*File created:* 23-Apr-86 15:06:59 **{POGO:PARC:XEROX}<LOOPS>TESTER>LTSUB.;3**

*changes to:* (FNS RunTest)

*previous date:* 7-Apr-86 17:12:53 {POGO:PARC:XEROX}<LOOPS>TESTER>LTSUB.;2

*Read Table:* OLD-INTERLISP-FILE

*Package:* INTERLISP

*Format:* XCCS



```
(RPAQQ LTSUBCOMS
       [(* File created by MITTAL)
        (CLASSES LOOPSCasesMeta LOOPSClassSuper LOOPSTestAbstract LOOPSTestMeta LOOPSTestSimple)
        (METHODS LOOPSClassSuper.Destroy LOOPSClassSuper.GetTestCode LOOPSClassSuper.SaveInstance
               LOOPSTestAbstract.Describe LOOPSTestAbstract.GetTestCode LOOPSTestAbstract.Reset
               LOOPSTestAbstract.ResetSelf LOOPSTestAbstract.TestSelf LOOPSTestMeta.BeginLOOPSTest
               LOOPSTestMeta.GetTestCode LOOPSTestMeta.New LOOPSTestSimple.MakeEditSource LOOPSTestSimple.TEST)
        (FNS ATEST BasicTest BeginLoopsTest CleanupTester CloseCurrentEnvironment ControlledErrorset DoTest
             DoTestSelf EvaluateANDALLTest EvaluateANDTest EvaluatePROGTest EvaluateTest ExaminePreviousTry
             GenerateTestList InteractiveLoopsTest MakeTest PerformSetup PerformAltTest PerformTest
             PrintFailedExp PrintTestCode PrintTestHeader ReasonNotDone RunTest TestErrorBreak TestObjectDesc
             FlashTestBrowser InformTestBrowser TestFromFiles LTLoadFile LTLoadFile?)
        (MACROS NotSetValue? PrintIfLev UnknownValue? ValueExists? ValueNonNIL?)
        (* LTTestFiles - list of test files loaded so far)
        (INITVARS (LTBROWSER NIL)
               (KBTestsFlg T)
               (LTDontAsk NIL)
               (LTCompIntFlg T)
               (LTLOADEDREST NIL)
               (LTLOGFLAG NIL)
               (LTError NIL)
               (LTResult T)
               (LTTestFiles NIL))
        (INSTANCES * LTSUBINSTANCES)
        (DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                             (NLAML DoTest)
                                                                             (LAMA])
```

(* * File created by MITTAL)

```
(DEFCLASSES LOOPSCasesMeta LOOPSClassSuper LOOPSTestAbstract LOOPSTestMeta LOOPSTestSimple)

(DEFCLASS LOOPSCasesMeta (MetaClass MetaClass Edited:            (* sm: "19-OCT-82 11:01"))
       (Supers LOOPSTestMeta)
       (ClassVariables (Instances NIL)
              (InstanceComsVar LTCASESINSTANCES)
              (InstancePrefix LTUC)
              (UnnamedInstanceCount 0)))

(DEFCLASS LOOPSClassSuper (MetaClass LOOPSTestMeta Edited:       (* sm: "18-Mar-85 10:04"))
       (Supers NamedObject)
       (ClassVariables (InstanceComsVar LTSUBINSTANCES)))

[DEFCLASS LOOPSTestAbstract (MetaClass LOOPSTestMeta Edited:     (* sm: "18-Mar-85 16:11"))
       (Supers LOOPSClassSuper)
       [ClassVariables (ClassPreTest NIL doc
```

(* list of objects that need to be tested for a class of test objects and are tried before an objects own PreTest)

```
)
              (ClassTested? T doc                                (* set to T/NIL depending on ClassPreTest results))
              (ResetList ((IV Tested? (Tested? DoneOnce)
                              (Tested? Ignored)
                              (SetUp Tested?)
                              (SetUp FailedExp)
                              (TestExpr FailedExpr)
                              (TestExpr HowFailed]
       (InstanceVariables (TestDesc "the feature indicated by the name" doc
                                                                 (* A brief description of what is being tested))
              (SetUp NIL Tested? #. ($A U NIL FlashTestBrowser)
                     FailedExp NIL DontSave (Tested? FailedExp)
                     doc                                         (* a Lisp-like exp for setting up the test environment)
                     )
              (TestExpr T FailedExp NIL HowFailed NIL DontSave (HowFailed FailedExp)
                     doc                                         (* this is the actual test expression)
                     )
              (ResetExp NIL doc                                  (* a lisp like expression used for undoing any state change
                                                                 introduced by a test))
              (Tested? #. ($A U NIL InformTestBrowser)
                     Ignored NIL DoneOnce NIL DontSave Any doc   (* result of running the TestExpr))
              (AfterTest NIL doc
```

(* expression executed after TestExpr to do some resetting. Currently only used by LOOPSTestEnvironment and their

```
          subtests)
)
              (PreTestOf NIL DontSave (Value)
                     doc                                    (* dummy for current code. DONT USE HERE)
                     )
              (SubTest NIL doc                              (* dummy for current code. DONT USE HERE))
              (PreTest NIL doc                              (* dummy for current code. DONT USE HERE))
              (AltTest NIL doc                              (* dummy for current code. DONT USE HERE)]

[DEFCLASS LOOPSTestMeta (MetaClass MetaClass Edited:        (* sm: "13-OCT-82 16:14"))
       (Supers Class)
       (ClassVariables (Instances NIL doc                   (* list of all instances of a LOOPSTest class))
              (InstanceComsVar LTKERINSTANCES doc           (* variable used for saving instances on file.
                                                            should be changed if file is changed))
              (UnnamedInstanceCount 0 doc

          (* the counter used to generate new names of instances, if one is not given to NEW)
)
              (InstancePrefix LTU doc                       (* prefix attached to names of instances for convenient
                                                            identification)]

(DEFCLASS LOOPSTestSimple (MetaClass LOOPSTestMeta Edited:  (* sm: "18-Mar-85 16:11"))
       (Supers LOOPSTestAbstract)
       (ClassVariables (InstanceComsVar LTSUBINSTANCES)
              (InstancePrefix LoopsTest)
              (UnnamedInstanceCount 0)))

(METH LOOPSClassSuper Destroy NIL NIL)

(METH LOOPSClassSuper GetTestCode NIL                       (* returns NIL, as there is currently no Test Code here))

(METH LOOPSClassSuper SaveInstance NIL NIL)

(METH LOOPSTestAbstract Describe NIL NIL)

(METH LOOPSTestAbstract GetTestCode NIL                     (* returns the Test Code fields))

(METH LOOPSTestAbstract Reset (AlreadyReset)

          (* resets the Tested type values so prior test results are wiped out.
          Also Resets PreTest and ClassPreTest)

      )

(METH LOOPSTestAbstract ResetSelf (AlreadyReset)            (* resets the Tested type values so prior test results are wiped
                                                            out.)
      )

(METH LOOPSTestAbstract TestSelf (TestedLst)                (* performs the basic TEST for a TestObject but continues even
      if PreTests fail)
      )

(METH LOOPSTestMeta BeginLOOPSTest NIL                      (* TESTS THE KERNEL FEATURES OF LOOPS))

(METH LOOPSTestMeta GetTestCode NIL                         (* returns NIL, as no TestCode yet))

(METH LOOPSTestMeta New (name)

          (* inherits its Super's New and then adds the instance's name to filecoms and CV Instances)

      )

(METH LOOPSTestSimple MakeEditSource NIL                    (* Collect just those IVs the user should see))

(METH LOOPSTestSimple TEST NIL                              (* Run the simple test. No pretests are checked))

(DEFINEQ
```

