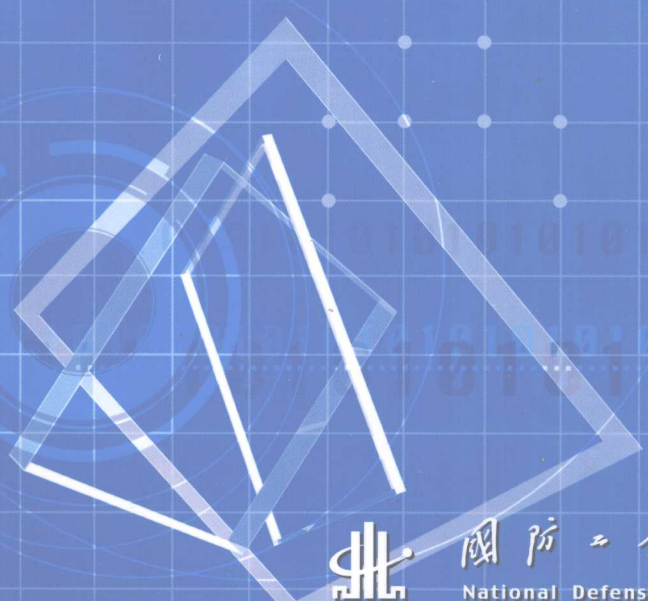
随书附光盘一张

UG/Open API、MFC和COM开发实例精解

黄勇 著



国防工业出版社
National Defense Industry Press

UG/Open API、MFC 和 COM 开发实例精解

黄勇 著

国防工业出版社
·北京·

图书在版编目(CIP)数据

UG/Open API、MFC 和 COM 开发实例精解/黄勇著.
—北京:国防工业出版社,2009.2
ISBN 978-7-118-06125-3

Ⅰ.U... Ⅱ.黄... Ⅲ.计算机辅助设计－应用软件,UG NX5.0 Ⅳ.TP391.72

中国版本图书馆 CIP 数据核字(2008)第 207659 号

※

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号 邮政编码 100048)
涿中印刷厂印刷
新华书店经售

*

开本 787×1092 1/16 **印张** 24¾ **字数** 572 千字
2009 年 2 月第 1 版第 1 次印刷 **印数** 1—4000 册 **定价** 45.00 元(含光盘)

(本书如有印装错误,我社负责调换)

国防书店:(010)68428422 发行邮购:(010)68414474
发行传真:(010)68411535 发行业务:(010)68472764

前 言

UG(Unigraphics)是美国EDS公司推出的当今世界上最先进的CAD/CAM/CAE高端软件平台之一,广泛应用于航空、航天、机械、汽车、船舶、模具和家用电器等领域。许多世界著名公司均选用UG作为企业计算机辅助设计、制造和分析的平台,如美国通用汽车公司、波音飞机公司、贝尔直升机公司、英国宇航公司、惠普发动机公司等。自从1990年UG进入中国市场以来,在我国得到了越来越广泛的应用,已成为我国主要使用的高端CAD/CAM/CAE软件之一。

随着软件应用面的扩大,以及各个专业领域对UG软件应用的要求不同,越来越多的用户希望在UG软件平台的基础上通过二次开发来实现专业化、智能化和高效化的定制,从而提高企业CAD/CAM/CAE的应用水平,以增强企业的竞争力。为满足用户的特殊需要,EDS公司为UG提供了功能强大的二次开发接口UG/Open,随UG一起发布,所开发的应用程序和UG可以很好地融合。为了使广大用户更好地掌握UG/Open开发工具,著者总结了教学科研和企业对UG二次开发应用的经验,并结合使用UG、MFC和COM进行二次开发的心得,编著了本书。

全书共分8章,第1章介绍了UG/Open API开发的基础知识;第2章介绍了UG/Open API、MFC和COM开发原理,并以UG平台上基于MFC的两层C/S结构数据库的访问和基于COM组件模型特征的获取为例,阐述了利用UG/Open API、MFC和COM进行UG二次开发的过程;第3章通过内齿圈参数化设计实例阐述了在UG平台上进行参数化设计的方法;第4章阐述了利用COM组件进行数据库访问的过程;第5章通过UG平台上基于MFC变位直齿轮参数化设计为例,阐述了在UG平台上利用UG/Open API和MFC进行基于数据库为支撑的参数化设计的方法;第6章阐述了在UG平台上利用COM组件进行三层C/S结构数据库开发的过程;第7章阐述了在UG平台上利用UG/Open API和MFC进行基于数据库的模型文件管理的开发;第8章以花键轴为例,阐述了在UG平台上基于COM组件参数化设计的方法。

