File created: 12-Sep-2021 15:59:37 **{DSK}<home>larry>medley>sources>DEBUGGER.;2**

changes to: (IL:FUNCTIONS STACK-FRAME-PROPERTIES)

previous date: 11-Sep-2021 12:57:01 {DSK}<home>larry>medley>sources>DEBUGGER.;1

Read Table: XCL

Package: DEBUGGER

Format: XCCS


```
(IL:RPAQQ IL:DEBUGGERCOMS
          ((IL:COMS (IL:VARIABLES IL:*DEBUGGER-MENU*)
                 (IL:ADDVARS (IL:CACHEDMENUS IL:*DEBUGGER-MENU*)
                        (IL:FONTVARS (IL:BACKTRACEFONT IL:TINYFONT T))))
           (IL:COMS (IL:VARIABLES XCL:*DEBUGGER-PROMPT* *IN-THE-DEBUGGER* XCL:*DEBUGGER-ENTRY-POINTS*)
                 (IL:VARIABLES IL:BRKEXP IL:BRKTYPE IL:BRKCOND IL:BRKPOS)
                 (IL:FUNCTIONS XCL:ENTER-DEBUGGER-P)
                 (IL:FUNCTIONS XCL:DEBUGGER EMERGENCY-PANIC-LOOP IL:FIND-DEBUGGER-ENTRY-FRAME
                        PRINT-ENTRY-MESSAGE SIMPLE-REPORT-CONDITION XCL::INTERESTING-FRAME-P))
           (IL:COMS (IL:INITVARS (IL:WBREAK))
                 (IL:VARIABLES XCL:*DEBUGGER-MENU-ITEMS* *DEBUGGER-TERMINAL-TABLE* IL:BREAKREGIONSPEC)
                 (IL:FNS IL:WBREAK)
                 (IL:ADDVARS (IL:WINDOWUSERFORMS (IL:WBREAK T))
                        (IL:ENDOFWINDOWUSERFORMS (IL:WBREAK NIL)))
                 (IL:FUNCTIONS REUSE-CURRENT-WINDOW CREATE-DEBUGGER-WINDOW SET-UP-DEBUGGER-WINDOW
                        CLOSE-DEBUGGER-WINDOW RELEASE-DEBUGGER-WINDOW NEAR-BY-REGION)
                 (IL:FUNCTIONS DEBUGGER-BUTTON-EVENT DEBUGGER-MENU-HELP))
           (IL:COMS (IL:VARIABLES IL:LASTPOS)
                 (IL:COMMANDS "@" "REVERT" "?=" "EVAL" "VALUE" "UB")
                 (IL:FUNCTIONS DEBUGGER-EVAL)
                 (IL:FUNCTIONS FIND-DEBUGGER-STACK-FRAME FIND-NAMED-STACK-POSITION)
                 (IL:FUNCTIONS FIND-ORIGINAL-NAME-AND-DEFINITION STKPTR-CCODE))
           (IL:COMS (IL:INITVARS (IL:AUTOBACKTRACEFLG))
                 (IL:VARS IL:BAKTRACELST)
                 (IL:DECLARE\: IL:DOCOPY IL:DONTEVAL@LOAD (IL:INITVARS (IL:AUTOBACKTRACEFLG NIL)
                                                                 (IL:BACKTRACEFONT)))
                 (IL:DECLARE\: IL:DOEVAL@COMPILE IL:DONTCOPY (IL:RECORDS IL:BKMENUITEM))
                 (IL:VARIABLES IL:*SHORT-BACKTRACE-FILTER* IL:|MaxBkMenuWidth|)
                 (IL:FNS IL:BAKTRACE IL:BAKTRACE1)
                 (IL:COMMANDS "BT" "BT!" "BTV" "BTV!" "DBT" "DBT!")
                 (IL:FUNCTIONS ATTACH-BACKTRACE-MENU REGION-NEXT-TO BACKTRACE-MENU-BUTTONEVENTFN
                        BACKTRACE-ITEM-SELECTED STACK-FRAME-PROPERTIES STACK-FRAME-FETCHFN STACK-FRAME-STOREFN
                        STACK-FRAME-VALUE-COMMAND STACK-FRAME-PROPERTY MAKE-FRAME-INSPECT-WINDOW
                        %RELEASE-STACK-DATUM PRINT-BACKTRACE))
           (IL:COMS (IL:COMMANDS "STOP" "^" "RETURN" "PR" "PR!" "PROCEED" "OK")
                 (IL:FUNCTIONS EXIT-DEBUGGER)
                 (IL:FUNCTIONS INVOKE-ESCAPE-FROM-MENU ESCAPE-FROM-DEBUGGER MENU-FROM-ESCAPE-LIST
                        KEYLIST-FROM-ESCAPE-LIST COLLECT-ACTIVE-ESCAPES))
           (IL:COMS (IL:FUNCTIONS IL:FIND-LEXICAL-ENVIRONMENT)
                 (IL:FNS IL:FIND-STACK-FRAME))
           (IL:PROP (IL:MAKEFILE-ENVIRONMENT IL:FILETYPE)
                  XCL:DEBUGGER)
           (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS (IL:ADDVARS (IL:NLAMA)
                                                                                              (IL:NLAML)
                                                                                              (IL:LAMA
                                                                                                     IL:WBREAK)
                                                                                             ))))
```

```
(DEFVAR IL:*DEBUGGER-MENU* NIL
   "Menu for debugger windows")

(IL:ADDTOVAR IL:CACHEDMENUS IL:*DEBUGGER-MENU*)

(IL:ADDTOVAR IL:FONTVARS (IL:BACKTRACEFONT IL:TINYFONT T))
```

```
(DEFVAR XCL:*DEBUGGER-PROMPT* "(debug)"
   "Prompt used by debugger")
```

```
(DEFVAR *IN-THE-DEBUGGER* NIL
```

;;; globally NIL, rebound in the debugger. Used to prevent stack overflow when there's a bug in the debugger. Values are NIL (not in the debugger), T (in ;;; the debugger), and :READ (reading user input in the debugger).

