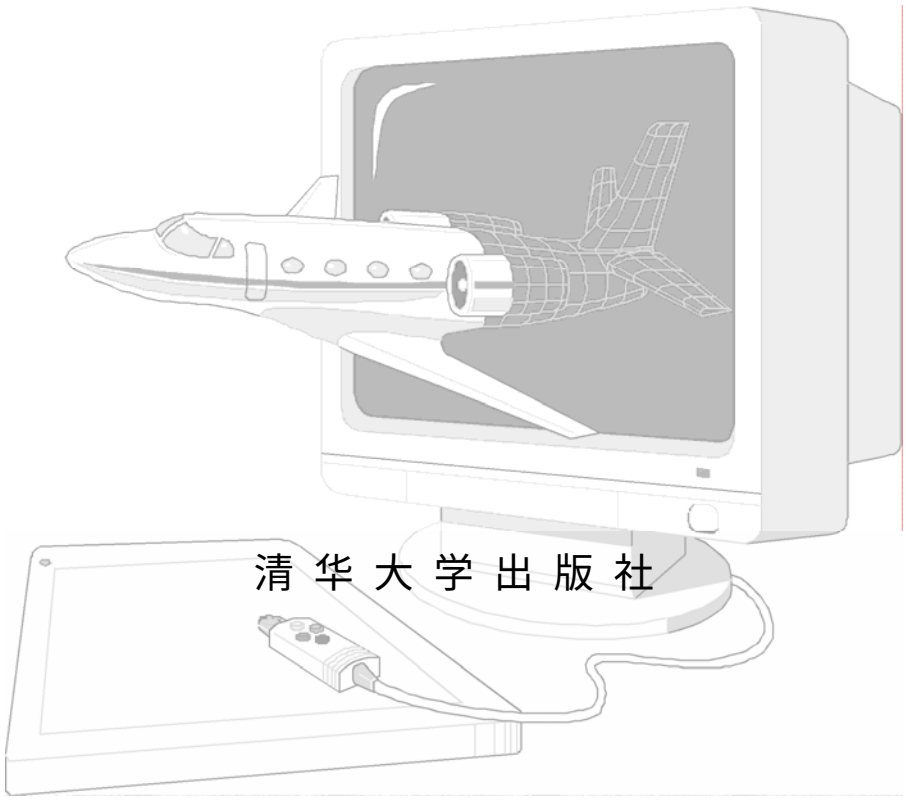
AutoCAD 2000 定制与 Visual LISP 开发技术

李学志　编著



清 华 大 学 出 版 社

需要全本请访问 www.ertongbook.com

（京）新登字 158 号

内 容 简 介

本书系统地介绍了对 AutoCAD 2000 的定制及 Visual LISP 的开发技术。主要内容有：建立专用的线型、图案和符号库，定制带有专业特色的各种菜单、工具栏和对话框，利用图形交换文件编写 C 语言和 AutoCAD 的接口程序，在 Visual LISP 环境下的 AutoLISP 程序设计，定义新的 AutoCAD 命令，扩充 AutoCAD 的功能等。

本书可作为从事 CAD 技术的工程技术人员的参考书，也可作为高等院校或继续教育的培训教材。

版权所有，翻印必究。
本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

书　名：AutoCAD 2000 定制与 Visual LISP 开发技术
作　者：李学志　编著
出版者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
　　http://www.tup.tsinghua.edu.cn
印刷者：北京市振兴印刷厂
发行者：新华书店总店北京发行所
开　本：787×1092 1/16　印张：26.75　字数：614 千字
版　次：2001 年 5 月第 1 版　2001 年 5 月第 1 次印刷
书　号：ISBN 7-302-04447-3/TP・2621
印　数：0001~5000
定　价：39.00 元

前 言

Preface

AutoCAD 是通用性很强的图形系统，它适用于多种领域和行业。由于每个用户所要做的工作只是某一具体的任务，都有其行业、专业的特点，有专业的标准和规范，因此要建立适合于本专业的作图环境，就需要对 AutoCAD 进行定制工作，使其专业化、用户化。只有建立自己的线型、图案、字体及符号库，定制具有便捷、专用、全中文环境的菜单、工具栏和对话框的用户界面，才能提高工作效率。

