LOOPSBACKWARDS

By: Bob Bane (Bane.pa@Xerox.com)

15-Dec-87

INTRODUCTION

LOOPSBACKWARDS allows you to run files which were previously converted from the Buttress release of LOOPS to the Koto release. Unlike the Koto version of LOOPSBACKWARDS conversion methods for moving files from Buttress LOOPS to Koto LOOPS are included but **not supported**. *We strong recommend that you convert LOOPS source code from Buttress to Koto using the Koto release of LOOPS.* Conversion of Koto LOOPS source code to Lyric LOOPS is done using the CONVERSION-AIDS users' module.

The changes between the Buttress and Koto releases are described in detail in the *LOOPS Release Notes* for the **Koto** release. Old features that are included in LOOPSBACKWARDS are summarized here:

- Many functions that were removed from the Buttress release are defined.
- The messages **List** and **List!** are available.
- The operation of old style active values, from the Buttress release, are provided.
- Support for reading old style macros, such as #(localState getFn putFn) or #$Mumble, is available.

LOOPSBACKWARDS is an unsupported LOOPS users module. It is strongly recommended that it only be used as part of an effort to upgrade very old LOOPS code to newer releases. It will allow very old LOOPS code to run well enough that it can be rewritten for a newer relese.

INSTALLATION

LOOPSVCOPY will be automatically loaded by LOOPSBACKWARDS.

FUNCTIONS

LOOPSBACKWARDS includes **ExplicitFnActiveValue** and **DefAVP**. **ExplicitFnActiveValue** allows the user code triggered by Get- or Put- accesses to be stored within functions which are pointed to by instance variables rather than requiring the redefinition of **GetWrappedValue** or **PutWrappedValue**. These functions must have the form specified in the **DefAVP** function.

ExplicitFnActiveValue [Class]

Purpose: Mimics the behavior of the Buttress style of active values.

Behavior: Get- accesses to the wrapped variable cause the **getFn** to be called, Put- accesses cause **putFn** to be called. Enables the old style activeValue to look like the new style without changing any functionality.

The **getFn** is called by the **ExplicitFnActiveValue GetWrappedValue** method. This method passes to the **getFn** the arguments defined by **DefAVP** as described in the *LOOPS Users' Modules*.

The **putFn** is called by the **ExplicitFnActiveValue PutWrappedValue** method. This method passes to the **putFn** the arguments defined by **DefAVP** as described in the *LOOPS Users' Modules*.

Instance Variables:

localState	A place for data storage.
getFn	The name of a function applied when the active variable is read.
putFn	The name of a function applied when the active variable is changed.

Example:

```
32← (← ($ Bin) New 'bin4)
#,($& Bin (|DAW0.1Y:.H53.]99| . 521))

33← (← ($ ExplicitFnActiveValue) New 'EFAV1)
#,($& ExplicitFnActiveValue (|DAW0.1Y:.H53.]99| . 522))

34←(DEFINEQ (PrintOnGet
          (self varName localSt propName activeVal type)
          (PRINTOUT T "I am:" , activeVal T) localSt))
(PrintOnGet)

35←(←@ ($ EFAV1) getFn 'PrintOnGet)
PrintOnGet

36←(← ($ EFAV1) AddActiveValue ($ bin4) 'height)
#,($A #,NestedNotSetValue PrintOnGet NIL)

37←(←@ ($ bin4) height 123)
123

38←(@ ($ bin4) height)
```

```
I am: #,($ EFAV1)
123
```

(**DefAVP** *fnName putFlg*) [Function]

Purpose: Creates a template for defining an active value function.

Behavior: Creates a template and leaves you in the Interlisp function display editor.

Arguments:

fnName	Name of the function.
putFlg	T indicates function is a **putFn**, NIL indicates a **getFn**.

Returns: The function name on exit from the editor .

Example: In each of the following cases the template only is shown. User code is to be added immediately after the comment by using the display editor.

```
66←(DefAVP 'AGetFn)
AGetFn

67←PP* AGetFn
(AGetFn
  [LAMBDA (self varName localSt propName activeVal type)
         (* This is a getFn. The value of this getFn is
returned as the value of the enclosing GetValue.)
    localSt])
(AGetFn)

68←(DefAVP 'APutFn T)
APutFn

69←PP* APutFn
(APutFn
  [LAMBDA (self varName newValue propName activeVal type)
          (* This is a putFn. ***NOTE*** The value of this
function will be returned as the value of any enclosing
PutValue. This usually means that you want to return the value
returned by PutLocalState.)
    (PutLocalState activeVal newValue self varName propName
type])
(APutFn)
```

[This page intentionally left blank.]