## (LOOPSClassSuper.Destroy

```
  (Method ((LOOPSClassSuper Destroy)
           self)                                            (* smL " 7-Apr-86 16:37")

          (* * Destroys an instance; updates Instancecoms var and CV Instances)

                (LET ((name (@ name)))
                     (change (@ ::Instances)
                            (DREMOVE name DATUM))
                     [SET (@ ::InstanceComsVar)
                          (DREMOVE name (EVAL (@ ::InstanceComsVar]
                     (_Super)
                     name)))
```

## (LOOPSClassSuper.GetTestCode

```
  (Method ((LOOPSClassSuper GetTestCode)
           self)                                            (* sm: "21-OCT-82 14:06")
                                                            (* returns NIL, as there is currently no Test Code here)
```

```
                NIL))
```

## (LOOPSClassSuper.SaveInstance

```
  (Method ((LOOPSClassSuper SaveInstance)
           self)                                                (* sm: "15-Mar-85 13:30")
                                                                (* saves the instance in the var stored under CV
                                                                InstanceComsVar)
                [LET ((name (GetObjectRec self)))               (* (PushClassValueNew self (QUOTE Instances) name))
                     (EVAL (LIST (QUOTE PUSHNEW)
                                 (GetClassValue self (QUOTE InstanceComsVar))
                                 (KWOTE name)))
```

## (LOOPSTestAbstract.Describe

```
  (Method ((LOOPSTestAbstract Describe)
           self)                                                (* sm: "18-Mar-85 13:09")
                                                                (* prints a description of the TestObj)
                [PROG NIL
                      (printout TTY "Test Desc::" -2 (@ TestDesc)
                              T)
                      (COND
                         ((@ self Tested? DoneOnce)
                          (DescribePreviousTry self))
                         (T (printout TTY "Not tested yet. To test, send TEST message or use TestBrowser" T]))
```

## (LOOPSTestAbstract.GetTestCode

```
  (Method ((LOOPSTestAbstract GetTestCode)
           self)                                                (* sm: "18-Mar-85 13:09")
                                                                (* returns the Test Code fields)
                [MAPCAR (QUOTE (SetUp TestExpr ResetExp))
                       (QUOTE (LAMBDA (X)
                                      (LIST X (GetValue self X]))
```

## (LOOPSTestAbstract.Reset

```
  (Method ((LOOPSTestAbstract Reset)
           self AlreadyReset)                                   (* smL " 7-Apr-86 16:39")

          (* * resets the Tested type values so prior test results are wiped out.
          Also Resets PreTest and ClassPreTest)

                          (LET (HELPFLAG)
                               [COND
                                  ((NULL AlreadyReset)
                                   (SETQ AlreadyReset (CONS]
                               [COND
                                  ((NOT (FMEMB (@ name)
                                               (CAR AlreadyReset)))
                                   (TCONC AlreadyReset (@ name))
                                   (_ self ResetSelf AlreadyReset)
                                   (for x in (@ PreTest) do (_ (GetObjectRec x)
                                                               Reset AlreadyReset))
                                   (for x in (@@ ClassPreTest) do (_ (GetObjectRec x)
                                                                     Reset AlreadyReset]
                               T)))
```

## (LOOPSTestAbstract.ResetSelf

```
  (Method ((LOOPSTestAbstract ResetSelf)
           self AlreadyReset)                                   (* sm: "18-Mar-85 16:03")
                                                                (* resets the Tested type values so prior test results are wiped
                                                                out.)
                          (PROG NIL
                                (COND
                                   ((ValueNonNIL? (@ self Tested? DoneOnce))
                                    (printout TTY "RESETting.." -4 (@ name)
                                            T)
                                    (for x in (@@ ResetList)
                                       do (SELECTQ (CAR x)
                                                   (IV [for y in (CDR x) do (COND
                                                                              ((ATOM y)
                                                                               (PutValue self y NotSetValue))
                                                                              (T (PutValue self (CAR y)
                                                                                           NotSetValue
                                                                                           (CADR y])
                                                   (CV [for y in (CDR x) do (COND
                                                                              ((ATOM y)
                                                                               (PutClassValue self y NotSetValue)
                                                                               )
                                                                              (T (PutClassValue self (CAR y)
                                                                                           NotSetValue
                                                                                           (CADR y])
                                                   (printout TTY "Illegal ResetList.." T x T ".. for" -4
                                                          (@ name)
                                                          T)))
```

```
                                    (ControlledErrorset (@ ResetExp)
                                          T))
                                   (T (printout TTY (@ name)
                                                "  is already Reset" T)
                                      (RETURN NIL)))
                                (RETURN T))))
```

## (LOOPSTestAbstract.TestSelf

```
  (Method ((LOOPSTestAbstract TestSelf)
           self TestedLst)                                    (* smL " 7-Apr-86 16:41")
                                                              (* performs the basic TEST for a TestObject but continues even
                        if PreTests fail)
                        (PROG (FMSG (TestingObject self)
                                    PrevRes HELPFLAG (HeaderFlag T))
                                                              (* HeaderFlag is used by: PreTestsSatisfied?, DoTestSelf)
                              (CloseCurrentEnvironment)
                              (PutValue self (QUOTE Tested?)
                                     T
                                     (QUOTE DoneOnce))
                              (COND
                                 ([AND TestedLst (NOT (FMEMB self (CAR TestedLst]
                                  (TCONC TestedLst self)))
                              (COND
                                 ((ValueExists? (SETQ PrevRes (ExaminePreviousTry self TestedLst)))
                                  (PrintIfLev LTMsgLev 7 (printout TTY "Test for " -4 (@ TestDesc)
                                                                   -4 "was" -5 (COND
                                                                                  (PrevRes "Successful")
                                                                                  (T "Unsuccessful"))
                                                                   T))
                                  (RETURN PrevRes)))          (* check pretests)
                              (AND LTBROWSER (_ LTBROWSER BoxNode self))
                                                              (* (COND ((EQ (SETQ Res (PreTestsSatisfied? self TestedLst))
                        T)) (T (* pretests failed. Return) (printout TTY "PreTests failed for" -4 (@ TestDesc) -4 "Nonethless, continuing
                                                              with test - interpret results accordingly" T))))
                              (RETURN (DoTestSelf self TestedLst)))))
```

