MicroStation MDL

程序设计

2006 年

第一章　MDL编程基本方法 5
1.1 基本概念 5
1.1.1 MDL程序的结构 5
1.1.2 注释 5
1.1.3 包含文件 5
1.1.4 变量说明 7
1.1.5 变量类型 7
1.1.6 变量范围 7
1.1.7 MDL内部变量 8
1.2 MDL函数 8
1.2.1 显示信息 9
1.3 结构和联合 10
1.3.1 元素的联合 11
1.4 指针 13
1.4.1 通过引用传递 13
1.4.2 通过值传递 14
1.5 MicroStation--状态机器 15
1.6 资源 16
1.6.1 命令表 16
1.7 状态控制函数 17
1.7.1 用户函数 17
1.8 元素函数 19
1.8.1 元素生成函数 20
1.8.2 元素提取函数 21
1.9 第一章小结 29
第二章　编译MDL程序 30
2.1 预处理指令 30
2.1.1 条件编译 30
2.1.2 编译程序指令—#pragma 31
2.2 开发流程 32
2.2.1 开发流程概述 32
2.2.2 生成MDL应用 33
2.2.3 编译MDL应用 33
2.2.4 生成应用命令表 33
2.2.5 信息表(Messagelist) 35
2.2.6 编译资源 36
2.2.7 链接MDL应用 37
2.2.8 使用资源管理程序 37
2.3 bmake实用程序 38
2.3.1 制作文件(Makefile)的格式 38
2.3.2 bmake宏命令 38
2.3.3 相关(Dependencies) 39

2.3.4 推理规则 40
2.3.5 预定义的推理规则 40
2.3.6 条件编译 40
2.3.7 执行bmake 43
2.4 执行MDL应用 46
2.5 调试程序 47
2.5.1 调试的设立 47
2.5.2 具有调试功能的编译 48
2.6 使用调试程序 48
2.6.1 安装调试程序 48
2.6.2 显示变量内容 49
2.7 错误信息 50
2.8 第二章小结 50
第三章 MDL对话框 51
3.1 开发流程 51
3.2 对话管理程序 52
3.2.1 设计对话框 52
3.3 生成对话框资源 54
3.3.1 对话框条目 55
3.3.2 通用条目资源字段 56
3.3.3 文本对话条目 56
3.3.4 选项按钮条目 57
3.3.5 按钮条目 58
3.3.6 开关按钮(ToggleButton)条目 58
3.4 对话框通用函数 59
3.4.1 对话钩函数 60
3.4.2 对话框通讯 60
3.5 资源管理 62
3.5.1 用户优先选择资源 63
3.5.2 存贮资源 63
3.6 C表达式 64
3.6.1 类型发生器 65
3.6.2 C表达式函数 65
3.6.3 图形组 66
第四章 元素的搜寻和操作 75
4.1 元素搜寻 75
4.1.1 元素定位函数 75
4.1.2 元素编辑函数 76
4.1.3 选择集函数 76
4.2 激活设置 78
4.3 元素位置搜寻 84
4.3.1 扫描函数 84

4.4 围栅搜寻 90
第五章　元素描述符 101
5.1 元素描述符函数 101
5.2 复杂元素的操作 107
5.3 工作文件 109
5.4 孤立单元 114
第六章　数学与几何运算 119
6.1 浮点常量 119
6.2 向量几何 120
6.3 向量操作函数 121
6.4 当前变换 123
6.5 数据格式 131
6.6 旋转矩阵函数 136
第七章　高级对话框 143
7.1 对话条目 144
7.1.1 选项按钮条目 144
7.2 高级对话钩函数 151
7.3 元素位置 154
7.4 视图函数 156
7.5 视图钩函数 172
7.6　第七章小结 175
第八章　输入队列 177
8.1 工作方式 177
8.2 输入队列函数 177
8.3 队列元素 178
8.4 命令过滤器 179
8.5 MDL应用的自动执行 180
8.6 取消命令类别 182
8.7 系统函数 184
8.8 演示程序 186
第九章　走进NativeCode世界 189
9.1 生成NativeCode程序 189
9.1.1 安装Bentley MFC Application Wizard 189
9.1.2 建立一个NativeCode项目 189
9.1.3 编译链接生成DLL和MA 191
9.1.4 在MicroStation中运行 194
9.2 NativeCode程序结构分析 194
9.2.1 MDL制作文件及资源文件分析 195
9.2.2 MDL主程序文件分析 196
9.2.3 MFC程序文件分析 196
9.3 调试NativeCode程序 197
9.3.1 制作调试版本的NativeCode程序 197

9.3.2 建立针对MicroStation的调试项目并设置调试参数..... 198
9.3.3 开始调试过程..... 199
9.4 NativeCode编程注意事项 199
9.4.1 内存管理..... 199
9.4.2 等待消息完成..... 200
9.4.3 使用TCB变量..... 200

第一章　MDL 编程基本方法

由于MDL语言以C编程语言为基础，所以我们来看一下C语言编程的基本方法以及MDL如何使用这些方法的。本章并不想教你如何用C语言编程，现在有很多关于这方面的书。我们假定读者已经具有C语言的编程能力，因此只涉及一些MDL所需要的C的功能。下面我们就来讨论MDL编程基本方法。

1.1 基本概念

MDL语言是一种结构化编程语言，它具有让我们生成一个应用程序的标准命令、函　　数和目标。如果我们想简单地描述一下MDL，可以这样说：

MDL是这样一种语言，它采用C的结构，具有自己的运行时间库、编译程序、链接程　　序、库管理程序，并在MicroStation环境下运行。

在你读这本书时，你将发现MDL超越一种程序设计语言。它是一种进入　　MicroStation内部来开发应用程序的工具。在我们开始编写MDL代码前，我们应该把一　　些术语、标准和约定搞清楚。

1.1.1 MDL 程序的结构

MDL程序是一个按照MDL语言的规则组合起来的语句集合。程序语句包括以下内　　容：

- 控制结构或说明(declaration)语句
- 赋值语句
- 函数／程序调用语句
- 预处理指令
- 注释

由这些语句组合成的MDL程序是一个文本文件，采用扩展名“.mc”。我们要写一个　　MDL程序，用它在一个字符串周围画一个方框，同时从方框上伸出一条引线。我们将把　　它作为一个典型的MDL程序的例子。然后，我们将用新放置的元素做一个单元。这个程　　序名字叫plbox.mc(参见图1. 1)。可以用许多办法写这个程序，同时在你读这本书时你也会注意到这个程序有许多变化，完整的程序列在本章的最后。我们将一步一步地学习它。

1.1.2 注释

注释使读者了解程序的含义，也使程序员在修改程序时了解程序的功能。用向前斜线及星号(/*)表示注释的起始；而用相反的符号(*/)表示注释结束。编译程序将不处理注释。MDL不支持嵌套的注释。在MicroStation V8中，当然也可以采用双斜线(//)对单行进行注释。

1.1.3 包含文件

包含文件是这样一种文件，它包含在编译时预处理程序读入程序供使用的MDL源码。包含文件有时也叫头文件，因为它们在源码的起始部分。

我们可以把公用的常量放置在一个单独的文件内，然后可以让几个不同的程序包含这个文件，以保证这些常量协调一致地处理。有两种定义包含文件的方法：

```
#include  <mdl. h>
```

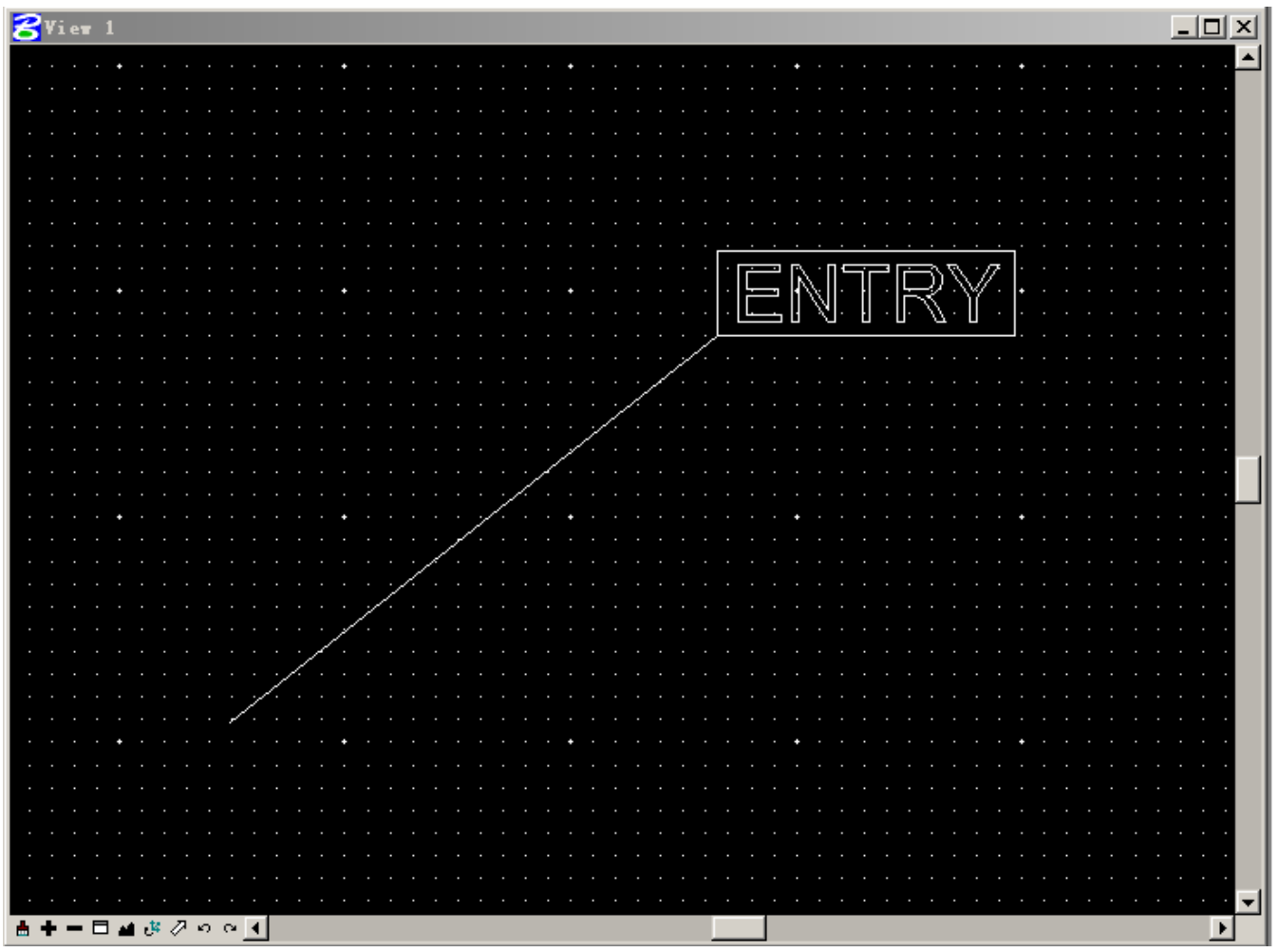


图1．1　PLBOX程序运行的结果

这个约定是用“．h”作为包含文件的扩展名。如果使用了尖括号，预处理程序就试图在 MDL的包含目录…Bentley\Program\MicroStation\mdl\include中去查找<mdl．h>文件。

或者，我们可以把文件名放在引号内，它通知预处理程序到含有包含文件说明语句的文件所在目录寻找包含文件。

```
#include    ″plbox. h″ /* Generated by” rcomp -h plbox.r” */
```

我们如何知道要使用哪一个包含文件呢?所有文件名都列在<<MDL手册>>中。例如，当我们要用mdlText_extractShape时，就必须包含<mselems．h>文件。这些MicroStation内的包含文件可在…Bentley\Program\MicroStation\mdl\include目录下找到。

在Plbox. mc程序中，将使用下述包含文件：

```
/*----------------------------------------------------------------------+
|        Include Files                                                  |
+----------------------------------------------------------------------*/
  #include     <mdl.h>           /*system include files*/
  #include     <global.h>
  #include     <mselems.h>
  #include     <userfnc.h>
  #include     <rscdefs.h>
  #include     <tcb.h>
  #include     <plbox.h>         /*Generated by rcomp -h plbox.r″ */
```

最后一个包含文件plbox.h应引起注意，因为它是由资源文件编译而成的。我们到第二章再讨论资源文件。

在MicroStation V8中,引入了新的文件类型“.fdf”,它们是函数定义文件(Function Definition File)。这些文件中含有某一类别的MDL函数原型定义，同时又有对包含文件的引用，所以，在V8编程中您会看到我们经常只需在程序开头包含这些.fdf文件就可以了。

1.1.4 变量说明

在我们使用一个变量以前必须首先定义它。大多数情况下，在开始程序设计以前我们不会知道将用到哪些变量。在这一节中我们假设已经知道需要用哪些变量。MDL中有许多定义变量的方法，我们从最基本的概念开始讨论。

变量的定义

术语“变量”使名称和内存地址联系起来。存贮变量所需内存量取决于变量类型。术语“地址”就是变量的存贮地址。例如，一个整型变量count的值是20，它的地址由&count给出。也就是“&”符号返回count的地址，而不是count的值，这是一个很重要的概念。许多程序员由于错误理解了变量和它的地址的意义而在程序中产生了许多问题。

MDL是一种对大小写字母敏感的语言。当定义变量或函数名时应该牢牢记住这一点。例如，变量linePts就与LinePts不同。

1.1.5 变量类型

变量类型定义变量如何存贮以及哪些操作适用于它。基本的变量类型是short，int，float，doublet和char。

下表定义了各种MDL变量类型：

类　型	存　贮　值
short	整型数-32,678～32,767
unsigned short	整型数0～65,535
int	整型数-32,678～32,767
long int	整型数-2,147,483,648～2，147,483,647
unsigned int	整型数0～65,535
unsigned long int	整型数0～4,294,967,265
float	按double型处理
double	有6或7个有效数字的浮点数
char	ASCII字符值，或整型数-127～128
unsigned char	整型数0～255

1.1.6 变量范围

变量的范围定义变量的说明适用于程序的哪些部分。为了决定变量的范围，一个MDL程序的源码按层次分成许多块。一个块可以是一个程序或一个函数。整个文件可以认为是一个块。一个块可以嵌套在其它块中，因此一个在外部块中定义的变量被所有嵌套的块承认。一个内部块中定义的变量名可以与一个外部块的变量名相同。在这种情况下，内部块的变量将取代外部块的变量，但是这仅局限在内部块的范围之内。在这种情况下，内部块变量被认为是局部变量。它们的值及它们的存在都只被那些说明它们的函数所承认。全局变量则被你的MDL程序中的所有函数承认。

静态变量和自动变量

变量具有存贮类别。存储类别决定变量如何存贮，它的生命有多长以及在程序的哪些地方变量名可以被引用。如果我们在函数中说明一个变量时不做特殊处理，它就是一个自动型变量。这意味着当我们调用一个函数时，这个变量自动生成，当我们离开这个函数时它被放弃。有时函数需要保留赋给局部变量的最后的值，静态static修饰词指示编译程序存贮从一个调用转到下一个调用的局部变量的值。缺省的变量是自动型变量。全局变量可能是静态的(在MDL源模块中被任何调用承认，但不能被其它源模块使用)。

如果一个全局变量没有被说明为静态的，它也可以被其它源模块使用。

在我们的例题PLBOX中，我们须定义两个用于我们的所有函数的全局变量，我们要定义一个可以容纳128个字符串的变量textin，这一点值得注意，因为“键入”缓冲区有128个字符。我们定义变量pntP用于保存数据点。

```
/*---------------------------------------------------------+
|    Private Global variables                              |
+---------------------------------------------------------*/
   static char textin[128];
   Dpoint3d    pntP[2];
```

1.1.7 MDL 内部变量

MDL提供全局变量供我们的程序使用。由于它们装载在编译程序中，所以我们可以不经过定义直接使用这些变量。许多内部变量是结构或联合。例如，变量*tcb是一个包含所有当前设计文件信息的一个结构。若一个内部变量具有简单类型，例如int，它不必包含头文件。但是如果内部类型是一个结构或联合，或是一个指向结构或联合的指针，则它必须包含适当的头文件。关于头文件的详细内容请参考附录。

类型	变　量	说　明
short	dgnbuf[]	含有dgnbuf中当前元素的所有信息。全部元素通过定位或元素操作命令装入此缓冲区
MSStateData	statedata	含有当前状态函数的所有信息，在<global. h>中定义
Tcb	*tcb	含有当前DGN文件的所有信息，在<tcb. h>中定义
Mgds_modes	mgds_modes	含有MicroStation当前执行的方式信息，在<global. h>内定义
byte	cmplx-hdr[]	如果元素是一个复杂头，则为TRUE。例如，以下为真：cmplx__hdr[CELL_TYPE]
char	mgdsPrompt[35]	含有提示字符，缺省为“nStn>”
short	element-drawn[s]	在每一位都是元素类型处它是一个位掩码，如果元素是可显示的则位被设置
short	msversion	MicroStation当前版本号,16进制
short	database	如果当前连接了数据库，则为真
int	mdlErrno	各种MDL函数的错误信息编号
MSGraphConfig	graphConfig	包含图形配置，定义在<global.h>
int		各种操作系统函数的错误编号
long	mdlCommandNumber	包含最近一次启动MDL应用的命令编号

1.2 MDL 函数

这里有两种函数类型，预定义函数和我们生成的函数。为了使用一个函数，在可以访问它以前必须定义它。为了使用预定义函数，必须在我们文件的开始部分有正确包含文件。MDL提供了两组不同的预定义函数，即标准C函数和MDL特有的函数(带有前缀mdl)。

在本书中不包括对标准C函数的解释。你可以在许多C语言教程中找到它们。MDL支持以下标准C函数：

文件操作、输入和输出：fclose，leof，ferror，fflush，fSetc，fgets，fopen，fprintf，fputc，fputs，fread，freopen，fscanf，fwrite，fseek，ftell，getc，printf，putc，remove，rename，rewind，sscanf，setbuf，setvbuf，sprintf，tmpfile，tmpnam，ungetc，unlink，vfprintf，vprintf，vsprintf。

字符的分类及转换：isalmum，isalpha，iscntrl，isdigit，isgraph，islowcr，isprint，ispuct，isspace，isupper，isxdigit，toascii，tolower。toupper。

字符串操作：strcat，strchr，strcmp，strcmpi，strcpy，strcspn，strlen，strlwr，strncat，strncmp，strncpy，strpbrk，strrchr，strspn，strstr，strok，strupr。

存存分配、缓冲区管理及数据转换：atof，atoi，atol，calloc，exit，free，getenv，malloc，memchr，memcmp，memcpy，memmove，memset，rand，realloc，srand，strlod，strol，strloul。

数学运算：acos，asin，atan，atan2，ceil，cos，cosh，exp，labs，flooy，fmod，frexp，log，log10，ldexp，ntodf，pow，sin，sinh，sqrt，tan，tanh。

日期和时间：sctime，ctime，difftime，gmtime，localtime，strftime，time。

变量：va_arg，va_end，va_start。MDL支持ANSI及K&R控制变量的方法。

main 函数

一个MDL程序虽然不是必须含有一个main函数，但是没有main函数的情况却是很少的。在MDL程序中main函数服务于三种目的：

●main是初始入口。

●它实质上是一个初始函数。

●当main返回到MicroStation时，MDL程序仍然驻留(除非对这个MDL程序卸载)。

```
main()
{
RscFileltandle  rfHandle;
/*load our command table */
if(mdlparse_loadCommandTable (NULL)= NULL)
mdloutput_error ("Unable to load command table. ");
mdlResource_openFile (&rfHandle, NULL, FALSE);
mdloutput_prompt ("Key-in in place BOX to execute");
}
```

main函数将把命令表装入MicroStation。下一章我们将讨论命令表及资源文件。

1.2.1 显示信息

MicroStation命令窗口分成6个可以显示信息的区域。这6个区域是错误信息域(er-ror)、提示域(prompt)、命令域(command)；键入域(keyin)、信息域(message)及状态域(status)。若tcb->control.inh__msg置为非零值，我们可以禁止所有信息送到这些域中。如果tcb->cont如rol.inh__err设为非零值，函数mdlOutput__error将不显示信息。

信息函数概述如下：

mdlOutput__error	在命令窗口的指定区域内显示信息
mdlOutput__prompt	
mdlOutput__command	

mdlOutput__keyin	
mdlOutput__status	
mdlOutput__errorU	在命令窗口的指定区域内显示信息，这些函数忽略禁止位
mdlOutput__promptU	
mdlOutput__commandU	
mdlOutput__keyinU	
mdlOutput__messageU	
mdlOutput__statusU	
mdlOutput__printf	根据参数 MSG__MESSAGE，MSG__ERROR，MSG__PROMPt，MSG__STATUS，MSG__COMMAND及MSG__KEYIN中的任何一的规定在命令窗口域中显示信息
mdlOutput__vprintf	同上，但根据格式串显示信息
mdlOutput__rscPrintf	在命令窗口域内显示Messagelist中的资源信息
mdlOutput__rscvPrintf	同上，但根据格式串显示信息

1.3 结构和联合

结构使我们把不同类型的相关数值组合起来。正如我们将在MDL包含文件中看到的，由于使用方便，结构被广泛使用。下面例子中，我们定义了一个叫做elm__hdr的结构，并生成一个类型ELm__hdr。这里的预处理指令＃if,＃else和＃endif在下一章讨论。

```
/*---------------------------------------------------------------+
|   ELement Header structure-common to all MicroStation elements |
+---------------------------------------------------------------*/
typedef struct elm_hdr
 {
 UInt16   type;     /* element type */
#if !defined (BITFIELDS_REVERSED)
 UInt16  reserved:6;  /* reserved for future flags - always 0 */
 UInt16  archive:3;   /* modified or added and not yet committed to history; or conflict
                      */
 UInt16  deletedCmplxHdr:1;  /* deleted complex by header */
 UInt16  nonModel:1;  /* true if this element is file-specific and does not belong in any
                      model */
 UInt16  locked:1;    /* element is locked */
 UInt16  isGraphics:1; /* element is a graphics element, has Dsp_hdr */
 UInt16  isComplexHeader:1;  /* is a complex header, must have number of elements following
                             its header */
 UInt16  complex:1;   /* this element is part of a complex element */
 UInt16  deleted:1;   /* this element is deleted */
```

```
#else
    UInt16  deleted:1;    /* this element is deleted */
    UInt16  complex:1;    /* this element is part of a complex element */
    UInt16  isComplexHeader:1;  /* is a complex header, must have number of elements following
                                   its header */
    UInt16  isGraphics:1; /* element is a graphics element, has Dsp_hdr */
    UInt16  locked:1;     /* element is locked */
    UInt16  nonModel:1;   /* true if this element is file-specific and does not belong in any
                          model */
    UInt16  deletedCmplxHdr:1;  /* deleted complex by header */
    UInt16  archive:3;   /* modified or added and not yet committed to history; or conflict */
    UInt16  reserved:6;  /* reserved for future flags - always 0 */
#endif
    UInt32   elementSize; /* number of words in element */
    UInt32  attrOffset;   /* offset (in words) from start of element to
                          attributes */
    UInt32   level;       /* element level */
    ElementID uniqueId;   /* unique ID of element */
    double   lastModified; /* last time this element was changed */
    } Elm_hdr;
```

为了使用这个结构，我们必须定义类型为ELm__hdr的变量，并用点运算符存取它。

例如我们定义类型为ELm__hdr的变量hdr：

```
ELm__hdr     hdr;
```

为了改变元素的层，我们将使用语句：

```
hdr.level=3
```

1.3.1 元素的联合

联合是一种特殊的结构，它允许一个变量存贮多种类型的值。但是在任何时候只有一个值可以占用这个变量，这在Fortran语言中相当于“等价”。TCB变量UCBYT、UCWRD和UCASC都是联合概念的例子。

联合是一种允许我们以相对容易的方法来处理数据的有力手段。考虑下面的元素联合 msELementUnion。在这个联合中，我们写进了可能加到DGNBUF内的每一个 MicroStation 元素。固定的元素头ELm__hdr也在这个联合内。

```
/*+---------------------------------------------------------------------------+
 |          name              element_unio   -   union   of   all   element   types              |
 +---------------------------------------------------------------------------*/
  typedef union msELementUnion
    {
    Elm_hdr       ehdr;
    Header    hdr; /* NOTE: hdr.dhdr is not valid unless ehdr.isGraphics is set */
    Cell_2d       cell_2d;
    Cell_3d       cell_3d;
```

```
Line_2d       line_2d;
Line_3d       line_3d;
Line_String_2d      line_string_2d;
Line_String_3d      line_string_3d;
Text_node_2d  text_node_2d;
Text_node_3d  text_node_3d;
Complex_string      complex_string;
Ellipse_2d    ellipse_2d;
Ellipse_3d    ellipse_3d;
Arc_2d    arc_2d;
Arc_3d    arc_3d;
Text_2d       text_2d;
Text_3d       text_3d;
Point_string_2d     point_string_2d;
Point_string_3d     point_string_3d;
Cone_3d       cone_3d;
Surface       surf;
Bspline_pole_2d     bspline_pole_2d;
Bspline_pole_3d     bspline_pole_3d;
Bspline_curve       bspline_curve;
Bspline_surface     bspline_surface;
    Bspline_weight      bspline_weight;
    Bspline_knot  bspline_knot;
    Bsurf_boundary      bsurf_boundary;
    Raster_hdr    raster_hdr;
    Raster_comp       raster_comp;
    ApplicationElm      applicationElm;
    ColorTable    colorTable;
    ReferenceFileElm    referenceFileElm;
    Int16     tmp[MAX_ELEMENT_WORDS];
    Int16     buf[MAX_INTERNAL_ELEM_WORDS];
    } MSELementUnion,     MSELement;
```

这样，我们就可以定义MSELementUnion类型的变量 el：

```
MSELementUnion  el;
```

当我们读一个元素时，我们不能确定有哪些元素。使用元素联合，我们可以测试固定的元素头el.ehdr。在下面的例子中我们可以检查一个线元素，使用联合我们可以访问 el.line_2d中同样的数据。

```
if(el.ehdr.type=LINE__ELM
{
el.Line_2d.start.x +=100;
el.Line_2d.start.y +=100;
}
```

请注意我们是如何使用一个联合成员从联合中提取数据，并用另一个成员来操作信息的。在这个例子中我们使一个 2D 线的起点向右移动 100 个单位并向上移动 100 个单位。

1.4 指针

在 MDL 中对指针可能产生许多错误理解。和使用变量一样，我们可以用 int，float，double 及 char 来说明指针。考虑 MDL 中指针的普通用法。

注意:参数表中*PString 变量的定义,在本书中,我们要使用 ANSI 方法定义　参数表中的变元类型,这种方法有检查变元类型的优点。

```
private void parselnputstring(char * pString)
{
char*pResult;
if(pResult=strtok(pString.""))!=NULL)
{
strncpy(oldstring.pResult.sizeof(newString));
if((pResult = strtok(NULL""))! = NULL)
   STRRCPY(newString.pResult.sizeof(oldstring));
}
}
```

在上例中，我们看到了两个字符指针*pString 和*pResult。星号(*)告诉程序到存贮在指针 pString 内的内存地址去寻找字符串的值。

1.4.1 通过引用传递

这里有两种方法传递参数，即通过值传递和通过引用传递。当我们通过引用来传递参数时，我们把变元的地址送给接收参数。因此，被调用程序中变元的任何变化都会改变调用程序中变元的值，因为二者共享相同的内存地址。我们必须通过引用来传递字符串。我们可以通过使用指向字符串地址的指针来实现字符串的传递，当一个字符串是一个字符数组时我们不需要用&，并且在 MDL 中所有数组都用引用来传递。

例如，字符串的说明告诉我们必须当作一个指针把它传递给 parselnputstring。

```
char string[20];       /*definition of string on main program*/
.
.
.
parseInputstring(string);/*pass the string into our function*/
.
.
.
Private void parselnputString(pstring) /*pass itby reference*/
char *pstring;              /*define as a pointer*/
```

回来看我们的main函数，结构rfHandle当作一个指针传递。字符(&)是地址操作符。它返回到内存中一个变量的地址，而不是变量的值。

```
 mdlResource__OpenFile(&rfHandle,NULL,FALSE);
```

程序mdlResouce__openFile是一个预定义函数。当我们在MDL手册中查找它时，我们可以看到第一个变量必须是一个指针。通过在rfHandle前面使用&符号，我们给函数提供了正确格式的变量。

1.4.2 通过值传递

我们也可以通过值来传递变量。就是说这个变量被赋值，同时这个值被拷贝到接收参数。被调用程序中变量的改变是局部的。当程序结束时它的值就会丢失。请看下面例子，这个程序要在视图2中放置一段弧。其中，viewNumber通过值来传递。不管我们对接收参数做了些什么，当我们完成调用时，变量没有变化。

```
  ..
 vieWNumber=2;
 placeArcbyCenter(viewNumber)
.
.
Private void placeArcbyCenter (int view)
 {
 MSELementUnion  arc;
 Dpoint3d        arcp[3];
 double          origin;

 arcp[0].x=25.;         /*start point*/
 arcp[0].y=35.;
 arcp[0].Z=fc__zero;    /*floating point constant*/
 arcp[1].X=55.;         /*center point*/
 arcp[1].y=35. ;
 arcp[1].Z=fc__zero;    /*floating point constant*/
 arcp[2].X=85.;         /*end point*/
 arcp[2].y=35.;
 arcp[2].Z=fc__zero;    /*floating point constant*/
 origin=55.0;

 if(mdlArc__createByCenter(&arc,Null,arcpt,True,origin,view)==success)
   {
    mdlELement__display(&arc,NORMALDRAW);
    mdlELement__add(&arc);
    }
 }
```

注意: *MDL 数组符合C语言的约定，从0开始。例如，在上面的定义 Dpoint3darcP[3]中，数组下标是arcp[0]，arcp[1]和arcp[2]。*

函数指针

在MDL中我们可以把函数的地址做为一个变元来传递。因此，程序可以通过引用指针的值来调用这个

函数。函数指针广泛地应用于MDL。例如，函数mdlState__startPrimitive要求前两个变量是指向的函数指针。

mdlstate__startPrimitive(placeBOX__FirStPOint, placeBOX__start,1,2)：

函数PlaceBox__firstPoint和placeBox__Start必须在引用它们以前出现。我们可以在程序开始地方定义函数名来满足这一要求；或者在调用它以前有完整的函数出现(这是贯穿书的用法)。

1.5 MicroStation--状态机器

MicroStation是一个事件驱动机器。就是说，在任何一点上，一个事件使MicroStation进入一种给定的状态，并有状态处理程序处理对它的输入。我们可以通过调用PLACELINE命令及放置线上的第一个点进行演示。当我们移动光标时，第二个点像拉着一根橡皮筋一样在屏幕上移动，我们将选择WINDOWAREA命令并选择新的窗口，而不是给出第二个数据点。在窗口更新时，这条线消失了吗?作为一个状态机器，MicroStation可区分命令状态。视图命令与放线命令有不同的状态。因此，我们可以假设这条线仍然是激活的，按reset钮使我们退出视图命令并返回放置线的命令。

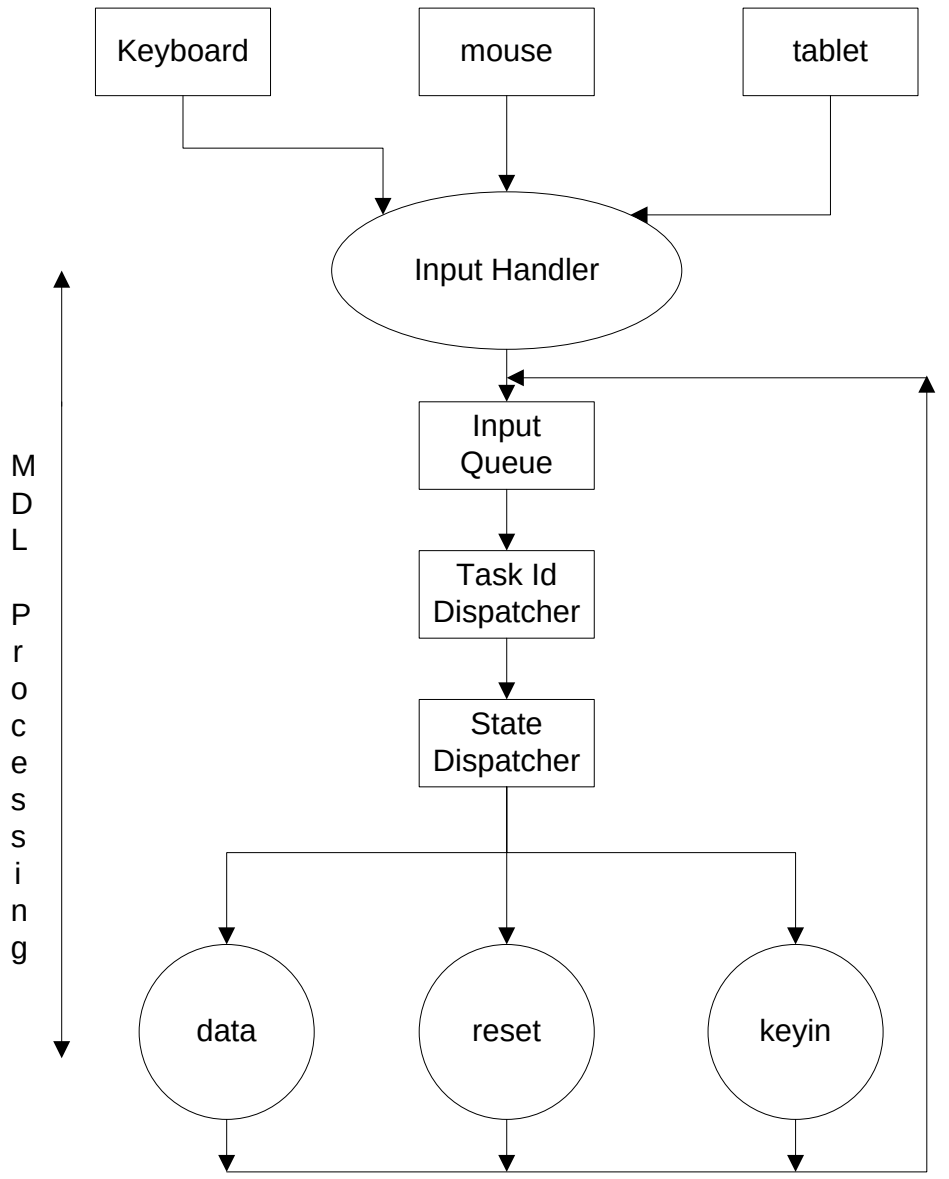


图1.2 MicroStation输入流程

我们可以写一个MDL程序建立状态函数来处理诸如数据点或键入信息这样的输入信息。用这种方法我们可以进入MicroStation的基本(primitive)层。我们写的任何MDL应用程序都会有以上的方便，而且不需要另外编写程序。

MicroStation程序设计的其它工具，如UCM和MicroCSL，是把MicroStation的命令排列起来。通过图1．3我们可以看出，UCM和MicroCSL在输入队列、任务分配和外部应用之间循环。因此，在状态分配层上排列的应用将不受控制。

1.6 资源

资源是一组存贮在被叫做资源文件中的数据。典型的数据组是字符串、错误信息、光栅图符及命令表。使用资源文件的目的是要把数据和应用源码分开。使用这种方法有许多优点:

●减少对内存的总需求，因为只安装需要的数据。

●多个应用可以共享同一个资源。

●维护方便。对提示和信息的改变不需改动应用程序。例如，可把应用转换成其他语言,如法语。

在以后的章节里，我们将讨论资源及其生成的方法。

1.6.1 命令表

下边的main程序从资源文件中安装了应用命令表。命令表定义了一种命令语言。为了进行练习，我们假定main将把命令PLACEBOX装入MicroStation。在以后章节中我们还要讨论命令表。

```
main()
{
RscFileHandle rfHandle;
if(mdlParse__loadcommandTable(NULL)=NULL)
mdloutput__error(“Unable to load command table.”);
mdlResource__openFile(&rfHandle,NULL,FALSE);
mdloutput__prompt(“key-in PLACE BOX to execute”);
}
```

在这里我们告诉MicroStation,我们的应用命令只是存在,我们并没有执行程序。这是MDL的优越之处。它允许应用的开发者进入MicroStation"基本”层。我们现在定义了一个称为PLACEBOX的基本命令。

为了激活我们的MDL程序，我们应该在uStn>提示下键入PLACE BOX。Microsration将一个字一个字地分析我们的键入内容。因此，只键入PLACE将产生含混的命令。MicroStation不能把它与PLACELINE区分开来。生成一个名叫PLACEBOX的基本命令将不能使用命令表，因为命令没有层次，使该命令无法分类。命令分类非常重要，它允许MicroStation决定或分配应用的当前状态。在我们给PLACEBOX应用生成命令表时，我们还要详细讨论这个问题。

当MicroStation成功地分析用户的键入信息后，就生成了一个命令编号。一个与命令表相联系的应用的任务ID和命令编号用来确定应用执行这个命令并调用这个函数。在下面的程序中，cmdNumber使函数placeBox__start和一个命令编号相结合。当我们键入PLACEBOX时，MDL开始从PlaceBox__start执行。

```
cmdName  placeBOX__start()
cmdNUmber CMD__PLACE__BOX
{
mdlstate__startPrimitive(placeBox__firstpoint,placeBox__start,1,2);
}
```

程序mdlState__startPrimitive将建立状态函数，以便用一个数据点激活PLaceBOx__firstpoint,一个rest将循环地执行placeBOx__start。同时，它将从0号信息表产生的1号信息，并把它显示在命令域，而把2号信息显示在提示域。在第二章我们将讨论息表及其生成方法。

I.7 状态控制函数

一个由MicroStation命令序列组成的应用，在执行另一个应用之前要结束它的最后一个命令。在执行命令PLACELINE中，执行一个用户命令开窗口，这要有明确的提示：WINDOW AREA。写一个作为事件驱动应用的MDL程序不会使这个应用被挂起。确定几个函数来控制这个事件。在这一节我们将讨论不同的状态控制函数，这些函数是MDL程序设计核心。共有四种MicroStation命令状态。

一．基本命令用于生成、修改及删除元素。启动一个基本命令会终止上一个基本命令，并且在另外一个基本命令取代它之前，该命令一直处于激活状态。启动一个基本命令必须调用一下命令：

mdlstate__startPrimitive	启动一个只用于生成元素的基本命令
mdlstate__startModifyCommand	启动一个用于放置及修改元素的基本命令
mdlstate__startFenceCommand	建立一个用作篱笆操作基本命令的函数

二．视图命令状态

视图命令用于修改或更新视图。我们必须调用mdlState_startViewCommand启动视图命令。启动一个视图命令将挂起一个基本命令，当视图控制结束时，挂起来的基本命令继续工作。如果用户输入reset，我们必须用mdlState_exitViewCommand结束视图命mdlState_startViewCommand启动一个视图命令

mdlState_exitViewCommand　退出视图命令.

三．直接命令状态

直接命令用于修改设置(如改变捕捉状态，或键入WT=)。直接命令不调用状态函数，因此也不影响基本命令或视图命令的状态。

四．实用命令状态

实用命令执行非交互式功能，如COMPRESS.在我们结束实用命令时需用调用

mdlState__startDefaultCommand。

其余的状态控制函数如下：

mdlState__clear	重置命令状态，以便没有命令被激活后
mdlState__checkSingleShot	检查命令是否以一次运算的(SingleShot)方式操作
mdlState__dynamicUpdate	规定函数使用简单动态操作
mdlState__registerStringIds	规定一个基本命令启动时使用的提示信息
mdlState__restartCurrentComman	重新启动当前基本命令
mdlState__setFunction	根据事件规定使用的函数
mdlState__setKeyinPrompt	规定用于键入区的字符串，缺省值为”nSTN>″
mdlState__startDefaultCommand	当前命令结束后启动缺省命令

1.7.1 用户函数

当某些事件在MicroStation中出现时，MDL用函数指针来指定要执行的用户提供的函数。函数指针使程序设计方便而有力。例如，当某些事件发生时，函数mdlState__startModifyCommand要执行5个用户函数。这些事件包括用户输入reset、输入数据点或动态显示。剩下的两个参数显示或清除，用于显示

在DGNBUF中被修改的元素。显示和清除经常一起配对使用，以便清除可以恢复到先前显示的函数。我们可以向这些变元传递NULL。以下是用户函数清单：

注意：斜体字的函数名不是函数真实名字。请用你的程序中使用你用的函数名替换它们。

userState__clean	用于修改函数的函数
userState__CommandCleanUp	在另一个命令启动前需要清除时使用的函数
userState__complexDynamicUpdate	用于复杂动态显示的函数
userState__datapoint	在输入数据点时调用的函数
userState__dynamicUpdate	用于简单动态显示的函数
userState__fenceContent	用于处理篱笆内容的函数
userState__fenceOutline	用于重新显示篱笆边界的函数
userState__Keyin	用于处理用户键入的函数
userState__reset	用于处理重置(reset)的函数
userState__Show	当修改命令装入DGNBUF时用于显示元素的函数

返回来看我们的PLBOX例子，我们需要建立放置方框的基本命令。当用户传送数据点时，第一个参数placeBox—firstPoint是要执行的函数的指针。第二个参数是用户输入reset时调用的函数。在这种情况下我们循环并重新开始放置方框的基本命令。最后的两个参数用于定义屏幕上的信息。我们将在第二章讨论信息表。

```
cmdName   placeBOx()
cmdNUmber   CMD__PLACE__BOX
{
   mdlOutput__rscprintf(MSG__RPOMPT,NULL. 0，2)
   mdlState__StartPrimitive(PlaceBOX__firStPoint，placeBOX__start,1,2);
}
```

事件类型

我们来看函数placeBox—firstPoint，它需要从mdlState_startPrimitive传来的一个数据点。根据从用户处接收的信息，我们将存贮这个点并建立要调用的状态函数。函数mdlState—setFunction需要两个变量。第一个是事件的类型，第二个是要执行函数的指针。事件类型可能出现的值为：

STATE__DATAPOINT当用户输入一个数据点时

STATE__RESET当用户输入reset时

STATE__KEYIN当用户键人时

STATE__COMPLEX__DYNAMICS产生动态显示，见下

STATE__COMMAND__CLEANUP在另一个命令开始前清除当前命令

MDL提供提供对元素的动态操作。当元素生成、移动及修改时可以在屏幕上看到橡皮筋一样的显示。在我们需要单个元素动态显示时使用简单动态。复杂动态用于显示一个以上的元素。mdlState__dynamicUpdate是用于简单动态的函数。

对于复杂动态我们必须使用mdlState__setFunction。为了决定显示动态的方式，MDL执行用户函数generatelmage，并以高亮度方式显示元素。因此元素只是显示，而不存盘。我们将在讨论PLBOX使用动态显示时，将看到这个函数是如何工作的。

```
Private void placeBox__firstpoint
```

```
(
Dpoint3d  *pt,
int       view
)
{
/*save first point*/
pntP[0]=*pt;

/*set the datapoint state function for the second point.*/
mdlstate__setFunction(STATE__KEYIN,keyinText);
mdlstate__setFunction(STATE__DATAPOINT,placeBox__secondpoint);
mdlstate_setFunction(STATE_RESET, placeBox_done);
mdloutput_rscPrintf(MSG_PRCMPT,NULL, 0, 3);

mdlstate__setFunction(STATE__COMPLEX__DYNAMICS,generatelmage);
}
```

正如我们从placeBox__firstpoint函数看到的，对于每个可能的事件我们都建立了相应的状态函数。程序KeylnText将从输入队列中取字符并把它存在变量textin中。

```
Private void keyin Text()
{
  if (!*statedata.cmdstring)
    return;
  strncpy(textin,statedata,cmdstring,sizeof(textin));
}
```

应该特别注意函数generatelmage，因为它是每次我们移动光标时所调用的程序。在下行中，我们告诉MDL在动态操作中使用这个函数。

```
mdlState__setFunction(STATE__COMPLEX__DYNANICS.generatelmage)
```

generatelmage将放置这个字符串及一个方框，并把那条线与方框的一个角对齐。

1.8 元素函数

以下是操作单个元素的通用元素函数，我们将在以后的章节中讨论复杂元素的操作

mdlElement_add	对文件加上一个元素
mdlElement_append	对文件附加一个元素
mdlElement_appendAttributes	对一个已有元素附加属性数据
mdlElement_display	在所有视图中显示元素
mdlElement_displaylnSelectedView	在指定视图内显示元素
mdlElement_extractAttributes	从已有元素中提取属性数据
mdlElement_getFilepos	返回元素的文件位置
mdlElement__getProperties	返回元素的属性信息
mdlElement_getSymbology	返回元素的显示线符
mdlElement_getType	返回元素的类型号

mdlElement_isFilled	如果元素被填充则返回TRUE
mdlElement_offset	使元素移动一段距离
mdlElement_read	把一个元素从设计文件读到缓冲区
mdlElement_rewrite	重写一个已有元素
mdlElement_setFilePos	建立元素的文件位置
mdlElement__setProperties	建立元素的属性信息
mdlElement__setSympology	建立元素的显示线符
mdlElement__size	返回元素大小的字节数
mdlElement__stripAttributes	卸下一个已有元素所有的属性信息
mdlElement__stroke	通过把一个元素转换成一系列向量来处理一个元素
mdlElement__transform	通过变换矩阵来变换一个元素
mdlElement__undoableDelete	删除一个用UNDO命令能恢复的元素

1.8.1 元素生成函数

MDL提供了一些生成MicroStation可显示元素的函数。这些元素在内存中生成。因此，使用函数mdlELement__add把元素写在文件中是很重要的。元素的结构在<mselems. h>中定义。

使用这些函数生成元素时，把编码与随后的元素格式的修改分开是一种很好的工作习惯。对于两个MicroStation新--尺寸和多线，这一点很明显，<mselems. h>中没有详细的元素结构。

下面列出了元素生成函数：

mdlArc__create	生成一个弧元素
mdlArc__createByCenter	通过中心和两个端点生成一条弧
mdlArc__createByPoint	通过3点生成一条弧
mdlCell__create	生成单元的头元素
mdlCircle__createBy3Pts	用3个点生成圆、椭圆
mdlCone__create	生成一个圆锥元素
mdlCone__createRightCylinder	生成一个正圆柱元素
mdlCone__create	生成一个曲线元素
mdlEllipse__create	生成一个椭圆元素
mdlLine__create	生成一个线元素
mdlLineString__create	生成一个线串、轮廓或曲线元素
mdlPointString__create	生成一个点串元素
mdlText__create	生成一个文本元素
mdlTextNode__create	生成一个文本结点元素

我们将用mdlText__create生成字符串。我们生成字符串的方式没有什么特殊，但是在显示时，需要考虑采用动态显示。

绘图方式

当我们用mdlELement_display在屏幕上显示一个元素时，可能的绘图方式是：

绘图方式	**含　意**
NORMALDRAW	用一般颜色画元素
ERASE	从屏幕上擦除

HILITE	用高亮度颜色画元素
TEMPDRAW	画临时元素
TEMPERASE	擦去临时图形
TEMPROTATE	使用“异”操作或半调色
XORDRAW	使用“异”操作

这个动态程序在屏幕上临时绘出我们的图象，并当我们移动光标时擦除它。当我们接收第二个点的位置时，函数placeBox—secondpoint被激活，因为先前已经建立了这种状态。

```
privatevoid placeBox__secondpoint
(
Dpoint3d*pt
int  view
)
{
generatelmage(pt,view,NORMALDRAW);
}
```

函数 placeBox__Secondpoint 直接调用 generatelmage，并传递给它一个NORMALDRAW的选项。我们将用PLBOX中的元素生成一个孤立单元。调用Mdlcell_begin将生成类型2的单元头。从此以后放置的每一个元素将成为这个单元的一部分。在generatelmage中我们可以看到NORMALDRAW将把元素加在DGN文件内。

```
/*createText in dgnBuffor MicroStation Dynamicsto display*/
mdlText__create(&el,NULL,textin,&pntp[1], NULL,NULL,NULL,NULL);
mdlELement__display(&el,drawMode);
if (drawMode==NORMALDRAW)
{
   CellFilepos=mdlCell__begin(”plbox”, &origin,NULL,NULL,0);
   mdlELement__add(&el);
}
```

1.8.2 元素提取函数

当我们生成一个字符串时,它仍然在DGNBUF中。只有当我们调用mdlELement__add时，它才被加在DGN文件中。下一步，我们在字符串周围放置一个矩形；我们可以用一个元素捉取函数来取出元素的信息。这些函数允许程序员检索元素信息，无需了解元素的存贮格式。程序员最好使用这种函数，以便保护自己的程序不至因为元素格式的改变而改变。

下面是元素提取函数清单

mdlArc__extract	提取弧或椭圆元素
mdlCell__extract	提取单元头信息
mdlCone__extract	提取圆锥元素信息

mdlLinear__extract	从线、线串、轮廓形、圆锥形、曲线、多义线或B曲线中提取坐标数组
mdlLinear__getClosestSegment	离一个点很近的一条线性元素上一点的附近寻找一个线段
mdlSharedCell__extract	提取共享单元实例或它的定义信息
mdlText__extract	提取文本元素信息
mdlText__extractShape	提取围绕文本元素的最小边框
mdlText__extractString	从文本元素中提取字符串
mdlTextNode__extract	提取文本结节元素
mdlTextNode__extractShape	返回围绕一个文本结点的最小的边框

我们要用mdlText__extractShape得到矩形的5个点(第一个点与最后一个点重合)。把第四个变量设为TRUE，依照当前捕捉精度使边框大些(和SETLOCATE的键入相同)。我们没有从DGN文件中读字符串，因为上一个操作使DGNBUF中已经有了这个字符串.

```
mdlText__extractshape(Shapep. NULL, &el, TRUE, view)
mdlShape__Create(&el. NULL, Shapep, 5, -1);
mdlElement__disPlay(&el, drawMode);
if(drawMode＝＝NORMALDRAW)
mdlELement__add(&el);
```

我们程序的下一步是使引线对准框的一角。图1. 3中的点划线表明，如果我们得到用户输入的两个点，将画出所需要的线。用这种简单的几何方法，根据第一个点所在的象限我们很容易地把这条线的端点确定在矩形的一个合适的角上。

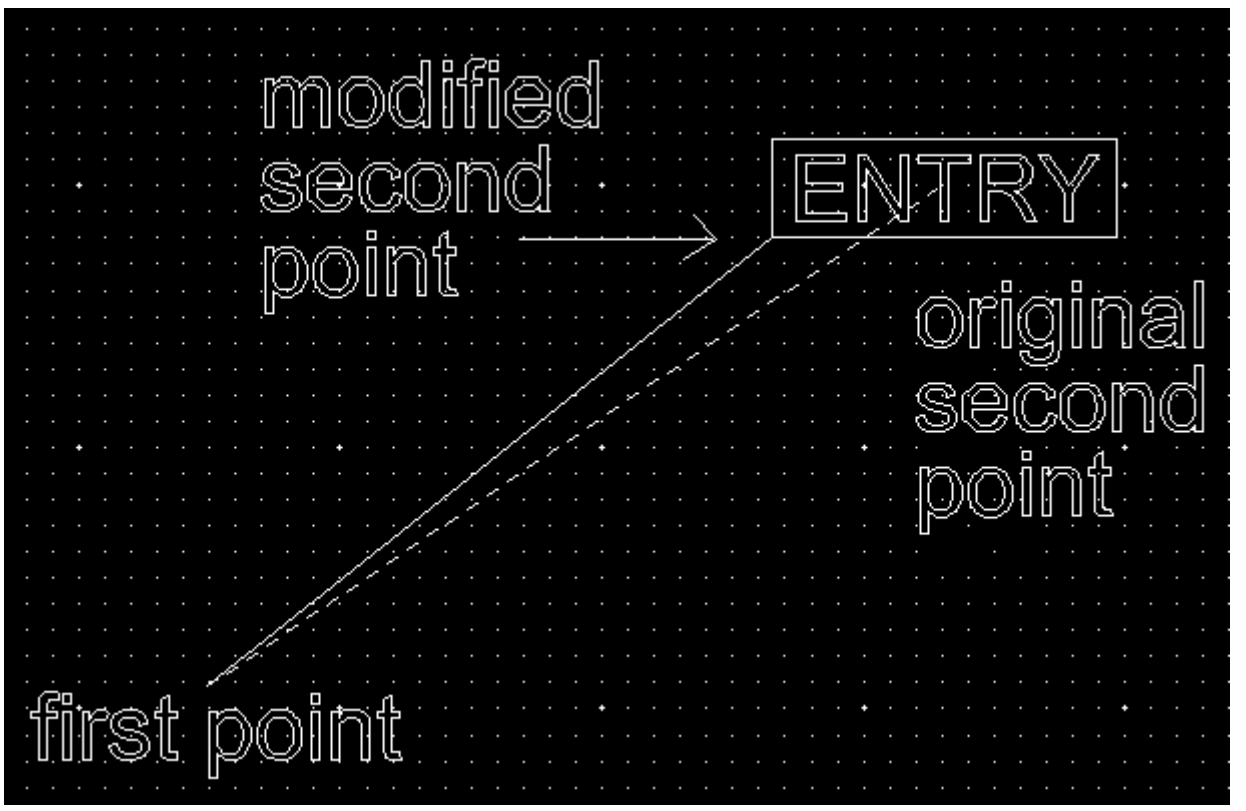


图 1.3 放置引线

下面是计算新的端点:

```
if (pntP[0].x < origin.x)
{
     if (pntP[0].y < origin.y)
     {
         pntm[1].x=shapep [0].x;
         pntP[1].y=shapep [0].y;
```

```
        }else{
            pntP[1].x=shapep [3].x;
            pntm[1].y=shapep [3].y;
        }
    }
    else
    {
        if (pntP[0].y < origin.y)
        {
            pntm[1].x=shapep [1].x;
            pntP[1].y=shapep [1].y;
        } else {
            pntP[1].x=shapep [2].x;
            pntP[1].y=shapep[2].y;
        }
    }
```

一旦计算出新的端点，我们就可以使用 mdlLine_create 来放置这条线。通过调用 mdlCell_end 来关闭这个单元，它用描述字、组成元素编号和正确的文件指针更改单元头。

```
    / * place the modified leader line * /
    mdlLine_Create(&el, NULL, pnpt);
    mdlELement_display(&et,drawMode);
    if (drawMode = NORMALDRAW)
    {
       mdlELement_add(&el);
       mdlCell_end (cellFilePos):
```

我们把学习的第一个 MDL 程序 plbox.mc 完全列出如下：

```
  +--------------------------------------------------------------------------+
   |     Example MDL function to place a box around                           |
   |     a text string with a leader line.                                    |
  +--------------------------------------------------------------------------*/
  /*+-------------------------------------------------------------------------+
   |     Include Files                                                        |
  +--------------------------------------------------------------------------*/
#include <cmdlist.h>
#include <toolsubs.h>
#include <string.h>

#include <msrsrc.fdf>
```

```
#include <msparse.fdf>
#include <msoutput.fdf>
#include <mselemen.fdf>
#include <msvar.fdf>
#include <msstate.fdf>
#include <mssystem.fdf>
#include <msdialog.fdf>
#include <mscell.fdf>
#include <mselmdsc.fdf>

#include "PlaceBox.h"
#include "PlaceBoxcmd.h"

static char textin [128];
Dpoint3d   pntP[2];

/*----------------------------------------------------------------------+
|      name      generateImage  -  dynamic function for box.            |
+----------------------------------------------------------------------*/
Private int generateImage
(
Dpoint3d*pt,
int    view,
int    drawMode
)
{
MSElementUnion el;
Dpoint3d  origin;
Dpoint3d  shapeP[5];
MSElementDescr * pDescr = NULL;

pntP[1]=origin=*pt;

mdlCell_create (&el, "BOX", &pntP[0], FALSE);
mdlElmdscr_new (&pDescr,NULL,&el);

/*create Text in dgnBuf for Microstation Dymamics to display */
mdlText_create (&el,NULL,textin,&pntP[1],NULL,NULL,NULL,NULL);
mdlElement_display(&el,drawMode);
mdlElmdscr_appendElement(pDescr,&el);

mdlText_extractShape(shapeP,NULL,&el,TRUE,view);
```

```
mdlShape_create(&el, NULL, shapeP, 5, -1);
mdlElement_display(&el, drawMode);
mdlElmdscr_appendElement(pDescr,&el);

if (pntP[0].x < origin.x)
{
    if (pntP[0].y < origin.y)
    {
    pntP [1] .x=shapeP [0] .x;
     pntP[ 1] .y=shapeP[0] .y;
  } else {
    pntP[1] .x=shapeP [3].x;
    pntP [1] .y=shapeP [3].y;
  }
}else {
    if (pntP[0].y < origin.y)
  {
    pntP[1] .x=shapeP[1] .x;
    pntP[1].y=shapeP[1].y;
  } else {
    pntP[1] .x=shapeP[2] .x;
    pntP [1] .y=shapeP[2] .y;
  }
}
/* Place the modified leader line */
mdlLine_create(&el, NULL, pntP);
mdlElement_display (&el, drawMode);
mdlElmdscr_appendElement(pDescr,&el);
if (drawMode==NORMALDRAW)
{
    mdlElmdscr_add(pDescr);
}
mdlElmdscr_freeAll(&pDescr);
return  SUCCESS;
}

/*-------------------------------------------------------------------------+
|      name      keyinText                                                 |
+-------------------------------------------------------------------------*/
Private void keyinText(char  *cmdStringP)
{
strcpy(textin,cmdStringP);
```

```
}

/*-----------------------------------------------------------------------+
 |      name       placeBox_done                                         |
 +-----------------------------------------------------------------------*/
Private void placeBox_done()
{
mdlOutput_rscPrintf(MSG_PROMPT,NULL,0,4);
mdlState_restartCurrentCommand();
}

/*-----------------------------------------------------------------------+
 |      name       placeBox_seconPoint                                   |
 +-----------------------------------------------------------------------*/
Private void placeBox_secondpoint
(
Dpoint3d *pt,
int  view
)
{
generateImage(pt,view,NORMALDRAW);
}

/*-----------------------------------------------------------------------+
 |      name       placeBox_firstPoint                                   |
 +-----------------------------------------------------------------------*/
Private void placeBox_firstpoint
(
Dpoint3d*pt,
int   view
)
{
pntP[0]=*pt;
/*Set the datapoint state function for the second point.*/
mdlState_setKeyinPrompt ("KEY IN TEXT:");
mdlState_setFunction (STATE_KEYIN,keyinText);
mdlState_setFunction (STATE_DATAPOINT,placeBox_secondpoint);
mdlState_setFunction (STATE_RESET,placeBox_done);
mdlOutput_rscPrintf (MSG_PROMPT,NULL,0,3);

/*Setup Rubber Banding function*/
mdlState_setFunction (STATE_COMPLEX_DYNAMICS,generateImage);
```

```
}

/*----------------------------------------------------------------------+
|      name      placeBox_start                                         |
+----------------------------------------------------------------------*/
void placeBox_start()
cmdNumber   CMD_PLACE_BOX
    {
    mdlState_startPrimitive(placeBox_firstpoint,placeBox_start,1,2);
    }

/*--------------------------------------------------------------------------------**//**
* @description  PlaceBox_unloadFunction
* @param   unloadType      Reason for the unload of the app
* @return  SUCCESS to allow the unload; ERROR to disallow the unload
* @bsimethod                          BSI             06/03
+---------------+---------------+---------------+---------------+---------------+------*/
Private int     PlaceBox_unloadFunction
(
int     unloadType
)
    {
    BoolInt     disallow = FALSE;
    /*-----------------------------------------------------------------
    |   If the value of unloadType is negative, then MicroStation ignores
    |   the return value of this function.  In this case, we can assume
    |   that MicroStation is shutting down or was terminated so this
    |   function should not attempt to abort the unload.
    +----------------------------------------------------------------*/
    if (unloadType < 0)
        {
        return FALSE;
        }
    /*
    ** Your processing, which may include setting the disallow flag to TRUE
    */
    return  (disallow ? ERROR : SUCCESS);
    }

/*--------------------------------------------------------------------------------**//**
* Dialog/Item Hooks; Command Numbers and Command Names
+---------------+---------------+---------------+---------------+---------------*/
```

```
typedef void (*Handler)(char *);
typedef void (*Hook)();

/*----------------------------------------------------------------------------**//**
* @description  MdlMain
* @param   argc     Count of arguments into the function
* @param   argv     Array of arguments
* @return  Return code from the application
* @bsimethod                          BSI             06/03
+---------------+---------------+---------------+---------------+---------------*/
Public int main
(
int         argc,
char        *argv[]
)
    {
    RscFileHandle   rscFileH;

    /*-----------------------------------------------------------------------
    mdlResource_openFile:
    This function opens a resource file, thus making its contents
    available to the application.  In this case, we need to open
    treexmpl.ma as a resource file so that we have access to the the
    string lists for our messages.
    -----------------------------------------------------------------------*/
    if (SUCCESS   != mdlResource_openFile  (&rscFileH, NULL, 0))
    {
    mdlOutput_rscPrintf (MSG_ERROR, 01, MESSAGELISTID_Messages,
                MESSAGEID_ResourceLoadError);

    /* unload the application */
    mdlDialog_cmdNumberQueue (FALSE, CMD_MDL_UNLOAD,
                 mdlSystem_getCurrTaskID (), TRUE);
    }

    /* --- Set up function to get called at unload time --- */
    mdlSystem_setFunction (SYSTEM_UNLOAD_PROGRAM,    PlaceBox_unloadFunction);
    mdlState_registerStringIds(MESSAGELISTID_Commands,       MESSAGELISTID_Commands);
    mdlParse_loadCommandTable (NULL);
    mdlOutput_prompt("Key-in PLACE BOX to execute");
```

```
return 0;
}
```

1.9 第一章小结

学习一种新的程序设计语言往往是很困难的。我们相信，最好的方法是先看一个例子。如果在这个阶段你感到困难，不要失望；只要记住下面几点：

- MDL的基础是C语言。
- MicroStation是一种状态机器。利用这一点，MDL可以进入MicroStation的基本层来利用它。
- 状态机器的意思就是命令不一定依次执行，而是根据MicroStation内的事件激活部分命令。
- 用户命令和MicroCSL程序按顺序排列MicroStation命令。MDL也可以这样做，但是它真正的威力在于它可以根据状态的变化提供调用的函数。

第二章　编译 MDL 程序

上一章我们看到了如何编写一个MDL程序，而没有谈及程序的运行。在这一章，我们要讨论开发流程的下一个步骤，要学会如何生成命令表。同时，我们将学习理解资源文 件、编译、链接并执行MDL应用。

我们的MDL源程序是MicroStation执行的指令集合，是ASCⅡ字符格式文件。我们可以读这些指令，但是它们不能被MicroStation执行。在我们编译这个程序时，它被转换成一个目标程序。这是一个要链接成可执行文件的中间文件。

一个目标文件有对MDL预定义函数的调用。链接过程将分析这些调用，生成一个MDL程序。用库管理程序将目标程序、资源程序及其它程序文件互相结合产生一个MDL应用。这个应用被装入内存，作为一个MDL任务来执行。

MDL一个非常有力的功能是它的预处理指令，它们是对编译程序的指令。因此我们 应该首先了解这些指令。

2.1 预处理指令

预处理指令是MDL程序的一部分。它们是对于编译程序的指令，而不是被编译的语句。

预处理指令总是以#符号开始，我们已经在程序PLBOX中见到过这种符号。例如：

```
#include<mdl.h>
```

当编译程序读到这一行时，它将把<mdl. h>的内容读到当前文件内，使之成为当前 程序的一部分，因此，它的所有函数和数据定义都可以使用。

预处理指令另一个常见的用法是定义常量。这个指令对标识符的名字赋给一个常量的值。当编译程序在程序中发现这个标识符时，将用相应的值替代这个标识符。例如：

```
#define GRAFIC 1160
```

这种定义常量的方法由于不占用内存，所以是一种好方法。同时也使程序可读并易于维护。看看下面的例子，在这里我们用新的图形组编号来更新类型9的元素头。

```
.
int   Size=1, offset=GRAFIC;
.
mdlParams__stozeType9Variable(&tcb->graphic,size,offset);(该函数在V8版本上已经取消)
.
```

当编译程序扫描文件中的预处理指令时，它将用1160来代替GRAFIC。

2.1.1 条件编译

预处理指令也用于定义条件编译。如果一个标识符被定义，它们是编译一段编码的指令。这些指令为#if,#ifdef,#ifndef,#else,#elif,#endif和#undef。应用预处理指令的最好例子是多平台开发环境，在这里我们需要保留同一程序的不同版本。

```
#if defined(Unix)
```

```
short  prompt__color[4];
#else
short prompt__color[6];
#endif
```

