17. READING AND PRINTING

This chapter describes the macros, functions, and methods used to read LOOPS objects from and print LOOPS objects to file storage, hash array storage, and the user display.

17.1 Reading Objects

This section describes the functions to read LOOPS objects.

Name	Type	Description
$	NLambda Function	Returns a pointer to the object; does not evaluate its argument.
$!	Function	Returns a pointer to the object; evaluates its argument.
$C	NLambda Function	Gets the class record.

These functions use the Common Lisp form **#,** in the return display. This form signals a read-time evaluation and is briefly described here.

Form	Description
#,<*form*>	Reads <*form*>, evaluates it, and returns that value.
#,($& <*form*>)	Form in which instances appear if they are not prettyprinted.
#,($C *className)*	Similar to **#,**($ *className*), except that it creates the class if it does not already exist.

(**$** *name*) [NLambda Function]

Purpose/Behavior: Returns a pointer to the LOOPS object specified by *name* and does not evaluate *name*. If no object exists for *name*, NIL is returned. If ***PRINT-PRETTY*** is set to T, the object will be prettyprinted in the Executive window.

Arguments: *name* A LOOPS name.

Returns: Pointer to a LOOPS object or NIL; see Behavior.

Example: In line 18, *name* is an instance. The value is returned and the **DEFINST** form is printed.

In line 19, *name* is a class whose class name is returned and printed.

In line 20, **NotAnObject** has not been declared as a LOOPS object and therefore returns NIL.

```
18←($ Window1)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))

19←($ Window)
#,($C Window)

20←($ NotAnObject)
NIL
```

(**$!** *name*) [Lambda Function]

Purpose/Behavior: Returns a pointer to the LOOPS object specified by *name* where *name* is evaluated. If no object exists for *name*, NIL is returned. If ***PRINT-PRETTY*** is set to T, the object will be prettyprinted in the Executive window.

Arguments: *name* Evaluates to a valid LOOPS name.

Returns: Pointer to a LOOPS object or NIL; see Behavior.

(**$C** *name*) [NLambda Function]

Purpose: Allows forward references to classes.

Use ($ *name*) instead of (**$C** *name*).

Behavior: Varies according to the arguments.

- If there is a class record for *name*, the function returns the class name.
- If there is no class record for *name*, the function attempts to create the class. This differs from the behavior of ($ *name*) which does not attempt any initialization if no LOOPS object is found.

Arguments: *name* A LOOPS name.

Returns: Value depends on the arguments; see Behavior.

Example: If *name* is not a LOOPS object, as shown in line 21, **$C** defines and returns a class for *name*, as shown in line 22. Line 23 shows the default class which is created in the Common Lisp Executive by **$C** when no class is found for *name*.

```
21←($ aCompletelyNewClass)
NIL

22←($C aCompletelyNewClass)
#,($C aCompletelyNewClass)

23←(← ($C aCompletelyNewClass) PP)
aCompletelyNewClass

(DEFCLASS aCompletelyNewClass
   (MetaClass Class)
   (Supers Tofu))
```


17.2 PRINT FLAGS

17.2 Print Flags

This section describes three variables that affect the way that objects are printed in LOOPS:

- **ObjectDontPPFlag**
- **ObjectAlwaysPPFlag**
- ***PRINT-PRETTY***

All these variables have a default value of NIL.

The **ObjectDontPPFlag** and **ObjectDontPPFlag** variables affect how contained objects are printed and are used to override the ***PRINT-PRETTY***, which affects how the top-level objects are printed. (See the *Interlisp-D Reference Manual* for more information on the ***PRINT-PRETTY***.) These variables interact as follows:

- If **ObjectDontPPFlag** is NIL and ***PRINT-PRETTY*** is T, objects are prettyprinted.
- **ObjectDontPPFlag** is T overrides ***PRINT-PRETTY*** is T.
- **ObjectAlwaysPPFlag** is T overrides ***PRINT-PRETTY*** is NIL.