AutoCAD 具有丰富的绘制和编辑图形的功能，然而仅将其作为取代传统手工绘图的工具是远远不够的，应该对其进行二次开发，扩充现有功能，增加一些具有一定智能化、参数化的命令，将琐碎的画线、标注等操作由程序处理，这样才能使设计人员将主要精力用于产品的结构设计上，才能发挥设计人员的创造性，这才是 CAD 的初衷。

提高产品质量，缩短产品设计、制造周期，降低产品生产成本是企业一贯追求的目标。要实现这一目标仅靠 CAD 是不够的，CAD/CAPP/CAM 数据集成是实现这一目标的重要途径。AutoCAD 只提供了与外部程序或其他系统交换图形信息的接口，要真正实现 CAD/CAPP/CAM 数据集成，必须对 AutoCAD 进行二次开发。

本书的宗旨是：面向广大从事计算机辅助绘图与设计的人员，介绍定制与开发现有 CAD 系统的基本原理和方法，掌握定制与开发 CAD 系统的技术。

本书是作者在多年从事 CAD 教学和科研的基础上编写的，在内容的组织与安排上考虑到读者对象的专业特点和知识结构，尽量做到深入浅出、便于自学。在实例的选取上尽量紧密地联系生产实际。书中列举的大量实例，均以实际应用为背景，其实例本身不仅具有较高的实用价值，而且也便于学习和掌握本书的内容。

本书结合的具体图形系统是 AutoCAD 2000，但所介绍的定制与开发 CAD 系统的原理和方法也适用于 AutoCAD 的其他版本或其他图形系统。

由于 CAD 技术发展迅速、作者的学识和经验有限，书中难免有不当和错误之处，欢迎读者批评指正。

作 者

2000 年 10 月于清华园

目 录

Content

第 1 章 概述 1

1.1 定制与开发 AutoCAD 的目的 1
1.2 定制与开发 AutoCAD 的途径 1
1.3 定制与开发 AutoCAD 的主要内容 2
1.4 可自定义的 AutoCAD 支持的文件 2
1.5 AutoCAD 搜索文件的顺序 3
1.6 命令的验证过程 3

第 2 章 程序参数文件 5

2.1 概述 5
2.2 定义外部命令 5
2.2.1 定义外部命令的格式 5
2.2.2 定义外部命令 6
2.3 为现有的 AutoCAD 命令建立简单易记的别名 7

第 3 章 脚本和幻灯片文件 8

3.1 脚本文件 8
3.1.1 脚本文件的特点 8
3.1.2 建立脚本文件 8
3.1.3 运行脚本文件 9
3.2 幻灯片文件 10
3.2.1 幻灯片文件的特点 10
3.2.2 有关幻灯片文件的命令 10
3.2.3 产品的播放演示 10
3.2.4 利用脚本文件编写接口程序 11

需要全本请访问 www.ertongbook.com

第 4 章　形（符号）和字体 14

4.1　概述 14
4.2　形的定义 14
4.2.1　形文件 14
4.2.2　形的标题行和描述行 14
4.2.3　专用代码 15
4.2.4　矢量描述码 15
4.2.5　用比例因子缩放矢量的大小 18
4.2.6　利用堆栈存储位置 19
4.2.7　引用子形 19
4.2.8　双向书写 20
4.3　综合举例 20
4.4　形的编译、装入和调用 21
4.4.1　COMPILE 命令 21
4.4.2　LOAD 命令 22
4.4.3　SHAPE 命令 22
4.5　字体 22
4.5.1　字体的定义 22
4.5.2　字体文件 23
4.5.3　字体的编译、装入和调用 24
4.6　大字体文件 24
4.6.1　大字体文件的特点 24
4.6.2　大字体文件的标题 24
4.6.3　举例 25
4.7　扩展大字体 25
4.7.1　“0”码形的定义 26
4.7.2　调用子形 26
4.7.3　举例 27