## (LOOPSTestMeta.BeginLOOPSTest

```
  (Method ((LOOPSTestMeta BeginLOOPSTest)
           self)                                              (* smL " 7-Apr-86 16:48")
                                                              (* TESTS THE KERNEL FEATURES OF LOOPS)
                (PROG (LTKInst LTKClass Temp)                 (* TEST: Create a new class)
                                                              (* Also test that Class is set properly)
                      (DefineClass (QUOTE LTClass)
                            (QUOTE (Object)))
                      (COND
                         ((EQ (SETQ LTKClass (GetClass ($ LTClass)))
                              ($ Class)))
                         (T (printout TTY "Bug: Class of a newly created class not set properly" -5 LTKClass T)
                            ))                                (* TEST: Create an instance of this)
                                                              (* Also test that Class is set properly)
                                                              (* Also tests message passing and Method Inheritance)
                      (SETQ LTKInst (_ ($ LTClass)
                                       New))
                      (COND
                         ((EQ (SETQ LTKClass (Class LTKInst))
                              ($ LTClass)))
                         (T (printout TTY "Bug: Class of an instance of LTKernel not set properly" -5 LTKClass
                                    T)))                      (* TEST: Add CV and IVs to the class and Get from Instance)
                                                              (* Tests inheritance of IV and CVs)
                      (_ ($ LTClass)
                         Add
                         (QUOTE CV)
                         (QUOTE CVTest1)
                         (QUOTE CVal1))
                      (_ ($ LTClass)
                         Add
                         (QUOTE IV)
                         (QUOTE IVTest1)
                         (QUOTE IVal1))
                      (_ ($ LTClass)
                         Add
                         (QUOTE IV)
                         (QUOTE IVTest2)
                         (QUOTE IVal2))
                      (COND
                         [(EQUAL (GetClassValue ($ LTClass)
                                          (QUOTE CVTest1))
                                 (GetClassValue LTKInst (QUOTE CVTest1]
                         (T (printout TTY "Bug: CVs not being inherited properly" T)))
                      (COND
                         [(EQUAL (GetValue ($ LTClass)
                                           (QUOTE IVTest1))
                                 (GetValue LTKInst (QUOTE IVTest1]
                         (T (printout TTY "Bug: IVs not being inherited properly" T)))
                      (PutValue LTKInst (QUOTE IVTest2)
```

```
                        (QUOTE IVal3))
                  (COND
                     ((EQUAL (GetValue ($ LTClass)
                                    (QUOTE IVTest2))
                             (GetValue LTKInst (QUOTE IVTest2)))
                      (printout TTY "Bug: Inherited Values are overriding local values" T)))
                                                          (* TEST: Destroy the instance and its class)
                  (_ LTKInst Destroy)
                  (_ ($ LTClass)
                     Destroy)
                  [SETQ Temp (ERSETQ (GetClass ($ LTClass]
                  (COND
                     [(OR (NULL Temp)
                          (NULL (CAR Temp]
                     (T (printout TTY "Bug: Class Object not destroyed properly: GetClass returns non-NIL"
                               T)))
                  (printout TTY "Kernel Test Completed." T "Following features seem to be OK, unless
                        indicated by an earlier message " T "Creation of a new class" T "Instantiating a
                        class using NEW message" T "Setting of Class links" T "Inheritance of methods" T
                        "Inheritance of IVs and CVs" T "Destruction of a class and its instance" T
                        "Following functions/methods partially tested" T "FUNCTIONS:" .PARA 15 -2
                        (QUOTE ("DefineClass" "GetClass" "Class" "GetClassValue" "GetValue" "PutValue"))
                        T "METHODS:" .PARA 15 -2 (QUOTE ("New" "Add" "Destroy"))
                        T)
                  (RETURN T))))
```

## (LOOPSTestMeta.GetTestCode

```
  (Method ((LOOPSTestMeta GetTestCode)
           self)                                          (* sm: "21-OCT-82 14:07")
                                                          (* returns NIL, as no TestCode yet)
                NIL))
```

## (LOOPSTestMeta.New

```
  (Method ((LOOPSTestMeta New)
           self name)                                     (* smL " 7-Apr-86 16:50")

          (* inherits its Super's New and then adds the instance's name to filecoms and CV Instances)
                                                          (* if name is NIL, generate a name)
                  [COND
                     (name)
                     (T (PROG NIL
                          Again
                              (SETQ name (PACK* (GetClassValue self (QUOTE InstancePrefix))
                                                (add (@ ::UnnamedInstanceCount)
                                                     1)))
                              (COND
                                 (($! name)               (* already object. generate again)
                                  (GO Again]
                  (_Super
                   self New name)))
```

## (LOOPSTestSimple.MakeEditSource

```
  (Method ((LOOPSTestSimple MakeEditSource)
           self)                                          (* sm: "18-Mar-85 13:13")
                                                          (* Collect just those IVs the user should see)
                (for ivDescr in (_Super) when (FMEMB (CAR ivDescr)
                                                     (QUOTE (TestDesc SetUp TestExpr ResetExp AfterTest)))
                   collect ivDescr)))
```

## (LOOPSTestSimple.TEST

```
  (Method ((LOOPSTestSimple TEST)
           self)                                          (* sm: "18-Mar-85 13:23")
                                                          (* Run the simple test. No pretests are checked)
                (_ self TestSelf)))

)

(DEFINEQ
```