```
   )
```

```
(DEFPARAMETER XCL:*DEBUGGER-ENTRY-POINTS*
   '(IL:|\\interpret-BREAK1| ERROR CERROR XCL:DEBUG CONDITIONS:INVOKE-DEBUGGER BREAK IL:ERRORX IL:*CATCH*
          IL:CHECK-TYPE-FAIL SI::*UNWIND-PROTECT* IL:\\LISPERROR))
```

```
(DEFVAR IL:BRKEXP)

(DEFVAR IL:BRKTYPE)

(DEFVAR IL:BRKCOND)

(DEFVAR IL:BRKPOS)

(DEFUN XCL:ENTER-DEBUGGER-P (IL:N IL:POS XCL:CONDITION)
   (COND
      ((TYPEP XCL:CONDITION 'XCL:STORAGE-CONDITION)
       T)
      ((NULL IL:HELPFLAG)
       NIL)
      ((EQ IL:HELPFLAG 'IL:BREAK!)
       T)
      ((DO ((IL:POS (IL:STKNTH -1 IL:POS)                          ; start at argument
                    (IL:STKNTH -1 IL:POS IL:POS)))                 ; and go back one at a time
           ((NULL IL:POS)                                          ; hit the top of the stack
            NIL)
         (AND IL:N (DECF IL:N))
         (WHEN (EQ (IL:STKNAME IL:POS)
                   'IL:ERRORSET)
            (CASE (AND (IL:IGEQ (IL:STKNARGS IL:POS)
                                2)
                       (IL:STKARG 2 IL:POS))
               ((NIL)                                              ; NLSETQ case
                  (IL:SETQ IL:PRINTMSG (NULL IL:NLSETQGAG))
                  (IL:RELSTK IL:POS)
                  (RETURN NIL))
               (IL:INTERNAL                                        ; ignore this one
                            )
               (IL:NOBREAK
                  (IL:SETQ IL:PRINTMSG NIL)
                  (IL:RELSTK IL:POS)
                  (RETURN NIL))
               (T (IL:SETQ IL:PRINTMSG T)
                  (IL:RELSTK IL:POS)
                  (RETURN (AND IL:N (< IL:N 0))))))))
      ((OR (EQ IL:HELPFLAG 'IL:BREAK!)
           (AND (IL:FIXP IL:HELPCLOCK)
                (IL:FIXP IL:HELPTIME)
                (IL:IGREATERP (IL:IDIFFERENCE (IL:CLOCK 2)
                                              IL:HELPCLOCK)
                              IL:HELPTIME)))
       T)
      (T NIL)))

(DEFUN XCL:DEBUGGER (&KEY ((:FORM IL:BRKEXP))                      ; form to evaluate at EVAL
                          ((:ENVIRONMENT IL:BRKENV)
                           (IL:FIND-LEXICAL-ENVIRONMENT (IL:STKNTH -1)))
                                                                   ; Interpreter lexical environment in which to evalate it.
                          ((:CONDITION IL:BRKCOND))                ; condition which caused this debugger entry
                          ((:STACK-POSITION IL:BRKPOS))            ; location to start debugging, if not given will default. @ will
                                                                   ; always reset
                          ((:AT IL:BRKFN)
                           NIL XCL::BRKFUN-PROVIDED)               ; where's the breakpoint?

                          )
   (DECLARE (SPECIAL IL:WBREAK IL:AUTOBACKTRACEFLG IL:BRKEXP IL:BRKCOND IL:BRKENV IL:BRKPOS IL:BRKFN))
   ;; main entry to the debugger. BRKEXP, BRKFN, BRKTYPE are compatible with Interlisp's BREAK1 argument.  BRKCOND is the "condition" from
   ;; the error system, while BRKENV is the lexical environment of the break.
   (LET ((XCL::WAS-IN-THE-DEBUGGER *IN-THE-DEBUGGER*)
         (*IN-THE-DEBUGGER* T)                                     ; detect calls to debugger in critical sections.
