Personal Designer

User Programming Language (UPL)

Revision 6.0

User Reference Guide

Appendix E

Internal Data Format

Internal Data Format

This appendix contains the format of data used by UPL for:

Variables
Files: text and binary
System date and time
Transformation matrix

Variables

A variable's data is stored in memory according to its data type. The table below lists for each data type, the amount of memory occupied by a variable, and its format.

Integer	2 bytes	<Low byte> Bit 15-8	<High byte> Bit 7-0
Integer4	4 bytes		
Real	4 bytes	<Sign bit> Bit 31 <Significant> Bits 22-0	<Exponent> Bits 30-23
Boolean	2 bytes	Same as Integer (0 = False, 1 = True)	
String	4 bytes + declared maximum length		
		<Max length> Integer <String data> 'Max length' bytes	<Current length> Integer
Coord	12 bytes	3 Real numbers (X, Y, Z)	

Internal Data Format

File	22 bytes	At offset 0 – Operating system file handle # 2 – File position byte pointer 6 – Number of bytes to read/write 8 – Error number 10 – File record length 12 – Format type (0=Binary, 1=Text) 14 – Open tlag 16 – End – of – line flag 18 – End – of – file flag 20 – Written to tlag

Array variables require an additional two bytes which contain the number of bytes between the start of the first array element and the start of the last array element. This number can be treated as an integer and is used for array bounds checking. Following this comes the array elements in sequence. Each array element will use the same format as a variable of that data type (see above). Files cannot be stored in an array.

<byte offset to beginning of last element (2 bytes)>
<array element l >
<array element 2>
<array element 3>

<array element n>

String arrays vary slightly from this in that the maximum length is not repeated for each element. The format for a string array is:

<byte offset to beginning of last element (2 bytes)>
<Maximum declared length of all strings in array (2 bytes)>
<element 1: Current length (2 bytes)
Characters (one byte per character)>
<element 2: Current length (2 bytes)
Characters (one byte per character)>
<element 3: Current length (2 bytes)
Characters (one byte per character))

<element n: Current length (2 bytes)
Characters (one byte per character)>

File Formats

Data is stored in binary files using the format for variables, listed above. Note that this is the format used by UPL programs to use the Read and Write statements. Binary files written by other programs may use a different format.

Data is stored in text files as characters. When reading text files, strings of characters are scanned from the file and then converted to the appropriate data type for the specified variables. The following data rules are fo]lowed for scanning the variables:

For character strings, start at the current file pointer and read the shorter of:

1) the specified field width
2) the declared maximum string length.
3) 400 characters

If the specified field width is greater than the declared length of the string, the file pointer will be updated to the field width but the string will only be filled to its maximum declared size. If no field width is given and a <CR> is encountered before the maximum length is reached, it will stop scanning.

Internal Data Format

For integers and reals, start at the current file pointer and scan for the following items:

1) any leading blanks and/or tabs and ignore them
2) a+ or –
3) a series of digits in range of 0 – 9
4) a
5) more digits in range of 0 – 9
6) aneorE
7) a+or –
81 another series of digits in range of 0 – 9

Note that only 3) is mandatory and a given field width will stop the scanning. If the specified variable is a real variable and you scan an integer, it will be converted. If the specified variable is an integer and you scan a real, it will be converted by truncation. All integer values must lie in the range of
– 32,768 to 32,767 or the closest of these values will be suhstituted. Integer4 values must lie between – 2,147,483,648 and 2,147,483,647.

For coordinate data, the X, Y, and Z fields are first initialized to zero. Next scan as follows:

1) any leading blanks and tabs and/or ignore them
2) a [as the first non – blank character
3) X coordinate: a real or integer number as described above
4) a comma
5) Y coordinate: another real or integer as described above
6) another comma
7) Z coordinate: another real or integer as described above
8) a]

The [and] are mandatory. All coordinates are optional. If only two are given it will be assumed to be X and Y. If only one is given it will be X. If given they must be separated by commas.

For Boolean data, scan as follows:

1) any 1eading blanks and/or tabs and ignore them 2) a T or a t as the first non – blank

If a T or a t is found, the value will be TRUE;
otherwise it is assumed to be false.

Internal Data Format

System Date and Time

Some intrinsic procedures such as RSubrecIL, MSubrecIL, WSubrecIL use the system time and date. These values are encoded into two hytes. The bits are ordered from 15 to 0, from left to right. These are the formats:

System Time: <Hours>		<Min>	<Sec(2 sec.increments>
	Bit 15 – 11	Bit 10–5	Bit 4 – 0

System Date:<Year(since 1980)> <Month> <Day>
Bit 15 – 9 Bit 8 – 5 Bit 4 – 0

Transformation Matrix

Many intrinsic procedures have a parameter known as “Transform” in their description in Chapter 4 of this manual. This represents the transformation matrix used by Personal Designer.

When you insert entities into a part in Personal Designer, the data describing the entities is stored in the database. Even if you are working in different views or construction planes, the coordinate data is always stored in reference to the X, Y, and Z axis known as the model coordinate system. Personal Designer automatically converts data from the local coordinate system defined by the construction plane to model coordinates. This conversion is often referred to as “mapping” coordinates from one ”space” to another. For example, ”mapping from CPL space to model space.”

A transformation matrix is a matrix of numbers which is used to convert the data from the X, Y, and Z axis of model coordinates to the axes of your local coordinate system (CPL) and vice versa.

Many UPL intrinsic procedures require this matrix. You should declare an array of 15 real elements in your program and use this array to hold the transformation matrix. You rarely have to manipulate elements of the transformation matrix. In most situations, it will be returned from another procedure as an array parameter of 15 real numhers. The array can then be used to pass the matrix to another intrinsic procedure which requires it.

Internal Data Format

Example:

```
proc main

real Transform(15)

GetCPL(7, Transform(1))

end proc
```

Below is a description of the contents of the Transform array:

The first nine elements make up three sets of three real numbers. These sets can be thought of as unit vectors which describe the orientation of the local coordinate system's X, Y, and Z axes to the model coordinate system.

xx xy xz – unit vector for local system's x – axis

yz yy yz – unit vector for local system's y – axis

zx zy zz – unit vector for local system's z – axis

The next three elements define a vector which gives the offset of the local coordinate system's origin from that of the model coordinate system.

offx offy offz

The last three elements give the scaling factors about the model coordinate system's axes.

SCl X SCl Y SCl Z