## (ATEST

```
  [LAMBDA (Exp Comm)                                      (* sm: "30-NOV-82 16:17")
    Exp])
```

## (BasicTest

```
  [LAMBDA NIL                                             (* smL " 7-Apr-86 16:34")

          (* * Tests the kernel features of LOOPS)

    (PROG (Temp OKList LTInst LTInst2)
          (AND LTLOGFLAG (DRIBBLE (QUOTE LTLOG1)))
          (SETQ LTError NIL)
          (SETQ LTResult T)
```

```
          (DoTest (LTLoadFile (QUOTE LTBCLS))
                 (AND (GetObjectRec (QUOTE LTClass1))
                      (GetObjectRec (QUOTE LTClass3))
                      (type? class ($ LTClass2))
                      (type? class ($ LTClass3)))
                 "Load classes from file" "Classes loaded properly")
          (DoTest T (EQUAL (GetValue ($ LTClass)
                                   (QUOTE GV1))
                           (QUOTE V1))
                 "" "Get Value in class")
          (DoTest (SETQ LTInst (_ ($ LTClass)
                                  New))
                 (type? instance LTInst)
                 "Sending message and using New method to create instance" "Creation of instance")
          (DoTest T (EQ (Class LTInst)
                        ($ LTClass))
                 "" "Checking Class of instance")
          (DoTest T (EQUAL (GetValue LTInst (QUOTE GV1))
                           (QUOTE V1))
                 "" "GetValue inherits in instance")
          (DoTest (PutValue ($ LTClass)
                            (QUOTE PV1)
                            (QUOTE LTClass))
                 (EQUAL (GetValue ($ LTClass)
                                  (QUOTE PV1))
                        (QUOTE LTClass))
                 "PutValue in class" "PutValue in class")
          (DoTest (PutValue LTInst (QUOTE GV3)
                            (QUOTE Inst))
                 (EQUAL (GetValue LTInst (QUOTE GV3))
                        (QUOTE Inst))
                 "PutValue in instance" "GetValue locally in instance")
          (DoTest T (EQUAL (GetValue LTInst (QUOTE GV2))
                           (QUOTE V2))
                 "" "GetValue from active value")
          (DoTest T (EQUAL (GetClassValue ($ LTClass)
                                   (QUOTE GCV1))
                           (QUOTE V1))
                 "" "GetClassValue in class")
          (DoTest (SETQ LTInst2 (_ ($ LTClass2)
                                   New))
                 (EQUAL (_ LTInst2 BasicSS1)
                        21)
                 "" "_Super: Directly Invoked")
          (DoTest (SETQ LTInst2 (_ ($ LTClass3)
                                   New))
                 (EQUAL (_ LTInst2 BasicSS1)
                        21)
                 "" "_Super: Invoked from Super class")
          (DoTest (LTLoadFile (QUOTE LTLOAD))
                 (AND (TestLoadedInstances ($ LD4))
                      (TestLoadedInstances ($ LD1)))
                 "Load instances from file" "Instances loaded properly")
          (printout TTY "Following features tested OK" T)
          [for x in (DREVERSE OKList) do (COND
                                           ((EQUAL x ""))
                                           (T (printout TTY x T]
          (AND LTLOGFLAG (DRIBBLE NIL))                          (* (LISTFILES LTLOG))
          (RETURN LTInst])
```

## (**BeginLoopsTest**

```
  [LAMBDA NIL                                                    (* sm: " 5-Jun-84 10:57")
                                                                 (* begins the test of rest of LOOPS after the basic tests
                                                                 succeeded)
    (PROG (Tlis Inp)

          (* Variables set: Seed -
          list of initial testobjs;HasTest -
          ones which have TestExpr; Tested -
          ones actually tested; Failed -
          those which failed; NotDone -
          those not completed)

          [COND
             ((OR LTError (NOT LTResult))
              (SETQ INP (ASKUSER 60 (QUOTE Y)
                                "The basic tests failed. Do you still want to continue?"
                                (QUOTE ((Y "es
                                           " CONFIRMFLG NIL)
                                        (N "o
                                           " CONFIRMFLG NIL)))
                                T))
              (COND
                 ((EQ INP (QUOTE N))
                  (RETURN NIL]
          (AND LTLOGFLAG (DRIBBLE (QUOTE LTLOG2)))
```

```
          (SETQ Tlis (GenerateTestList))
          (SETQ Seed (CAR Tlis))
          (SETQ HasTest (CDR Tlis))
          (SETQ Tested (RunTest Seed))
          (SETQ Failed (for x in Tested when [AND (NULL (GetValue x (QUOTE Tested?)))
                                                  (NOT (EQ T (GetValue x (QUOTE Tested?)
                                                                         (QUOTE Ignored]
                       collect x))
          (SETQ NotDone (APPEND (LDIFFERENCE HasTest Tested)
                                (for x in Tested when (UnknownValue? (GetValue x (QUOTE Tested?))) collect x)))
          (printout TTY T T T 15 "Summary of LOOPS Test:" T T)
          [AND Failed (PROGN (printout TTY "Following failed:" T)
                             (for x in Failed do (printout TTY 10 (GetValue x (QUOTE TestDesc))
                                                           T]
          [AND NotDone (PROGN (printout TTY T "Following could not be run to completion:" 45 "Reason" T)
                              (for x in NotDone do (printout TTY 5 (GetValue x (QUOTE TestDesc))
                                                             45
                                                             (ReasonNotDone x)
                                                             T]
          (COND
             [(OR Failed NotDone)
              (SETQ LTABROWSER (SETQ LTBROWSER (DisplayTestBrowser (APPEND Failed NotDone)
                                                                   (QUOTE FailedTestBrowser)
                                                                   NIL NIL NIL T]
             (T (printout TTY "Congratulations!! You have a fully tested LOOPS System. Happy LOOPing" T)))
          (AND LTLOGFLAG (DRIBBLE NIL))
          (RETURN (NOT (OR Failed NotDone])
```

## (CleanupTester

```
  [LAMBDA NIL                                                    (* smL " 7-Apr-86 16:53")
                                                                 (* cleansUp by deleting temporary files etc after a Tester run)
    (PROG NIL
          (SETQ NOTLISTEDFILES (DREMOVE (QUOTE LTDUMP)
                                        NOTLISTEDFILES))
          (SETQ NOTCOMPILEDFILES (DREMOVE (QUOTE LTDUMP)
                                          NOTCOMPILEDFILES))
          (DREMOVE (QUOTE LTDUMP)
                 FILELST)                                        (* (for f in MainTesterFiles do (DREMOVE f FILELST)))
          (for x in TempTesterFiles do (DELFILE x))
          (for x in TesterTempObjects do (AND (GetObjectRec x)
                                              (_ (GetObjectRec x)
                                                 Destroy)))
          (RETURN T])
```

## (CloseCurrentEnvironment

```
  [LAMBDA NIL                                                    (* smL " 7-Apr-86 16:35")

          (* * Dummied out, because KB's no longer supported)

    NIL])
```

## (ControlledErrorset

```
  [LAMBDA (exp msgFlg)                                           (* sm: "15-Mar-85 11:10")
                                                                 (* works like an errorset, except if SPECVAR BrkErrorFlg is
                                                                 non-NIL does an unprotected eval)
    (COND
       [BrkErrorFlg (LET ((HELPFLAG (QUOTE BREAK!)))
                         (LIST (EVAL exp]
       (T (ERRORSET exp msgFlg])
```