上面的编码取自<tcb. h>，由于操作平台不同，其中的数据结构互不相同。在程序的开头我们可以写这样一行。

```
#define  Unix
```

这样，prompt_color是一个含有4个短整型数据的数组。否则它就是一个含有6个短整型数据的数组。内部指令为mdl，dos，pm386，mc68000,macintosh，vax，unix和ip32。

2.1.2 编译程序指令—#pragma

编译指示是一种把特定指令送给编译程序的编译指令。由于编译指示放在源码中，所以经常用于替代编译选项。MDL编译程序只支持2，version和资源ID。

一、版本编译指示

编译指示Version定义与一个MDL程序有关的版本。为了使用Version编译指示，我们把下面一行放在包含文件中

```
#pragma  Version  4:1:2
```

如果我们不规定版本编译指示，则它的缺省值为4:0:0。

版本号必须是以冒号(:)分隔开的整数。如果不指定数字，则假设为0。

指定给版本编译指示的值显示在MDL应用Detial对话框的Version域内。

二、资源ID编译指示

如果我们想把一个以上MDL程序放入一个资源文件，并且没有程序含有用于改变资源ID的编译指示，资源文件库管理程序将生成以下警告：

```
###Warning: dUplmateresourcedetected
 resourceclass=(0X4D646c49,(Mdl)resourcelD=(1296124238.0x4D41494E.(MAIN))
```

它的出现是由于两个程序都有对于MDL应用有相同的资源ID。为了避免警告，我们可以用资源ID编译指示给另一个应用赋给别的资源ID。为此，我们在资源源文件中设置以下语句：

```
#pragma   resourceJ0   ‘plreSC’
```

#pragma和资源ID必须按规定方式出现。’plresc’值可以被任何整数表达式代替。

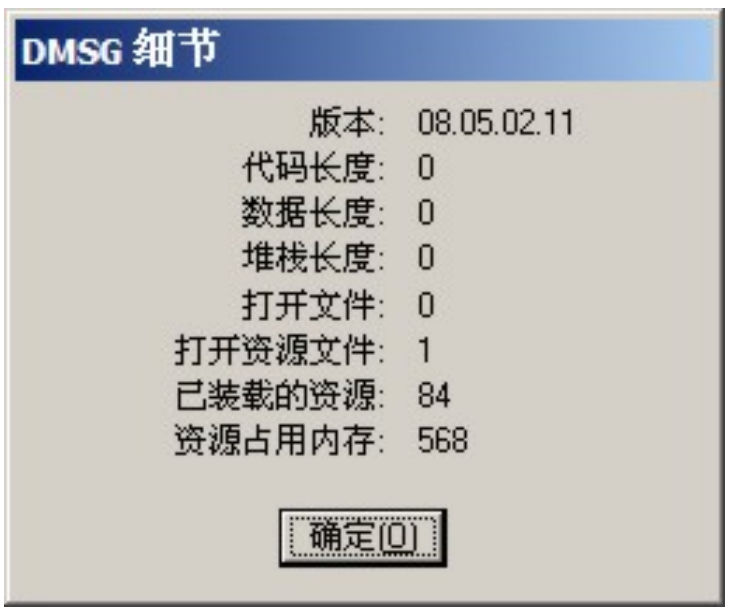


图2. 1 MDL应用Detail对话框

2.2 开发流程

编写一个MDL程序的开发流程要比编写一个典型的C程序应用要包含更多的内容。我们需要考虑命令表、资源文件及对话框。现在，我们要讨论没有对话框的 PLBOX 程序的开发流程。下一章再详细讨论对话框。

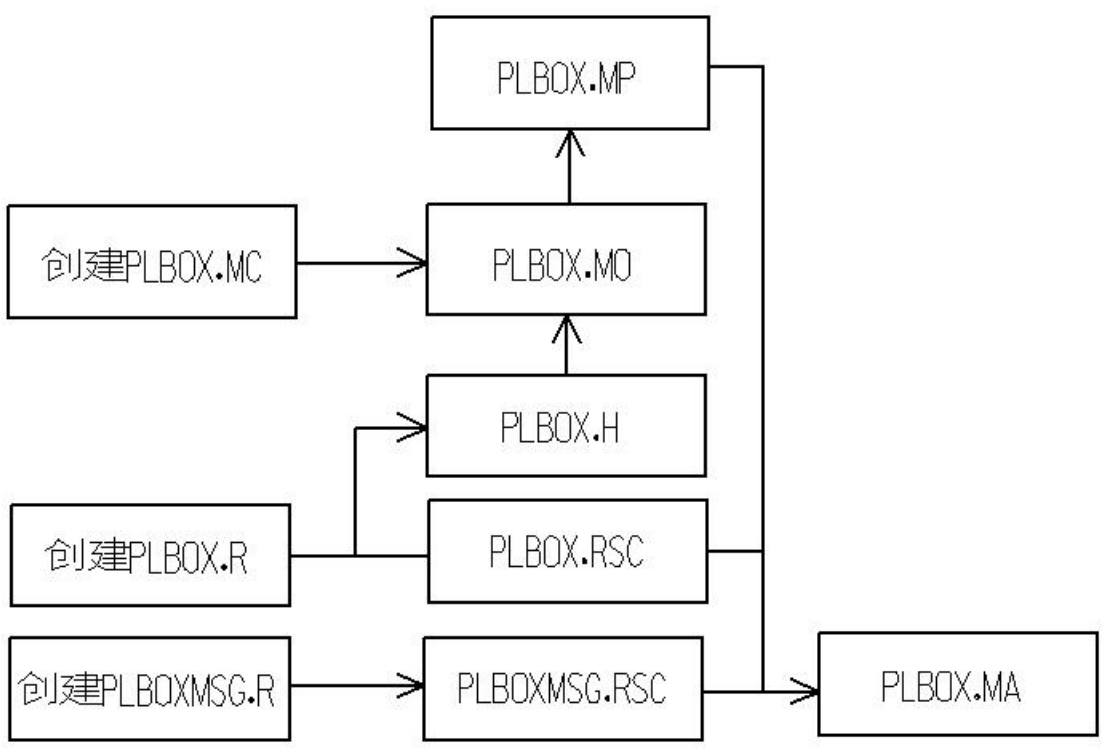


图2.2 PLBOX流程

2.2.1 开发流程概述

我们可以把这个流程分成4个主要部分,即用文字编辑程序生成及编辑源码,编译源码生成目标文件,链接目标文件和用MDL库管理程序建立应用。在开发流程的每一阶段都要使用工具。这些工具可以在\microstation\mdl\bin 中找到。在使用unix操作系统的机器上，这些文件在\usr\ip32'~,mstation\bin目录下。文字编缉程序是可以生成ASCⅡ文件的任何编辑程序。

MDL开发工具	说　　明
任何文字编辑程序	生成MDL源码文件
mcomp.exe	编译目标文件并生成MDL目标文件
mlink.exe	链接目标文件并生成MDL资源 / 应用文件
mlib.exe	生成并管理MDL目标文件库
rcomp.exe	编译资源源码文件生成MDL资源文件
rdump.exe	用可读格式显示资源文件
rlib.exe	生成并管理MDL资源文件库
rsctype.exe	把C类型定义编入资源文件中

bmake是从制作文件(makefile)生成应用程序的实用程序。使用bmake可以简化程序开发过程。通常我们拷贝一个已有的制作文件，并做必要的改动。在本章末尾，我们再讨论bmake实用程序。而现在

我们需要了解开发流程的每一个步骤。

2.2.2 生成 MDL 应用

为了生成源程序plbox.mc，你可以使用文字编缉程序键入第一章提供的程序。

2.2.3 编译 MDL 应用

MDL编译程序是保证MDL源文件符合MDL规定的软件。如果源文件没有语法错误，MDL编译程序就生成一个具有“.mo”扩展名的中间文件。编译程序只能查找语法错误信息，而不能发现逻辑错误。

MDL编译程序命令行语法如下：

mcomp[-flag，-flag…] input-file

这里的input-file是我们的源码程序。如果不指定扩展名，则编译程序把“.mc”追加在文件名上。

—flag　是可选项。合法的flag是：

—c　只进行编译。用于告诉mcomp只进行编译但不自动链接。没有这个选项，UNIX工作站上，如果编译成功，将自动开始链接。在PC机上，不自动链接。

—dname=数值　用于以数值定义名字。这个名字出现在预处理#define语句中。

—g　把所有调试信息转储到输出文件。

—p　显示预处理输出。

—v　显示编译过程。

—ofilename　目标输出文件名。如不指定，则用扩展名“.mo加在输出文件名后。

—idir　为搜索包含文件增加额外目录。MDL编译程序可以支持40个包含文件目　　录。操作系统环境变量MDL_COMP对编译程序定义命令行变量。它用于指定在哪个目录下寻找包含文件。例如，DOS系统下的autoexec. bat文件中的以下行指定了寻找包含文件的两个目录：

Set MDL_COMP=-ic:\ustation\mdl\include -ic:\include

在UNIX工作站命令行为：

MDL_COMP = -i/usr/ip32/mstation/include

实际上我们不使用MDL_COMP作为寻找包含文件的路径。使用制作文件比较好，以后的章节将较多地涉及到制作文件。为了编译PLBOX程序，我们用以下命令行：

mcomp - c -v plbox

这行命令行将编译(-c)plbox. mc，并显示编译过程(-v)，并在当前目录中生成目标文件。

2.2.4 生成应用命令表

在上一章我们简要地学习资源文件和命令表。在这一节我们详细讨论MDL的这两个特性。每个命令表都必须含有两个包含文件<rscdefs.h>和<cmdclass.h>。

```
#include     <rscdefs.h>
#include     <cmdclass.h>
```

为了生成命令表，我们首先要建立一个称为Table的结构，这个结构将定义命令语法的层次。Table结构的语法如下：

Table <tabteid>=

{

<number>, <subtableid>, <commandclass>, <options>, <commandword>

}

tableid 是一个32位无符号整型数。主或根命令表必须赋给一个值为1的tableid。

number 定义命令号。

subtableid 指定命令中下一个词的子表。如果没有子表，则使用 #define CT_NONE。

commandclass 指定命令类别。命令类别是预先定义的，可以为下面所示的任一个：

INHERIT	0
PLACEMENT	1
VIEWING	2
FENCE	3
PARAMETERS	4
LOCKS	5
MACROCMD	6
MANIPULATION	7
SHOW	8
PLOT	9
NEWFILE	10
MEASURE	11
INPUT	12
CELLLIB	13
FILEDESIGN	14
COMPRESS	15
REFERENCE	16
DATABASE	17
DIMENSION	18
LOCATE	19
TUTCLASS	20
WORKINGSET	21
ELEMENTLIST	22
UNDO	23
SUBPROCESS	24
VIEWPARAM	25
VIEWIMMED	26
WINDOWMAN	27
DIALOGMAN	28
INPUTGEN	29

```
DIALOGOPEN      30
LEVEL           31
DEFAULTCMD      32
REPORTCMD       33
SELECTCMD       34
```

其它应注意的类别为INHERIT，在我们分析命令的类别时，它的意思是采用最后一个命令类别。共有63个允许的类别，前48个类留给MicroStation，后15个供给应用使用。现在回到例子PlBOX中，我们可以看出PLACEMENT对于我们的基本命令是最合适的命令类别。

options　可用的选项是NONE，DEF，REQ，TRY和CMDSTR(n)。这些选项可以用OR(|)相连。例如DEF|TRY(n)。

DEF　(缺省)，给特定表指定缺省值。

REQ　(要求)，告诉命令语法分析程序必须从子表中选择。

TRY　(试图分析)，这个选项将试图分析子表中的一个值。如果这个值不匹配，则把不匹配部分传递给应用程序。例如：ACTIVESTYLE 3，这里的3被传递给应用程序。

CMDSTR(n)　每次执行命令时将显示信息表(Messagelist)中的字符串n。

请看以下的资源文件plbox. r(含有命令表)。我们可以发现根命令词是PLACE，它属于命令类PLACEMENT。子表为CT__PLACE，这是强制性的，因为我们用REQ选页已做了规定。如果第二个字是BOX，CMDSTR将显示信息列表中的第一个字符串。

```
#include    “rscdefs.h”
#include    “cmdclass.h”
#define     CT_NONE     0
#define     CT_MAIN     1
#define     CT_PLACE    2

Table  CT MAIN=
{
{1,   CT_PLACE, PLACEMENT,     REQ,      “PIACE”),
};
Table CT PLACE=
{
{1, CT__NONE,  INHERIT,    CMDSTR(1),   ″BOX″},
};
```

2.2.5 信息表(Messagelist)

信息表(Messagelist)是用于在屏幕上显示信息的预定义资源类。之所以把信息从源程序中分离出来，是为了便于管理并增强可移植性。例如，我们可能希望移植应用，使之使用另外一种语言，使用信息表使这种改变非常容易做到，只需重新编译这个资源文件而不是整个应用。信息表结构的语法如下：

```
Messagelist <messageid> =
{
   {
```

```
    {<messagenumber>, "<message>"},
    ......
    }
  }
```

messageid　　识别这个表的唯一编号

messagenumber　　唯一的编号且必须以升序排列

message　　包含信息的字符串

messageid和messagenumber结合起来，可以唯一地识别列表中的信息。在plbox.r的信息表中，我们可以看到下述MDL命令将在提示域中显示字符串"Define start point"：

```
mdlOutput_rscPrintf (MSG_PROMPT, NULL, 0, 2);
```

PLBOX应用的信息列表如下：

```
Messagelist 0 =
{
  {
  {0, ""  ),
  {1, "MDL Place Box"},
  {2, "Define start point"},
  {3, "Define Box position"},
  {4, "PlBox complete"},
  }
}
```

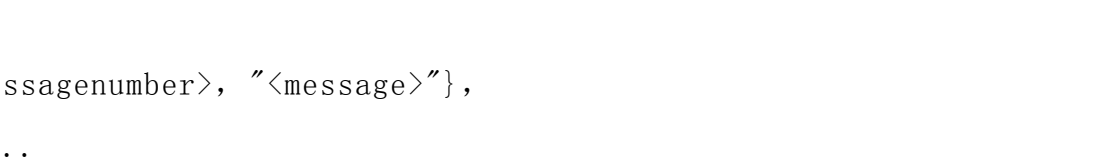


图2.3 在执行PLBOX时的提示显示

2.2.6 编译资源

资源源文件的命名约定用扩展名“.r”。编译过的输出文件的缺省扩展名为“.rsc”。

MDL编译程序的命令行语法如下：

```
rcomp[-flag, -flag…]input-file
```

这里input-file是我们的资源文件。如不指定扩展名，则编译文件在文件名后加“.r”

这里-flag是选项。合法标志为：

—dname=value　用值来定义名字。这个名字可以在预处理语句#define中找到。

—h　　当编译命令表时生成命令号头文件。

—hofilename　　指定头文件文件名。

—idir　　为搜寻包含文件增加额外的目录。

—ofilename　　目标输出编译文件名。如不指定则在输入文件的前缀后面加的扩展名“.rsc”为目标文件名。

—P　显示预处理器输出。

—V　显示编译过程。

我们将使用以下命令行编译资源文件plbos.r：

rcomp　-h　-v　plbox　　…

执行该命令行编译plbox.r，显示编译过程(-v)，并在当前目录中生成头文件(-h)plbox.h。生成的plbox.h内容如下：

```
#define CMD_PLACE           0x01000000     /*PLACEMENT*/
#define CMD_PLACE_BOX       0x01010000     /*PLACEMENT*/
```

我们已经在我们的源文件中见过CMD__PLACE__BOX。当我们在uSTN>提示符下输入PLACE BOX时，MicroStation生成一个命令编号，通过查询与应用程序有关的命令编号列表，来决定调用的函数，在我们的例题中这个函数是PlaceBox_start：

```
cmdName  PlaceBox_start()
cmdNumber CMD_PLACE_BOX
```

2.2.7 链接 MDL 应用

MDL 链接程序把目标文件结合到可以用作应用的程序文件中。或者，使用资源库管理程序，把它与其它资源文件连结。MDL 链接程序的命令行语法如下：

```
mlink [-flag, -flag, …] input-file
```

这里的 input-file 是我们的目标文件或库文件。如果不指定扩展名，则链接程序在文件名后增加“.mo”。

-flag　是选项，合法选项为：

-afilename　指定输出应用文件名。如果不指定，则把“.ma”加在第一个文件名的前缀后面。

-g 或-gd　把查错信息传送到输出文件。

-gn　　为有意义的信息提供充足的查错缓冲区。当—g 标志省略时，这是一个缺省值。

-go　　省略所有的查错信息。

-ml　　用按地址存贮的标记装入映象。

-mn　　用按名字存贮的标记装入映象。

-s　　[堆栈数]　指定堆栈数。如不指定，链接程序则分配一个。如果堆栈数太小，则当堆栈溢出时将引起 MDL 应用程序中断。如堆栈数太大，则浪费内存。

-t[taskID]　指定一个应用程序的任务 ID。在一个程序与其它程序合并时，每个程序都应有一个单一的任务 ID。

-v　　显示链接过程。

2.2.8 使用资源管理程序

资源管理程序把资源文件(“.rsc”文件)和程序文件(“.mp”)结合最终形成一个应用。开发 MDL 应用程序时可以不用任何资源文件，这时也不需要使用资源管理程序，开发流程在链接阶段也就停止了。

资源库管理程序命令行的语法如下：

```
rlib[-flag, -flag, …]input-file
```

这里 input-file 是资源文件或程序文件。

-flag　　是选项。合法的标志为：

-ofilename　　指定输出文件名。如果不指定，则在文件名后加扩展名“.rsc”。

-t<class>　　结合一个特定的资源文件类别。

-r<resouceID> 结合一个特定的 resourceID。

-v　　显示库管理程序工作过程。

在例子 PLBOX 中，资源库管理程序命令行如下：

```
rlib -od:\ustation\mdlapps\plbox.ma
d:\ustation\mdl\eamples\objects\plbox.rsc
d:\ustation\mdl\examples\objects\plbox.mp
```

2.3 bmake 实用程序

一个应用程序可以有几个包含文件、几个资源文件和几个源文件。如果我们修改其中一个文件，可能要重新编译其它文件。因为这些文件相互关联，所以要保证它们全部被更新。

bmake 是简化应用程序编译过程的程序维护实用程序。有一个 bmake(制作文件，它说明一个程序使用了哪些文件以及这些文件之间特定关系。bmake 是一个管理开发流程的强有力的实用程序，因此我们在本书中一直使用它。

bmake 根据比较两组文件的日期和时间来工作。这些文件就是目标文件和相关源文件。目标文件是从源文件中生成的文件。

2.3.1 制作文件(Makefile)的格式

制作文件是一个定义源文件和目标文件之间相关关系的文本文件。在我们运行 bmake 时，它要读所提供的制作文件。制作文件由一些块组成，这些块列出目标文件、源文件和建立目标文件的命令。

一个典型的块可能象下述一段编码

```
#-------------------------------------------
#Compile the MDL source file using mcomp
#-------------------------------------------
$(o)plbox.mo          : $(BaseDir)plbox.mc $(BaseDir)plbox.h
```

我们可以看到，目标文件是 plbox.mo，它依赖于源文件 plbox.mc。如果目标文件的日期比源文件早(就是说我们修改了源文件)，则建立命令将重新编译源文件。

在我们用 bmake 以前，我们必须设立 MS 环境变量，并查看文件\mdl\include\mdl. mki。为了设置 MS 环境变量，我们可以在 DOS 提示符下使用下面命令，或把命令加在 autoexec. bat 中。

```
SET  MS=D:\ustation\
```

我们可以在附录 A 中找到 mdl.mki 文件的清单。这个文件中的几个重要的行将在下面讨论。

2.3.2 bmake 宏命令

宏命令是对一个表达式或一组指令的缩写符号。bmake 宏命令是专用宏，当在制作文件中发现"$(macro)"时就执行它。(这里 macro 是宏命令名)。

一、预定义宏命令

bmake 支持的预定义宏命令列在下面。当 bmake 发现这些宏命令中的一个时，它将用适当的文件代替它。我们可以使用合适的操作系统宏命令,msdos、IP32、unix、macintosh 和 vax 是预先定义的。

$@　　用当前目标文件替换。

$?　　替换所有的比目标文件新的相关文件。

$=　　用最新的相关文件替换。

$<　替换当前相关文件。

$*　替换目标文件的基本文件名。

$%　替换第一个相关文件的目录。

二、宏命令的定义

用三种方式定义用户宏命令：

1．在制作文件中。

2．通过DOS环境变量。MS环境变量是这种方法的一个例子。

3．通过bmake命令行选项-d。

我们可以用下面的格式在制作文件中定义我们的宏命令：

macro=definition

例如：

msg=Building $@ from $=

这样我们就定义了显示信息"Buiding target file from source file"的宏命令msg。当bmake通过制作文件扫描并发现$(msg)时，则显示以上信息。

在制作文件中定义宏命令的第二种方法是用以下格式把一个定义和一个已有的宏命令连结起来：

macro+definition

三、定义宏命令路径

宏命令用于定义搜索路径。考虑mdl.mki中以下三个宏命令。由于d：\ustation代替宏命令$(MS)，这样我们就用d：\ustation\mdlapps定义了宏命令mdlapps。程序中凡是$(mdlapps)出现的地方，其路径都将被替换。

```
mdlapps         =$(MS)/mdlapps/
toolsPath       =$(MS)/mdl/bin/
src             =$(MS)/mdl/examples/
```

四、建立命令(Build Commands)

有些指令用于从相关文件建立目标文件。如果在建立过程中出现错误(也就是编译或链接错误)，bmake将中断。如果没有相关文件的建立命令，bmake则将使用预定义的规则，这是很重要的。下一节我们将看到bmake程序如何工作。

在一个建立命令中，有一些具有特殊意义的控制字符：

—　当这个命令返回一个错误时，不终断。它常用于表明一个变量。

@　没有回送。

|　回送该行。

>filename把制作文件中所有行重新指向filename，直到遇到<符号，用于建立一个链接文件。

-current　印出当前时间。

-time　用于使最新的相关文件的时间与目标文件相“联系”。它通常是建立命令的最后一行，用于保证文件被更新。

2.3.3 相关(Dependencies)

相关是文件间的一种关系。借助这种关系，对一个文件的修改将意味着重建目标文件。相关的格式是：

target1 target2…targetn：depl dep2…depn

[buildcommands]

例如：

$(objectDir)plbox.rsc:$(baseDir)plbox.r

如果一个dep文件比target文件新，则执行建立命令。如果没有指定建立命令，则使用一个预先定义推理规则。

2.3.4 推理规则

推理规则是定义如何从源文件用给出的扩展名建立目标文件的样板。一般格式为：

.[dir1；dir2；…；dir n]source.target：

例如，下面的推理规则告诉我们如何用一个“.mo”文件来建立“.ma”文件：

```
.mo.ma:
$(msg)
>$(objectDir)temp.cmd
—a$@
$(linkopts)$%$%*.mo
<
$(MLinkCmd)@$(objectDir)temp.cmd
～time$
```

要记住最重要的是源文件和目标文件都有扩展名。dir1；dir2；…；dirn是选项。如果没指定，则在当前目录中搜寻这个文件。宏命令$(linkOpts)和$(MLinkCmd)都在mdl.mki 中定义。

2.3.5 预定义的推理规则

文件mdl.mki提供了进行一般开发任务的预定义推理规则。

.mc.mo	编译".mc"源文件和生成目标文件".mo"
.mo.ma	链接目标文件".mo"及生成应用文件".ma"
.mt.r	读类型文件".m"广和产生资源源文件".r"
.r.rsc	编译资源源文件".rsc"
.r.h	建立命令包含文件".h"

2.3.6 条件编译

bmake 通过排除或包含制作文件的某些部分来支持制作文件中的条件编译。合法的条件为：

%incdude	包含文件好象是当前制作文件中的一部分
%ifdef	如果这个宏命令存在，包含随后的行
%ifndef	如果宏命令不存在，包含随后的行

%else %elif 与%ifdef 和%ifndef 一起使用

```
    %endif      关上一个%ifdef 块
```

例如：

```
    %ifdef msdos
    .c.obj:
    $(msg)
    $(CCompCmd)$%$*.c-ob$@
    ～time
    %endif
```

理解制作文件的最好途径是看 plbox.mke 文件。plbox.mke 的第一部分很明白。我们现在给我们的文件定义搜寻路径。

```
#-------------------------------------------------------------
#  PLBOX MDL Make File
#-------------------------------------------------------------
%if defined (_MakeFilePath)
BaseDir     = $(_MakeFilePath)
%else
BaseDir     = $(MS)/mdl/examples/plbox/
%endif

appName      = plbox
privateInc   = $(BaseDir)

#-------------------------------------------------------------------------
# mdl.mki contains the default rules for creating .rsc, .mo, etc files
#-------------------------------------------------------------------------
%include    $(MS)/mdl/include/mdl.mki

#-------------------------------------------------------------------------
# Define macros for files included in our link and resource merge
#-------------------------------------------------------------------------
lchObjs = \
     $(o)$(appName).mo

lchRscs = \
     $(o)$(appName).rsc \
     $(rscObjects)$(appName)msg.rsc \
     $(o)$(appName).mp
```

符号(#)告诉 bmake 不理睬该行的其余部分，把它当作注释处理。后划线(\)告诉 bmake 这一行与下一行相连。为了避免后划线和 DOS 目录路径的混淆，bmake 用前划线

(/)来指定路径。

下一步是定义相关关系。我们还没有给下面四行指定建立命令，所以 bmake 将使用在 mdl.mki 中找到的缺省推理规则规定建立命令。

```
#--------------------------------------------------------------------------
#        Generate command table include & resource file
#--------------------------------------------------------------------------
$(genSrc)$(appName).h            : $(BaseDir)$(appName).r

$(o)$(appName).rsc               : $(BaseDir)$(appName).r

#--------------------------------------------------------------------------
#        Compile the MDL source object file
#--------------------------------------------------------------------------
$(o)$(appName).mo                : $(BaseDir)$(appName).mc

#--------------------------------------------------------------------------
#        Builds any translatable resource modules for the application.
#--------------------------------------------------------------------------
$(rscObjects)$(appName)msg.rsc   : $(BaseDir)$(appName)msg.r
```

在写建立命令时，我们将在temp.cmd中放入链接命令的变量。

```
$(o)$(appName).mp      : $(lchObjs)
    $(msg)
    > $(o)temp.cmd
    -a$@
    -s6000
    $(linkOpts)
    $(lchObjs)
    <
    $(MLinkCmd) @$(o)temp.cmd
    ~time
```

当资源文件与程序文件连结时也要做同样的处理来生成应用文件。

```
$(mdlapps)$(appName).ma    : $(lchRscs)
        $(msg)
        > $(o)make.opt
        -o$@
        $(lchRscs)
        <
        $(RLibCmd) @$(o)make.opt
        ~time
```

2.3.7 执行 bmake

bmake实用程序的命令行语法如下：

bmake[-flag—flag……]makefile

这里的makefile是我们的相关描述文件，如果没有指定扩展名，则bmake把".mke"加在文件名后。

这里的-flag是选项，其内容如下：

-a　重建所有文件且不理睬相关关系。

-i　不理睬错误并在错误处继续bmake。

-l　用于在执行建立命令以前列出目标文件。

-n　显示建立命令，但不执行。

-m　即使遗漏相关文件也继续进行。

-p　按定义印出宏命令。

-s　沉默方式，在建立命令执行时不打印这些命令，

-t　不使用建立命令，但是把目标文件日期设为最新相关文件日期。

-w　停止警告及错误，通常bmake只遇到错误时才停止。

-dmacro 定义一个宏命令。这是定义宏命令三种方法中的第三种方法。

Bmake -a plbox

下面是它的输出内容：

```
Bentley Systems Make Utility. Version 08.05.02.27, Feb 28 2005
Fri Apr 07 12:03:58 2006
[==                    Building                    C:\Bentley\Program\MicroStation\mdl\objects\plbox.h,
(C:\Bentley\Program\MicroStation\mdl\examples\plBox\plbox.r) ==]
C:\Bentley\Program\MicroStation\mdl\bin\rcomp @C:\Bentley\Program\MicroStation\mdl\objects\make.opt
MicroStation Resource Compiler 08.05.02
   Generating header file (C:\Bentley\Program\MicroStation\mdl\objects\plbox.h) ... done.

[==                    Building                    C:\Bentley\Program\MicroStation\mdl\objects\plbox.rsc,
(C:\Bentley\Program\MicroStation\mdl\examples\plBox\plbox.r) ==]
C:\Bentley\Program\MicroStation\mdl\bin\rcomp @C:\Bentley\Program\MicroStation\mdl\objects\make.opt
MicroStation Resource Compiler 08.05.02

[==                    Building                    C:\Bentley\Program\MicroStation\mdl\objects\plbox.mo,
(C:\Bentley\Program\MicroStation\mdl\examples\plBox\plbox.mc) ==]
C:\Bentley\Program\MicroStation\mdl\bin\mcomp @C:\Bentley\Program\MicroStation\mdl\objects\make.opt
MicroStation Development Language Compiler 08.05.02

[==                    Building                    C:\Bentley\Program\MicroStation\mdl\rscobj\plboxmsg.rsc,
(C:\Bentley\Program\MicroStation\mdl\examples\plBox\plboxmsg.r) ==]
C:\Bentley\Program\MicroStation\mdl\bin\rcomp @C:\Bentley\Program\MicroStation\mdl\objects\make.opt
MicroStation Resource Compiler 08.05.02

[==                    Building                    C:\Bentley\Program\MicroStation\mdl\objects\plbox.mp,
(C:\Bentley\Program\MicroStation\mdl\objects\plbox.mo) ==]
C:\Bentley\Program\MicroStation\mdl\bin\mlink @C:\Bentley\Program\MicroStation\mdl\objects\temp.cmd
```

```
MicroStation Development Language Linker  08.05.02

[==                    Building                    C:\Bentley\Program\MicroStation\mdlapps\plbox.ma,
(C:\Bentley\Program\MicroStation\mdl\objects\plbox.mp) ==]
C:\Bentley\Program\MicroStation\mdl\bin\rlib @C:\Bentley\Program\MicroStation\mdl\objects\make.opt
MicroStation Resource Librarian 08.05.02
Fri Apr 07 12:03:58 2006, elapsed time: 0:00
```

我们可以生成一个seed.mke文件，以便我们用它生成将来的制作文件。为了生成新的制作文件，做一个seed.mke文件的拷贝。使用文字编辑程序做全面的改变，在所有出现"seed"的地方替换成新的应用名。

```
#--------------------------------------------------------------------
#       The MDL Make File
#--------------------------------------------------------------------
%if defined (_MakeFilePath)
BaseDir     = $(_MakeFilePath)
%else
BaseDir     = $(MS)/mdl/examples/plbox/
%endif

appName      = plbox
privateInc   = $(BaseDir)
#--------------------------------------------------------------------
# mdl.mki contains the default rules for creating .rsc, .mo, etc files
#--------------------------------------------------------------------
%include    $(MS)/mdl/include/mdl.mki

#--------------------------------------------------------------------
# Define macros for files included in our link and resource merge
#--------------------------------------------------------------------
lchObjs = \
     $(o)$(appName).mo

lchRscs = \
     $(o)$(appName).rsc \
     $(rscObjects)$(appName)msg.rsc \
     $(o)$(appName).mp

#--------------------------------------------------------------------
#        Generate command table include & resource file
#--------------------------------------------------------------------
$(genSrc)$(appName).h            : $(BaseDir)$(appName).r
```

```
$(o)$(appName).rsc                : $(BaseDir)$(appName).r

#--------------------------------------------------------------------------
#       Compile the MDL source object file
#--------------------------------------------------------------------------
$(o)$(appName).mo                 : $(BaseDir)$(appName).mc

#--------------------------------------------------------------------------
#       Builds any translatable resource modules for the application.
#--------------------------------------------------------------------------
$(rscObjects)$(appName)msg.rsc   : $(BaseDir)$(appName)msg.r
#--------------------------------------------------------------------------
#  The following section generates the MDL Program module using
#  mlink. This module should contain ALL CODE resources and/or
#  libraries used by the application.
#--------------------------------------------------------------------------
$(o)$(appName).mp      : $(lchObjs)
        $(msg)
        > $(o)temp.cmd
        -a$@
        -s6000
        $(linkOpts)
        $(lchObjs)
        <
        $(MLinkCmd) @$(o)temp.cmd
        ~time

#--------------------------------------------------------------------------
#   The final step in building the application is lib'ing the applications
#   intermediate application with the translatable resources built in
#   this makefile. This step generates the final, and possibly translated,
#   MDL application.
#--------------------------------------------------------------------------
$(mdlapps)$(appName).ma    : $(lchRscs)
    $(msg)
    > $(o)make.opt
    -o$@
    $(lchRscs)
    <
    $(RLibCmd) @$(o)make.opt
    ~time
```

2.4 执行 MDL 应用

为了执行MDL程序，我们必须从应用中把这个MDL程序装入MicroStation。一个应用可以包括一个以上程序。当一个程序被装入时，MDL将生成一个任务。当这个程序卸下时，这个任务被消除。

当MDL安装一个应用时，它将力图在MS__EXE及MS__MDL定义的目录内寻找有扩展名“.ma”，的文件。如找不到这个文件，则它在同样目录下搜寻有“. rsc”扩展名的文件。

如果你使用bmake 实用程序，会看到MDL 应用程序(.ma) 在目录 ustation\mdlapps下生成。这是由MS__MDL定义的缺省的MDL应用程序目录。

为了载入任何一个MDL应用程序，我们需进入MicroStation,并键入：

MDL LOAD <application name>

这里的<application name>是MDL应用程序。在我们的例子中，我们将键入下行：

MDL LOAD Plbox

我们可把上行缩写为：MDL L plbox。同时，我们也可以通过随后的MDL应用(MDL Application)对话框来安装MDL应用程序(参见图2.4)。MDL还可以用其它几种方法安装应用程序，第八章将涉及到这些方面的内容。

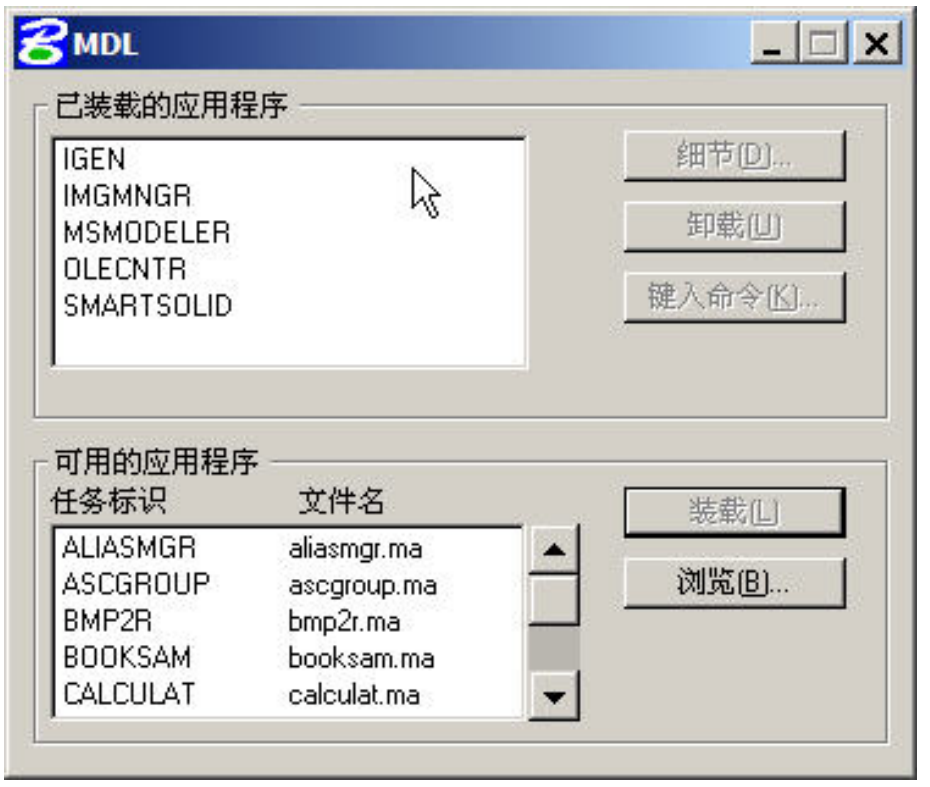


图2.4

PLBOX程序的执行开始于main函数，安装命令表并初始化变量，以便MicroStation理解我们的应用程序。

当我们写一个使用对话框的应用程序时，main 函数必须发布供对话框管理程序使用的数据结构。同时，需设立应用程序卸载时要执行的函数。对此下一章将详细介绍。

执行MDL应用程序

以上，我们只是载入了PLBOX，要执行其中的功能，须在uSTN>提示符下键入命令PLACE BOX。

MDL还提供了执行MDL功能的另一种方法:

MDL COMMAND FunName

只有在MDL代码中用关键字cmdName指定的功能函数，才能用这种办法来执行。在我们的例子中，这个函数是placeBox__start()。

2.5 调试程序

由于有调试程序，我们就有了开发MDL应用的有力工具。这就是本书要较早地介绍它的原因。

在传统的程序设计中，我们较容易按编码的自然顺序进行跟踪。这些程序进行调试，我们只需要设置一些适当的打印语句及出口。由于MDL语言的事件驱动特性，使这些发生变化。没有调试程序要跟踪一个MDL程序是非常困难的。这是因为我们不能确定我们在哪种状态下工作。

下面讨论调试命令。这些命令不分字符大小写。然而，传递给命令的变量名分大小写。

调试程序的STEP功能走遍我们程序并执行每一行。我们可以进入单行或多行并观察应用程序的运行。

Step/into　　使调试程序进入一个被调用函数，否则调试程序就在最顶层执行。

Step/over　　下一次从当前函数执行一个新的代码行时激活调试程序。

Step/from　　下一次在调用当前函数的函数内执行一个指令时激活调试程序。

Step/<count> 在控制返回调试程序以前，执行由count规定的次数。这在循环中非常有用，我们可以控制循环次数。

在任何一个阶段上，我们都可以键入变量名，调试程序会显示它的内容。DISPLAY命令可以显示/char，/short，/long，/double或/string等类型的数据，不考虑隐含的数据类型。对于数字数据类型(上面除了/string以外的所有类型)，输出格式可以指定为/HEX，/OCTAL或/DECIMAL。DISPLAY命令的变量必须是指针。

如果我们不能确定定义的是哪种变量，则SYMBOLS命令将显示所有已知的/functions，/variables和/structures。例如，“SYM/F/Vpla”命令将显示所有以“pla”开始的函数及变量。

我们可以设立一个断点(BREAKPOINT)，当搜寻到指定的行或函数名时，它将暂停程序的执行并对调试程序返回控制。例如，我们用“BR main 237”在main函数的第237行设立一个断点。我们可以用BR/DEL来删除断点，它将出现确认提示。我们也可以用命令/<count>以指定步长设立断点。这样，当调试程序搜寻达到这个步长时将产生中断。

用WATCH命令可以把中断功能用于变量。当一个变量被修改时，该命令可以暂停一个程序的执行。使用WATCH功能是有限制的，它只能用于MicroStation的PC版本。我们可以建立4个监视点，每个监视点占1个字节。

当一个应用程序的执行暂停时，我们可以用GO命令重新执行，一直到遇到另一个断点或程序结束。

其它调试功能包括CALLS命令，该命令显示到达当前函数前被调用函数的经历。/<COUNT>选项将只显示到达当前函数的函数调用次数。

MEMORY命令将显示应用使用的内存起始地址。TYPE命令允许我们列出从当前位置开始向前或向后的任何行。/<count>选项将印出指定的行数。

2.5.1 调试的设立

调试程序允许我们进入源程序的每一行，并观察它在我们的设计文件中的作用。这是一个非常有力的功能，但它需要我们有一个字符屏幕和一个同时激活的图形屏幕。

因为Intergraph工作站是一个多任务平台，所以它将在自己的窗口内写字符。对于使用单屏幕的PC系统，在需要向字符屏幕写时，MDL将生成一个DOS窗口。这个DOS窗口实际上就是一个图形屏幕，因此在写字符时较慢。同时,由于它占用了屏幕固定区域，因而减小了工作的空间。

一个最好的解决办法是再买一个监视器。例如，我们可以有一个VGA(或EGA)监视器，再有一个便宜的单色监视器。我们把MicroStation配置为使用一个VGA监视器。在进入MicroStation以前，我们在DOS提示符下键入MODE MONO。它将把屏幕输出开关拨向单色屏幕。在我们通过MicroStation输入一个设计文件时，我们可以看到图形在VGA屏幕上输出。这是因为DOS对于单色屏幕和VGA屏幕使用不同的内存空间。在这里，我们已经告诉DOS使用单色屏幕，MicroStation使用VGA屏幕。MicroStation并不检查DOS在做什么，它把图形写在内存的VGA部分内。因此我们不能把VGA和EGA屏幕结合起来使用，因为它们使用同一内存空间。

在我们离开MicroStation时，键入MODE C080可以使DOS返回彩色屏幕。

2.5.2 具有调试功能的编译

为了使调试信息进入我们的应用程序，我们需要告诉bmake在debug on状态下编译和链接程序。这可以通过在命令行加入—g选项来完成。例如：

```
bmake -g plbox
```

我们可以通过在调用mdl.mki以前定义调试来规定制作文件中的调试选项。

```
debug＝1
%include $(MS)/mdl/include/mdl.mki
```

如果我们不规定调试，我们则无法检查变量的内容。

2.6 使用调试程序

一旦我们成功地进入编译和链接的debug on状态，我们就可以准备对我们的应用程序进行调试。调试程序需要访问我们的源文件，因为当它执行时要显示每一行。我们可以通过建立环境变量MS__DBGSOURCE指定的路径来定义源文件的位置。

例如，如果我们的源文件在c:\progmdl\disk\plbox\，我们就在uconfig.dat内写上这样一行：

```
MS__DBGSOURCE＝C:\progmdl\disk\plbox\
```

最有效的途径是使我们的源文件目录为当前目录，然后调用MicroStation。这样我们不用顾虑环境变量，特别是在对一个以上应用程序进行工作时。在mdb>提示符下，我们可以用!(叹号)进入操作系统。然后我们可以把目录改在源文件驻留的位置。键入exit，可以返回调试程序。

2.6.1 安装调试程序

假定象在前面讨论过的一样，我们有了两个监视器，在DOS提示符下键入MODE MONO。我们现在打开了单色屏幕的开关，然后通过MicroStation进入图形环境。在uSTN>提示符下，我们用以下命令装入我们的应用程序：

```
MDL LOAD DEBUG <appllcaeion name>
```

这里的application name是一个MDL应用。在我们的例子中，我们将键入下面内容

MDL LOAD DEBUG plbox

我们也可以缩写为：

MDL L D plbox

我们将在单色屏幕上得到mdb>提示符。我们可以通过键入STEP进入这个程序。

调试程序会显示源文件要执行的行。因此，我们按return键可以观察这一行的作用。调试程序记忆最后一个命令(在这里是STEP)。回车键将使之响应并执行该命令。同时，我们可以用向上的箭头键调出最后一个命令，然后按回车键。

在调试程序的每一阶段上，我们都可以键入变量名来观察它的内容。不要落入查看源码变量名，然后查询其内容的陷阱。由于这行尚未执行，这是不正确的。要记住调试程序只是显示这一行。只有我们走到下一行时，我们才能查询调试程序执行的上一行。

在我们进入main函数后，我们把控制返回到MicroStation。如果我们现在在nSTN>提示符下键入PLACE BOX，将执行这个MDL程序，同时我们回到调试程序。现在我们在函数placebox__start内。为了解释调试程序的作用及事件驱动的概念，我们增加下面语句：

```
mdlOutput_message ("Selectposltion"),
```

我们的函数现在如下：

```
/*-----------------------------------------------------------------+
 |  name        placeBox_start (modified)                          |
 +-----------------------------------------------------------------*/
Public void  placeBox__start()
cmdNumber  CMD_PLACE_BOX
{
mdlState_startPrimitive (placebox_firstPoint, placebox_start, 2, 2);
mdlOutput_message ("Select position");
}
```

进入程序后，要注意mdlState_startPrimitive建立了一个控制事件的函数(在这里是一个数据点)，下一步将在信息域显示信息"Select position"。这不同于传统的从上至下的编程技巧。我们一般希望一个函数有一个入口和一个出口点。事件驱动的概念意思是状态函数建立了一个“中间体”。

许多程序员很难理解MicroStation事件驱动的本质。当我们使用任何一个mdlState__函数时，我们必须建立处理事件的“控制(haudles)”。我们使用的状态函数越多，我们可以控制的事件也越多，因此我们的MDL程序也有多个入口点。这样更强调了调试程序的重要性。没有调试程序，将不可能跟踪一个事件驱动软件的过程。

2.6.2 显示变量内容

当我们显示一个变量的内容时，我们必须确定我们要求调试程序显示的内容。调试程序有一个内部C表达式解释程序。它将鉴定表达式并按我们的要求赋值。例如，我们正在通过一个程序进行循环并输入：

mdb>i=5

来改变循环次数，i将得到一个新值5，并且用新值继续循环。

当我们显示一个变量的内容并得到如ox126678信息时，我们立即知道显示的是变量的地址而不是

其内容。

若我们要显示一个字符串的内容，则把变量名键入调试程序将使每行显示一个字符。我们可以按CTRL-C键停止显示，并返回mdb>提示符。

2.7 错误信息

一、语法错误

程序错误分为语法或语义错误和运行期间错误。

编译程序捕捉语法错误而不是运行期间错误。MDL语法以C为基础，因此我们可以 对照C设计手册来检查语法错误。

二、运行期间错误

运行期间错误通常由逻辑错误引起。这就是调试程序可以起作用的地方。通过进入程序，我们可以检查程序所做的是否是我们所期望的。

三、指针错误

甚至当我们的逻辑正确时也可以产生指针错误。MDL广泛地使用指针，不正确的用法经常使机器锁死。恢复控制的唯一方法只能是重新启动机器。如果是UNIX工作站，操作系统将对所有没关闭的文件做全面的检查。在DOS系统中，我们需要运行CHKDSK/F。该命令将在根目录下生成带有扩展名“.chk”的未关闭文件。我们可以把这些文件删去。

当我们得到一个类似bad memory access(错误内存访问)的信息时，我们知道我们传递了一个数值，而不是一个地址。

2.8 第二章小结

在这一章，我们看到了一个完整的MDL应用开发流程。我们介绍了许多适用于MDL的支持工具。我们建议你学习怎样使用bmake和调试程序。任何正规软件的开发都需要广泛使用这些工具。在进一步使用MDL以前，掌握这些工具是很重要的。

在开始的两章我们讨论了MDL程序设计的基础。如果你对MDL的C语言方面还有问题，可以停下来先参阅一本C语言程序设计的书。在本书其余部分，我们将假设你是一个胜任的C语言程序员。

第三章　MDL 对话框

在这一章我们要在前面章节中的PLACE BOX程序内增加一个对话框界面。我们要调用应用PBDLG，运行结果如图3.1所示。

图3.1 PBDLG对话框

3.1 开发流程

使用对话框应用的典型开发流程(如图3.2所示)是非常复杂的。它强调bmake实用程序的重要性，以及它的维护、更新与应用有关的所有文件的功能。我们的任务是生成图3.2中画有阴影线的那些框内的文件。

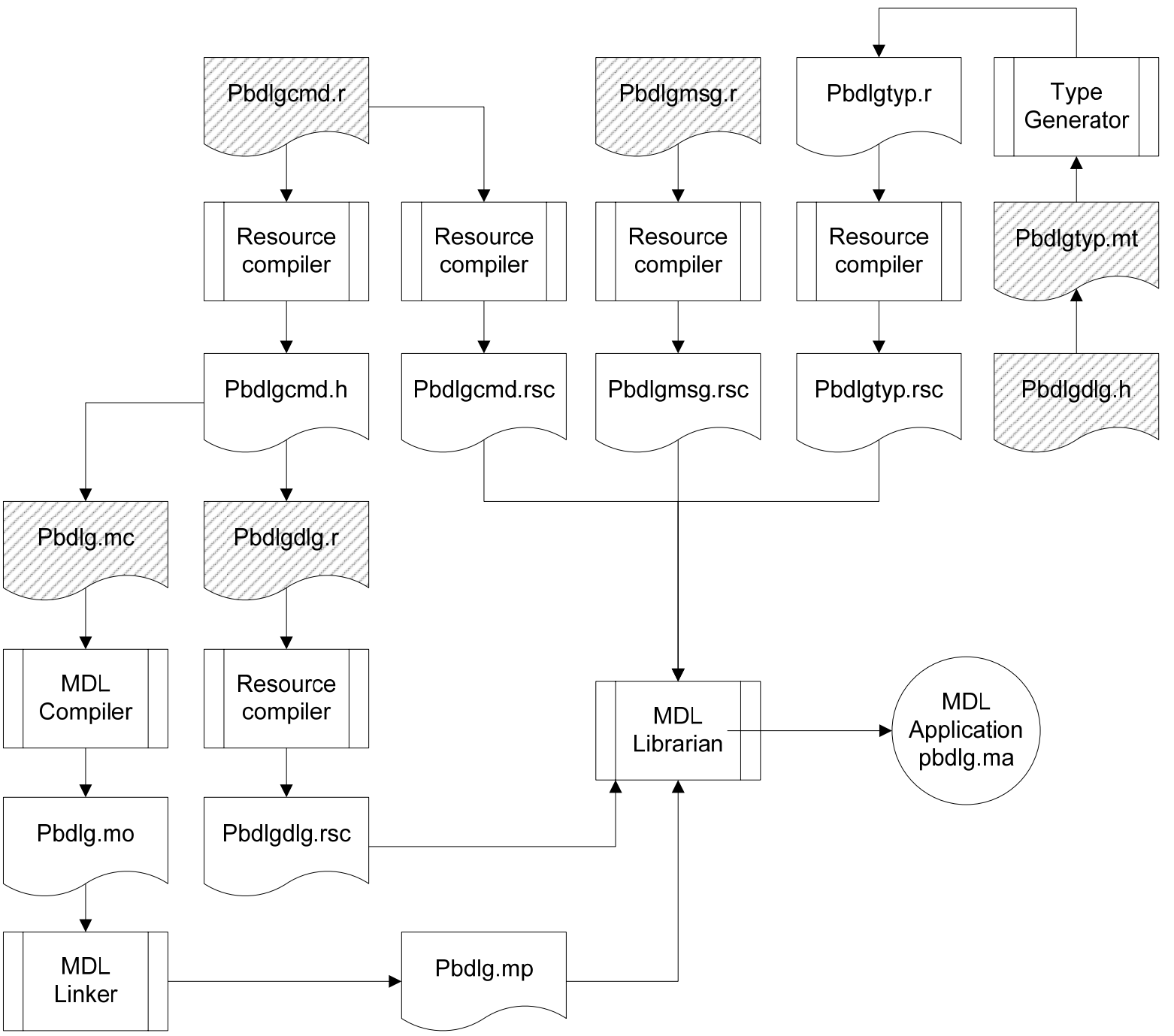


图3.2 PBDLG开发阶段

用于PBDLG的文件是：

pbdlgcmd.r	命令句法
pbdlgmsg.r	屏幕信息
pbdlgdlg.r	对话框资源源文件
pbdlgdlg.h	含有对话框类型的包含文件
pbdlgtyp.mt	要发布的数据结构
pbdlg.mc	MDL源文件
pbdlg.mke	这个应用的制作文件

我们把资源文件分成对话框、信息、命令句法及发布的数据结构几部分。在pbdlg.mke中将用推理规则控制资源文件的编译按正确的顺序进行。

注意：*不要做重复工作。在开始一个新的应用时，先拷贝一个已有程序，并进行适当修改。这种方式可以缩短开发过程。我们将使用拆开的PBDLG文件做为我们的SEEDDLG的种子文件。*

对于一个应用，生成对话窗口的第一步是应决定在对话框中要做什么。从这里我们可以生成一个数据结构。数据结构要与对话框相匹配。一旦我们把对话框数据结构分好类，我们就做完了一半工作。

我们将生成含有单元名、对话框内使用的字符串、组合方式(如何组合元素)以及是否有引线的数据结构。这个结构如下：

```
typedef struct placeboxinfo
{
char     cellName[8];
char     String[127];
int     groupMode;
int     leaderLine;
}
placeBoxInfo;
```

3.2 对话管理程序

对话管理程序对屏幕上可重定尺寸的MicroStation窗口的操作进行控制。有两类窗口，即视图窗口及对话框窗口。视图窗口用于显示及改变MicroStation设计文件。“高级对话窗口中”一章将介绍视图窗口。本章我们要讨论设计并生成一个对话框窗口。一个对话框包括对话框条目，这些条目是用户界面控件。

3.2.1 设计对话框

在对话框中，可以有如下对话框条目：

ColorPicker	从一个有255种颜色的调色板中选色
Generic	可用户化的条目，无缺省内容
GroupBox	围绕一组对话条目画一个立体外观的矩形
Label	在对话框中写一个字符串
LevelMap	选择或改变文件层的状态
ListBox	允许显示多个字符串
MenuBar	用于生成下拉菜单的滚动条
OptionButton	从一组选择中选出一项
OptionPulldown	带有选项列表的下拉菜单，可以是字符或图符
PushButton	按一个按钮激活一个命令
TextPulldown	包含一系列字符串的菜单
ScrollBar	在最大及最小值之间改变变量值的一个滚动条
Text	接收一个输入字符串
ToggleButton	使一个变量状态打开或关闭

在例子中，我们将用Text，OptionButton，ToggleButton和PushButton。对话窗口的Text部分提示PLBOX的单元名。我们将用字符串"PLBOX"启动这个框并允许用户改变它。对于组合方式我们将使用OptionButton，不管选择的是一个单元、图形组或者什么也没有。ToggleButton是使一个选项打开或关闭的最好方式。在我们的例子中，要用ToggleButton选择所划的方框有无引线。选按钮"OK"将启动程序。

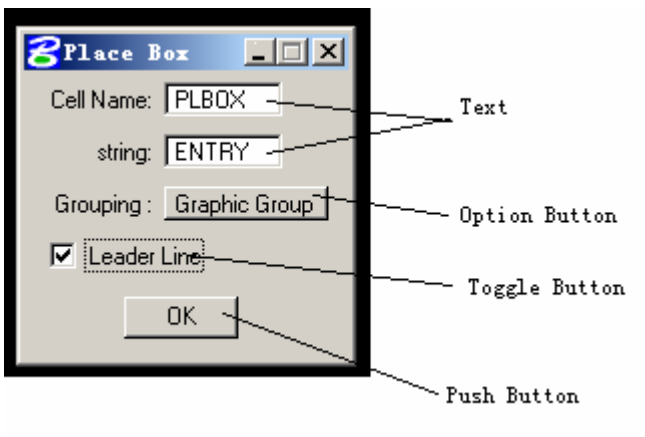
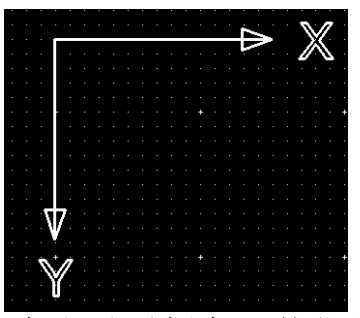


图3.3 PBDLG中使用的对话框条目

对话框分辨率

我们使用局部座标定义对话框。对话框的原点在左上角，Y的正方向向下，X正方向向右。对话框中的座标单位是象素或对话框座标。象素单位是实际的屏幕象素。对话框座标定义为对话框的当前字符高度的1／12。



我们将用对话框座标来确定对话框信息和对话框条目的位置。这将保证使用不同的字体或对话框从一个屏幕移到另一个屏幕时的一致性。

在包含文件<dlogbox.h>中有三个常量影响到我们所讨论的各点：

```
#define DCOORD_RESOLUTION 12
#define XC (DCOORD_RESOLUTION / 2)
#define YC DCOORD_RESOLUTON
```

常量DCOORD_RESOLUTION被规定为12个象素。XC是一个字符的宽度，是字符高的一半。YC是一个字符的高。为了定义对话框的尺寸，我们要把下列常量放在我们的对话框资源文件中。

```
#define OVERALLWIDTH 25 * XC
#define OVERALLHEIGHT 11 * YC
```

简单地说，我们的对话框将为25个字符宽，11个字符高。在<dlogbox.h>中有一个帮助构筑对话框的宏命令。这个宏命令用行数计算竖直位置。使用这个宏命令将确定对话条目的适当位置，并使它们之间有适当间隔。该宏命令如下：

```
#define GAP 3
#define GENY(row)  ((row-1)*(YC+GAP)+YC / 2)
```

下面常量定义了对话框中信息的位置：

```
#define OVERALLWIDTH           25 * XC
#define OVERALLHEIGHT          11* YC
#define NEWLINE                2 * YC

#define X1 11 * XC             /*cell Name*/
#define X2 11* XC              /*Leader Line */
#define X3 2 * XC              /*guoup Mode */
#define X4 (OVERALLWIDTH/2) - (5 * XC) /* middle of OKAY button */

#define Y1 GENY (1)            /* cell Name */
#define Y2 Y1 + NEWLINE        /*input string */
#define Y3 Y2 + NEWLINE        /* guoup Mode */
#define Y4 Y3 + NEWLINE        /* leader Line */
#define Y5 Y4 + NEWLINE     /*okay button */
#define BW XC * 9              /* box width */
```

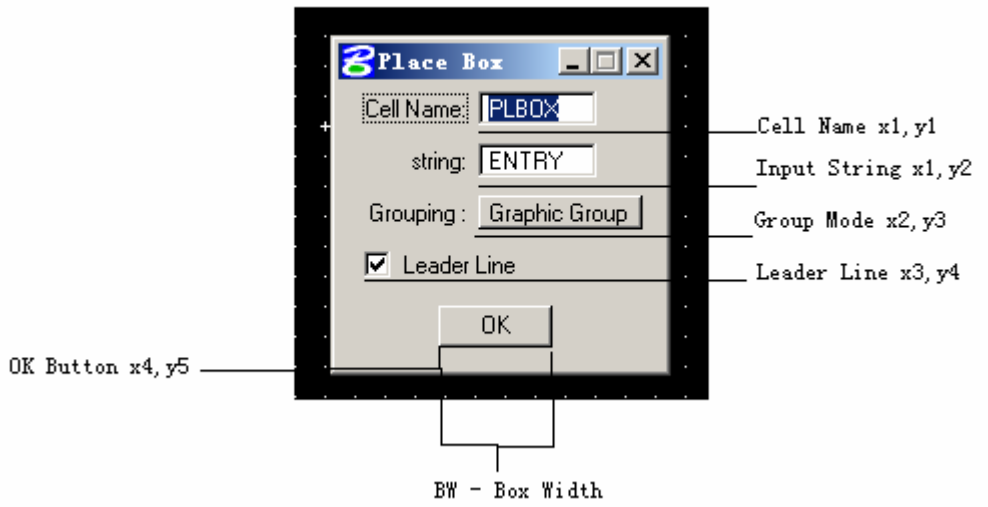


图3．4对话框条目的位置

图3．4显示出PBDLG对话框条目的布置。注意文本条目(Text)和选项按钮条目的座标定义条目的位置，而不是字符标注的开始位置。

注意我们如何把2＊YC(NEWLINE)加在GENY宏命令上。这是因为要保证不让这些条目彼此太靠近。若我们用GENY按如下所示的语句定义这些行：

```
#define Y1 GENY(1)    /*cell Name*/
#define Y2 GENY(2)    /*inputstring*/
#define Y3 GENY(3)    /*group Mode*/
#define Y4 GENY(4)    /*leader Line*/
#define Y5 GENY(5)    /*okay button*/
```

结果产生的对话框中的各行条目靠得太紧。

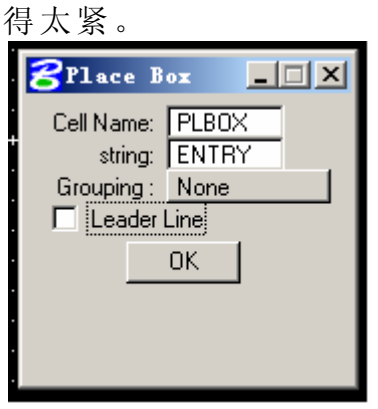


图3.5 宏命令GENY的使用

3.3 生成对话框资源

对话管理程序使用资源文件从应用程序中分离出用户定义对话框。因此，对话框的任何改变只需修改对话资源，而不是修改应用本身。

用结构DialogboxRsc定义对话框：

```
typedef struct dialogboxrsc
{
Ulong attributes;      /*dialog attributes */
int     width;            /* dialog coords */
int     height;         /*dialog coords  */
ULong helpInfo;         /* help for dialog */
ULong helpsource;       /*help task ID */
long  dialogHookId;     /* dialog hook ID */
long parentDialogId;    /* inform when destroy */
#if defined (resource)
 char  label[];         /* dialong title */
 DialogItenRsc itemList[];   /* array of dialog items */
#else
 long labelLength;          /*dialog title string lengt */
 char  label[1];         /*dialog title */
#endif
}DialogBoxRsc;
```

attributes(属性)规定对话框的特性。这个字段使用常量。我们将使用缺省属性DIALOGATTR_DEFAULT。允许使用的属性是：

DIALOGATTR_DEFAULT	缺省，未设置属性
DIALOGATTR_MODAL	指定模态对话框，详见第七章
DIALOGATTR_GROWABLE	对话框可重定尺寸
DIAIOGATTR_SINKABlE	对话框可送到视图窗口后面
DIALOGATTR_UNCLOSEABLE	不能关闭对话窗口
DIALOGATTR_NOAUTOSWITCH	键盘焦聚不能自动打开另一个窗口的开关
DIALOGATTR_CLOSEONNEW	当打开一个新设计文件时关闭对话框
DIALOGATTR_ALWAYSSETSTATE	当用户与条目交互时设置条目的状态

width和height定义我们对话框的尺寸。我们将使用常量OVERALLWIDTH和 OVERALLHEIGHT。

helpinfo和helpSource是用户为这个应用定义帮助功能的。由于不存在帮助，所以它们分别设置为NOHELP和MHELPo

本章后边将讨论对话框钩子。因为现在我们不使用对话框钩子，所以对于这个字段我们使用常量NOHOOK。

在设计我们的对话框中，我们可以用一个对话框调用另一个对话框。parentDialogId规定父对话框，用于说明什么时候消除子对话框。在我们的例子中，我们没有父对话，因此我们给这个字段赋值为NOPARENTID。

label含有对话框菜单条中显示的字符串。对话框条目itemlist在下一节讨论。我们的资源DIALOGID_PlaceBox如下所示：

```
DialogBoxRsc DIALOGID__PlaceBox =
{
DIALOGATTR__DEFAULT,
OVERALLWIDTH, OVERALLHEIGHT,
NOHELP, MHELP, NOHOOK, NOPARENTID,
"PIace BOX",
 .
 .
 .
```

3.3.1 对话框条目

反过来看本章开头，我们看到我们对话框中有一些条目，命名为Text、Option、Button、PushButton和ToggleButton。对话框管理程序把对话框条目分为对话框条目资源和对话框条目列表说明。对话框条目列表用结构dialogitemrsc定义。定义这些条目的顺序是很重要的，因为这也是对话框管理程序显示及提示的顺序。对话条目资源结构如下：

```
typedef struct dialogitemrsc
{
Sextent extent;    /*sensitive area, origin (in dialog coords), if width*/
                   /*height is zero use dimensions specified in item*/
long  type;        /* item type*/
long  id;
byte  attributes;
long  itemArg;
#if defined(resource)
 char  label[];
 char  auxInfo[];
#else
 long  labelLength;
 char  label[1];
#endif
} DialogItemRsc;
```

Sextent的定义如下：

```
typedef struct spoint2d
{
short    x;
short    y;
}Spoint2d, SPoint2d;

typedef struct extent
{
```

```
SPoint2d    origin;      /* upper left */
short    width;
short    height;
}Sextent;
```

范围字段定义对话框中对话框条目将要复盖的区域。前两个范围字段指定从对话框的原点(左上角)算起的条目的起点。下面两个范围字段指定条目的宽度和高度。如果我们对宽度和高度设定为0，则使用缺省值。在下面例子中，第一个对话条目从向右11个字符 及向下1行处开始。该条目9个字符长。

```
{/* dialog item list*/
{{x1, Y1, BW, 0},   Text, TEXTID cellName, ON,   0,   "", ""},
{{x1, Y2, BW, 0},   Text, TEXTID string, ON, 0, "", ""},
{{X2, Y3, 0, 0},   optionButton, OPTIONBUTTONID groupMOde, ON, 0, "", ""},
{{X3, Y4, 0, 0},   ToggleButton, TOGGLEID leaderLine, ON, 0, "", ""},
{{x4, Y5, BW, 0},   PushButton, PUSHBUTTONID ok, ON,  0,   "",  ""}
}
```