第 5 章　定义线型和填充图案 29

5.1　线型 29
5.1.1　概述 29
5.1.2　定义简单线型 29
5.1.3　定义复杂线型 30
5.2　定义图案 32

5.2.1 构造填充图案 32
5.2.2 图案文件 32
5.2.3 定义图案 32
5.2.4 举例 33

第 6 章 图形交换文件 34

6.1 概述 34
6.1.1 图形信息交换与 DXF 文件 34
6.1.2 输出一个 DXF 文件 34
6.1.3 读入一个 DXF 文件 35
6.2 DXF 文件的结构 36
6.2.1 DXF 文件的总体结构 36
6.2.2 组代码和跟随值 37
6.2.3 DXF 文件实例 39
6.3 从 DXF 文件中提取数据 44
6.4 用 C 语言编写构造 DXF 文件的接口程序 46

第 7 章 AutoLISP 程序设计语言 49

7.1 概述 49
7.2 AutoLISP 的数据类型 49
7.3 AutoLISP 变量 51
7.3.1 符号（SYMBOL） 51
7.3.2 变量的数据类型 52
7.3.3 为变量赋值 52
7.3.4 预定义的符号 53
7.3.5 显示变量的值 53
7.3.6 在交互方式下将变量的值传递给 AutoCAD 54
7.4 AutoLISP 表达式 54
7.4.1 前缀表示法 54
7.4.2 运算的优先级 54
7.4.3 AutoLISP 表达式的求值过程 54
7.5 AutoLISP 的数据存储结构 56
7.6 AutoLISP 函数 57
7.6.1 定义 AutoLISP 函数 57
7.6.2 调用 AutoLISP 函数 58
7.6.3 调用 AutoCAD 命令 58

7.6.4 定义 AutoCAD 命令 59
7.7 AutoLISP 程序文件 60
7.7.1 AutoLISP 文件的特点 60
7.7.2 程序中的注释 60
7.7.3 加载和卸载 Auto LISP 文件 61
7.8 应用举例 63
7.8.1 定义 AutoLISP 函数 63
7.8.2 定义绘制曲线的命令 64
7.8.3 定义参数化绘图的命令 66
7.8.4 定义修改图形数据库的命令 71
7.8.5 定义含有访问符号表功能的命令 73

第 8 章 Visual LISP 程序设计语言 74

8.1 Visual LISP 概述 74
8.1.1 Visual LISP 的特点 74
8.1.2 进入和退出 Visual LISP 74
8.1.3 Visual LISP 的用户界面 75
8.1.4 控制台操作 77
8.1.5 加载和运行 AutoLISP 程序 79
8.2 编辑源程序代码 80
8.2.1 文本编辑工具 81
8.2.2 文件操作 83
8.2.3 文本操作 84
8.2.4 设置代码格式 87
8.3 调试程序 97
8.3.1 概述 97
8.3.2 数据查看工具 97
8.3.3 利用断点调试程序 109
8.3.4 跟踪程序运行 116
8.4 构造应用程序 122
8.4.1 通过 vlisp_compile 函数编译单个文件 122
8.4.2 用应用程序生成器创建应用程序 125
8.4.3 修改应用程序生成过程中确定的选项 132
8.4.4 重新编译应用程序 135
8.4.5 更新应用程序 136
8.5 利用工程维护多文件应用程序 136
8.5.1 工程的作用 136

8.5.2 定义一个新工程 137
8.5.3 利用工程窗口管理工程文件 140
8.5.4 Visual LISP 对工程的操作 142
8.5.5 优化应用程序代码 145
8.6 在 AutoLISP 中使用 ActiveX 对象 149
8.6.1 AutoCAD 对象模型 150
8.6.2 用检验工具获取 AutoCAD 对象 151
8.6.3 在 Visual LISP 函数中使用 ActiveX 方法 154
8.6.4 转换对象引用 162
8.6.5 通过 AutoLISP 函数获取和更新图形对象的特性 164
8.6.6 用参数带回返回值的 ActiveX 方法 167
8.6.7 列出对象的特性和方法 168
8.6.8 使用集合对象 171
8.6.9 释放 VLA 对象和释放内存 174
8.6.10 处理 ActiveX 方法返回的错误 174
8.7 将反应器附着到 AutoCAD 图形上 176
8.7.1 基本概念 176
8.7.2 定义回调函数 179
8.7.3 创建反应器 180
8.7.4 在多文档中使用反应器 184
8.7.5 查询、修改和控制反应器的状态 185
8.7.6 临时反应器和永久反应器 189
8.7.7 反应器的使用规则 190

