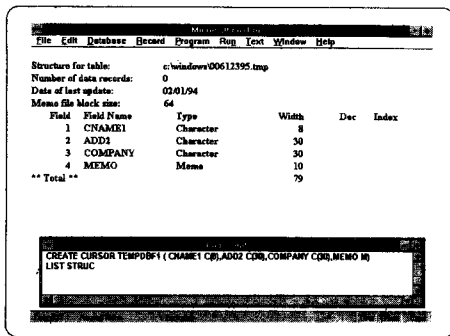


图 1-10 CREATE CURSOR 建立存储虚拟数据库

注意：由图 1-10 看出 CURSORS 数据库名称设置为 .TMP，而且不同于 CREATE CURSOR (database-name) 语法所指定的数据库名称。读者可能有疑问，为何 FoxPro 要这样定义临时文件数据库名称？根据作者经验，这是 FoxPro 设计精妙之处，既然是为临时文件，就应该随机赋予数据库名称，这样可在网络操作时，避免用户用到同一数据库，而造成数据冲突或数据被锁定等问题。也就是说在网络操作时，同一操作 (CREATE CURSOR) 会因用户进入时间不同，而赋予不同的 .TMP 文件。

既然 CREATE CURSOR 所打开的是临时文件 (.TMP)，也就是无实际数据库存储在磁

盘中, 所以也就无法使用 APPEND FROM (dbf name) 指令, 将其数据加入另一数据库中。若您使用 APPEND FROM (cursor file), 将会出 File (dbf name) . DBF does not exist 的错误信息。

1.3.2.5 INSERT INTO

语法

```
INSERT INTO (dbf name)
[ ( (field name) [, (field name2) [, ... ]])]
VALUES ( (expr1) [, (expr2) [, ... ]])
```

或

```
INSERT INTO (dbf name) FROM ARRAY (array name) [FROM MEMVAR
```

说明: INSERT INTO 可将一笔记录加入指定的数据库中。

范例:

```
SELECT 0
USE customer
SELECT 0
* 建立临时文件
CREATE CURSOR tempdbf1 (cname1 C (8), add2 C (30), company C (30))
SELECT customer
GOTO TOP
* 将 customer 所有记录加入临时文件中
SCAN
INSERT INTO tempdbf1 (cname1, add2, company)
VALUES (customer.cname1, customer.add2, customer.company)
ENDSCAN
```

采用个别字段加入方式时, 必须注意到 VALUSE 后面的字段或变量数目, 要与数据库所指定的字段相等, 否则会产生错误。

```
INSERT INTO (dbf name) FROM MEMVAR
```

采用此方式加入记录时, 可从内存变量组一次性填入记录内容。

范例:

```
SELECT 0
USE customer
SELECT 0
* 建立临时文件
CREATE CURSOR tempdbf1 (cname1 C (8), add2 C (30), company C (30))
SELECT customer
GOTO TOP
* 将 customer 所有记录加入临时文件
SCAN
* 声明内存变量数组
SCATTER MEMVAR
INSERT INTO tempdbf1 FROM MEMVAR
ENDSCAN
```

使用 FROM MEMVAR 存入内存时,若数据字段数与源数据库字段不相等时,只会填入相同字段名称的内容,如图 1.11 所示

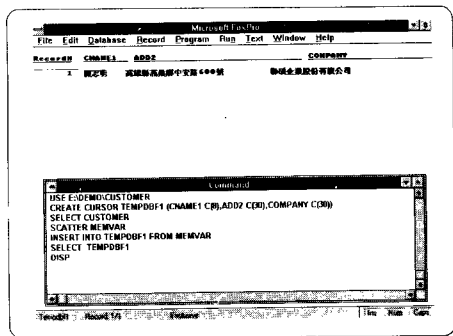


图 1.11 INSERT INTO <dbf name> FROM MEMVAR 应用

1.3.2.6 SHOW GET/GETS

语法

```
SHOW GET <var>
[., <expN1>] [PROMPT <expC>]]
[ENABLE|DISABLE]
[LEVEL <expN2>]
[COLOR SCHEME <expN3>] [COLOR <color pair list>]
```

说明: SHOW GET 可用来重新显示 (Refresh) 以 GET 定义的对象内容。所谓对象可以是字段、变量、控制按钮、检查框等。

范例:

```
m.cname1 = '陈宗兴'
@ 10, 10 GET m.cname1
= inkey (0)      && 暂停画面
m.cname1 = '张三郎'
SHOW GET m.cname1      && 不必重复@ GET 指令一次
READ CYCLE
```

加参数 Disable 可让控制按钮或检查框等对象呈现灰色而无法选择, Enable 参数则可以使

之前被关掉 (Disable) 的对象重新显示使用。

语法

SHOW GETS

```
[ENABLE | DISABLE]
[LEVEL <expN1>]
[OFF | ONLY]
[WINDOW <window name>]
[LOCK]
[COLOR SCHEME <expN2>] [COLOR <color pair list>]
```