本书深入阐述了UG/Open API二次开发工具,UG开发环境的设置,菜单及工具条的编写,UG对话框的制作,UG/Open API、MFC和COM开发原理,数据库技术在UG二次开发中的应用,并综合前述开发方法及工具,以基于UG/Open API的UG特

征输出、UG平台上基于MFC的两层C/S数据库访问、基于COM组件模型特征获取的开发、内齿圈参数化设计的开发、基于COM组件的数据库信息获取的开发、UG平台上基于MFC变位直齿轮参数化设计的开发、UG平台上基于COM组件的三层数据库系统的开发、UG平台上基于MFC的模型文件管理系统的开发、基于COM组件的花键轴参数化设计的开发为例，阐述了UG/Open API、MFC和COM在UG二次开发中的具体应用，并公开了源代码，使读者能够快速掌握利用UG/Open API、MFC和COM进行UG二次开发的精髓，提高二次开发的能力，以满足广大用户的实际开发需要。

作者著本书的目的是为UG二次开发者提供一本针对性和实用性方法和技术手段以解决产品开发过程中的实际问题。作者希望通过阅读本书后，能对使用UG/Open API、MFC和COM进行UG二次开发有一个完整的认识，通过书中实例快速掌握UG/Open API、MFC和COM三者相结合的开发方法，并能应用于产品开发的实践。

本书面向产品设计人员、有志于从事UG二次开发的程序员和相关专业教师，同时，也可作为高校学生、研究生实践性教学的参考用书。

在本书的编著过程中，张博林同志做了大量收集和整理资料的工作，华东交通大学和出版单位也给予了大力支持和帮助，在此一并表示感谢。

由于UG/Open API、MFC和COM的内容非常丰富，加之编著时间仓促，不当之处还望各位读者提出宝贵意见。

编著者

2008年12月

目　录

第 1 章　UG/Open API 开发基础

1.1　UG/Open 开发工具

UG是CAD、CAM和CAE一体化的软件系统，可应用于产品从概念设计到实际产品的开发全过程，包括产品的概念设计、建模、分析和加工。该软件具有实体建模模块、特征建模模块、曲线曲面建模模块、工程制图模块、装配模块、分析模块、加工模块、知识工程模块和二次开发模块，不仅可以完成建模、装配、工程出图、数控加工等功能，还可以对建立的模型进行运动学、动力学仿真及有限元分析等强大的分析功能。它所提供的二次开发语言简单易学，功能强大，便于用户开发专用CAD系统，可以实现单凭交互方式操作UG难以实现的功能，如复杂模型的参数化建模，装配路径规划，UG平台上的PDM、CAPP等功能。因此，通过在通用CAD软件上进行二次开发可以明显提高设计效率，为企业在市场上的竞争提供有力的平台。

由于计算机技术、虚拟现实技术及现代设计理论与方法的迅速发展，CAD 技术已经从过去最简单的二维绘图工具发展成了一个智能化、网络化和高度集成化的三维CAD软件平台。很多企业在引入CAD/CAM/CAE软件后，发现通用的CAD/CAM/CAE软件的功能虽然解决了他们的大部分实际需求，在一定程度上提高了产品设计、制造及管理的效率，但是很多专业的、更为具体的问题，如符合本企业设计用的产品数据管理(PDM)、编制产品工艺用的计算机辅助工艺设计(CAPP)、产品虚拟装配的路径规划、异地产品协同设计以及本企业复杂零件的参数化设计等，单靠操作UG是很难实现的，以致于CAD/CAM/CAE软件的应用水平不高，仅仅停留在操作层面，没有充分挖掘软件平台的潜力，浪费了很多人力和物力。目前大部分企业已经意识到了开发满足企业实际应用软件的重要性，并且很多企业都有成功实施的经验，把特殊的、专业的知识与通用的软件集成为一个高效的、满足企业实际应用的系统平台，为企业在市场上的竞争提供有力的保障。