ObjectDontPPFlag [Variable]

Purpose/Behavior: Used internally to prevent recursive printing of objects. If **ObjectDontPPFlag** is set to a non-NIL value, and **ObjectAlwaysPPFlag** is set to NIL, only the object name is printed. If this flag is NIL, all of the information contained within an instance is printed. The setting of this flag interacts with ***PRINT-PRETTY*** as shown in the examples below.

ObjectAlwaysPPFlag [Variable]

Purpose/Behavior: Controls printing the long form of all instances. When this variable is set to a non-NIL value, the long form of all instances are printed. This is the same form generated by (←*obj* **PP**). The **ObjectAlwaysPPFlag** overrides the effect of the **ObjectDontPPFlag**. Printing the long form of instances can lead to infinite loops or very long printouts. For example, if you have an object referencing another object which in turn references the first object, printing causes an infinite loop. If you have references to other LOOPS objects in the object you are printing, the long form of every object that can be reached from the top object is printed.

Example: This example shows the interaction of all print flags.

```
23←(SETQ *PRINT-PRETTY* NIL)
NIL

24←(SETQ ObjectDontPPFlag NIL)
NIL

25←(← ($ Window) New 'Window2)
#,($& Window (NEW0.1Y%:.;h.eN6 . 502))

26←(← ($ Window2) Shape)
(47 145 99 89)

27←($ Window2)
#,($& Window (NEW0.1Y%:.;h.eN6 . 502))
```

- Change the value of ***PRINT-PRETTY*** to T.

```
28←(SETQ *PRINT-PRETTY* T)
T
```

```
29←($ Window2)
(DEFINST (Window2 (NEW0.1Y%:.;h.eN6 . 502))
         (left 47)
         (bottom 145)
         (width 99)
         (height 89))
```

- Change the value of **ObjectDontPPFlag** to T.

```
30←(SETQ ObjectDontPPFlag T)
T

31←($ Window2)
#,($& Window (NEW0.1Y%:.;h.eN6 . 502))
```

- Assume the following commands have been entered:

```
(DefineClass 'PPTest)
(← ($ PPTest) AddIV 'testIV)
(← ($ PPTest) New 'PPTest1)
(← ($ PPTest) New 'PPTest2)
(←@ ($ PPTest1) testIV ($ PPTest2))
(←@ ($ PPTest2) testIV ($ PPTest1))

(SETQ *PRINT-PRETTY* T)
(SETQ ObjectDontPPFlag T)
(SETQ ObjectAlwaysPPFlag T)
```

- Print the instances.

```
53←($ PPTest1)
(DEFINST PPTest (PPTest1 (NEW0.1Y%:.;h.eN6 . 502))   )
```

- Reset the ***PRINT-PRETTY*** and print the instances again.

```
54←(SETQ *PRINT-PRETTY* NIL)
NIL

55←($ PPTest1)
#,($& PPTest (NEW0.1Y%:.;h.eN6 . 513))
```


17.3 PRINTING CLASSES

17.3 Printing Classes

This section describes the methods used to print classes and information about classes.

Name	Type	Description
FileOut	Method	Prints long pretty form of the class to a file or a display stream.
PP	Method	Prettyprints the class definition to a file or a display stream.
PP!	Method	Prints the information about the class from all levels of inheritance.

PPV!	Method	Prints the variable information about the class from all levels of inheritance.

(← *self* **FileOut** *file*) [Method of Class]

Purpose: Prints the long pretty form of the class to a file or to display stream.

Behavior: Prints a **DEFCLASS** form for the class *self*. The **DEFCLASS** form, which is the way classes are defined, always includes the name of the class, the **MetaClass**, and the **Supers**. If there are **ClassVariables** and **InstanceVariables** defined for the class, these along with their values are also included in the **DEFCLASS** form. **FileOut** formats the output with special fonts and tab positions.

Arguments:

self	A class.
file	The stream on which *self* is to be printed. If NIL, or not given, prints to the **TTYDisplayStream**.