## (DoTest

```
  [NLAMBDA (Setup Test Msg1 Msg2)                                (* sm: "20-SEP-83 16:36")

          (* Executes Setup, followed by Test. If error, sets LTError to T.
          If Test fails (NIL), sets LTResult to NIL)

    (PROG (Res)
          (SETQ Res (ERRORSET Setup T))
          (COND
             ((NULL Res)
              (SETQ LTError T)
              (printout TTY "Error in setup" -3 Msg1 T "Code:" .PPF Setup T)
              (printout TTY "Not testing" -3 Msg2 T)
              (RETURN NIL)))
          (SETQ OKList (CONS Msg1 OKList))
          (SETQ Res (ERRORSET Test T))
          (COND
             ((NULL Res)
              (SETQ LTError T)
              (printout TTY "Error in test" -3 Msg2 T "Code:" .PPF Test T)
              (RETURN NIL))
             ((NULL (CAR Res))
              (SETQ LTResult NIL)
```

```
            (printout TTY "Bug: in" -3 Msg2 T "Code:" -3 .PPF Test T)
            (RETURN NIL)))
         (SETQ OKList (CONS Msg2 OKList))
         (RETURN T])
```

## (DoTestSelf

```
  [LAMBDA (self TestedLst)                                    (* sm: "18-Mar-85 14:25")
                                                              (* executes the SetUp, TestExpr, and AfterTest of a Test Object)

    (PROG (TestExp)
          (PrintTestHeader self)
          (SETQ TestExp (@ TestExpr))
          [COND
             ((NULL TestExp)
              (printout TTY 5 "No test is currently available for.." -3 (@ TestDesc)
                     T)
              (_@
               Tested?
               (COND
                  (LTAskNoTestAvailable (ASKUSER 15 (QUOTE Y)
                                                (QUOTE ("Indicate if this feature works OK"))
                                                (QUOTE ((Y "es
                                                           " RETURN T EXPLAINSTRING "Yes- the test for this is
                                                           marked as successful")
                                                        (N "o
                                                           " RETURN NIL EXPLAINSTRING "No- the test for this is
                                                           marked as Unsuccessful")
                                                        (U "nknown
                                                           " RETURN NotSetValue EXPLAINSTRING "Unknown- the test
                                                           for this is marked as incomplete")))
                                                NIL NIL (QUOTE (CONFIRMFLG NIL))
                                                NIL))
                  (T (printout TTY "Assuming it is OK" T)
                     T)))
              (RETURN (@ Tested?]
          [COND
             ((NOT (EQ (PerformSetup self)
                       T))
              (AND LTBROWSER (SEND LTBROWSER BoxNode self))
              (RETURN (PerformAltTest self (QUOTE SetUp)
                              NIL TestedLst]
          (PerformTest self)
          (COND
             ((ValueExists? (@ self SetUp Tested?))
              (ControlledErrorset (GetValue self (QUOTE AfterTest))
                     T)))
          (AND LTBROWSER (SEND LTBROWSER BoxNode self))
          (RETURN (@ Tested?])
```

## (EvaluateANDALLTest

```
  [LAMBDA (self Test)                                         (* sm: "15-Mar-85 11:11")
                                                              (* evaluates test expressions with ANDALL for EvaluateTest)
                                                              (* evaluates all Clauses, even if one fails or causes error)

    (PROG (Msg (Res T)
               FailedVal Val Exp)
          [for x in (CDR Test) do (PROGN (SETQ Val (ControlledErrorset (COND
                                                                          ((ATOM x)
                                                                           (SETQ Msg NIL)
                                                                           (SETQ Exp x))
                                                                          ((EQUAL (CAR x)
                                                                                  (QUOTE ATEST))
                                                                           (SETQ Exp (CADR x))
                                                                           [SETQ Msg (COND
                                                                                        ((NULL (CDDR x))
                                                                                         NIL)
                                                                                        (T (CADDR x]
                                                                           Exp)
                                                                          (T (SETQ Msg NIL)
                                                                             (SETQ Exp x)))
                                                                       T))
                                         (COND
                                            ((OR (NULL Val)
                                                 (NULL (CAR Val)))
                                             (SETQ FailedVal Val)
                                             (SETQ FMSG (CONS [LIST Msg Exp (COND
                                                                               ((NULL Val)
                                                                                "*ERROR*")
                                                                               (T (CAR Val]
                                                              FMSG))
                                             (SETQ Res NIL]
          (COND
             ((NOT Res)
              (RETURN FailedVal)))
          (RETURN Val])
```

## (EvaluateANDTest

```
  [LAMBDA (self Test)                                                    (* sm: "15-Mar-85 11:11")
                                                                         (* evaluates test expressions with AND for EvaluateTest)
    (PROG (Msg (Res T)
               Val Exp)
          [for x in (CDR Test) while Res do (PROGN (SETQ Val (ControlledErrorset (COND
                                                                                   ((ATOM x)
                                                                                    (SETQ Msg NIL)
                                                                                    (SETQ Exp x))
                                                                                   ((EQUAL (CAR x)
                                                                                           (QUOTE ATEST))
                                                                                    (SETQ Exp (CADR x))
                                                                                    [SETQ Msg
                                                                                     (COND
                                                                                        ((NULL (CDDR x))
                                                                                         NIL)
                                                                                        (T (CADDR x]
                                                                                    Exp)
                                                                                   (T (SETQ Msg NIL)
                                                                                      (SETQ Exp x)))
                                                                                 T))
                                                   (COND
                                                      ((OR (NULL Val)
                                                           (NULL (CAR Val)))
                                                       (SETQ Res NIL]
          [COND
             ((NOT Res)
              (SETQ FMSG (CONS (LIST Msg Exp (COND
                                               ((NULL Val)
                                                "*ERROR*")
                                               (T (CAR Val]
          (RETURN Val])
```

## (EvaluatePROGTest

```
  [LAMBDA (self Test)                                                    (* sm: "15-Mar-85 11:12")
                                                                         (* evaluates test expressions with PROG for EvaluateTest)
    (PROG (Msg (Res T)
               Val Exp)
          [for x in (CDR Test) while Res do (PROGN (SETQ Val (ControlledErrorset (COND
                                                                                   ((ATOM x)
                                                                                    (SETQ Msg NIL)
                                                                                    (SETQ Exp x))
                                                                                   ((EQUAL (CAR x)
                                                                                           (QUOTE ATEST))
                                                                                    (SETQ Exp (CADR x))
                                                                                    [SETQ Msg
                                                                                     (COND
                                                                                        ((NULL (CDDR x))
                                                                                         NIL)
                                                                                        (T (CADDR x]
                                                                                    Exp)
                                                                                   (T (SETQ Msg NIL)
                                                                                      (SETQ Exp x)))
                                                                                 T))
                                                   (COND
                                                      ((NULL Val)
                                                       (SETQ Res NIL]
          [COND
             ((NOT Res)
              (SETQ FMSG (CONS (LIST Msg Exp (COND
                                               ((NULL Val)
                                                "*ERROR*")
                                               (T (CAR Val]
          (RETURN Val])
```