3.3.2 通用条目资源字段

对于不同对话条目有些域是相同的，这些字段依次为：

commandNumber，含有放在MicroStation输入队列末尾的命令编号。在执行MicroStation命令时，这是很有用的。例如，我们可以设计一个设置激活层的对话框。要使用的命令编号为CMD_ACTIVE_LEVEL。至于完整的命令编号表请参考包含文件<cmdlist.h>。若我们不想传送一个命令，则我们使用NOCMD。

commandSource，规定执行commandNumber的源程序。用MCMD指定MicroStation将执行这个命令；否则，用LCMD来指出拥有对话的任务应该执行该命令。如果我们已经指定为NOCMD，则在这个字段使用LCMD。

unparsed，用commandNumber把字符串传递到输入队列。如果我们不想用commandNumber放置字符串，则用双引号“”指定为空串。

HelpInfo，指定帮助ID，如不存在帮助，则在这个字段中使用NOHELP。

HelpSource，指定管理帮助信息的任务。若无可使用的帮助信息，则在这个字段中使用MHELP。

itemHookId，定义钩函数的ID。如果不用钩子函数，则在这个字段中指定为NOHOOK。钩子函数允许我们修改对话框和条目的缺省动作。“高级对话框”一章将详细讨论钩子函数。

itemHookArg，为钩子函数规定变量。如不需要变量，则在这个字段中使用NOARG。

label，是将在对话框中显示的用户提供的字符串。

accessStr，对话框条目控制的变量。例如，我们想使对话框中的任何变化都反映在变量plBoxInto—>String中，所以，accessStr将是"plBoxlnfo—>String"。对话框能识别这个变量，因为它已被说明。我们在讨论MDL源程序pbdlg.mc时会得到这方面更多的资料。

SynonymsId，定义在对话条目被修改时与所做的修改相对应的对话条目。最好的例子是ColorPicker对话条目，它有SynonysmsId，如Text Item。在ColorPicker改变时，Text Item影响相应的颜色编号。在“高级对话框”一章中有一个例子。

3.3.3 文本对话条目

文本对话框条目用于提示用户进行字符输入。在我们的例子中提示单元的名字及放置在设计文件中的字符串。以下是文本对话框条目的条目说明列表：

```
{{X1, Y1, BW, 0}, Text, TEXTID_cellName, ON, 0, "", ""};
{{X1, Y2, BW, 0}, Text, TEXTlD_String, ON, 0, "",""},
```

文本资源对话框条目的结构用ditem_textrsc定义。

```
typedef struct ditem_textrsc
{
ULong CommandNumber;
ULong commandSource;
long  synonymsId;
ULong helpInfo;
ULong  helpSource;
long  itemHookId;
long  itemHookArg;
byte  maxSize;                  /*max # of chars in field */
```

```
  char formatToDisplay[16];      /* format str to comvert from internal */
  char formatToInternal[16];     /* convert to internal from display str */
  char    minimum[16];         /*minimum value  */
  char    maximum[16];         /*maximum value */
  ULong   mask;              /*Only Used with integer types */
  UShort attributes;         /* other attributes */
$if defined (resource)
  char  label[];
  char  accessStr[];
#else
  long  labelLength;
  char  label[1];
#endif
}DItem_TextRsc;
```

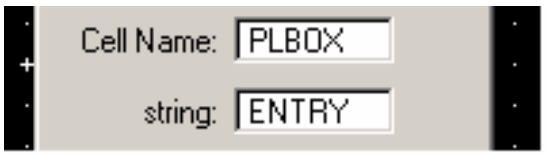


图3.6 PBDLG的文本条目

在我们的对话框资源文件pbdgdlg.r中定义的文本条目资源如下：

```
DItem_TextRsc TEXTID_cellName =
{
NOCMD, LCMD, NOSYNONYM,  NOHELP, MHELP, NOHOOKM, NOARG,
8, "%S",  "%S' , "", "", NOMASK, NOCONCAT,
"Cell Name: ",
"plBoxInfo->cellName"
};

DItem_TextRsc TEXTID_String=
{
NOCMD,LCMD, NOSYNONYM,NOHELP, MHELP, NOHOOK, NOARG,
127,  "%S",  "%S", "", "", NOMASK,NOCONCAT,
"string: ",
"plBoxlnfo->String"
};
```

大多数字段都在前面的“命令条目资源字段”一节中讨论过。maxSize规定这个文本条目可编辑的最大字符数。注意虽然在对话框中只显示了9个字符，但是TEXTID_String中的maxSize是127。由于文本条目允许水平移动，所以这是可能的。它可以编辑长度超过允许显示范围的字符串。

1. 键盘聚焦

当前正在处理用户键入的对话条目称为聚焦，即当一个对话准备接收键入信息时，它变成黑色(如图3.7所示)。文本对话条目及滚动条是可聚焦对话条目的两个例子。不是所有的对话条目都可以聚焦。这时MicroStation将使用键盘焦点自动开关，把焦点移动到命令窗口。

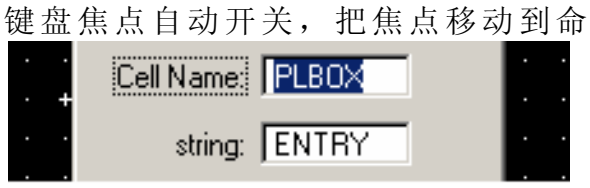


图3.7 聚焦的单元名对话框

3.3.4 选项按钮条目

选项按钮允许用户从一组选择中选择一个选项。当我们选择按钮时，将出现一个选项列表。拖动光标使选项变亮，变亮的条目即被选中。

```
typedef Struct optionbutton_iteminfo
{
int     subItemIndex;
char   *labelP;
ULong   *iconTypeP;
long   *iconldP;
int     *commandsourceP;
ULong   *valueP;
ULong   *maskP;
int     *enabledP;
```

```
void  *userDataP;
} OptionButton_ItemInfo;
```

图3.8 使用选项按钮

在我们的对话资源文件pbdgdlg.r中，我们定义的选项按钮条目资源如下：

```
DItem_OptionButtonRsc OPTIONBUTID_groupMode
{
NOSYNONYM, NOHELP, MHELP, NOHOOK,NOARG,
"Grouping : ",
"plBoxInfo->groupmode",
{
{NOTYPE, NOICON, NOCMD, LCMD, 0, NOMASK, ON, "None"},
{NOTYPE, NOICON, NOCMD, LCMD, 1, NOMASK, ON, "cell"},
{NOTYPE, NOICON, NOCMD, LCMD, 2, NOMASK, ON, "Graphic Group" },
}
};
```

3.3.5 按钮条目

按钮条目用于激活一个命令。它经常用于起动另一个对话框，或者认可做的操作。在我们的程序中，我们使用按钮条目执行PLACEBOX程序。在我们的程序中还提供了用户钩子函数。这允许对话框在所有时间内出现，但我们可以去访问其它命令。当用户按OK按钮时，用户钩子函数将激活PLACE BOX程序。按钮资源条目由ditem_pushbuttonrsc定义。

```
typedef struct ditem_pushbuttonrsc
{
char  isDefault;      /*TRUE if this iS default button*/
uLong  helpInfo;
ULong  helpSource;
long  itemHookId;
long  itemHookArg;
ULong  CammandNumber;
ULong  cammandSource;
#if defined(resource)
 charUrparsed[];
 char  label[];
#else
 long  unparsedLength;
 char  unparsed[1];
endif
}DItem PushButtonRsc;
```

若isDefault被设置为TRUE，则规定这是个缺省按钮。因此，敲回车键与用鼠标按绎的作用是相同的。我们应用中的按钮条目如下：

```
DItem_PushButtonRsc PUSHBUTTONID_Ok=
{
NOT_DEFAULT_BUTTON. NOHELP, MHELP,
HOOKITEMID_Button_Place,0,NOCMD, MCMD, ''''.
" OK"
}
```

3.3.6 开关按钮(ToggleButton)条目

开关按钮的动作和外观象一个开关。它只有两个状态，关或开(1或0)。开关按钮的一个特点就

是它具有直接改变变量的能力，而不需要过多的编程。图3．9显示了开关对话打开状态。如下所示，开关按钮由结构ditem_togglebuttonrsc定义。其中大多数字前面已讨论过了。

```
typedef struct ditem_togglebuttonrsc
{
ulong    commandNumber;
ULong    commandSource;
long     synonymsId;
unong    helpInfo;
ULong    helpSource;
long     itemHookld;
long     itemHookArg;
ULong    mask;
char     invartFlag;
#if defined(resource)
 char        label[];
 char        accessStr[];
#else
 long        labelLength;
 char        label[1];
#endif
)DItem_ToggleButtonRsc;
```

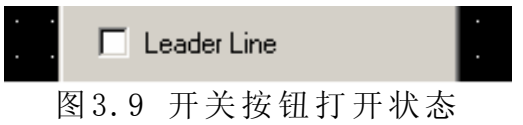


图3.9 开关按钮打开状态

mask和invertFlag与accessStr一起使用来改变accessStr的值，这使它具有强大的功能。

例如，元素的搜寻基于一个搜寻掩码。如果我们为每一个元素定义了一个开关按钮，按下这个按钮则表示查找这个元素的位置。我们可以把invertFlag设置为NOINVRT，并提供一个元素类型掩码。例如，一条类型为3的线元素，开关按钮将如下所示：

```
DItem_ToggleButtonRscTOGGLEID_Line=
{
NOCMD, MCMD, NOSYNONYM, NOHELP, MCMD, NOHOOK, NOARG.
Ox4, NOINVERT, ”Line”, ”typmask[01”
};
```

在上例中，typmask[0]要与0x4进行”或“运算，因此第三位设为ON。这个功能不需要任何编程，却提供了一个大型程序的强大功能。

```
Dltem_ToggleButtonRsc TOGGLEID_leaderLine=
{
NOCMD, MCMD, NOSYNONYM, NOHELP. MCMD. NOHOOK. NOARG,
NOMASK, NOINVERT,
”Leader Line”,
”plBoxlnfo->leaderLine”
};
```

3.4 对话框通用函数

在我们进入pbdlg.mc程序以前，我们必须看一看通用话函数，这些函数基本上在对话框中操作。

mdlDialog_closeCommandQueue	使关闭对话框的命令排队
mdlDialog_cmdNumberQueue	使一个命令加入输入队列
mdlDialog_find	返回资源ID指定的对话框
mdlDialog_hookDialogSendUserMsg	把一个用户信息传给对话钩函数
mdlDialog_hookItemSendUserMsg	把一个用户信息传给条目钩函数
mdlDialog_hookPublish	用钩函数地址发布钩函数ID编号
mdlDialog_lastActionTypeSet	用于指出对话框关闭的原因
mdlDialog_open	打开一个非模式化的对话框，即允许与其它对　话框对话的对话框
mdlDialog_openAlert	打开一个报警对话框
mdlDialog_openModal	打开一个模态对话框.这个对话框将迫使所有操作在模态对话框内进行
mdlDialog_openPalette	打开一个非模式化的对话框,它包含一个单图符命令调色板条

	目
mdlDialog_parentIdGet	获取一个对话框的父ID
mdlDialog_parentIdSet	设置一个对话框的父ID
mdlDialog_publishBasicPtr	发布一个C的基本数据类型的指针
mdlDialog_publishBasicVariable	发布一个C表达式串中使用的C的基本数据类型变量
mdldialog_publishComplexPtr	发布一个指向结构类型变量的指针
mdlDialog_publishComplexVariable	发布一个结构类型变量
mdlDialog_publishStructure	发布一个结构说明
mdlDialog_synonymsSynch	迫使所有条目的外观与它们外部状态相匹配
mdlDialog_userDataPtrGet	获取一个与对话框有关的用户指针
mdlDialog_userDataPtrSet	设置一个与对话框有关的用户指针

3.4.1 对话钩函数

对话框管理程序有三个管理对话框的子系统，就是对话框函数，条目管理函数和用户钩子函数。子系统间通过相互传送信息数据结构来进行通信。对话框条目管理器定义缺省函数性和对话框条目的外观。

由于所有对话框条目都具有缺省值，所以经常需要修改缺省动作。对话框钩子函数允许程序员把用户函数与对话框(称为对话框钩子函数)或对话框条目(称为条目钩子函数)连接。在我们的例子PBDLG中，我们要看条目钩子函数执行的简单功能。在“高级对话框”一章里，我们将讨论条目钩子函数的高级特性。

3.4.2 对话框通讯

对话框的子系统通过传送各种信息进行通讯(见图3.10)。每种信息都是包括一个信息类型和一个对应联合的数据结构。对话管理程序通过DialogMessage结构(在<dlogitem.h>中定义)与对话框钩子函数通讯。某些公用的对话钩子信息类型为：

DIALOG_MESSAGE_CREATE	在条目钩函数被传送生成信息以前传送
DIALOG_MESSAGE_INIT	在所有条目钩函数被传送生成信息以后传送
DIALOG_MESSAGE_DESTROY	在对话框要清除时传送
DIALOG_MESSAGE_UPDATE	在对话管理程序更新对话框后传送
DIALOG_MESSAGE_BUTTON	在一个鼠标按钮事件出现在对话框内时传送

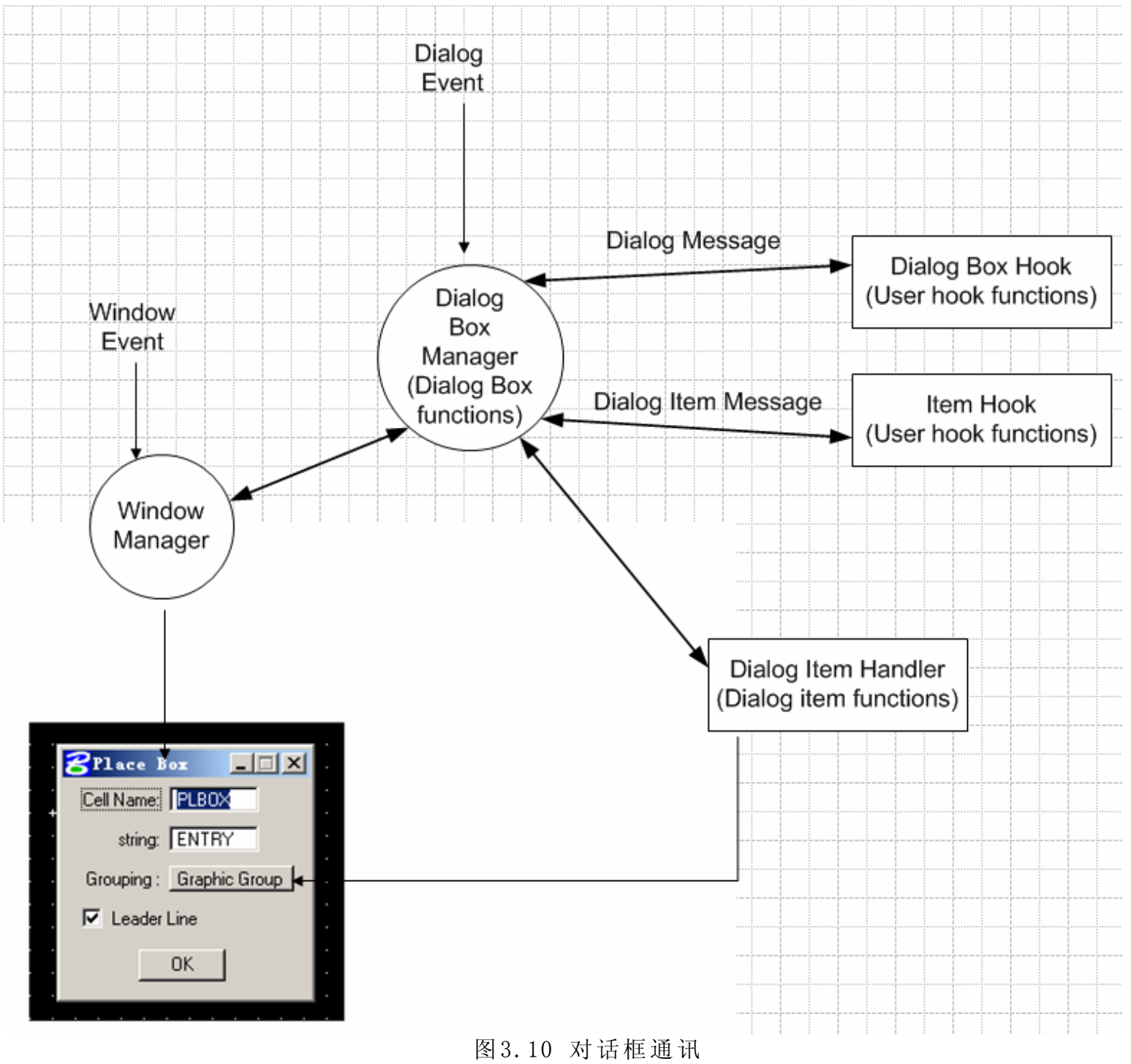


图3.10 对话框通讯

当对话框条目钩子函数被调用时,它接收信息结构DialogItemMessage(在<dlogitem.h>中定义)。一些公用信息类型为:

DITEM_MESSAGE_CREATE	条目生成后传送
DITEM_MESSAGE_DESTROY	条目清除时传送
DITEM_MESSAGE_BUTTON	当鼠标敏感条目中出现鼠标按钮事件时传送
DITEM_MESSAGE_INIT	条目生成后传送
DITEM_MESSAGE_SETSTATE	在一个条目管理器需要设置一个条目的状态时传送
DITEM_MESSAGE_GETSTATE	在一个条目控制需要决定一个条目的状态时传送

一. 调试对话信息

对话框管理程序通过命令DMSG提供一个信息跟踪功能。下面键入信息在对话框钩子函数、条目钩子函数或条目管理器之间打开一个对话框，并显示所有信息。

键　入	解　释
DMSG ITEMDEBUG[ON / OFF]	打开或关闭传送到对话条目钩函数的信息的显示开关
DMSG HANDLERDEBUG[ON / OFF]	打开或关闭已传送给条目控制信息显示开关
DMSG DIALOGDEBUG	打开或关闭送给一个对话钩函数的信息的显[ON/OFF]示开关
DMSG VERBOSEDEBUG[ON/OFF]	上面信息的简要及详细调试信息的开关
DMSG CLEARDEBUG	清除对话调试信息屏幕

二. 钩子函数 ID

钩子函数 ID 是我们可以分配的正的长整型数。使用 ID 允许对话框管理程序通过给定的 ID 去寻找适当的钩函数。通过把钩函数 ID 及函数地址传送到 mdldialog_hookPublish，我们让对话框管理程序识别该钩子函数。

/* publish dialog item hooks */

```
mdlDialog_hookPublish(sizeof(uHooks)/sizeof(DialogHookInfo), UHooks);
```

当我们执行 PLACE BOX 命令时，通过函数 mdlDialog_open 把对话框显示在屏幕上。

```
/* display the dialog box if it is notalready displayed. */
if(!mdlDialog_find(DIALOGID_PlaceBox, NULL))
mdlDialog_open(NULL, DIALOGID_PlaceBox);
```

我们可以通过操作适当的对话框条目来改变缺省的设置。然而，直到我们按"OK"按钮时，方框内的字符才被放置。在对话框资源文件 pbdlgdlg.r 中，我们已经定义了按钮对话框条目，以便得到一个钩子函数 ID，即 HOOKITEMID_Button_Place。

```
DItem_PushButtonRSC PUSHBUTTONID_Ok =
{
NOT_DEFAULT_BUTTON, NOHELP, MHELP,
HOOKITEMID_Button_Place, 0, NOCMD, MCMD, "",
"OK"
};
```

在我们的程序 pbdlg.mc 开始时，我们已定义了钩函数 id 为 HOOKITEMID_Button_Place，指向函数 PlButtonHook，当按“OK”按钮时，它便激活 PlButtonHook 函数。

```
static DialogHookInfo uHooks[] =
{
{HOOKITEMID_Button_Place, plButtonHook}
};
```

plButtonHook 函数的变量是指向 DialogItemMessage 结构的指针，这是前面涉及到的信息结构。不同项支持不同的对话框条目信息。参考 MDL 手册我们发现按钮条目支持信息类型 DITEM_MESSAGE_BUTTON。这表明用户已经按下“OK”按钮。

如果我们把结构 DialogItemMessage 中的变量 msgUnderStood 设为 TRUE，我们的钩子函数将处理这个信息。如果设为 FALSE，我们把它留给条目管理器进行缺省处理。

```
Private int plButtonHook(DialogItemMessage *dimP)
{
dimP－>msgUnderStood = TRUE;     。
Switch (dimP -> messageTyge)
{
 case  DITEM_MESSAGE_BUTTON:
  {
  mdlState_startPrimitive(placeBox_firstPoint, placeBox_done, 1, 2);
  Break;
  }
  Default:
  {
  /*tell the dialog manager that we dont handle this message: */
  dimP->msgUnderstood=FALSE;
  break;
  }
 }
}
```

3.5 资源管理

对话框使用资源文件存贮屏幕上的对话框位置。用户资源存贮在环境变量MS_RSRCPATH指定的目录内的文件userpref.rsc中。MS_RSRCPATH的缺省目录是＼ustation＼data。

用户优先选择是一个生成、修改和删除的动态资源的实例。开发和测试动态资源中， 删除userpref.rsc并让我们的应用重建它是较为安全的。我们已经发现访问旧资源文件格式会在MDL程序中将引起不正常行为。

注意： *在开发流程中删除旧的userfref.rsc文件，并使应用重建它。*

下面是资源管理函数的汇总：

mdlResource_add	给资源文件增加一个新资源
mdlResource_changeAlias	给资源指定一个新的别名
mdlResource_closeFile	关闭一个资源文件
mdlResource_createFile	从当前文件生成一个新资源文件
mdlResource_delete	从资源文件删除一个资源
mdlResource_directAdd	用fwrite增加一个新资源
mdlResource_directAddComplete	告诉资源管理程序直接增加已经完成

mdlResource_directLoad	用fread安装一个资源
mdlResource_free	除去一个已装入的资源
mdlResource_getRscldByAlias	用别名得到资源ID编号
mdlResource_load	把资源装入内存
mdlResource_loadFromStringList	从StringList资源装入一个字符串
mdlResource_openFile	打开一个资源文件
mdlResource_query	查询一个资源的信息
mdlResource_queryClass	查询一个资源类别的信息
mdlResource_queryFile	查询一个资源文件的信息
mdlResource_resize	调整资源的大小
mdlResource_write	把一个更新的资源写回资源文件

3.5.1 用户优先选择资源

函数mdlResource_openFile打开用户资源文件。如果对话框管理程序找不到指定资源文件，则打开缺省资源文件userpref.rsc。

如果我们初次运行我们的MDL应用，资源文件中将没有任何优先选择。如果我们的应用没有发现资源，则用一组初始值来开始。否则我们使用上一次对话的优先选择。

```
if(!boxRscP)
{
/*  No resource was found  *  /
plBoxInfo->groupMode=0;
plBoxInfo->leaderline=0;
strcpy(plBoxInfo->cellName,” PLBOX”);
plBoxInfo->String[0]=’ \0’ ;
}else{
/*Copy resource into internal structure*/
*plBoxInfo=*boxRscP;
mdlResource_free(boxRscP);
}
```

3.5.2 存贮资源

如果我们结束我们的应用，则需要存贮参数以便下一次使用。我们可以建立一个状态函数，当卸下应用时便执行该函数。 用传递给mdlsystem—setFuction参数

SYSTEM__UNLOAD__PROGRAM，可以保证在卸载时调用清除函数。

mdISystem__setFUncition(SYSTEM__UNLOAD__PROGRAM，UnloadFUnction);

卸载函数从磁盘打开用户优先选择文件。它将为这个具体应用寻找资源位置。如果没找到，将生成资源。如果找到，则将获取当前内存中的资源信息并把它存在磁盘上。卸载程序是一个标准函数，它将包含在seeddlg应用中。

```
Private int unloadFunction()
{
RscFileHandle  userPrefsH;
PlaceBoxlnfo    *boxRscp;
/*open userpref。rsc t0 hold our  small pref res(xlrce  */
mdiDialog userPrefFileOpen(&userPrefsH);
boXRscP=(PlaceBoxlnfo*)mdlResource_load(NULL, RTYPE__plBox,
RSCID__plBoxprefs);
if(lboXRscP)
{
/*  Our pref resource does  not exist, SO add it*/
mdlResource addd(userPrefsH,RTYPE plBox,RSCID__plBoxprefs
plBoxInfo, sizeof(placeBoxInfo),NULL);
}
else
{
*boXRscP=*PlBoxInfo;
/*write out and free the updated resource*/
mdlResource__write(boXRscP);
mdlResource__free(boxascP);
}
```

```
/*clean up*/
   mdlResource__closeFile(userPrefsH);
   free(plBoxInfo)'
   return(FALSE);
   }
```

3.6 C 表达式

MDL提供在运行期间计算C表达式的功能。对话管理程序用C表达式去管理访问字符串，这个C表达式可能含有对结构或联合中字段的引用，这些结构或联合必须在资源文件中定义。

访问字符串

在本章开始，我们简要介绍了访问字符串。在下面例子中，被访问字符串是plBoxInfo->cellName。

```
   DItem__TextRsc TEXTID__CELLName=
   {
   NOCMD, LCMD, NOSYNONYM, NOHELP, MHELP, NOHOOK。NOARG。
   8, ”%s”, ”%s”, ”” . ””, NOMASK, NOCONCAT,
   ”Cell Name”: ”,
   “plBoxInfo->cellName”
   }:
```

访问字符串为检查和修改对话框条目定义变量、结构和结构的指针。我们必须通过发布变量、结构或指针，使对话框条目能够管理我们的数据。

在我们的应用中，我们已经在文件pbdlgtyp.mt中定义结构和联合的说明。这个文件含有包含文件和publishStructures语句。publishStructures语句识别在资源源文件中定义的结构。这个结构不能含有动态说明(分配空间的那些说明)或可执行语句。

```
   #include”pbdlgdlg.h”
   publishStructures(placeboxinfo);
```

通过类型发生器传送文件pbdlgtyp.mt将产生一个资源源文件pbdgtyp.r。我们可以把它做为对资源编译程序的输入，以便生成资源文件。

头文件pbdlgdlg.h包含所需要的结构和联合的定义，pbdlgdlg.h如下。注意所有的ID都是正数，因为MicroStation使用负数。

```
/*-------------------------------------------------------------------+
|  Dialog  Box  IDs                                                  |
+-------------------------------------------------------------------*/
   #define DIALOGID_PlaceBox         1
/*-------------------------------------------------------------------+
|  PopupMenu  Item  IDs                                              |
+-------------------------------------------------------------------*/
   #define OPTIONBUTTONID_groupmode  1
/*-------------------------------------------------------------------+
|   PushButton  Item  IDs                                            |
+-------------------------------------------------------------------*/
   #define PUSHBUTTONID_ok           1
/*-------------------------------------------------------------------+
|  Resource Type and ID for Prefs                                    |
+-------------------------------------------------------------------*/
   #define RTYPE_plBox            “pLBt”
   #define RSCID_plBoxPrefs       1
/*-------------------------------------------------------------------+
|Text Item IDs                                                       |
+------------------------------------- ------------------------------*/
   #define TEXTID_cellName         1
   #define TEXTID_String           2
/*-------------------------------------------------------------------+
|Toggle Button IDs                                                   |
+-------------------------------------------------------------------*/
   #define TOGGLEID_leaderLine     1
/*-------------------------------------------------------------------+
|   Hook   Id’s                                                      |
+-------------------------------------------------------------------*/
```

```
   #define HOOKITEMID_Button_Place    1
/*-------------------------------------------------------------------------+
|Local Structure Definitions                                               |
+-------------------------------------------------------------------------*/
   typedef struct placeboxinfo
   {
   char     cellName[8];
   char     String[127];
   int     groumMode;
   int     leaderLine;
   }
   PlaceBoxInfo;
```

3.6.1 类型发生器

类型发生器rsctype编译结构和联合的说明，并生成一个定义。

类型发生器的命令行语法如下：
rsctype[-flag，-flag，…]input-file
这里的input-file是我们的源程序。如果不指定扩展名，则类型发生器在文件名后加上“.mt”。
这里的-flag是选项。有效的选项是：
-dname 值用于使用一个值来定义名字。名字在预处理器#define语句中可找到。
-idir 为搜寻包含文件增加附加目录。编译程序最多可支持40个包含目录。
-ofilename 资源源输出文件名，若不指定，则输入文件前缀的后面加扩展名”.r”。
-p   显示预处理器输出。
-v   显示编译过程。

3.6.2 C 表达式函数

MDL提供了一些发布及管理符号集的函数。这些符号集可以含有一个C变量名或一个C函数名。以下是这些函数的汇总；

mdlCExpression_freeSet	释放一个符号集
mdlCExpression_generateMessage	生成一个错误信息
mdlCExpression_getValue	计算一个C表达式并返回这个值
mdlCExpression_initializeSet	初始一个与对话框、计算器/预处理器一起使用的符号集
mdlCExpression_isArray	决定哪一种类型定义是一个数组
mdlCExpression_isCharPointer	决定哪一种类型定义是一个字符指针,
mdlCExpression_isStructUnion	决定哪一种类型定义是一个结构或联合
mdlCExpression_symbolPublish	发布对话管理程序的符号
mdlCExpression_setValue	计算一个表达式并设置其值
mdlCExpression_typeArray	做一个数组类型定义
mdlCExpression_typeFromRsc	根据资源文件中的定义，生成一个结构或联合，的内存常驻定义
mdlCExpression_typePointer	做一个指针类型定义符号集.

符号集是一个指向内存区域的指针。在这个内存区域内对话条目可以找到应用的变量、结构和指针。为了建立并公布要存取的结构plBoxlnfo，我们用以下命令：

```
setP=mdlCExpresslon_initializeSet(VISIBILITY_DIALOG_BOX, 0, TRUE)：
boxTypeP=mdlCExpression_typeFromRsc(setP,”placeboxinfo”, NULL);
boxTypeP=mdlCExpression_typePointer(setP,boxTypeP)：
mdlCExpression_symbolPublish(setP,”plBoxInfo”,SYMBOL_CLASS_VAR,
boxTypeP, plBoxInfo);
```

第一行，函数mdlCExpression_initializeSet使setP符号集初始化。可见标志被分配给每一个符号。在搜寻过程中，检查可见标志来决定符号集是否包括在搜寻中。

当这些符号用于对话框时，我们规定VISIBILITY_DIALOG_BOX。这些符号如果用于计算器或预处理器，我们则指定VISIBILITY_CALCVLATOR。如果符号同时用于二者，则规定(VISIBILITY_DIALOG_BOX＼VISIBILITY_CALULATOR)。若符号与调试程序一起使用，则我们规定VISIBILITY_DBUGGER。

在下一行，我们用一个对mdlCExpression_typePointer的调用生成一个结构placeboxinfo的记忆定义。函数mdlCExpression_typePointer把这个符号定义为一个指针。

最后我们用函数mdlCExpression_symbolPblish发布要使用的符号。符号类别如下：

```
SYMBOL_CLASS_FUNCTION     一个MDL函数
SYMBOL_CLASS_SCOPE        只被调试程序使用
```

SYMBOL_CLASS_SOURCE　　只被调试程序使用
SYMBOL_CLASS_STRUCT　　结构及联合
SYMBOL_CLASS_VAR　　变量

3.6.3 图形组

在我们的第一个例子中，我们说明了如何生成一个孤立单元(把元素组合成一组的一种方法)。在这个例子中我们将使用这个图形组，使用的方法是简单易懂的。TCB变量tcb->cugraf的内容用于给出元素的图形组号。如果编号为零，则元素不属于任何组。

TCB变量tcb->graphic含有下一个空闲图形组编号。如果我们把这个编号数赋给 tcb->cuqraf，我们从此以后放置的所有元素都将属于这个编号的组。

```
int Size=1,offset=GRAPHIC;
    if(plBoxInfo->groupMode\=2)
    {
    tcb->cugraf=tcb->graphic:
    tcb->graphic++;
    }
    generateImage(pt, view, NORMALDRAW);
    if(plBoxInfo->groupMode=2)
    {
    mdlParams_storeType9Variable(tcb->9raphic, size, offset);
    tcb->cugraf=0;
    }
```

PBDLG程序的大部分取自PLBOX，除了图形组、放置箭头及钩函数以外。我们要到有关章节再讨论如何放置箭头，这一章只讨论PBDLG应用。

这里是我们的对话框应用的完整程序。键入mdl l pbdlg安装应用。键入PLACE BOX则执行它。因为这个命令与安装以前的应用PLBOX使用的命令相同，所以你必须卸去先前的应用。

```
/*----------------------------------------------------------------------+
|  Copyright (C) 1991 Mach Dinh-Vu, All Rights Reserved
|  Pbdlg.mc  -  place a text string and a box enclosing the string.
|              This MDL program shows how to use the dialog box in
|              A working application
|                                                                  | |         PLACE
BOX                                                  |
+----------------------------------------------------------------------*/
#include <tcb.h>
#include <mselems.h>
#include <global.h>
#include <scanner.h>
#include <msinputq.h>
#include <userfnc.h>
#include <mdl.h>
#include <cexpr.h>
#include <rscdefs.h>
#include <dlogitem.h>

#include  “pbdlgcmd.h”      /* Generated by resource compiler(rcomp) */
#include  “pbdlgdlg.h”       /* Need to know dialog id to open */

/*----------------------------------------------------------------------+
|   Private Function declaration                                       |
+----------------------------------------------------------------------*/
Private int plButtonHook ();

/*----------------------------------------------------------------------+
|   Private Global variables                                           |
+----------------------------------------------------------------------*/
Static DialogHookInfo uHooks [] =
{
```

```
{HOOKITEMID__Button__place,plButtonHook}
};
Dpoint3d   pntP [2];
static PlaceBoxInfo     *placeBoxInfo;

#define GRAPHIC 1160
/*-----------------------------------------------------------------------+
|   name    unloadFunction                                               |
+-----------------------------------------------------------------------*/
private int unloadFunction()
{
 RscFileHandle     userPrefsH;
 PlaceBoxInfo      *boxRscP;
/*Open userpref.rsc to hold our small pref resource*/
mdlDialog_userPrefFileopen (&userPrefsH);
boXRscP = (PlaceBoxInfo *)mdlResource_load(NULL,RTYPE__plBox,
          RSCID__plboxprefs);

if (! boxRscP)
{
  /* Our pref resource does not exist,so add it */
mdlResource__add(userprefsH,RTYPE__plBox,RSCID__PlBoxPrefs,
                plBoxInfo,sizeof(placeBoxInfo),NULL);
}
else
{
*boxRscp=*plBoxInfo;
/*Write out and free the updated resource*/
mdlResource__write(boxRscp);
mdlResource__free(boxRSCP);
}
/*Clean up*/
mdlresource__closeFile(userPrefsH);
free(plBoxInfo);
return(FALSE);
}

/*-----------------------------------------------------------------------+
|  name    main                                                         |
+-----------------------------------------------------------------------*/
main()
{
RscFileHandle rfHandle,userprefsH;
placeBoxInfo *boxRscp;
char   *setp,*boxTypep;

/*publish dialog item hooks*/
mdlDialog__hookPublish(sizeof(uHooks)/sizeof(DialogHookInfo),uHooks);

/* Open our file for access to command table and dialog */
mdlResource_openFile (&rfHandle, NULL, FALSE);

/* setup plBoxInfo */
plBoxInfo = malloc ( sizeof (PlaceBoxInfo));


/* Prepare to read resource. The resource file was used to save
 information the last time place box was used. */
 boxRscP = NULL;
 userPrefsH = NULL;
mdlDialog__userPrefFilopen(&userPrefsH);
 if (userPrefsH)
```

```
     boxRscP = (PlaceBoxInfo *)mdlResource_load (NULL, RTYPE_plBox,
           RSCID__plBoXPrefs);
if (!boXRscP)
{
      /* No resource was found */
      plBoxInfo->groupMode = 0;
     plBoxInfo->leadeline = 0;
      strcpy (plBoxInfo->cellName, "PLBOX" );
      plBoxInfo->String[0]= '';
} else {
      /* Copy resource into internal structure */
   *plBoxInfo = *boxRscP;
      /* This is unnecessary because the closeFile will free all resources,
      * but it is recommended practice */
      mdlResource free (boxRscP);
}
if (userPrefsH)
      mdlResource closeFile (userPrefsH);
/* Set up and Publish plBoxInfo for access by the dialog manager */
setP = mdlCExpression_initializeset (VISIBILITY_DIALOG__BOX, 0, TRUE);
boxTypeP = mdlCEXpression__typeFromRsc (setP, "placeboxinfo", NULL);
boxTypeP = mdlCExpression--typePointer (setP, boXTypeP );
mdlCExpression_symbolPublish(setP, "plBoxInfo", SYMBOL_CLASS_VAR,
            boxTypeP, &plBoxInfo);
/* Make sure our function gets called at unload time */
mdlsystem__setFunbction (SYSTEM_UNLOAD_PROGRAM, unloadFunction);
/* Load the command table */
if (mdlParse_loadCommandTable (NULL) = NULL)
      mdlOutput_error ("unable to load command table.");
else
      mdlOutput_error ("Key-in PLACE BOX to execute.");
```

```
/*-------------------------------------------------------------------+
|  name generateImage - dynamic function for complex case.           |
+-------------------------------------------------------------------*/
Private int      generateImage
(
Dpoint3d *pt,
int           view,
int           drawMode
)
{
  MSELementUnion el;
  Dpoint3d       tPts[3];
  Dpoint3d       origin;


Dpoint3d       shapep [5]
double         zangle;
long           cellFilePos;
unsigned long arrowsize;
arrowsize = tcb->chheight / 2;
pntP[1] = *pt;
mdlCurrTrans__begin( );
mdlCurrTrans__identity ( );
mdlCurrTrans__translateOrigin (&pntP[0])
mdlCurrTrans__invtransPointArray ( tPts, pntP, 2 );
origin = tPts[1];     /* origin of text */
/* Create Text in dgnBuf for MicroStation Dynamics to display */
mdlText create (&el, NULL, plBoxInfo->String, &tPts[1],
                NULL,NULL,NULL,NULL);
mdlELement display (&el,drawMode)
if (drawMode = NORMALDRAW)
```

```
{
      if (plBoxInfo->groupMode = 1)
            cellFilePos=mdlCell_begin(plBoxInfo->cellName,NULL,NULL,0);
     mdlELement__add(&el);
}
mdlText__extractShape(shapep, NULL, &el, TRUE, view);
if (tPts[0].x < origin.x)
{
if (tPts[0].y < origin.y)
    {
            tPts[1].x=shapep[0].x;
            tPts[1].y=shapep[0].y;
      } else {
            tPts[1].x=shapep[3].x;
            tPts[1].y=shapep[3].y;
    }
} else {

%
      if (tPts[0].y < origin.y)
    {
            tPts[1].x=shapep[1].x;
            tPts[1].y=shapep[1].y;
            tPts[1].x=shapep[2].x;
            tPts[1].y=shapeP[2].Y;
     }
mdlshape_create(&el, NULL, shapep, 5, -1);
mdlELement__display (&el, drawMode)
if (drawMode = NORMALDRAW)
      mdlELement__add(&el)
 /* Create shape in dgnBuf for MicroStation Dynamics to display */
 if (plBoxInfo->leaderLine)
 {
     mdlLine__create (&el, NULL, tPts);
     mdlELement__display (&el, drawMode);
     if (drawMode = NORMALDRAW)
        mdlELement__add(&el)


       /* calculate angle of line */
      zangle = atan2 ((tPts[0].y-tPts[1].y), (tPts[0].x-tPts[1].x));
      mdlCurrTrans rotateByAngles ( 0.0, 0.0, zangle);
       /* Create arrowhead */
      tPts[1] = tPts[0];
      tPts[2].X = tPts[0].x - arrowsize;
      tPts[2].y = tPts[0].y - (arrowsize/2);
      tPts[0].x -= arrowsize;
      tPts[0].y += arrowsize/2;
      mdlLinestring create (&ei, NULL, tPts, 3);
      mdlELement display (&el, drawMode);
      if (drawMode = NORMALDRAW)
            mdlELment_add (&els);
 }
 if (drawMode = NORMALDRAW && plBoxInfo->groupMode = 1)
 {
      mdlELement__add(&el);
      mdlCell_end(cellFilePos);
 }
 return  SUCCESS;
/*-------------------------------------------------------------------+
 |  name          placeBox_done                                      |
+-------------------------------------------------------------------*/
```

```
Private void    placeBox_done
(
)
{
 mdlstate+__clear ( );
 mdloutput__rscPrintf (MSG__PROMPT, NULL, 0, 4);
}
/*-----------------------------------------------------------------------+
 |  name          placeBox_secondPoint                                   |
+-----------------------------------------------------------------------*/
Private void   placeBox_secondPoint
(
Dpoint3d *pt,
int          view
)
{
  int size=l, offset=GRAPHIC;
  if (plBoxInfo->groypMode = 2)
 {
         tcb->cugraf=tcb->graphic;
         tcb->graphic++;
 }
  genarateImage(pt, view, NORMALDRAW);
  if (plBoxInfo->groupMode = 2 )
 {
         md]Params storeType9Variable(&tcb->graphic, size, offset);
         tcb->cugraf=0


}
  placeBox_done ( );
/*-----------------------------------------------------------------------+
 |    name          placeBox_firstPoint                                  |
+-----------------------------------------------------------------------*/
Private void    placeBox_firstPoint
(
Dpoint3d    *pt,
int          view
)
{
  /* save first point */
  pntP[0] = *pt;
  /* Set the datapoint state function for the second point. */
  mdlstate_setFunction (STATE_DATAPOINT, placeBox_secondPoint);
  mdlState_setFunction (STATE_RESET, placeBox_done);
  mdlOutput rscPrintf (MSG PROMPT, NULL, 0, 3);
  mdlstate_setunction (STATE DyNaMICS, generateImage);
}
/*-----------------------------------------------------------------------+
 |    name      placeSox_start                                           |
+-----------------------------------------------------------------------*/
cmdName placeBox_start
(
void
)
cmdNumber CMD PLACE BOX
{
  /* Display the dialog box if it it not already displayed.    */
  if (1 mdlDialogfind (DIALOGID_placeBox, NULL))
        md]Dialog_open (NULL, DIALOGID_PlaceBox);
/*-----------------------------------------------------------------------+
 |    name           plButtonHook                                        |
```

```
+--------------------------------------------------------------------------*/
Private int  plButtomHook
(
DialogItemMessage *dimP
  dimP->mxgUnderstood = TRUE;
  switch (dimP->messageType)
  {
  case DITEM MESSAGE BUTTON:
       {
         mdlstate_startPrimitive (placeBox_firstPoint, placeBox_done, 1, 2);
         break;
       }
  default:
            {
         /* tell the dialog manager that we don't handle this message */
         dimP->msgUnderstood = FALSE;
         break;
       }
  }
}


pbdlgdlg.r 含有定义对话杠的资源程序。
/*--------------------------------------------------------------------------+
|                    Place Box Dialog Resources                            |
+--------------------------------------------------------------------------*/
  #include <rscdefs.h>
  #include <dlogbox.h>
  #include <dlogids.h>
  #include "pbdlgdlg.h"
  #include "pbdlgcmd.h"
/*--------------------------------------------------------------------------+
|   Diago Box                                                               |
+--------------------------------------------------------------------------*/
  #define OVERALLWIDTH           25 * XC
  #define OVERALLHEIGHT          11 * YC
  #define NEWLINE                2 * YC
  #define  X1 11 * XC            /* cell Name */
  #define  X2 11 * XC            /* Leader L/ne */
  #define  X3 2 * XC             /* group Mode */
  #define  X4 (WVERALLWIDTH/2) - (5 * XC)  /* middle of OKAY button */
  #define  Y1 GENY(1)            /* cell Name */
  #define  Y2 Y1 + NEWLINE       /* input string */
  #define  Y3 Y2 + NEWLINE       /* group Mode */
 #define  Y4 Y3 + NEWLINE        /* leader Line */
  #define  Y5 Y4 + NEWLINE       /* Okay button  */
  #define  BW XC * 9             /* box width */
 DialogBoxRsc DIALOGID PlaceBox =
 {
    DIALOGATTR_DEFAULT,
    NOHELP, MHELP, NOHOOK, NOPARENTID,
    "Place Box",
   {
    {{X1, Y1, BW, 0}, Text, TEXTID_cellName, ON, 0, "", ""},
   {{mt, X2, Y2, 0}, Text, TEXTID_ String, ON, 0,"",        ""},
   {{X2, Y3, 0, 0},  0ptionButton, OPTIONBUTTON_ groupMode, ON, 0,"", ""},
    {{X3, Y4, 0, 0},  ToggleButton, TOGGLEID_leaderLine,  ON, 0, "", ""},
    {{X4, Y5, BW, 0},  PushButton, PUSHBUTTON_ok, ON, 0, "", ""}
   }
 };
/*--------------------------------------------------------------------------+
|        opuon Items                                                        |
```

```
+-------------------------------------------------------------------------*/
DItem_ OptionButtonRsc  OPTIONBUTTONID groupMode =
{
  NOSYNONYM, NOHELP, MHELP, NOHOOK,  NOARG,
  "plBoxInfo->groupMode",
  {
  {NOTYPE,  NOICON, NOCMD, LCMD,  0, NOMASK, ON,  "None"},
  {NOTYPE NOICON, NOCMD, LGMD,  1, NOMASK, ON,  "Cell"},
  {NOTYPE NOICON, NORD, LCMD, 2, NOMASK, ON, "Graphic Group"},
 }
};
/*-------------------------------------------------------------------------+
|            Text Items                                                    |
+-------------------------------------------------------------------------*/
     DItem_TextRsc TEXTID cellName =
    {
        NOCMD, LCMD,  NOSYNONYM, NOHELP,  MHELP, NOHOOK, NOARG,
        8,  "%S",  "%S",   "",      "",   "" ,  NOMASKt CONCAT,
        "Cell Name:",
        "plBoxInfo->cellName"
    {
        NOCMD, LCMD, NOSYNONYM,  NOHELP,  MHELP,  NOHOOK,  NOARG,
        127, "%S", "%S", "" , "", NOMASK,  CONCAT,
        "string:" ,
        "plBoxInfo->String"
    };
/*-------------------------------------------------------------------------+
|              Toggle Buttons                                              |
+-------------------------------------------------------------------------*/
*/
     DItem_ToggleButtonRsc TOGGLEID_leaderLine =
    {
        NOCMD,  MCMD,  NOSYNONYM,  NOHELP,  MCMD,  NOHOOK,  NOARG,
        NOMASK,  NOINVERT,
        "Leader Line",
        "plBoxInfo->leaderLine"
    };
/*-------------------------------------------------------------------------+
|                    Push Button Items                                     |
+-------------------------------------------------------------------------*/
DItem PushButtonRsc PUSHBUTTONID ok =
     {
        NOT DEFAULT BUTTON, NOHELP, MHELP,
        HOOKITEMID_Button_Place, 0, NOCMD, MCMD, "",
             "  OK  "
     };
```

在上一章中，信息和命令文件是一个文件。在这个例子里，我们把它分成 pbdlgmsg.r 和 pbdlgcmd.r。

```
/*-------------------------------------------------------------------------+
|                    PBDLGMSG.R                                            |
+-------------------------------------------------------------------------*/
      #include "rscdefs.h"
      #include "cmdclass.h"
      MessagaList 0 =
     {
          {
          { 0,  ""  },
             { 1, "MDL Place Box 2" },
             { 2, "Define Start Point" }
             { 3, "Define Box Position" },
          { 4, "PlBox Done" },
          }
```

```
    };
```

这个命令表与用于PLBOX应用使用的命令表相同。因此用PLACE BOX命令只能运行一个应用。如果你想使用PBLG应用，你必须卸去PLBOX应用。

```
/*-----------------------------------------------------------------------+
|       pbdlgcmd.r - command table for pbdlg.mc                          |
+-----------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"
#define    CT NONE       0
#define    CT PLACE       1
#define    CT BOX        2
Table  CT PLACE =
{
          1, CT_BOX,     PLACENENT,      REQ,     "PLACE" },
};
Table  CT BOX =
{
     { 1,   CT_NCNE,      INHERIT,       DEF,      "BOX" },
};
```

为了建立PBDLG应用，键入bmake-a pbdlg.

```
#-----------------------------------------------------------------------
#       PBDLG MDL Make File
#-----------------------------------------------------------------------
%include  $(MS)/mdl/include/mdl.mki
#        Define constants specific to this example
#
progmdl     = d:/proc3mdl/disk/
baseDir    = $ (progmdl) pbdlg/
objectDir   = $ (mdlexample) ob jects /
privateInc   = $(baseDir)
pbdlgObjs   = $ (objectDir) pbdlg.mo
pbdlgRscs    =   $ (objectDir)pbdlgcmd.rsc \
                           $ (objectDir)pbdlgdlgorsC \
                           $ (objectDir)pbdlgtyp.rsc \
                           $ (objectDir)pbdlgmsg.rsc \
                           $ (objectDir) pbdlg .top
#-----------------------------------------------------------------------
#        Generate Command Tables
#-----------------------------------------------------------------------
$ (PrivateInc) pbdlgcmd, h    : $ (baseDir) pbdlgcmd, r
$ (objectDir) pbdlgcmd, rsc   : $ (baseDir) pbdlgcmd, r
#-----------------------------------------------------------------------
#         Compile Dialog Resources
#-----------------------------------------------------------------------
$(objectDir)pbdlgdlg.rsc   : $(baseDir)pbdlgdlg.r $(PrivateInc)pbdlgcmd.h
#-----------------------------------------------------------------------
#       Prompts and command numbers
#-----------------------------------------------------------------------
#   Don't generate an include file for the prompts and command
$ (objectDir) pbdlgmsg. rsc : $ (baseDir) pbdlgmsg. R


#-----------------------------------------------------------------------
#        Make resource to publish structure (s)
#-----------------------------------------------------------------------
$(objectDir)pbdlgtyp.r       :$(baseDir)pbdlgtyp.mt    $(privateInc)pbdlgdlg.h
$ (objeccDir) pbdlgtyp.rsc    : $ (objectDir) pbdlgtyp.r
#-----------------------------------------------------------------------
#        Compile and link MDL Application
#-----------------------------------------------------------------------
```

```
$(objectDir)pbdlg.mo      : $(baseDir)pbdlg.mc $(PrivateInc)pbdlgcmd.h
$ (objectDir) pbdlg.mp      : $ (objectDir) pbdlg.mo
  $(msg)
  >$ (objectDir) temp.cmd
  -aS@
  -s8000
  $(linkOpts)
  $ (pbdlgObjs)
  $ (mdlLibs) ditemlib.ml
  <
  $ (linkCmd) @$ (objectDir)temp.cmd
  ~time
#--------------------------------------------------------------------------
#       Merge objects into one file
#--------------------------------------------------------------------------
  $ (mdlapps)pbdilg.ma              : $ (pbdlgRscs)
    $ (msg)
    >$ (objectDir) temp.cmd
    -o$@
    $ (pbdlgRscs)
    <
    $ (rscLibCmd) @$ (objectDir) temp. cmd
    ~time
```

第四章 元素的搜寻和操作

在前三章，我们看到了如何把一个元素加在设计文件内，在这一章我们讨论对已有元素的操作。我们需要对元素进行查找、修改并把修改过的元素写回设计文件。象UCM和MicroCSL这类开发工具，要求我们掌握所有步骤。从确保复杂元素的后面跟随字的正确，到如果新元素的大小不适合原先的位置就删除旧元素。这需要对设计文件，元素格式和文件指针有一个全面的了解。一个简单的失误就会引起文件损坏。

元素操作是开发一个MDL应用时最普通的操作。为了使我们的任务简单些，MDL提供了一系列处理元素的顶层操作函数。例如，函数mdlModify_elementSingle从设计文件中读一个元素，并为元素的每一组成部分调用一个函数。我们可以传递变量MODIFY_ORIG，这个变量对原来的元素做修改。如果新的元素的大小不适合原来的位置，这个函数将删除旧元素并在文件尾部增加新元素。这些特点使MDL成为一种功能强大的语言，它使程序员有了较多的时间去编写应用，而不必在繁琐无味的工作上花太多时间。

让我们用一个对元素进行提示并把它的层改变到当前层的例子来开始这一章。

4.1 元素搜寻

查找元素有三种方法：

- 通过指定元素。
- 在整个文件中搜索元素。
- 在围栅中搜索元素。

MDL提供了几种元素定位函数，我们将分别讨论每一种方法。不能说哪一种方法比另一种更好，只能说某些方法对某些应用最合适。

4.1.1 元素定位函数

以下是元素定位函数的汇总，关于变量的详细解释，请看MDL手册。

mdlLocate_allowLocked	为所有可显示元素设置搜索掩码。
mdlLocate_findElement	在主文件及其所有参考文件中搜寻一个元素。
mdlLocate_getProiectedPoint	返回靠近定位点的元素上的一个点。
mdlLocate_identifyElement	完成一个标识元素、高亮、提示的操作逻辑。
mdlLocate_init	初试化元素定位状态。
mdlLocate_noElemAllowLocked	设置从所有文件查找所有元素的搜索掩码，然后清除搜索类型掩码。
mdlLocate_noElemNoLocked	为所有未锁定元素设立搜索掩码，然后清除搜索类型掩码。
mdlLocate_normal	设置只搜索当前文件中可显示及未锁定元素的搜索掩码。
mdlLocate__clearElemSearchMask	清除特定元素类型的搜索掩码。
mdlLocate__setElemSearchMask	为指定元素类型设置搜索掩码
mdlLocate__setFunction	设置有关元素定位的异步功能。

我们首先要告诉MicroStation要寻找哪些元素。这要从一个元素类型数组设置一个搜索掩码。下面的函数setsearchType确定搜索11种常用的元素。元素的类型可以在头文件<mselems.h>中找到。(参见附录)

```
Private setsearchType()
   {
   static int searchType[]={CELL_HEADER_ELM, LINE_ELM, LINE_STRING_ELM,
                            SHAPE_ELM, TEXT_NODE_ELM, CURVE_ELM,
                            CMPLX_STRING_ELM, CONIC_ELM, TEXT_ELM ,
                            SURFACE_ELM, SOLID_ELM };

    mdlLocate_noElemNoLocked();
    mdlLocate_setElemSearchMask(sizeof(searChType)/sizeof(int),
                                searChType);
   }
```

函数mdlLocate_noElemNoLocked为主文件中所有未锁定元素设置了搜寻掩码，并清除了所有元素的搜寻位。因此，我们需要调用mdlLocate_setElemsearchMask来确定要搜寻的元素。

我们可以使用mdlLocate_normal，它与mdlLocate_noElemNoLocked相同。但是它设立寻找所有可显示元素的搜寻掩码。用mdlLocate_normal来避免列出搜寻元素的麻烦，它和我们使用数组searchType一样。使用mdlLocate_setElemSearchMask的优点是我们可以用函数setSearchType做为未来程序的种子。

4.1.2 元素编辑函数

元素编辑通过三个单独的函数进行。每个函数都处理三种不同的编辑方法，即单个元素的修改、元素组(一个选择集或一个图形组)的修改和元素的描述符(在本书后部分讨论)的修改，最后一个函数mdlModify_freeGGMap是元素编辑函数的支持程序。

mdlModify_elementSingle	用于编辑单个元素(或复杂元素)的函数。
mdlModify_elementMulti	与单个元素编辑函数相同。但它对选择集或图形组(如果图形组打开)工作。
mdlModify_elementDescr	在内存中修改元素描述符。
mdlModify_freeGGMap	在拷贝一个元素时，要分配内存来处理从原图形组编号到新的编号的映象。这个函数释放分配的内存。

函数mdlState_startModifyCommand将为定位和修改命令设置步骤。由于它是一个状态函数，所以我们要建立相应的事件响应函数。当函数change_single执行时，它在提示域中显示”IdentifyElement”，并在命令域和提示域中显示信息表中的相应字符串。当选择一个元素并确认时，则执行函数mod__accept。

```
Private void change_single()
{
setSearchType();
mdlState_startModifyCommand(
                change single, /*function to call on RESET*/
                mod_accept,    /*function to call on DATA */
                NULL,           /*function to call on SHOW*/
                NULL,          /*function to call on CLEAN*/
                1,             /*index into MESSAGE LIST*/
                2,             /*index PROMPT LIST*/
                TRUE,          /*Modify SELECTION SET ? */
                FALSE);        /*additional data points required*/
 mdlLocate_init();
}
```

4.1.3 选择集函数

选择集是操作元素的新方法。在MicroStation的老版本中，我们选择操作命令(如拷贝命令)，然后指定要操作的元素。选择集与之相反，它要求用户先指定元素，然后再选操作命令。这很象英语的名—动词与法语的动—名词之间的差别。选择集函数的汇总如下：

mdlSelect_addElement	把一个元素追加到当前选择集中。
mdlSelect_allElements	把所有可显示元素追加到选择集中。
mdlSelect_freeAll	清空选择集。
mdlSelect_isActive	若选择集激活则为真。
mdlSelect_removeElement	从选择集中移去一个指定元素。
mdlSelect_returnPositions	返回选择集中元素的文件位置。

一. 修改选择集

前面，我们定义了用一个选择集或一个元素上的数据点激活的函数mod_accept。函数mod_accept如下：

```
  Private void  mod_accept()
    {
    filePos    compOffset;
    int         currFile=0;
```

```
filePos=mdlElement_getFilePos(FILEPOS_CURRENT,&currFile);
if (mdlSelect_isActive())
    {
    mdlModify_elementMulti (currFile,  /*file to process*/
    filePos,                    /* file position for elemen */
    MODIFY_REQUEST_HEADERS,     /*process complex headers*/
    MODIFY_ORIG,                /* modify original element */


    editelem,                /*modify routine for each element*/
    NULL,                    /*parameters for editelem*/
    TRUE);                   /*process graphic group*/
   }
else
    {
    mdlModify elementSingle(currFile,
    filePos,                   /*file position for element*/
    MOOIFY REQUEST HEADERS,    /*process complex headers*/
    MODIFY ORIG,               /* modify original element */
    editelem,                  /*modify routine for each element*/
    NULL,                      /* parameters for editelem*/
    FALSE);                    /*offset for component elements*/
    }
mdlLocate_restart(FALSE);
}
```

函数 mdlElement_getFilePos()根据第一个变量返回一个文件位置。宏 FILEPOS_CURRENT 表示我们想要它返回当前元素的文件位置。

若选择集是激活的，则函数 mdllSelect__isActive 返回真值。 mdlModify_elementMulti 处理选择集(或图形组)中的每一个元素，并调用 editelem 进行修改。这类函数的优点在于它在命令行为上的一致性。

变量 MODIFY_REQUEST_HEADERS 表示将调用 editelem 来处理复杂头及所有组件元素。但是,如果我们对一个单元定位并改变了元素的层，同时我们不想改变单元头的层，因为它永远是零。因此，在函数 editelem 中我们必须检查单元头。允许的变量为：

MODIFY_REQUEST_NOHEADERS	不调用修改函数处理复杂头。
MODIFY_REQUEST_HEADERS	调用修改函数处理所有元素。
MODIFY_REOUEST_ONLYONE	为组件元素只调用一次修改函数。

变量 MODIFY_ORIG 通知 mdlModify_elementMulti 去调用 editelem，并修改原 来的元素。如果这个修改改变了元素的尺寸，则旧元素被删除，并把新的元素追加到文件末尾。修改标记的允许值为：

MODIFY_ORIG	修改原来的元素
MODIFY_COPY	修改原来的元素的拷贝
MODIFY_DONTERASEORIG	不从屏幕上擦除原来的元素
MODIFY_DONTDRAWNEW	不画出新生成的元素
MODIFY_DRAWINHILITE	用高亮色显示新元素

函数 editelem 是用户定义的修改元素的函数。我们将在以后讨论。

二. 修改单个元素

用函数 mdlModify_elementSingle 来修改单个元素。大部分变量与 mdlModify_elementMuiti 相同。对于复杂元素，我们必须做许多工作来保证修改函数 editelem 没被调用去处理复杂头。这并不是说复杂元素头不被改变。

一旦我们确定了元素的位置，我们用函数 mdlLocate_restart(FALES)重新开始一个元素搜索流程。这个函数允许我们使用第二个数据点做为前一个搜索的确定键，并同时成为下一个元素定位的开始。因此我们的应用看起来和感觉上都象一个 MicroStation 基本命令。

实际进行修改的函数是非常简单易懂的。要修改的元素作为一个变量传递给 editelem，同时我们给它指定一个新层。返回值 MODIFY_STATUS_REPLACE 通知 mdlModify_函数用新的元素替换原来的元素。在 editelem 内赋值会使内存发生变化，这一点是非常重要的。正是这个返回值改变文件中的元素。返回值汇总如下：

MODIFY_STATUS_ABORT	停止处理组件元素
MODIFY_STATUS_DELETE	删除当前元素
MODIFY_STATUS_ERROR	相当于 MODIFY_STATUS_FAIL|MODIFY_STATUS_ABORT。
MODIFY_STATUS_FAIL	出现错误，放弃所有修改
NODIFY_STATUS_NOCHANGE	不改变原来的元素
MODIFY_STATUS_REPLACE	用新元素替代原来的元素
MODIFY_STATUS_REPLACEDSCR	用新描述符替换原来的元素描述符.