Returns: self

Categories: Classes

Specializations: Class, Method

Example: This example shows the **DEFCLASS** form for **TestClass**. If a **DEFCLASS** form cannot be generated for *self*, a **Break** occurs with the message "*var* is not defined as a class. Type OK to ignore this class and go on."

```
24←(← ($ TestClass) FileOut)
(DEFCLASS TestClass
    (MetaClass Class Edited%:  (* --) )
    (Supers Object)
    (InstanceVariables (testIV 1234 testProp1 1
                                testProp2 2 doc
   (* --))))

#,($C TestClass)
```

(← *self* **PP** *file*) [Method of Class]

Purpose: Prettyprints LOOPS **OBJECT.CLASS.PP** to a file or to display stream.

Behavior: Prettyprints the class on *file*, if provided. If *file* is not given, look first to the **PPDefault**, which is by default the Common Lisp Executive Window, and then to the **TTYDisplayStream**. The output is printed and formatted by the method **Class.FileOut**.

Arguments:

self	A pointer to a class.
file	Stream to prettyprint to.

Returns: Name of class.

Categories: Class

Specializes: Object

Example: This example shows a call to **PP** on the class **SupersBrowser**, which uses the **TTYDISPLAYSTREAM** as the default output stream.

```
26←(← ($ SupersBrowser) PP)
(DEFCLASS SupersBrowser
    (MetaClass Class Edited%:  **COMMENT**
```

```
                    doc  "Browses upwards from a class
to all of its supter.")
   (Supers ClassBrowser)
   (InstanceVariables (title "Supers browser")))

SupersBrowser
```

(← *self* **PP!** *file*) [Method of Class]

Purpose: Prints the information about LOOPS **OBJECT.CLASS.PP** from all levels of inheritance.

Behavior: Prints a listing of the following items along with any applicable documentation, values and arguments for each item: **MetaClass**, **Supers**, **Instance Variables**, **Class Variables**, **Prototypes**, and **Methods**.

Prints the information on *file*, if provided. If *file* is not given, look first to the **PPDefault**, which is by default the Common Lisp Executive Window, and then to the **TTYDisplayStream**.

Arguments:

self	A pointer to a class.
file	Stream to print to.

Returns: *self*

Categories: Classes

Specializes: Object

Example: This example shows a partial output of the call to **PP!** on the class **SupersBrowser** which uses the **TTYDISPLAYSTREAM** as the default output stream. The dots indicate additional information.

```
27←(← ($ SupersBrowser) PP!)
#,($ SupersBrowser)
MetaClass and its Properties
  Class  Edited: (* smL 11-Jun-86 13:18)  doc
Browses upwards from a class to all of its
supers.
Supers
  (ClassBrowser IndexedObject LatticeBrowser --)
Instance Variable Descriptions
  left NIL doc left position of window
  bottom NIL doc
bottom position of window
  width 64 doc
outer width of window, including border
  height 32 doc
outer height of window, including border
.
.
.
Class Variables
  RightButtonItems ((Close (Close (Close --)
)) Snap Paint --) doc
Items to be done if Right button is selected
.
.
.
Methods
  AddCategoryMenu ClassBrowser.AddCategoryMenu
doc NIL args NIL
```

```
  AddNewCV ClassBrowser.AddNewCV
doc NIL args NIL
  AddNewIV ClassBrowser.AddNewIV
doc NIL args NIL
  AddNewMethod ClassBrowser.AddNewMethod
doc NIL args NIL
.
.
.

#,($C SupersBrowser)
```

(← *self* **PPV!** *file*) [Method of Class]

Purpose: Prints the variable information about the class from all levels of inheritance.

Behavior: Similar to (← *self* **PP!** *file*), except that only the **MetaClass**, **Supers** list and information about **Class Variables** and **Instance Variables** is printed.

Arguments:

self	A pointer to a class.
file	Stream to print to.