第 9 章 DIESEL 表达式 193

9.1 DIESEL 表达式的特点 193
9.2 定义用户的状态行 193
9.2.1 系统变量 MODEMACRO 的作用 193
9.2.2 定义用户的状态行 194
9.3 在菜单中使用 DIESEL 表达式 196
9.3.1 用 DIESEL 表达式控制菜单标签的显示状态 196
9.3.2 用 DIESEL 表达式改变菜单宏 197
9.3.3 用 DIESEL 表达式将系统变量的当前值写到图形中 198
9.4 在 AutoLISP 中使用 DIESEL 表达式 198
9.5 DIESEL 函数一览表 199
9.6 出错信息 203

第 10 章　定义菜单、工具栏 204

10.1　概述 204
10.1.1　菜单的作用 204
10.1.2　菜单文件的作用 204
10.1.3　菜单文件的类型 204
10.1.4　MENU 命令 205
10.1.5　基菜单和局部菜单 205
10.2　菜单文件结构 207
10.2.1　菜单文件的总体结构 207
10.2.2　菜单组 208
10.2.3　菜单段 208
10.2.4　子菜单 208
10.2.5　菜单项 209
10.3　菜单项语法 209
10.3.1　菜单项的组成 209
10.3.2　菜单宏语法 210
10.3.3　替换子菜单 213
10.3.4　条件宏表达式 214
10.3.5　在菜单宏中使用 AutoLISP 表达式 214
10.4　屏幕菜单 215
10.4.1　屏幕菜单的特点 215
10.4.2　屏幕子菜单 215
10.4.3　屏幕菜单段实例 216
10.4.4　MENUCTL 系统变量 218
10.5　下拉菜单和快捷菜单 218
10.5.1　下拉菜单 218
10.5.2　快捷菜单 220
10.5.3　控制菜单项标签的显示状态 222
10.5.4　插入和删除下拉菜单 225
10.6　按钮和辅助菜单 226
10.6.1　按钮和辅助菜单的特点 226
10.6.2　按钮和辅助菜单举例 226
10.7　图像菜单 227
10.7.1　图像菜单的特点 227
10.7.2　图像菜单的文法 227

10.7.3 菜单项标题的格式 228
10.7.4 显示图像菜单 228
10.7.5 建立幻灯片文件或幻灯片库 230
10.8 数字化仪菜单 231
10.8.1 数字化仪菜单的特点 231
10.8.2 数字化仪菜单举例 231
10.9 工具栏段 232
10.9.1 工具栏段的结构 232
10.9.2 应用实例 235
10.10 面向菜单的帮助 236
10.10.1 菜单帮助段的格式 236
10.10.2 建立菜单帮助段与下拉菜单段和工具栏的联系 236
10.10.3 举例 236
10.11 定义快捷键 237
10.11.1 定义快捷键的两种方法 237
10.11.2 用于定义快捷键的组合键 237
10.11.3 举例 238

第 11 章 对话框设计 240

11.1 概述 240
11.2 对话框控件 241
11.3 控件的属性 253
11.4 对话框设计原则与规范 260
11.4.1 一般原则 260
11.4.2 通用规范 261
11.4.3 预定义控件的使用规范 263
11.5 对话框控制语言 264
11.5.1 对话框文件 264
11.5.2 DCL 文件的结构 265
11.5.3 DCL 语法 266
11.6 对话框驱动程序 268
11.6.1 对话框驱动程序的流程图 268
11.6.2 对话框回调 270
11.6.3 对话框的特殊处理 271
11.6.4 对话框驱动函数 272
11.7 综合举例 276