         ;; rebind variables which are possibly reset by redoing the window
         (IL:\\LINEBUF.OFD IL:\\LINEBUF.OFD)
         (IL:\\TERM.OFD IL:\\TERM.OFD)
         (*STANDARD-INPUT* *STANDARD-INPUT*)
         (*STANDARD-OUTPUT* *STANDARD-OUTPUT*)
         (IL:\\INBQUOTE NIL)
         (IL:\\#DISPLAYLINES IL:\\#DISPLAYLINES)
         (IL:\\CURRENTDISPLAYLINE 0)                               ; to get around problem that pagehiehgt isn't per-stream
         (IL:\\PRIMTERMTABLE IL:\\PRIMTERMTABLE)
         (IL:\\PRIMTERMSA IL:\\PRIMTERMSA)
         (IL:\\INTERRUPTABLE T)
         XCL::DEBUGGER-WINDOW                                      ; window for this break
         (IL:LASTPOS)
         (IL:!VALUE IL:*NOT-YET-EVALUATED*)
         IL:BRKVALUES *EVALHOOK*                                   ; because some of the reporting stuff can be interpreted
         *APPLYHOOK*                                               ; likewise
```

```
 )
(DECLARE (SPECIAL IL:\\#DISPLAYLINES IL:\\CURRENTDISPLAYLINE IL:\\INTERRUPTABLE IL:\\INBQUOTE
                  IL:LASTPOS IL:\\LINEBUF.OFD IL:\\TERM.OFD *STANDARD-INPUT* *STANDARD-OUTPUT*
                  IL:BRKVALUES IL:!VALUE IL:\\PRIMTERMTABLE IL:\\PRIMTERMSA))
(WHEN (EQ XCL::WAS-IN-THE-DEBUGGER T)
      (EMERGENCY-PANIC-LOOP))
(IL:OUTPUT T)
(IL:INPUT T)
(IL:SETTERMTABLE *DEBUGGER-TERMINAL-TABLE*)
(IL:RESETLST
    (IL:|if| IL:BRKPOS
        IL:|then| (IL:SETQ IL:LASTPOS (IL:STKNTH 0 IL:BRKPOS))
      IL:|else| (IL:SETQ IL:BRKPOS (IL:STKNTH 0 (IL:SETQ IL:LASTPOS (IL:FIND-DEBUGGER-ENTRY-FRAME))))
           (IL:RESETSAVE NIL (LIST 'IL:RELSTK IL:BRKPOS)))
    (IL:RESETSAVE NIL (LIST 'IL:RELSTK IL:LASTPOS))
    (COND
       ((AND IL:WBREAK (IL:IMAGESTREAMP IL:\\TERM.OFD)
             (TYPEP IL:BRKCOND 'XCL::CRITICAL-STORAGE-CONDITION))
        (REUSE-CURRENT-WINDOW))
       ((AND IL:WBREAK (IL:IMAGESTREAMP IL:\\TERM.OFD))
        (SETF XCL::DEBUGGER-WINDOW (CREATE-DEBUGGER-WINDOW))
        (IL:RESETSAVE NIL (LIST 'RELEASE-DEBUGGER-WINDOW XCL::DEBUGGER-WINDOW))
        (SET-UP-DEBUGGER-WINDOW XCL::DEBUGGER-WINDOW)))
    ;; clear typin buffer on errors
    (WHEN (TYPEP IL:BRKCOND 'ERROR)
          (IL:CLEARBUF T))
    ;;  on a revert, fix up the stack
    (WHEN (TYPEP IL:BRKCOND 'SI::REVERT)
        (AND (IL:LISTP (IL:STKNAME IL:LASTPOS))
             (IL:LITATOM (SI::REVERT-FUNCTION IL:BRKCOND))
             (IL:SETSTKNAME IL:LASTPOS (SI::REVERT-FUNCTION IL:BRKCOND))))
    ;; Show where we are...
    (COND
       (XCL::BRKFUN-PROVIDED (FORMAT T "In ~S:~&" IL:BRKFN))
       ((TYPEP IL:BRKCOND 'SI::BREAKPOINT)
        (SETF IL:BRKFN (SI::BREAKPOINT-FUNCTION IL:BRKCOND))))
    (PRINT-ENTRY-MESSAGE)
    ;; Automatically backtrace, if necessary
    (AND XCL::DEBUGGER-WINDOW (CASE IL:AUTOBACKTRACEFLG
                                ((IL:ALWAYS! IL:ALWAYS) T)
                                ((NIL) NIL)
                                (OTHERWISE
                                   ;;  only backtrace on errors
                                   (TYPEP IL:BRKCOND 'ERROR)))
         (ATTACH-BACKTRACE-MENU NIL (NOT (IL:FMEMB IL:AUTOBACKTRACEFLG '(IL:BT! IL:ALWAYS!)))))
    ;; Finally, the main debugger loop. This is simply an inferior exec with the appropriate command tables and eval function.
    (CATCH 'DEBUGGER-EXIT
        (LET ((*READ-SUPPRESS* NIL)
              (*IN-THE-DEBUGGER* :READ))
             (XCL:EXEC :TITLE NIL :COMMAND-TABLES (LIST IL:*DEBUGGER-COMMAND-TABLE*
                                                        IL:*EXEC-COMMAND-TABLE*)
                   :ENVIRONMENT IL:BRKENV :PROMPT XCL:*DEBUGGER-PROMPT* :FUNCTION
                   #'(LAMBDA (XCL::INPUT XCL::ENV)
                             (LET ((*IN-THE-DEBUGGER* NIL))
                                  (IL:EVAL-INPUT XCL::INPUT XCL::ENV)))))))
;; Now, determine the appropriate error action:
(ECASE (CAR IL:BRKVALUES)
    ((NIL) (VALUES))
    ((T) (VALUES-LIST (CDR IL:BRKVALUES)))
    ((IL:ERROR!) (IL:ERROR!))
    ((RETURN)
       ;; see RETURN command
       (IL:RETAPPLY (THIRD IL:BRKVALUES)
              'VALUES-LIST
              (LIST (SECOND IL:BRKVALUES))
              T))
    ((:REVERT)
       ;; see REVERT command
       (IL:RETAPPLY (THIRD IL:BRKVALUES)
              (FOURTH IL:BRKVALUES)
              (SECOND IL:BRKVALUES)
              T))))
```

```
(DEFUN EMERGENCY-PANIC-LOOP ()
   (IL:PRIN1 "Call to debugger while in the debugger, entering read-eval-print-loop" T)
   (IL:TERPRI T)
```

```
   (LET ((*READ-SUPPRESS* NIL)
         (*IN-THE-DEBUGGER* NIL))
        (LOOP (IL:PRIN1 "eval:" T)
              (IL:PRINT (IL:EVAL (IL:READ T T))
                     T))))
```

```
(DEFUN IL:FIND-DEBUGGER-ENTRY-FRAME (&OPTIONAL (IL:POS 'XCL:DEBUGGER)
                                               IL:SKIP-FAKE?)
