

Personal Designer
User Programming Language
(UPL)

Revision 6.0

User Reference Guide

Chapter 1

Introduction

Introduction

- UPL Application.....1-3
- What's in This User Reference Guide.....1-4
- How to Use This Manual.....1-5
- Notation Conventions.....1-6
- Sample Page..... 1-8
- UPL Statement, Procedure, Function..... 1-8
 - Type..... 1-8
 - Purpose..... 1-8
 - Syntax..... 1-8
 - Remarks..... 1-8
 - Example..... 1-8
- Creating a UPL Program.....1-9
- Running a UPL Program.....1-11

Introduction

User Programming Language (UPL), is the macro programming language designed for use with the following Computervision Personal CAD/CAM products:

Personal Designer

microDRAFT

Personal Machinist

Personal Machinist Universal

UPL Applications

UPL lets you write programs that enhance the performance of these products. The following are examples of what you can do with UPL.

Automate Design and Drafting Procedures. UPL programs perform established procedures by automating several commands. For example, you can write a UPL program to lay the groundwork for a design session. The program can call up your company's standard drawing border and insert the existing parts common to all company projects, such as a nameplate or assembly for a mechanical part.

Design Interactive Programs for Families of Parts. Interactive UPL programs simplify the design of parts that share characteristics or belong to the same family. For example, when creating gears, you must answer basic questions such as how many teeth the gear will have and what the gear's pitch is. A UPL program can ask these questions, build a gear design based on your responses, and interact directly with the drawing database.

Create Customized Commands. With UPL, you can write programs that behave in the same way as a Personal Designer command. You supply verbs, nouns, modifiers, and digitize marks and the program does the work.

Design Tutorials. UPL can be used to design interactive tutorials that familiarize new users with Personal Designer and with your company's practices. UPL tutorials let new users learn at their own pace. UPL tutorials let you train large numbers of users at the same time, since the tutorials can be easily distributed on diskette,

Introduction

What's in This User Reference Guide

The *Personal Designer User Programming Language (UPL) Revision 6.0 UserReference Guide* contains detailed information on all UPL Statements, Intrinsic, and Program Components. In this book, the term *intrinsic* refers to all procedures and functions that are already defined in the code and can be called from your program.

This manual is not a programming tutorial, and is intended for individuals with some programming background.

This manual has four sections and several appendices. A brief description of each section and appendix follows.

Section 1, Introduction, outlines the scope and contents of the book, identifies related material, presents the notation conventions used in the book, and describes the installation and startup process.

Section 2, Program Structure, presents a program structure overview. This section contains descriptions and definitions of the components of a UPL program.

Section 3, Functional Listing of Statements and Intrinsic, groups UPL statements and intrinsic procedures and functions by the tasks they perform.

Section 4, Statements and Intrinsic, lists alphabetically, all of the UPL statements and intrinsic. Consult this section for detailed information on statement and intrinsic capabilities, procedures, and format.

Appendix A, Reserved Words, contains words that have special meaning in UPL. These words cannot be used as names of variables, constants, procedures, or functions in a UPL program.

Appendix B, System Variables, contains a description of the special variables that are predeclared in all UPL programs. These variables hold information that is passed between Personal Designer and the UPL program.

Appendix C, Compiler Limits, lists the limits placed on a program by the UPL compiler.

Appendix D, Error Messages, lists runtime error messages that the UPL compiler outputs.

Introduction

Appendix E, Internal Data Storage Format, defines the format in which UPL stores data. The format for each data type and aggregate type is given. Also included is information on file formats and the Transformation Matrix.

Appendix F, ASCII Character Set, lists the standard character codes used by PCs to represent alphanumeric information.

Appendix G, Direct Database Access, gives detailed information on direct access to the Personal Designer database. This is a very advanced topic and is included for the advanced user. Improper use of Direct Database Access routines can damage your part database.

Appendix H, Writing Personal Designer Commands Using UPL, demonstrates the use of intrinsics that give you access to the Personal Designer user interface: the Verb-Noun processor, the Modifier processor, and the getdata processor. This lets you write UPL programs that have the same command syntax as Personal Designer commands.

Appendix I, System Variables, contains sample UPL programs.

How to Use This Manual

This manual contains detailed information on all UPL elements and concepts. It is formatted so that you can quickly locate information. Statements and intrinsics are listed in alphabetical order.

Each statement or intrinsic description lists its purpose, type, syntax, remarks and an example where appropriate. Intrinsics also have a list of parameters.

If you are a new UPL user, first read Sections 1, 2, and 3. These sections introduce the UPL program elements and concepts that you will need to know in order to use UPL. Then read some of the programs found in Appendix I. You should be ready to start writing UPL programs of your own.

For users who are already familiar with UPL, use this manual for detailed information. Section 3 lists statements and intrinsics grouped by the tasks they accomplish. Statements and intrinsics are listed alphabetically in Section 4 giving detailed information. Appendix A lists the reserved words in UPL.

Introduction

Notation

Conventions

bold type

Indicates information that must be spelled as shown.

italics

Indicates a mandatory replacement that you type in.

regular type

Indicates an optional replacement.