当我们接收一个单元头时，我们不希望改变它的层，因为它总为零。通过使用MODIFY_STATUS_NOCHANGE，我们可一跳过单元头。

```
Private int editelem
(
MSElemenUnlon  * el
)
     {
     int    actlevel;

     if (el->ehdr->type == CELL_HEADER_ELM)
        return(MODIFY_STATUS_NOCHANGE);
     mdlParams_getActive(&actLevel,ACTIVEPARAM_LEVEL);
     el->ehdr.level=actlevel;
     return(MODIFY_STATUS_REPLACE);
     }
```

4.2 激活设置

有一系列函数可用于查询及改变激活的设置。我们使用这些函数可防止我们的代码破坏元素格式。另一个优点是我们不必为了元素的正确格式去查阅内部变量。例如，激活变量在 TCB 变量中定义为 tcb->symbology.color。等价的函数调用为：

```
mdlParams_getActive (&color,ACTIVEPARAM_COLOR);
```

mdlParams_getActive	得到激活设置
mdlParams_setActive	改变激活设置

常用的激活参数为：

激活参数	说明	键入
ACTIVEPARAM_ANGLE	激活角度	AA=
ACTIVEPARAM_AREAMODE	激活区域(实/空)	ACTIVE AREA
ACTIVEPARAM_AXISANGLE	轴增量	ACTIVE AXIS
ACTIVEPARAM_AXISORIGIN	轴原点	ACTIVE ORIGIN
ACTIVEPARAM_CAPMODE	3D 类型(面或实体)	ACTIVE CAPMODE
ACTIVEPARAM_CELlNAME	激活单元	AC=
ACTIVEPARAM_ClASS	激活类别	ACTIVE CLASS
ACTIVEPARAM_COLOR	激活颜色号	CO=
ACTIVEPARAM_COLOR_BY_NAME	激活颜色名	CO=
ACTIVEPARAM_DIMCOMPAT	尺寸标注兼容性	SET COMPATIBLE DIMENSION
ACTIVEPARAM_FILLMODE	激活填充	ACTIVE FlLL
ACTIVEPARAM_FONT	激活字型	FT=
ACTIVEPARAM_GRIDMODE	正交或轴测网格	ACTIVE GRIDMODE
ACTIVEPARAM_GRIDRATIO	网格高宽比	ACTIVE GRIDPATIO
ACTIVEPARAM_GRIDREF	参考网格	GR=
ACTIVEPARAM_GRlDUNITS	主网格	GU=
ACTIVEPARAM_KEYPOINT	捕捉因子	KY=
ACTIVEPARAM_LEVEL	激活层	LV=
ACTIVEPARAM_LINELENGTH	激活行长	LL=
ACTIVEPARAM_LINESPACING	激活行距	LS=
ACTIVEPARAM_LINESTYLE	激活线型	LC=
ACTlVEPARAM_L1NEWEIGHT	激活线宽	LW=
ACTIVEPARAM_MLINECOMPAT	多线兼容性	SET COMPATIBLE MLINE

ACTIVEPARAM_NODEJUST	文本结点对齐	ACTIVE TNJ
ACTIVEPARAM_PATTERNDELTA	激活图案增量	PD=
ACTIVEPARAM_PATTERNANGLE	激活图案角度	PA=
ACTIVEPARAM_PATTERNSCA1E	激活图案比例	AP=
ACTIVEPARAM_POINT	激活点	ACTIVE POINT
ACTIVEPARAM_SCALE	激活比例	AS=
ACTIVEPARAM_STREAMDELTA	激活流动增量	SD=
ACTIVEPARAM_STREAMTOLERANCE	激活流动容差	ST=
ACTIVEPARAM_STREAMANGLE	激活流动角度	ACTIVE STREAM ANGLE
ACTIVEPARAM_STREAMAREA	激活流动面积	ACTIVE STREAM AREA
ACTIVEPARAM_TAB	代替 TAB 的空格	NONE
ACTIVEPARAM_TAGINGCREMENT	TAG 增量	TI=
ACTIVEPARAM_TREMINATOR	激活线端符	LT=
ACTIVEPARAM_TERMINATORSCALE	端符比例	TS=
ACTIVEPARAM_TEXTHEIGHT	激活字符高度	TH=
ACTIVEPARAM_TEXTWIDTH	激活字符宽度	TW=
ACTIVEPARAM_TEXTJUST	激活字符对齐	ACTIVE TXJ
ACTIVEPARAM_UNITROUNDOFF	激活单位取舍	UR=

我们可通过用 MDL 函数 mdlElement_setProperties 来替代 el->ehdr.level=actlevel，来增强我们的代码的可移植性。修改过的 editelem 函数如下：

```
Private int editelem
(
MSElementUnion        *element
)
     {
     int     actlevel;

     if (mdlElement_getType(element) == CELL_HEADER_ELM)
          return MODIFY_STATUS_NOCHANGE;
     mdlParams_getActive(&actlevel, ACTIVEPARAM_LEVEL);
     mdlElement_setProperties (element,    /* element to change */
                               &actlevel, /* level */
                               NULL,       /* graphic group nmuber */
                               NULL,       /* class */
                               NULL,       /* locked */
                               NULL,       /* new */
                               NULL,       /* modified */
                               NULL,       /* view independent*/
                               NULL);      /* solid hole */
     return MODIFY_STATUS_REPLACE;
     }
```

下面是程序 modsing.mc 的完整清单。键入 mdl load modsing 来装入这个应用，键入 MODIFY SINGLE 执行这个应用。你也可以把它缩写为 MOD SING。

```
/*--------------------------------------------------------------------+
|IncludeFiles                                                         |
+--------------------------------------------------------------------*/
#include <time.h>
#include <stdarg.h>
#include <mdl.h>          /* MDL Library funcs structures & constants */
#include <global.h>
#include <mselems.h>
#include <userfnc.h>
#include <tcb.h>
#include <msrmgr.h>

#include <msparse.fdf>
#include <msoutput.fdf>
#include <mssystem.fdf>
```

```
#include <msstate.fdf>
#include <mslocate.fdf>
#include <mselemen.fdf>
#include <msmisc.fdf>
#include <msselect.fdf>

#include "modsing.h"

/*----------------------------------------------------------------------+
|      name setSearchType                                               |
+----------------------------------------------------------------------*/
Private void setsearchType
(
void
)
     {
     static int searchType[]={CELL_HEADER_ELM,LINE_ELM,
                              LINE_STRING_ELM,SHAPE_ELM,
                              TEXT_NODE_ELM,CURVE_ELM,
                              CMPLX_STRING_ELM,
                              CONIC_ELM,CMPLX_SHAPE_ELM,
                              ELLIPSE_ELM,ARC_ELM,
                              TEXT_ELM,SURFACE_ELM,SOLID_ELM};
     /*initialize search creteria to find nothing*/
     mdlLocate_noElemNoLocked();
     /*add elements to search to list*/
     mdlLocate_setElemSearchMask(sizeof(searchType)/sizeof(int),
                                 searchType);
     }
/*----------------------------------------------------------------------+
|        name       editelem                                            |
+----------------------------------------------------------------------*/
Private int editelem
(
MSElementUnion        *element
)
     {
     int    actlevel;

     if (mdlElement_getType(element) == CELL_HEADER_ELM)
          return MODIFY_STATUS_NOCHANGE;
     mdlParams_getActive(&actlevel, ACTIVEPARAM_LEVEL);
     mdlElement_setProperties (element,    /* element to change */
                           &actlevel, /* level */
                           NULL,       /* graphic group nmuber */
                           NULL,       /* class */
                           NULL,       /* locked */
                           NULL,       /* new */
                           NULL,       /* modified */
                           NULL,       /* view independent*/
                           NULL);      /* solid hole */
     return MODIFY_STATUS_REPLACE;
     }
/*----------------------------------------------------------------------+
|     name       mod_accept                                             |
+----------------------------------------------------------------------*/
Private  void  mod_accept
(
void
)
     {
     ULong            filePos;
```

```
    DgnModelRefP      currFile=0;

    filePos = mdlElement_getFilePos(FILEPOS_CURRENT,&currFile);
    if (mdlSelect_isActive())
        {
        mdlModify_elementMulti(currFile,/* file to process */
                    filePos,     /* file position for element */
                    MODIFY_REQUEST_HEADERS,/* process complex headers */
                    MODIFY_ORIG, /* modify original element */
                    editelem,     /* modify routine for each element */
                    NULL,         /* parameters for editelem */
                    TRUE);        /* process graphic group */
        }
    else
        {
        mdlModify_elementSingle(currFile,
                    filePos,       /* file position for element */
                    MODIFY_REQUEST_HEADERS, /* process complex headers */
                    MODIFY_ORIG,   /* modify original element */
                    editelem,      /* modify routine for each element */
                    NULL,         /* parameters for editelem */
                    0);           /* offset for component elements */
        }
    /*restart the element location process*/
    mdlLocate_restart(FALSE);
    }
/*----------------------------------------------------------------------+
| name   change_sing                                                    |
+----------------------------------------------------------------------*/
cmdName void change_sing
(
void
)
cmdNumber  CMD_MODIFY_SINGLE
    {
    setsearchType();
    mdlState_startModifyCommand(
                    change_sing,
                    mod_accept, /* function to call on DATA */
                    NULL,        /* function to call for DYNAMIC */
                    NULL,        /* function to call on SHOW */
                    NULL,        /* function to call on CLEAN */
                    1,           /* index into MESSAGE LIST */
                    3,           /* index into PROMPT LIST */
                    TRUE,        /* Modify SELECTION SET ? */
                    FALSE);      /* additional data points required */
    mdlLocate_init();
    }
/*----------------------------------------------------------------------+
| name   main                                                           |
| author Chaohua Lin          8/04/2006                                 |
+----------------------------------------------------------------------*/
Public int main
(
int                argc,
char              *argv[]
)
    {
    RscFileHandle   rfHandle;

    if (mdlParse_loadCommandTable(NULL)== NULL)
       mdlOutput_error("Unable to load command table.");
```

```
    mdlResource_openFile(&rfHandle, NULL, FALSE);
    mdlState_registerStringIds(0,1);
    mdlOutput_prompt("to execute, key-in MODIFY SINGLE");
    return SUCCESS;
    }
```

文件 modsing.r 有命令表。

```
/*----------------------------------------------------------------------+
|     name       modsing.r                                              |
+----------------------------------------------------------------------*/
#include   "rscdefs.h"
#include   "cmdclass.h"
#define     CT_NONE     0
#define     CT_MAIN     1
#define     CT_MODIFY   2

Table  CT_MAIN =
   {
   {1,  CT_MODIFY, MANIPULATION, REQ,  "Modify" },
   };
Table CT_MODIFY =
   {
   {1, CT_NONE,  INHERIT,  NONE,  "Single"},
   };
```

文件 modsingmsg.r 提示信息。

```
/*----------------------------------------------------------------------+
|   Include Files                          |
+----------------------------------------------------------------------*/
#include <rscdefs.h>
#define  MESSAGELISTID_Commands      0
#define  MESSAGELISTID_Prompts       1

MessageList MESSAGELISTID_Commands =
{
    {
    {1, "Modify Element Level"},
    }
};

MessageList MESSAGELISTID_Prompts =
{
    {
    {1,  "Enter first point"},
    {2,  "Enter next point"},
    {3,  "Accept/Reject element"},
    }
};
```

键入 bmake -a modsing 来建立 MODIFY 应用。

```
#--------------------------------------------------------------------------
# name   modsing.mke
# author Chaohua Lin       8/04/2006
#--------------------------------------------------------------------------
%if defined (_MakeFilePath)
BaseDir    = $(_MakeFilePath)
%else
BaseDir    = $(MS)/mdl/examples/modsing/
%endif

appName     = modsing
```

```
privateInc  = $(BaseDir)

#-------------------------------------------------------------------------
# mdl.mki contains the default rules for creating .rsc, .mo, etc files
#-------------------------------------------------------------------------
%include   $(MS)/mdl/include/mdl.mki

#-------------------------------------------------------------------------
# Define macros for files included in our link and resource merge
#-------------------------------------------------------------------------
lchObjs = \
    $(o)$(appName).mo

lchRscs = \
    $(o)$(appName).rsc \
    $(rscObjects)$(appName)msg.rsc \
    $(o)$(appName).mp

#-------------------------------------------------------------------------
#        Generate command table include & resource file
#-------------------------------------------------------------------------
$(genSrc)$(appName).h            : $(BaseDir)$(appName).r

$(o)$(appName).rsc               : $(BaseDir)$(appName).r

#-------------------------------------------------------------------------
#        Compile the MDL source object file
#-------------------------------------------------------------------------
$(o)$(appName).mo                : $(BaseDir)$(appName).mc

#-------------------------------------------------------------------------
#        Builds any translatable resource modules for the application.
#-------------------------------------------------------------------------
$(rscObjects)$(appName)msg.rsc   : $(BaseDir)$(appName)msg.r

#-------------------------------------------------------------------------
#  The following section generates the MDL Program module using
#  mlink. This module should contain ALL CODE resources and/or
#  libraries used by the application.
#-------------------------------------------------------------------------
$(o)$(appName).mp      : $(lchObjs)
        $(msg)
        > $(o)temp.cmd
        -a$@
        -s6000
        $(linkOpts)
        $(lchObjs)
        <
        $(MLinkCmd) @$(o)temp.cmd
        ~time

#-------------------------------------------------------------------------
#   The final step in building the application is lib'ing the applications
#   intermediate application with the translatable resources built in
#   this makefile. This step generates the final, and possibly translated,
#   MDL application.
#-------------------------------------------------------------------------
$(mdlapps)$(appName).ma   : $(lchRscs)
    $(msg)
    > $(o)make.opt
    -o$@
    $(lchRscs)
```

```
<
$(RLibCmd) @$(o)make.opt
~time
```

4.3 元素位置搜寻

在这一节，我们将通过扫描整个文件来查找一系列元素的位置。该程序的大部分与前面讨论过的单个元素修改程序类似。

4.3.1 扫描函数

下述函数用于确定搜索准则：

```
mdlScan_initScanList       初始化一个扫描表(ScanList)
mdlScan_initialize         把扫描表装入 MicroStation 设计文件扫描器。
mdlScan_setDrawnElements   设定扫描表返回可显示元素
mdlScan_noRangeCheck       设定扫描表在整个文件中查找所有元素位置
            ，用这种方法可以处理所有元素
mdlScan_singleviewClass    只返回当前在特定视图内显示的元素
mdlScan_viewRange          只寻找特定范围内的元素
mdlScan_file               根据已定的标准扫描文件
```

函数 change_all 查找所需要的元素的位置。第一步建立扫描表，表中有我们要查找并要进行操作的元素。

4.3.2 扫描标准

我们将定义一个 ExtScanList 类型的变量 scanList，ExtScanList 是在<scanner.h>中定义一个的结构体。使用 scanList，我们可以为扫描程序规定扫描标准，以便很快地查找元素的位置。下述常量规定扫描类型：

```
ELEMDATA       如果为真，则存贮数据
NESTCELL       如果为真，把一个单元当成一个元素来处理
PICKCELL       如果为真，则比较单元名
PROPCLAS       如果为真，则比较属性和类别
GRAPHGRP       如果为真，则比较图形组
MULTl          如果为真，则有多个扫描范围
SKEW           如果为真，则作非对称扫描
BOTH           如果为真，则同时得到指针和数据
ONEELEM        如果为真，则只取一个元素
ATTRENT        如果为真，比较属性连接实体

ATTROCC        如果为真，比较属性信息的出现
STOPSECT       如果为真，则检查停止段
LEVELS         如果为真，则比较层
ELEMTYPE       如果为真，则比较元素类型
```

以下常量用在 extendenType 中：

```
RETURN3D       如果为真，则从 2D 文件返回 3D 元素
FILEPOS        如果为真，则返回文件位置，不是块/字节
EXTATTR        如果为真，则做扩展的属性扫描
```

以下常量为属性标志符：

```
ELENEW         如果为新的则设置
ELEMOD         如果修改过则设置
ELEINVISIBLE   如果非可视则设置
ELERELTVE      如果与数据库相关则设置
ELEPLANR       如果为平面的则设置
ELESNAP        如果元素可捕捉则设置
ELEHOLE        如果元素为空心则设置
```

我们的第一步是确定扫描类型(Scantype)。我们希望扫描程序比较元素类型并一次返回一个元素。

```
scanList.scanType = ELEMTYPE | ONEELEM;
scanList.extendedType = FlLEPOS;
```

第二步，我们建立一个类型掩码，告诉扫描程序查找哪种类型元素的位置。typmask 变量是一个 16 位掩码，它的每一位都是一个元素类型。例加，若 typmask[0)的 0 位被设置，则我们将查找元素类型 1 的位置。若 typmask[1]的 0 位被设置，将查找元素类型 17 的位置，类型 17 元素是文本元素。typmask 数组为八个字符长，因此我们有 128(8 * 16)个可能的元素类型。大多数可显示元素类型在 1 到 32 范围之内。(注意：新的 MicroStation 元素，即多线，尺寸和和共享单元等是大干 32 的元素类型)。了解这些后，我们可以在 typmask 头两个字中设置所有的位。如下所示：

```
scanList.typmask[0]     =0xffff;     /*set all bits using HEX*/
scanList.typmask[l]     =0xffff;
```

我们并不这样做，因为扫描线和圆要做许多工作来计算十六位数值。在包含文件 <mselems.h> 中有一系列元素类型掩码的预定义。例如，掩码 TMSK0_LINE 必须与 typmask[0]进行“或”运算，同样 TMSK1_TEXT 必须与 typmask[1]进行“或”运算，我们的程序如下：

```
scanList.typmask[0] =TMSK0_LINE | TMSK0_LINE_STRING | TMSK0_SHAPE;
scanList.typmask[0]|=TMSK0_TEXT_NODE | TMSK0_CURVE;
scanList.typmask[0]|=TMSK0_CONIC | TMSK0_CMPLX_SHAPE;
scanList.typmask[0]|=TMSK0_ARC | TMSK0_CMPLX_STRING | TMSK0_ELLIPSE;
scanList.typmask[1] =TMSK1_TEXT | TMSK1_SURFACE | TMSK1_SOLID;
```

在调试程序中，如果我们在 mbd>提示符下键入 scanList.typmask [0]，我们可以看到 scanList.typmask[0]的带有适当设置位的十进制值。

注意：扫描程序将在满足搜寻标准的文件中查找所有元素的位置。不显示的 元素(即在当前视图中关闭层上的元素)也将被搜寻并修改。这也适用于不 可显示元素如复杂头、组数据元素、文件头元素和类型 66 元素。所以在扫描全文件时要非常小心。

现在，我们准备开始扫描。我们用 mdlScall_initScanList 初始扫描表。我们必须确定扫描程序不检查元素范围。如果我们正在扫描设计文件中的所有元素，我们则使用 mdlScan_noRangeCheck 来停止扫描程序的范围检查。我们使用函数 mdlScan_initialize 来初始化扫描程序，并调用 mdlScan_file 来扫描这个文件。

```
mdlScan_initScanlist (&scanList);
mdlScan_noRangecheck (&scanList);
mdlOutput_message ( "Scanning file  ....  " );
scanList.scantype        = ELEMTYPE | ONEELEM;
scanList.extendedType    = FILEPOS;
scanList.typmask[0] =TMSK0_LINE | TMSK0_LINE_STRING | TMSK0_SHAPE;
scanList.typmask[0]|=TMSK0_TEXT_NODE | TMSK0_CURVE;
scanList.typmask[0]|=TMSK0_CONIC | TMSK0_CMPLX_SHAPE;
scanList.typmask[0]|=TMSK0_ARC|TMSK0_CMPLX_STRING|TMSK0_ELLIPSE;
scanList.typmask[1]=TMSK1_TEXT | TMSK1_SURFACE | TMSK1_SOLID;
eofPos = mdlElement_getFilePos (FILEPOS_EOF, NULL);
filePos = 0L;          /* start seacrh from top of file */
mdlScan_initialize (0, &scanList);
do
     {
     scanWords = sizeof(elemAddr)/sizeof(short);
     status =mdlScan_file(elemAddr&scanwords,
                          sizeof(elemAddr), &filePos );
     numAddr    = scanWords/sizeof(short);
     for (i=0; i < numAddr; i++)
          {
          if (elemAddr[i] >= eofPos)    break;
```

```
            mdlModify_elementSingle (0, elemAddr[i]
                                     MODIFY_REQUEST_HEADERS,
                                     MODIFY_ORIG,editelem,
                                     &numchanged, 0L);
            }
        } while (status = BUFF_FULL);
```

下面列出完整的 modall.mc 程序。键入 mdl load modall 装入这个应用。键入 MODLEVEL ALL 执行这个应用。

```
/*----------------------------------------------------------------------+
|IncludeFiles                                                           |
+----------------------------------------------------------------------*/
#include <time.h>
#include <stdarg.h>
#include <mdl.h>          /* MDL Library funcs structures & constants */
#include <global.h>
#include <mselems.h>
#include <userfnc.h>
#include <tcb.h>
#include <msrmgr.h>
#include <scanner.h>

#include <msparse.fdf>
#include <msoutput.fdf>
#include <mssystem.fdf>
#include <msstate.fdf>
#include <mslocate.fdf>
#include <mselemen.fdf>
#include <msmisc.fdf>
#include <msselect.fdf>
#include <msscan.fdf>

#include "modall.h"
/*----------------------------------------------------------------------+
| name  edltelem                                                        |
+----------------------------------------------------------------------*/
Private int editelem
(
MSElement    *el
)
    {
    int   actlevel;

    if (mdlElement_getType(el) == CELL_HEADER_ELM)
       return MODIFY_STATUS_NOCHANGE;
    mdlParams_getActive(&actlevel, ACTIVEPARAM_LEVEL);
    mdlElement_setProperties(el,    /* element to change */
                             &actlevel,  /* level */
                             NULL,       /* graphic group number */
                             NULL,       /* class */
                             NULL,       /* locked */
                             NULL,       /* new */
                             NULL,       /*modified */
                             NULL,       /*view independent */
                             NULL);        /* solid hole */
    return MODIFY_STATUS_REPLACE;
    }
/*----------------------------------------------------------------------+
|   name        change_all                                              |
+----------------------------------------------------------------------*/
Private void change_all
```

```
(
void
)
     {
     ULong             elemAddr[50], eofPos, filePos;
     int               scanWords, status, i, numAddr;
     Scanlist          scanList;

     mdlScan_initScanlist(&scanList);
     mdlScan_noRangeCheck(&scanList);
     mdlOutput_message("Scanning file....");

     scanList.scantype        = ELEMTYPE|ONEELEM;
     scanList.extendedType    = FILEPOS;
     scanList.typmask[0] = TMSK0_LINE | TMSK0_LINE_STRING | TMSK0_SHAPE;
     scanList.typmask[0]|= TMSK0_TEXT_NODE|TMSK0_CURVE;
     scanList.typmask[0]|= TMSK0_CONIC|TMSK0_CMPLX_SHAPE;
     scanList.typmask[0]|= TMSK0_ARC|TMSK0_CMPLX_STRING|TMSK0_ELLIPSE;
     scanList.typmask[1] = TMSK1_TEXT|TMSK1_SURFACE|TMSK1_SOLID;

     eofPos  = mdlElement_getFilePos (FILEPOS_EOF, NULL);
     mdlScan_initialize (0, &scanList);
     filePos=0L;

     /* loop through all text elements in file */
     do
        {
        scanWords = sizeof(elemAddr)/sizeof(short);
        status    = mdlScan_file (elemAddr, &scanWords,
                                  sizeof(elemAddr), &filePos);
        numAddr   = scanWords / sizeof(short);
        for (i=0; i<numAddr; i++)
             {
             if (elemAddr[i] >= eofPos)
                break;
              mdlModify_elementSingle (0,elemAddr[i],
                                       MODIFY_REQUEST_HEADERS,
                                       MODIFY_ORIG,editelem,NULL,0L);
             }
         } while (status == BUFF_FULL);
     mdlOutput_prompt("");
     }
/*----------------------------------------------------------------------+
|     name modall                                                       |
+----------------------------------------------------------------------*/
cmdName void modall
(
void
)
cmdNumber  CMD_MODIFY_LEVEL_ALL
     {
     change_all();
     }
/*----------------------------------------------------------------------+
| name   main                                                           |
| author Chaohua Lin                 8/04/2006                          |
+----------------------------------------------------------------------*/
Public int main
(
int                argc,
char              *argv[]
```

```
)
     {
     RscFileHandle   rfHandle;

     if (mdlParse_loadCommandTable(NULL)== NULL)
        mdlOutput_error("Unable to load command table.");
     mdlResource_openFile(&rfHandle, NULL, FALSE);
     return SUCCESS;
     }
```

文件 modall.r 含有命令表及提示信息。

```
/*----------------------------------------------------------------------+
|    modalI.r - command table and messages for modall.mc               |
+----------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"

#define        CT_NONE        0
#define        CT_MODIFY      1
#define        CT_LEVEL       2
#define        CT_ALL         3

Table  CT_MODIFY =
{
       { 1, CT_LEVEL,    MANIPULATION,     REQ,        "MODIFY"};
};
Table  CT LEVEL =
{
       { 1, CT_ALL,     INHERIT,      REQ,          "LEVEL"},
}
Table  CT ALL =
{
      { 1, CT_NONE,     INHERIT,    NONE,          "ALL"},
};
```

文件 modallmsg.r 提示信息。

```
/*----------------------------------------------------------------------+
|    Include Files                                                     |
+----------------------------------------------------------------------*/
#include <rscdefs.h>

#define  MESSAGELISTID_Commands       0
#define  MESSAGELISTID_Prompts        1

MessageList MESSAGELISTID_Commands =
{
     {
     {1, "Modify Element Level"},
     }
};

MessageList MESSAGELISTID_Prompts =
{
     {
     {1,  "Enter first point"},
     {2,  "Enter next point"},
     {3,  "Accept/Reject element"},
     {4,   "Modify All Element Level" },
     }
};
```

键入 bmake -a modall 建立 MODALL 应用。

```
#--------------------------------------------------------------------
# name   modall.mke
# author Chaohua Lin       8/04/2006
#--------------------------------------------------------------------
%if defined (_MakeFilePath)
BaseDir    = $(_MakeFilePath)
%else
BaseDir    = $(MS)/mdl/examples/modall/
%endif

appName     = modall
privateInc  = $(BaseDir)

#--------------------------------------------------------------------
# mdl.mki contains the default rules for creating .rsc, .mo, etc files
#--------------------------------------------------------------------
%include   $(MS)/mdl/include/mdl.mki

#--------------------------------------------------------------------
# Define macros for files included in our link and resource merge
#--------------------------------------------------------------------
lchObjs = \
    $(o)$(appName).mo

lchRscs = \
    $(o)$(appName).rsc \
    $(rscObjects)$(appName)msg.rsc \
    $(o)$(appName).mp

#--------------------------------------------------------------------
#       Generate command table include & resource file
#--------------------------------------------------------------------
$(genSrc)$(appName).h            : $(BaseDir)$(appName).r

$(o)$(appName).rsc               : $(BaseDir)$(appName).r

#--------------------------------------------------------------------
#       Compile the MDL source object file
#--------------------------------------------------------------------
$(o)$(appName).mo                : $(BaseDir)$(appName).mc

#--------------------------------------------------------------------
#       Builds any translatable resource modules for the application.
#--------------------------------------------------------------------
$(rscObjects)$(appName)msg.rsc   : $(BaseDir)$(appName)msg.r

#--------------------------------------------------------------------
#  The following section generates the MDL Program module using
#  mlink. This module should contain ALL CODE resources and/or
#  libraries used by the application.
#--------------------------------------------------------------------
$(o)$(appName).mp      : $(lchObjs)
           $(msg)
        > $(o)temp.cmd
        -a$@
        -s6000
        $(linkOpts)
        $(lchObjs)
        <
        $(MLinkCmd) @$(o)temp.cmd
        ~time
```

```
#--------------------------------------------------------------------
#    The final step in building the application is lib'ing the applications
#    intermediate application with the translatable resources built in
#    this makefile. This step generates the final, and possibly translated,
#    MDL application.
#--------------------------------------------------------------------
$(mdlapps)$(appName).ma   : $(lchRscs)
     $(msg)
     > $(o)make.opt
     -o$@
     $(lchRscs)
     <
     $(RLibCmd) @$(o)make.opt
     ~time
```

4.4 围栅搜寻

在应用 MODSYMB 中，我们使用了所有的搜寻方式，即对单个元素、对整个文件及围栅包围部分的搜寻。我们也介绍了元素操作的范围。我们的应用可以改变元素的层、颜色、线宽及线型。

我们也使用 mode 变量告诉应用要修改什么。例如，当我们键人命令 MODIFY WEIGHTSINGLE 时，mode 将被设为 5。mode 用于对信息表进行索引并显示适当的提 示，它还用于在 editelem 中修改层和线符。

```
cmdName void modfence_weight
(
void
)
cmdNumber        CMD_FENCE_MODIFY_WEIGHT
 {
 setSearchType();
 mode = WEIGHT;
 mdlState_startFenceCommand(changeFenceContents,
         NULL,                    /* function to define fence outline */
         NULL,                    /* function for DATA point */
         modfence_weight,         /* function for RESET */
         6,                       /* message for command name */
         mode);                   /* prompt for fence */
  }
```

这个程序的大部分由我们先前的例子中派生出来，只是先前的例子没有围栅搜寻。函数 mdlState_startFenceCommand 为搜寻满足围栅搜寻标准的元素调用用户函数 changeFenceContents。函数 changeFenceContents 为由围栅返回的每一个元素调用 mdlModify_elementSingle。如果围栅内发现一个复杂元素，则它把这个复杂元素头及它的所有组成元素传递给 mdlModify_elementSingle。从此，按照 MODIFY_REQUEST_HEADERS 的规定，为操作每一个组成元素及元素头调用 editelem。

```
Private int changeFenceContents
(
)
     {
     ULong            filePos;
     DgnModelRefP     currFile;

      filePos    = mdlElement_getFilePos (FILEPOS_CURRENT, &currFile);
      mdlModify_elementSingle (currFile,
          filePos,                    /* file position for element */
          MODIFY_REQUEST_HEADERS,     /* process complex headers */
          MODIFY_ORIG,                /* modify original element */
          editelem,                   /* modify routine for each element */
          &numchanged,                /* parameters for editelem */
          FALSE);                     /* offset for component elements */
     return SUCCESS;
     }
```

函数 editelem 于先前的版本稍有不同。我们加进了对修改方式的测试并用 mdlElement_setSymbology 来进行改变。

```
switch (mode)
{
 case LEVEL:
       if (mdlElemet_getType(el) = CELL_HEADER_ELM)
          return (MODIFY_STATUS_NOCHANGE)
       mdlElement_setProperties (el, &actlevel, NULL, NULL, NULL,
                                 NULL,  NULL,  NULL,  NULL);
       break;
 case COLOR:
       mdlElement_setSymbology(el, &actcolor, NULL, NULL);
       break;
 case WEIGHT:
       mdlElement_setSymbology(el, NULL, &actweight, NULL);
       break;
 case STYLE:
       mdlElement_setsymbology(el, NULL, NULL, &actstyle);
       break;
}
```

下面列出完整的 modsymb.mc 程序。键入 mdl load modsymb 装入这个应用。

```
/*----------------------------------------------------------------------+
|IncludeFiles                                                           |
+----------------------------------------------------------------------*/
#include <time.h>
#include <stdarg.h>
#include <mdl.h>         /* MDL Library funcs structures & constants */
#include <global.h>
#include <mselems.h>
#include <userfnc.h>
#include <tcb.h>
#include <msrmgr.h>
#include <scanner.h>

#include <msparse.fdf>
#include <msoutput.fdf>
#include <mssystem.fdf>
#include <msstate.fdf>
#include <mslocate.fdf>
#include <mselemen.fdf>
#include <msmisc.fdf>
#include <msselect.fdf>
#include <msscan.fdf>

#include "modsymb.h"

#define   LEVEL   2
#define   STYLE   3
#define   COLOR   4
#define   WEIGHT  5
int        mode;
/*----------------------------------------------------------------------+
|  name edltelem                                                        |
+----------------------------------------------------------------------*/
Private int editelem
(
MSElement   *el
)
    {
```

```
    int      actstyle;
    ULong    actlevel,actcolor,actweight;

    mdlParams_getActive(&actlevel, ACTIVEPARAM_LEVEL);
    mdlParams_getActive(&actcolor, ACTIVEPARAM_COLOR);
    mdlParams_getActive(&actweight, ACTIVEPARAM_LINEWEIGHT);
    mdlParams_getActive(&actstyle, ACTIVEPARAM_LINESTYLE);
    switch (mode)
            {
            case LEVEL:
                 if (mdlElement_getType(el) == CELL_HEADER_ELM)
                    return MODIFY_STATUS_NOCHANGE;
                 mdlElement_setProperties(el,&actlevel,NULL,NULL,NULL,
                                          NULL,NULL,NULL,NULL);
                 break;
            case COLOR:
                 mdlElement_setSymbology(el,&actcolor,NULL,NULL);
                 break;
            case WEIGHT:
                 mdlElement_setSymbology(el,NULL,&actweight,NULL);
                 break;
            case STYLE:
                 mdlElement_setSymbology(el,NULL,NULL,&actstyle);
                 break;
            }
    return MODIFY_STATUS_REPLACE;
    }
/*----------------------------------------------------------------------+
|  name change_all                                                      |
+----------------------------------------------------------------------*/
Private void change_all
(
void
)
    {
    ULong             elemAddr[50], eofPos, filePos;
    int               scanWords, status, i, numAddr;
    Scanlist          scanList;

    mdlScan_initScanlist(&scanList);
    mdlScan_noRangeCheck(&scanList);
    mdlOutput_message("Scanning file....");

    scanList.scantype        = ELEMTYPE|ONEELEM;
    scanList.extendedType    = FILEPOS;
    scanList.typmask[0] = TMSK0_LINE|TMSK0_LINE_STRING|TMSK0_SHAPE;
    scanList.typmask[0]|= TMSK0_TEXT_NODE|TMSK0_CURVE;
    scanList.typmask[0]|= TMSK0_CONIC|TMSK0_CMPLX_SHAPE;
    scanList.typmask[0]|= TMSK0_ARC|TMSK0_CMPLX_STRING|TMSK0_ELLIPSE;
    scanList.typmask[1] = TMSK1_TEXT|TMSK1_SURFACE|TMSK1_SOLID;

    eofPos  = mdlElement_getFilePos (FILEPOS_EOF, NULL);
    mdlScan_initialize (0, &scanList);
    filePos=0L;

    /* loop through all text elements in file */
    do
        {
        scanWords = sizeof(elemAddr)/sizeof(short);
        status   = mdlScan_file (elemAddr, &scanWords,
                                 sizeof(elemAddr), &filePos);
       numAddr    = scanWords / sizeof(short);
```

```
        for (i=0; i<numAddr; i++)
             {
             if (elemAddr[i] >= eofPos)
                 break;
              mdlModify_elementSingle(0,elemAddr[i],
                                      MODIFY_REQUEST_HEADERS,
                                      MODIFY_ORIG,editelem,NULL,0L);
              }
        } while (status == BUFF_FULL);
     mdlOutput_prompt("");
     }

/*----------------------------------------------------------------------+
|   name setSearchType                                                  |
+----------------------------------------------------------------------*/
Private void setsearchType
(
void
)
     {
     static int searchType[]={CELL_HEADER_ELM,LINE_ELM,
                              LINE_STRING_ELM,SHAPE_ELM,
                              TEXT_NODE_ELM,CURVE_ELM,
                              CMPLX_STRING_ELM,
                              CONIC_ELM,CMPLX_SHAPE_ELM,
                              ELLIPSE_ELM,ARC_ELM,
                              TEXT_ELM,SURFACE_ELM,SOLID_ELM};
     /*initialize search creteria to find nothing*/
     mdlLocate_noElemNoLocked();
     /*add elements to search to list*/
     mdlLocate_setElemSearchMask(sizeof(searchType)/sizeof(int),
                                 searchType);
     }

/*----------------------------------------------------------------------+
|    name mod_accept                                                    |
+----------------------------------------------------------------------*/
Private  void  mod_accept
(
void
)
     {
     ULong             filePos;
     DgnModelRefP     currFile;

     filePos = mdlElement_getFilePos(FILEPOS_CURRENT,&currFile);
     mdlModify_elementMulti(currFile,     /* file to process */
                    filePos,      /* file position for element */
                    MODIFY_REQUEST_HEADERS,/*process complex headers */
                    MODIFY_ORIG, /*modify original element */
                    editelem, /*modify routine for each element */
                    NULL,         /* parameters for editelem */
                    TRUE);        /* process graphic group */
     /*restart the element location process*/
     mdlLocate_restart(FALSE);
     }


/*----------------------------------------------------------------------+
| name changeFenceContents                                              |
+----------------------------------------------------------------------*/
```

```
Private int changeFenceContents
(
void
)
     {
     ULong             filePos;
     DgnModelRefP    currFile;

     filePos = mdlElement_getFilePos(FILEPOS_CURRENT,&currFile);
     mdlModify_elementSingle (currFile,
                filePos,               /* file position for element */
                MODIFY_REQUEST_HEADERS,/* process complex headers */
                MODIFY_ORIG,          /* modify original element */
                editelem,            /* modify rout/ne for each element */
                NULL,               /* parameters for editelem */
                0L);                /* offset for component elements */
     return SUCCESS;
     }
/*----------------------------------------------------------------------+
|name   change single                                                   |
+----------------------------------------------------------------------*/
Private void change_single
(
void
)
     {
     setsearchType();
     mdlState_startModifyCommand(
                    change_single,
                    mod_accept, /* function to call on DATA */
                    NULL,      /* function to call for DYNAMIC */
                    NULL,      /* function to call on SHOW */
                    NULL,      /* function to call on CLEAN */
                    1,          /* index into MESSAGE LIST */
                    3,          /* index into PROMPT LIST */
                    TRUE,      /* Modify SELECTION SET ? */
                    FALSE);    /* additional data points required*/
     mdlLocate_init();
     }
/*----------------------------------------------------------------------+
|name     modsing_level                                                 |
+----------------------------------------------------------------------*/
cmdName void modsing_level
(
)
cmdNumber CMD_MODIFY_LEVEL_SINGLE
     {
     mode=LEVEL;
     change_single();
     }
/*----------------------------------------------------------------------+
|name    modsing_color                                                  |
+----------------------------------------------------------------------*/
cmdName void modsing_color
(
)
cmdNumber CMD_MODIFY_COLOR_SINGLE
     {
     mode=COLOR;
     change_single();
     }
```

```
/*-------------------------------------------------------------------+
|name    modsing_weight                                              |
+-------------------------------------------------------------------*/
cmdName void modsing_weight
(
)
cmdNumber CMD_MODIFY_WEIGHT_SINGLE
     {
     mode=WEIGHT;
     change_single();
     }
/*-------------------------------------------------------------------+
|name    modsing_style                                               |
+-------------------------------------------------------------------*/
cmdName void modsing_style
(
)
cmdNumber CMD_MODIFY_STYLE_SINGLE
     {
     mode=STYLE;
     change_single();
     }
/*-------------------------------------------------------------------+
|name    modall_level                                                |
+-------------------------------------------------------------------*/
cmdName void modall_level
(
)
cmdNumber CMD_MODIFY_LEVEL_ALL
     {
     mode=LEVEL;
     change_all();
     }
/*-------------------------------------------------------------------+
|name    modall_color                                                |
+-------------------------------------------------------------------*/
cmdName void modall_color
(
)
cmdNumber CMD_MODIFY_COLOR_ALL
     {
     mode=COLOR;
     change_all();
     }
/*-------------------------------------------------------------------+
|name    modall_weight                                               |
+-------------------------------------------------------------------*/
cmdName void modall_weight
(
)
cmdNumber CMD_MODIFY_WEIGHT_ALL
     {
     mode=WEIGHT;
     change_all();
     }
/*-------------------------------------------------------------------+
|name    modall_style                                                |
+-------------------------------------------------------------------*/
cmdName void modall_style
(
)
cmdNumber CMD_MODIFY_STYLE_ALL
```

```
     {
     mode=STYLE;
     change_all();
     }
/*-------------------------------------------------------------------+
|name     modfence_level                                             |
 +-------------------------------------------------------------------*/
cmdName void modfence_level
(
)
cmdNumber CMD_FENCE_MODIFY_LEVEL
     {
     setsearchType();
     mode=LEVEL;
     mdlState_startFenceCommand(changeFenceContents,
                         NULL,        /*function to define fence outline*/
                         NULL,        /*function for DATA point*/
                         modfence_level, /*function for RESET*/
                         1,           /*message for command name*/
                         4,           /*prompt for fence*/
                         FENCE_NO_CLIP);
     }
/*-------------------------------------------------------------------+
|name     modfence_color                                             |
+-------------------------------------------------------------------*/
cmdName void modfence_color
(
)
cmdNumber CMD_FENCE_MODIFY_COLOR
     {
     setsearchType();
     mode=COLOR;
     mdlState_startFenceCommand(changeFenceContents,
                         NULL,          /*function to define fence outline */
                         NULL,          /*function tor DATA point*/
                         modfence_color,  /*function for RESET*/
                         1,             /*message for command name*/
                         4,             /*prompt tor fence*/
                         FENCE_NO_CLIP);
     }
/*-------------------------------------------------------------------+
|name     modfence_weight                                            |
+-------------------------------------------------------------------*/
cmdName void modfence_weight
(
)
cmdNumber CMD_FENCE_MODIFY_WEIGHT
     {
     setsearchType();
     mode=WEIGHT;
     mdlState_startFenceCommand(changeFenceContents,
                         NULL,          /*function to define fence outline*/
                         NULL,          /*function tor DATA point*/
                         modfence_weight, /*function for RESET*/
                         1,              /*message for command name*/
                         4,              /*prompt tor fence*/
                         FENCE_NO_CLIP);
     }
/*-------------------------------------------------------------------+
|name     modfence_style                                             |
+-------------------------------------------------------------------*/
cmdName void modfence_style
```

```
(
)
cmdNumber CMD_FENCE_MODIFY_STYLE
     {
     setsearchType();
     mdlState_startFenceCommand(changeFenceContents,
                    NULL,     /*function to define fence outline*/
                    NULL,     /*function for DATA point*/
                    modfence_style, /*function for RESET*/
                    1,              /*message for command name*/
                    4,              /*prompt tor fence*/
                    FENCE_NO_CLIP);
     }

/*----------------------------------------------------------------------+
| name   main                                                           |
| author Chaohua Lin                 8/04/2006                          |
+----------------------------------------------------------------------*/
Public int main
(
int                argc,
char              *argv[]
)
     {
     RscFileHandle   rfHandle;

     if (mdlParse_loadCommandTable(NULL)== NULL)
        mdlOutput_error("Unable to load command table.");
     mdlResource_openFile(&rfHandle, NULL, FALSE);
     mdlState_registerStringIds(0,1);
     return SUCCESS;
     }
```

文件 modsymb.r 含有命令表。

```
/*----------------------------------------------------------------------+
| modsymb.r  -  command table for modsymb.mc                            |
+----------------------------------------------------------------------*/
#include   "rscdefs.h"
#include   "cmdclass.h"

#define      CT_NONE     0
#define      CT_MODIFY   1
#define      CT_MODOPT   2
#define      CT_FENCE    3
#define      CT_ALL      4
#define      CT_OPTION   5

Table  CT_MODIFY =
{
        {1, CT_MODOPT,   MANIPULATION,   REQ,   "MODIFY"},
        {2, CT_FENCE,    FENCE,           REQ,   "FENCE"},
};

Table CT_MODOPT =
{
        {1, CT_ALL,   INHERIT, NONE, "LEVEL"},
        {2, CT_ALL,   INHERIT, NONE, "STYLE"},
        {3, CT_ALL,   INHERIT, NONE, "WEIGHT"},
        {4, CT_ALL,   INHERIT, NONE, "COLOR"},
};

Table CT_FENCE =
```

```
{
        {1, CT_OPTION, INHERIT, REQ,   "MODIFY"},
};

Table CT_ALL =
{
        {1, CT_NONE,    INHERIT,   DEF,   "SINGLE"},
        {2, CT_NONE,    INHERIT,   NONE, "ALL"},
};

Table  CT_OPTION =
{
        { 1, CT_NONE,   INHERIT,      REQ,       "LEVEL" },
        { 2, CT_NONE,   INHERIT,      REQ,       "STYLE" },
        { 3, CT_NONE,   INHERIT,      REQ,       "WEIGHT" },
        { 4, CT_NONE,   INHERIT,      REQ,       "COLOR" },
};
```

文件 modsymbmsg.r **含有提示信息。**

```
/*----------------------------------------------------------------------+
|    Include Files                                                      |
+----------------------------------------------------------------------*/
#include <rscdefs.h>

#define  MESSAGELISTID_Commands       0
#define  MESSAGELISTID_Prompts        1
#define  MESSAGELISTID_Msgs     2

MessageList MESSAGELISTID_Commands =
{
     {
     {1, "Modify Element Properties"},
     }
};

MessageList MESSAGELISTID_Prompts =
{
     {
     {1,  "Enter first point"},
     {2,  "Enter next point"},
     {3,  "Accept/Reject element"},
     {4,  "Accept/Reject FenceContents"},
     }
};

MessageList MESSAGELISTID_Msgs =
{
     {
     {1,   "MDL Place Box"},
     {2,   "Define start point"},
     {3,    "Accept/Reject element"},
     {4,   "Modify All Element Level" },

     }
};
```

键入 bmake -a modsymb 建立 MODSYMB 应用。

```
#--------------------------------------------------------------------
# name   modsymb.mke
# author Chaohua Lin        8/04/2006
#--------------------------------------------------------------------
%if defined (_MakeFilePath)
```

```
BaseDir     = $(_MakeFilePath)
%else
BaseDir     = $(MS)/mdl/examples/modsymb/
%endif

appName      = modsymb
privateInc   = $(BaseDir)

#-------------------------------------------------------------------------
# mdl.mki contains the default rules for creating .rsc, .mo, etc files
#-------------------------------------------------------------------------
%include    $(MS)/mdl/include/mdl.mki

#-------------------------------------------------------------------------
# Define macros for files included in our link and resource merge
#-------------------------------------------------------------------------
lchObjs = \
     $(o)$(appName).mo

lchRscs = \
     $(o)$(appName).rsc \
     $(rscObjects)$(appName)msg.rsc \
     $(o)$(appName).mp

#-------------------------------------------------------------------------
#        Generate command table include & resource file
#-------------------------------------------------------------------------
$(genSrc)$(appName).h            : $(BaseDir)$(appName).r

$(o)$(appName).rsc               : $(BaseDir)$(appName).r

#-------------------------------------------------------------------------
#        Compile the MDL source object file
#-------------------------------------------------------------------------
$(o)$(appName).mo                : $(BaseDir)$(appName).mc

#-------------------------------------------------------------------------
#        Builds any translatable resource modules for the application.
#-------------------------------------------------------------------------
$(rscObjects)$(appName)msg.rsc   : $(BaseDir)$(appName)msg.r

#-------------------------------------------------------------------------
#  The following section generates the MDL Program module using
#  mlink. This module should contain ALL CODE resources and/or
#  libraries used by the application.
#-------------------------------------------------------------------------
$(o)$(appName).mp      : $(lchObjs)
         $(msg)
        > $(o)temp.cmd
        -a$@
        -s6000
        $(linkOpts)
        $(lchObjs)
        <
        $(MLinkCmd) @$(o)temp.cmd
        ~time

#-------------------------------------------------------------------------
#   The final step in building the application is lib'ing the applications
#   intermediate application with the translatable resources built in
#   this makefile. This step generates the final, and possibly translated,
#   MDL application.
```

```
#-----------------------------------------------------------------------
$(mdlapps)$(appName).ma   : $(lchRscs)
     $(msg)
     > $(o)make.opt
     -o$@
     $(lchRscs)
     <
     $(RLibCmd) @$(o)make.opt
     ~time
```

第五章　元素描述符

从程序中操作元素总是很繁琐的。这需要对设计文件的工作状态十分清楚。例如，在UCM(注：UCM是User Command的英文缩写，即用户命令的意思。MicroStation V8_2004 已经不再支持该旧式的二次开发语言。)中要将元素读入DGNBUF，进行修改，再将这个元素写入设计文件中。由于这一操作绕过显示处理器，所以只有屏幕更新后才会显示其结果。在这一层上读及写元素要求管理文件指针，并且错误的计算结果会破坏文件。

由于 DGNBUF 同一时间只能放一个元素，因而对于复杂元素上述问题的处理就显得更为复杂。就是说要求每次把一个复杂头及其组成元素读入 DGNBUF 并进行适当的处理。如将一个单元从一层移到另一层的问题，首先要依据单元头计算出组成元素的位置，然后检索并修改组成元素的层，最后回到这个单元头，修改单元层位掩码以便反映新的层。

MDL 提供了一系列称为元素描述符的函数。这些函数使得用户可以在不必管理复杂头的情况下对组成元素进行处理。元素描述符不只限用于复杂元素，也可用于简单元素或简单元素表。基本特点是这些元素描述符在内存中操作，并且 MDL 可以通过诸如 mdlElmdscr_validate 这样的函数保证数据的完整性。程序员调用 MDL 函数分配内存，所以必须用 mdlElmdscr_freeAll 函数释放内存。

5.1 元素描述符函数

元素描述符函数汇总如下：

mdlElmdscr_add	把描述符所表达的元素添加到设计文件中
mdlElmdscr_addToChain	在另一元素描述符中追加描述符
mdlElmdscr_append	把描述符中的元素追加到文件中
mdlElmdscr_appendDscr	把描述符加到另一元素描述符中(用于单个复杂元素)
mdlElmdscr_appendElement	在描述符中追加元素
mdlElmdscr_convertTo2D	将 3D 描述符转换为 2D 描述符
mdlElmdscr_convertTo3D	将 2D 描述符转换为 3D 描述符
mdlElmdscr_display	在所有的视图中显示描述符
mdlElmdscr_displayFromFile	在所有的视图中显示描述符，它不需要事先将元素读入一个元素描述符，而是直接从设计文件中显示。
mdlElmdscr_duplicate	拷贝一个元素描述符
mdlElmdscr_freeAll	释放用于存放元素描述符的内存
mdlElmdscr_markElement	为后续处理给描述符中的元素做标志
mdlElmdscr_new	从已有的元素生成一个新的描述符
mdlElmdscr_igdsSize	返回描述符的长度
mdlElmdscr_insertElement	将已有的元素插入描述符
mdlElmdscr_operation	为描述符中的每一元素调用一次指定的用户函数
mdlElmdscr_read	从设计文件中读元素并生成新的描述符
mdlElmdscr_removeElement	从描述符中删除元素
mdlElmdscr_replaceElement	替换描述符中的元素
mdlElmdscr_rewrite	重写一个已有的元素描述符

mdlElmdscr_stroke	将元素描述符分解为向量
mdlElmdscr_transform	用变换矩阵变换描述符
mdlElmdscr_undoableDelete	删除设计文件中描述符内的元素，UNDO 可恢复所删除的元素
mdlElmdscr_validate	从描述符更新复杂头

5.1.1 元素描述符结构

元素描述符是用 MSElementDescr 结构定义的，可在<mselems.h>中找到这个结构。

```
typedef struct msElementDescr  MSElementDescr;
struct msElementDescr
     {
     struct
         {
         MSElementDescr  *next      /* ptr to first entry in list */
         MSElementDescr  *previous; /* ptr to last entry in list */
         MSElementDescr  *myHeader; /* ptr to my hdr (NULL = not cmplx) */
         MSElementDescr  *firstElem; /* ptr to first element if header */
         int             isHeader;   /* is this a complex header */
         int             isValid;    /* INTERNAL USE ONLY */
         long            userData1;  /* available for user */
         long            userData2;  /* available for user */
         } h;
MSElement                el;         /* elem data (hdr only if complex) */
};
```

使用递归程序 elemDscr_show 可以访问描述符中的元素链接表。这个程序在调试元素描述符的内容时非常有用。

```
Public int    elemDscr_show
(
MSElementDescr *elemDescr,
char           *currentIndent
)
{
    char    indent[128];
    UInt32  color, weight, level, ggNum;
    int     style, class, locked, new, modified, viewIndepend, solidHole;
    if (elemDescr == NULL)
       return SUCCESS;
    strcpy (indent, currentIndent);
    strcat(indent, "|    ");
    do
```

```
    {
       mdlElement_getSymbology (&color, &weight, &style, &elemDescr->el);
       mdlElement_getProperties (&level, &ggNum, &class, &locked, &new,
                         &modified, &viewIndepend, &solidHole, &elemDescr->el);
       printf ("%shdr=%d, typ=%d, cmplx=-%d", currentIndent,
                         elemDescr->h.isHeader, elemDescr->el.hdr.ehdr.type,
                         elemDescr->el.hdr.ehdr.complex);
       printf (" , c=%d, w=%d, s=%d, l=%d, gg=%d, cl=%d\n",
                     color, weight, style, level, ggNum, class);
       if (elemDescr->h.isHeader)
       {
              elemDscr_show (elemDescr->h.firstElem, indent);
              printf ( "%send of chain\n", currentIndent);
       }
       elemDescr = elemDescr->h.next;
    } while (elemDescr);
    return SUCCESS;
}
```

5.1.2 元素描述符的生成

现在我们将使用元素描述符重写 PLBOX 应用,这个应用用于放置复杂元素。该程序除了 generateImage 函数之外，基本上与 PLBOX 相同。

第一步是通过调用 mdlElmdscr_new 开始新的描述符。第二步则使用 mdlElmdscr_appendElement 逐步地建立一个复杂元素。

```
  /* create new descriptor using cell header */
  mdlCell_create (&el, "pbdscr", &origin, FALSE);
  mdlElmdscr_new (&elmDP, NULL, &el);
  mdlText_create (&el, NULL, textin, &pntP[1], NULL, NULL, NULL, NULL);
  mdlElmdscr_appendElement (elmDP, &el);
  mdlText_extractShape (shapeP, NULL, &el, TRUE, view);
  mdlShape_create (&el, NULL, shapeP, 5, -1);
  mdlElmdscr_appendElement (elmDP, &el);
```

最后，我们用 mdlElmdscr_add 把内存中的元素描述符写入设计文件。

```
  if (drawMode == NORMALDRAW)
      mdlElmdscr_add (elmDP);
```

记住用 mdlElmdscr_freeAll 清理分配的内存。

```
  mdlElmdscr_freeAll ( &elmDP);
```

PBDSCR 应用的修改过的 PLBOX 如下所示。装入应用键入 mdl l pbdscr, 运行应用程序键入 PLACE DBOX。键入 PLACE DBOX 后在命令键入域随意键入一个字串即会动态显示在方框中。

```
/*----------------------------------------------------------------------+
|   Copyright (C) 1991 Mach N. Dinh-Vu, All Rights Reserved             |
```

```
|   Program    : pbdscr.mc                                                    |
|   Revision   : 1. 0.a                                                       |
|   UpgradeToV8: MicroStationFan 2006/05                                      |
+-----------------------------------------------------------------------------+
|Example MDL function to place box around a text string with a leader         |
|line using element descriptors.                                             |
+-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------+
|    Include Files                                                            |
+-----------------------------------------------------------------------------*/
#include <mdl.h>         /* system include files */
#include <global.h>
#include <msrmgr.h>
#include <userfnc.h>
#include <stdlib/string.h>
#include <mscell.fdf>
#include <mselemen.fdf>
#include <mselmdsc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msstate.fdf>
#include "pbdscr.h"

/*-----------------------------------------------------------------------------+
|    Private Global variables                                                 |
+-----------------------------------------------------------------------------*/
static char textin[128];
Dpoint3d    pntP[2];

/*-----------------------------------------------------------------------------+
|    name        main                                                         |
+-----------------------------------------------------------------------------*/
Private void main(void)
{
   RscFileHandle   rfHandle;

   /* load our command table */
   if (mdlParse_loadCommandTable (NULL) == NULL)
       mdlOutput_error ("Unable to load command table. ");
   mdlResource_openFile (&rfHandle, NULL, FALSE);
   mdlOutput_prompt ("Key-in PLACE DBOX to execute");
 }
/*-----------------------------------------------------------------------------+
|   name          generateImage - dynamic function for box.                   |
+-----------------------------------------------------------------------------*/
Private int    generateImage
(
 Dpoint3d     *pt,
 int          view,
 int      drawMode
)
{
   MSElementDescr  *elmDP;
   MSElementUnion  el;
   Dpoint3d        origin;
   Dpoint3d        shapeP[5];

   pntP[1] = origin = *pt;
   /* create new descriptor using cell header */
   mdlCell_create (&el, L"pbdscr", &origin, FALSE);
   mdlElmdscr_new (&elmDP, NULL, &el);
   mdlText_create (&el, NULL, textin, &pntP[1], NULL, NULL, NULL, NULL);
```

```
mdlElmdscr_appendElement (elmDP, &el);
mdlText_extractShape (shapeP, NULL, &el, TRUE, view);
mdlShape_create (&el, NULL, shapeP, 5, -1);
mdlElmdscr_appendElement (elmDP, &el);
if (pntP[0].x < origin.x)
{
   if (pntP[0].y  < origin.y)
   {
      pntP[1].x = shapeP[0].x;
      pntP[1].y = shapeP[0].y;
   } else {
      pntP[1].x = shapeP[3].x;
      pntP[1].y = shapeP[3].y;
   }
} else {
   if (pntP[0].y < origin.y)
   {
      pntP[1].x = shapeP[1].x;
      pntP[1].y = shapeP[1].y;
   } else {
      pntP[1].x = shapeP[2].x;
      pntP[1].y = shapeP[2].y;
   }
}
// Place the modified leader line
mdlLine_create (&el, NULL, pntP);
mdlElmdscr_appendElement (elmDP, &el);
mdlElmdscr_display (elmDP, 0, drawMode);
if (drawMode == NORMALDRAW)
   mdlElmdscr_add (elmDP);
mdlElmdscr_freeAll (&elmDP);
   return SUCCESS;
}
/*----------------------------------------------------------------------------+
 |   name         keyinText                                                   |
 +----------------------------------------------------------------------------*/
Private void  keyinText (char *cmdStrP)
{
   if (!*statedata.cmdstring)
      return;
   strncpy (textin, statedata.cmdstring, sizeof(textin));
}
/*----------------------------------------------------------------------------+
 |   name         placeBox_done                                               |
 +----------------------------------------------------------------------------*/
Private void   placeBox_done (void)
{
   mdlOutput_rscPrintf (MSG_PROMPT, NULL, 0, 4);
}
/*----------------------------------------------------------------------------+
 |   name         placeBox_secondPoint                                        |
 +----------------------------------------------------------------------------*/
Private void   placeBox_secondPoint
(
Dpoint3d  *pt,
int       view
)
{
   generateImage (pt, view, NORMALDRAW);
}
/*----------------------------------------------------------------------------+
 |    name      placeBox_firstPoint                                           |
```

```
+--------------------------------------------------------------------------*/
Private void    placeBox_firstPoint
(
Dpoint3d    *pt,
int         view
)
{
   /* save first point */
   pntP[0] = *pt;

   /* Set the datapoint state function for the second point. */
   mdlState_setFunction (STATE_KEYIN, keyinText);
   mdlState_setFunction (STATE_DATAPOINT, placeBox_secondPoint);
   mdlState_setFunction (STATE_RESET, placeBox_done);
   mdlOutput_rscPrintf (MSG_PROMPT, NULL, 0, 3);
   /* Setup Rubber Banding function  */
   mdlState_setFunction (STATE_COMPLEX_DYNAMICS, generateImage);
}
/*--------------------------------------------------------------------------+
 |   name         placeBox_start                                            |
+--------------------------------------------------------------------------*/
cmdName  placeBox_start (void)
cmdNumber  CMD_PLACE_DBOX
{
  mdlState_startPrimitive (placeBox_firstPoint, placeBox_start, 1, 2);
  return 0;
}
```

pbdscr.r 文件中含有命令表和提示信息。源程序如下：

```
/*--------------------------------------------------------------------------+
 |     PBDSCR.R command table and messages for PBDSCR.MC                    |
+--------------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"

#define    CT_NONE       0
#define    CT_PLACE      2

Table   1 =
{
     { 1, CT_PLACE,     PLACEMENT,     REQ,     "PLACE" },
};

Table   CT_PLACE =
{
     { 1, CT_NONE,      INHERIT,       NONE | CMDSTR (1),     "DBOX" },
};

MessageList 0 =
{
     {
     { 0, ""  },
     { 1, "MDL Place BOx (descriptors)" },
     { 2, "Define start point"  },
     { 3, "Define Box position" },
     { 4, "PlBox complete" },
     }
  };
```

键入 bmake -a pbdscr 建立 PBDSCR 应用。pbdscr.mke 文件内容如下：

```
#--------------------------------------------------------------------
#       PBDSCR MDL Make File
#--------------------------------------------------------------------
```

```
%include  mdl.mki

#--------------------------------------------------------------------
#       Define constants specific to this PBDSCR application
#--------------------------------------------------------------------
baseDir      = ./
privateInc   = $(baseDir)

pbdscrObjs = $(o)pbdscr.mo
pbdscrRscs = $(o)pbdscr.rsc  $(o)pbdscr.mp

#--------------------------------------------------------------------
# Create needed output directories if they don't exist
#--------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#-----------------------------------------
#       Generate Command Tables
#-----------------------------------------
$(privateInc)pbdscr.h      :   $(baseDir)pbdscr.r

#-----------------------------------------
#       Compile Resources
#-----------------------------------------
$(o)pbdscr.rsc             :   $(baseDir)pbdscr.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)pbdscr.mo              :   $(baseDir)pbdscr.mc
$(o)pbdscr.mp              :   $(pbdscrObjs)
             $(msg)
             >$(o)temp.cmd
             -a$@
             $(linkOpts)
             $(pbdscrObjs)
             <
             $(MLinkCmd) @$(o)temp.cmd
             ~time

#-----------------------------------------
#       Merge Objects into one file
#-----------------------------------------
$(mdlapps)pbdscr.ma        : $(pbdscrRscs)
             $(msg)
             >$(o)temp.cmd
             -o$@
             $(pbdscrRscs)
             <
             $(RLibCmd) @$(o)temp.cmd
             ~time
```

5.2 复杂元素的操作

PBDSCR 应用使用元素描述符为设计文件添加元素。然而，实际上元素描述符真正的威力是对已有的复合元素进行操作。在应用程序STRTEXT中，我们将先扫描整个文件中的文本及文本节点，然后把文本字符串变成线段，并将它们输出到指定的文件中。将文本转换为线后，我们可以用转换软件（如DXFOUT）将文本字符输出到别的CAD系统。

显然对于文本字符串是没有问题的，但是当扫描程序读到一个文本节点时就会将整个复杂元素装入内存。如果

我们对整个描述符进行操作，就不能访问组成元素。函数 mdlElmdscr_operation 根据一个操作旗标，调用用户函数解决这一问题。

以下旗标是有效的：

ELMD_ELEMENT	调用用户函数处理每一个组成元素
ELMD_PRE_HDR	在调用用户函数处理每一个组成元素之前，调用该函数处理最外层的头元素
ELMD_PRE_NESTEDHDR	在调用用户函数处理每一个组成元素之前，调用该函数处理嵌套的头元素
ELMD_POST_HDR	调用用户函数处理每一个组成元素，然后再调用它处理最外层的头元素
ELMD_POST_NESTDHDR	调用用户函数处理每一个组成元素，然后再调用它处理嵌套的头元素
ELMD_ALL_ONCE	任一个组成元素对用户函数只进行一次调用。等价于(ELMD_ELEMENT | ELMD_PRE_HDR | ELMD_PRE_NESTEDHDR)
ELMD_HDRS_ONCE	仅对每个头元素调用一次用户函数。和(ELMD_PRE_HDR | ELMD_PRE_NESTEDHDR)相同。

在我们的应用程序中，我们将调用 stroke_elm 处理每一个组成元素。

```
mdlElmdscr_operation (edP, stroke_elm, NULL, ELMD_ELEMENT);
```

在 stroke_elm 中，我们要检查 mdlElmdscr_operation 传递的元素是否为文本字符串。这也许是多余的，因为只有处理的复杂元素才会是文本节点。那么我们就必须假定组成元素必须是字符串。当这一假定成立时，我们的防错性程序设计将防止因元素格式的改变而改变。stroke_elm 必须向 mdlElmdscr_operation 返回 SUCCESS，否则导致操作失败。

```
if (mdlElement_getType (el) != TEXT_ELM)
   return SUCCESS;
```

向 mdlElement_stroke 传递字符串将返回一个定义文本的点数组。用这些点我们可生成一条线串。因为一条线串最多可以有 101 个点(注：V8 以前如此，V8 以后线串最多点数增加到 5000 个)，因此我们将检验点数是否在这一范围内。

```
if (mdlElement_stroke (&points, &numPnts, el, TOLERANCE) == SUCCESS)
{
  if (numPnts > MAX_VERTICES)
     numPnts = MAX_VERTICES;
```

然后，我们使用字符串的线符和属性从这条线串建立一个新的描述符。

```
/* get element display symbology of current element */
mdlElement_getSymbology (&color, &weight, &style, el);
/* get properties of current element */
mdlElement_getProperties (&level, &ggNum, NULL, NULL,
                          NULL, NULL, NULL, NULL, el);
mdlLineString_create (&elm, NULL, points, numPnts);

 /* set element display symbology of line string */
mdlElement_setSymbology (&elm, &color, &weight, &style);
```

```
 /* set properties of line string */
mdlElement_setProperties (&elm, &level, &ggNum, NULL, NULL,
                          NULL, NULL, NULL, NULL);
```

使用 mdlWorkDgn_write 可以向另一个 DGN 文件输出这条线串，在下一节中我们将讨论 mdlWorkDgn_ 函数。最后，我们使输出字符串发亮，然后释放被这条线串元素描述符占用的内存。

```
mdlEImdscr_new (&elmDP, NULL, &elm);
workPos = mdlWorkDgn_write (elmDP, workPos, NULL);

/ * Display text written * /
mdlElmdscr_display (el, HILITE);

/ * Free temporary descriptor * /
mdlEImdscr_freeAll (&elmDP),
```

5.3 工作文件

在设计文件被打开时，MDL 提供了一系列函数处理临时工作文件。临时工作文件可以是一个设计文件或单元库。

工作文件用 mdlWorkDgn_openFile 打开，用 mdlWorkDgn_ 函数管理。这些函数是从元素描述符读写格式化数据的基本程序。

mdlWorkDgn_closeFile　关闭工作文件
mdlWorkDgn_delete　从工作文件中删除元素描述符
mdlWorkDgn_openFile　打开指定的文件作为工作文件
mdlWork_read　从工作文件中读一个元素描述符
mdlWork_write　向工作文件写一个元素描述符

工作文件用 mdlWorkDgn_openFile 打开，用下列测试检查工作文件打开是否正确。当对工作文件的操作完成以后，必须用 mdlWorkDgn_closeFile 函数关闭。

```
if (mdlWorkDgn_openFile (modelRefP, &format, &isThreeD, filename, NULL, FALSE)
                         != SUCCESS)
{
   mdlOutput_rscPrintf (MSG_MESSAGE, NULL, 0, 3, filename);
   mdlUtil_beep (1);
   return;
}
```

我们用到的工作文件函数有 mdlWorkDgn_openFile、mdlWorkDgn_write 和 mdlWorkDgn_closeFile。mdlWorkDgn_write 函数中的第二个参数 workPos 定义向工作文件写元素描述符的位置。在第一次向工作文件写时，我们把 workPos 设为 -1L，通知它要在文件的结束处追加元素。由于 mdlWorkDgn_write 函数返回下一个空位置，所以我们将把这个位置赋给 workPos，供下一次写操作时使用。

```
workPos = mdlWorkDgn_write (elmDP, workPos, modelRefP);
```

mdlWorkDgn_write 函数写 MicroStation 元素。它不知道前后元素。当我们要做一个新的设计文件时，需要用 mdlWorkDgn_createFile 函数先生成这个含有设计头的设计文件。