   ;; return initial value of LASTPOS for backtrace; called when entering the debugger and by @ command.
   (IL:BIND IL:NAME IL:INBRK IL:DO (IL:SETQ IL:POS (IL:IF IL:SKIP-FAKE?
                                                          IL:THEN (IL:REALSTKNTH -1 IL:POS NIL IL:POS)
                                                                  ; this will ignore the ones that aren't REALFRAMEP
                                                          IL:ELSE (IL:STKNTH -1 IL:POS IL:POS)))
      IL:REPEATWHILE (OR (IL:FMEMB (IL:SETQ IL:NAME (IL:STKNAME IL:POS))
                                   XCL:*DEBUGGER-ENTRY-POINTS*)
                         (IL:GENSYM? IL:NAME)
                         (AND IL:INBRK (IL:FMEMB IL:NAME '(EVAL IL:EVAL-PROGN)))
                         (IL:SETQ IL:INBRK (EQ IL:NAME (SPECIAL-FORM-P 'IL:BREAK1))))
            ;; this will ignore the things that are generated subfunctions or internal debugger functions, e.g. ERRORX or CL:ERROR etc.

      IL:|finally| (RETURN IL:POS)))
```

```
(DEFUN PRINT-ENTRY-MESSAGE ()
   (OR (IL:NLSETQ (PRINC IL:BRKCOND))                    ; should this go to *ERROR-OUTPUT* or *DEBUG-IO* instead?
       ;; Do something simple if printing the condition breaks...
       (PROGN (PRINC "<Condition ")
              (PRIN1 (TYPE-OF IL:BRKCOND))
              (PRINC " @ ")
              (LET ((*PRINT-BASE* 8))
                   (PRIN1 (IL:\\HILOC IL:BRKCOND))
                   (PRINC ",")
                   (PRIN1 (IL:\\LOLOC IL:BRKCOND)))
              (PRINC ">")
              (TERPRI)
              (PRINC "(problems trying to report it!)"))))
```

```
(DEFUN SIMPLE-REPORT-CONDITION (XCL:CONDITION STREAM)
   ;; produce a short description of the condition, e.g. the condition-type
   (PRINC (TYPECASE XCL:CONDITION
            (XCL:SIMPLE-CONDITION XCL:CONDITION)
            (T (TYPE-OF XCL:CONDITION)))
          STREAM))
```

```
(DEFUN XCL::INTERESTING-FRAME-P (&OPTIONAL XCL::POS XCL::INTERPFLG)
   ;; Value is T if frame should be visible for backtrace, and error retry
   ;; user did write a call to the function at POS, and either INTERPFLG is T, or else the functio call would also exist if compiled
   (LET ((XCL::NAME (IF (IL:STACKP XCL::POS)
                        (IL:STKNAME XCL::POS)
                        XCL::POS)))
        (AND (SYMBOLP XCL::NAME)
             (CASE XCL::NAME
                 (IL:*ENV*                                 ; *ENV* is used by ENVEVAL etc.
                    NIL)
                 (IL:ERRORSET (OR (<= (IL:STKNARGS XCL::POS)
                                      1)
                                  (IL:NEQ (IL:STKARG 2 XCL::POS NIL)
                                          'IL:INTERNAL)))
                 (IL:EVAL (OR (<= (IL:STKNARGS XCL::POS)
                                  1)
                              (IL:NEQ (IL:STKARG 2 XCL::POS NIL)
                                      'XCL::INTERNAL)))
                 (IL:APPLY (OR (<= (IL:STKNARGS XCL::POS)
                                   2)
                               (NOT (IL:STKARG 3 XCL::POS NIL))))
                 (OTHERWISE (OR (NOT (SYMBOLP XCL::NAME))
                                (COND
                                   ((IL:FMEMB XCL::NAME IL:OPENFNS)
                                    XCL::INTERPFLG)
                                   (T (OR (IL:NEQ (IL:CHCON1 XCL::NAME)
                                                  (IL:CHARCODE IL:\\))
                                          (IL:EXPRP XCL::NAME)))))))))))
```

```
(IL:RPAQ? IL:WBREAK )
```

```
(DEFPARAMETER XCL:*DEBUGGER-MENU-ITEMS* '("EVAL" "EDIT" "REVERT" "^" "PROCEED" "OK" "BT" "BT!" "?=")
                                       "Elements of debugger menu")
```

```
(DEFVAR *DEBUGGER-TERMINAL-TABLE* (IL:COPYTERMTABLE NIL)
                                  "Terminal table for use in debugger")


(XCL:DEFGLOBALVAR IL:BREAKREGIONSPEC
   (IL:|create| IL:REGION
          IL:LEFT IL:_ 17
          IL:BOTTOM IL:_ -120
          IL:WIDTH IL:_ 400
          IL:HEIGHT IL:_ 120))

(IL:DEFINEQ

(IL:WBREAK
  (LAMBDA (&OPTIONAL (IL:ON T IL:ONP))                    ; start and stop creating window debugging
         (PROG1 IL:WBREAK
             (AND IL:ONP (IL:SETQ IL:WBREAK IL:ON)))))

)

(IL:ADDTOVAR IL:WINDOWUSERFORMS (IL:WBREAK T))

(IL:ADDTOVAR IL:ENDOFWINDOWUSERFORMS (IL:WBREAK NIL))


(DEFUN REUSE-CURRENT-WINDOW ()
   ;; would want to create new window but won't because of storage error
   (IL:PRINTOUT IL:PROMPTWINDOW T "Ran out of space " "running in process '" (IL:PROCESSPROP (IL:THIS.PROCESS)
                                                                                'IL:NAME)
          "' ")
   (UNLESS (IL:HASTTYWINDOWP)
       ;; if this process doesn't have a tty then it is a background process that ran out of array space. Switch its tty to the PROMPT window because
       ;; it should not have a process associated with it yet.