Returns: *self*

Categories: Classes

Specializes: Object

Example: This example shows a partial output of the call to **PPV!** on the class **SupersBrowser** which used the **TTYDISPLAYSTREAM** as the default output stream. The dots indicate additional information.

```
28←(← ($ SupersBrowser) PP!)
#,($ SupersBrowser)
MetaClass and its Properties
  Class Edited: (* smL 11-Jun-86 13:18)  doc
Browses upwards from a class to all of its
supers.
Supers
  (ClassBrowser IndexedObject LatticeBrowser --)
Instance Variable Descriptions
  left NIL doc left position of window
  bottom NIL doc
bottom position of window
  width 64 doc
outer width of window, including border
  height 32 doc
outer height of window, including border
.
.
.
Class Variables
  RightButtonItems ((Close (Close (Close --)
)) Snap Paint --) doc
Items to be done if Right button is selected
.
.
.
#,($C SupersBrowser)
```


17.4 PRINTING OBJECTS

17.4 Printing Objects

This section describes the methods for printing LOOPS objects.

Name	Type	Description
PrintOn	Method	Provides the default print function for LOOPS objects.
FileOut	Method	Prettyprints a LOOPS instance.
PP	Method	Prettyprints an object to a file or display stream.
PP!	Method	Prints all the information about the instance from all levels of inheritance.
PPV!	Method	Prints the variable information about the instance from all levels of inheritance.

(← *self* **PrintOn** *file*) [Method of Object]

Purpose: Provides the default print function for LOOPS objects.

Behavior: Returns a form suitable for the Lisp function **DEFPRINT**, which produces the standard LOOPS object print form **#**,($ *objname*). (See the *Lisp Release Notes* and the *Interlisp-D Reference Manual* for more information on **DEFPRINT**.)

Arguments:
self A LOOPS object.
file A stream to print to.

Returns: ("#," $ ObjectName)

Categories: Object

Example: This example shows the results of calling **PrintOn** with the instance, **Window1**.

```
28←(← ($ Window1) PrintOn)
("#," $ Window1)
```

(←*self* **FileOut** *file*) [Method of Object]

Purpose: Prettyprints a LOOPS instance.

Behavior: If an object is found for *self*, this method prints the **DEFINST** form for the object to the *file*. For a description of **FileOut** where *self* is a class, see Section 17.3 "Printing Classes."

The **DEFINST** form always includes the name of the class to which the object belongs and the UID for the object. Names attached to the object and **InstanceVariables** bindings for the object are also included in the **DEFINST** form. **FileOut** formats the output with special fonts and tab positions.

Arguments:
self A LOOPS object.
file Stream to print to.

Returns: *self*

Categories: Instances

Example: This example shows the **DEFINST** forms for the object **Window1**.

```
29←(← ($ Window1) FileOut)
```

```
(DEFINST Window (Window1 (
NEW0.1Y%:.;h.eN6 . 495))
      (left 288)
      (bottom 242)
      (width 331)
      (height 149))
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))
```

(← *self* **PP** *file*) [Method of Object]

Purpose: Prettyprints an object to a file or display stream.

Behavior: Temporarily sets the **ObjectDontPPFlag** to prevent infinite loops in the print. Prettyprints the output with special fonts and tab positions and prints the **DEFINST** form of the object. If *file* is not given, look first to the **PPDefault**, which is by default the Common Lisp Executive Window, and then to the **TTYDisplayStream**.

Arguments:

self	A LOOPS object.
file	Stream to print to.

Returns: Name of object.

Categories: Object

Specializations: Class

Example: This example shows the results of sending the instance **Window1** the message **PP**.

```
30←(← ($ Window1) PP)
(DEFINST Window (Window1 (
NEW0.1Y%:.;h.eN6 . 495))
      (left 288)
      (bottom 242)
      (width 331)
      (height 149))
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))
```

(← *self* **PP!** *file*) [Method of Object]

Purpose: Prints the information about the instance from all levels of inheritance.

Behavior: Prints a listing of the following items along with any applicable documentation, values and arguments for the each item: **Instance Variables**, **Class Variables**, and **Methods**.

If *file* is not given, look first to the **PPDefault**, which is by default the Common Lisp Executive Window, and then to the **TTYDisplayStream**

Arguments:

self	A LOOPS object.
file	Stream to print to.