## (EvaluateTest

```
  [LAMBDA (self Field APFlg)                                             (* sm: "15-Mar-85 11:12")
                                                                         (* evaluates testfield of a TestObj.
                                                                         Returns: NIL -
                                                                         if error; (LIST val) otherwise)

          (* APFlg -
          determines if Exps without AND,ANDALL or PROGN are to be treated as one or the other.
          Default is PROGN, i.e. APFlg=NIL means PROGN will be used)

          (* Sets global FMSG as follows: for each failed test (error or NIL), makes a dotted pair from TestMsg and TestExp.
          If no TestMsg, then makes dotted pair from TestExp and NIL.)
                                                                         (* Expressions such as AND, ATEST etc receive special
                                                                         treatment)
    (PROG ((Test (GetValue self Field))
           (DefType (QUOTE PROGN))
           Res)
          (SETQ FMSG NIL)
          [COND
             (APFlg (SETQ DefType (QUOTE AND]
```

```
          [COND
             ((NULL Test)
              (RETURN (QUOTE (T]
          [COND
             ((ATOM Test)
              (RETURN (ControlledErrorset Test T]
          [COND
             [(MEMBER (CAR Test)
                      (QUOTE (PROGN AND ANDALL]
             (T (SETQ Test (LIST DefType Test]
          [COND
             ((EQUAL (CAR Test)
                      (QUOTE AND))
              (RETURN (EvaluateANDTest self Test)))
             ((EQUAL (CAR Test)
                      (QUOTE ANDALL))
              (RETURN (EvaluateANDALLTest self Test)))
             ((EQUAL (CAR Test)
                      (QUOTE PROGN))
              (RETURN (EvaluatePROGTest self Test]
          (printout TTY "Should not reach here in function: EvaluateTest" T)
          (RETURN NIL])
```

## (ExaminePreviousTry

```
  [LAMBDA (self TestedLst)                                  (* sm: "29-MAR-83 14:32")
                                                            (* checks the result of previous try at running this TestObj)

    (PROG (AltRes)
          [COND
             ((ValueNonNIL? (@ Tested?))
              (RETURN (@ Tested?]
          [SETQ AltRes (CONS (@ Tested?)
                             (for x in (GetValue self (QUOTE AltTest)) collect (GetValue (GetObjectRec x)
                                                                                         (QUOTE Tested?]
          (RETURN (COND
                     ((FMEMB T AltRes)
                      T)
                     ((OR (FMEMB NotSetValue AltRes)
                          (FMEMB (QUOTE U)
                                 AltRes))
                      (QUOTE U))
                     (T NIL])
```

## (GenerateTestList

```
  [LAMBDA NIL                                               (* sm: "18-Mar-85 13:09")

          (* returns Seed.HasTest, where Seed is list of tests with no preconditions and HasTest is list of all tests which are DEFINED)
                                                            (* sets global AllTest)
    (PROG (Seed)
          (SETQ AllTest (for z in (LDIFFERENCE (_ ($ LOOPSTestAbstract)
                                                  AllInstances!)
                                               (_ ($ LOOPSTestPrimitive)
                                                  AllInstances!))
                           when (AND (@ z TestExpr)
                                     (NEQ T (@ z TestExpr)))
                           collect z))
          (for z in AllTest do                              (* Put in back links)
                               (for x in (@ z PreTest) bind (name _ (GetObjectName z))
                                  do (PutValue ($! x)
                                               (QUOTE PreTestOf)
                                               name)))
          (SETQ Seed (for x in AllTest when [AND (NULL (GetValue x (QUOTE PreTest] collect x))
          (RETURN (CONS Seed AllTest])
```

## (InteractiveLoopsTest

```
  [LAMBDA (DontAsk NoBrowserFlg)                            (* sm: "15-Mar-85 16:31")
                                                            (* sets up the TestBrowser and offers to run all tests)
                                                            (* If DontAsk is non-nil it will run the tests)

    (PROG (Seed Res)
          (COND
             ((NULL NoBrowserFlg)
              (SetUpTestBrowser)))
          [SETQ Res (OR DontAsk LTDontAsk (ASKUSER 20 (QUOTE Y)
                                                   "Should I proceed with testing the system? "
                                                   (QUOTE ((Y "es
                                                              " RETURN T)
                                                           (N "o
                                                              " RETURN NIL)))
                                                   NIL NIL (QUOTE (CONFIRMFLG NIL]
          (COND
             (Res (BasicTest)
                  (BeginLoopsTest)))
          (RETURN LTBROWSER])
```

## (MakeTest

```
[LAMBDA (name file)                                          (* sm: "18-Mar-85 13:42")
  (LET [(objName (OR name (PromptRead "Name of the test"]
       (COND
          ((NULL objName)
           (printout PROMPTWINDOW objName " is NIL. Aborted" T)
           NIL)
          (($! objName)
           (printout PROMPTWINDOW objName " already a name. Aborted" T)
           NIL)
          (T (_ (_ ($ LOOPSTestSimple)
                   New objName)
                Edit)
             (AND file (ADDTOFILE objName (QUOTE INSTANCES)
                                  file))
             (AND (MOUSECONFIRM "Should test be run?")
                  (_ ($! objName)
                     TEST])
```

## (PerformSetup

```
[LAMBDA (self)                                               (* sm: "20-SEP-83 16:33")
                                                             (* executes the SetUp of a testobj)
  (PROG (Sval Exp Res FMSG)
        (COND
           ([ValueExists? (SETQ Res (GetValue self (QUOTE SetUp)
                                              (QUOTE Tested?]
            [COND
               ((NULL Res)
                (printout TTY 5 "Error in setting up the test environment." T)
                (PrintFailedExp self (GetValue self (QUOTE SetUp)
                                               (QUOTE FailedExp]
            (RETURN Res)))
        (SETQ Exp (@ SetUp))
        [COND
           ((NULL Exp)
            (PrintIfLev LTMsgLev 2 (printout TTY "Sysnote: No SetUp expression for" -3 (@ name)
                                           T]
        (SETQ Sval (EvaluateTest self (QUOTE SetUp)
                          NIL))
        (PutValue self (QUOTE SetUp)
               (SETQ Res (COND
                            ((NULL Sval)
                             NIL)
                            (T T)))
               (QUOTE Tested?))
        [COND
           ((NULL Res)
            (ERRORSET (@ AfterTest)
                  T)
            (printout TTY 5 "Error in setting up the test environment." T)
            (COND
               (FMSG (printout TTY "[Error in the following:" T)
                     (PutValue self (QUOTE SetUp)
                            FMSG
                            (QUOTE FailedExp))
                     (PrintFailedExp self FMSG]
        (RETURN Res])
```