1

Vertical bars indicate to choose the items on either side.

...

Ellipses indicate that an entry may be repeated as many times as is needed or desired.

Enter other punctuation such as commas, colons, and parentheses as shown.

Introduction

When the word string is used in this manual, it refers to a UPL character string. Do not confuse these with the string entities in Personal Designer.

The table below lists the abbreviations used with statements and intrinsics.

var	name of variable
array	name of array
flvar	name of file variable
ivar	name of 16-bit integer variable
i4var	name of long-integer (32 bit) variable
iarray	name of integer array
i4array	name of long-integer array
cvar	name of coordinate variable
carray	name of coordinate array variable
rvar	name of real variable
rarray	name of real array variable
bvar	name of boolean variable
barray	name of boolean array variable
svar	name of character string variable
sarray	name of character string array variable
const	literal or named constant
iconst	literal or named integer constant
rconst	literal or named real constant
cconst	literal or named coordinate constant
bconst	literal or named boolean constant
sconst	literal or named character string

constant

expr	expression of any type
iexpr	integer expression
i4expr	long integer expression
rexpr	real expression
bexpr	boolean expression
cexpr	coordinate expression

Introduction

Sample Page

Below is a sample page used in Chapter 4, "Statements and Intrinsic":

UPL Statement, Procedure, Function

Type

Identifies command as a statement, procedure, or function and names the class it belongs to.

Purpose

Describes what the statement or intrinsic does.

Syntax

Tells you how to input the statement or intrinsic. The syntax also shows you how the statement or intrinsic appears in the program.

Remarks

Lists and defines the modifiers or parameters for each of the statements or intrinsics.

Example

Provides an example of the statement or intrinsic.

Introduction

Creating a UPL Program

When you are ready to enter the programming code into a file, use a text editor. Enter your program into a file with the extension UPL.

The source is the program you write, using an ASCII editor of your choice. The source file becomes the input for the next step in the programming process.

The UPL compiler acts as a language translator, converting the source program to a form the computer can understand and execute. As it translates the statements, the compiler uncovers errors.

On DOS systems, compile UPL programs using the following syntax:

```
upl      source,code,list /option.../option...
```

On UNIX systems, compile UPL programs using the following syntax:

```
upl      source,code,list -option...-option...
```

Source, code, and list can be any legal path and file name. There must be at least one space between the **upl** and *source*. There must be no spaces between *source, code, and list*. There must be at least one space between *list* and *option*.

Source is the input source text file name. '.UPL' is assumed if no file name extension is given.

Code is the binary code file used as input to Personal Designer. You can give *code* a different name from *source* if you want the executable UPL program to have a different name. '.UCD' is assumed if no file name extension is given. If *code* is specified, it must be separated from *source* by a comma.

List specifies the name of the error listing file with a different file name than the source file name. '.LST' is assumed if no file name extension is given. A comma must separate *code* from *list*. If *code* is omitted but *list* is not, separate *source* and *list* with two commas. You must specify the LST option with this choice.

Option has three possible choices. Each option must be preceded by a hyphen (-) or a slash (/).

Introduction

- lst** This option places the error messages in a list file instead of displaying them on the screen. Printing the list file gives you a hard copy of error messages to use in debugging.
- stat** This option displays the compilation statistics during compilation. You must also use the **lst** option when you use **stat**.
- intel** This option always generates code in Intel format (default).
- spare** Always generates code in SPARC format.
- native** Generates code that is native to the platform on which the compiler is running.

The compiler lists errors along with the location and type of each error to make correction easy. To correct the errors or debug the program, edit the source file with a text editor and then recompile the program. When no errors are found, the compiler outputs a binary code file. After you have obtained a binary code file of your program, you are ready to run your program.

The table below lists a few compiling commands and shows the filenames that result from these commands.

Command	Source	Code	Listing
upl gear	gear.upl	gear.ucd	
upl gear,xgear	gear.upl	xgear.ucd	
upl gear,xgear /lst /stat	gear.upl	xgear.ucd	gear.lst
upl gear,xgear,list -lst	gear.upi	xgear.ucd	list.lst
upl gear,,list -lst	gear.upl	gear.ucd	list.lst

Table 1-1. UPL Compiling Commands

Introduction

Running a UPL Program

UPL programs must be run from within Personal Designer. When you run the program, Personal Designer executes the code generated by the compiler. Using Personal Designer's RUN command is the most common method of running a UPL program.

Enter RUN followed by the name of your program. The system looks for a file with your program name and a UCD extension and executes this file as a UPL program.

You can program Personal Designer to run a UPL program upon startup. Copy your.UCD program into INIT.UCD. Make sure that INIT.UCD is in either the current directory or the Personal Designer startup directory, \PD6. When your system starts up, it will run this UPL program first.

You can also edit a tablet key or on-screen icon to carry out the RUN command. See the Personal Designer documentation for more information.

To make a UPL program behave like a Personal Designer command, you can modify Personal Designer's verb-noun processor table. See Appendix H, Writing Personal Designer Commands Using UPL, for more information on this method.