说明: SHOW GETS 可使用同一窗口内所有对象, 显示更新。假如要使另一个窗口内所有对象显示更新, 请以 SHOW GETS WINDOW <windowname> 指定窗口名称。

1.3.2.7 CUROBJ

语法

CUROBJ = <expN>

说明: <expN> 为对象编号, CUROBJ = <expN> 可将光标移至该编号对象上。

注意: <expN> 所代表的对象编号, 是指同一窗口上的对象。一般在窗口上定义对象时, 不会记录对象建立顺序 (编号), 所以常以 OBJNUM () 函数获取某对象名称后, 响应其编号。

1.3.2.8 OBJNUM ()

语法

OBJNUM (<var> [, <expN>])

说明: OBJNUM () 函数可响应某层中某对象名称的编号, <var> 代表对象名称, 可以是内存变量、字段、下推按钮等。<expN> 代表所读取的窗口层, 若不指定, 则代表当前 READ 层。

范例:

```
_CUROBJ OBJNUM (m.name1)
```

注意: 在事件驱动程序 (Event Driven) 设计中, 常常直接将鼠标移到被选择的对象上, 因此事件驱动程序设计中, 为了使光标移到被选择的对象, 就得使用 CUROBJ 指令。

1.3.2.9 LABEL FORM

语法

LABEL

```
[FORM <file1>] [?]
[ENVIRONMENT]
[ <scope>]
[FOR <expl1>]
[WHILE <expl2>]
[NOCONSOLE]
[NOOPTIMIZE]
[PDSETUP]
[PREVIEW]
```

[SAMPLE]

[TO PRINTER [PROMPT] TO FILE <file>]

说明: LABEL FORM 可将 TABLE/DBF 送到用 LABEL WRITER 邮寄标签制作器制作的 LABEL 文件 (.lbr) 中浏览或打印。

范例:

* 将客户数据送到邮寄标签格式文件中打印

USE customer && 打开数据库

* 送到 demo8-1.lbr 中打印

LABEL FORM demo8-1.lbr TO PRINTER NOCONSOLE

demo8-1.lbr 为 LABEL WRITER 所建立的邮寄标签文件,其中设置的字段或变量,要与数据库字段或程序中定义的变量相符合,否则会产生错误信息。

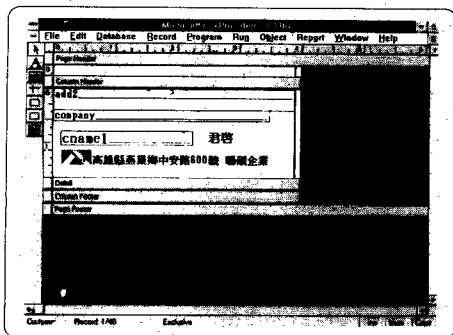


图 1.12 LABEL WRITER——demo8-1.lbr 文件建立

TO PRINTER 参数,表示将结果送到打印机。

NOCONSOLE 参数,表示将结果送到打印机时,屏幕上不做任何结果显示。

* 将客户数据送到屏幕浏览

USE customer

* 将数据库内容送到标签文件中浏览

LABEL FORM demo8-1.lbr PREVIEW

PREVIEW 参数,可将数据通过标签文件送到屏幕浏览,如图 1.12 及 1.13 所示。

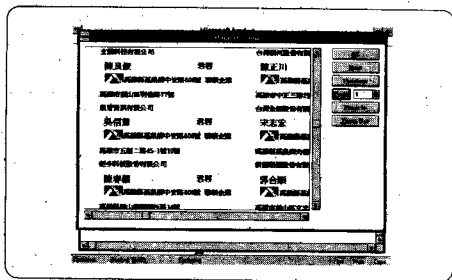


图 1-13 LABEL FORM 输出结果

1.3.3 程序范例

在介绍完基本指令与函数后, 将利用范例说明双列表框 (Double List) 如何用上述指令与函数来设计, 现就邮寄标签范例依次说明 (此范例为内附范例 DEMO.EXE)。

首先启动屏幕生成器 (Screen Builder), 我们将要定义窗口对象, 以及所有对象上的事件驱动程序。如图 1-14。

1.3.3.1 屏幕文件设置

启动 Screen Builder 后, 选择系统菜单中的 Screen Layout, 即出现如图 1-14 所示 Screen Layout 对话框。请在 Title 写入窗口标题, 以及 Name 字段内定义窗口名称 (请以英文书写), 并在 Position 选择 Center 检查框, 表示将窗口置于屏幕中央, 同时在 Screen Layout 显示窗口位图浏览区内, 用鼠标拖动右下角, 定义窗口大小。