## (PerformAltTest

```
[LAMBDA (self type default TestedLst)                        (* sm: "20-SEP-83 16:32")
                                                             (* tries AltTest if any. otherwise returns default)
  (PROG (lis)
        (SETQ lis (GetValue self (QUOTE AltTest)))
        (COND
           ((AND (NULL lis)
                 (NOT (AddAltTest self)))
            (printout TTY " Not continuing with test.." T)
            (RETURN default)))
        [COND
           ((NOT (EQ (GetValue self (QUOTE AltTest)
                               (QUOTE Tested?))
                     T))
            (for x in lis do (SEND (GetObjectRec x)
                                   TEST TestedLst))
            (PutValue self (QUOTE AltTest)
                   T
                   (QUOTE Tested?]
        (RETURN (ExaminePreviousTry self])
```

## (PerformTest

```
[LAMBDA (self)                                               (* sm: "20-SEP-83 16:33")
                                                             (* actually tests the TestExpr of self)
                                                             (* Globals: FMSG -
                                                             list of pairs for failed tests -
                                                             (Msgstring.Testexp))
  (PROG (Tval FMSG)
```

```
          (SETQ Tval (EvaluateTest self (QUOTE TestExpr)
                           T))
          (COND
             ((OR (NULL Tval)
                  (NULL (CAR Tval)))
              (printout TTY "Test failed for.." (@ TestDesc)
                     T 10 "Send bug report to LOOPSCORE^.pa" T)
              (COND
                 (FMSG (printout TTY "[Following subtests failed:  " T)
                       (PutValue self (QUOTE TestExpr)
                              FMSG
                              (QUOTE FailedExp))
                       (PrintFailedExp self FMSG)))
              (_@
               Tested?
               (TestErrorBreak self)))
             (T (PrintIfLev LTMsgLev 7 (printout TTY "Test successful!! for.." (@ TestDesc)
                                               T))
                (_@
                 Tested? T)))
          (RETURN (@ Tested?])
```

## (PrintFailedExp

```
  [LAMBDA (self FailedExp)                                          (* sm: "20-SEP-83 16:34")
                                                                    (* prints the FailedExp of the type returned by EvaluateTest)
    [for x in FailedExp do (PROGN (printout TTY 1 "Comment:" 12 (CAR x))
                                  (PrintIfLev LTELev 9 (printout TTY 1 "Code" 12 .PPF (CADR x)))
                                  (PrintIfLev LTELev 8 (printout TTY 1 "Returned:" 12 (CADDR x]
    (printout TTY "]" T)
    FailedExp])
```

## (PrintTestCode

```
  [LAMBDA (self)                                                    (* sm: "20-SEP-83 16:34")
                                                                    (* prints the Test Code fields, if type is TestObj else nothing)
    (PROG (TC)
          (SETQ TC (_ self GetTestCode))
          [COND
             (TC (printout TTY "Test Code for" -4 (ObjectName self)
                         T)
                 (for x in TC do (printout TTY .PPFTL x T]
          (RETURN TC])
```

## (PrintTestHeader

```
  [LAMBDA (self Msg)                                                (* sm: "20-SEP-83 16:34")
                                                                    (* prints TestDesc header if global HeaderFlag is T.
                                                                    Also sets the flag to NIL)
                                                                    (* Msg is optional and "TESTING.." by default)
    (COND
       (HeaderFlag (printout TTY (COND
                                    ((NULL Msg)
                                     "TESTING..")
                                    (T Msg))
                             -4
                             (@ TestDesc)
                             T)
                   (SETQ HeaderFlag NIL)
                   T)
       (T NIL])
```

## (ReasonNotDone

```
  [LAMBDA (self)                                                    (* sm: "29-OCT-82 17:02")
                                                                    (* given a TestObj, tries to generate a reason why test was not
                                                                    performed)
    (SETQ self (GetObjectRec self))
    (COND
       ((NULL (@@ ClassTested?))
        "ClassPreTest failed")
       ((NULL (@ self PreTest Tested?))
        "PreTests failed")
       ((NULL (@ self SetUp Tested?))
        "SetUp caused error")
       (T "Not tried or pretests were indeterminate"])
```

## (RunTest

```
  [LAMBDA (Seed)                                                    (* smL "23-Apr-86 15:01")

          (* * Runs test starting with objects in Seed)

    (PROG ((Tlis (CONS NIL))
           Next Ptr Y (Tested (CONS NIL)))
          (LCONC Tlis (APPEND Seed))
          (SETQ Ptr (CAR Tlis))
```

```
      LOOP
          [COND
             ((NULL Ptr)
              (RETURN (CAR Tested]
          (SETQ Next (GetObjectRec (CAR Ptr)))
          [for x in (GetValue Next (QUOTE SubTest)) eachtime (SETQ Y (GetObjectRec x))
             do (COND
                    ((FMEMB Y (CAR Tlis))
                     NIL)
                    ((Object? Y)
                     (TCONC Tlis Y))
                    (T (printout T x " not an object!" "(A SubTest of " Next ")" T]
          [for x in (GetValue Next (QUOTE PreTestOf)) eachtime (SETQ Y (GetObjectRec x))
             do (COND
                    ((FMEMB Y (CAR Tlis))
                     NIL)
                    ((Object? Y)
                     (TCONC Tlis Y))
                    (T (printout T x " not an object!" "(A PreTestOf " Next ")" T]
          (SETQ Ptr (CDR Ptr))
          [COND
             ((NOT (FMEMB Next (CAR Tested)))
              (TCONC Tested Next)
              (ERSETQ (SEND Next TEST Tested]
          (GO LOOP])
```

## (TestErrorBreak

```
  [LAMBDA (self)                                                    (* sm: "27-OCT-82 16:46")

          (* come here when TestExpr has error/NIL. should return value that Tested? will be set to)

    NIL])
```

## (TestObjectDesc

```
  [LAMBDA (self)                                                    (* sm: "21-OCT-82 13:53")
                                                                    (* returns the value of TestDesc IV if self has it, else "")
    (COND
       ((SEND self HasIV (QUOTE TestDesc))
        (@ TestDesc))
       (T ""])
```

## (FlashTestBrowser

```
  [LAMBDA (self varName newValue propName activeVal type)           (* sm: "15-NOV-82 10:25")
                                                                    (* This is a putFn for flashing the test node in testbrowser)
    [COND
       (LTBROWSER (COND
                      ((AND (NULL newValue)
                            (EQ (GetValue self varName propName)
                                NotSetValue))
                       (ERSETQ (_ LTBROWSER FlashNode self 3]
    (PutLocalState activeVal newValue self varName propName type])
```