       (IL:WINDOWPROP IL:PROMPTWINDOW 'IL:PAGEFULLFN NIL)
       ;; clobber PAGEFULLFN so that when user does BT it doesn't just scroll off screen. This changes PROMPTWINDOW but with arrays full they
       ;; shouldn't be in this sysout long anyway.
       (IL:PRINTOUT IL:PROMPTWINDOW "which does not have a TTY window." "Using PROMPTWINDOW as TTY window." T)
       (IL:TTYDISPLAYSTREAM IL:PROMPTWINDOW)))


(DEFUN CREATE-DEBUGGER-WINDOW ()
   (DECLARE (SPECIAL IL:\\TERM.OFD IL:DEFAULTTTYREGION))
   (IL:CREATEW (IF (IL:HASTTYWINDOWP)
                   (NEAR-BY-REGION (IL:WINDOWPROP (IL:WFROMDS (LET ((IL:POS (IL:STKPOS 'XCL:EXEC NIL
                                                                              'XCL:DEBUGGER)))
                                                                    (IL:IF IL:POS
                                                                        IL:THEN (PROG1 (IL:EVALV 'IL:\\TERM.OFD
                                                                                                 IL:POS)
                                                                                       (IL:RELSTK IL:POS))
                                                                         IL:ELSE IL:\\TERM.OFD)))
                                                              'IL:REGION)
                                                    (OR (IL:REGIONP IL:BREAKREGIONSPEC)
                                                        (IL:CREATEREGION 17 -120 400 120)))
                   ;; "in the case of break in a process that doesn't have a real tty yet.  create one"
                   IL:DEFAULTTTYREGION)
          "Debugger Window"))


(DEFUN SET-UP-DEBUGGER-WINDOW (W)
   (IL:WINDOWPROP W 'STACK-POSITION IL:BRKPOS)
   (IL:WINDOWPROP W 'LASTPOS IL:LASTPOS)
   (IL:WINDOWPROP W 'IL:TITLE                             ; this is the wrong title, it doesn't show enough
          (XCL:CONDITION-CASE (WITH-OUTPUT-TO-STRING (S)
                                     (SIMPLE-REPORT-CONDITION IL:BRKCOND S))
                 ;; Do something simple if SIMPLE-REPORT-CONDITION breaks...
                 (ERROR NIL (STRING (TYPE-OF IL:BRKCOND)))))
   (IL:WINDOWPROP W 'IL:BUTTONEVENTFN 'DEBUGGER-BUTTON-EVENT)
   (IL:WINDOWADDPROP W 'IL:CLOSEFN 'CLOSE-DEBUGGER-WINDOW)
   (IL:WINDOWPROP W 'PROCESS (IL:THIS.PROCESS))
   (IL:TTYDISPLAYSTREAM W)
   ;; presumably *DEBUG-IO* points at something that points at the TTYDISPLAYSTREAM so that this affects where *DEBUG-IO* goes

   )


(DEFUN CLOSE-DEBUGGER-WINDOW (W)
   (LET ((PROCESS (IL:WINDOWPROP W 'PROCESS)                ; get window's process

               ))
         (IL:\\CARET.DOWN)
         (COND
```

```
          ((AND (IL:PROCESSP PROCESS)
                (EQ W (IL:WFROMDS (IL:PROCESS.TTY PROCESS))))
           (COND
              ((EQ PROCESS (IL:THIS.PROCESS))                    ; if this is the process, just make sure that the caret is down
               (IL:WINDOWPROP W 'IL:PROCESS NIL)
               (EXIT-DEBUGGER))
              ((IL:PROCESS.EVALV PROCESS '*IN-THE-DEBUGGER*)     ; if the process associated with this window has its tty as this
                                                                 ; window and is tty waiting, flush it.
               (IL:WINDOWPROP W 'IL:PROCESS NIL)
               (IL:PROCESS.APPLY PROCESS 'EXIT-DEBUGGER NIL NIL)
               (IL:BLOCK)))
           ;; otherwise, don't bother, just let the window close

           ))))

(DEFUN RELEASE-DEBUGGER-WINDOW (W)
   (COND
      ((IL:WINDOWP W)
       (IL:RELSTK (IL:WINDOWPROP W 'LASTPOS NIL))
       (IL:WINDOWPROP W 'PROCESS NIL)
       (IL:WINDOWPROP W 'IL:BUTTONEVENTFN 'IL:TOTOPW)
       (IL:\\CARET.DOWN)                                         ; just in case the caret is in the debugger window, this makes
                                                                 ; sure it goes away before closing the window.