该应用程序执行成功的前提有二：①假设存在一个设计文件(如 C:\OUTPUT.DGN)；②当前设计文件中存在独立

的文本或文本节点元素(独立是指文本元素不能位于单元中等情况)。

在 MicroStation 命令键入域键入 mdl l strtext 加载应用程序，键入 STRTEXT 即可运行该应用程序，待出现 Enter DGN filename 提示时，进一步键入 C:\OUTPUT.DGN。程序的执行结果是把当前 DGN 中所有独立的文本或文本节点元素以字符串形式写入到 C:\OUTPUT.DGN 中。当然，C:\OUTPUT.DGN 可以是任何位置的一个设计文件。实际上，该设计文件也不一定非得为空。

```
/*----------------------------------------------------------------------+
|  Copyright (C) 1991 Mach N. Dinh-Vu, All Rights Reserved              |
|  Program    : strtext.mc                                              |
|  Revision   : 1.0.a                                                   |
|  UpgradeToV8: MicroStationFan    2006/05                              |
+----------------------------------------------------------------------+
|         scan the entire DGN file and located all text string.         |
|         Stroke the text into linestrings and then export to another DGN |
|         file.                                                         |
+----------------------------------------------------------------------*/
/*----------------------------------------------------------------------+
|          Include Files                                                |
+----------------------------------------------------------------------*/
#include <mdl.h>          /* system include files */
#include <global.h>
#include <msrmgr.h>
#include <userfnc.h>
#include <stdlib/string.h>
#include <mscell.fdf>
#include <mselemen.fdf>
#include <mselmdsc.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msstate.fdf>
#include <mswrkdgn.fdf>
#include "strtext.h"

#define   NEWELEMENT -1
#define   TOLERANCE  1000

DgnModelRefP   modelRefP;
ULong          workPos = -1L;
int            number_exported = 0;

/*----------------------------------------------------------------------+
|   name           main                                                 |
+----------------------------------------------------------------------*/
Private void  main (void)
{
   RscFileHandle   rfHandle;

   /* load our comnand table */
   if (mdlParse_loadCommandTable (NULL) == NULL)
       mdlOutput_error ("Unable to load command table.");
   mdlResource_openFile (&rfHandle, NULL, FALSE);
   mdlOutput_prompt ("Key-in STRTEXT [filename] to execute");
}
/*----------------------------------------------------------------------+
|    name         stroke_elm                                            |
+----------------------------------------------------------------------*/
Private int  stroke_elm ( MSElement *el,  void *params,  int  operation,
                          ULong  offset,  MSElementDescr *edP)
{
   DPoint3d        *points;
```

```
MSElementDescr  *elmDP;
MSElement        elm;
UInt32           color, weight, level, ggNum;
Int32            style;
int              numPnts;

if (mdlElement_getType(el) != TEXT_ELM)
   return SUCCESS;
if (mdlElement_stroke (&points, &numPnts, el, TOLERANCE) == SUCCESS)
{
   if (numPnts > MAX_VERTICES)
      numPnts = MAX_VERTICES;
   mdlOutput_rscPrintf (MSG_MESSAGE, NULL, 0, 2, ++number_exported);

   /* get element display symbology of current element */
   mdlElement_getSymbology (&color, &weight, &style, el);
   /* get properties of current element */
   mdlElement_getProperties (&level, &ggNum, NULL, NULL,
                             NULL, NULL, NULL, NULL, el);

   mdlLineString_create (&elm, NULL, points, numPnts);

   /* set element display symbology of line string */
   mdlElement_setSymbology (&elm, &color, &weight, &style);
   /* set properties of line string */
   mdlElement_setProperties (&elm, &level, &ggNum, NULL, NULL,
                             NULL, NULL, NULL, NULL);

   mdlElmdscr_new (&elmDP, NULL, &elm);
   workPos = mdlWorkDgn_write (elmDP, workPos, modelRefP);
      /* Display text written */
      mdlElement_display (el, HILITE);

      /* Free temporary descriptor */
      mdlElmdscr_freeAll (&elmDP);
   }
   return SUCCESS;
}
/*-------------------------------------------------------------------------------+
|    name     writeFile                                                          |
+-------------------------------------------------------------------------------*/
Private void writeFile ( char *filename )
{
   ULong            position = 0L;
   MSElementDescr  *edP;
   int              format;
   BoolInt          isThreeD;

   if (mdlWorkDgn_openFile (&modelRefP, &format, &isThreeD, filename, NULL, FALSE)
                            != SUCCESS)
   {
      mdlOutput_rscPrintf (MSG_MESSAGE, NULL, 0, 3, filename);
      mdlUtil_beep (1);
      return;
   }
   mdlOutput_rscPrintf (MSG_PROMPT, NULL, 0, 1, filename);

   while ((position = mdlElmdscr_read (&edP, position, 0, FALSE, NULL)) != 0L)
   {
      if ((edP->el.ehdr.type == TEXT_ELM ) ||
          (edP->el.ehdr.type == TEXT_NODE_ELM ))
```

```
      {
         mdlElmdscr_operation (edP, stroke_elm, NULL, ELMD_ELEMENT);
      }
      mdlElmdscr_freeAll (&edP);
   } /* end-while */

   /* cleanup */
   mdlOutput_prompt(" ");
   mdlOutput_command(" ");
   mdlWorkDgn_closeFile (modelRefP);
}
/*----------------------------------------------------------------------+
|   name        keyinText                                               |
+----------------------------------------------------------------------*/
Private void keyinText (void)
{
   char   filename [128];

   /* do nothing on <CR> only */
   if (!*statedata.cmdstring)
      return;

   /* save DGN filename */
   strncpy (filename, statedata.cmdstring, sizeof(filename));
   writeFile (filename);
}
/*----------------------------------------------------------------------+
|   name          setTextState                                          |
+----------------------------------------------------------------------*/
Private void setTextState (void)
{
   mdlState_setFunction (STATE_KEYIN, keyinText);
   mdlOutput_prompt ("Enter DGN filename");
}
/*----------------------------------------------------------------------+
|   name           strfile                                              |
+----------------------------------------------------------------------*/
cmdName strfile ( char *pStrings )
cmdNumber CMD_STRTEXT
{
   char    filename [128];

   mdlOutput_rscPrintf (MSG_COMMAND, NULL, 0, 0);
   if (*pStrings)
   {
      strcpy (filename, pStrings);
      writeFile (filename);
   }
   else
      setTextState ();
   return 0;
}
```

文件 strtext.r 中含有 STRTEXT 应用的命令表及信息。内容如下：

```
/*----------------------------------------------------------------------+
|   strtext.r command table and messages for strtext.mc                 |
+----------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"

#define    CT_NONE       0
#define    CT_STRTEXT    1
```

```
Table  CT_STRTEXT =
{
   { 1,  CT_NONE,    MANIPULATION,     NONE | TRY,    "STRTEXT" },
};

MessageList 0 =
  {
        {
        { 0, "Stroke TEXT to FILE" },
        { 1, "Output to File (%s)" },
        { 2, "TEXT written to file = %d"},
        { 3, "Unable to open file = %s" },
        }
  };
```

键入 bmake -a strtext 建立 STRTEXT 应用程序。strtext.mke 文件内容如下：

```
#--------------------------------------------------------------------
#      STRTEXT MDL Make File
#--------------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------------
#      Define constants specific to this STRTEXT application
#--------------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

strtextObjs = $(o)strtext.mo
strtextRscs = $(o)strtext.rsc  $(o)strtext.mp

#--------------------------------------------------------------------
# Create needed output directories if they don't exist
#--------------------------------------------------------------------
$(o)$(tstdir)               :    $(o)$(tstdir)

#-----------------------------------------
#       Generate Command Tables
#-----------------------------------------
$(privateInc)strtext.h      :   $(baseDir)strtext.r

#-----------------------------------------
#       Compile Resources
#-----------------------------------------
$(o)strtext.rsc             :   $(baseDir)strtext.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)strtext.mo              :   $(baseDir)strtext.mc

$(o)strtext.mp              :   $(strtextObjs)
             $(msg)
             >$(o)temp.cmd
             -a$@
             $(linkOpts)
             $(strtextObjs)
             <
             $(MLinkCmd) @$(o)temp.cmd
             ~time

#-----------------------------------------
```

```
#       Merge Objects into one file
#-------------------------------------------
$(mdlapps)strtext.ma       : $(strtextRscs)
             $(msg)
             >$(o)temp.cmd
             -o$@
             $(strtextRscs)
             <
             $(RLibCmd) @$(o)temp.cmd
             ~time
```

5.4 孤立单元

孤立单元就是由程序生成而不是从单元库产生的单元。在 PLACEBOX 应用程序中，我们已遇到过孤立单元的情况。这些孤立单元易于生成，和把元素加到设计文件一样。下面的例子是用一个设计文件中已有的元素生成一个孤立单元。

首先设定搜索标准，mdlLocate_normal 函数将只查寻设计文件中可显示和未锁定的元素。

```
mdlLocate_noElemNoLocked ();
mdlLocate_normal ();
```

我们将使用一个篱笆为我们的单元定义元素，mdlState_startFenceCommand 可以确定 cellFenceContents 做为处理篱笆元素时调用的函数。当用户定义单元原点时，调用 startCell 将生成单元头并建立元素描述符。

```
mdlState_startFenceCommand
                (cellFenceContents, /* routine for fence content */
                  NULL,             /* function to define fence outline */
                  startCell,        /* function for DATA point */
                  modfence,         /* function for RESET */
                  1,                /* message for command name */
                  2,               /* prompt for fence */
                  FENCE_NO_CLIP); /* Do not continue if clip lock turned on */
```

startCell 函数从从其参数 *pt 中取数据点作为单元的原点，调用 mdlCell_create 生成单元头。我们称单元为“ORPHAN”，表明这是一个孤立单元。

```
/* create orphan cell header */
mdlCell_create (&el, CELLNAME, pt, FALSE);
mdlElmdscr_new (&elmDP, NULL, &el);

mdlFence_process (NULL);

mdlElmdscr_display (elmDP, 0, NORMALDRAW);
```

mdlFence_process 函数为篱笆内被接收的每一个元素调用 cellFenceContents 函数。一旦在元素描述符 elmDP 中建立了单元，就可以将它加到设计文件中。只要直接调用 mdlElmdscr_add 就可以在文件中添加这个元素。

我们必须更改那些含有如层位掩码、描述符的总字长和类别位掩码等组成元素描述符的信息。如果调用 mdlCell_end 就有许多十分复杂的工作要做，因此在一开始就不要用 mellCell_begin 来启动单元。为了解决这

个问题，我们可以在把单元写入设计文件前调用 mdlElmdscr_validate 函数，该函数能自动设置复杂元素头元素中的相关信息以完成以上所描述的工作。

```
mdlElmdscr_validate (elmDP, 0);
mdlElmdcr_add (elmDP);
mdlElmdscr_freeAll(&elmDP);
```

cellFenceContents 函数要完成两件事。首先它要从篱笆中将元素读入元素描述符。 mdlElmdscr_read 函数可以读单个非复杂元素或是复杂元素头及其所有的组成元素。这是一个非常有用的函数，它允许我们生成一个有嵌套单元的孤立单元。第二步用 mdlElmdscr_undoableDelete 删除文件中原来的元素。

```
filePos = mdlElement_getFilePos (FILE_CURRENT, &currFile);
mdlElmdscr_read (&cellDP, filePos, MASTERFILE, FALSE, NULL);
mdlElmdscr_addToChain (elmDP, cellDP);
mdlElmdscr_undoableDelete (cellDP, filePos, TURE);
```

将所有的编码段组合起来，我们就得到 ORCELL 应用的源程序。键入 mdl l orcell 装入应用，在 MicroStation 命令键入域键入 CELL ORPHAN 运行应用。如果事先没有在设计文件中放置篱笆，将会在提示区显示 No fence defined。

```
/*----------------------------------------------------------------------+
|      copyright (c) 1991 Mach Dinh-Vu, All Rights Reserved             |
|      Create an ORPHAN cell from existing elements in the design file. |
|      CELL ORPHAN                                                      |
|      UpgradeToV8 : MicroStationFan     2006/05                        |
+----------------------------------------------------------------------*/
#include <mdl.h>
#include <msrmgr.h>
#include <mscell.fdf>
#include <mscnv.fdf>
#include <mselemen.fdf>
#include <mselmdsc.fdf>
#include <mslocate.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msstate.fdf>
#include "orcell.h"

#define  CELLNAME  L"ORPHAN"

MSElementDescr  *elmDP;
MSElementUnion   el;
ULong            filePos;

/*----------------------------------------------------------------------+
|   name          main                                                  |
+----------------------------------------------------------------------*/
Private void main (void)
{
   RscFileHandle   rfHandle;

   /* -  load our command table - */
   if (mdlParse_loadCommandTable (NULL) == NULL)
       mdlOutput_error ("Unable to load command table.");
   mdlOutput_error ("to execute, key-in CELL ORPHAN" );
   mdlResource_openFile (&rfHandle, NULL, FALSE);
```

```
}
/*----------------------------------------------------------------------+
|    name      setSearchType                                            |
+----------------------------------------------------------------------*/
Private void setSearchType (void)
{
   /* initialize search criteria to find nothing */
   mdlLocate_noElemNoLocked();
   mdlLocate_normal ();
}
/*----------------------------------------------------------------------+
|    name        startCell                                              |
+----------------------------------------------------------------------*/
Private void  startCell (DPoint3d *pt, int view)
{
   /* create orphan cell header */
   mdlCell_create (&el, CELLNAME, pt, FALSE);
   mdlElmdscr_new (&elmDP, NULL, &el);
   mdlFence_process (NULL);
   mdlElmdscr_display(elmDP, 0, NORMALDRAW);
   mdlElmdscr_validate (elmDP, 0);
   status = mdlElmdscr_add (elmDP);
   mdlElmdscr_freeAll (&elmDP);
}
/*----------------------------------------------------------------------+
| name        cellFenceContents                                        |
+----------------------------------------------------------------------*/
Private int   cellFenceContents ()
{
   MSElementDescr  *cellDP;
   filePos = mdlElement_getFilePos (FILEPOS_CURRENT, NULL);
   mdlElmdscr_read (&cellDP, filePos, MASTERFILE, FALSE, NULL);
   mdlElmdscr_appendDscr (elmDP, cellDP);
   mdlElmdscr_undoableDelete (cellDP, filePos, TRUE);
   return SUCCESS;
}
/*----------------------------------------------------------------------+
|  namen      modfence                                                 |
+----------------------------------------------------------------------*/
cmdName        modfence (void)
cmdNumber      CMD_CELL_ORPHAN
{
   setSearchType ();
   mdlState_startFenceCommand
              (cellFenceContents,  /* routine for fence content */
                   NULL,           /* function to define fence outline */
                   startCell,      /* function for DATA point */
                   modfence,       /* function for RESET */
                   1,              /* message for command name */
                   2,              /* prompt for fence */
                   FENCE_NO_CLIP); /* Do not continue if clip lock turned on */
   return 0;
}
```

Orcell.r 文件中含有 ORCELL 应用的命令表及信息。

```
/*----------------------------------------------------------------------+
|   orcell.r command table and messages for orcell.mc                   |
+----------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"

#define    CT_NONE        0
```

```
#define    CT_CELL        1
#define    CT_ORPHAN      2

Table  CT_CELL =
{
   { 1,  CT_ORPHAN,  MANIPULATION,     REQ,        "CELL" },
}

Table  CT_ORPHAN =
{
   { 1,  CT_NONE,    INHERIT,        REQ,         "ORPHAN" },
};

MessageList 0 =
{
   {
      { 1, "Create Orphan Cell" },
      { 2, "Define Cell Origin" },
   }
};
```

键入 bmake -a orcell, 建立 ORCELL 应用。

```
#-------------------------------------------------------------------
#      ORCELL MDL Make File
#-------------------------------------------------------------------
%include  mdl.mki

#-------------------------------------------------------------------
#      Define constants specific to this orcell application
#-------------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

orcellObjs = $(o)orcell.mo
orcellRscs = $(o)orcell.rsc  $(o)orcell.mp

#-------------------------------------------------------------------
# Create needed output directories if they don't exist
#-------------------------------------------------------------------
$(o)$(tstdir)               :    $(o)$(tstdir)

#-----------------------------------------
#       Generate Command Tables
#-----------------------------------------
$(privateInc)orcell.h      :   $(baseDir)orcell.r

#-----------------------------------------
#       Compile Resources
#-----------------------------------------
$(o)orcell.rsc             :   $(baseDir)orcell.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)orcell.mo              :   $(baseDir)orcell.mc

$(o)orcell.mp              :   $(orcellObjs)
            $(msg)
            >$(o)temp.cmd
            -a$@
            $(linkOpts)
            $(orcellObjs)
            <
```

```
            $(MLinkCmd) @$(o)temp.cmd
            ~time

#-----------------------------------------
#       Merge Objects into one file
#-----------------------------------------
$(mdlapps)orcell.ma       : $(orcellRscs)
            $(msg)
            >$(o)temp.cmd
            -o$@
            $(orcellRscs)
            <
            $(RLibCmd) @$(o)temp.cmd
            ~time
```

【注】: 本章所有源代码都在 MicroStation V8 2004 下调试通过。

第六章　数学与几何运算

本章中包含向量几何、当前变换、视图变换、变换和旋转矩阵，后四项内容归属于一般的几何变换，而几何变换是所有图形应用的组成部分。如果一想到矩阵计算就使你头痛，那么不必担心，让 MDL 函数帮助你。你自己只要考虑调用哪一个函数才能得到期望的结果就行了。

6.1 浮点常量

MDL 提供预定义浮点常量供我们的程序使用。这些常量实际上是变量，所以它们的使用随着程序的变化而变化。使用 fc_1 代替 1.0 似乎不合逻辑，不过我们应该使用计算的常量，如 fc_piover180，因为这样会增强程序的可读性。

浮点常量	值
fc_onehalf	0.5
fc_zero	0.0
fc_1	1.0
fc_m1	-1.0
fc_2	2.0
fc_3	3.0
fc_4	4.0
fc_5	5.0
fc_10	10.0
fc_100	100.0
fc_p1	0.1
fc_p01	0.01
fc_p001	0.001
fc_p0001	0.0001
fc_pi	π
fc_180overpi	180.0 / π
fc_piover180	π / 180.0
fc_piover2	π / 2.0
fc_piover3	π / 3.0
fc_piover4	π / 4.0
fc_piover6	π / 6.0
fc_30	30.0
fc_60	60.0
fc_90	90.0
fc_180	180.0
fc_270	270.0
fc_360	360.0
fc_750	750.0

fc_1000	1000.0
fc_10000	10000.0
fc_100000	100000.0
fc_2pi	2.0 * π
fc_epsilon	0.00001
fc_mm_per_in	25.40
fc_360000	360000.0
fc_iang_to_rad	(π / 180.0) / 360000.0
fc_rad_to_iang	(180.0 * 360000.0) / π
fc_miang_to_rad	-1.0 * (π / 180.0) / 360000.0
fc_rmaxi4	RMAXI4
fc_rmini4	RMINI4
fc_rmaxui4	RMAXUI4
fc_tan30	0.5773502692

表 6.1　浮点常量表

6.2 向量几何

向量是一种表示距离和方向的方法。向量可以看作是从起点开始的一个箭头。向量的分量定义箭头的顶点。每一个向量都有三个分量，即 X 方向分量、Y 方向分量及 Z 方向分量。向量的长度称为向量的模，用一些单位来表示。

比例可以改变向量的长短。例如按比例表示一个向量，我们写 A = sB，这里 s 是比例因子，向量 A 的长度或模用各个分量平方和的平方根来计算，向量 A 的长度用｜A｜来表示。

6.2.1 点乘

向量 A 与向量 B 的点乘可以表示为如下等式：

A • B = ｜A｜｜B｜Cos θ

这里 θ 是向量 A 与向量 B 之间的最小夹角。向量 B 在向量 A 上的投影得到的向量示为 $proj_AB$(见图 6.1)。

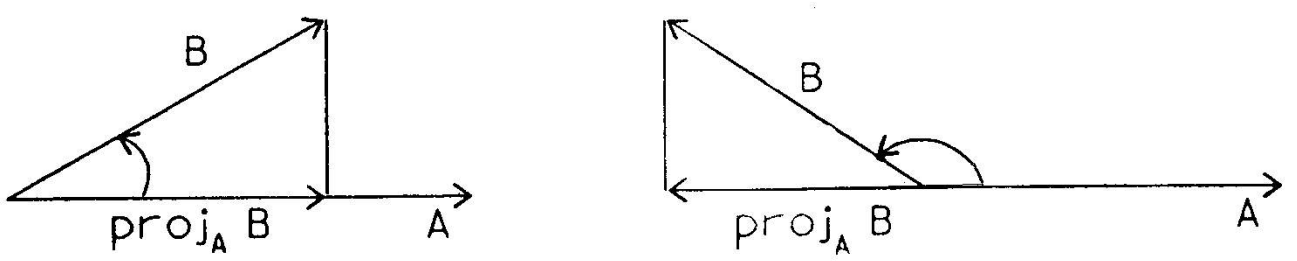


图 6.1 向量 B 在向量 A 上的投影

点乘是计算向量 B 在向量 A 上的分量的一种简便方法。因此在 B 向量沿 A 向量方向的分量可表示为：

A • B/｜A｜

如果 $Proj_AB$ 与+A 方向相同则符号为正，如果与-A 方向相同则符号为负。

6.2.2 叉乘

A、B 两个向量叉乘返回一个垂直于 A、B 向量所确定平面的单位向量。在图 6.2 的例子中 A×B 与 B×A 不能互换(结果是不同的)。颠倒因子的顺序，可以改变单位向量的方向。从第一个向量向第二个向量按右手旋转法则即可

确定乘积的方向。

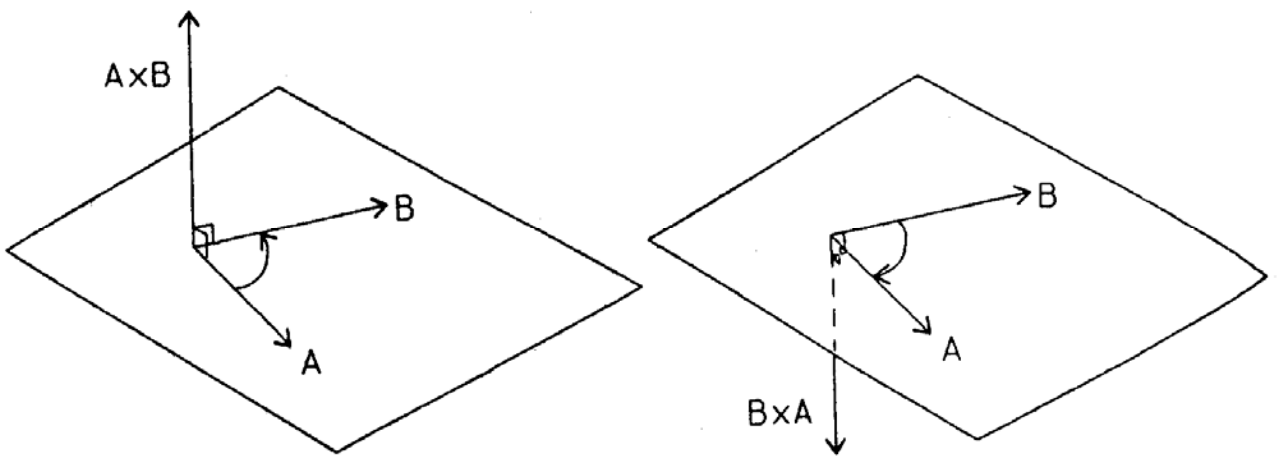


图 6.2　A×B 与 B×A

6.3 向量操作函数

MDL 提供许多向量操作函数，汇总如下：

mdlVec_addPoint	两个方向不同的向量相加返回一个向量
mdlVec_addPointArray	与上同，但对一个点的数组操作
mdlVec_areParallel	确定两个向量是否平行
mdlVec_arePerpendicular	确定两个向量是否垂直
mdlVec_colinear	确定三个向量是否共线
mdlVec_computeNormal	计算两点间的法向向量
mdlVec_crossProduct	返回两个向量叉积
mdlVec_distance	返回两点间向量的模
mdlVcc_dotProduct	给出两个向量点积
mdlVec_forcePlanarity	向平面内的最近点投影一个点
mdlVec_intersect	计算两条线的交点
mdlVec_magnitude	返回向量的模
mdlVec_normalize	归一化向量使其模为 1
mdlVec_pointEqual	检查两点是否相同
mdlVec_pointEqualUOR	在一个 UOR 内检查两点是否相同
mdlVec_projectPoint	沿矢量按一个距离投影一个点
mdlVcc_scale	按比例缩放一个向量
mdlVec_subtractPoint	两向量相减
mdlVec_subtractPointArray	与上同，但对一个点的数组操作

讨论 PLCIRC 程序将有助于对向量几何的理解。这个程序基于我们以前讨论过的 PLBOX 程序。在这里不是用方框，而是用圆圈圈住一段注释并带有引线及箭头。

我们的引线从圆的边缘一直到箭头的顶点。这条线段需要计算，圆心及箭头端顶点为已知量。使用向量，我们可以容易地算出圆和直线的交点。

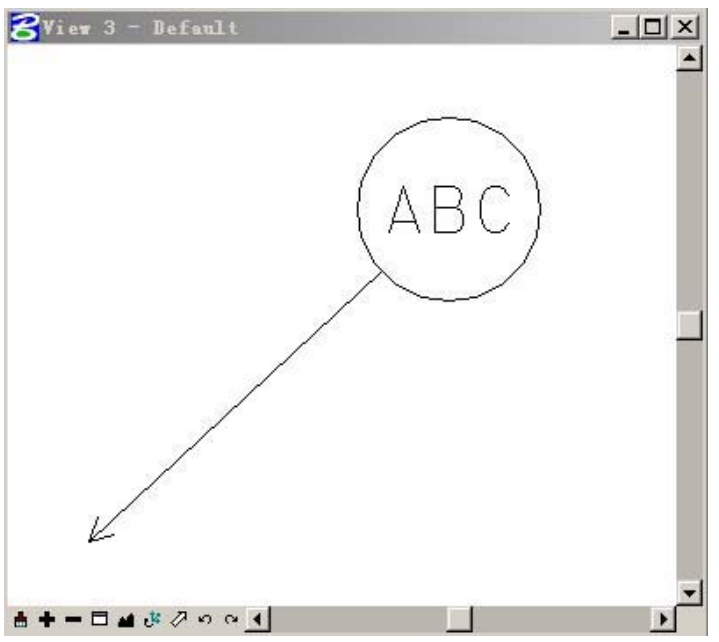


图 6.3 PLCIRC 程序的结果

必须给出两个点，第一个是圆的圆心，第二个是箭头所指的位置。用 mdlVec_substractPoint 两点相减就可以得到方向向量 dirVector。归一化 dirVector 使其模为 1。以圆的半径为比例缩放 dirVector 将得到以圆的半径为模的方向矢量。将这个矢量加在圆心，就得到了向量与圆周的交点（见图 6.4）。

这个 MDL 函数列在下面：

```
/*----------------------------------------------------------------------+
|  name    constIntersectionPoint                                       |
+----------------------------------------------------------------------*/
Private void constIntersectionPoint
(
  Dpoint3d    *intersectionPt,    /* intersection point on circle */
  double      *radius,            /* radius of cricle */
  Dpoint3d    *centerPt,          /* center of circle */
  Dpoint3d    *directionPt        /* end point of arrowhead */
)
{
   Dpoint3d   dirVector;

   mdlVec_subtractPoint (&dirVector, directionPt, centerPt);
   mdlVec_normalize (&dirVector);
   mdlVec_scale (&dirVector, &dirVector, *radius);
   mdlvec_addPoint (intersectionPt, centerPt, &dirVector);
}
```

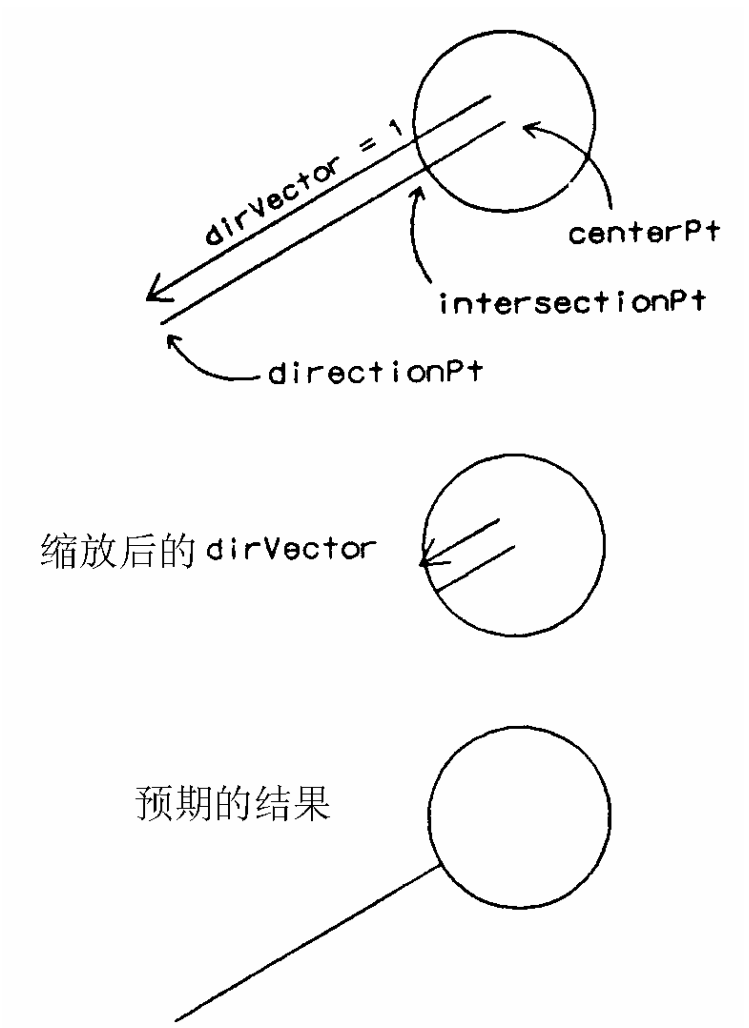


图 6.4　计算引线

6.4 当前变换

当前变换提供了一种十分有效的放置元素的功能。某一个点或一组点，通过当前变换矩阵可以移动、旋转和/或缩放。当前变换最普遍的使用是用工作单位表示所有坐标。它使得应用能在使用不同工作单位的设计文件之间转换。

下面是当前变换函数的汇总：

mdlCurrTrans_begin	将当前变换矩阵的拷贝推入堆栈
mdlCurrTrans_clear	将所有的变换矩阵从堆栈之中弹出
mdlCurrTrans_end	将变换矩阵弹到栈顶
mdlCurrTrans_getAddress	返回当前正向或反向变换的地址
mdlCurrTrans_identity	用单位矩阵代替当前的变换矩阵
mdlCurrTrans_invTransPointArray	从设计文件坐标变换到当前坐标系统
mdlCurrTrans_rotateByAngles	按角度旋转当前变换矩阵
mdlCurrTrans_rotateByRMatrix	按旋转矩阵旋转当前变换矩阵
mdlCurrTrans_rotateByView	按视图旋转当前变换矩阵
mdlCurrTrans_scale	按比例缩放当前变换矩阵
mdlCurrTrans_transformPointArray	从当前坐标系统转换到设计文件坐标系统
mdlCurrTrans_translateOrigin	为当前变换矩阵设定新的原点
mdlCurrTrans_translateOriginWorld	用 UOR 为当前变换矩阵设定新的原点
mdlCurrTrans_masterUnitsIdentity	用每个主单位含有的位置单位的单位矩阵代替当前变换矩阵

我们在应用 PLCIRC 中的引线末端放置一个箭头(设想一下不借助当前变换矩阵写这样一个应用)。首先必须计算出这条线段的角度，然后放置箭头，再旋转并移动这个箭头，把它放到正确位置上。

调用函数 mdlCurrTrans_begin 会将当前变换的拷贝推入堆栈。mdlCurrTrans_identity 函数将生成从已知点的变换。mdlCurrTrans_translateOrigin 将移动要作为箭头顶点的当前变换的原点。mdlCurrTrans_invTransPointArray 将从位置单位到当前坐标系(即当前工作单位)映射所有的点。

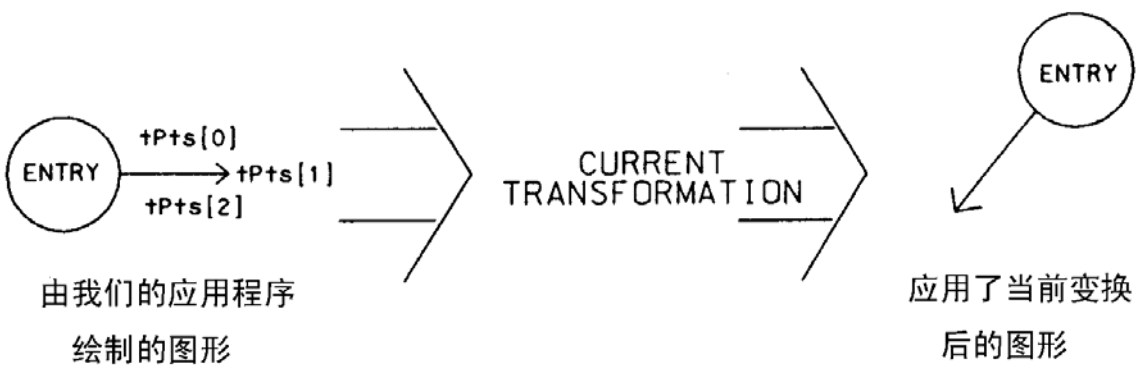


图 6.5　变换的结果

```
mdlCurrTrans_begin ();
mdlCurrTrans_identity ();
mdlCurrTrans_traslateOringin (&pntP[0]);
mdlCurrTrans_invtransPointArray(tPts, pntP, 2);
```

使用当前变换时，我们假定箭头水平放置，并且朝右。我们可以做到这一点，因为我们已经用当前变换旋转了那条线的倾角。例如，如果线旋转了 34.52°，我们使用 mdlCurrTrans_rotateByAngles 使当前变换旋转-34.52°，使它们看起来都是水平的。

```
/* calculate angle of line */
zangle = atan2 ((tPts[0].y-tPts[1].y), (tPts[0].x-tPts[1].x));
mdlCurrTrans_rotateByAngles (0.0, 0.0, zangle);
```

在水平方向上工作便于计算生成箭头所需要的点。

```
/* Create arrowhead */
tPts[1] = tPts[0];


tPts[2].x = tPts[0].x - arrowsize;
tPts[2].y = tPts[0].y - (arrowsize / 2);
tPts[0].x -= arrowsize;
tPts[0].y += arrowsize / 2;
```

如果不把旋转原点移动到箭头的顶点，旋转围绕点则是 0,0。如图 6.6A 是没有经过 mdlCurrTrans_translateOrigin 的旋转，图 6.6B 是正确的方法。

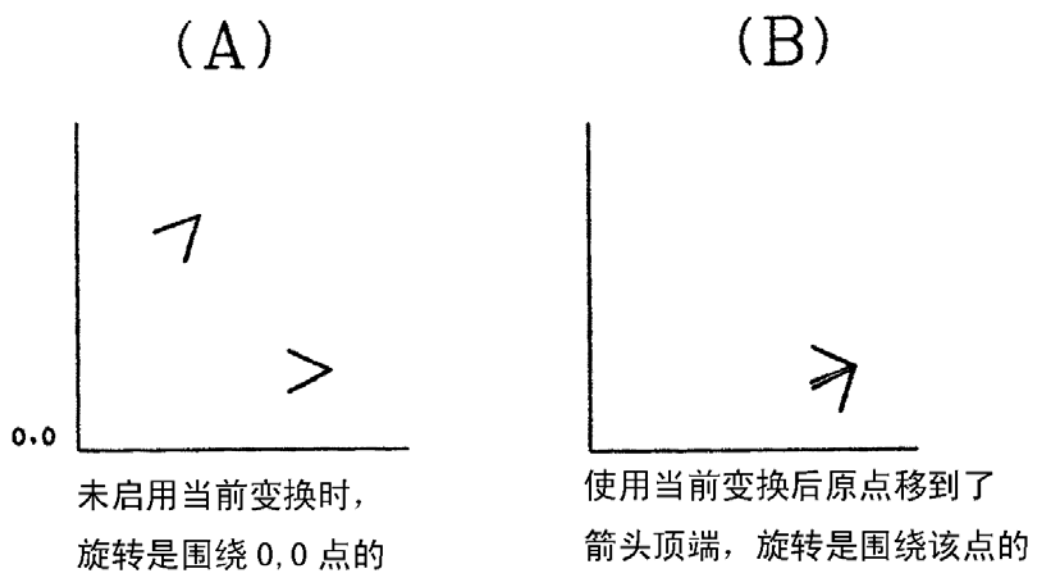


图 6.6　箭头的旋转

应用 plcirc.mc 和 plbox.mc 相似。我们已经介绍了函数 constIntersectionPoint，并把当前变换加进 generatelmage。键入 mdl l plcirc 装入这个应用，键入 PLACE TCIRC 执行这个应用。要正确显示本应用的效果还需要：①设置文本对齐方式为 Center-Center；②键入 PLACE TCIRC 后的动态显示过程中还要在命令键入域键入一个字符串，如 ABC。

```
/*----------------------------------------------------------------------+
|   Copyright (c) 1991, Mach N. Dinh-Vu, All Rights Reserved           |
|   Program   : plcirc.mc                                              |
|   Revision  : 1.0.a                                                  |
|   UpgradeToV8 : MicroStationFan   2006/05                            |
+----------------------------------------------------------------------+
|   Example MDL function to place a circle around                      |
|   a text string with a leader line and an arrowhead                  |
+----------------------------------------------------------------------*/
/*----------------------------------------------------------------------+
|   Include   Files                                                    |
+----------------------------------------------------------------------*/
#include <mdl.h>        /* system include files */
#include <msrmgr.h>
#include <stdlib/math.h>
#include <stdlib/string.h>
#include <mscell.fdf>
#include <mscnv.fdf>
#include <mscurrtr.fdf>
#include <mselemen.fdf>
#include <mselmdsc.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msstate.fdf>
#include <msvec.fdf>
#include "plcirc.h"

/*----------------------------------------------------------------------+
|   Private Global variables                                           |
+----------------------------------------------------------------------*/
static char textin[128];
Dpoint3d    pntP[3];

/*----------------------------------------------------------------------+
|   name    main                                                       |
+----------------------------------------------------------------------*/
Private void main ( void )
```

```
{
   RscFileHandle  rfHandle;

   /* load our command tabla */
   if (mdlParse_loadCommandTable (NULL) == NULL)
      mdlOutput_error ("Unable to load command table.");
   mdlResource_openFile (&rfHandle, NULL, FALSE);
   mdlOutput_error ("Enter PLACE TCIRC to start");
}
/*----------------------------------------------------------------------+
|   name    constIntersectionPoint                                      |
+----------------------------------------------------------------------*/
Private void constIntersectionPoint
(
Dpoint3d    *intersectionPt,      /* intersection point on circle */
double      *radius,              /* radius of circle */
Dpoint3d    *centerPt,            /* center of circle */
Dpoint3d    *directionPt          /* end point of arrowhead */
)
{
   Dpoint3d     dirVector;

   mdlVec_subtractPoint (&dirVector, directionPt, centerPt);
   mdlVec_normalize (&dirVector);
   mdlVec_scale (&dirVector, &dirVector, *radius);
   mdlVec_addPoint (intersectionPt, centerPt, &dirVector);
}
/*----------------------------------------------------------------------+
|    name    generateImage - dynamic function for complex case.        |
+----------------------------------------------------------------------*/
Private int generateImage
(
Dpoint3d     *pt,
int          view,
int          drawMode
)
{
   MSElementDescr  *elmDP;
   MSElementUnion   el;
   Dpoint3d         origin;
   Dpoint3d         tPts[3];
   double           zangle, radius, txtHeight;
   ULong            arrowsize, chheight;

   mdlParams_getActive ((double *)&txtHeight, ACTIVEPARAM_TEXTHEIGHT);
   chheight = (ULong)txtHeight;
   arrowsize = chheight / 2;
   radius = chheight * 2;
   pntP[1] = *pt;

   mdlCurrTrans_begin ();
   mdlCurrTrans_identity ();
   mdlCurrTrans_translateOrigin (&pntP[0]);
   mdlCurrTrans_invtransPointArray(tPts, pntP, 2);

   origin = tPts[1];        /* origin of text */
   mdlCell_create (&el, L"plcirc", &origin, FALSE);
   mdlElmdscr_new (&elmDP, NULL, &el);
   mdlText_create (&el, NULL, textin, &tPts[1], NULL, NULL, NULL, NULL);
   mdlElmdscr_appendElement (elmDP, &el);

   constIntersectionPoint (&tPts[1], &radius, &origin, &tPts[0]);
```

```
   mdlLine_create (&el, NULL, tPts);
   mdlElmdscr_appendElement (elmDP, &el);

   /* Create arrowhead */
   mdlEllipse_create (&el, NULL, &origin, radius, radius, NULL, 0);
   mdlElmdscr_appendElement (elmDP, &el);

   /* calculate angle of line */
   zangle = atan2 ((tPts[0].y-tPts[1].y), (tPts[0].x-tPts[1].x));
   mdlCurrTrans_rotateByAngles (0.0, 0.0, zangle);

   /* Create arrowhead */
   tPts[1]   = tPts[0];
   tPts[2].x = tPts[0].x - arrowsize;
   tPts[2].y = tPts[0].y - (arrowsize/2);
   tPts[0].x -= arrowsize;
   tPts[0].y += arrowsize/2;

   mdlLineString_create (&el, NULL, tPts, 3);
   mdlElmdscr_appendElement (elmDP, &el);
   mdlElmdscr_display (elmDP, 0, drawMode);
   if (drawMode == NORMALDRAW)
   {
      mdlElmdscr_add (elmDP);
   }
   mdlElmdscr_freeAll (&elmDP);
   mdlCurrTrans_end( );
   return  SUCCESS;
}
/*----------------------------------------------------------------+
|   name    keyinText                                             |
+----------------------------------------------------------------*/
Private void  keyinText (char *cmdStrP)
{
   if (!*statedata.cmdstring)
      return;
   strncpy (textin, statedata.cmdstring, sizeof(textin));
}
/*----------------------------------------------------------------+
|   name    placeCirc_secondPoint                                 |
+----------------------------------------------------------------*/
Private void   placeCirc_secondPoint
(
Dpoint3d     *pt,
int          view
)
{
   generateImage (pt, view, NORMALDRAW);
}
/*----------------------------------------------------------------------+
|   name    placeCirc_done                                              |
+----------------------------------------------------------------------*/
Private void   placeCirc_done (void)
{
   mdlState_restartCurrentCommand ();
}
/*----------------------------------------------------------------------+
|   name     placeCirc_firstPoint                                       |
+----------------------------------------------------------------------*/
Private void   placeCirc_firstPoint
(
Dpoint3d      *pt,
```

```
int           view
)
{
   pntP[0] = *pt;    /* save first point */

   /* Set the datapoint state function for the second point. */
   mdlState_setFunction (STATE_KEYIN, keyinText);
   mdlState_setFunction (STATE_DATAPOINT, placeCirc_secondPoint);
   mdlOutput_rscPrintf  (MSG_PROMPT, NULL, 0, 3);
   /* setup dynamics for the second point */
   mdlState_setFunction (STATE_COMPLEX_DYNAMICS, generateImage);
}
/*----------------------------------------------------------------------+
|   name       placeCirc_start                                         |
+----------------------------------------------------------------------*/
cmdName       placeCirc_start (void)
cmdNumber     CMD_PLACE_TCIRC
{
   mdlState_startPrimitive (placeCirc_firstPoint, placeCirc_start, 1, 2);
   return 0;
}
```

文件 plcric.r 含有命令表和提示信息。源文件内容如下：

```
/*----------------------------------------------------------------------+
|   PLCIRC.R command table and messages for PLCIRC.MC                   |
+----------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"

#define    CT_NONE       0
#define    CT_PLACE      2

Table     1 =
{
   { 1,  CT_PLACE,       PLACEMENT,    REQ,                "PLACE" },
}
Table  CT_PLACE =
{
   { 1,  CT_NONE,        INHERIT,      NONE | CMDSTR(1),   "TCIRC" },
};
MessageList  0 =
{
   {
   { 0, "" },
   { 1, "MDL Place Text in Circle" },
   { 2, "Define start point" },
   { 3, "Define circle position" },
   }
}
```

制作文件 (makefile) 使用了标准的 seed.mke, 用 plcirc 来代替所有出现的 seed。

```
#--------------------------------------------------------------
#       PLCIRC MDL Make File
#--------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------
#       Define constants specific to this plcirc application
#--------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

plcircObjs = $(o)plcirc.mo
```

```
plcircRscs = $(o)plcirc.rsc  $(o)plcirc.mp

#--------------------------------------------------------------------
# Create needed output directories if they don't exist
#--------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#-----------------------------------------
#       Generate Command Tables
#-----------------------------------------
$(privateInc)plcirc.h      :   $(baseDir)plcirc.r

#-----------------------------------------
#       Compile Resources
#-----------------------------------------
$(o)plcirc.rsc             :   $(baseDir)plcirc.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)plcirc.mo              :   $(baseDir)plcirc.mc

$(o)plcirc.mp              :   $(plcircObjs)
             $(msg)
             >$(o)temp.cmd
             -a$@
             $(linkOpts)
             $(plcircObjs)
             <
             $(MLinkCmd) @$(o)temp.cmd
             ~time

#-----------------------------------------
#       Merge Objects into one file
#-----------------------------------------
$(mdlapps)plcirc.ma       : $(plcircRscs)
             $(msg)
             >$(o)temp.cmd
             -o$@
             $(plcircRscs)
             <
             $(RLibCmd) @$(o)temp.cmd
             ~time
```

6.4.1 变换矩阵函数

MicroStation 使用几何变换来移动、缩放和旋转。这三种基本变换可以结合起来并用矩阵表示。从数学上证明一个矩阵可以表示所有三种变换，已超出了本书的范围。我们假设这个结论已经成立，看看用一个 MDL 程序来如何做到这一点。

让我们来看一下可用的变换矩阵函数：

mdlTMatrix_fromRMatrix	从一个旋转矩阵返回变换矩阵
mdlTMatrix_getIdentity	设置变换矩阵为单位矩阵
mdlTMatrix_getTranslation	从变换矩阵返回一个移动量
mdlTMatrix_masterToReference	返回可以从主文件坐标系到参考文件坐标系变换元素的矩阵
mdlTMatrix_multiply	两个矩阵相乘
mdlTMatrix_referenceToMaster	返回可以从参考文件坐标系到主文件坐标系

	变换元素的矩阵
mdlTMatrix_rotateByAngles	按角度(弧度)旋转变换矩阵
mdlTMatrix_rotateByRMatrix	按照旋转矩阵旋转变换矩阵
mdlTMatrix_rotateScalePoint	按照变换矩阵旋转并按比例缩放一个点
mdlTMatrix_setOrigin	建立变换矩阵的原点
mdlTMatrix_scale	按比例系数缩放变换矩阵
mdlTMatrix_setTranslation	设置变换矩阵的平移
mdlTMatrix_transformPoint	按照变换矩阵变换一个点
mdlTMatrix_transformPointArray	按照变换矩阵变换一组点
mdlTMatrix_translate	改变变换矩阵的原点
mdlTMatrix_transpose	交换变换矩阵的行和列

6.4.2 旋转、缩放和移动

为了证明一个矩阵能够表示旋转、缩放和平移，我们要写一个管理文本或文本节点的 MDL 程序。这个程序 textrms.mc 将首先提示文本或文本节点，然后按照激活角度进行旋转，按照激活比例系数进行缩放，并把它移动到新的位置。在这个程序中我们将不提供选择集或篱笆操作。如果你想扩展 textrms.mc 使它包含这些功能，请参考第四章。

我们用 mdlState_startModifyCommand 作为状态函数来修改元素。transformElement 函数为我们的操作定义动态功能。

```
mdlLocate_noElemNoLocked ();
mdlLocate_setElemSearchMask (sizeof(searchType)/sizeof(int), searchType);
mdlState_startModifyCommand (rotMove_start, rotMove_accept,
                             transformElement, NULL, NULL, 1, 0, TRUE, 0);
mdlLocate_init ();
```

这个 transformElement 函数将旋转、缩放和移动文本。其中操作的次序非常重要，因为最后结果的变换依赖于这个次序。图 6.7 表示了变换一个字符串的正确次序。函数 mdITMatrix_rotateByAngles 和 mdlTMatrix_scale 从当前变换的原点开始操作。因此，我们需要建立一个新的变换，并使这个点成为当前变换的原点(见图 7.6B)。这是用函数 mdlCurrTrans_translateOrigin 完成的。

使用 mdlTMatrix_getIdentity 之所以重要，是因为它把变换矩阵设置到一个已知的状态。为了缩放这个矩阵，我们将使用 mdlTMatrix_scale，并传给它激活比例系数。 mdlTMarix_rotateByAngles 则根据激活角度旋转这个矩阵。这个函数要求角度以弧度计，因此我们需要通过乘以 fc_piover180 把角度旋转为弧度。

```
mdlTMatrix_getIdentity (&tMatrix);
mdlTMatrix_scale (&tMatrix, &tMatrix, tcb->xactscle,
                  tcb->yactscle, tcb->zactscle);
mdlTMatrix_rotateByAngles (&tMatrix, &tMatrix, 0.0, 0.0,
                           tcb->actangle*fc_piover180);
mdlTMatrix_translate (&tMatrix, &tMatrix, distance.x,
                      distance.y, distance.z);
```

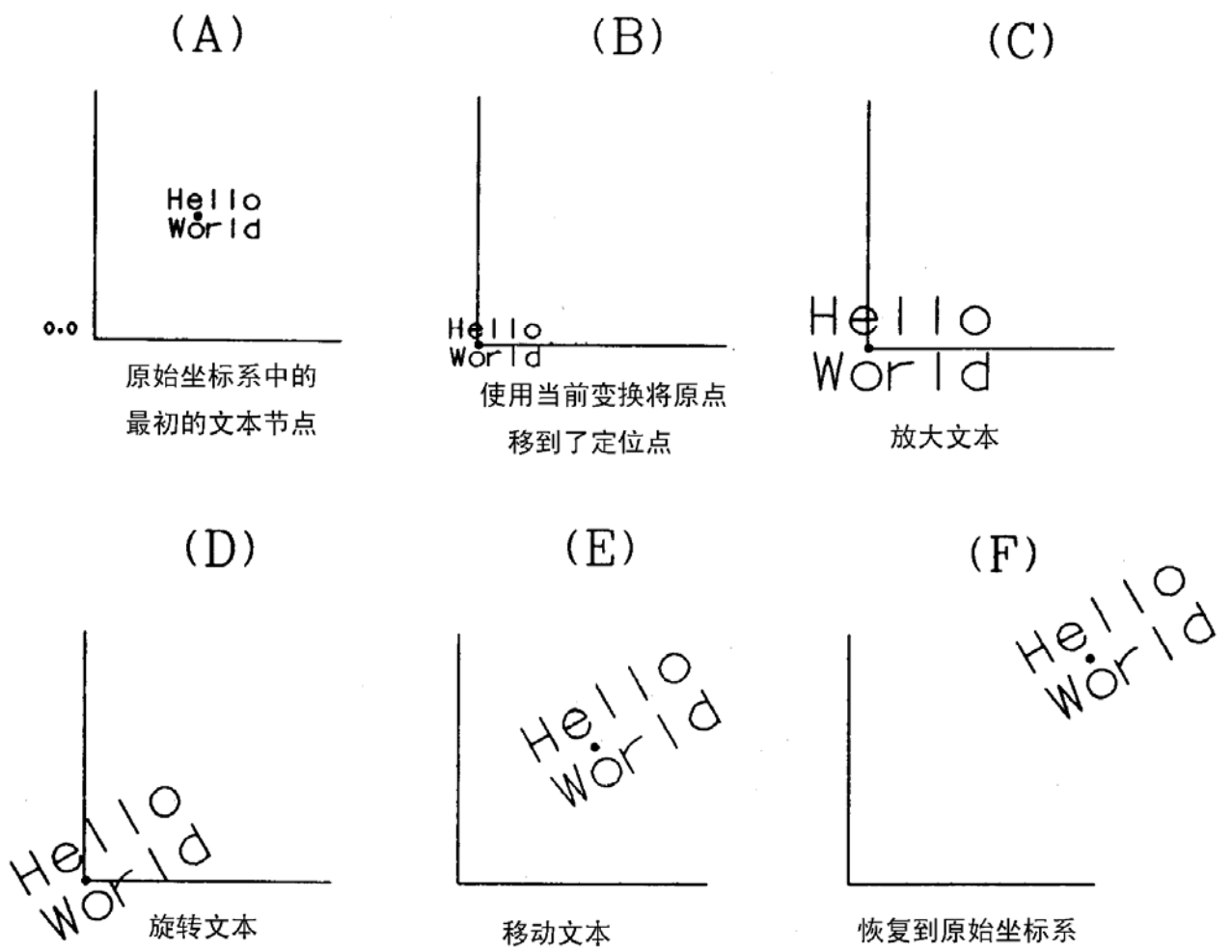


图 6.7 “Hello World”的转换

移动通过 mdlTMarix_translate 进行。为了把这个矩阵用于我们的字符串，我们将使用函数 mdlElement_transform。最后一步是处理当前转换矩阵，这将使我们回到原来的坐标系。注意，如果 AS=1 和 AA=0，则这个程序的作用与 MOVE ELEMENT 相同。

通过使用变换矩阵，我们已把非常复杂的数学问题转化为几行简单的编码。

6.5 数据格式

MicroStation 和 MDL 支持多种不同类型的数据格式。下面列出在这些格式间使用的数据变换函数：

mdlCnv_doubleToNativeFloat	把双精度数据转换为本机浮点数
mdlCnv_nativeFloadToDouble	把浮点数转换为本机双精度数
mdlCnv_DPointToIPoint	把 Dpoint3d 转换为 Point3d
mdlCnv_IPointToDPoint	把 Point3d 转换为 DPoint3d
mdlCnv_DPointToIPointArray	把一个 Dpoint3d 数组转换为一个 Point3d 数组
mdlCnv_IPointToDPointArray	把一个 Point3d 数组转换为一个 DPoint3d 数组
mdlCnv_fromAsciiToR50	把 ASCII 串转换为 RAD50 串
mdlCnv_fromR50ToAscii	把 RAD50 串转换为 ASCII 串
mdlCnv_fromScanFormat	把扫描格式的无符号长整型数转换为内部格式中的有符号长整型数
mdlCnv_roundDoubleToLong	把双精数转换为长整型数
mdlCnv_roundDoubleToULong	把双精数转换为无符号长整型数
mdlCnv_swapWordArray	在一个 4 字节整数数组中交换字
mdlCnv_toScanFormat	把内部格式的有符号长整型数转换为扫描格式中的无符号长整型数

函数 mdlCnv_fromR50ToAscii 值得注意，因为它允许在 RAD50 和 ASCII 之间进行转换。RAD50 的使用可追溯到 Intergraph 原来在小型机 DEC-PDP11 系列机上的软件系统。RAD50 的优点在于它在每 16 位的两个字节中存贮 3 个字符，而用 ASCII 码只能存贮 2 个字符。RAD50 的缺点是字符限制在一个有限的字符集内，当我们希望抽取单元名时，必须向 ASCII 转换，如下所示：

```
mdlCnv_fromR50ToAscii (6, &celtHeader->cell_lib_hdr.name, CellName);
mdlCnv_fromR50ToAscii (27, &cellHeader->cell_lib_hdr.descrip,
                       cellDescription);
```

【注】:RAD50 只限于在 MicroStation V8 以前的版本中使用!在 V8 的 MDL API 中不存在 mdlCnv_fromR50ToAscii 和 mdlCnv_fromR50ToAscii 函数。

6.5.1 点堆栈

传送到 MicroStation 的数据点可以放在一个点堆栈中。内部变量 statedata.dPointStack 保存这些信息。元素的位置逻辑以实数格式即 DPoint3d 存贮位置点。在函数 getMoveDistance 中我们要存贮固定指针，并用 mdlVec_subtractPoint 计算移动的距离。

```
Dpoint3d   anchor = statedata.dPointStack[0];
/*  subtract anchor point from current point t0 get the distance  */
mdIVec_subtractPoint (distance, pt, &anchor);
```

这里是 TEXTRMS 应用的完整的源程序。键入 mdl l textrms 装入这个应用。键入 TEXT RMS 执行这个应用。在提示区出现 Text Rotate Move Scale>Identify element 时用鼠标选取文本元素，该文本元素将会按照激活角度旋转、按激活比例缩放并移动到光标所在的位置。如果事先未设置激活角度(Active Angle=0)和激活比例(Active Scale=1)，则该应用的执行效果和元素移动命令相同。

```
/*----------------------------------------------------------------------+
|  Copyright (C) 1991, Mach N. Dinh-Vu, All Rights Reserved.            |
|  Program    : textrms.mc                                              |
|  1 Revision : 1.0.a                                                   |
|  UpgradeToV8: MicroStationFan    2006/05                              |
+----------------------------------------------------------------------+
|  This program will rotate, scale and move the identified              |
|  text or text node by the active angle and active scale.              |
+----------------------------------------------------------------------*/
/*----------------------------------------------------------------------+
|  Include Files                                                        |
+----------------------------------------------------------------------*/
#include <mdl.h>    /* system include files */
#include <msrmgr.h>
#include <tcb.h>
#include <mscurrtr.fdf>
#include <mselemen.fdf>
#include <mslocate.fdf>
#include <mstmatrx.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msstate.fdf>
#include <msvec.fdf>
#include "textrms.h"

/*----------------------------------------------------------------------+
|  Private Global, variables                                            |
+----------------------------------------------------------------------*/
int          currFile;
ULong        filepos;
```

```
Transform    tMatrix;
void         rotMove_start (void);

/*----------------------------------------------------------------------+
|  name    main                                                         |
+----------------------------------------------------------------------*/
Private void main (void)
{
   RscFileHandle  rfHandle;

   /* load our command tabla */
   if (mdlParse_loadCommandTable (NULL) == NULL)
      mdlOutput_error ("Unable to load command table.");
   mdlResource_openFile (&rfHandle, NULL, FALSE);
   mdlOutput_prompt ("Key-in TEXT RMS to execute");
}
/*----------------------------------------------------------------------+
|   name   getMoveDistance                                             |
+----------------------------------------------------------------------*/
Private void   getMoveDistance
(
Dpoint3d    *distance,           /* <= distance from anchor */
Dpoint3d    *pt                  /* => current point */
)
{
   Dpoint3d   anchor = statedata.dPointStack[0];

   /* subtract anchor point from current point to get the distance */
   mdlVec_subtractPoint (distance, pt, &anchor);
}
/*----------------------------------------------------------------------+
|   name      transformElement                                         |
+----------------------------------------------------------------------*/
Private void    transformElement
(
Dpoint3d    *pt          /* => current location of cursor */
)
{
   DPoint3d  distance;

   getMoveDistance (&distance, pt);
   mdlCurrTrans_begin ();
   mdlCurrTrans_identity ();
   mdlCurrTrans_translateOrigin (pt);

   mdlTMatrix_getIdentity (&tMatrix);
   mdlTMatrix_scale (&tMatrix, &tMatrix, tcb->xactscle,
                     tcb->yactscle, tcb->zactscle);
   mdlTMatrix_rotateByAngles (&tMatrix, &tMatrix, 0.0, 0.0,
                              tcb->actangle*fc_piover180);
   mdlTMatrix_translate (&tMatrix, &tMatrix, distance.x,
                         distance.y, distance.z);
   mdlElement_transform (dgnBuf, dgnBuf, &tMatrix);
   mdlCurrTrans_end ();
}
/*----------------------------------------------------------------------+
|   name  elementModify_move                                           |
+----------------------------------------------------------------------*/
Private int    elementModify_move
(
MSElementUnion  *el,      /* <> element to be modified */
Dpoint3d        *pt       /* => fram params in mdlModify_element... */
```

```
)
{
   Dpoint3d    distance;

   getMoveDistance (&distance, pt);
   mdlCurrTrans_begin ();
   mdlCurrTrans_identity ();
   mdlCurrTrans_translateOrigin (pt);
   /* start from a known matrix */
   mdlTMatrix_getIdentity (&tMatrix);

   /* scale it by current active scale */
   mdlTMatrix_scale(&tMatrix, &tMatrix, tcb->xactscle,
                   tcb->yactscle, tcb->xactscle);

   /* rotate it by current active angle */
   mdlTMatrix_rotateByAngles(&tMatrix, &tMatrix, 0.0, 0.0,
                             tcb->actangle*fc_piover180);
   mdlTMatrix_translate (&tMatrix, &tMatrix, distance.x,
                         distance.y, distance.z);

   if (mdlElement_transform (el, el, &tMatrix))
      return  MODIFY_STATUS_ERROR;
   mdlCurrTrans_end ();
   return  MODIFY_STATUS_REPLACE;
 }
/*-------------------------------------------------------------------------+
|   name   rotMove_accept                                                  |
+-------------------------------------------------------------------------*/
Private void    rotMove_accept
(
Dpoint3d    *pt       /* => final point for move element */
)
{
   DgnModelRefP   modelRef;
   ULong          filePos;

   /* Get the file position and modelRef of the element to move. */
   filePos = mdlElement_getFilePos (FILEPOS_CURRENT, &modelRef);
   /* Now move each element, no selection set */
   mdlModify_elementSingle (modelRef, filePos, MODIFY_REQUEST_HEADERS,
                            MODIFY_ORIG, elementModify_move, pt, NULL);
   /* Save new anchor point (the current acceptance point) */
   statedata.dPointStack[0] = *pt;
   /* Reload the dynamic buffer with the new element. */
   mdlDynamic_loadElement (NULL, modelRef, filePos);
   /* a RESET will restart this command */
   mdlState_setFunction (STATE_RESET, rotMove_start);
   /* if in singleshot mode then restart default command */
   mdlState_checkSingleShot ();
}
/*-------------------------------------------------------------------------+
|   name     rotMove_start                                                 |
+-------------------------------------------------------------------------*/
cmdName  void  rotMove_start (void)
cmdNumber   CMD_TEXT_RMS
{
    static int searchType[] = {
                               TEXT_ELM,
                               TEXT_NODE_ELM
                               };
    mdlLocate_noElemNoLocked ();
```

```
    mdlLocate_setElemSearchMask (sizeof(searchType)/sizeof(int), searchType);
    mdlState_startModifyCommand (rotMove_start, rotMove_accept,
                                 transformElement, NULL, NULL, 1, 0, TRUE, 0);
    mdlLocate_init ();
}
```

这个制作文件 (makefile) 使用了标准的 seed.mke, 用 textrms 来代替所有出现的 seed。键入 bmake -a textrms 建立这个应用。

```
#--------------------------------------------------------------------
#       TEXTRMS MDL Make File
#--------------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------------
#       Define constants specific to this textrms application
#--------------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

textrmsObjs = $(o)textrms.mo
textrmsRscs = $(o)textrms.rsc  $(o)textrms.mp

#-------------------------------------------------------------------------
# Create needed output directories if they don't exist
#-------------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#-----------------------------------------
#       Generate Command Tables
#-----------------------------------------
$(privateInc)textrms.h      :   $(baseDir)textrms.r

#-----------------------------------------
#       Compile Resources
#-----------------------------------------
$(o)textrms.rsc             :   $(baseDir)textrms.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)textrms.mo              :   $(baseDir)textrms.mc

$(o)textrms.mp              :   $(textrmsObjs)
            $(msg)
            >$(o)temp.cmd
            -a$@
            $(linkOpts)
            $(textrmsObjs)
            <
            $(MLinkCmd) @$(o)temp.cmd
            ~time

#-----------------------------------------
#       Merge Objects into one file
#-----------------------------------------
$(mdlapps)textrms.ma       : $(textrmsRscs)
            $(msg)
            >$(o)temp.cmd
            -o$@
            $(textrmsRscs)
            <
            $(RLibCmd) @$(o)temp.cmd
```

```
            ~time
```

文件 textrms.r 含有命令表和提示信息。

```
/*----------------------------------------------------------------------+
| TEXTRMS.R commandtable and messages for TEXTRMS                       |
+----------------------------------------------------------------------*/
#include "rscdefs.h"
#include "cmdclass.h"

#define    CT_NONE    0
#define    CT_TEXT    1
#define    CT_RMS     2

Table      CT_TEXT=
{
   {1,  CT_RMS,    MANIPULATION,REQ,    "TEXT"},
};
Table  CT_RMS=
{
   {1,  CT_NONE,    INHERIT,   REQ,    "RMS"},
};
MessageList 0=
{
   {
   {0,""  },
   {1,"Text Rotate Move Scale"},
   }
};
```

6.6 旋转矩阵函数

旋转矩阵是变换矩阵的一个子集。这种处理和我们旋转元素的过程是相同的。看看下面围绕点 P1 旋转元素的过程(见图 6.8)。我们将把元素平移到原点，旋转它，然后再把它移回到原来的 P1 点。在这个过程中，平移元素使用 mdlCurrTrans_函数，旋转则使用 mdlMatrix_函数。