Returns: *self*

Categories: Object

Specializations: Class

Example: This example shows a partial output of a call to **PP!** on the instance **Window1**. Dots indicate additional information.

```
31←(← ($ Window1) PP!)

#,($ Window1)
Instance Variables
  left NIL doc left position of window
  bottom NIL doc
bottom position of window
  width 12 doc
outer width of window, including border
  height 12 doc
outer height of window, including border
.
.
.
Class Variables
  RightButtonItems ((Close (Close (Close --)
)) Snap Paint --) doc
Items to be done if Right button is selected
.
.
.
Methods
  AfterMove Window.AfterMove doc NIL
args NIL
.
.
.
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))
```

(← *self* **PPV!** *file*) [Method of Object]

Purpose: Prints the variable information about the instance from all levels of inheritance.

Behavior: Similar to (← *self* **PP!** *file*) except that only the information about the **Class Variables** and **Instance Variables** is printed.

Arguments:

self	A LOOPS object.
file	Stream to print to.

Returns: *self*

Categories: Object

Specializations: Class

Example: This example shows a partial output of a call to **PPV!** on the instance **LCDInstance**. Dots indicate additional information.

```
31←(← ($ Window1) PPV!)
#,($ Window1)
Instance Variables
  left NIL doc left position of window
  bottom NIL doc
bottom position of window
  width 12 doc
outer width of window, including border
  height 12 doc
outer height of window, including border
.
.
```

```
.
Class Variables
  RightButtonItems ((Close (Close (Close --)
)) Snap Paint --) doc
Items to be done if Right button is selected
.
.
.
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))
```


17.5 PRINTING ACTIVE VALUES

17.5 Printing Active Values

This section describes methods and variables used for printing active values. For more information on active values, see Chapter 8, Active Values.

(← *self* **AVPrintSource**) [Method of ActiveValue]

Purpose: Constructs a form used by **DEFPRINT** to write active values to files.

Behavior: An annotatedValue determines how it prints out by sending the **AVPrintSource** message to its wrapped **ActiveValue**.

The default method in **ActiveValue** returns a list of the form:

```
("#,"$AV className avNames(ivName value propName value ...)(ivName ...) ...)
```

which causes the annotatedValue to print out as

```
#,($AV className avNames(ivName value propName value ...)(ivName ...) ...)
```

Arguments: *self* An **ActiveValue**

Returns: A form suitable for use by the Interlisp-D function **DEFPRINT**. Result should be a pair of the form (item1 . item2); item1 will be printed using **PRIN1**, and item2 will be printed using **PRIN2** (see the *Lisp Release Notes* and the *Interlisp-D Reference Manual* description of **DEFPRINT**).

In the return list,

`className` Name of the class of the **ActiveValue**.

`avNames` List of names of *self*; the last element being the unique identifier (UID) of *self*

`(ivName value propName value ...)`
List that describes the state of the instance variables of the **ActiveValue**.

Categories: Instances of the **ActiveValue** class

Example: The following command gets a pointer to an active value:

```
32←(GetValueOnly ($ Window1) 'window)
#,($AV LispWindowAV ((N^W0.1Y%:.;h.Lh9 . 503)) (localState
{WINDOW}#374,55554))
```

The following shows the result of an **AVPrintSource** message. (This is typically passed on to **DEFPRINT** within the internals of the system.)

```
33←(←(GetValueOnly ($ Window1) 'window) AVPrintSource)
("#," $AV LispWindowAV ((N^W0.1Y%:.;h.Lh9 . 503))
(localState {WINDOW}#374,55554))
```

DefaultActiveValueClassName (Variable)

Purpose: The class **ExplicitFnActiveValue** is the default class for active values. This class mimics the previous style of LOOPS active values (see Appendix A, Previous Active Values). For specialized applications, you may want a different class of active value to use for this purpose.