       (IL:WINDOWDELPROP W 'IL:CLOSEFN 'CLOSE-DEBUGGER-WINDOW)
       (IL:CLOSEW W))))

(DEFUN NEAR-BY-REGION (REGION REGIONTEMPLATE)
   (LET ((WIDTH (IL:|fetch| (IL:REGION IL:WIDTH) IL:|of| REGIONTEMPLATE))
         (HEIGHT (IL:|fetch| (IL:REGION IL:HEIGHT) IL:|of| REGIONTEMPLATE)))
        (IL:|create| IL:REGION
               IL:LEFT IL:_ (MOD (+ (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| REGION)
                                    (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| REGIONTEMPLATE))
                                 (- IL:\\CURSORDESTWIDTH WIDTH))
               IL:BOTTOM IL:_ (MOD (+ (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| REGION)
                                      (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| REGIONTEMPLATE))
                                   (- IL:\\CURSORDESTHEIGHT HEIGHT))
               IL:WIDTH IL:_ WIDTH
               IL:HEIGHT IL:_ HEIGHT)))

(DEFUN DEBUGGER-BUTTON-EVENT (W &REST IGNORE &AUX SELECTION)
                                                                 ; button event function for debugger windows
   (OR IL:*DEBUGGER-MENU*                                        ; create on demand
       (SETF IL:*DEBUGGER-MENU* (IL:|create| IL:MENU
                                     IL:ITEMS IL:_ XCL:*DEBUGGER-MENU-ITEMS*
                                     IL:WHENHELDFN IL:_ 'DEBUGGER-MENU-HELP)))
   (IL:|if| (IL:LASTMOUSESTATE IL:MIDDLE)
       IL:|then| (IL:CASE-EQUALP (IL:SETQ SELECTION (IL:MENU IL:*DEBUGGER-MENU*))
                       (NIL)
                       ("PROCEED" (INVOKE-ESCAPE-FROM-MENU))
                       ("BT" (ATTACH-BACKTRACE-MENU W T))
                       ("BT!" (ATTACH-BACKTRACE-MENU W NIL))
                       (T (IL:CLEARBUF T)
                          (IL:BKSYSBUF SELECTION)
                          (IL:BKSYSCHARCODE (IL:CHARCODE IL:CR))))))

(DEFUN DEBUGGER-MENU-HELP (COMMAND &REST IGNORE)                 ; whenheld function for debugger menu. Get doc from
                                                                 ; documentation system
   (IL:PRINTOUT IL:PROMPTWINDOW T (IL:IF (IL:LISTP COMMAND)
                                      IL:THEN (OR (THIRD COMMAND)
                                                  (DOCUMENTATION (FIRST COMMAND)
                                                         'IL:COMMANDS)
                                                  (FOURTH (IL:GETHASH (FIRST COMMAND)
                                                                      IL:*DEBUGGER-COMMAND-TABLE*)))
                                    IL:ELSE (OR (DOCUMENTATION COMMAND 'IL:COMMANDS)
                                                (FOURTH (IL:GETHASH COMMAND IL:*DEBUGGER-COMMAND-TABLE*))))))

(DEFVAR IL:LASTPOS)

(XCL:DEFCOMMAND ("@" :DEBUGGER) (&REST IL:PLACE &ENVIRONMENT IL:ENV)
   "Set debugger stack pointer to location specified by PLACE (or default)"
   (FORMAT T "@ = ~S~%" (IL:STKNAME (FIND-DEBUGGER-STACK-FRAME IL:PLACE IL:ENV)))
   (VALUES))

(XCL:DEFCOMMAND ("REVERT" :DEBUGGER) (&REST IL:PLACE &ENVIRONMENT IL:ENV) (DECLARE (SPECIAL IL:BRKVALUES))
   "Unwind to specified frame (or LASTPOS) and enter breakpoint"
   ;; Find the stack frame that the user asked to unwind to , if any:
   (AND IL:PLACE (FIND-DEBUGGER-STACK-FRAME IL:PLACE IL:ENV))
   ;; LASTPOS is now set to the REVERT target.
```

```
   (LET
    ((IL:FN (IL:STKNAME IL:LASTPOS)))
    (WRITE IL:FN :STREAM *DEBUG-IO* :RADIX 10 :BASE 10 :ESCAPE T :CIRCLE NIL :PRETTY NIL :LEVEL 3 :LENGTH 3)
    ;; There's still an odd problem because the frame created by the cl:lambda application has one too many arguments... somehow STKNARGS
    ;; returns 2 when called with a &rest???
    (SETF IL:BRKVALUES
          (LIST ':REVERT (IL:STKARGS IL:LASTPOS)
                (IL:STKNTH 0 IL:LASTPOS)
                `(IL:LAMBDA IL:NOBIND
                   (LET ((IL:POS ',(IL:STKNTH 0 IL:LASTPOS)))
                        (FUNCALL #'(LAMBDA NIL (UNWIND-PROTECT
                                                 (XCL:DEBUGGER :FORM
                                                        '(APPLY ',IL:FN (IL:STKARGS ',IL:FN))
                                                        :CONDITION
                                                        (XCL:MAKE-CONDITION 'SI::REVERT :FUNCTION
                                                              ',IL:FN)
                                                        :STACK-POSITION IL:POS)
                                                 (IL:RELSTK IL:POS)))))))
    (THROW 'DEBUGGER-EXIT NIL)))


(XCL:DEFCOMMAND ("?=" :DEBUGGER) NIL "Show arguments"
   (MULTIPLE-VALUE-BIND (IL:NAME IL:DEFN)
       (FIND-ORIGINAL-NAME-AND-DEFINITION IL:LASTPOS)
     (MULTIPLE-VALUE-BIND (IL:LAMBDA-CAR IL:ARGLIST)
         (SI::NAMED-FUNCTION-WRAPPER-INFO IL:NAME IL:DEFN NIL)
       (LET ((*PRINT-LENGTH* 3)
             (*PRINT-LEVEL* 3)
             (IL:ARGUMENTS (IL:STKARGS IL:LASTPOS)))
         (ECASE IL:LAMBDA-CAR
             ((IL:LAMBDA IL:NLAMBDA) (COND
                                       ((LISTP IL:ARGLIST)
                                        (IL:FOR IL:NAME IL:IN IL:ARGLIST IL:AS IL:VALUE IL:IN IL:ARGUMENTS
                                           IL:DO (FORMAT T "  ~a = ~s~%" IL:NAME IL:VALUE)))
                                       ((OR (EQ IL:LAMBDA-CAR 'IL:LAMBDA)
                                            (LISTP IL:ARGUMENTS))
                                        (IL:FOR IL:VALUE IL:IN IL:ARGUMENTS IL:AS IL:ARGNUM IL:FROM 0
                                           IL:DO (FORMAT T "  Arg ~d = ~s~%" IL:ARGNUM IL:VALUE)))
                                       (T (FORMAT T "  ~a = ~s~%" IL:ARGLIST IL:ARGUMENTS))))
             ((LAMBDA) (MULTIPLE-VALUE-CALL 'SI::PRINT-TRACED-CL-ARGLIST IL:ARGUMENTS (SI::PARSE-CL-ARGLIST
                                                                                        IL:ARGLIST)
                                       0 T))))))
   (VALUES))


(XCL:DEFCOMMAND ("EVAL" :DEBUGGER) (&OPTIONAL (IL:EXPRESSION NIL IL:EXPRESSION-PROVIDED?))