UG/Open 作为 UG 平台上提供的二次开发语言是为满足用户特殊需要而随 UG 一起发布的。它为 UG 软件的二次开发提供了许多函数和工具集，便于用户进行二次开发。利用该模块可以对 UG 系统进行用户化定制和开发，实现特定的功能。UG/Open 包括以下几个部分：UG/Open API 为 UG 软件提供直接的编程接口；UG/Open Grip 是 UG 内部开发语言，具有通俗、易懂的特点，是 UG 二次开发早期的主要语言，用户利用它可以生成 NC 自动化或自动建模等特殊应用；UG/Open MenuScript 对 UG 软件操作的菜单、工具条进行用户化开发；UG/Open UIStyle 是一个可视化编辑器，用户可以为 UG/Open 应用程序开发友好的交互界面。

① UG/Open MenuScript是UG提供定制菜单的专用模块，可以生成自己的菜单，替换

UG的原有菜单，也可以实现对UG某个菜单的编辑并生成自己的菜单。MenuScript支持UG主菜单和快速弹出式下拉菜单的修改，通过它可以改变UG主菜单和快速弹出式下拉菜单，可以改变UG菜单的布局、添加新的菜单项以执行用户二次开发程序。应用MenuScript编程，有两种方法可以实现菜单的用户化：一种方法是重新生成，并替换UG标准菜单；另一种方法是对标准UG菜单进行编辑。

② UG/Open UIStyler是开发UG对话框的可视化工具。使用这个工具最大的优点是可以避免复杂的图形用户接口编程。其设计对话框的方式与VC很相似，即利用对话框中基本控件的组合生成不同的对话框，对话框中所有的控件设计是可见即可得的。

③ UG/Open Grip 语言用来创建满足需求的专用软件，与 UG 系统集成，具有通俗、易懂的特点。利用 Grip 程序，可以完成与 UG 的各种交互操作。如调用一些曲线、曲面及实体生成语句，创建几何体和制图实体；控制 UG 系统参数，实现文件管理功能；存取 UG 数据库，提取几何体的数据和属性；编辑修改已存在的几何体参数等。Grip 语言与一般的通用语言一样，有完整的语法规则、程序结构、内部函数，并可以与其他通用语言程序相互调用。Grip 程序同样需要经过编译、链接后生成可执行程序才能运行。

④ UG/Open API 是 UG 与外部应用程序之间的接口，它提供了一系列函数的集合。通过 UG/Open API 的编程，用户几乎能实现所有的 UG 功能。开发者可以通过 C 语言编程调用这些函数，从而达到实现用户化的需要。UG/Open API 主要用于用户化定制 CAD 环境、开发在 UG 软件平台上的专用软件、开发 UG 软件与其他 CAD 软件的接口。目前，商品化的 CAD 软件很多，如 UG、CATIA、Pro/E、SolidWorks、AutoCAD 等。这些软件都有各自的数据结构，如果需要将它们的信息相互利用，就必须开发它们之间的数据转换接口。

1.2 UG/Open API 基础知识

1.2.1 UG/Open API 的开发模式

根据程序运行环境的不同，UG/Open API 程序可分为两种模式。

1) 外部(External)程序模式

UG/Open API 程序的运行与 UG 的环境无关，只能在 UG 环境外运行 UG/Open API 程序，在操作系统下单独运行。它作为操作系统中子进程存在，调用灵活，但不能与 UG 图形界面进行交互。因此，运行的结果通常不能显示在 UG 图形界面中，所以应用较少，通常用于打印机、出图和数据管理等。

在调用访问 UG 格式数据的函数以前必须先打开 UG 的部件(part)文件。

External 程序的一般格式如下：

```
#include <uf.h>
int main(int argc,char **argv)
{
    /*定义变量，如：int i;char s;*/
    UF_initialize();
    /*应用程序主体，如：uc1601("hi",1);*/
    UF_terminate();
```

}

2) 内部(Internal)程序模式

UG/Open API 程序的运行与 UG 的环境有关，只能在 UG 中运行。它是经过编译、连接后得到的 DLL 文件，程序代码小、连接速度快，运行结果在 UG 界面的图形窗口中可见，入口函数主要是 ufusr 或 ufsta。运行在 UG 内部的 API 程序通过动态链接成为 UG 的一部分，并可以与用户进行交互，实现与 UG 的无缝集成。