需要全本请访问 www.ertongbook.com

附录 A　AutoLISP 函数 286
A.1　赋值、求值与禁止求值函数 286
A.2　数值计算函数 287
A.3　关系运算函数 290
A.4　逻辑运算函数 291
A.5　几何运算函数 293
A.6　判断函数 296
A.7　流程控制函数 299
A.8　表处理函数 301
A.9　字符串处理函数 303
A.10　数据类型转换函数 306
A.11　交互输入函数 308
A.12　有关文件操作的函数 313
A.13　打印输出函数 316
A.14　有关函数的函数 318
A.15　错误处理函数 322
A.16　选择集操作函数 323
A.17　与 AutoCAD 直接相关的函数 331
A.18　实体操作函数 335
A.19　符号表操作函数 339
A.20　扩展数据操作函数 341
A.21　内存管理函数 342
A.22　对话框操作函数 344
A.23　菜单控制函数 348
A.24　数字化仪控制函数 348
A.25　控制环境变量的函数 349
A.26　词典操作函数 349
A.27　应用程序操作函数 352
A.28　VisualLISP 扩展函数 354
A.29　有关 ActiveX 的函数 371
A.30　与反应器有关的函数 389

附录 B　AutoLISP 程序出错代码 409

附录 C　AutoLISP 程序出错信息 412

第 1 章 概述

1.1 定制与开发 AutoCAD 的目的

AutoCAD 是通用的图形系统，它适用于多种领域和行业。每个用户所要做的工作都是某一具体的任务，都有其行业、专业的特点，有专业的标准和规范。如果要建立适合于本专业的作图环境，就需要对 AutoCAD 进行定制工作，使其专业化、用户化。

AutoCAD 提供了线型、图案、字体及符号库，尽管这些库的内容比较丰富，但仍然不能覆盖某些用户的某些产品。如果要扩充这些库或建立自己的线型、图案、字体及符号库，就需要对其进行定制与开发。

由于专业或项目的需求，AutoCAD 的某些功能可能很少用到，另一些功能可能频繁地被使用。调整或修改现有的用户界面，将不常使用的一些功能从界面上卸载，将频繁使用命令安排在更方便、更容易调用的位置，既提高了系统的运行速度，又方便了用户对系统的操作。通过对 AutoCAD 的开发，可以定制令自己满意的用户界面，也可以定制全中文环境的用户界面。

产品或工程图是由数量繁多的直线、圆、圆弧等这样一些基本的图形元素组成的，也可以看成是由少量的较复杂的一些图形组成的，但通用的绘图软件缺少绘制这一层次图形对象的命令。例如，输入螺钉的型号、大径、长度、定位点和旋转角就能画出螺钉，输入齿轮的模数、齿数、齿宽、孔径等少量参数就能画出齿轮。通过对 AutoCAD 进行二次开发，就能够增加这样的功能，提高绘图和设计的质量和效率。

提高产品质量，缩短产品设计、制造周期，降低产品成本是企业一贯追求的目标。仅靠 CAD 是不够的，CAD/CAPP/CAM 数据集成是实现这一目标的重要途径。AutoCAD 只提供了与外部程序或其他系统交换图形信息的接口。要真正实现 CAD/CAPP/CAM 数据集成，就必须对 AutoCAD 进行二次开发。

1.2 定制与开发 AutoCAD 的途径

AutoCAD 是开放式的绘图系统，它为用户提供了两类定制和开发 AutoCAD 的工具。

(1) AutoCAD 的各种支持文件（主要是 ASCII 码的文本文件）的格式，如程序参数文件、线型文件、图案文件、形文件、脚本文件、图形交换文件和菜单文件等。用户可修改或创建这些文件来定义适合于本专业的需要。

(2) AutoCAD 的应用程序编程接口(API)及各种接口的开发指南。包括 AutoLISP、Visual LISP、DCL、DIESEL、Visual Basic、ObjectARX 等。

用户可以利用这些语言创建新的 AutoCAD 命令，使 AutoCAD 更好地满足用户自己的需要。用户可以编写自己的 AutoLISP 程序，也可以使用第三方程序和共享程序。