   (DECLARE (SPECIAL IL:BRKENV IL:BRKVALUES))
   "Evaluate expression in debugged context"
   (XCL:CONDITION-CASE (IF IL:EXPRESSION-PROVIDED?
                           (DEBUGGER-EVAL IL:EXPRESSION IL:BRKENV)
                           (VALUES-LIST (CDR (SETF IL:BRKVALUES (CONS T (MULTIPLE-VALUE-LIST (DEBUGGER-EVAL
                                                                                              IL:BRKEXP
                                                                                              IL:BRKENV)))))))
         (SI::DEBUGGER-EVAL-ABORTED (IL:C)
                (VALUES :ABORTED (SI::DEBUGGER-EVAL-ABORTED-CONDITION IL:C)))))


(XCL:DEFCOMMAND ("VALUE" :DEBUGGER :QUIET) NIL "Show value from previous evaluation of debug expression"
   (IF IL:BRKVALUES
       (VALUES-LIST (CDR IL:BRKVALUES))
       (PROGN (FORMAT T "Not yet evaluated~&")
              (VALUES))))


(XCL:DEFCOMMAND ("UB" :DEBUGGER) (&OPTIONAL (IL:FN IL:BRKFN)) "Unbreak function with breakpoint"
   (DECLARE (SPECIAL IL:BRKFN))
   (IL:EVAL (LIST 'XCL:UNBREAK IL:FN)))


(DEFUN DEBUGGER-EVAL (EXP ENV)
   ;; evaluate exp in the context that called the debugger.
   (LET* ((ABORT-CONDITION NIL)
          (ABORTED NIL)
          (VALUES (MULTIPLE-VALUE-LIST (IL:ENVAPPLY #'(LAMBDA (EVAL-FN EXP ENV)
                                                         (XCL:PROCEED-CASE (FUNCALL EVAL-FN EXP ENV)
                                                                (XCL:ABORT (CONDITION)
                                                                       :REPORT "Return to previous
                                                                       debugger" (SETF ABORTED T
                                                                                       ABORT-CONDITION
                                                                                       CONDITION)
                                                                       (VALUES NIL CONDITION))))
                                                       (LIST (COND
                                                               (ENV      ; If there's a lexical environment around, we need to use CL:eval
                                                                         ; to watch for it.
                                                                     'EVAL)
```

```
                                                    (T XCL:*EVAL-FUNCTION*))
                                                EXP ENV)
                                            (IL:STKNTH -1 'XCL:DEBUGGER)
                                            NIL T))))
        (WHEN ABORTED
            (XCL:SIGNAL 'SI::DEBUGGER-EVAL-ABORTED :CONDITION ABORT-CONDITION))
        (VALUES-LIST VALUES)))

(DEFUN FIND-DEBUGGER-STACK-FRAME (PLACE ENV)
    ;; Find stack position denoted by place
    ;; Freely sets LASTPOS to the stack pointer corresponding to PLACE.
    (DECLARE (SPECIAL IL:LASTPOS))
    (IL:|bind| (LSTPOS IL:_ (IL:FIND-DEBUGGER-ENTRY-FRAME)) IL:|while| PLACE
       IL:|do| (IL:CASE-EQUALP (FIRST PLACE)
                    (IL:@                                   ; @ @ foo means leave LASTPOS alone
                          (SETF LSTPOS (IL:STKNTH 0 IL:LASTPOS LSTPOS))
                          (POP PLACE))
                    (=                                      ; @ = FOO means to evaluate FOO
                       (SETF LSTPOS (IL:STKNTH 0 (EVAL (SECOND PLACE)
                                                       ENV))
                             PLACE
                             (CDDR PLACE)))
                    (T (IL:IF (INTEGERP (FIRST PLACE))
                           IL:THEN (IF (MINUSP (FIRST PLACE))
                                       (SETF LSTPOS (IL:STKNTH (FIRST PLACE)
                                                               LSTPOS LSTPOS))
                                       (PROG ((N (FIRST PLACE))
                                              (POS1 (IL:STKNTH -1 'XCL:DEBUGGER)))
                                       ;; Returns the stack position N below LSTPOS by starting at current position and backing up the
                                       ;; control links until it reaches a point N frames before POS.