## (InformTestBrowser

```
  [LAMBDA (self varName newValue propName activeVal type)           (* sm: "29-MAR-83 15:42")

          (* This is a putFn for informing TestBrowser if testobj node is to be flashed or flipped)

    [COND
       (LTBROWSER (COND
                      ((AND (UnknownValue? (GetValue self varName propName))
                            (ValueExists? newValue))
                       (ERSETQ (DoMethod LTBROWSER (COND
                                                       ((EQ newValue T)
                                                        (QUOTE FlipNode))
                                                       (T (QUOTE FlashNode)))
                                         NIL self 5)))
                      ((AND (UnknownValue? newValue)
                            (EQ (GetValue self varName propName)
                                T))
                       (ERSETQ (_ LTBROWSER FlipNode self]
    (PutLocalState activeVal newValue self varName propName type])
```

## (TestFromFiles

```
  [LAMBDA (fileList BrkErrorFlg)                                    (* sm: "18-Mar-85 12:37")

          (* fileList -
          Listy of test files to be loaded. If Atmic, then no files are loaded and only those already loaded are run)

    (PROG ((SaveFlg LTDontAsk)
           files)
          (SETQ LTDontAsk T)
          (SETQ LTLOADEDREST T)
```

```
          [COND
             ((AND fileList (ATOM fileList)))
             (T [DOFILESLOAD (SETQ files (OR fileList (MENU (create MENU
                                                                    ITEMS _ LOOPSTESTGROUPS
                                                                    TITLE _ "Select File group to test"]
                [COND
                   ((NULL files)
                    (printout TTY "No files specified. Abort or run tests already loaded" T "Type RETURN to
                           abort. Anything else followed by RETURN to continue" T)
                    (COND
                       ((NOT (TTYINREAD TTY))
                        (RETURN "Aborted"]
                (for x in files when (NOT (FMEMB x LTTestFiles)) do (SETQ LTTestFiles (CONS x LTTestFiles]
          (UNWINDPROTECT (PROGN (CNDIR (QUOTE {DSK}))
                                (printout TTY "The tester will temporarily connect you to the local disk" T)
                                (InteractiveLoopsTest LTDontAsk BrkErrorFlg)
                                (CleanupTester))
                 (CNDIR)
                 (SETQ LTDontAsk SaveFlg))
          (RETURN "Done!!"])
```

## (LTLoadFile

```
  [LAMBDA (file)                                          (* sm: "18-Mar-85 16:28")
                                                          (* always loads file. If LTCompIntFlg is T, then source, else
                                                          compiled version)

    (COND
       (LTCompIntFlg                                      (* load the source only)
              (LOAD file))
       (T                                                 (* try to load the DCOM if available, else source)
          (DOFILESLOAD (LIST file])
```

## (LTLoadFile?

```
  [LAMBDA (file)                                          (* sm: "18-Mar-85 16:28")
                                                          (* loads file. (if not already loaded) If LTCompIntFlg is T, then
                                                          source, else compiled version)

    (COND
       (LTCompIntFlg                                      (* load the source only)
              (LOAD? file))
       (T                                                 (* try to load the DCOM if available, else source)
          (DOFILESLOAD (LIST file])

)

(DECLARE: EVAL@COMPILE

(PUTPROPS NotSetValue? MACRO [LAMBDA (val)
                               (EQ val NotSetValue])

(PUTPROPS PrintIfLev MACRO [NLAMBDA (CLEV LEV EXP)        (* executes EXP if LEV>=CLEV)
                            (COND
                               ((GEQ (EVAL LEV)
                                     (EVAL CLEV))
                                (EVAL EXP])

(PUTPROPS UnknownValue? MACRO [LAMBDA (val)
                                (OR (EQ val NotSetValue)
                                    (EQ val (QUOTE U])

(PUTPROPS ValueExists? MACRO [LAMBDA (val)                (* Returns T iff val is not eq NotSetValue)
                              (AND (NOT (EQ val NotSetValue))
                                   (NOT (EQ val (QUOTE U])

(PUTPROPS ValueNonNIL? MACRO [LAMBDA (val)                (* Returns T iff val is NOT NIL and NOT EQ NotSetValue)
                               (AND val (NOT (EQ val (QUOTE U)))
                                    (NOT (EQ val NotSetValue])
)


          (* * LTTestFiles -
          list of test files loaded so far)

(RPAQ? LTBROWSER NIL)

(RPAQ? KBTestsFlg T)

(RPAQ? LTDontAsk NIL)

(RPAQ? LTCompIntFlg T)

(RPAQ? LTLOADEDREST NIL)

(RPAQ? LTLOGFLAG NIL)

(RPAQ? LTError NIL)
```

```
(RPAQ? LTResult T)

(RPAQ? LTTestFiles NIL)

(RPAQQ LTSUBINSTANCES (#. ($& LOOPSTestSimple "MRU0.RUC1.9J8.140")
                          #.
                          ($& LOOPSTestSimple "MRU0.RUC1.9J8.139")
                          #.
                          ($& LOOPSTestSimple "MRU0.RUC1.9J8.136")))

[DEFINST LOOPSTestSimple (DynMixin2 "MRU0.RUC1.9J8.140")
       (name #. ($A DynMixin2 NIL RememberName))
       (TestDesc "Tests the supers of a dynamic mixin")
       (SetUp (SETQ I1 (_ ($ (NamedObject TextItem))
                          New)))
       (TestExpr (EQUAL (_ I1 List (QUOTE Supers))
                        (QUOTE (NamedObject TextItem]

[DEFINST LOOPSTestSimple (SecondTest "MRU0.RUC1.9J8.139")
       (name #. ($A SecondTest NIL RememberName))
       (TestDesc "Canonical name preserved for Dynamic mixins")
       (SetUp (SETQ I1 (_ ($ (NamedObject TextItem))
                          New)))
       (TestExpr (EQ (ClassName ($ (NamedObject TextItem)))
                     (ClassName I1]

[DEFINST LOOPSTestSimple (FirstTest "MRU0.RUC1.9J8.136")
       (name #. ($A FirstTest NIL RememberName))
       (TestDesc "Test Named object iv name.")
       (SetUp (_ ($ NamedObject)
                 New
                 (QUOTE FOO)))
       (TestExpr (EQ (GetValue ($ FOO)
                               (QUOTE name))
                     (QUOTE FOO)))
       (ResetExp (AND ($ FOO)
                      (_ ($ FOO)
                         Destroy]

(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML DoTest)

(ADDTOVAR LAMA )
)

(PUTPROPS LTSUB COPYRIGHT ("Xerox Corporation" 1985 1986))
```

## FUNCTION INDEX



## VARIABLE INDEX



## MACRO INDEX