1.3 定制与开发 AutoCAD 的主要内容

(1) 通过合理地设置系统变量，建立符合本企业标准的初始绘图环境。

(2) 通过建立模板图，可获得初始图形的高起点。

(3) 借助于程序参数文件为经常使用的 AutoCAD 命令建立简短易记的别名。

(4) 借助于程序参数文件为常用的外部程序建立可在 AutoCAD 内部执行的命令名。

(5) 编写脚本文件，自动地完成成组的任务。利用脚本文件还可以编写外部程序与 AutoCAD 的接口程序。

(6) 通过线型文件定义符合企业标准和工作需要的线型。

(7) 利用图案文件建立符合企业标准的填充图案。

(8) 通过形文件定义符合企业标准和工作需要的符号和字体。

(9) 通过菜单文件建立适合自己工作需要的菜单、工具栏和快捷键。

(10) 用 DIESEL 语言定义满足自己工作要求的状态行。

(11) 利用 AutoLISP、Visual LISP 或 ObjectARX 提供的集成开发环境定义 AutoCAD 新命令，实现参数化绘图，扩充 AutoCAD 的现有功能。

(12) 利用图形交换文件编写外部程序与 AutoCAD 双向交换图形信息的接口程序。

(13) 用 DCL 语言定义对话框,得到方便实用的用户界面。

1.4 可自定义的 AutoCAD 支持的文件

表 1-1 列出了可编辑的 AutoCAD 支持的文件和文件类型，这些文件都是 ASCII 码的文本文件，用任何文本编辑器都可以创建或扩充这些文件。大多数文件在后面还有详细的介绍。名字为 acad 或 acadiso 的文件是 AutoCAD 系统定义的文件。若对这些文件进行修改或扩充必须事前制作备份。

表 1-1 AutoCAD 支持的文件和文件类型

acad.lin	标准 AutoCAD 线型库文件
acadiso.lin	标准 AutoCAD ISO 线型库文件
acad.mnl	标准 AutoCAD 菜单调用的 AutoLISP 程序文件
acad.mnu	标准 AutoCAD 菜单的模板源文件
acad.mns	标准 AutoCAD 菜单的源文件
acad.pat	标准 AutoCAD 填充图案库文件
acadiso.pat	标准 AutoCAD ISO 填充图案库文件
acad.pgp	AutoCAD 程序参数文件
acad.psf	AutoCAD PostScript 支持的文件
acad.rx	启动 AutoCAD 时自动加载的 ObjectARX 应用程序
acad.unt	AutoCAD 单位定义文件
asi.ini	数据库连接的转换映射文件
fontmap.ps	AutoCAD 字体位图文件
*.ahp	帮助文件，文件的扩展名为.ahp，相关的帮助索引文件的扩展名为.hdx

续表

*.dcl	用 DCL 语言编写的定义对话框的文件
*.lin	线型文件
*.lsp	AutoLISP 程序文件
*.mln	多线库文件
*.mnl	同名菜单文件调用的 AutoLISP 程序文件
*.mns	AutoCAD 生成的菜单源文件，用户可以对其修改或扩充
*.mnu	菜单源文件。包含定义 AutoCAD 菜单的命令字符串和宏语法
*.pat	定义填充图案文件
*.scr	脚本文件
*.shp	定义形/字体的源文件

1.5 AutoCAD 搜索文件的顺序

AutoCAD 的支持路径是一系列的目录，AutoCAD 在这些目录中搜索它的支持文件。AutoCAD 以下面的顺序进行搜索。

(1) 用户明确指定的路径。

(2) 当前工作目录。

(3) 当前正在操作的图形文件（DWG）目录。

(4) AutoCAD 支持的目录。

选择下拉菜单 Tools → Options...菜单项，将弹出 Option 对话框，通过该对话框的 Files 选项卡，用户可以设置或修改 AutoCAD 支持文件的搜索目录。初始的 AutoCAD 支持文件的搜索目录为：