17.6 PRINTING METHODS

17.6 Printing Methods

This section describes the following methods used to print methods:

Name	Type	Description
PPDefault	Variable	Identifies where the output for prettyprinting is sent.
PPMethod	Method	Prettyprints the method for a class.
MethodDoc	Method	Prints the documentation for the method for a class.
MethodSummary	Method	Prints a summary of the methods attached to a class.

PPDefault [Variable]

Purpose: Bound to a window used as the default output stream for the methods **PPMethod**, **MethodDoc**, and **MethodSummary**. Initially set to the Common Lisp Executive Window.

(← *self* **PPMethod** *selector*) [Method of Class]

Purpose: Prettyprints the method specified by *selector* for the class *self*.

Behavior: If *selector* is not specified, this opens a menu of the methods attached to the class *self*. The method, as chosen either from the menu or passed to the method in *selector*, is prettyprinted to the primary output stream. If *self* is not a class, a break occurs with the error,"(← ($ *self*) **PPMethod** *selector*) not understood."

The output is sent to the value of the variable **PPDefault**, which is by default the Common Lisp Executive Window.

Arguments: *self* A LOOPS object.

selector Method to print.

Returns: Class.Selector

Categories: Classes

Example: This example shows the results of prettyprinting the method **Shape** on the class **Window** using **PPMethod**.

```
35←(← ($ Window) PPMethod 'Shape)

(Method ((Window Shape) self newRegion noUpdateFlg) (* ...)
 "Shapes outside of region to specified shape."
 (_ self Shape1 [OR newRegion (GETREGION NIL NIL
 (WINDOWPROP (@ window) 'REGION]
     noUpdateFlg))
```

with **(Window Shape)** bold.

(← *self* **MethodDoc** *selector*) [Method of Class]

Purpose: Prints the documentation for the method specified by *selector* for the class *self*.

Behavior: If selector is not specified, this opens a menu of all methods attached to the class from all levels of inheritance. When you choose an item, the documentation for that method, the arguments needed, and the class defining the method are prettyprinted to the **PPDefault** window, which is by default the Common Lisp Executive Window. You can continue to make selections from the menu or press a mouse button outside the menu to stop.

Arguments: *self* A pointer to a class.

selector Method to be printed.

Returns: NIL

Categories: Class

Example: This example shows the output from calling **MethodDoc** for the class **LoopsIcon**. Three methods were chosen from the menu in succession: **AfterMove**, **BrowseObject,** and **Clear**. **BrowseObject** is attached to **Window** so the class where it is defined is not explicitly listed. **AfterMove** and **Clear** are defined, respectively, on the classes **NonRectangularWindow** and **Window**.

```
36←(← ($ LoopsIcon) MethodDoc)

class: LoopsIcon (from NonRectangularWindow)
 selector: AfterMove
args: NIL
doc: The window has been moved. Update the
left and bottom.

class: LoopsIcon selector:
BrowseObject
args: NIL
doc: Put up a browser starting on selected
object.

class: LoopsIcon (from Window) selector:
Clear
args: NIL
doc: Calls CLEARW on window.
```

(← *self* **MethodSummary** *dontPrintFlg printFile*) [Method of Class]

Purpose: Prints a summary of the methods attached to the class *self*.

Behavior: Prettyprints the documentation from the classes directly attached to the class *self*. Printing is done to the file *printFile*. If *printFile* is not specified, **MethodSummary** prints to the **PPDefault** window, which is by default the Common Lisp Executive Window. If the **ObjectDontPPFlg** is T, the output is not displayed in pretty format.

Arguments:
self A pointer to a class.

dontPrintFlg If non-NIL, does not prettyprint.

printFile File to print to.

Returns: NIL

Categories: Class

Example: This example shows the results of sending the message **MethodSummary** to the class **IconWindow**. Only information about the methods defined at the class **IconWindow** are printed.

```
37←(← ($ IconWindow) MethodSummary)
((GetMenuItems IconWindow.GetMenuItems args
               (itemType)
               doc
               NIL))
```


17.7 Unique Identifiers (UIDs)

Unique Identifiers (UIDs) are used to store and retrieve objects. In general, objects do not have UIDs, with the following exceptions:

- When an object is named.
- When an instance of an indexed obect is created, it gets a UID.
- When an object is printed.