旋转矩阵函数的摘要列在这里：

mdlRMatrix_from3Points	从一个平面上的三个点返回一个旋转矩阵
mdlRMatrix_fromAngle	从角返回一个旋转矩阵
mdlRMatrix_fromColumnVector	返回一个旋转矩阵，这个矩阵基于定义列的向量
mdlRMatrix_fromNormalVector	从一个方向矢量返回一个旋转矩阵
mdlRMatrix_fromRowVector	返回一个旋转矩阵，这个矩阵基于定义行的向量
mdlRMatrix_fromQuat	从四元数中提取旋转矩阵
mdlRMatrix_fromTMatrix	从变换矩阵中提取旋转和比例信息，并放入旋转矩阵
mdlRMatrix_fromView	返回一个视图或当前辅助坐标系的旋转矩阵
mdlRMatrix_getColumnVector	从一个旋转矩阵中提取一列放到一个向量中
mdlRMatrix_getIdentity	把一个旋转矩阵设置为单位矩阵
mdlRMatrix_getInverse	返回旋转矩阵的逆矩阵
mdlRMatrix_getRowVector	从一个旋转矩阵中提取一行放入一个向量中
mdlRMatrix_transpose	返回旋转矩阵的位移
mdlRMatrix_multiply	两个矩阵相乘

mdlRMatrix_multiplyByMatrix	把一个变换矩阵和一个旋转矩阵相乘
mdlRMatrix_normalize	使旋转矩阵的列标准化
mdlRMatrix_rotate	根据角度转动旋转矩阵
mdlRMatrix_multiplyPointArray	把一组点按旋转矩阵转动
mdlRMatrix_multiplyRange	在旋转过的坐标系中缩放并设置范围立方体
mdlRMatrix_toAngle	返回一个二维旋转矩阵旋转角度
mdlRMatrix_toQuat	从一个三维旋转矩阵返回一个四元数
mdlRMatrix_multiplyTransposePoint	用旋转矩阵的逆矩阵旋转一个点
mdlRMatrix_multiplyTransposePointArray	用旋转矩阵的逆矩阵旋转一组点

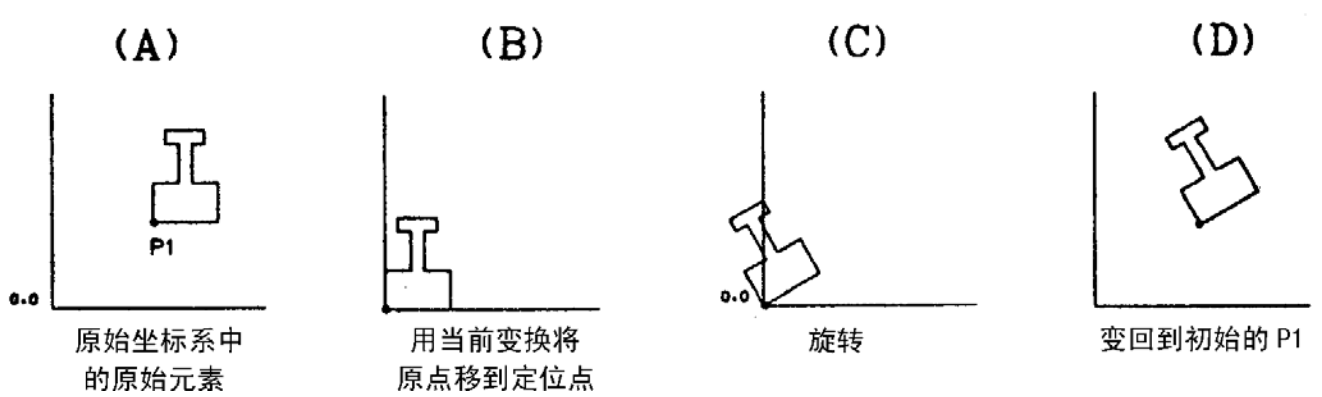


图 6.8　围绕 P1 旋转

为了解释旋转矩阵，让我们看看例子 rotcell.mc。在这个应用中，我们旋转一个单元，使它沿着激活角度放置。例如，一个单元原来以 25° 角放置，然后我们把激活角改为 45°，并调用　rotcell，该单元将被修改为以 45° 角放置。

大部分程序和第四章中的 modsing 相同。我们将使用 mdlModify_elementMulti 调用 rotCell 处理单元头及其组成元素。有了单元头，我们将 mdlCell_extract 提取旋转矩阵及旋转点。

```
mdlCell_extract (&rotPoint, NULL, &rMatrix, NULL, NULL, el);
```

我们需要建立与指定的角度对齐的变换矩阵。最好的方法是先将单元转到 0°，然后再根据激活角度旋转它。函数 mdlRMatrix_transpose 将交换旋转矩阵的行和列(实际上不旋转回去)。例如，如果一个单元旋转了 45°，调用函数 mdlRMatrix_transpose 将变换矩阵向回转 45°(即旋转-45°)。

以下编码把变换矩阵设为一种已知状态(单位矩阵)。然后我们计算单元反向旋转的角度。在调用 mdlTMatrix_rotateByRMatrix 后，变换矩阵将使单元 0° 放置的旋转量。

```
mdlTMatrix_getIdentity (&tMatrix);
mdlRMatrlx_transpose (&rMatrix, &rMatrix);
mdlTMatrix_rotateByMatrix (&tMatrix, &tMatrix, &rMatrix);
```

下一步我们需要决定旋转是在局部还是在整个坐标系中进行。如果我们在局部坐标系中工作，则需要考虑视图旋转。函数 mdlTMatrix_fromView 在我们选择单元的视图中抽取旋转矩阵。调用 mdlRMatrix_transpose 使视图向回转，mdlRMatrix_rotateByMatrix 在我们的变换矩阵中生成视图旋转信息。

```
    mdlRMatrix_fromView (&rMatrix, rotView, FALSE);
    mdlRMatrix_transpose (&rMatrix, &rMatrix);
    mdlTMatrix_rotateByRMatrix (&tMatrix, &tMatrix, &rMatrix);
```

最后，我们根据激活角度旋转变换矩阵：

```
    mdlTMatrix_rotateByAngles (&tMatrix, &tMatrix, 0.0, 0.0,
                               tcb->actangle*fc_pioverl80);
```

现在有了变换矩阵，我们可以把它用于我们的元素。使用当前转换把旋转原点设在 rotPoint。

```
    mdlCurrTrans_begin ();
    mdlCurrTrans_identity ();
    mdlCurrTrans_translateOriginWorld (&rotPoint);
    status = mdlElement_transform (el, el, &tmatrix);
    mdlCurrTrans_end ();
```

把所有这些编码段组合在一起，我们得到了旋转单元的 MDL 应用。下面给出完整的源码。键入 mdl l rotcell 装入这个应用，键入 ROTATE CELL 执行这个应用。要能看到该应用的执行效果，应当保证激活角度和您预操作的单元的放置角度不同。

```
/*----------------------------------------------------------------+
| Copyright(c)1991 Mach Dinh-Vu, All Rights Reserved              |
| rotcell.mc - rotate the cell so that it aligns with the         |
| active angle                                                    |
|                     ROTATE CELL                                 |
| UpgradeToV8: MicroStationFan          2006/06                   |
+----------------------------------------------------------------*/
#include <mdl.h>
#include <msrmgr.h>
#include <tcb.h>
#include <mscell.fdf>
#include <mscurrtr.fdf>
#include <mselemen.fdf>
#include <mslocate.fdf>
#include <mstmatrx.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msrmatrx.fdf>
#include <msselect.fdf>
#include <msstate.fdf>
#include <msvec.fdf>
#include "rotcell.h"

int    rotView;
/*----------------------------------------------------------------+
|  name    main                                                   |
+----------------------------------------------------------------*/
Private void main (void)
{
   RscFileHandle  rfHandle;

   /*------load our command table  --------*/
   if (mdlParse_loadCommandTable(NULL) == NULL)
      mdlOutput_error ("Unable to load command table.");
   mdlOutput_prompt ("to execute, key-in ROTATE CELL");
   mdlResource_openFile (&rfHandle, NULL, FALSE);
}
/*----------------------------------------------------------------+
|    name         setSearchType                                   |
+----------------------------------------------------------------*/
Private void setSearchType (void)
{
   static int searchType[] = {CELL_HEADER_ELM, LINE_ELM, LINE_STRING_ELM,
                     SHAPE_ELM, TEXT_NODE_ELM, CURVE_ELM, CMPLX_STRING_ELM,
                     CONIC_ELM, CMPLX_SHAPE_ELM, ELLIPSE_ELM,
```

```
                    TEXT_ELM, SURFACE_ELM, SOLID_ELM};

   /* initialize search criteria to find nothing */
   mdlLocate_noElemNoLocked ();
   /* add elements to search to list */
   mdlLocate_setElemSearchMask (sizeof(searchType)/sizeof(int), searchType);
}
/*----------------------------------------------------------------------------+
|name          rotCell                                                        |
+----------------------------------------------------------------------------*/
Private int rotCell ( MSElementUnion *el )
{
   Dpoint3d    rotPoint;
   RotMatrix   rMatrix;
   Transform   tMatrix;
   int         status;

   /*   build transformation matrix from the cell header
     for each component element.                    */
   if (mdlElement_isComplexHeader (el))
   {
      if (mdlElement_getType(el) != CELL_HEADER_ELM)
         return MODIFY_STATUS_NOCHANGE;
      /* get origin for the rotation point, and rotation matrix */
      mdlCell_extract (&rotPoint, NULL, &rMatrix, NULL, NULL, 0, el);
      /* build transformation matrix with the rotation component
         of the cell.                                          */
      mdlTMatrix_getIdentity (&tMatrix);
      mdlRMatrix_transpose (&rMatrix, &rMatrix);
      mdlTMatrix_rotateByRMatrix (&tMatrix, &tMatrix, &rMatrix);
      /* compensate for view rotation so the transformation is local */
      mdlRMatrix_fromView (&rMatrix, rotView, FALSE);
      mdlRMatrix_transpose (&rMatrix, &rMatrix);
      mdlTMatrix_rotateByRMatrix (&tMatrix, &tMatrix, &rMatrix);
      mdlTMatrix_rotateByAngles (&tMatrix, &tMatrix, 0.0, 0.0,
                                 tcb->actangle* fc_piover180);
   }
   mdlCurrTrans_begin ();
   mdlCurrTrans_identity ();
   mdlCurrTrans_translateOriginWorld (&rotPoint);
   status = mdlElement_transform (el, el, &tMatrix);
   mdlCurrTrans_end ();
   return (status ? 0 : MODIFY_STATUS_REPLACE);
}
/*----------------------------------------------------------------------------+
|   name   mod_accept                                                         |
+----------------------------------------------------------------------------*/
Private void     mod_accept
(
Dpoint3d  *pt,
int       view
)
{
   ULong          filePos;
   DgnModelRefP   modelRef;

   rotView = view;
   filePos = mdlElement_getFilePos (FILEPOS_CURRENT, &modelRef);
   if (mdlSelect_isActive())
   {
      mdlModify_elementMulti (modelRef,      // modelRef to process
                              filePos,       // file position for element
```

```
                    MODIFY_REQUEST_HEADERS, // process complex headers
                    MODIFY_ORIG,            // modify original element
                    rotCell,      // modify routine for each element
                    NULL,         // parameters for rotCell
                    TRUE);        // process graphic group
   } else {
      mdlModify_elementSingle (modelRef,
                               filePos,      // file position for element
                    MODIFY_REQUEST_HEADERS, // process complex headers
                    MODIFY_ORIG,            // modify original element
                    rotCell,      // modify routine for each element
                    NULL,         // parameters for rotCell
                    FALSE);       // offset for component element
   }
   /* restart the element location process */
   mdlLocate_restart (FALSE);
}
/*----------------------------------------------------------------------+
| name        change_cell                                               |
+----------------------------------------------------------------------*/
Private void change_cell (void)
{
   setSearchType ();
   mdlState_startModifyCommand (
            change_cell,   /* function to call on RESET */
            mod_accept,    /* function to call on DATA */
            NULL,          /* function to call for DYNAMIC */
            NULL,          /* function to call on SHOW */
            NULL,          /* function to call on CLEAN */
            1,             /* index into MESSAGE LIST */
            2,             /* index into PROMPT LIST */
            TRUE,            /* Modify SELECTION SET ? */
            FALSE);      /* additional data points required */
   /* start element search from the beginning of file */
   mdlLocate_init ();
}
/*----------------------------------------------------------------------+
|    name         rotcell                                               |
+----------------------------------------------------------------------*/
cmdName  void rotcell (void)
cmdNumber  CMD_ROTATE_CELL
{
   change_cell ();
}
```

文件 rotcell.r 含有 rotcell 应用的命令表和信息。

```
/*----------------------------------------------------------------------+
 |    ROTCELL.R                                                         |
 +----------------------------------------------------------------------*/
 #include  "rscdefs.h"
 #include  "cmdclass.h"

 #define    CT_NONE      0
 #define    CT_ROTATE    1
 #define    CT_CELL      2

 Table  CT_ROTATE =
 {
      { 1,  CT_CELL,    PLACEMENT,      REQ,       "ROTATE" },
 };
 Table  CT_CELL =
 {
      { 1,  CT_NONE,    INHERIT,     DEF,        "CELL" },
```

```
};
MessageList 0 =
{
      {
      { 1, "Rotate Cell" },
      { 2, "Accept/Reject element" },
      }
};
```

键入 bmake -a rotcell 建立这个应用。

```
#--------------------------------------------------------------
#       ROTCELL MDL Make File
#--------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------
#       Define constants specific to this rotcell application
#--------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

rotcellObjs = $(o)rotcell.mo
rotcellRscs = $(o)rotcell.rsc  $(o)rotcell.mp

#-----------------------------------------------------------------------
# Create needed output directories if they don't exist
#-----------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#-----------------------------------------
#       Generate Command Tables
#-----------------------------------------
$(privateInc)rotcell.h      :   $(baseDir)rotcell.r

#-----------------------------------------
#       Compile Resources
#-----------------------------------------
$(o)rotcell.rsc             :   $(baseDir)rotcell.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)rotcell.mo              :   $(baseDir)rotcell.mc

$(o)rotcell.mp              :   $(rotcellObjs)
            $(msg)
            >$(o)temp.cmd
            -a$@
            $(linkOpts)
            $(rotcellObjs)
            <
            $(MLinkCmd) @$(o)temp.cmd
            ~time

#-----------------------------------------
#       Merge Objects into one file
#-----------------------------------------
$(mdlapps)rotcell.ma       : $(rotcellRscs)
            $(msg)
            >$(o)temp.cmd
            -o$@
            $(rotcellRscs)
            <
```

```
	$(RLibCmd) @$(o)temp.cmd
	~time
```

【注】: 本章所有源代码都在 MicroStation V8 2004 下调试通过。

第七章　高级对话框

我们已经看到，对话框是一种功能强大的用户界面。在这一章中我们将讨论一些高级对话框功能。

我们将写一个应用 LOCELE 来探讨模态对话框、同义资源及对话条目通讯的用法。

应用 LOCELE 的目的是显示用搜寻标准规定的一个元素。每一个标准与一个开关按钮相连。如果开关按钮打开，LOCELE 则去查寻符合规定标准的元素位置。例如，如果层锁的开关打开，则 LOCELE 只返回那些指定类型的在层掩码规定的层内的元素。

图 7.1 显示了与这个应用相联系的对话框。

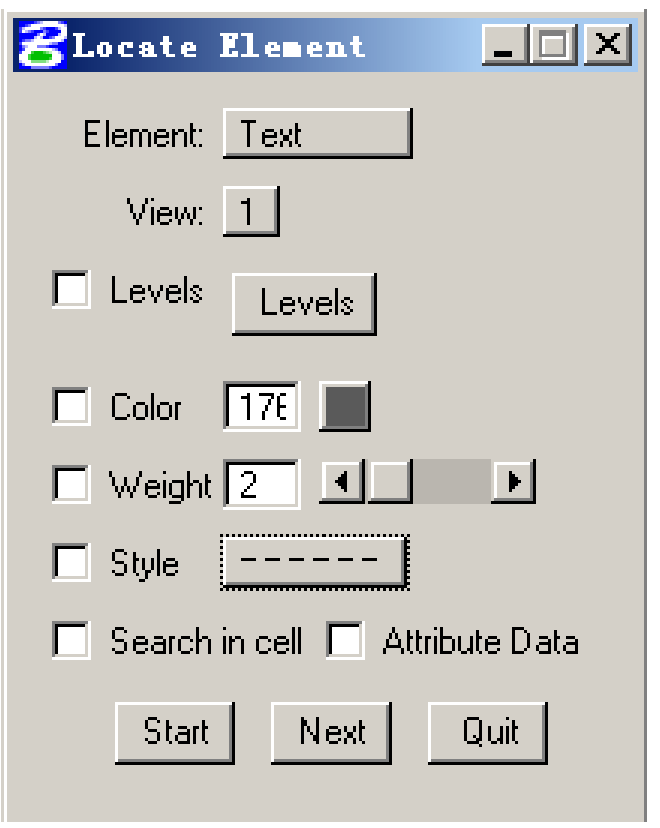


图 7.1 LOCELE 对话框

LOCELE 内使用的文件：

Locele.mc	MDL 源文件
Loceldlg.r	对话资源源文件
Loceldlg.h	含有对话类型的包含文件
Locelmsg.r	屏幕信息
Locelcmd.r	命令语法
Loceltyp.mt	要发布的数据结构
Locele.mke	这个应用的制作文件

设计 LOCELE 应用的第一步是决定需要哪些标准（因此，还要决定哪些放在对话框内）。我们要使用下面的结构 LocElemInfoP：

```
/*+----------------------------------------------------------------------+
|      Local Structure Definitions                                       |
+----------------------------------------------------------------------*/
typedef struct loceleminfo
       {
       int     eleType;              /* element to search */
       int     view;                 /* view to display element */
```

```
    int     colors;             /* boolean for toggle */
    int     weights;            /* boolean for toggle */
    int     styles;             /* boolean for toggle */
    int     levels;             /* boolean for toggle */
    int     cellElem;           /* boolean for toggle */
    int     attribData;         /* boolean for toggle */
    int     color;              /* search color */
    int     weight;             /* search weight */
    int     style;              /* search style */
    short   level [4];          /* search levels */
    }
    LocElemInfo;
```

7.1 对话条目

7.1.1 选项按钮条目

选项按钮条目用于三个方面——选择要搜寻的元素、选择显示元素的视图和要搜寻的线型。选择元素 (Element) 选项将返回 locElemlnfoP->eleType 中的元素类型(见图 7.2)。在我们的对话资源文件 loceldlg.r 中，我们把元素搜寻的资源定义为:

```
DItem_OptionButtonRsc OPTIONBUTTONID_EleType =
    {
NOSYNONYM, NOHELP, MHELP, NOHOOK, NOARG,
"Element:",
"locElemInfoP->eleType",
{
{NOTYPE, NOICON, NOCMD, LCMD,  2, NOMASK, ON, "Cell"},
{NOTYPE, NOICON, NOCMD, LCMD,  3, NOMASK, ON, "Line"},
{NOTYPE, NOICON, NOCMD, LCMD,  4, NOMASK, ON, "Line String"},
{NOTYPE, NOICON, NOCMD, LCMD,  6, NOMASK, ON, "Shape"},
{NOTYPE, NOICON, NOCMD, LCMD,  7, NOMASK, ON, "Text Node"},
{NOTYPE, NOICON, NOCMD, LCMD, 11, NOMASK, ON, "Curve"},
{NOTYPE, NOICON, NOCMD, LCMD, 13, NOMASK, ON, "Conic"},
{NOTYPE, NOICON, NOCMD, LCMD, 15, NOMASK, ON, "Ellipse"},
{NOTYPE, NOICON, NOCMD, LCMD, 16, NOMASK, ON, "Arc"},
{NOTYPE, NOICON, NOCMD, LCMD, 17, NOMASK, ON, "Text"},
}
};
```

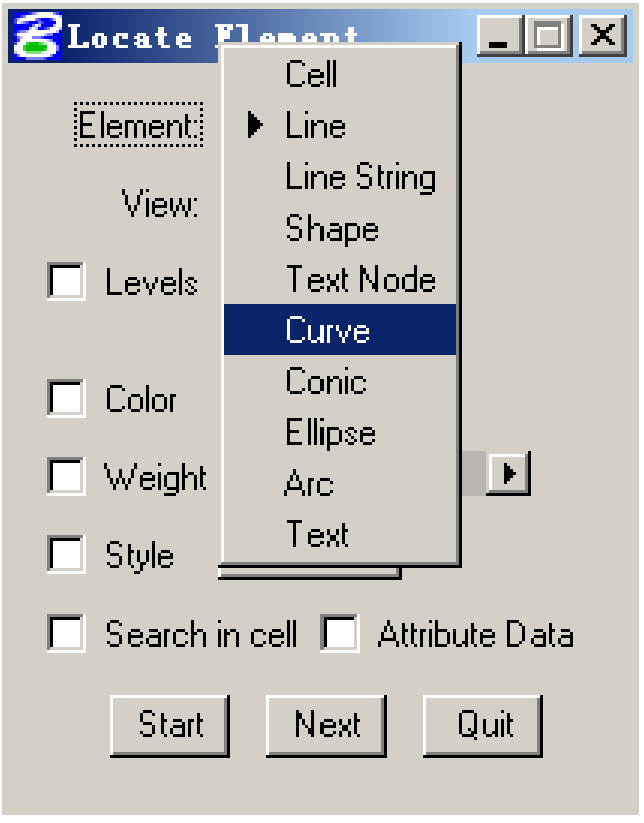


图 7.2 选择元素类型

此外，我们将使用选项按钮返回我们希望在其中显示所搜索的元素的视图（见图 7.3），其资源如下：

```
DItem_OptionButtonRsc OPTIONBUTTONID_View =
        {
        NOSYNONYM, NOHELP, MHELP, NOHOOK, NOARG,
        "View:",
        "locElemInfoP->view",
        {
        {NOTYPE, NOICON, NOCMD, LCMD,  1, NOMASK, ON,  "1"},
        {NOTYPE, NOICON, NOCMD, LCMD,  2, NOMASK, ON,  "2"},
        {NOTYPE, NOICON, NOCMD, LCMD,  3, NOMASK, ON,  "3"},
        {NOTYPE, NOICON, NOCMD, LCMD,  4, NOMASK, ON,  "4"},
        {NOTYPE, NOICON, NOCMD, LCMD,  5, NOMASK, ON,  "5"},
        {NOTYPE, NOICON, NOCMD, LCMD,  6, NOMASK, ON,  "6"},
        {NOTYPE, NOICON, NOCMD, LCMD,  7, NOMASK, ON,  "7"},
        {NOTYPE, NOICON, NOCMD, LCMD,  8, NOMASK, ON,  "8"},
        }
        };
```

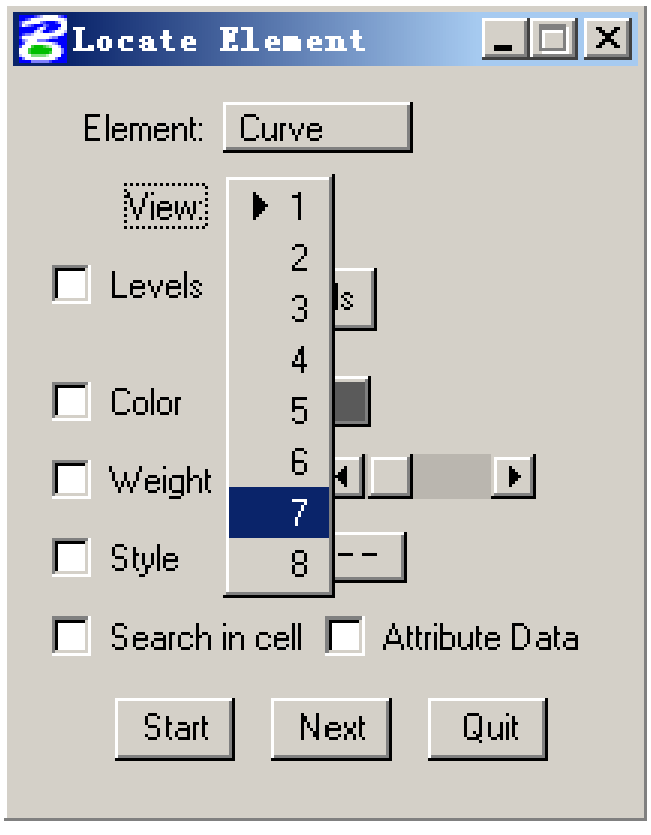


图 7.3 选择视图

为了选择元素线型，我们将使用图象块来表示这个选项(见图 7.4)，图象块定义在包含文件 <dlogids.h> 中。

```
DItem_OptionButtonRsc OPTIONBUTTONID_LoceleStyle =
        {
        NOSYNONYM, NOHELP, MHELP, NOHOOK, NOARG, "",
        "locElemInfoP->style",
        {
        {Icon, ICONID_LineStyle0, NOCMD, LCMD, 0, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle1, NOCMD, LCMD, 1, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle2, NOCMD, LCMD, 2, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle3, NOCMD, LCMD, 3, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle4, NOCMD, LCMD, 4, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle5, NOCMD, LCMD, 5, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle6, NOCMD, LCMD, 6, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle7, NOCMD, LCMD, 7, NOMASK, ON, ""},
        }
        };
```

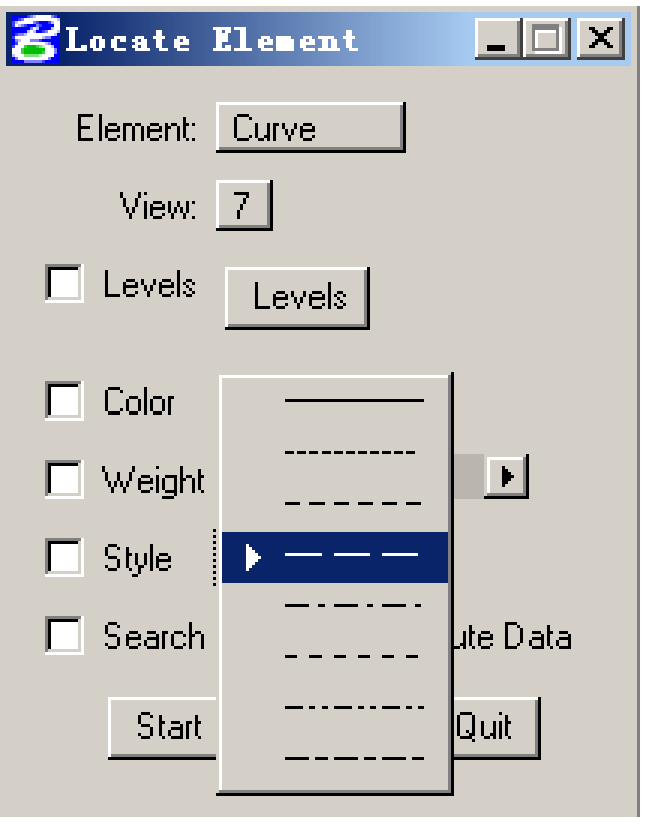


图 7.4 选择元素的线型

7.1.2 层映像条目

层映像用一个 8×8 的图像块矩阵来显示设计文件的层状态。层号周围显示黑框表示这个层是打开的。层号周围显示黑圈表示该层为激活层。

在 LOCELE 应用中，我们用层映像来表示要搜寻的层，我们不使用或不设置激活层，因此，不显示黑圈。在图 7.5 中我们指定的搜寻层是 1，35，43，53，54 和 58。（【注】：V8 中虽然也支持这种最多 63 层的层映像条目，但 V8 已经扩展为可支持任意多个层。所以，如果在 V8 下设计新的应用时将不再使用这种旧式的层映像条目而改用新的层条目。）

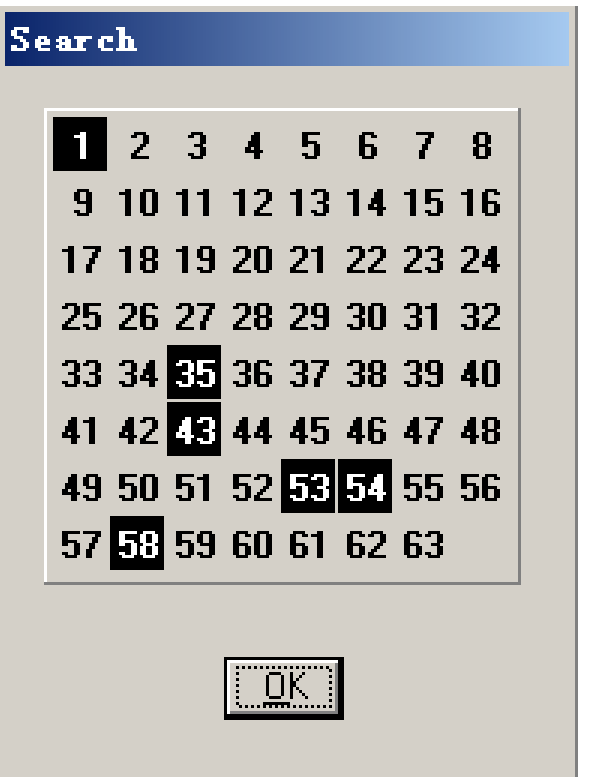


图 7.5 层映像对话条目

下边一行是对我们映像条目的说明：

```
{{X3, Y1, 0, 0}, LevelMap, LEVELMAPID_Levels, ON, 0, "","",},
```

层映像(LevelMap)资源对话框条目的结构由结构 DItem_LevelMapRsc 定义。

```
typedef struct ditem_levelmaprsc
    {
    ULong  helpInfo;
```

```
    ULong  helpSource;
    #if defined (resource)
        char    label[];
        char    accessStr[];
        char    activeLevelAccessStr[];
    #else
       long    labelLength;
       char    label[1];
    #endif
    }DItem_LevelMapRsc;
```

在我们的对话资源文件 Loceldlg.r 中，我们把层映像资源定义为：

```
DItem_LevelMapRsc LEVELMAPID_Levels =
      {
      NOHELP, MHELP,
      "Levels" ,
      "locElemInfoP->level",
      ""
      };
```

由于我们不设激活层，所以不提供 activelevelAccessStr。用于层映像的 accessStr 是 locElemInfoP->level，它是一个包括四个短整形数的数组，每一位指定一个层，图 7.6 说明层掩码是如何工作的。

变量 locElemInfoP->level 定义为：

```
typedef struct loceleminfo
{

short  level[4];
}
LocElemInfo;
```

在第三章的命令条目资源域一节中讨论了层映像应用的其它方面。

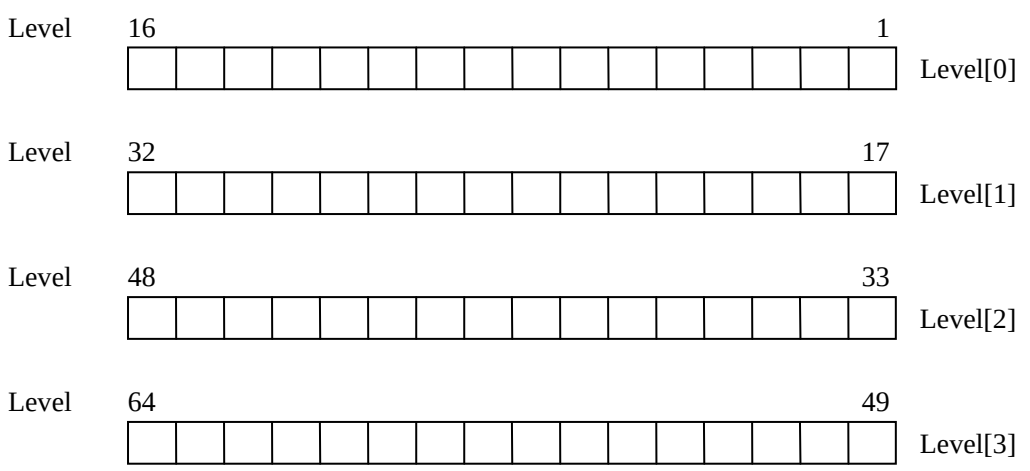


图 7.6 使用层掩码

7.1.3 颜色拾取器条目

颜色拾取器条目用一个斜削的方框显示当前颜色，颜色本身是从具有 255 种颜色调色板中选取的。

下面是对我们的颜色拾取条目的条目列表说明：

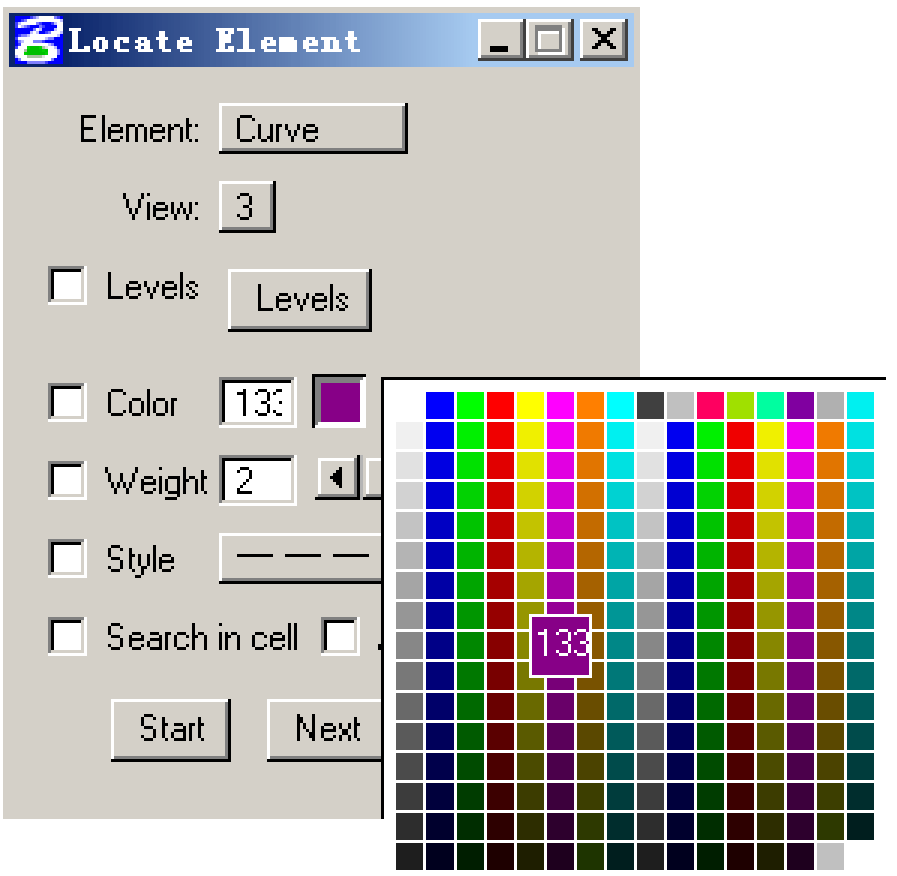


图 7.7 颜色拾取器对话条目

颜色拾取器资源对话条目由结构表 ditem_colorpickerrsc 定义。

```
typedef struct  ditem_colorpickerrsc
{
    ULong    commandNumber;
    ULong    commandSource;
    long     synonymsId;
        ULong    helpInfo;
        ULong    helpSource;
        long     itemHookld;
        long     itemHookArg;
        long     associatedTextId;
        ULong    mask;
    #if defined (resource)
        char     label[];
        char     accessStr[];
    #else
        long     labelLength;
        char     label[1];
    #endif
    }DItem_ColorPickerRsc;
```

在我们的资源文件 loceldlg.r 中，我们把颜色拾取器资源定义为：

```
DItem_ColorPickerRsc COLORPICKERID_Color =
     {
     NOCMD, MCMD, SYNONYMID_ColorLocele, NOHELP, MHELP, NOHOOK, NOARG,
     TEXTID_LoceleColor, NOMASK, "", "locElemInfoP->color"
     };
```

同义资源条目

当一个对话框条目修改时，用同义 ID，SYNONYMID_ColorLocele，来定义与之相应的对话条目。

```
DItem_SynonymsRsc SYNONYMID_ColorLocele =
     {
     {
     {Text,         TEXTID_LoceleColor},
     {ColorPicker, COLORPICKERID_Color},
     }
     };
```

在文本条目中我们定义它的同义 ID 指向 SYNONYMID_ColorLocele。在我们改变文本的值时，在斜削的方框中将显示出相应的颜色。同样地，当我们在调色板中拖动光标时，在文本域中显示相应的颜色编号。

```
DItem_TextRsc TEXTID_LoceleColor =
      {
      NOCMD, LCMD,  SYNONYMID_ColorLocele, NOHELP, MHELP,
      NOHOOK,  NOARG,
      3,  "%-1d",  "%1d",  "0",  "253",  NOMASK,  NOCONCAT,
      "",
      "locElemInfoP->color"
      };
```

7.1.4 滚动条条目

滚动条显示一个滑块，滑块可以在两个值的范围内移动。我们用滚动条来设置搜寻元素的线宽。如果 height 为零，则滚动条的方向是水平的；如果 width 为零，则产生竖直的滚动条。在我们的例子中，我们把 width 设为 SBW，height 设为零，由此生成了一个水平条。

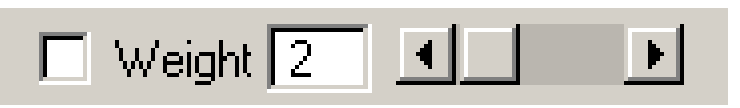


图 7.8 滚动条和文本条目

下面一行是对我们滚动条条目的条目列表说明：

```
{{X4, Y5, SBW, 0}, ScrollBar, SCROLLBARID_Weight, ON, 0, "","},
```

滚动条资源对话条目的结构由结构 ditem_scrollbarrsc 定义。

```
typedef struct ditem_scrollbarrsc
    {
    long    itemHookId;
    long    itemHookArg;
    int     minValue;
    int     maxValue;
    int     incAmount;
    int     pageIncAmount;
    double  sliderSize;
#if defined (resource)
    char    accessStr[];
#else
    long    accessStrLength;
    char    accessStr[1];
```

```
#endif
    } DItem_ScrollBarRsc;
```

我们要使对话框在滑块移动时显示出相应的线宽值。遗憾的是 ditem_scrollbarrsc 不提供一个同义 ID 字段。不要担心，我们可以用一个对话条目钩函数 HOOKITEMID_ScrollBar 来解决这个问题，我们在下一节讨论这个函数。

minValue 和 maxValue 设置线宽的范围，我们分别指定 0 和 31 作为一个元素可以具有的最小和最大线宽值。

incAmount 是按动箭头时要改变的线宽量。我们已经设定，按右箭头时线宽递增 1，按左箭头时递减 1。pageIncAmount 是当我们按滚动条页面区域时改变线宽的量。

sliderSize 是滑块的尺寸，它与整个滚动条有关。我们使滑块的宽度等于滚动条宽度的十分之一。滑块的高度是滚动条的高。在对话资源文件 loceldlg.r 中，我们把滚动条资源定义为：

```
DItem_ScrollBarRsc SCROLLBARID_Weight =
     {
     HOOKITEMID_ScrollBar, NOARG,   0,   31,   1,   5,   0.1,
     "locElemInfoP->weight"
     };
```

7.2 高级对话钩函数

在第三章我们看到了一个对话钩起动一个函数的例子。在这一节我们将捕捉并解释对话条目信息，并通过调用 mdlDialog_item 函数操作对话条目。

如前边讨论过的一样，滚动条没有同义 ID 字段，但是文本条目有。这说明我们可以定义 SYNONYMID_WeightLocele，当文本条目改变时，该函数将使滑块移动，反之则让滑块停住。

```
DItem_TextRsc TEXTID_LoceleWeight =
       {
       NOCMD, LCMD, SYNONYMID_WeightLocele, NOHELP, MHELP,
       NOHOOK, NOARG,
       3,  "%-1d",  "%1d",  "0",  "31", NOMASK, NOCONCAT,
       "",
       "locElemInfoP->weight"
       };
DItem_SynonymsRsc SYNONYMID_WeightLocele =
      {
      {
      {Text,        TEXTID_LoceleWeight},
      {ScrollBar,   SCROLLBARID_Weight},
      }
      };
```

为了解决滚动条变化时使文本条目中的值相应改变的问题，我们定义了一个对话条目钩函数 HOOKITEMID_ScrollBar,它激活函数 locele_scrollBarHook。在这个函数中，我们摸拟一个同义资源。当我们移动滑块时，相应的线宽值在文本条目中显示。

对话条目状态

对话条目具有内部值和外部状态。内部值是一个简单的 C 变量，它在这个条目显示时被引用。因此内部值就是在屏幕上显示的值。

外部状态(EXTERNAL STATE)是一个我们用 accessStr 规定的数值。在我们的例子中，它是locElemInfoP->weight。在我们操作滚动条对话条目时，内部值发生改变并且外部状态相应更新。我们的问题是如何设置用于显示线宽的文本条目的内部值。使用调试程序时我们知道，当移动滑块时传递的消息是DITEM_MESSAGE_SETVALUE。

我们将捕捉这个消息并进行我们自己的处理。首先我们需要设置滚动条的外部状态。一般来说对话条目管理程序将照管它，但是当我们设立 dimP->msgUnderstood = TRUE 时，我们已经接管了这个处理。

有一些用于操作对话框条目的函数。我们感兴趣的三个是：　　mdlDialog_itemGetByTypeAndId　通过指定资源类型及 ID 返回对话条目指针

mdlDialog_itemSetState　　设置一个条目的外部状态

mdlDialog_itemSetValue　　设置一个条目的内部值

我们必须确保在操作正确的对话条目，函数 mdlDialog_itemGetByTypeAndId 将返回指向滚动条条目位置的指针。我们使用 mdlDialog_itemSetState 来设置滚动条的外部状态。

然后我们调用 mdlDialog_itemGetByTypeAndId 来返回文本条目的指针位置，我们将使用 mdlDialog_itemSetValue 设置线宽文本条目的内部值，随后对话管理程序更新屏幕上该条目的外观。

以下编码显示出我们如何接管这个处理并摸拟一个同义资源的。

```
switch (dimP->messageType)
{
   case DITEM_MESSAGE_SETVALUE:
   {
      outFieldDiP = mdlDialog_itemGetByTypeAndId (dimP->db, RTYPE_ScrollBar,
                                                  SCROLLBARID_Weight, 0);
      mdlDialog_itemSetState (NULL, dimP->db, outFieldDiP->itemIndex);
      outFieldDiP = mdlDialog_itemGetByTypeAndId (dimP->db, RTYPE_Text,
                                                  TEXTID_LoceleWeight, 0);
      sprintf (str, "%d\0", locElemInfoP->weight);
      mdlDialog_itemSetValue (NULL, 0, NULL, str,
                              dimP->db, outFieldDiP->itemIndex);
      break;
   }
```

7.2.1 模态对话框

到目前为止我们只使用了非模态对话框，它允许用户在任何给定时间与多个对话框进行交互。LOCELE 应用是这方面的例子。在这里，用户可以查找一个元素的位置，然后离开这个应用做别的事，然后再回到 LOCELE 对话框，在离开的地方继续工作。

一个模态对话框要求所有的用户界面都对它聚焦。当一个应用必须得到信息才能继续时，这是很有用的。资源 ID 的 PUSHBUTTONID_OK 和 PUSHBUTTONID_Cancel 用于消除模态对话框，我们将使用模态对话框来设置搜寻层。

起动模态对话框的按钮对话条目如下：

```
DItem_PushButtonRsc PUSHBUTTONID_OLevel =
   {
   NOT_DEFAULT_BUTTON, NOHELP, MHELP,
   HOOKITEMID_Dummy, 0, CMD_OPENMODAL, LCMD, "",
   "Levels"
   };
```

下面 locele.mc 中的编码段支持打开模态对话框的功能。

```
Public cmdName void level_openModal
(
char    *unparsedP  /* => unparsed part of command */
)
cmdNumber  CMD_OPENMODAL
{
   int lastAction;
   /* open child modal dialog box */
   if (mdlDialog_openModal (&lastAction, NULL, DIALOGID_LevelModal) )
   {
      mdlDialog_dmsgsPrint ("Unable to open modal");
      return;
   }
}
```

DIALOGID_LevelModal 的定义如下所示，注意在对话属性中 DIALOGATTR_MODAL 的使用，用它来规定这是一个模态对话框。使用标准资源 ID，即 PUSHBUTTONID_OK，可以自动管理消除模态对话框所需的所有细节。

```
/*-----------------------------------------------------------------------+
|     Level Modal Dialog, opened when PUSHBUTTONID_OLevel is activated    |
+-----------------------------------------------------------------------*/
DialogBoxRsc DIALOGID_LevelModal =
     {
     DIALOGATTR_DEFAULT | DIALOGATTR_MODAL,
     29*XC,  18*YC,
     NOHELP,  MHELP,  HOOKDIALOGID_Level,  NOPARENTID,
    "Search",
    {
    {{X3, Y1,      0, 0},      LevelMap, LEVELMAPID_Levels,     ON, 0, "",""},
    {{X8, Y7+YC, BW, 0},    PushButton, PUSHBUTTONID_OK,       ON, 0, "",""},
    }
    };
```

7.2.2 执行应用

按钮 "Start", "Next" 和 "Quit" 都有激活 LOCELE 应用的对话钩函数。"Start" 设置从设计文件开始部分搜寻，"Next" 继续从上一个文件位置搜寻，"Quit" 调用 mdlDialog_closeCommandQueue 关闭对话框。

```
switch (dimP->dialogItemP->rawItemP->itemHookArg)
{
   case 1: /* start button */
      filePos = 0L;
      locateElemString ();
      break;
   case 2: /* next button */
      locateElemString ();
```

```
        break;
    case 3: /* quit button */
        mdlDialog_closeCommandQueue (dimP->db);
        break;
}
```

7.2.3 卸下对话框

在 LOCELE 中我们将使用对话钩函数捕捉对话信息 DIALOG_MESSAGE_DESTROY, 并对 CMD_MDL_UNLOAD 进行排队，以便调用卸载函数。

```
switch (dmP->messageType)
{
    case DIALOG_MESSAGE_DESTROY:
    {
        /* unload this MDL task when the Locele Dialog is closed */
        mdlDialog_cmdNumberQueue (FALSE, CMD_MDL_UNLOAD,
                                  mdlSystem_getCurrTaskID(), TRUE);
        break;
    };
    default:
        dmP->msgUnderstood = FALSE;
        break;
}
```

7.3 元素位置

我们将使用扫描程序查找所需要的元素位置。首先，我们设置 scanList.scantype 查找 所有元素的位置。

```
scanList.scantype  = ELEMTYPE | ELEMDATA | ONEELEM;
```

我们已经设计了元素类型选项钮来返回元素的实际类型编号。因此，我们需要在元素类型搜寻掩码中设置正确的位。

```
elemMask = 1 << (((locElemInfoP->eleType)-1) % 16);
if ((locElemInfoP->eleType) < 17)
    scanList.typmask[0] = (UShort)elemMask;
else
    scanList.typmask[1] = (UShort)elemMask;
```

如果层开关“打开”，则我们告诉扫描程序只返回与我们的层掩码相匹配的元素。

```
if (locElemInfoP->levels)
{
    scanList.scantype |= LEVELS;
    scanList.levmask[0] = locElemInfoP->level[0];
    scanList.levmask[1] = locElemInfoP->level[1];
    scanList.levmask[2] = locElemInfoP->level[2];
    scanList.levmask[3] = locElemInfoP->level[3];
}
```

如果“搜寻单元”的开关“打开”，则在类型掩码中包含这个单元头。

```
if (locElemInfoP->cellElem)
   scanList.typmask[0] |= CELL_HEADER_ELM;
```

如果“属性数据”开关“打开”，则我们需要扫描程序比较元素的特性和类别。属性数据是一个特性，因此我们将通过在 scanList.clasmsk 中设置每一个位来接受所有的类别(请参考<scanner.h>)，有两个变量用于规定搜索的属性。

在 scanlist.pch.propmsk 中设置的每一位都必须有 scanList.pch.propval 中设为 1 或 0 的对应位来表示搜寻标准。第十一位是属性数据搜寻位。在 propmsk 中，我们告诉扫描程序搜寻属性数据。在 propval 中，我们希望扫描程序返回有当前属性数据的元素。

```
if (locElemInfoP->attribData)
{
   scanList.scantype |= PROPCLAS;
   scanList.clasmsk = 0xFFFF;
   scanList.pcl.propval = 0x0800;
   scanList.pch.propmsk = 0x0800;
}
```

元素线符

由于扫描程序不提供基于元素线符的搜寻，所以我们需要自己作这个测试。函数 mdlElement_getSymbology 返回元素的颜色、线宽和线型。使用简单的 if 语句我们可以排除不符合搜寻标准的元素。

```
   mdlElement_getSymbology (&color, &weight, &style, &element);
   if (locElemInfoP->colors)
      if (locElemInfoP->color != color) continue;
   if (locElemInfoP->weights)
      if (locElemInfoP->weight != weight) continue;
   if (locElemInfoP->styles)
      if (locElemInfoP->style != style) continue;
```

当我们找到了期望的元素，我们将把这个元素置于窗口并在指定视图中显示它。我们将使用两个开窗口的程序，一个用于文本，另一个用于其它元素。在我们可以看这些程序以前，我们必须考虑适用的视图函数。

```
   switch (mdlElement_getType (&element) )
   {
      case CELL_HEADER_ELM:
      case LINE_ELM:
      case LINE_STRING_ELM:
      case SHAPE_ELM:
      case CONIC_ELM:
      case CURVE_ELM:
      case ELLIPSE_ELM:
      case ARC_ELM:
           showElement (&element);
           return;
      case TEXT_ELM :
      case TEXT_NODE_ELM :
           showTextElement (&element);
```

```
            return;
        default :
            continue;
    }
```

7.4 视图函数

MDL 提供了一些在屏幕窗口内操作和显示设计文件的函数，这些函数的摘要如下：

mdlView_attachNamed	连接并显示一个存贮的视图
mdlView_deleteNamed	从文件中删去一个存贮的视图
mdlView_findNamed	在文件中搜寻一个存贮的视图
mdlView_fit	贴合一个主文件/参考文件
mdlView_getCamera	返回视图照相机设置
mdlView_getDisplayControl	返回视图显示信息
mdlView_getLevels	返回视图打开的层
mdlView_getParameters	返回视图参考信息
mdlView_getStandard	用标准视图设置旋转矩阵。(如轴侧视图、顶视图等等)
mdlView_isActive	决定视图是否打开
mdlView_isStandard	决定一个视图是否正在显示一个标准视图
mdlView_isVisible	决定我们是否可以在视图内画图
mdlView_rotateToRMatrix	用一个旋转矩阵旋转视图
mdlView_setArea	定义视图区域
mdlView_saveNamed	存贮视图信息
mdlView_setActiveDepth	从前修剪平面设置激活深度
mdlView_setActiveDepthPoint	从一个点设置激活深度
mdlView_setDisplayControl	设置视图显示信息
mdlView_setDisplayDepth	从前、后修剪平面设置显示深度
mdlView_setDisplayDepthPoints	从一组点设立显示深度
mdlView_setFunction	定义一个视图用户函数
mdlView_setLevels	打开或关闭视图的层显示
mdlView_turnOff	关上一个视图
mdlView_turnOn	打开一个视图
mdlView_updateMulti	刷新几个视图
mdlView_updateSingle	刷新单个视图
mdlView_zoom	改变视图范围

7.4.1 视图中的元素

在 LOCELE 中，函数 showElement 将用元素的范围来限制视图窗口的大小，由于所有元素都有一个范围，所以这种方法使该函数的使用很普遍。

首先要确保用户指定的视图是激活视图，使用 mdlView_isVisible 做一个简单的测试可做到这点。

```
if (!mdlView_isVisible (locElemInfoP->view - 1))
{
```

```
    mdlDialog_openAlert ("View is inactive" );
    return -1;
}
```

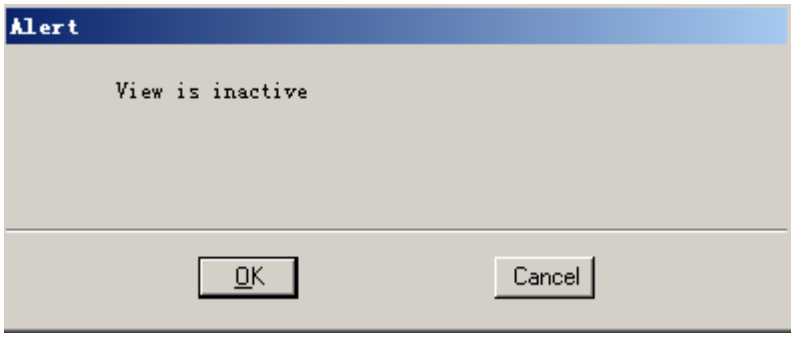


图 7.9 使用 mdlDialog_openAlert

如果我们在 3D 文件中，我们将在轴测视图中显示这个元素。

用 mdlView_getStandard 抽取视图的旋转矩阵。

```
if (tcb->ndices == 3)
    stdViewNum = STDVIEW_ISO;
else
    stdViewNum = STDVIEW_TOP;
```

调用 mdlElement_extractRange 可取得元素的范围，调用 mdlRMatrix_multiplyRange 和 mdlRMatrix_multiplyTransposePointArray 将为元素设立一个视图范围立方体(对 3D 文件)或一个视图矩形(对 2D 文件)。

```
mdlElement_extractRange ((DVector3d *)viewRange, element);
mdlRMatrix_multiplyRange (&viewRange[0], &viewRange[1 ], &rMatrix);
mdlRMatrix_multiplyTransposePointArray (viewRange, &rMatrix, 2);
```

最后，我们准备显示元素。在 3D 文件中，我们需要对 mdlView_setArea 提供显示深度范围，调用 mdlView_updateSingle 将刷新我们元素所在的视图，mdlElement_display 将在所有视图中使这些元素增加亮度，请注意，mdlElement_display 只使简单元素增加亮度。为了使复杂元素增加亮度，这个程序必须通过使用元素描述符来进行升级。

```
range = fabs (viewRange[0].z - viewRange[1].z);
mdlView_setArea (locElemInfoP->view-1, viewRange, &viewRange[0],
                 range, range/2.0, &rMatrix);
mdlView_updateSingle (locElemInfoP->view-1);
mdlElement_display (element, HILITE);
```

这对所有元素起作用。然而，在文本及文本节点情况下，最好在我们旋转视图时，文本字符永远保持水平方向，函数 showTextElement 具有这种功能。文本的旋转矩阵由 mdlText_extract 提取。我们在第六章已讨论过该矩阵的运算，通过 mdlRMatrix_transpose 使文本反向旋转为水平状态。其余的窗口命令与 showElement 的功能相同。

```
/* Find elem element's position and orientation */
mdlText_extract (&elemOrigin, NULL, NULL, NULL, NULL, &elemRMatrix,
                 NULL, NULL, NULL, &elemSize, element);
/* Set up view to show elem element */
viewRange[0].x = viewRange[0].y = -2.0 * elemSize.height;
viewRange[1].x = elemSize.width + 2.0 * elemSize.height;
viewRange[1].y = elemSize.height * 3.0;
viewRange[0].z = viewRange[1] .z = 0.0;
```

```
   mdlRMatrix_rotatePointArray (viewRange, &elemRMatrix, 2);
   mdlVec_addPointArray (viewRange, &elemOrigin, 2);
   /* Display Text element */
   mdlRMatrix_transpose (&invElemMatrix, &elemRMatrix);
   mdlView_setArea (locElemInfoP->view-1, viewRange, &viewRange[0],
                    elemSize.width, elemSize.width/2.0, &invElemMatrix);
```

LOCELE 应用的整个清单在下面给出：

```
/*----------------------------------------------------------------------------+
|             Copyright (c) 1991 Math Dinh-Vu, All Rights Reserved            |
|              locele.mc  - Display an element in a view, according to        |
|                               the search criteria.                          |
|                               LOCATE ELEMENT                                |
|  UpgradeToV8 : MicroStationFan    2005/06                                   |
+----------------------------------------------------------------------------*/
#include <cmdlist.h>
#include <mdl.h>
#include <msrmgr.h>
#include <tcb.h>
#include <stdlib/math.h>
#include <stdlib/string.h>
#include <ditemlib.fdf>
#include <mscell.fdf>
#include <mscexpr.fdf>
#include <mscurrtr.fdf>
#include <msdialog.fdf>
#include <mselemen.fdf>
#include <mslocate.fdf>
#include <mstmatrx.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msrmatrx.fdf>
#include <msscan.fdf>
#include <mssystem.fdf>
#include <msstate.fdf>
#include <msvec.fdf>
#include <msview.fdf>
#include    "locelcmd.h"  /*  Generated by resource compiler (rcomp) */
#include    "loceldlg.h"  /*  Need to know dialog id to open */

/*----------------------------------------------------------------------------+
|      Local function declarations                                            |
+----------------------------------------------------------------------------*/
void    locele_dummyItemHook(DialogItemMessage *), locele_dialogHook(DialogMessage *),
        locele_buttonHook(DialogItemMessage *), locele_scrollBarHook(DialogItemMessage *);
void    locateElem (void);

/*----------------------------------------------------------------------------+
|      Private Global variables                                               |
+----------------------------------------------------------------------------*/
static LocElemInfo   *locElemInfoP;
static int            commandName;
static ULong          filePos;
static DialogBox     *dbP;
ExtScanlist           scanList;
static DialogHookInfo uHooks[] =
      {
      {HOOKITEMID_Button_Locele, locele_buttonHook},
      {HOOKITEMID_ScrollBar,     locele_scrollBarHook},
```

```
      {HOOKITEMID_Dummy,          locele_dummyItemHook},
      {HOOKDIALOGID_Level,        locele_dialogHook},
      };

/*-------------------------------------------------------------------------+
|       name    unloadFunction                                             |
+-------------------------------------------------------------------------*/
Private int unloadFunction ()
{
   RscFileHandle   userPrefsH;
   LocElemInfo    *boxRscP;

   /* Open userpref.rsc to hold our small pref resource */
   mdlDialog_userPrefFileOpen (&userPrefsH, RSC_READWRITE);
   boxRscP = (LocElemInfo *)mdlResource_load (NULL, RTYPE_LocElem,
                                              RSCID_LocElemPrefs);
   if (!boxRscP)
   {
      /* Our pref resource does not exist, so add it */
      mdlResource_add (userPrefsH, RTYPE_LocElem, RSCID_LocElemPrefs,
                       locElemInfoP, sizeof(LocElemInfo), NULL);
   }else{
      *boxRscP = *locElemInfoP;
      /* Write out and free the updated resource */
      mdlResource_write (boxRscP);
      mdlResource_free (boxRscP);
   }
   /* Clean up */
   mdlResource_closeFile (userPrefsH);
   free (locElemInfoP);
   return (FALSE);
}
/*-------------------------------------------------------------------------+
|   name  main                                                             |
+-------------------------------------------------------------------------*/
int   main (char *pArgument)
{
   RscFileHandle   rfHandle, userPrefsH;
   LocElemInfo    *elemRscP;
   char           *setP;

   /* Open our file for access to command table and dialog */
   mdlResource_openFile (&rfHandle, NULL, FALSE);
   /* Load the command table */
   if (mdlParse_loadCommandTable(NULL) == NULL)
      mdlOutput_error ( "Unable to load command table." );
   /* Publish the dialog item hooks */
   mdlDialog_hookPublish (sizeof(uHooks)/sizeof(DialogHookInfo), uHooks);
   /* Commands start in string list 0, prompts start in string 1 */
   mdlState_registerStringIds (0, 0);
   locElemInfoP = malloc (sizeof(LocElemInfo));
   /* Prepare to read resource */
   elemRscP = NULL;
   userPrefsH = NULL;
   mdlDialog_userPrefFileOpen (&userPrefsH, RSC_READWRITE);
   if (userPrefsH)
      elemRscP = (LocElemInfo *)mdlResource_load (NULL, RTYPE_LocElem,
                                                  RSCID_LocElemPrefs );
   if (!elemRscP)
   {
      /* No resource was found */
      locElemInfoP->eleType = 3;
```

```
      locElemInfoP->color = 0;
      locElemInfoP->weight = 0;
      locElemInfoP->style = 0;
      locElemInfoP->level[0] = 0x0000;
      locElemInfoP->level[1] = 0x0000;
      locElemInfoP->level[2] = 0x0000;
      locElemInfoP->level[3] = 0x0000;
   }else{
      /* Copy resource into internal structure */
      *locElemInfoP = *elemRscP;
      /* This is unnecessary because the closeFile will free all resources,
         but it is recommended practice */
      mdlResource_free (elemRscP);
   }
   if (userPrefsH)
      mdlResource_closeFile (userPrefsH);
   /* Set up and Publish locElemInfoP for access by the dialog manager */
   setP = mdlCExpression_initializeSet (VISIBILITY_DIALOG_BOX |
                                        VISIBILITY_DEBUGGER, 0, FALSE);
   mdlDialog_publishComplexPtr (setP, "loceleminfo", "locElemInfoP",
                                &locElemInfoP);
   /* Make sure our function gets called at unload time */
   mdlSystem_setFunction (SYSTEM_UNLOAD_PROGRAM, unloadFunction);
   locateElem ();
   return  SUCCESS;
}
/*----------------------------------------------------------------------+
|   name      showElement                                               |
+----------------------------------------------------------------------*/
Private int showElement ( MSElement *element )
{
   Dpoint3d     viewRange[2];
   RotMatrix    rMatrix;
   double       range;
   int          stdViewNum;

   if (!mdlView_isVisible (locElemInfoP->view - 1))
   {
      mdlDialog_openAlert ("View is inactive" );
      return -1;
   }
   if (tcb->ndices == 3)
      stdViewNum = STDVIEW_ISO;
   else
      stdViewNum = STDVIEW_TOP;
   mdlView_getStandard (&rMatrix, stdViewNum);
   mdlElement_extractRange ((DVector3d *)viewRange, element);
   mdlRMatrix_multiplyRange (&viewRange[0], &viewRange[1 ], &rMatrix);
   mdlRMatrix_multiplyTransposePointArray (viewRange, &rMatrix, 2);
   range = fabs (viewRange[0].z - viewRange[1].z);
   mdlView_setArea (locElemInfoP->view-1, viewRange, &viewRange[0],
                    range, range/2.0, &rMatrix);
   mdlView_updateSingle (locElemInfoP->view-1);
   mdlElement_display (element, HILITE);
   return  SUCCESS;
}
/*----------------------------------------------------------------------+
|  name           showTextElement                                       |
+----------------------------------------------------------------------*/
Private int showTextElement (MSElement  *element)
{
   Dpoint3d    elemOrigin, viewRange[2];
```

```
   TextSize    elemSize;
   RotMatrix   elemRMatrix, invElemMatrix;

   /* Find elem element's position and orientation */
   mdlText_extract (&elemOrigin, NULL, NULL, NULL, NULL, &elemRMatrix,
                    NULL, NULL, NULL, &elemSize, element);
   /* Set up view to show elem element */
   viewRange[0].x = viewRange[0].y = -2.0 * elemSize.height;
   viewRange[1].x = elemSize.width + 2.0 * elemSize.height;
   viewRange[1].y = elemSize.height * 3.0;
   viewRange[0].z = viewRange[1] .z = 0.0;
   mdlRMatrix_rotatePointArray (viewRange, &elemRMatrix, 2);
   mdlVec_addPointArray (viewRange, &elemOrigin, 2);
   /* Display Text element */
   mdlRMatrix_transpose (&invElemMatrix, &elemRMatrix);
   mdlView_setArea (locElemInfoP->view-1, viewRange, &viewRange[0],
                    elemSize.width, elemSize.width/2.0, &invElemMatrix);
   mdlView_updateSingle (locElemInfoP->view-1);
   mdlElement_display (element, HILITE);
   return  SUCCESS;
}
/*-------------------------------------------------------------------------+
|   name          locateElemString                                        |
+-------------------------------------------------------------------------*/
Private void   locateElemString ( void )
{
   int            scanWords=0, status;
   int            elemMask, style;
   ULong          color, weight;
   MSElement      element;
   mdlScan_initScanlist (&scanList);
   mdlScan_noRangeCheck (&scanList);
   scanList.scantype  = ELEMTYPE | ELEMDATA | ONEELEM;
   /* if user wants to restrict search to levels then set the scanner */
   if (locElemInfoP->levels)
   {
      scanList.scantype |= LEVELS;
      scanList.levmask[0] = locElemInfoP->level[0];
      scanList.levmask[1] = locElemInfoP->level[1];
      scanList.levmask[2] = locElemInfoP->level[2];
      scanList.levmask[3] = locElemInfoP->level[3];
   }
   elemMask = 1 << (((locElemInfoP->eleType)-1) % 16);
   if ((locElemInfoP->eleType) < 17)
      scanList.typmask[0] = (UShort)elemMask;
   else
      scanList.typmask[1] = (UShort)elemMask;
   if (locElemInfoP->cellElem)
      scanList.typmask[0] |= CELL_HEADER_ELM;
   if (locElemInfoP->attribData)
   {
      scanList.scantype |= PROPCLAS;
      scanList.clasmsk = 0xFFFF;
      scanList.pcl.propval = 0x0800;
      scanList.pch.propmsk = 0x0800;
   }
   scanList.extendedType = FILEPOS;
   scanList.sector       = DGN_BLOCK (filePos);
   scanList.offset       = DGN_OFFSET(filePos);

   mdlScan_initialize (0, &scanList);
```

```
/* loop through the file until one matching the criteria is found */
while ((status = mdlScan_file (&element, &scanWords, sizeof(element),
                               &filePos)), (scanWords != 0) )
{
   mdlElement_getSymbology (&color, &weight, &style, &element);
   if (locElemInfoP->colors)
      if (locElemInfoP->color != color) continue;
   if (locElemInfoP->weights)
      if (locElemInfoP->weight != weight) continue;
   if (locElemInfoP->styles)
      if (locElemInfoP->style != style) continue;
   switch (mdlElement_getType (&element) )
   {
      case CELL_HEADER_ELM:
      case LINE_ELM:
      case LINE_STRING_ELM:
      case SHAPE_ELM:
      case CONIC_ELM:
      case CURVE_ELM:
      case ELLIPSE_ELM:
      case ARC_ELM:
           showElement (&element);
           return;
      case TEXT_ELM :
      case TEXT_NODE_ELM :
           showTextElement (&element);
           return;
      default :
           continue;
   }
}
   /* Display an alert and return result to caller */
   mdlDialog_openAlert ( "End of File ! ");
   return;
}
/*----------------------------------------------------------------------+
|    name          locele_dialogHook                                    |
+----------------------------------------------------------------------*/
Private void    locele_dialogHook
(
DialogMessage   *dmP         /* => a ptr to a dialog message */
)
{
   /* ignore any messages being sent to modal dialog hook */
   if (dmP->dialogId != DIALOGID_LocateElem)
      return;
   dmP->msgUnderstood = TRUE;
   switch (dmP->messageType)
   {
      case DIALOG_MESSAGE_DESTROY:
      {
         /* unload this MDL task when the Locele Dialog is closed */
         mdlDialog_cmdNumberQueue (FALSE, CMD_MDL_UNLOAD,
                                   mdlSystem_getCurrTaskID(), TRUE);
         break;
      };
      default:
         dmP->msgUnderstood = FALSE;
         break;
   }
}
/*----------------------------------------------------------------------+
```

```
|  name           locele_buttonHook                                          |
+--------------------------------------------------------------------------*/
Private void  locele_buttonHook (DialogItemMessage   *dimP)
{
   if ((dimP->messageType != DITEM_MESSAGE_BUTTON) ||
       (dimP->u.button.buttonTrans != BUTTONTRANS_UP))
   {
      /* Tell the dialog manager that we didn't handle this message */
      dimP->msgUnderstood = FALSE;
      return;
   }
   /* Tell the dialog manager that we are handling this message */
   dimP->msgUnderstood = TRUE;
   /*  Call mdlState_startPrimitive to terminate current command and
      identify the current command.    */
   mdlState_startPrimitive (NULL, NULL, 1, 0);
   switch (dimP->dialogItemP->rawItemP->itemHookArg)
   {
      case 1: /* start button */
         filePos = 0L;
         locateElemString ();
         break;
      case 2: /* next button */
         locateElemString ();
         break;
      case 3: /* quit button */
         mdlDialog_closeCommandQueue (dimP->db);
         break;
   }
}
/*--------------------------------------------------------------------------+
| name          locele_scrollBarHook                                       |
+--------------------------------------------------------------------------*/
Private void locele_scrollBarHook (DialogItemMessage *dimP)
{
   DialogItem *outFieldDiP;
   char       str[3];

   /* Tell the dialog manager that we are handling this message */
   dimP->msgUnderstood = TRUE;
   switch (dimP->messageType)
   {
      case DITEM_MESSAGE_SETVALUE:
      {
         outFieldDiP = mdlDialog_itemGetByTypeAndId (dimP->db, RTYPE_ScrollBar,
                                                     SCROLLBARID_Weight, 0);
         mdlDialog_itemSetState (NULL, dimP->db, outFieldDiP->itemIndex);
         outFieldDiP = mdlDialog_itemGetByTypeAndId (dimP->db, RTYPE_Text,
                                                     TEXTID_LoceleWeight, 0);
         sprintf (str, "%d\0", locElemInfoP->weight);
         mdlDialog_itemSetValue (NULL, 0, NULL, str,
                                 dimP->db, outFieldDiP->itemIndex);
         break;
      }
      default:
      {
         /* Tell the dialog manager to handle this message */
         dimP->msgUnderstood = FALSE;
         break;
      }
   }
}
```

```
/*-----------------------------------------------------------------------+
|    name      locele dummyltemHcok                                      |
+-----------------------------------------------------------------------*/
Private void locele_dummyItemHook
(
DialogItemMessage   *dimP   /* => a ptr to a dialog item message */
)
{
   dimP->msgUnderstood = TRUE;
   switch (dimP->messageType)
   {
      default:
         dimP->msgUnderstood = FALSE;
         break;
   }
}
/*-----------------------------------------------------------------------+
|    name          level_openModal                                      |
+-----------------------------------------------------------------------*/
Public cmdName void level_openModal
(
char    *unparsedP  /* => unparsed part of command */
)
cmdNumber  CMD_OPENMODAL
{
   int lastAction;
   /* open child modal dialog box */
   if (mdlDialog_openModal (&lastAction, NULL, DIALOGID_LevelModal) )
   {
      mdlDialog_dmsgsPrint ("Unable to open modal");
      return;
   }
}
/*-----------------------------------------------------------------------+
|   name          locateElem                                            |
+-----------------------------------------------------------------------*/
cmdName  void  locateElem (void)
cmdNumber  CMD_LOCATE_ELEMENT
{
   /* Initialize view */
   locElemInfoP->view = (tcb->lstvw) + 1;
   filePos = 0L;   /* start search from begining of file */
   dbP = mdlDialog_open (NULL, DIALOGID_LocateElem);
mdlDialog_openAlert ("View is inactive" );
}
```