Internal 程序的一般格式如下：

```
#include <uf.h>
void ufusr(char *param,int *retcod,int parm_len)
{
   /*定义变量，如：int i;char s;*/
   UF_initialize();
   /*应用程序主体，如：uc1601("hi",1);*/
   UF_terminate();
}
```

1.2.2 UG/Open API 语法

1. 数据类型

任何一种编程语言或者编程接口都有自己的数据类型，UG/Open API 也不例外。数据类型表明了这种数据在内存中需要多大的空间。UG/Open API 编程接口是 C 语言的语法格式，因而，它支持 C 语言的标准数据类型，除此以外，UG/Open API 大量使用了类型定义，如 structures(结构体)、enums(枚举)、unions(共用体)等。UG/Open API 数据结构的命名规定是：_t——数据类型；_s——结构体类型；_u_t——共用体类型；_t_p——数据类型指针；_u_p_t——共用体类型指针。

如：

```
union UF_STYLER_value_u
{
   char  *string;
   char  **strings;
   int   integer;
   int   *integers;
   double  real;
   double  *reals;
   UF_UI_selection_p_t  selection;
   UF_STYLER_notification_p_t  notify;
   UF_UI_attachment_t  attach;
   UF_UI_option_toggle_t  option_toggle;
};
typedef union UF_STYLER_value_u UF_STYLER_value_t;
```

上面的数据类型 UF_STYLER_value_u 是定义的共用体数据类型，通过 typedef union 关键字把 UF_STYLER_value_u 数据类型定义成 UF_STYLER_value_t 别名，即通过 UF_STYLER_value_u 和 UF_STYLER_value_t 定义的数据类型都表示同一种类型。

UF_STYLER_value_t 数据类型主要用在 UF_STYLER_item_value_type_s 数据类型中，作为它的成员，UF_STYLER_item_value_type_s 表示定义的是结构体类型的数据，定义如下：

```
struct UF_STYLER_item_value_type_s
{
        int  reason;
        const char  *item_id;
        int  subitem_index;
        int  count;
        int  item_attr;
        int  indicator;
        UF_STYLER_value_t value;
};
```

此外，在 UG/Open API 中，用来识别对象的数据类型是 tag_t，是对象句柄，实际上，tag_t 是无符号整型数据类型，在 uf_defs.h 中定义如下：

```
typedef unsigned int tag_t,*tag_p_t;
```

大多数情况下，数据类型都在相应的头文件中有说明，因此，对于在程序中使用到的数据类型，也应在程序开头将相应的头文件用#include 包含进来。下面是获取当前显示模型句柄 tag_t，并将其关闭的代码：

```
tag_t tModel;
tModel =UF_PART_ask_display_part();
UF_PART_close( tModel , 1 , 1 );
```

2. 函数

1) 函数名称

在 UG/Open API 中，函数名称的命名有两种形式：UF_<模块或应用字母的缩写>_<动作:名词+动词>；uc<数字>或 uf<数字>。在这两种方式中，第一种方式是新版本的 UG/Open API 中函数的命名方式，并且将代替第二种方式；第二种方式是老版本的命名方式。由于第二种命名方式简单、函数参数少，因而，仍有不少函数被使用，如 uc1601 (“hi” , 1);。

(1) UF_<模块或应用字母的缩写>_<动作:名词+动词>

```
UF_PART_new();
UF_PART_open();
UF_PART_save();
UF_PART_save_as();
```

(2) uc<数字>或 uf<数字>

```
uc1601();
uc1605();
```

2) 函数参数

在 UG/Open API 中，大部分参数是用 C 语言编写的，函数的定义遵守以下形式：

<返回数据类型><函数名>(参数列表)

其中，参数列表中的参数分为 Input、Output、Output Free 三种，其中，Input 表示输入参数，Output 表示输出参数，Output Free 表示输出参数，但是，此输出参数在使用完毕后，必须释放占用的内存空间。

如：

(1) UF_PART_new(const char * part_name , int units , tag_t * part);

函数功能：新建模型。其中，part_name 是输入参数，表示新建模型名称的字符指针；units 是输入参数，表示模型单位，当 units=1 时，模型单位是 Metric，当 units=0 时，模型单位是 English；part 是输出参数，表示新建模型句柄的指针。下面是此函数的用法：

```
tag_t part;
const char part_name[ ]="D:\\model_1.prt";
UF_PART_new( part_name , 1 , &part );
```