The following table shows the functions in this section.

Name	Type	Description
HasUID?	Function	Returns the UID for a specified object.
UID	Function	Returns the UID for a specified object and creates a UID for the object if one does not already exist.
GetObjFromUID	Function	Retrieves the LOOPS object records.
MapObjectUID	Function	Applies a function to every LOOPS object that has a UID.

(**HasUID?** *obj*) [Function]

Purpose: Returns the UID for *obj*.

Behavior: If the *obj* has a UID, the function returns the UID. If *obj* is an object but has no UID, it returns NIL. If *obj* is not an object, it generates an error with the message, "ARG NOT OBJECT."

Arguments: *obj* A LOOPS object.

Returns: The UID for *obj*.

Example: Line 39 shows the results of calling **HasUID?** for an instance **Window1**, line 40 for a class **Window**, and line 41 for a new instance of **Window**.

```
39←(HasUID? ($ Window1))
(NEW0.1Y%:.;h.eN6 . 495)

40←(HasUID? ($ Window))
(NEW0.1Y%:.;h.eN6 . 255)

41←(HasUID? (← ($ Window) New))
NIL
```

(**UID** *obj*) [Function]

Purpose: Returns UID for *obj*. If object does not have UID, this function creates a UID for the *obj*.

Behavior: If the object has a UID, this function returns the UID; otherwise it creates a UID for the object.

Arguments: *obj* A LOOPS object.

Returns: The UID for *obj*.

Example: Line 45 shows the results of calling UID with the class **Object**. Line 46 shows the results of calling UID with an instance which does not have a UID.

```
45←(UID ($ Object))
(NEW0.1Y%:.;h.eN6 . 7)

46←(UID (← ($ Window) New))
(NEW0.1Y%:.;h.eN6 . 519)
```

(**GetObjFromUID** *uid*) [Function]

Purpose: Retrieves the LOOPS object records of object whose UID is *uid*.

Behavior: Returns the object associated with a UID, or returns NIL if *uid* is not a valid UID.

Arguments: *uid* The internal identifier.

Returns: Pointer to the object.

Example: In this example, **Window1UID** was previously set to the UID for the instance **Window1**. **GetObjFromUID** retrieves the record for **Window1** using **Window1UID** and prettyprints the **DEFINST** form for **Window1** to the **TTYDisplayStream**.

```
42←(SETQ Window1UID (UID ($ Window1]
(NEW0.1Y%:.;h.eN6 . 495)

43←GetObjFromUID Window1UID)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495)
```

(**MapObjectUID** *fn*) [Function]

Purpose: Applies the function *fn* to every LOOPS object.

Behavior: Maps the function *fn* to every UID object that has a UID.

Arguments: *fn* Function to be applied.

Returns: Used as a side effect only.

Example: This example shows a partial listing of the results of applying the user-defined function **PPUID** (see line 47) to every LOOPS object using **MapObjectUID**. **PPUID** prints the UID of *obj* to the **TTY** display stream. A complete output of this call to **MapObjectUID** lists the UID for every LOOPS object currently defined in the system.

```
45←(DEFINEQ (PPUID (LAMBDA (OBJ) (PRIN2
(UID OBJ)))))
(PPUID)

46←PP PPUID
FNS definition for PPUID:
(PPUID
  [LAMBDA (OBJ)  **COMMENT**
    (PRIN2 (UID OBJ])

47←(MapObjectUID 'PPUID)
(NEW0.1Y%:.;h.Lh9 . 526)(NEW0.1Y%:.;h.Lh9 . 527)
(NEW0.1Y%:.;h.Lh9 . 524)(NEW0.1Y%:.;h.Lh9 . 525)
(NEW0.1Y%:.;h.Lh9 . 522)(NEW0.1Y%:.;h.Lh9 . 523)
.
.
.
#<Hash-Table @ 66,72106>
```

[This page intentionally left blank]