Loceldlg.h 是这个应用的头文件，它定义了 LOCELE 应用使用的常量及数据结构。

```
/*-----------------------------------------------------------------------+
|      Dialog Box IDs                                                   |
+-----------------------------------------------------------------------*/
#define DIALOGID_LocateElem      1
#define DIALOGID_LevelModal      2

/*-----------------------------------------------------------------------+
_---+
|      Option Button Item IDs                                           |
+-----------------------------------------------------------------------*/
#define OPTIONBUTTONID_EleType          1
#define OPTIONBUTTONID_View             2
#define OPTIONBUTTONID_LoceleStyle      3

/*-----------------------------------------------------------------------+
|      Resource Type and ID for Prefs                                   |
```

```
+-----------------------------------------------------------------------*/
#define RTYPE_LocElem           'loEt'
#define RSCID_LocElemPrefs       1

/*-----------------------------------------------------------------------+
|      Elem Item IDs                                                     |
+-----------------------------------------------------------------------*/
#define TEXTID_LoceleColor       1
#define TEXTID_LoceleWeight      2

/*-----------------------------------------------------------------------+
|      Color Picker  IDs                                                 |
+-----------------------------------------------------------------------*/
#define COLORPICKERID_Color      1

/*-----------------------------------------------------------------------+
|      Scroll Bar IDs                                                    |
+-----------------------------------------------------------------------*/
#define SCROLLBARID_Weight       1

/*-----------------------------------------------------------------------+
|      LevelMap      IDs                                                 |
+-----------------------------------------------------------------------*/
#define LEVELMAPID_Levels        1
/*-----------------------------------------------------------------------+
|      Toggle Button IDS                                                 |
+-----------------------------------------------------------------------*/
#define TOGGLEID_Colors              1
#define TOGGLEID_Weights             2
#define TOGGLEID_Styles              3
#define TOGGLEID_Levels              4
#define TOGGLEID_CellElem            5
#define TOGGLEID_AttribData          6

/*-----------------------------------------------------------------------+
|      PushButton Item IDs                                               |
+-----------------------------------------------------------------------*/
#define PUSHBUTTONID_OLevel     1
#define PUSHBUTTONID_Start      2
#define PUSHBUTTONID_Next       3
#define PUSHBUTTONID_Quit       4

/*-----------------------------------------------------------------------+
|    Synonym  Id's                                                       |
+-----------------------------------------------------------------------*/
#define SYNONYMID_WeightLocele        1
#define SYNONYMID_ColorLocele         2

/*-----------------------------------------------------------------------+
|      Message Id's                                                      |
+-----------------------------------------------------------------------*/
#define STRINGID_Messages             1
#define STRINGID_Errors               2

/*-----------------------------------------------------------------------+
|      Hook  Id's                                                        |
+-----------------------------------------------------------------------*/
#define HOOKITEMID_Button_Locele        1
#define HOOKITEMID_Dummy                2
#define HOOKITEMID_ScrollBar            3
#define HOOKDIALOGID_Level              4
```

```
/*----------------------------------------------------------------------+
|      Local Structure Definitions                                      |
+----------------------------------------------------------------------*/
typedef struct loceleminfo
       {
       int     eleType;              /* element to search */
       int     view;                 /* view to display element */
       int     colors;               /* boolean for toggle */
       int     weights;              /* boolean for toggle */
       int     styles;               /* boolean for toggle */
       int     levels;               /* boolean for toggle */
       int     cellElem;             /* boolean for toggle */
       int     attribData;           /* boolean for toggle */
       int     color;                /* search color */
       int     weight;               /* search weight */
       int     style;                /* search style */
       short   level [4];            /* search levels */
       }
       LocElemInfo;
```

文件 Loceldlg.r 含有 LOCELE 应用的资源定义。

```
/*----------------------------------------------------------------------+
|     loceldlg.r  - Dialog Resources for MDL program  locele.mc         |
+----------------------------------------------------------------------*/
#include <rscdefs.h>
#include <dlogbox.h>
#include <dlogids.h>
#include "loceldlg.h"
#include "locelcmd.h"

/*----------------------------------------------------------------------+
|      Locate Elem Dialog Box                                           |
+----------------------------------------------------------------------*/
#define  OVERALLWIDTH  32 * XC
#define  OVERALLHEIGHT 19 * YC
#define  NEWLINE        2 * YC
#define  BW    6.5 * XC         /* button width - 6 chars   */
#define  SBW    11 * XC         /* scroll bar width */
#define  X1     11 * XC         /* String elem */
#define  X2     13 * XC         /* Option menus */
#define  X3      2 * XC         /* View */
#define  X4     16 * XC         /* Cell Elem */
#define  X5      5 * XC         /* left button */
#define  X6     21 * XC         /* right button */
#define  X7     28 * XC         /* modal width */
#define  X8     X7/2-(3*XC)     /* position of OK button */
#define  Y1     YC                    /* String elem item */
#define  Y2     Y1 + NEWLINE          /* EleType Option menu */
#define  Y3     Y2 + NEWLINE          /* Interactive toggle */
#define  Y4     Y3 + 3 * YC           /* Execute Buttons */
#define  Y5     Y4 + NEWLINE          /* Execute Buttons */
#define  Y6     Y5 + NEWLINE          /* Execute Buttons */
#define  Y7     Y6 + NEWLINE          /* Execute Buttons */
#define  Y8     Y7 + NEWLINE          /* Execute Buttons */
/*----------------------------------------------------------------------+
__+
|       Main Dialog                                                     |
+----------------------------------------------------------------------*/
DialogBoxRsc DIALOGID_LocateElem =
      {
      DIALOGATTR_DEFAULT | DIALOGATTR_SINKABLE,
      OVERALLWIDTH,  OVERALLHEIGHT,
      NOHELP, MHELP, HOOKDIALOGID_Level, NOPARENTID,
```

```
      "Locate Element",
      {
       {{X1, Y1, 0, 0},  OptionButton, OPTIONBUTTONID_EleType,  ON, 0, "",""},
       {{X1, Y2, 0, 0},  OptionButton, OPTIONBUTTONID_View,     ON, 0, "",""},
       {{X1, Y3, 0, 0},  PushButton,   PUSHBUTTONID_OLevel,     ON, 0, "",""},
       {{X3, Y3, 0, 0},  ToggleButton, TOGGLEID_Levels,         ON, 0, "",""},

       {{X3, Y4, 0, 0},  ToggleButton, TOGGLEID_Colors,         ON, 0, "",""},
       {{X1, Y4, 4*XC, 0},       Text, TEXTID_LoceleColor,      ON, 0, "",""},
       {{X4, Y4, 0, 0},   ColorPicker, COLORPICKERID_Color,     ON, 0, "",""},

       {{X3, Y5, 0, 0},  ToggleButton, TOGGLEID_Weights,        ON, 0, "",""},
       {{X1, Y5, 4*XC, 0},       Text, TEXTID_LoceleWeight,     ON, 0, "",""},
       {{X4, Y5, SBW, 0},   ScrollBar, SCROLLBARID_Weight,      ON, 0, "",""},

       {{X3, Y6, 0, 0},  ToggleButton, TOGGLEID_Styles,         ON, 0, "",""},
       {{X1, Y6, 0, 0},  OptionButton, OPTIONBUTTONID_LoceleStyle,
                                                                ON, 0, "",""},
       {{X3, Y7, 0, 0},  ToggleButton, TOGGLEID_CellElem,       ON, 0, "",""},
       {{X4, Y7, 0, 0},  ToggleButton, TOGGLEID_AttribData,     ON, 0, "",""},
       {{X5, Y8, BW, 0},   PushButton, PUSHBUTTONID_Start,      ON, 0, "",""},
       {{X2, Y8, BW, 0},   PushButton, PUSHBUTTONID_Next,       ON, 0, "",""},
       {{X6, Y8, BW, 0},   PushButton, PUSHBUTTONID_Quit,       ON, 0, "",""},
      }
      };
/*-------------------------------------------------------------------------+
|     Level Modal Dialog, opened when PUSHBUTTONID_OLevel is activated     |
+-------------------------------------------------------------------------*/
DialogBoxRsc DIALOGID_LevelModal =
       {
       DIALOGATTR_DEFAULT | DIALOGATTR_MODAL,
       29*XC,  18*YC,
       NOHELP,  MHELP,  HOOKDIALOGID_Level,  NOPARENTID,
      "Search",
      {
      {{X3, Y1,     0, 0},     LevelMap, LEVELMAPID_Levels,    ON, 0, "",""},
      {{X8, Y7+YC, BW, 0},   PushButton, PUSHBUTTONID_OK,      ON, 0, "",""},
      }
      };
/*-------------------------------------------------------------------------+
|    Dialog  items                                                         |
+-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------+
|    Option  Items                                                         |
+-------------------------------------------------------------------------*/
DItem_OptionButtonRsc OPTIONBUTTONID_EleType =
      {
      NOSYNONYM, NOHELP, MHELP, NOHOOK, NOARG,
      "Element:",
      "locElemInfoP->eleType",
      {
      {NOTYPE, NOICON, NOCMD, LCMD,  2, NOMASK, ON, "Cell"},
      {NOTYPE, NOICON, NOCMD, LCMD,  3, NOMASK, ON, "Line"},
      {NOTYPE, NOICON, NOCMD, LCMD,  4, NOMASK, ON, "Line String"},
      {NOTYPE, NOICON, NOCMD, LCMD,  6, NOMASK, ON, "Shape"},
      {NOTYPE, NOICON, NOCMD, LCMD,  7, NOMASK, ON, "Text Node"},
      {NOTYPE, NOICON, NOCMD, LCMD, 11, NOMASK, ON, "Curve"},
      {NOTYPE, NOICON, NOCMD, LCMD, 13, NOMASK, ON, "Conic"},
      {NOTYPE, NOICON, NOCMD, LCMD, 15, NOMASK, ON, "Ellipse"},
      {NOTYPE, NOICON, NOCMD, LCMD, 16, NOMASK, ON, "Arc"},
      {NOTYPE, NOICON, NOCMD, LCMD, 17, NOMASK, ON, "Text"},
      }
```

```
        };
DItem_OptionButtonRsc OPTIONBUTTONID_View =
        {
        NOSYNONYM, NOHELP, MHELP, NOHOOK, NOARG,
        "View:",
        "locElemInfoP->view",
        {
        {NOTYPE, NOICON, NOCMD, LCMD,  1, NOMASK, ON,  "1"},
        {NOTYPE, NOICON, NOCMD, LCMD,  2, NOMASK, ON,  "2"},
        {NOTYPE, NOICON, NOCMD, LCMD,  3, NOMASK, ON,  "3"},
        {NOTYPE, NOICON, NOCMD, LCMD,  4, NOMASK, ON,  "4"},
        {NOTYPE, NOICON, NOCMD, LCMD,  5, NOMASK, ON,  "5"},
        {NOTYPE, NOICON, NOCMD, LCMD,  6, NOMASK, ON,  "6"},
        {NOTYPE, NOICON, NOCMD, LCMD,  7, NOMASK, ON,  "7"},
        {NOTYPE, NOICON, NOCMD, LCMD,  8, NOMASK, ON,  "8"},
        }
        };
DItem_OptionButtonRsc OPTIONBUTTONID_LoceleStyle =
        {
        NOSYNONYM, NOHELP, MHELP, NOHOOK, NOARG, "",
        "locElemInfoP->style",
        {
        {Icon, ICONID_LineStyle0, NOCMD, LCMD, 0, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle1, NOCMD, LCMD, 1, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle2, NOCMD, LCMD, 2, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle3, NOCMD, LCMD, 3, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle4, NOCMD, LCMD, 4, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle5, NOCMD, LCMD, 5, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle6, NOCMD, LCMD, 6, NOMASK, ON, ""},
        {Icon, ICONID_LineStyle7, NOCMD, LCMD, 7, NOMASK, ON, ""},
        }
        };
/*-------------------------------------------------------------------------+
|   Toggle  Buttons                                                        |
+-------------------------------------------------------------------------*/
DItem_ToggleButtonRsc  TOGGLEID_Colors =
     {
     NOCMD, MCMD, NOSYNONYM, NOHELP, MCMD, NOHOOK, NOARG,
     NOMASK, NOINVERT,
     "Color",
     "locElemInfoP->colors"
     };
DItem_ToggleButtonRsc  TOGGLEID_Weights =
     {
     NOCMD, MCMD, NOSYNONYM, NOHELP, MCMD, NOHOOK, NOARG,
     NOMASK, NOINVERT,
     "Weight",
     "locElemInfoP->weights"
     };
DItem_ToggleButtonRsc  TOGGLEID_Styles =
     {
     NOCMD, MCMD, NOSYNONYM, NOHELP, MCMD, NOHOOK, NOARG,
     NOMASK, NOINVERT,
     "Style",
     "locElemInfoP->styles"
     };
DItem_ToggleButtonRsc  TOGGLEID_Levels =
     {
     NOCMD, MCMD, NOSYNONYM, NOHELP, MCMD, NOHOOK, NOARG,
     NOMASK, NOINVERT,
     "Levels",
     "locElemInfoP->levels"
```

```
     };
DItem_ToggleButtonRsc  TOGGLEID_CellElem =
      {
      NOCMD, MCMD, NOSYNONYM, NOSYNONYM, MCMD, NOHOOK, NOARG,
      NOMASK, NOINVERT,
      "Search in cell",
      "locElemInfoP->cellElem"
      };
DItem_ToggleButtonRsc  TOGGLEID_AttribData =
      {
      NOCMD, MCMD, NOSYNONYM, NOHELP, MCMD, NOHOOK, NOARG,
      NOMASK, NOINVERT,
      "Attribute Data",
      "locElemInfoP->attribData"
      };
/*-------------------------------------------------------------------------+
|    Push Button Items                                                     |
+-------------------------------------------------------------------------*/
DItem_PushButtonRsc  PUSHBUTTONID_Start =
     {
     NOT_DEFAULT_BUTTON, NOHELP, MHELP,
     HOOKITEMID_Button_Locele, 1, NOCMD, MCMD, "",
     "Start"
     };
DItem_PushButtonRsc PUSHBUTTONID_Next =
     {
     NOT_DEFAULT_BUTTON, NOHELP, MHELP,
     HOOKITEMID_Button_Locele, 2, NOCMD, MCMD, "",
     "Next"
     };
DItem_PushButtonRsc  PUSHBUTTONID_Quit  =
     {
     NOT_DEFAULT_BUTTON, NOHELP, MHELP,
     HOOKITEMID_Button_Locele, 3, NOCMD, MCMD, "",
     "Quit"
     };
DItem_PushButtonRsc PUSHBUTTONID_OLevel =
     {
     NOT_DEFAULT_BUTTON, NOHELP, MHELP,
     HOOKITEMID_Dummy, 0, CMD_OPENMODAL, LCMD, "",
     "Levels"
     };
/*-------------------------------------------------------------------------+
|      Level Resources                                                     |
+-------------------------------------------------------------------------*/
DItem_LevelMapRsc LEVELMAPID_Levels =
      {
      NOHELP, MHELP,
      "Levels" ,
      "locElemInfoP->level",
      ""
      };
/*-------------------------------------------------------------------------+
|      Color Picker Item  Resources                                        |
+-------------------------------------------------------------------------*/
DItem_ColorPickerRsc COLORPICKERID_Color =
     {
     NOCMD, MCMD, SYNONYMID_ColorLocele, NOHELP, MHELP, NOHOOK, NOARG,
     TEXTID_LoceleColor, NOMASK, "", "locElemInfoP->color"
     };
/*-------------------------------------------------------------------------+
|      Scroll Bar Resources                                                |
```

```
+-------------------------------------------------------------------------*/
DItem_ScrollBarRsc SCROLLBARID_Weight =
     {
     HOOKITEMID_ScrollBar, NOARG,  0,  31,  1,  5,  0.1,
     "locElemInfoP->weight"
     };
/*-------------------------------------------------------------------------+
|      Text Item Resources                                                 |
+-------------------------------------------------------------------------*/
DItem_TextRsc TEXTID_LoceleColor =
      {
      NOCMD, LCMD,  SYNONYMID_ColorLocele, NOHELP, MHELP,
      NOHOOK,  NOARG,
      3,  "%-ld",  "%ld",  "0",  "253",  NOMASK,  NOCONCAT,
      "",
      "locElemInfoP->color"
      };
DItem_TextRsc TEXTID_LoceleWeight =
      {
      NOCMD, LCMD, SYNONYMID_WeightLocele, NOHELP, MHELP,
      NOHOOK, NOARG,
      3,  "%-ld",  "%ld",  "0",  "31", NOMASK, NOCONCAT,
      "",
      "locElemInfoP->weight"
      };
/*-------------------------------------------------------------------------+
|    Item  Syononyms                                                       |
+-------------------------------------------------------------------------*/
DItem_SynonymsRsc SYNONYMID_ColorLocele =
     {
     {
     {Text,         TEXTID_LoceleColor},
     {ColorPicker, COLORPICKERID_Color},
     }
     };
DItem_SynonymsRsc SYNONYMID_WeightLocele =
     {
     {
     {Text,        TEXTID_LoceleWeight},
     {ScrollBar,  SCROLLBARID_Weight},
     }
     };
```

locelmsg.r 含有 LOCELE 应用的信息。

```
/*-------------------------------------------------------------------------+
|   locelmsg.r                                                             |
+-------------------------------------------------------------------------*/
#include <rscdefs.h>
#include <cmdclass.h>
MessageList 0 =
{
   {
   { 0, "Locate Element" },
   }
};
```

locelcmd.r 含有 LOCELE 应用的命令表。

```
/*-------------------------------------------------------------------------+
|  locelcmd.r  -command table for locele application                       |
+-------------------------------------------------------------------------*/
#include <rscdefs.h>
#include <cmdclass.h>
```

```
#define  CT_NONE                0
#define  CT_LOCATE              1
#define  CT_ELEMENT             2

Table  CT_LOCATE =
     {
             { 1, CT_ELEMENT, SHOW,       REQ,      "LOCATE" },
             { 2, CT_NONE,    INPUT,      NONE,     "OPENMODAL" }
     };
Table  CT_ELEMENT =
     {
          { 1, CT_NONE, INHERIT, NONE, "ELEMENT" }
     };
```

loceltyp.mt 发布对话框管理程序的结构:

```
/*----------------------------------------------------------------------+
|   loceltyp.mt -- Locate Element Dialog Types                          |
+----------------------------------------------------------------------*/
#include    "loceldlg.h"

publishStructures (loceleminfo);
```

键入 bmake -a locele 建立 LOCELE 应用。

```
#--------------------------------------------------------------
#       LOCELE MDL Make File
#--------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------
#       Define constants specific to this locele application
#--------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

loceleObjs = $(o)locele.mo     $(mdlLibs)ditemlib.dlo

loceleRscs = $(o)locelcmd.rsc  $(o)loceldlg.rsc  \
             $(o)loceltyp.rsc  $(o)locelmsg.rsc  \
             $(o)locele.mp

#----------------------------------------------------------------------
# Create needed output directories if they don't exist
#----------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#------------------------------------------
#       Generate Command Tables
#------------------------------------------
$(privateInc)locelcmd.h  :  $(baseDir)locelcmd.r

$(o)locelcmd.rsc         :  $(baseDir)locelcmd.r

#------------------------------------------
#       Compile Dialog & Message Resources
#------------------------------------------
$(o)loceldlg.rsc         :  $(baseDir)loceldlg.r   $(privateInc)locelcmd.h

$(o)locelmsg.rsc         :  $(baseDir)locelmsg.r

#------------------------------------------
#       Make resource to publish structure(s)
#------------------------------------------
$(o)loceltyp.r           :  $(baseDir)loceltyp.mt   $(privateInc)loceldlg.h
```

```
$(o)loceltyp.rsc         :  $(o)loceltyp.r

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)locele.mo              :   $(baseDir)locele.mc

$(o)locele.mp              :   $(loceleObjs)
             $(msg)
             >$(o)temp.cmd
             -a$@
             $(linkOpts)
             $(loceleObjs)
             <
             $(MLinkCmd) @$(o)temp.cmd
             ~time

#-----------------------------------------
#       Merge Objects into one file
#-----------------------------------------
$(mdlapps)locele.ma       : $(loceleRscs)
             $(msg)
             >$(o)temp.cmd
             -o$@
             $(loceleRscs)
             <
             $(RLibCmd) @$(o)temp.cmd
             ~time
```

7.5 视图钩函数

MDL 允许我们定义一个当视图更新或光标移入视图时执行的函数。这一有力的特性展开了许多可能性。

7.5.1 视图用户函数

MDL 提供了当某些视图事件出现时调用用户函数的功能。请注意那些以斜体字示出的函数名，它们不是函数的真实名字。请在你的程序中用自己的函数名替换掉它们。

userView_update　　当视图刷新时要调用的函数

userView_motion　　当光标在视图内移动时要调用的函数

userView_noMotion　　当光标在一个视图中停止移动时要调用的函数

我们写一个使设计文件中所有激活点提高亮度的应用。这个程序将在视图更新后执行，并使用 mdlElement_display 在屏幕上画一个圆。

在屏幕刷新时，HIGHAP 应用将调用 mdlView_setFunction 规定要调用的函数。在屏幕刷新前，可用 UPDATE_PRE 调用该函数，也可以在刷新后用 UPDATE_POST 调用。适用的选项为：

UPDATE_PRE	在屏幕更新后调用用户函数
UPDATE_POST	在屏幕更新后调用用户函数
VIEW_NOMOTION	当光标移动时调用用户函数
VIEW_MOTION	当光标停止移动时调用用户函数
UPDATE_EACH_ELEMENT	当一个元素被更新时调用用户函数。这对隐藏元素或改变它们的线符很有用
PLOTUPDATE_PRE	当每次绘图更新前调用(即在生成一个绘图文件前)
PLOTUPDATE_POST	当每次绘图更新后调用(即在生成一个绘图文件后)

因为我们要在屏幕更新后在其上画一个圆，我们将使用 UPDATE_POST。如果我们要用 UPDATE_PRE，则我们必须

从我们的函数返回一个零值，以便告诉 MicroStation 继续更新。否则，它将停止。

main 函数调用 mdlView_setFunction 来建立一个用户函数 handleUptate。

```
main()
{
mdlView_setFunction (UPTATE_POST, handleUpdate);
}
```

当我们用任何一个视图命令更新屏幕时，如 ZOOM，WINDOW，UPDATE 或 FIT，函数 handleUpdate 被激活。它扫描视图 1 中的激活点，并围绕每个点放置一个高亮度圆。为了限制搜寻区域，我们把扫描范围传递给扫描程序。在这个例子中，它是视图 1 的范围。我们调用 mdlScan_viewRange 设置 scanList(扫描表)范围成员，以便从主文件(MASTERFILE)中找到扫描视图(SCANVIEW)中合适的元素。

```
mdlScan_viewRange (&scanList，SCANVIEW，MASTERFILE);
```

当 handleUpdate 找到符合搜索标准的一条线时，调用 highlight。

在 highlight 中，我们首先检查是否有一个线元素。

```
if (mdlElement_getType(el) != LINE_ELM)
   return (MODlFY_STATUS_NOCHANGE);
```

然后，我们提取这些点并用 mdlVec_pointEqual 测试它们是否相同。一个激活点定义为一条长度为零具有相同的起点和终点的线。

```
mdILinear_extract (pnts，&numVerts，el，MASTERFILE);
if (!mdlVec_pointEqual(&pnts[0], &pnts[1])
    return(MODIFY_STATUS_NOCHANGE);
```

最后，我们用 mdlEllipse_create 生成一个圆，并把它画在屏幕上。我们用当前字符高做为圆的半径。

```
mdlEllipse_create (&circle, NULL, &pnts[0], txtHeight，txtHeight, NULL, NULL);
mdlElement_display (&circle, HILITE);
return (MODIFY_STATUS_NOCHANGE);
```

注意，我们总是用 MODIFY_STATUS_NOCHANGE 退出。用这种方法，甚至在我们用 mdlModify_elementSingle 操作元素时也不会改变元素。

下面是程序 HIGHAP 的完整清单。当你装入这个应用时，程序立即建立视图状态函数。因此，我们不使用命令表或资源文件。键入 mdl l highap 装入这个应用。

```
    /*-------------------------------------------------------------------------------+
|    Copyright (C) 1991 Mach Dinh-Vu,All aights Reserved                          |
|                                                                                 |
|    highap.mc  -    Highlight the active points in the file。                     |
| UpgradeToV8 : MicroStationFan    2006/06                                        |
+-------------------------------------------------------------------------------*/
#include <mdl.h>
#include <userfnc.h>
#include <mselemen.fdf>
#include <msmisc.fdf>
#include <msoutput.fdf>
#include <msparse.fdf>
#include <msscan.fdf>
#include <msvec.fdf>
#include <msview.fdf>
#define  SCANVIEW  0  // view 1

Private double  txtHeight = 0;

/*-------------------------------------------------------------------------------+
```

```
|         name    highlight                                                      |
+-------------------------------------------------------------------------------*/
Private int highlight ( MSElementUnion  *el )
{
   Dpoint3d    pnts[2];
   int         numVerts;
   MSElement   circle;

   if (mdlElement_getType(el) != LINE_ELM)
      return (MODIFY_STATUS_NOCHANGE);
   mdlLinear_extract (pnts, &numVerts, el, MASTERFILE);
   if (!mdlVec_pointEqual (&pnts[0], &pnts[1]))
      mdlEllipse_create (&circle, NULL, &pnts[0], txtHeight, txtHeight, NULL, NULL);
      mdlElement_display (&circle, HILITE);
      return (MODIFY_STATUS_NOCHANGE);
}
/*-------------------------------------------------------------------------------+
|   name             handleUpdate                                               |
+-------------------------------------------------------------------------------*/
Private int handleUpdate
(
int                    preUpdalte,
int                    eraseMode,
DgnModelRefListP       modelRefList,
int                    numberRegions,
Asynch_update_view     regions [],
BSIRect               *coverLists [],
int                    numCovers [],
MSDisplayDescr         *displayDescr []
)
{
   ULong           elemAddr[50], eofPos, filePos;
   int             scanWords, status, i, numAddr;
   ExtScanlist     scanList;

   mdlScan_initScanlist (&scanList);
   mdlOutput_message ( "Scanning file  ....  ");
   scanList.scantype       = ELEMTYPE | ONEELEM;
   scanList.extendedType   = FILEPOS;
   scanList.typmask[0]     = TMSK0_LINE;
   mdlScan_viewRange (&scanList, SCANVIEW, MASTERFILE);
   eofPos  = mdlElement_getFilePos (FILEPOS_EOF, NULL);
   filePos = 0L;            /* start seacrh from top of file */
   mdlScan_initialize (MASTERFILE, &scanList);
   /* loop through all line elements in file */
   do{
      scanWords = sizeof(elemAddr) / sizeof(short);
      status    = mdlScan_file (elemAddr, &scanWords,
                                sizeof (elemAddr), &filePos);
      numAddr   = scanWords / sizeof (short);
      for (i=0; i < numAddr; i++)
      {
         if (elemAddr[i] >= eofPos)  break;
         mdlModify_elementSingle ( 0, elemAddr[i],
                                   MODIFY_REQUEST_NOHEADERS,  MODIFY_ORIG,
                                   highlight, NULL, 0L);
      }
   } while (status == BUFF_FULL);
   mdlOutput_message ( "" );
   return 0;
}
/*-------------------------------------------------------------------------------+
```

```
| name                    main                                                |
+-------------------------------------------------------------------------*/
Private void main (void)
{
   mdlParams_getActive ((double *)&txtHeight, ACTIVEPARAM_TEXTHEIGHT);
   mdlView_setFunction (UPDATE_POST, handleUpdate);
}
```

为了建立 HIGHAP 应用,键入 bmake -a highap。

```
#--------------------------------------------------------------
#       HIGHAP MDL Make File
#--------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------
#       Define constants specific to this highap application
#--------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

highapObjs = $(o)highap.mo     $(mdlLibs)ditemlib.dlo

#--------------------------------------------------------------------
# Create needed output directories if they don't exist
#--------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)highap.mo              :   $(baseDir)highap.mc
$(mdlapps)highap.ma        :   $(highapObjs)
            $(msg)
            >$(o)temp.cmd
            -a$@
            $(linkOpts)
            $(highapObjs)
            <
            $(MLinkCmd) @$(o)temp.cmd
            ~time
```

7.6 第七章小结

对话框是必须发挥作用、功能非常强大的用户界面。在第三章和本章，我们看到了使用对话框的许多不同的方法。还有一些你们可能希望继续探索的功能本书尚未涉及到。这些功能是：

●建立菜单条

●图符调色板

●用 rasticon.ma 生成图符

DLOGDEMO 程序(作为 MicroStation 演示程序提供的)是使用对话框的最好的例子。这个程序是一个很好的源码程序，其中对话条目定义可以”切”下来，接到其它应用中去。此外，DLOGDEMO 应用还演示了第三章已经讨论的 DMSG 的使用。

【注】: 本章所有源代码都在 MicroStation V8 2004 下调试通过。

第八章　输入队列

MicroStation 的所有输入都必须首先进入输入队列，在输入达到状态调度程序以前，允许我们对输入进行预处理。MicroStation 用户的最普遍的要求是要一个 Miorostation 的“只能看”的拷贝。这里的意思是把操作符只限于象 WINDOW AREA 或 ZOOM 这样的视图命令。当绘图达到稳定状态时，用它只做预先检查并绘出图纸，因此不再做任何修改是十分有利的。

8.1 工作方式

图 8.1 示出了控制输入队列的调度流程的一般操作。从图中我们可以看到，当 MicroStation 中出现某些事件时，将调用用户提供的函数，我们将为 userInput_commadFilter 提供我们自己的用户函数。

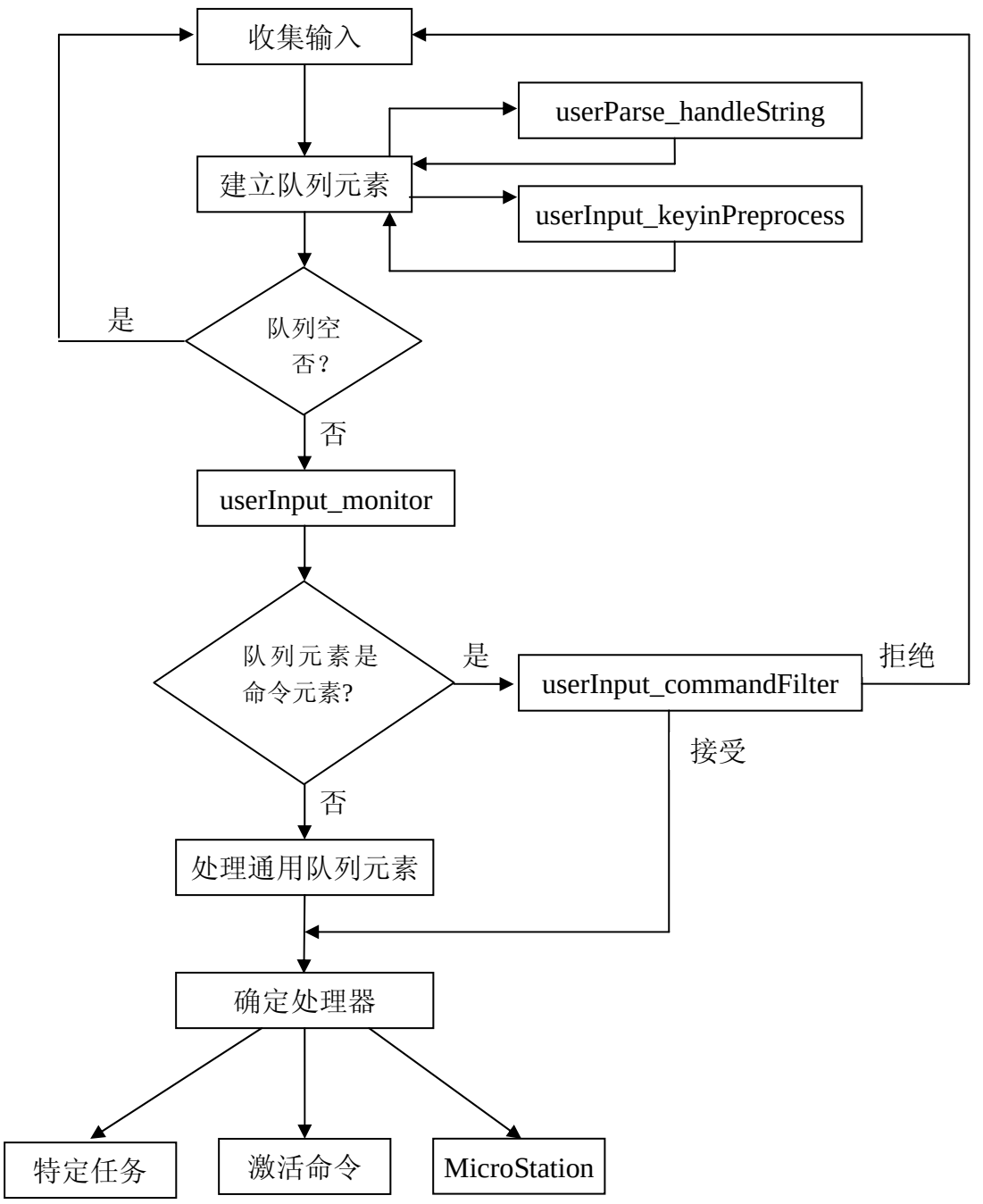


图 8.1　MicroStation 主调度分配流程

8.2 输入队列函数

MDL 提供了一些捕捉和操作队列元素的函数：

mdlInput_commandState　　返回视图命令或基本命令

mdlInput_disableCommandClass　　废除命令类别

mdlInput_enableCommandClass	启动命令类别
mdlInput_endCommand	停止“激活命令”状态
mdlInput_getMessage	把最后进入队列的元素拷贝到指定区域
mdlInput_getTabletType	决定我们是否使用数字化板或鼠标
mdlInput_pause	停止 MioroStation 并等待击键盘
mdlInput_requeueLastInput	把上一次排列的队列元素重新排队供 MioroStation 处理
mdlInput_sendCommand	生成并排列一个 CMDNUM 队列元素
mdlInput_sendKeyin	生成并排列一个 KEYIN 队列元素
mdlInput_sendMessage	生成并排列一个队列元素
mdlInput_sendReset	生成并排列一个 RESET 队列元素
mdlInput_sendResume	在队列的起始处安置一个恢复信息
mdlInput_sendUORPoint	生成并排列一个数据点元素
mdlInput_setMonitorFunction	指定一个用户函数在队列元素进入队列前处理它们
mdlInput_startCommand	为当前任务请求激活命令状态
mdlInput_waitForMessage	使 MDL 任务挂起，直到它接收到另一个信息

用户函数

在 MicroStation 队列处理某些事件出现时，我们可以指定被调用的程序。这里用斜体字示出的函数名，不是这些函数实际的名字，用你程序中使用的名字替换它们。

userInput_commandFilter	当 INPUT_COMMAND_FILTER 被指定为 mdlInput_setFunction 中的一个参数时调用的函数
usertInput_monitor	在调用 mdlInput_setMonitorFunction 中规定用户函数时调用的函数
usertInput_preprocessKeyin	当 INPUT_KEYIN_PREPROCESS 被指定为一个 mdlInput_setFunction 中的参数时调用的函数
usertInput_receive	当 INPUT_MESSAGE_RECEIVE 被指定为 mdlInput_setFunction 中的一个参数时调用的函数

8.3 队列元素

队列元素 Inputq_element 的结构在<msinputq.h>中定义。每个队列元素有一个头 Inputq_header，其后为输入队列类型的联合。

```
typedef struct inputq_element
{
Inputq_header hdr;
Union
{
Inputq_keyin      keyin;     /* keyed in (not command) */
Inputq_datapnt    data;      /* data point */
Inputq_command    cmd;       /* parsed command */
Inputq_tentpnt    tent;      /* tentative point */
Inputq_reset      reset;     /* reset */
Inputq_partial    partial;       /* incomplete keyin */
```

```
   Inputq_unassignedcb cursbutn;       /* unassigned cursor button */
   Inputq_menumsg    menumsg;      /* menu message to be posted */
   Inputq_submenu    submenu;      /* submenu to be displayed */
   Inputq_menuwait       menuwait;     /* go back, get another input */
   Inputq_contents       contents;     /* menu entry contents */
   Inputq_null       nullqelem;        /* NULL event */
   Inputq_nullcmd    nullcmd;         /* NULL command elem */
   Inputq_rawButton    rawButton;      /* raw button information */
   Inputq_rawKeyStroke rawKeyStroke;   /* raw keystroke information */
   Inputq_cookedKey    cookedKey;      /* virtual keystoke information */
   Inputq_rawIconEvent rawIconEvent;   /* raw Icon event information */
   Inputq_timerEvent   timerEvent;     /* timer event information */
   Inputq_virtualEOQ   virtualEOQ;     /* virtual end-of-queue */
   Inputq_userWinClose userWinClose;   /* user closing window */
   char fill[480];                     /* make sure it's big enough */
} u;
     } Inputq_element;
typedef struct inputq_header
   {
   int     cmdtype;        /* type of input following */
   int     bytes;          /* bytes in queue element */
   int     source;         /* source of input (user, app, etc) */
   int     uc_fno_value;   /* value to put in tcb->uc_fno */
   int    sourcepid;       /* source pid for queue element */
   char    taskId[16];     /* destination child task */
   int     unused;         /* make size of Inputq_header a multiple of 8 bytes! */
   } Inputq_header;
```

在我们的应用中,我们只关心命令的类别和结构 Inputq__command。

```
typedef struct inputq_command
   {
   int    commandClass;      /* command class */
   int    immediate;         /* has immediate mode */
   long   command;           /* unique command id */
   char   taskId[16];        /* destination child task */
   char   unparsed[1];       /* Unparsed portion (if any) */
   } Inputq_command;
```

8.4 命令过滤器

为了使设计文件“只能看”,在它进入队列前要对命令进行筛选。我们要把 INPUT_COMMAND_FILTER 传递给函数 mdlInput_setFuntion，这将建立起我们的用户函数 commandFilter。

```
   mdlInput_setFunction (INPUTCOMMAND_FILTER, commandFilter);
```

在用户函数 commandFilter 中我们要检查队列元素的类别。如果它属于下列中任何一类，它将传递到 MicroStation 中去。否则则被拒绝接收，并显示警告信息。视图类别是 VIEWING，VIEWPARAM 和 VIEWIMMED。我们允许用户使用 MEASURE 来进行度量。最后一个是要求用 EXIT(输入类别)退出设计文件的功能。

在包含文件<cmdclass.h>中可以找到中断命令的类别。

```
switch (queueElementP->u.cmd.commandClass)
{
case VIEWING:
case MEASURE:
case VIEWPARAM:
case VIEWIMMED:
case INPUT:
          return INPUT_COMMAND_ACCEPT;
           break;
default:   mdlUtil_beep(1);
           mdlOutput_error ("WARNING:This is a VIEW only file.");
           return INPUT_COMMAND_REJECT;
```

8.5 MDL 应用的自动执行

MicroStation 提供了两个环境变量 MS_DGNAPPS 和 MS_INITAPPS，指定在 MicroStation 启动时要装入的程序。MS_INITAPPS 在 MicroStation 进入图形状态前定义要执行的 MDL 应用，MicroStation 管理程序(mm.ma)是一个很好的例子。

当一个设计文件打开后，MS_DGNAPPS 指定要装入的 MDL 应用。我们将把 Viewonly 应用赋值给这个环境变量。

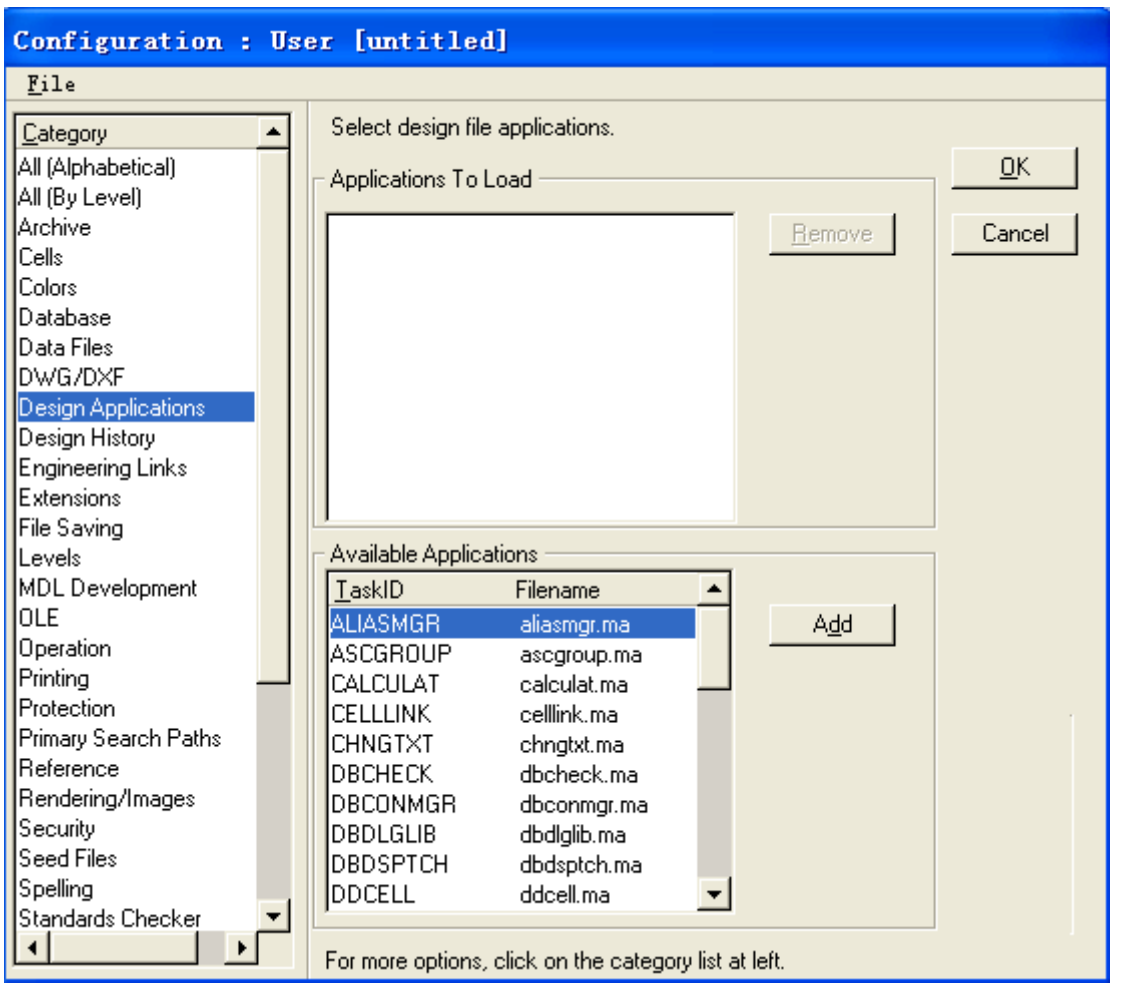


图 8.2 设置 MS_DGNAPPS 环境变量对话框

为了赋值，我们在 MicroStation 环境中选菜单 *Workspace(工作空间)→Configuration(配置)*打开如图 8.2 所

示的 Configuration 对话框，在左侧 *Category(类别)*中选择 *Design Applications(设计应用)*，在右侧的 *Available Applications(可用应用)*中选中 Viewonly 并点击 *Add* 按钮即可。

注意，我们可以启动多个应用。在图 8.2 的对话框中可选择多个应用即表示当一个设计文件打开后这些应用都将被执行。

这个程序与我们见过的其它程序不同。它没有命令表，因此也没有命令资源文件。当应用被装入后，从 main 开始执行，然后结束。要装入这个程序，应键入 mdl l viewonly 或使用前面讨论过的启动环境变量。

```
/*----------------------------------------------------------------------------+
|  Copyright(c)1991, Mach N.Dinh-Vu, All Rights Reserved                      |
|  Program    : viewonly.mc                                                   |
|  Revision   : 1.0.a                                                         |
|  UpdateToV8 : MicroStationFan                                               |
+----------------------------------------------------------------------------+
|  Set the file into VIEW ONLY mode                                           |
+----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------+
|  Include  Files                                                             |
+----------------------------------------------------------------------------*/
#include   <userfnc.h>
#include   <cmdlist.h>
#include   <cmdclass.h>
#include   <./stdlib/string.h>
#include   <msinput.fdf>
#include   <msmisc.fdf>
#include   <msoutput.fdf>

/*----------------------------------------------------------------------------+
|   name      commandFilter                                                   |
+----------------------------------------------------------------------------*/
Private int commandFilter
(
Inputq_element  *queueElementP
)
{
   switch (queueElementP->u.cmd.commandClass)
   {
      case VIEWING:
      case MEASURE:
      case VIEWPARAM:
      case VIEWIMMED:
      case INPUT:
                  return INPUT_COMMAND_ACCEPT;
                  break;
       default:
                  mdlUtil_beep(1);
                  mdlOutput_error ("WARNING: This is a VIEW only file.");
                  return INPUT_COMMAND_REJECT;
   }
}
/*----------------------------------------------------------------------------+
| name          main                                                          |
+----------------------------------------------------------------------------*/
Private void main (void)
{
   mdlInput_setFunction (INPUT_COMMAND_FILTER, commandFilter);
   mdlOutput_prompt("file set to VIEW ONLY");
}
```

VIEWONLY 制作文件是一个减缩版本，所以没有要编译的命令资源表。

```
#-----------------------------------------------------------------
#       VIEWONLY MDL Make File
```

```
#-------------------------------------------------------------------
%include  mdl.mki

#-------------------------------------------------------------------
#       Define constants specific to this viewonly application
#-------------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

viewonlyObjs = $(o)viewonly.mo

#-------------------------------------------------------------------------
# Create needed output directories if they don't exist
#-------------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)viewonly.mo              :   $(baseDir)viewonly.mc

$(mdlapps)viewonly.ma        :   $(viewonlyObjs)
             $(msg)
             >$(o)temp.cmd
             -a$@
             $(linkOpts)
             $(viewonlyObjs)
             <
             $(MLinkCmd) @$(o)temp.cmd
             ~time
```

8.6 取消命令类别

在 VIEWONLY 例子中，我们检查输出队列中的每一命令。MDL 用两个函数 mdlInput_disableCommandClass 和 mdlInput_enableCommandClass 提供一种交替使用的方法做这个工作。在下一个例子 VWCLASS 中，我们要使用这两个函数。首先，我们取消所有命令类别，然后重新启动我们所需要的那些命令类别。这个方法不像 VIEWONLY 那么出色，因为它不是动态的。它允许用户选择一个命令，只是为了发现它不做任何事。作为使用函数 mdlInput_disableCommandClass 和 mdlInput_enableCommandClass 的例子，VWCLASS 在此示出。mdlInput_disableCommandClass 的输入是两个 32 位掩码，第一个掩码从第 1 类到第 32 类，第二个从第 33 类到第 64 类。

首先，我们取消所有的类别。我们把变量的每一位设置为 16 进制的 oxffffffff。然后象在例子 VIEWONLY 中那样，通过清除与允许命令类别相对应的那些位，我们启动特定的类别。

```
/* disable all class */
mask=0xffffffff;
mdlInnput_disableCommandClass (mask, mask);
/* enable specific classes */
mask = 1<<(VIEWING-1);
mask |=1<<(PLOT-1);
mask |=1<<(MEASURE-1);
mask |=1<<(INPUT-1);
mask |=1<<(VIEWPARAM-1);
mask |=1<<(VIEWIMMED-1);
```

```
    mask |=1<<(WIMDOWMAN-1);
    mask |=1<<(DIALOGMAN-1);
    mdlInput_enableCommandclass (mask, 0L);
```

为了安装这个应用,键入 mdl l vwclass,它将从 main 开始执行,然后结束。

```
/*----------------------------------------------------------------------+
|  Copyright(c)1991, Mach N.Dinh-Vu, All Rights Reserved                |
|  Revision   : 1.0.a                                                   |
|  UpdateToV8 : MicroStationFan                                         |
+----------------------------------------------------------------------+
|  Set the file into VIEW ONLY mode by disabling command class          |
+----------------------------------------------------------------------*/
/*----------------------------------------------------------------------+
|  Include  Files                                                       |
+----------------------------------------------------------------------*/
#include   <cmdclass.h>
#include   <msinput.fdf>
#include   <msoutput.fdf>

/*----------------------------------------------------------------------+
| name          main                                                    |
+----------------------------------------------------------------------*/
Private void main (void)
{
   long    mask;
   /* disable all class */
   mask = 0xffffffff;
   mdlInput_disableCommandClass (mask, mask);
   /* enable specific classes */
   mask = 1 << (VIEWING - 1);
   mask |= 1 << (PLOT - 1);
   mask |= 1 << (MEASURE - 1);
   mask |= 1 << (INPUT - 1);
   mask |= 1 << (VIEWPARAM - 1);
   mask |= 1 << (VIEWIMMED - 1);
   mask |= 1 << (WINDOWMAN - 1);
   mask |= 1 << (DIALOGMAN  - 1);
   mdlInput_enableCommandClass (mask, 0L );
   mdlOutput_prompt ("file set to VIEW ONLY");
}
```

VWCLASS 的制作文件与 VIEWONLY 相似，没有要编译的资源文件。

```
#--------------------------------------------------------------------
#       VWCLASS MDL Make File
#--------------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------------
#       Define constants specific to this vwclass application
#--------------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

vwclassObjs = $(o)vwclass.mo

#--------------------------------------------------------------------
# Create needed output directories if they don't exist
#--------------------------------------------------------------------
$(o)$(tstdir)        : $(o)$(tstdir)
```

```
#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)vwclass.mo              :   $(baseDir)vwclass.mc

$(mdlapps)vwclass.ma        :   $(vwclassObjs)
            $(msg)
            >$(o)temp.cmd
            -a$@
            $(linkOpts)
            $(vwclassObjs)
            <
            $(MLinkCmd) @$(o)temp.cmd
            ~time
```

8.7 系统函数

MDL 提供了一些可用于 VIEWONLY 应用的系统函数。值得注意的函数是 mdlSystem_createStartupElement，它把一个启动元素装入设计文件。当 MicroStation 进入一个设计文件时，它将扫描类型 66、第 10 层启动元素。如果它找到了启动元素，它将排列这个命令，以便启动 MDL 程序。当 MicroStation 装完设计文件时，它开始处理所有的队列命令，包括来自起动元素的命令。

mdlSystem_abortRequested	决定用户是否要中断一个 MDL 任务
mdlSystem_cancelTimer	清除计时器请求
mdlSystem_closeDesignFile	如果 MS_INITAPPS 被定义则执行它，否则终止 MicroStation 并关闭设计文件
mdlSystem_createStartupElement	生成一个启动元素，该元素启动一个 MDL 应用
mdlSystem_deleteStartupElement	删除一个启动元素
mdlSystem_enterGraphics	启动 MicroStation 图形环境
mdlSystem_exit	中断并退出 MDL 任务
mdlSystem_fileDesign	存贮当前设置
mdlSystem_getChar	从键盘上得到一个字符
mdlSystem_getenv	返回环境变量的值
mdlSystem_getMdlTaskList	返回安装的任务名字
mdlSystem_getTaskStatistics	返回对指定任务的统计
mdlSystem_getTicks	返回经过的时间
mdlSystem_loadMdlProgram	安装指定的 MDL 程序
mdlSystem_newDesignFile	关闭当前文件并打开一个新文件
mdlSystem_pauseTicks	暂停 MicroStation 一段时间
mdlSystem_putenv	设定环境变量的值
mdlSystem_setFunction	指定一个 usersystem 用户函数(见下面)
mdlSystem_setTimerFunction	计时器到时建立调用的用户函数
mdlSystem_unloadMdlProgram	卸去指定的 MDL 程序
mdlSystem_userAbortEnable	控制用户是否可以中断 MDL 任务

一. 用户函数

当某些系统事件出现时，我们可以指定调用的函数。这里用斜体字表示的函数名不是函数真实的名字，在你的程序中用自己的函数名替换它们。

userSystem_mdlChildTerminated	当一个子任务结束时调用的函数
userSystem_newDesignFile	当 MicroStation 装入一个新的设计文件时调用的函数
userSystem_reloadProgram	当 MicroStation 接到一个来自已装载程序的装载请求时要调用的函数
userSystem_timerExpired	计时器到时要调用的函数
userSystem_unloadProgram	当一个程序卸载时要调动的函数
userSystem_writeToFile	当 MicroStation 对一个设计文件写入时要调用的函数

二. 自动启动

程序 instvo.mc 将在当前文件中生成一个启动元素。这个程序是很直接的，它首先用应用名调用 mdlSystem_createSatrtupElement，然后用 mdlElement_add 添加启动元素。

当您在某个 DGN 文件中执行完该应用后即在这个 DGN 中写入了一个启动元素。以后只要您打开这个 DGN 就会自动加载 VIEWONLY 应用从而使得该 DGN 为“只读”。

```
/*----------------------------------------------------------------------+
|  copyright(c)1991, Mach N. Dinh-Vu, All Rights Reserved               |
|  Program    :  instvo.mc                                              |
|  Revision   :  1.0.a                                                  |
|  UpgradeToV8:MicroStationFan                                          |
+----------------------------------------------------------------------+
|       install VIEWONLY startup command line                           |
+----------------------------------------------------------------------*/
/*----------------------------------------------------------------------+
|       Include Files                                                   |
+----------------------------------------------------------------------*/
#include    <mselemen.fdf>
#include    <msoutput.fdf>
#include    <mssystem.fdf>
/*----------------------------------------------------------------------+
|       name    main                                                    |
+----------------------------------------------------------------------*/
Private void main (void)
{
   MSElementUnion  startel;

   if  (mdlSystem_createStartupElement (&startel, "VIEWONLY") == SUCCESS){
      mdlElement_add (&startel);
      mdlOutput_prompt ("file set to VIEW ONLY");
   } else {
      mdlOutput_prompt("unable to install start up elment");
   }
}
```

INSTVO 应用不要求命令表资源，该应用装载时直接从 main 执行。键入 bmake -a instvo 编译这个应用。

```
#--------------------------------------------------------------
#       INSTVO MDL Make File
#--------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------
#       Define constants specific to this instvo application
#--------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

instvoObjs = $(o)instvo.mo
```

```
#--------------------------------------------------------------------------
# Create needed output directories if they don't exist
#--------------------------------------------------------------------------
$(o)$(tstdir)         : $(o)$(tstdir)

#-----------------------------------------
#       Compile and link MDL Application
#-----------------------------------------
$(o)instvo.mo               :   $(baseDir)instvo.mc

$(mdlapps)instvo.ma         :   $(instvoObjs)
             $(msg)
             >$(o)temp.cmd
             -a$@
             $(linkOpts)
             $(instvoObjs)
             <
             $(MLinkCmd) @$(o)temp.cmd
             ~time
```

8.8 演示程序

系统函数提供了编写演示程序的能力，这种程序能提供具有全部功能的应用，但是只能在一定时间内运行。这与 MicroStation 没有硬锁时运行的情况相同，10 分钟的演示期满便能退出。

我们的应用 DEMOMODE 将使 VIEWONLY 只运行 5 分钟，然后退出 MicroStation。时间到了以后，系统函数 mdlSystem__setTimerFuncion 将调用我们的函数 demoExpired。时间基于约为六十分之一秒的信号。用户函数暂停 5 秒钟，显示信息“Demo period expired”,然后调用 mdlSystem_closeDesignFile 退出 MicroStation。

demomode.mc 的源程序如下所示。安装命令将执行 main 函数,因为应用没有命令表。键入 mdl l demomode 装入这个应用。

```
/*+--------------------------------------------------------------------------+
|  Copyright(c)1991, Mach N.Dinh-Vu, All Rights Reserved                     |
|  Program    : viewonly.mc                                                  |
|  Revision   : 1.0.a                                                        |
|  UpdateToV8 : MicroStationFan                                              |
+--------------------------------------------------------------------------+
|  Set the file into DEMO mode using the VIEWONLY facilities                 |
+--------------------------------------------------------------------------*/
/*+--------------------------------------------------------------------------+
|  Include  Files                                                            |
+--------------------------------------------------------------------------*/
#include   <userfnc.h>
#include   <cmdclass.h>
#include   <msinput.fdf>
#include   <msmisc.fdf>
#include   <mssystem.fdf>
#include   <msoutput.fdf>

#define    TICK       1
#define    ASECOND    (60*TICK)
#define    AMINUTE    (60*ASECOND)
#define    DEMOPERIOD ( 5*AMINUTE)
Private  int timerHandle;

/*+--------------------------------------------------------------------------+
|   name      commandFilter                                                  |
+--------------------------------------------------------------------------*/
Private int commandFilter
```

```
(
Inputq_element  *queueElementP
)
{
   switch (queueElementP->u.cmd.commandClass)
   {
      case VIEWING:
      case MEASURE:
      case VIEWPARAM:
      case VIEWIMMED:
      case INPUT:
                  return INPUT_COMMAND_ACCEPT;
                  break;
       default:
                  mdlUtil_beep(1);
                  mdlOutput_error ("WARNING: This is a VIEW only file.");
                  return INPUT_COMMAND_REJECT;
   }
}
/*----------------------------------------------------------------------+
|   name          demoExpired                                          |
+----------------------------------------------------------------------*/
 Private void demoExpired (void)
{
   mdlSystem_cancelTimer (timerHandle);
   mdlSystem_pauseTicks (5*ASECOND);
   mdlSystem_closeDesignFile ();
}
/*----------------------------------------------------------------------+
| name           main                                                  |
+----------------------------------------------------------------------*/
Private void main (void)
{
   mdlSystem_setTimerFunction (&timerHandle, DEMOPERIOD, demoExpired, 0, TRUE);
   mdlInput_setFunction (INPUT_COMMAND_FILTER, commandFilter);
   mdlOutput_prompt("file set to VIEW ONLY");
}
```

键入 bmake -a demomode 建立 DEMOMODE 应用。

```
#--------------------------------------------------------------
#       DEMOMODE MDL Make File
#--------------------------------------------------------------
%include  mdl.mki

#--------------------------------------------------------------
#       Define constants specific to this demomode application
#--------------------------------------------------------------
baseDir      =   ./
privateInc   = $(baseDir)

demomodeObjs = $(o)demomode.mo

#-----------------------------------------------------------------------
# Create needed output directories if they don't exist
#-----------------------------------------------------------------------
$(o)$(tstdir)         : $(o)$(tstdir)

#-------------------------------------------
#       Compile and link MDL Application
#-------------------------------------------
$(o)demomode.mo               :    $(baseDir)demomode.mc

$(mdlapps)demomode.ma         :    $(demomodeObjs)
```

```
$(msg)
>$(o)temp.cmd
-a$@
$(linkOpts)
$(demomodeObjs)
<
$(MLinkCmd) @$(o)temp.cmd
~time
```