(2) UF_PART_open(const char * part_name, tag_t * part, UF_PART_load_status_t * error_status);

函数功能：打开指定模型名称的模型。其中，part_name 是输入参数，表示打开模型名称的字符指针；part 是输出参数，表示返回打开模型句柄的指针；error_status 是输出参数，但是使用完毕后，必须释放占用的内存空间，表示函数执行失败时显示错误信息。下面是此函数的用法：

```
tag_t part;
const char part_name[ ]= "D:\\model_1.prt";
UF_PART_load_status_t error_status;
UF_PART_open( part_name , &part , &error_status );
UF_PART_free_load_status( &error_status );
```

(3) UF_PART_save(void);

函数功能：保存当前处于活动状态的模型。函数没有参数列表，用法简单，如：

```
UF_PART_save();
```

(4) UF_PART_save_as(const char * new_part_name);

函数功能：将当前处于活动状态的模型另存为指定的模型名称。其中，new_part_name 是输入参数，表示新命名的模型名称。下面是此函数的用法：

```
const char new_part_name[ ]= "D:\\model_2.prt";
UF_PART_save_as( new_part_name );
```

3) 函数的使用

利用 UG/Open API 进行 UG 二次开发，当使用到函数时，必须先用 UF_initialize( ) 函数进行初始化，获得函数使用许可；当函数使用完毕后，必须用 UF_terminate( )释放掉对函数的使用许可。下面是使用 UG/Open API 函数时经常使用的框架：

```
if ( UF_initialize() != 0)
   return;
//执行满足功能需求的 UG/Open API 函数;
    UF_terminate ();
```

1.2.3 UG/Open API 表达式

在 UG 中进行参数化建模时，通过改变模型的尺寸可以使模型发生相应的变化。模型尺寸的变化实际上是约束模型的尺寸表达式发生了改变，这在 UG 表达式编辑器中可以看到。

同样，利用 UG/Open API 编程也可以创建表达式、改变表达式、更新模型，达到参数化建模的目的。

表达式名称必须以字母开头，后面可以是字母、数字及下划线等，区分大小写。一个模型中的表达式名必须是唯一的，其一般形式为“表达式名=值”，值可以是数字也可以是条件等式。

在 UG/Open API 中，关于表达式的各种操作一般在头文件 uf_modl_expressions.h 中，其中，常用的函数为：

1. 新建表达式

```
int UF_MODL_create_exp( char * expr_str );
```

其中，expr_str 表示新建表达式的字符指针，如：

```
char expr_str[ ]="p1=10";
UF_MODL_create_exp( expr_str );
```

2. 删除表达式

```
int UF_MODL_delete_exp( char * exp_name );
```

其中，exp_name 表示被删除的表达式名称的字符指针，如：

```
char expr_str[ ]="p1=10";
UF_MODL_create_exp( expr_str );
char exp_name[ ]="p1";
UF_MODL_delete_exp( exp_name );
```

3. 计算表达式的值

```
int UF_MODL_eval_exp( char * exp_name , double * exp_value );
```

其中，exp_name 表示被计算的表达式名称的字符指针，exp_value 是指向表达式值的双精度实数指针，如：

```
char expr_str[ ]="p1=10";
UF_MODL_create_exp( expr_str );
char exp_name[ ]="p1";
double exp_value;
UF_MODL_eval_exp( exp_name , &exp_value );
```

4. 编辑表达式

```
int UF_MODL_edit_exp( char * expr_str );
```

其中，expr_str 表示被编辑的新表达式的字符指针，如：

```
char expr_str[ ]="p1=10";
UF_MODL_create_exp( expr_str );
char expr_str1[ ]="p1=60";
UF_MODL_edit_exp( expr_str1 );
```

5. 查询表达式

```
int UF_MODL_ask_exp( char * exp_name , char exp_defn[133] );
```

其中，exp_name 表示被查询表达式名称的字符指针，exp_defn 表示查询的表达式结果，此表达式是完整的表达式，如：

```
char expr_str[ ]="p1=10";
UF_MODL_create_exp( expr_str );
char exp_name[ ]="p1";
```