① ...\AutoCAD2000\support；

② ...\AutoCAD2000\fonts；

③ ...\AutoCAD2000\help；

④ ...\AutoCAD2000\express；

⑤ ...\AutoCAD2000。

AutoCAD 按照搜索目录的顺序搜索文件。

(5) AutoCAD 可执行文件的目录。

在上述目录中可能有多个相同的被搜索的文件，AutoCAD 只要找到第一个满足要求的文件就不再向下搜索。

1.6 命令的验证过程

AutoCAD 首先对当前输入的命令进行有效性检查。当前的输入可能是：① 空响应（按空格键或回车键）；② AutoCAD 内部命令；③ AutoCAD 系统变量；④ acad.pgp 文件中定义的外部命令或内部命令的别名；⑤ 用 AutoLISP 语言定义的命令；⑥ 用 ObjectARX 应用程序定义的命令；⑦ 通过菜单选择的命令；⑧ 从脚本文件中或者由 AutoLISP 或 ObjectARX 应用程序调用的命令；⑨ 数据等其他任何字符串。

AutoCAD 按照以下的顺序和规则判断用户的当前输入及做出相应的对策。

(1) 是否是空响应。如果输入了空响应，AutoCAD 则调用上次执行的命令；如果此前尚未输入过命令，AutoCAD 则调用 HELP 命令；结束验证。

(2) 是否是 AutoCAD 内部命令，是则执行，否则继续验证。

(3) 是否是在程序参数文件(acad.pgp)中定义的外部命令，是则执行，否则继续验证。

(4) 是否是 AutoLISP 或 ObjectARX 应用程序定义的命令，是则执行，否则继续验证。

(5) 是否是 AutoCAD 系统变量名，是则执行，否则继续验证。

(6) 是否是程序参数文件中为内部命令定义的别名，是则执行。

(7) 如果以上哪一个条件都不满足，则验证终止，并显示 Unknown command "×××××". Press F1 for help.（不知道“×××××”命令，按 F1 键获得帮助）。

第 2 章　程序参数文件

2.1　概述

程序参数文件的文件名固定为 acad.pgp，是 ASCII 码的文本文件。该文件有两个主要作用，第一个作用是定义外部命令，即在 AutoCAD 的命令状态“ Command： ”下，执行外部命令或程序。第二个作用是为 AutoCAD 内部命令定义简短易记的别名。

该文件分为两部分：第一部分定义外部命令，第二部分定义 AutoCAD 内部命令的别名。文件中分号“ ; ”至行尾为注释部分。

2.2　定义外部命令

2.2.1　定义外部命令的格式

定义外部命令的工作就是建立在 AutoCAD 内部执行外部命令或程序的命令对照表，每一行定义一个外部命令，由逗号“ , ”隔开的以下 5 个字段组成：

command,[executable],flags[,[*]prompt[,return_code]]

(1) command：在 AutoCAD 内部称呼（调用）的外部命令名。即在命令提示“ Command： ”时输入的命令名。如果该名称是 AutoCAD 内部命令名，则该命令将被忽略。该名称不区分大小写。

(2) executable：外部命令或程序实际的名字。可包含开关或参数。该字符串是否区分大小写由运行的应用程序决定。

(3) flags：标志。该标志按位设置，用十进制表示，从右至左每个二进位的作用见表 2-1。

表 2-1　flags 每个二进位的作用

状态位 / 位	被设置（等于 1）	未被设置（等于 0）
第一位	启动外部程序并不等待其结束	启动外部程序并等待其结束
第二位	最小化运行外部程序	
第三位	隐藏（后台）运行外部程序	
第四位	可以将参数字符串括在引号中	

说明：① flags 的值用十进制表示。② 可同时设置多个位。例如，隐藏（后台）运行应用程序且不等待其结束，那么，第一、三位置 1，用十进制表示为 5。③ 第二位和第三位不能同时设置，如果同时设置了这两位（即 flags 的值为 6），那么只有第二位有效。④ 应尽量避免只设置第二位或第三位而不同时设置第一位，因为即使最小化运行应用程序或隐藏（后台）运行应用程序，AutoCAD 也只能等待其运行结束，而在此期间，AutoCAD 因被屏蔽而不能使用。⑤ 第四位如果被设置，像 del 这类命令就可以正确处理包含空格的文件名。但是，这将导致不能一次向这些命令传递用空格分隔的多个文件名，如果想避免这一现象，就不要设置第四位。

需要全本请访问 www.ertongbook.com