第九章　走进 NativeCode 世界

从 MicroStation V8 开始，我们完全可以用 NativeCode(本机代码)方式来开发基于 MicroStation 的程序了。所谓 NativeCode 是相对于 MDL 的伪编译而言的，由于早期的 MicroStation 定位于多平台系统，因此所开发的程序需要具有跨平台特性。类似于 Java 的跨平台，在每种平台的 MicroStation 系统中都有一套把 MDL 伪代码解释为本机代码的“虚拟机”，正是各平台上虚拟机的不同才保证了 MDL 在各平台的源代码相同。其最大的好处是我们在一种平台(如 Windows)下开发的 MDL 源代码拿到另一种平台(如 Unix)下只需要重新编译链接后就能顺利运行。但随着 Bentley 公司大的战略转移，目前的 MicroStation 只能在 Windows 平台下运行了，那么我们还必须依照以前的方式来开发 MDL 程序吗？答案是否定的。我们现在完全可以编写出直接运行于操作系统级的 MicroStation 程序——NativeCode 程序。

NativeCode 程序有如下几个优点：(1). 可以方便地调用所有操作系统级的功能，如 COM、ADO、ActiveX 等；(2). 对于复杂的计算量大的程序，在性能上会有较大的提升；(3). 可以利用 Visual Studio 提供的直观而强大的调试程序功能对程序进行调试。当然，从传统的 MDL 过渡到 NativeCode 编程也是有相当的复杂性的，本章将带您走进 NativeCode 的编程世界。

在这一章中，我们将用 Microsoft 的 Visual Studio6.0 来开发运行于 MicroStation 之上的 NativeCode 程序，NativeCode 运行程序实际上是动态链接库(DLL)，目前仍然需要一个最简化的 ma 文件来调用它。我们首先用 Bentley MFC Application Wizard(奔特力 MFC 应用向导)快速生成项目文件，编译并在 MicroStation 中运行；其次，详细分析 NativeCode 的代码结构以利于我们自己按要求修改代码；再次，讲述如何在 Visual Studio 6.0 中对 NativeCode 进行调试；最后，强调一些与纯 MDL 编程不同的一些注意事项。

9.1 生成 NativeCode 程序

安照这一节的操作，您将一步步生成能运行于 MicroStation 下的 NativeCode 程序。

9.1.1 安装 Bentley MFC Application Wizard

安装过程非常简单。只需复制文件 NativeWinWiz.awx 到 C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Templates 目录下即可。如果您还没有获得 NativeWinWiz.awx 文件，请与奔特力公司联系。

9.1.2 建立一个 NativeCode 项目

启动 Microsoft Visual C++，选菜单 File 下的 New，在打开的 New 对话框中选 Project 页中的 Bentley MFC Application Wizard，再进一步指定项目位置和项目名称，如图 9.1 所示。

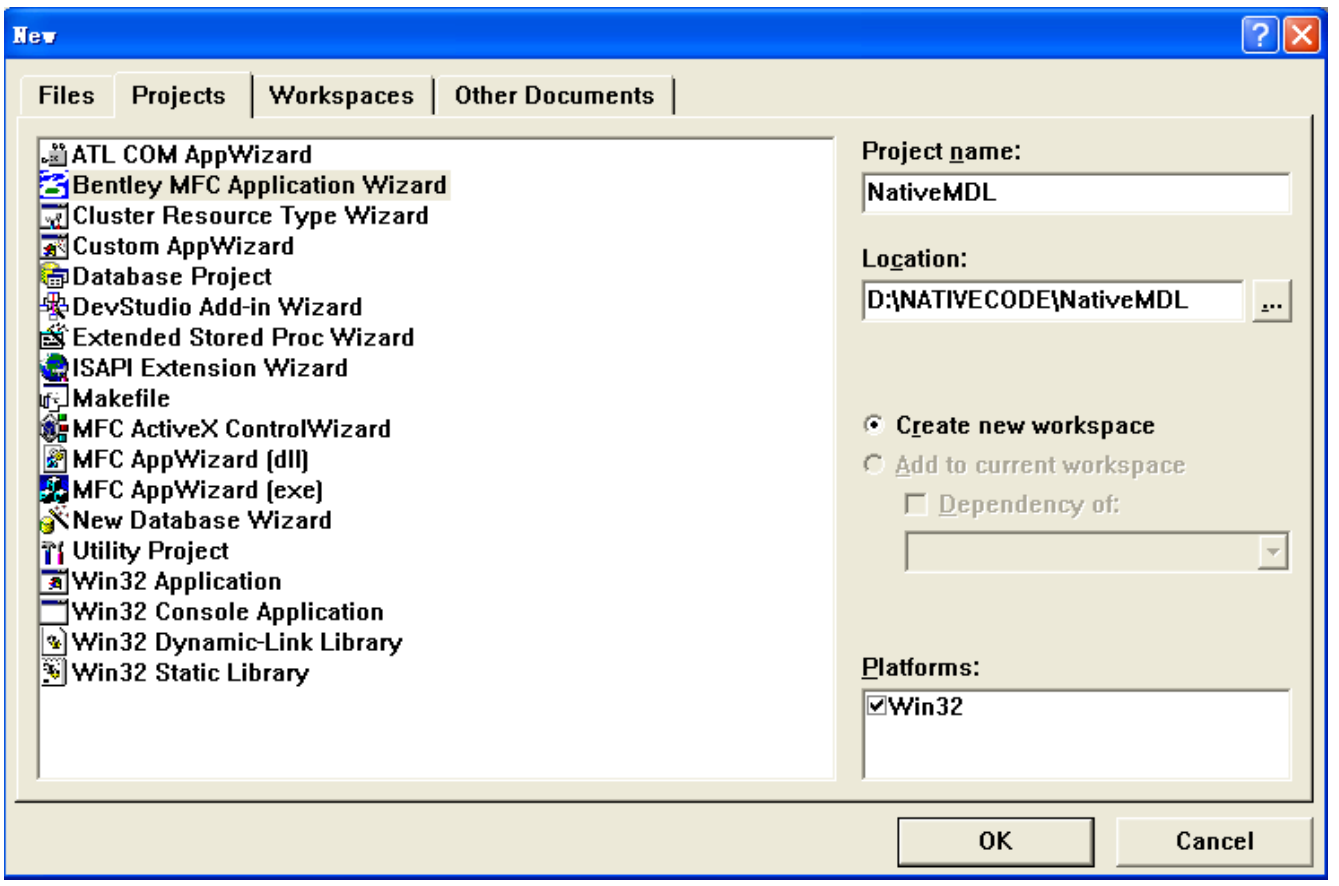


图 9.1　Bentley Native Code Application Wizard

点击 OK 按钮后进入向导的第一步，选择 Native MDL(cpp)和 DLL accessed from key-in and MDL Dialog buttons，如图 9.2 所示。

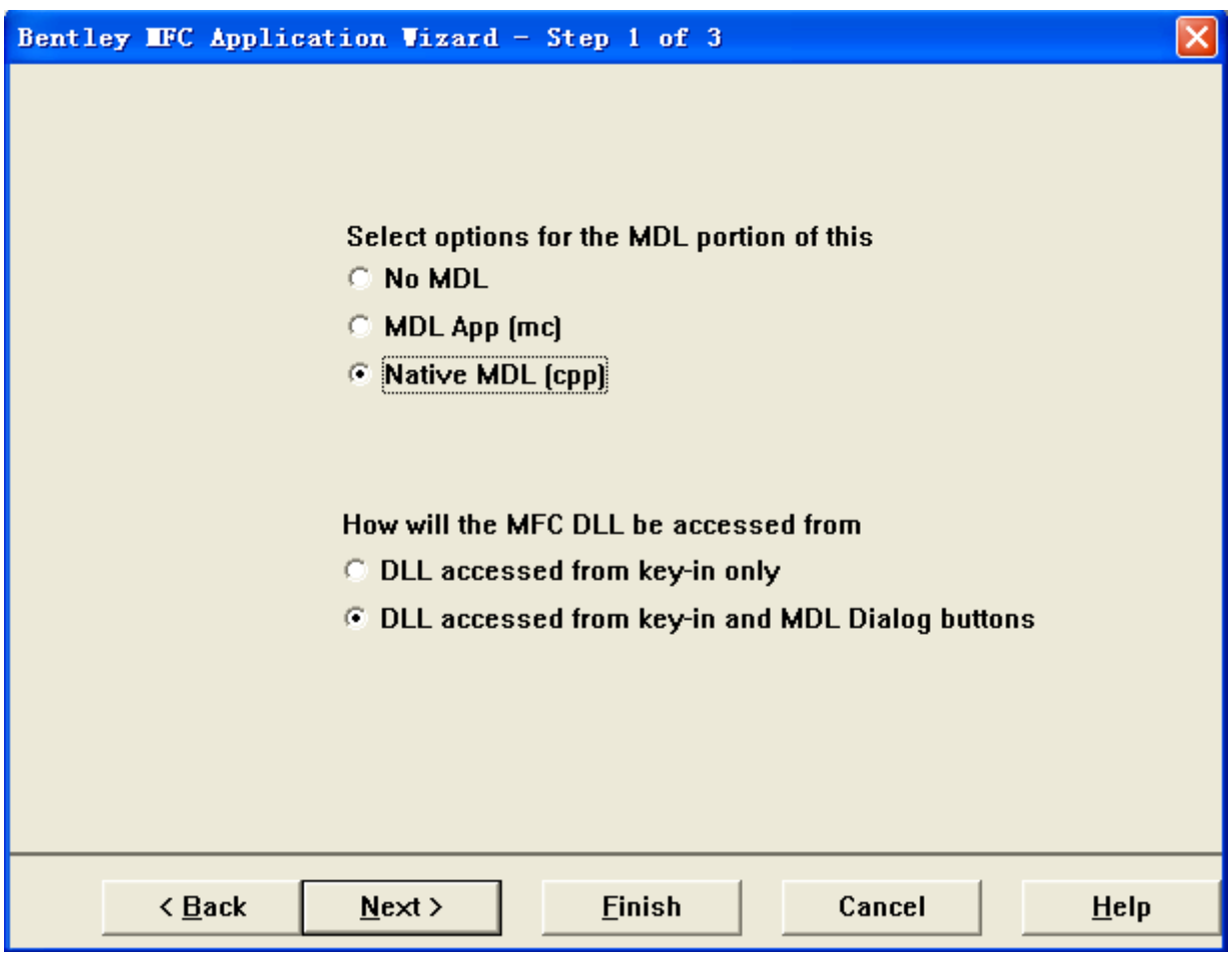


图 9.2　向导步骤一

点击 Next 按钮出现向导的第二步，选择项目中可能用到的各种对话选项，这里我们钩选所有选项，如图 9.3 所示。

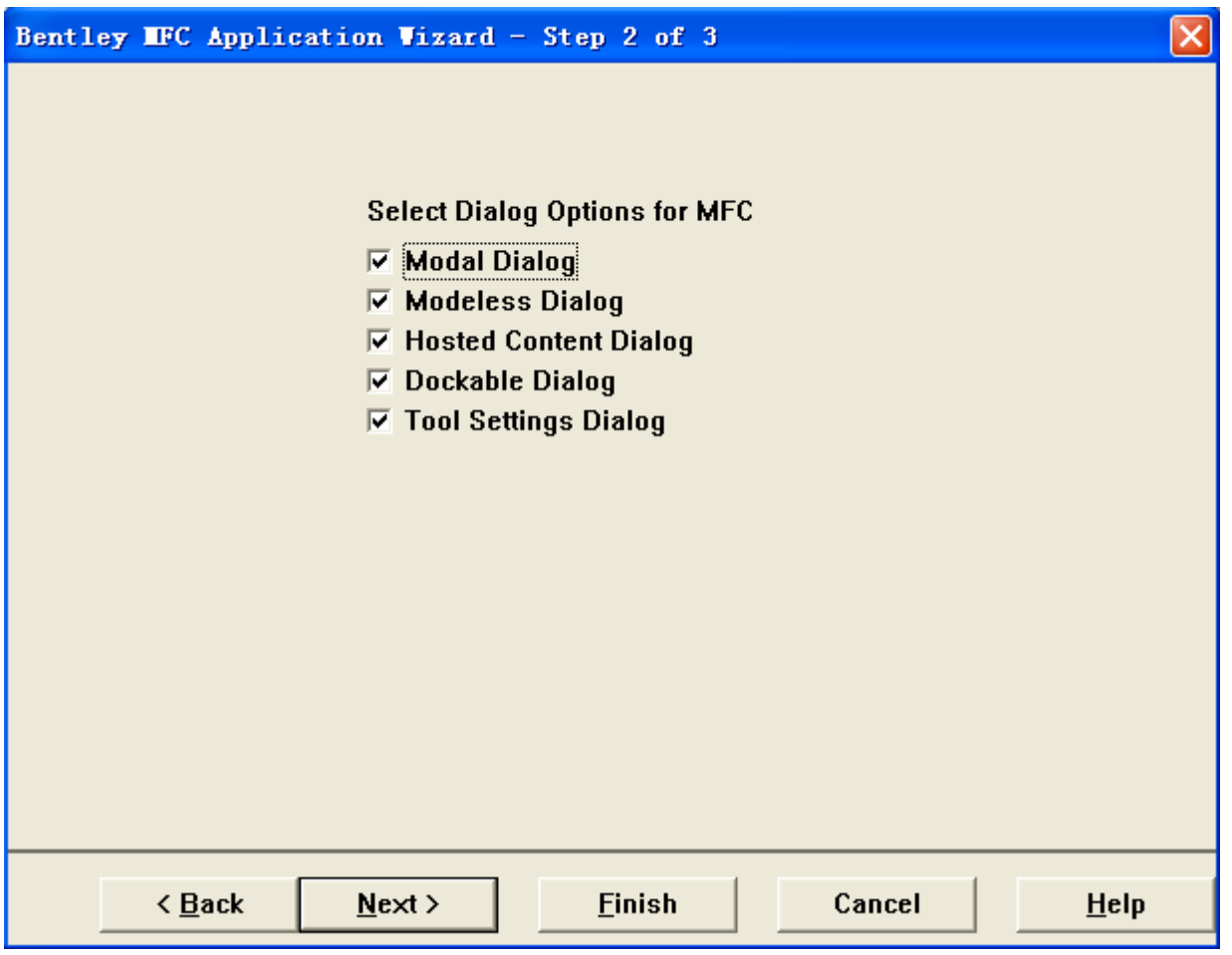


图 9.3　向导步骤二

再点击 Next 按钮出现向导的第三步，在 Path to MicroStation 后选择您机器的 MicroStation V8 的安装目录，其它三个目录会自动设置，如果不符合您机器上的设置可手动改变。如图 9.4 所示。

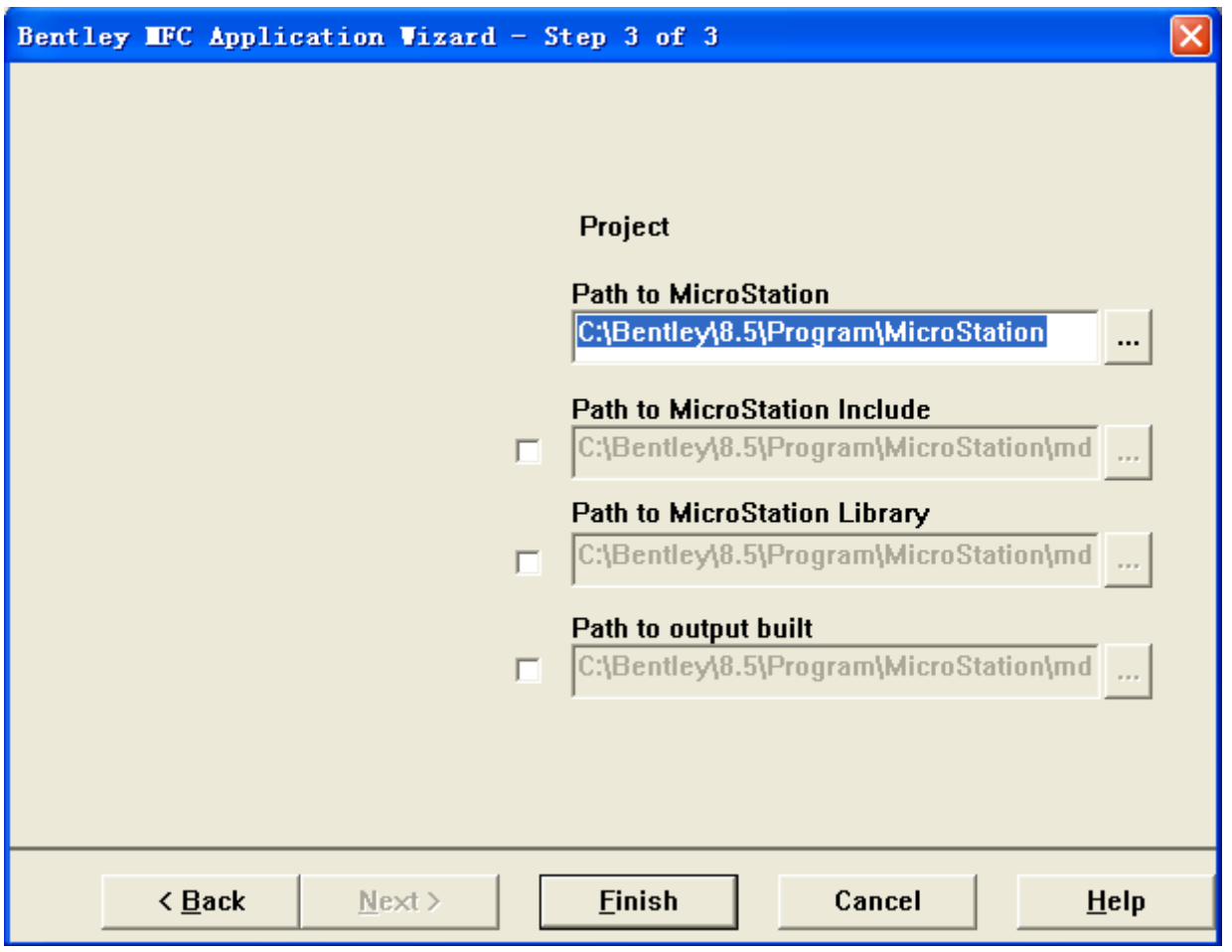


图 9.4　向导步骤三

进一步点击 Finish 按钮和接下来出现的 OK 按钮后完成向导。

9.1.3 编译链接生成 DLL 和 MA

编译链接分两个步骤，第一步是生成 NativeMDLMFC.DLL(同时还生成了 NativeMDLMFC.LIB 和 NativeMDLMFC.EXP)，第二步是生成 NativeMDL.DLL 和 NativeMDL.MA。

在 Visual Studio 中选菜单 Build→Rebuild All 即可生成 NativeMDLMFC.DLL。缺省为 Debug(调试)版本，文件比较大。可通过设置 Build→Set Active Configuration 将当前项目配置为 Release(发布)版本，这样生成的

NativeMDLMFC.DLL 会小很多。

要生成 NativeMDL.DLL 和 NativeMDL.MA 有两种方法：其一是直接采用 MicroStation Development Shell 命令窗口，其二可以直接在 Visual Studio 中加以配置后也集成在 Visual Studio 中执行。

我们先用第一种方法生成 NativeMDL.DLL 和 NativeMDL.MA。从 Windows 的“开始”处选“所有程序→MicroStation→MicroStation Development Shell”，弹出一个黑色背景的命令窗口，进入我们项目程序所在目录(键入命令 D:回车，再键入命令 CD \NativeCode\NativeMDL\MDL 回车)，键入命令 bmake nativemdl.mke 并回车即可生成以上所述的两个文件，如图 9.5 所示。

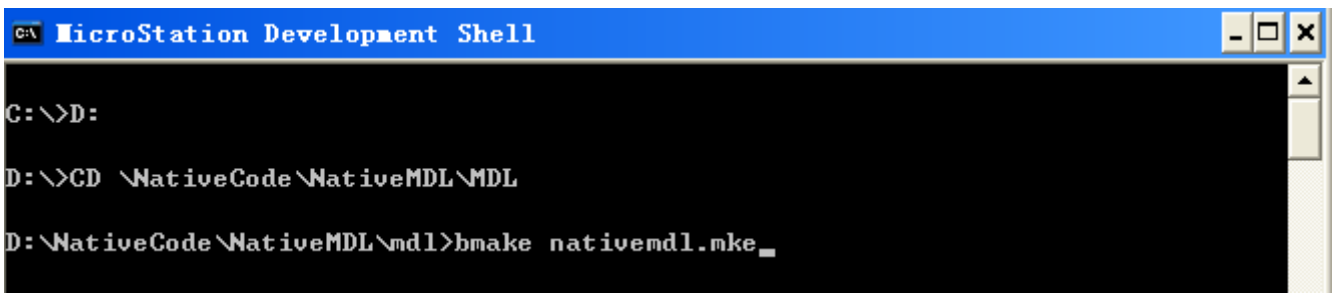


图 9.5　生成 NativeMDL.DLL 和 NativeMDL.MA 的命令窗口界面

至此，在…Bentley\Program\MicroStation\mdlapps 目录下共生成了 5 个文件，如图 9.6 所示。实际运行的文件只需要后三个，在将应用发布给用户时可将前两个文件删除。

图 9.6　生成的 NativeMDL 应用程序文件列表

下面讨论如何在 Visual Studio 中调用 bmake 程序来编译链接从而生成 MA 程序。如果你不喜欢那个背景黑乎乎的命令窗口的话，可以耐心地按照如下步骤设置。首先在任意目录下建立一个叫做 buildApp.bat 的批处理文件(我建立在 C:\Program Files\Microsoft Visual Studio 目录下)，其内容如下：

```
@echo off

SET MICROSTATIONDRIVE=C:\
SET MICROSTATIONDIR=Bentley\8.5\Program\MicroStation
SET CTOOLSDRIVE=C:\
SET CTOOLSDIR=Progra~1\Micros~4\

IF "%MS%." == "." SET MS=%MICROSTATIONDRIVE%%MICROSTATIONDIR%

IF "%NTTOOLS%."== "." SET NTTOOLS =%CTOOLSDRIVE%%CTOOLSDIR%vc98\

Call %CTOOLSDRIVE%%CTOOLSDIR%vc98\bin\vcvars32.bat

set BMAKE_OPT=-I%MS%\mdl\include
set PATH=;%MS%;%MS%\mdl\bin;%PATH%
set           MLINK_STDLIB=%MS%\mdl\library\builtin.dlo           %MS%\mdl\library\dgnfileio.dlo
%MS%\mdl\library\toolsubs.dlo

rem Create the following folders if they do not exist
if not exist %MS%\mdl\objects. mkdir %MS%\mdl\objects
if not exist %MS%\mdl\reqdobjs. mkdir %MS%\mdl\reqdobjs
if not exist %MS%\mdl\rscobj. mkdir %MS%\mdl\rscobj

bmake %1 %2 %3

REM ** Un-Set temp variables **
SET MICROSTATIONDRIVE =
```

```
SET MICROSTATIONDIR =
SET CTOOLSDRIVE=
SET CTOOLSDIR=
```

在这个批处理文件中，要根据您机器的具体情况修改其中的 MICROSTATIONDRIVE、MICROSTATIONDIR、CTOOLSDRIVE、CTOOLSDIR 四个环境变量使其分别指向 MicroStation V8 所在盘符、MicroStation V8 安装目录(不含盘符)、Visual Studio6 所在盘符、Visual Studio6 安装目录(不含盘符)。

然后，在 Visual Studio 中选菜单 Tools→Customize…打开 Customize 对话框，切换到 Tools 页，在最后一行加入 BentleyBuildTool，按如图 9.7 所示设置 Command、Arguments、Initial directory 并钩选 Use Output Window。点击该对话框右下角 Close 按钮后就在 Visual Studio 的 Tools 菜单下增加了一项 BentleyBuildTool。

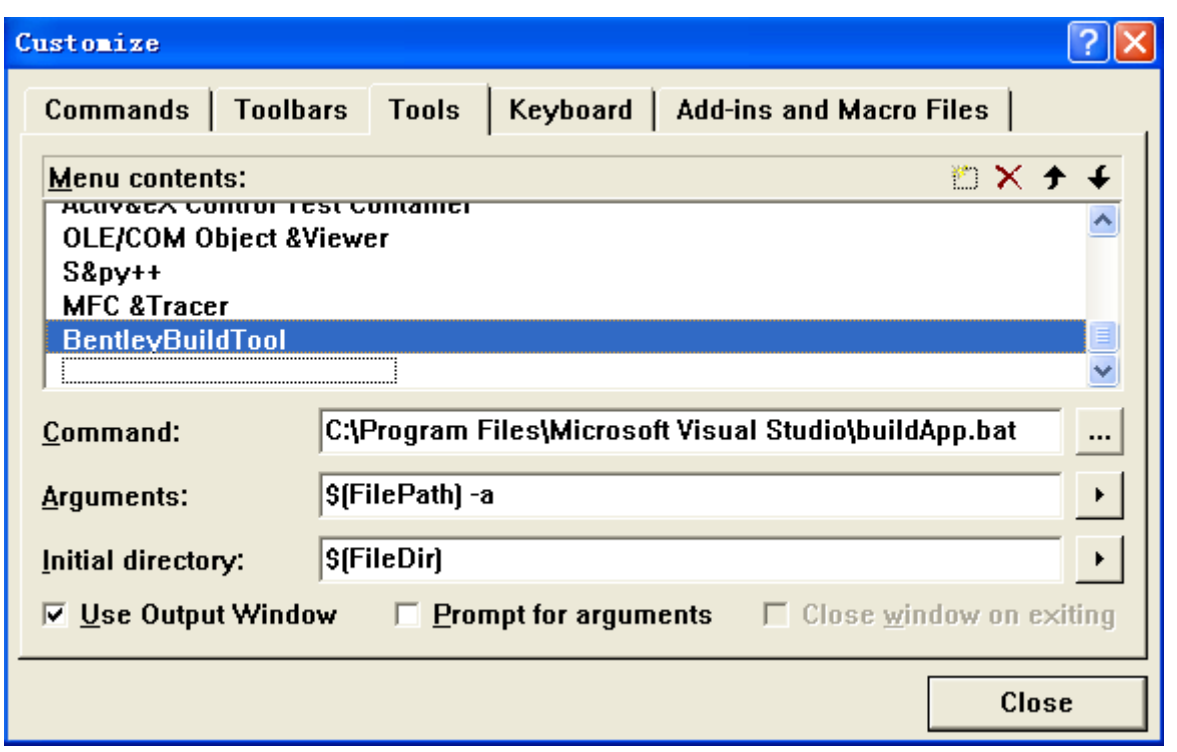


图 9.7　建立 BentleyBuildTool

一旦建立了 BentleyBuildTool 就可以在 Visual Studio 中生成 NativeMDL.MA 了。双击 Visual Studio 左侧 Workspace 列表中的 NativeMDL.mke 文件打开它，点击菜单 Tools→BentleyBuildTool 即可生成 NativeMDL.DLL 和 NativeMDL.MA，如图 9.8 所示。

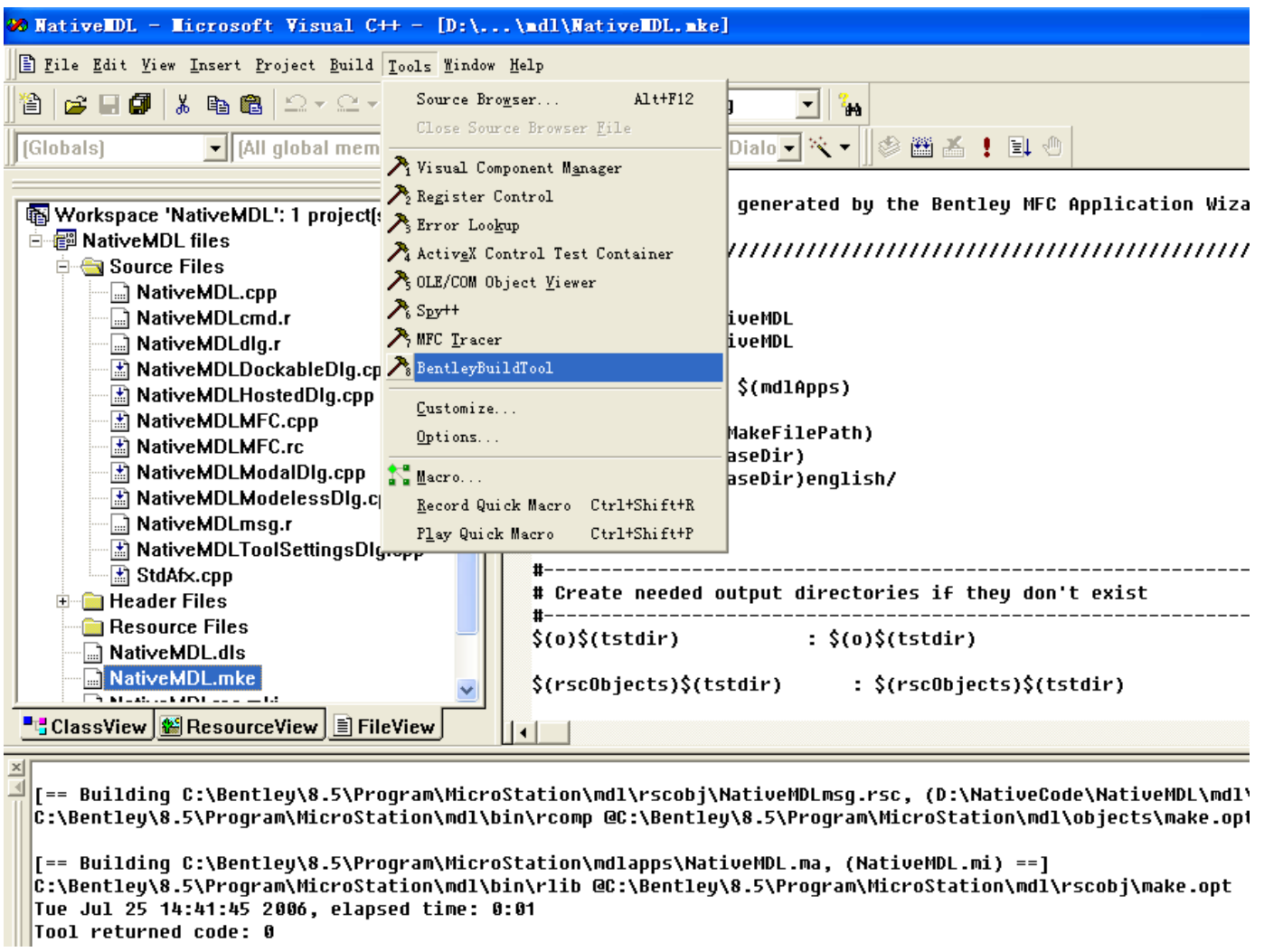


图 9.8　应用 BentleyBuildTool 生成 MA

9.1.4 在 MicroStation 中运行

启动 MicroStation 2004，在命令键入域键入 MDL LOAD NativeMDL 并回车(或从 Utilities→MDL Applications 中 Load 亦可)装载 NativeMDL 应用，进一步键入 NativeMDL Dialog Open 将出现如图 9.9 所示的主对话框。

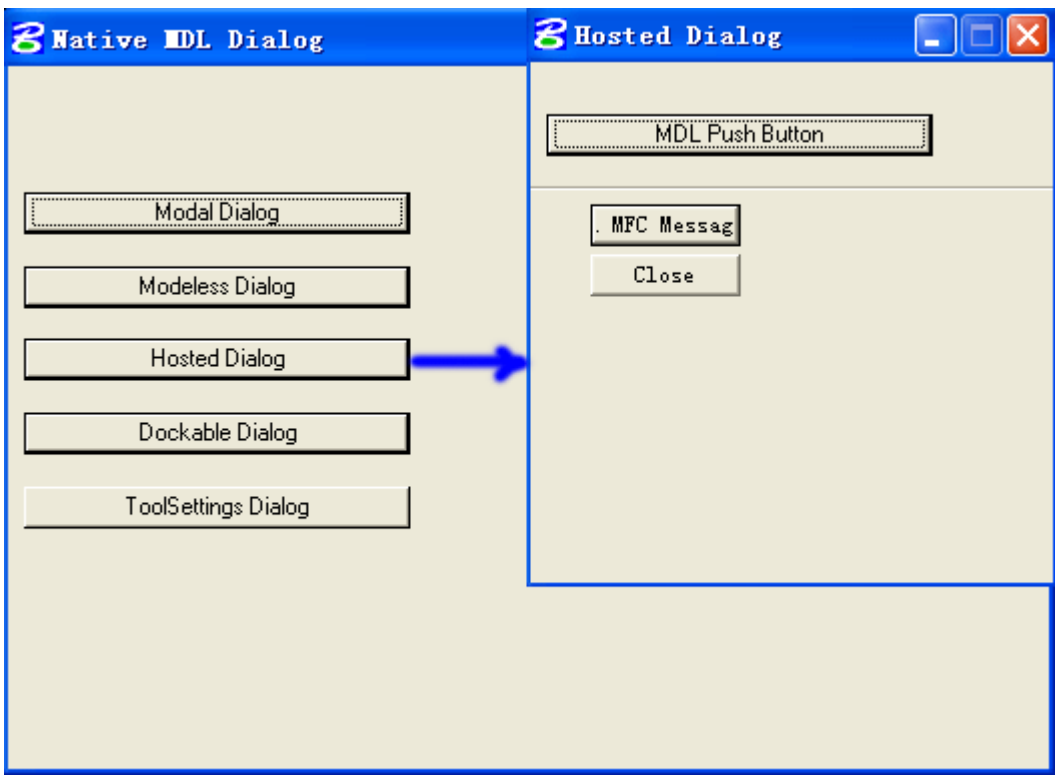


图 9.9　NativeMDL 主对话框

点击该对话框上的每个按钮都可打开子对话框，分别具有模态(Modal)、非模态(Modeless)、宿主(Hosted)、贴边(Dockable)、工具设置(ToolSettings)特点。图中还表示出了 Hosted Dialog 的样子，这个对话框的框体和 MDL Push Button 是在 MDL 的 NativeMDLdlg.r 资源文件中定义的，而下面的两个按钮.MFC Messag(注：全称应该为 Call MFC MessageBox，由于所给按钮区域太小而未显示出全部内容)和 Close 则是在 MFC 侧的 NativeMDLMFC.rc 文件中定义的，框体为“主”，下面的两个按钮寄宿在 MDL 框体上，这也正是“宿主”的含义所在。

点击图 9.9 中的 ToolSettings Dialog 按钮还能打开一个取代系统工具设置对话框的一个对话框，在此对话框的 Text to place 文本框中键入一个字符串，把光标放至视图区可动态放置您键入的字符串，如图 9.10 所示。

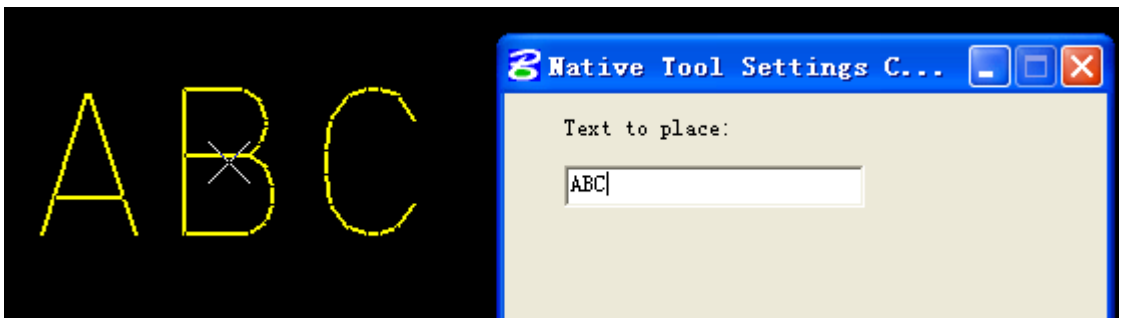


图 9.10 ToolSettins Dialog 放置文本串演示

9.2 NativeCode 程序结构分析

上一节详细介绍了 NativeMDL 应用的生成和过程和运行结果，本节将带您进入后台程序代码分析每一部分的组成。下图列出了各主要程序的名称及作用。

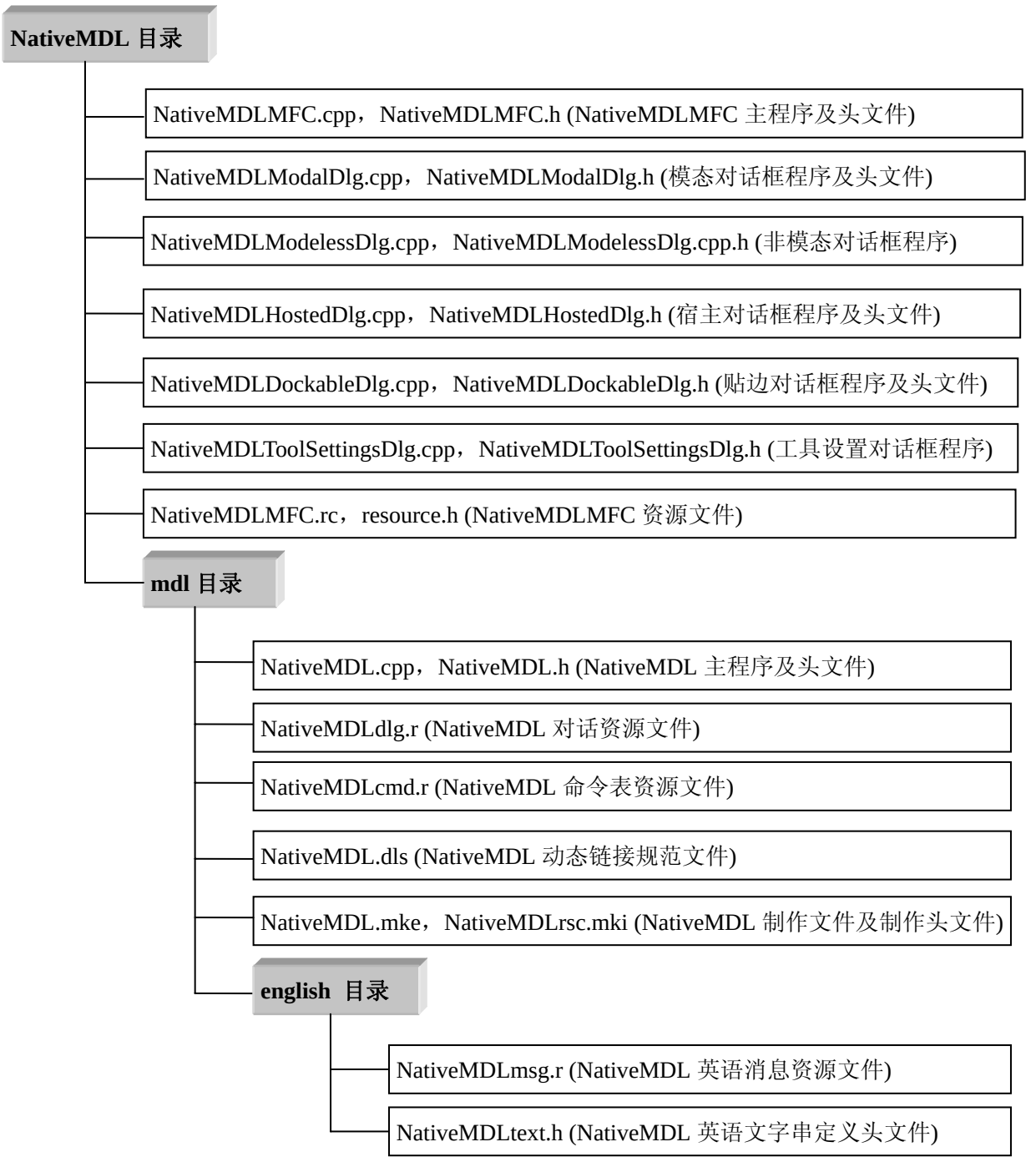


图 9.11 NativeMDL 应用源程序主要清单

9.2.1 MDL 制作文件及资源文件分析

在 NativeMDL.mke 中有如下定义：

```
DLM_LIBRARY_FILES    = $(mdlLibs)dgnfileio.lib \
            $(mdlLibs)toolsubs.lib \
            $(mdlLibs)ditemlib.lib \
   $(mdlLibs)mdllib.lib \
   $(mdlLibs)mgdshook.lib \
   $(mdlLibs)nativewindow.lib \
   C:\Bentley\8.5\Program\MicroStation\mdlapps\NativeMDLMFC.lib
```

这决定了生成的 NativeMDL.DLL 需要调用 NativeMDLMFC.DLL，关联了 MDL 与 MFC 程序。

```
$(o)$(appName)$(oext)    : $(baseDir)$(appName).cpp
```

使 NativeMDL.cpp 被编译为 NativeMDL.obj。

```
%include dlmlink.mki
```

使 NativeMDL.obj 进一步链接成 NativeMDL.DLL。

```
%include $(appName)rsc.mki
```

包含了 NativeMDLrsc.mki，在该文件中是传统的 MDL 制作文件语句，会生成最终的 NativeMDL.MA。

在 NativeMDLcmd.r 文件中有如下语句：

```
#define     DLLAPPID 1
/* associate app with dll */
DllMdlApp DLLAPPID =
    {
```

```
    "NativeMDL", "NativeMDL"
    }
```

这决定了当我们在 MicroStation 中键入 MDL LOAD NativeMDL 时会同时加载 NativeMDL.DLL 并执行。其中的 DllMdlApp 是 MicroStation V8 为开发 NativeCode 新增加的资源类别。

NativeMDLcmd.r 中的其它部分、NativeMDLdlg.r、NativeMDLmsg.r 都是标准的 MDL 资源，我们在前几章中都涉及过，这里就不再赘述。

9.2.2 MDL 主程序文件分析

在 NativeMDL.cpp 中有

extern "C" DLLEXPORT int MdlMain (int argc, char *argv[])

这是 C++形式的 MDL 程序主入口，具体函数体的内容和普通 MDL 程序类似，但其中有一行

```
mdlSystem_registerCommandNumbers (commandNumber);
```

是新的命令号注册函数。因为在 C++形式的 MDL 程序中不再支持 cmdName 和 cmdNumber 关键字，需要用 MdlCommandNumber 结构关联命令号和执行函数，NativeMDL 中的具体定义如下：

```
Private MdlCommandNumber commandNumber[] =
    {
    {(Handler) NativeMDL_openMainDialog,         CMD_NATIVEMDL_DIALOG_OPEN},
    {(Handler) NativeMDL_runModalDialog,         CMD_NATIVEMDL_RUN_MODALDIALOG},
    {(Handler) NativeMDL_runModelessDialog,      CMD_NATIVEMDL_RUN_MODELESSDIALOG},
    {(Handler) NativeMDL_runHostedDialog,        CMD_NATIVEMDL_RUN_HOSTEDDIALOG},
    {(Handler) NativeMDL_runMDLButton,           CMD_NATIVEMDL_RUN_MDLBUTTON},
    {(Handler) NativeMDL_runDockableDialog,      CMD_NATIVEMDL_RUN_DOCKABLEDIALOG},
    {(Handler) NativeMDL_runToolSettingsDialog,  CMD_NATIVEMDL_RUN_TOOLSETTINGSDIALOG},
    0
    };
```

注意以上结构中最后一行的 0，它是必须的。在 NativeMDL.cpp 中还有如下外部引用函数的定义：

```
extern "C"    int NativeMDLMfc_OpenModalDialog( void );
extern "C"    int NativeMDLMfc_OpenModelessDialog( void );
extern "C"    void *NativeMDLMfc_OpenHostedContent( DialogBox * );
extern "C"    int NativeMDLMfc_OpenDockableDialog( void );
extern "C"    void *NativeMDLMfc_OpenToolSettingsContent( void );
extern "C"    int NativeMDLMfc_CloseToolSettingsContent( void * );
extern "C"    char const * NativeMDLMfc_GetEditBoxText( void * );
```

这些函数的实际函数体都在 NativeMDL 目录下的各个.cpp 文件中，它们组成了标准 VC 项目 NativeMDLMFC 中的主体部分。

下面看一看 NativeMDL 和 NativeMDLMFC 是如何被关联起来的。当我们加载 NativeMDL 后，首先会执行 NativeMDL.cpp 中的 MdlMain 函数,在该函数中通过 mdlSystem_registerCommandNumbers 建立了各命令号和函数之间的关系，当我们点击图 9.9 所示的 Modal Dialog 按钮时，NativeMDL 应用会向 MicroStation 发出一个 CMD_NATIVEMDL_RUN_MODALDIALOG 命令(这一点是由 NativeMDLdlg.r 中的 PUSHBUTTONID_ModalDialog 资源定义的)，该命令会引发 NativeMDL.cpp 中的 NativeMDL_runModalDialog 函数的执行，该函数的内容如下：

```
extern "C" /*DLLEXPORT*/Public void NativeMDL_runModalDialog
(
char            *unparsed
)
    {
    NativeMDLMfc_OpenModalDialog();
    }
```

不难看出，该函数又进一步调用了 NativeMDLModalDlg.cpp 中的 NativeMDLMfc_OpenModalDialog 函数。其它几个命令的工作过程与此类似的。

9.2.3 MFC 程序文件分析

NativeMDLMFC.cpp 是 MFC 部分的主程序，其中在初始化 CNativeMDLMFCApp 的实例 theApp 时会调用

CNativeMDLMFCApp::InitInstance()函数，这个函数中执行了如下一行重要调用：

```
mdlNativeWindow_initialize("NativeMDL");
```

它是本机窗口(即指用操作系统级的外部程序设计的窗口)支持子系统的初始化函数，所有其它的mdlNativeWindow_函数的调用都必须在其后才能起作用。如果深入到MFC的后台去分析，你会发现在C:\Bentley\8.5\Program\MicroStation\mdl\mfc目录下的几个.cpp文件中存在大量的mdlNativeWindow_函数的调用，正是这些调用才简化了前台程序的编写，这也正体现了MFC的功能强大所在。

下面让我们拿NativeMDLMFCModalDlg为例来作进一步的分析，其它几个MFC程序结构都很类似。NativeMDLMFCModalDlg.cpp的主要部分如下：

```
extern “C” {
    __declspec(dllexport) int NativeMDLMfc_OpenModalDialog( void )
    {
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    NativeMDLModalDlg *pModalDlg;
    pModalDlg = new NativeMDLModalDlg();
    pModalDlg->ShowModal();
    delete pModalDlg;
    return true;
    }
}
```

extern “C”强制函数NativeMDLMfc_OpenModalDialog按C语言规范而不是C++语言规范调用；__declspec(dllexport)表示其后的函数作为DLL的导出函数(注：也可以通过定义.DEF文件来指定导出函数。所谓导出函数就是能被其它外部程序调用的DLL中的函数，未声明为导出函数的DLL中的函数不能被外部程序所调用)，该函数中用到了一个主要的类NativeMDLModalDlg，这个类是从CBModalDialog派生而来的，这一点可从NativeMDLMFCDlg.h文件中的如下行看出。

```
class NativeMDLModalDlg : public CBModalDialog
```

而类CBModalDialog的详细定义又位于…\MicroStation\mdl\include\mfc\CModalDialog.h和…\MicroStation\mdl\mfc\CModalDialog.cpp中。从中不难看出，CBModalDialog是从MFC的标准类CDialog中派生的，当构造CModalDialog时，调用了mdlNativeWindow_getMainHandle函数取得MicroStation主窗口的句柄；当初始化CModalDialog时，调用了mdlNativeWindow_createMSWindow建立一个反映本机窗口的MSWindow实例，同时调用mdlNativeWindow_getPreviousPosition取得窗口上一次的位置；当CModalDialog被注销时，先调用mdlNativeWindow_savePosition保存当前窗口位置已备下次恢复位置之用，尔后调用mdlNativeWindow_destroyMSWindow注销MSWindow实例，注意，mdlNativeWindow_destroyMSWindow函数并不关闭或注销本机窗口，本机窗口的关闭和注销还是靠NativeMDLMFCModalDlg.cpp中的delete pModalDlg来完成的。

9.3 调试NativeCode程序

由于NativeCode程序是直接依赖于操作系统的DLL，不像MA那样完全受MicroStation的控制，因而程序的调试方法也完全不同于第二章所描述的那样。如果您对Visual Studio很熟悉，就会很快适应这种新的程序调试方法，它能完全发挥Visual Studio强大的程序调试功能。

9.3.1 制作调试版本的NativeCode程序

参见本章9.1.3所述，先制作NativeMDLMFC.DLL的调试版本：(1). 在Visual Studio中选菜单Build→Set Active Configuration，进一步选择NativeMDL－Win32 Debug； (2). 选菜单Build→Rebuild All即可制作完成带有调试信息的NativeMDLMFC.DLL。

再制作NativeMDL.DLL的调试版本：(1). 启动MicroStation Development Shell命令窗口，通过键入命令进到D:\NativeCode\NativeMDL\mdl目录； (2). 键入bmake －a －ddebug NativeMDL.mke并回车。

9.3.2 建立针对 MicroStation 的调试项目并设置调试参数

启动 MicroStation 2004，在 Visual Studio 中选菜单 Build→Start Debug→Attach to Process…，出现如图 9.12 所示的选择连接进程对话框，在其中选中 ustation 并点击 OK 按钮完成。【如果在如下图中看不到任何进程，可能是您的 VS6 没有安装服务包(SP)，请从微软网站下载 VS6SP6 服务包安装后再试。下载地址如下：http://download.microsoft.com/download/1/9/f/19fe4660-5792-4683-99e0-8d48c22eed74/Vs6sp6.exe】

此时 MicroStation 进程被激活至前台，我们还需要切换回 Visual Studio 进行调试参数的设置。

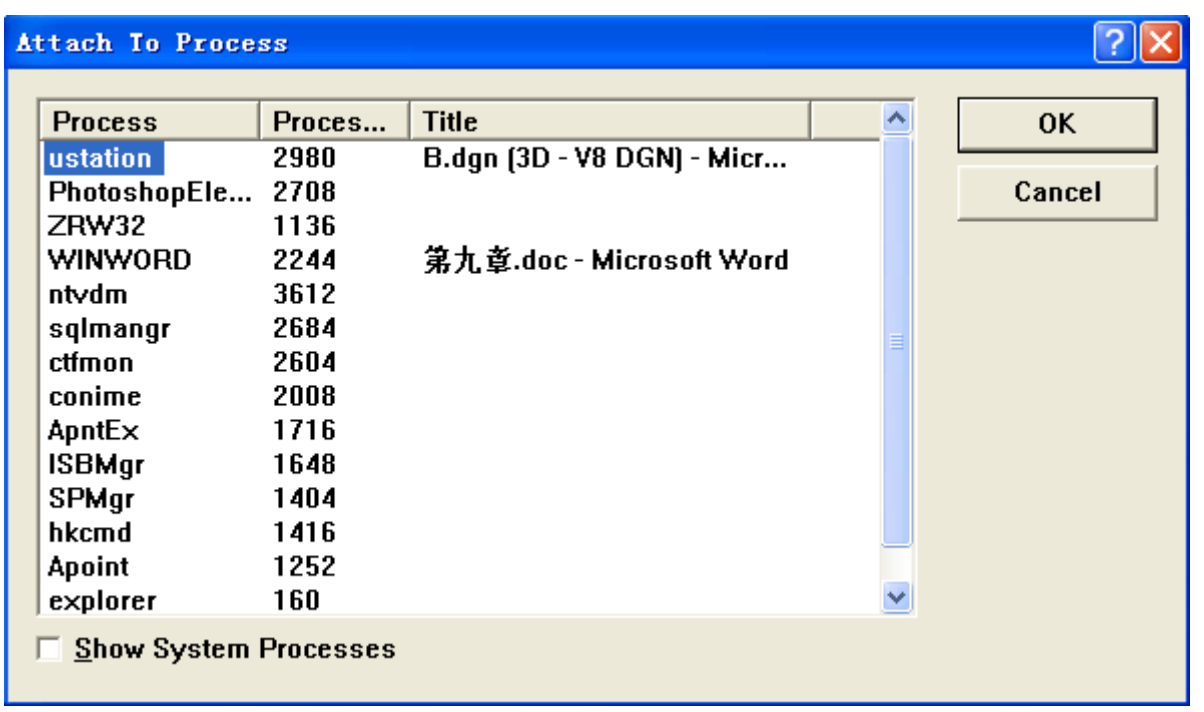


图 9.12 Visual Studio 中的选择连接进程对话框

在 Visual Studio 中选菜单 Project→Settings…打开 Project Settings(项目设置)对话框，在该对话框右侧选 Debug(调试)页，在该页的 Category(分类)下选 Additional DLLs，再在 Modules 的 Local Name 下增加 C:\Bentley\8.5\Program\MicroStation\mdlapps\NativeMDL.DLL，如图 9.13 所示。

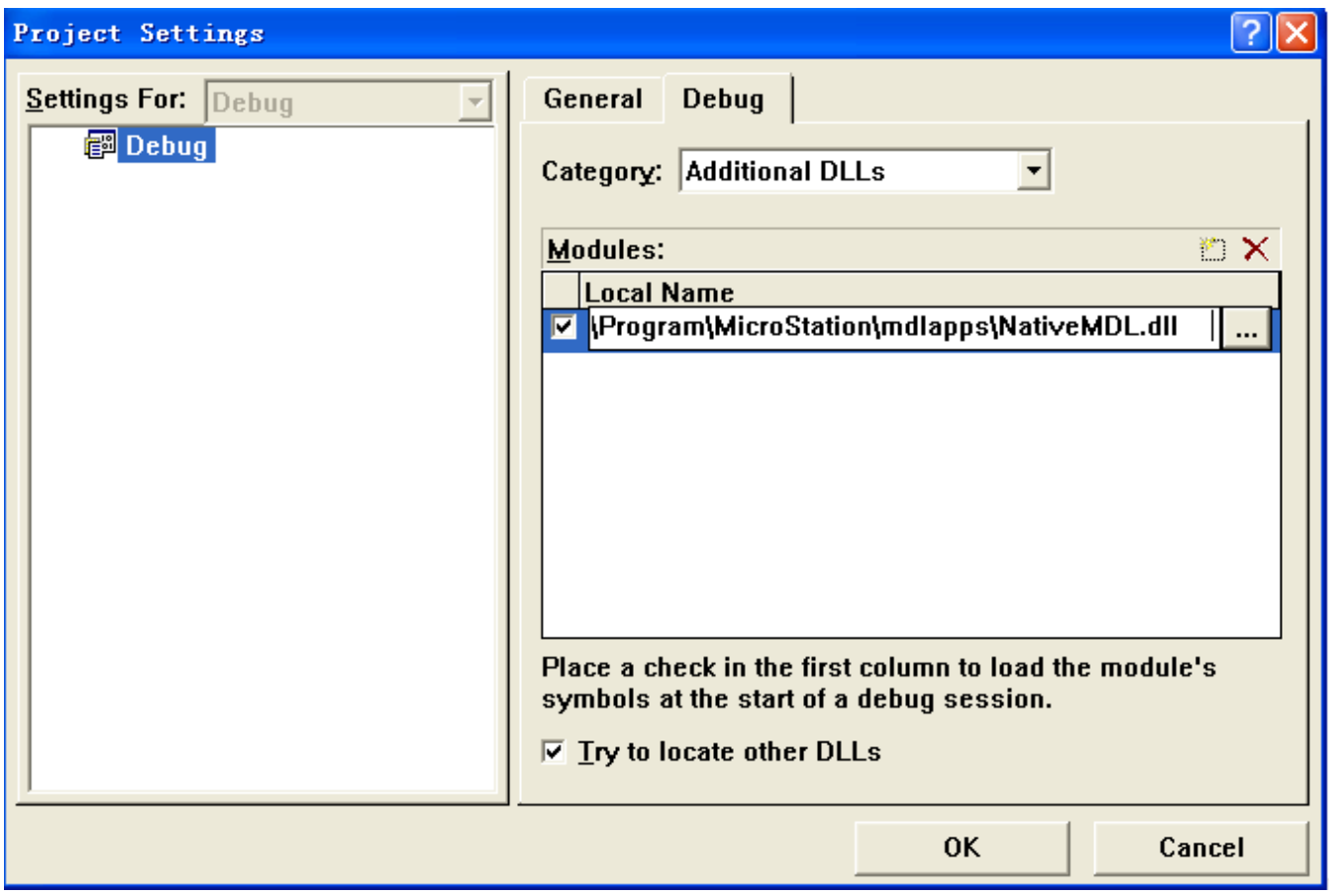


图 9.13 VS 中设置附加 DLL 的对话框

设置完附加 DLL 后还需要设置程序执行断点(Breakpoint)。在 Visual Studio 中选菜单 Edit→Breakpoint 打开如图 9.14 所示的断点设置对话框，在该对话框的 Break at 下键入 NativeMDL.DLL 中的一个函数名如 MdlMain 并点击 OK 完成设置。

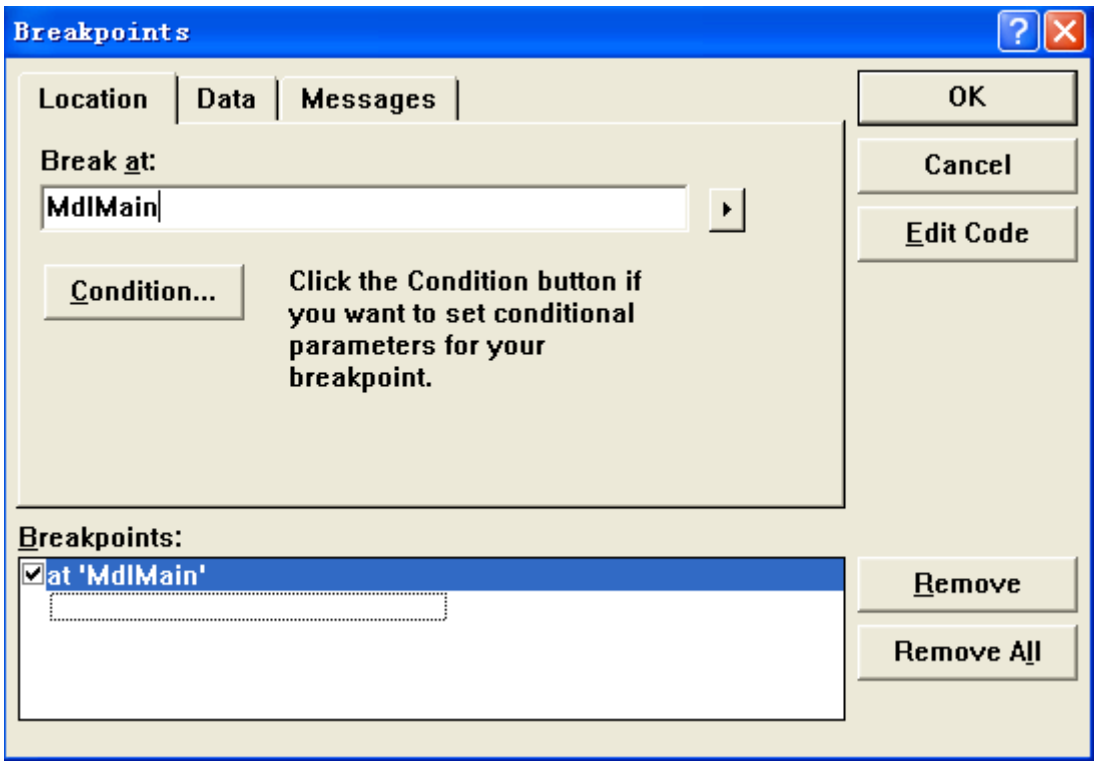


图 9.14 VS 中设置程序断点的对话框

9.3.3 开始调试过程

经过 9.3.2 的设置后切回到 MicroStation 进程，在命令键入域键入 MDL LOAD NativeMDL，此时会立刻启动 Visual Studio 的调试过程，如图 9.15 所示。如何使用 Visual Studio 对程序进行调试的具体步骤和操作有许多文章论述，在此就不多说了。

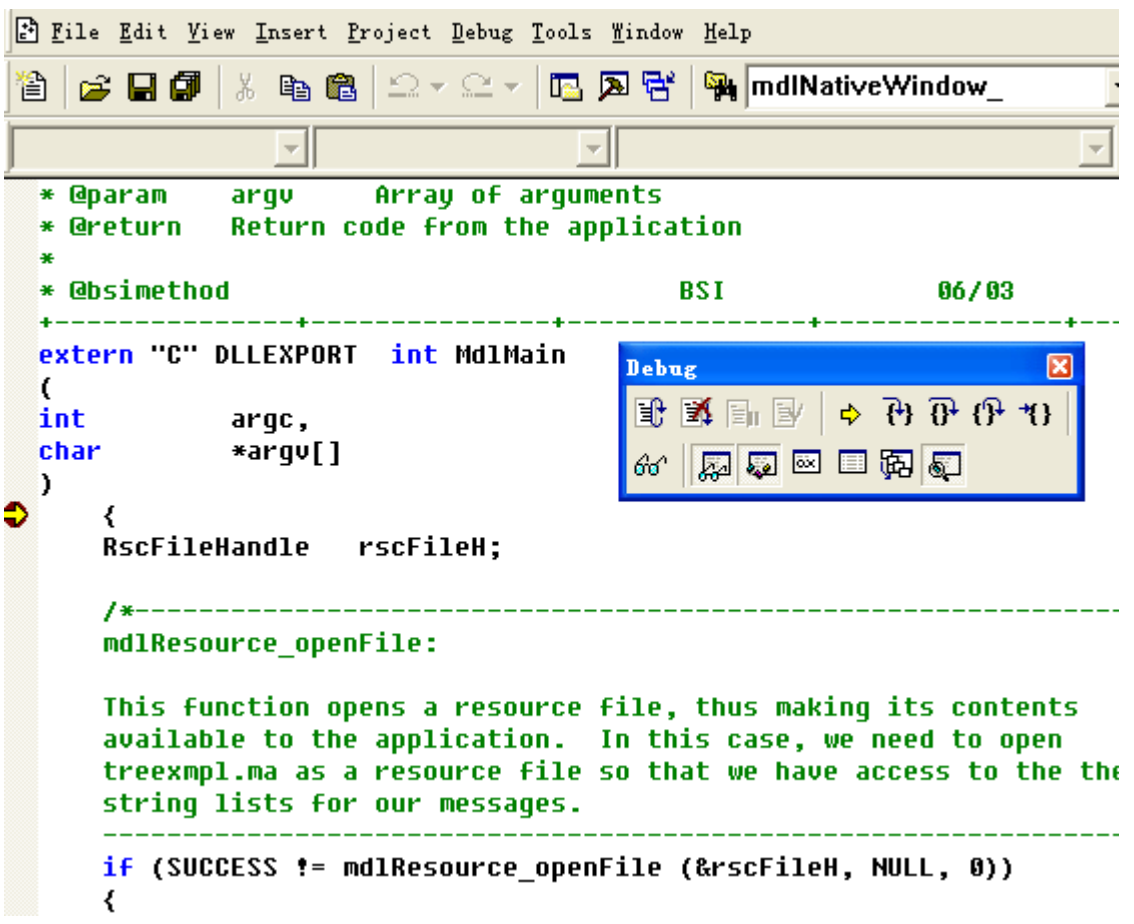


图 9.15 VS 中的调试界面

9.4 NativeCode 编程注意事项

由.mc 过渡到.cpp 需要有一个过程，但这是大势所趋。只要掌握了本章讲述的内容加之多实践、多练习，这种过渡应该不会太难做到。在 NativeCode 编程中，除了命令号处理方式有了大的变动以外，以下几点也是需要注意的：

9.4.1 内存管理

在原来的 MDL 应用中可用 malloc、calloc、realloc 分配内存，用 free 释放内存，当 MDL 卸载时 MicroStation

会自动释放 MDL 应用分配的内存，因而对 free 的要求并不十分严格。

过渡到 DLL 中后，这些函数需要分别用 dlmSystem_mdlMalloc、dlmSystem_mdlCalloc、dlmSystem_mdlRealloc、dlmSystem_mdlFree 来代替。同时必须明确用 dlmSystem_mdlFree 释放不用的内存，因为 DLL 卸载时不会自动释放其占用的内存。不注意这一点的话就很容易造成内存泄漏。

9.4.2 等待消息完成

在传统的 MDL 程序中，有一个 mdlInput_waitForMessage 函数，该函数能挂起 MDL 任务直到另一个消息到来(链接这样的应用时必须加上-csoff 选项)。但在 NativeCode 中该特性不再被支持！因此，MicroStation 2004 提供了一套新的 mdlInput_sendSynchronizedXXX 函数用来完成类似的工作。它们的对应关系如下：

```
mdlInput_sendSynchronizedCommand   → mdlInput_sendCommand
mdlInput_sendSynchronizedKeyin     → mdlInput_sendKeyin
mdlInput_sendSynchronizedKeyinFrom → mdlInput_sendKeyinFrom
mdlInput_sendSynchronizedKeystroke → mdlInput_sendKeystroke
mdlInput_sendSynchronizedMessage   → mdlInput_sendMessage
mdlInput_sendSynchronizedReset     → mdlInput_sendReset
mdlInput_sendSynchronizedUORPoint  → mdlInput_sendUORPoint
```

9.4.3 使用 TCB 变量

要在.cpp 中使用 TCB 变量，需要 #include <msvar.fdf> ！