                                         LP  (COND
                                                ((IL:EQP POS1 LSTPOS)
                                                 (IL:RELSTK POS1)
                                                 (RETURN NIL))
                                                ((> N 0)
                                                 (DECF N)
                                                 (SETF POS1 (IL:STKNTH -1 POS1 POS1))
                                                 (GO LP)))
                                             (SETF LSTPOS (IL:STKNTH -1 'IL:DEBUGGER-LOOP))
                                         LP1                        ; POS1 stays N ahead of POS2.  When POS1 reaches END,
                                                                    ; LSTPOS is the desired position.
                                             (COND
                                                ((NULL POS1)
                                                 (IL:RELSTK LSTPOS)
                                                 (RETURN NIL))
                                                ((IL:EQP POS1 LSTPOS)
                                                 (IL:RELSTK POS1)
                                                 (RETURN LSTPOS)))
                                             (SETF POS1 (IL:STKNTH -1 POS1 POS1)
                                                   LSTPOS
                                                   (IL:STKNTH -1 LSTPOS LSTPOS))
                                             (GO LP1)))
                                   (POP PLACE)
                               IL:ELSE (SETF LSTPOS (FIND-NAMED-STACK-POSITION (FIRST PLACE)
                                                                               NIL
                                                                               (IL:STKNTH -1 LSTPOS LSTPOS)))
                                   (POP PLACE))))
       IL:|finally| (OR LSTPOS (IL:ERROR "not found"))
             (IL:STKNTH 0 LSTPOS IL:LASTPOS)
             ;; smashes LSTPOS into the LASTPOS stack pointer, cannot just reset lastpos to lstpos because of RELSTK etc
             (IL:RELSTK LSTPOS)
             (RETURN IL:LASTPOS)))

(DEFUN FIND-NAMED-STACK-POSITION (FN N LSTPOS &AUX TEM)
   (COND
      ((SETF TEM (IL:STKPOS FN N LSTPOS))
       (IL:RELSTK LSTPOS)
       TEM)
      ((AND IL:DWIMFLG (IL:NEQ IL:NOSPELLFLG T)
            (XCL:DESTRUCTURING-BIND (IGNORE NCXWORD NDBLS &REST LST)
                   (IL:EDITFPAT (IL:CONCAT FN "ŸŸ"))
                   (DECLARE (IGNORE IGNORE))
                   (SETF TEM (IL:SEARCHPDL #'(LAMBDA (FN)
                                                     (IL:SKOR0 FN NCXWORD NDBLS LST))
                                             LSTPOS))))
       (IL:PRIN1 '= T)
       (IL:PRINT (FIRST TEM)
              T)
       (IL:RELSTK LSTPOS)
       (CDR TEM))
      (T (IL:RELSTK LSTPOS)
         (IL:ERROR FN '"not found" T))))
```

```
(DEFUN FIND-ORIGINAL-NAME-AND-DEFINITION (STKPTR)
   (LET ((NAME (IL:STKNAME STKPTR)))
        (COND
           ((SYMBOLP NAME)
            (VALUES NAME (STKPTR-CCODE STKPTR)))
           ((OR (ATOM NAME)
                (NOT (MEMBER (CAR NAME)
                             '(:BROKEN :ADVISED :TRACED))))
            (VALUES NIL (STKPTR-CCODE STKPTR)))
           (T (LET ((SYMBOL (FIRST (IL:MKLIST (SECOND NAME)))))
                   (VALUES SYMBOL (IL:GETD (OR (GET SYMBOL 'IL:ADVISED)
                                               (GET SYMBOL 'IL:BROKEN)
                                               SYMBOL)))))))))


(DEFUN STKPTR-CCODE (STKPTR)
   (IL:MAKE-COMPILED-CLOSURE (IL:FETCH (IL:FX IL:FNHEADER) IL:OF (IL:\\STACKARGPTR STKPTR))))

(IL:RPAQ? IL:AUTOBACKTRACEFLG )

(IL:RPAQQ IL:BAKTRACELST
          ((IL:APPLY (IL:**BREAK** IL:LISPX IL:ERRORSET IL:BREAK1A IL:ERRORSET IL:BREAK1)
                 (IL:**TOP** IL:LISPX IL:ERRORSET IL:EVALQT T)
                 (IL:**EDITOR** IL:LISPX IL:ERRORSET IL:ERRORSET IL:ERRORSET IL:EDITL1 IL:EDITL0 IL:ERRORSET
                        ((IL:ERRORSET IL:ERRORSET IL:ERRORSET IL:EDITL1 IL:EDITL0 IL:ERRORSET)
                         -)
                        IL:EDITL IL:ERRORSET IL:ERRORSET IL:EDITE ((IL:EDITF)
                                                                   (IL:EDITV)
                                                                   (IL:EDITP)
                                                                   -))
                 (IL:**USEREXEC** IL:LISPX IL:ERRORSET IL:ERRORSET IL:USEREXEC))
           (IL:EVAL (IL:**BREAK** IL:LISPX IL:ERRORSET IL:BREAK1A IL:ERRORSET IL:BREAK1)
                 (IL:**TOP** IL:LISPX IL:ERRORSET IL:EVALQT T)
                 (IL:**EDITOR** ((IL:MAPCAR IL:APPLY)
                                 (IL:ERRORSET IL:LISPX))
                        IL:ERRORSET IL:ERRORSET IL:ERRORSET IL:EDITL1 IL:EDITL0 IL:ERRORSET
                        ((IL:ERRORSET IL:ERRORSET IL:ERRORSET IL:EDITL1 IL:EDITL0 IL:ERRORSET)
                         -)
                        IL:EDITL IL:ERRORSET IL:ERRORSET IL:EDITE ((IL:EDITF)
                                                                   (IL:EDITV)
                                                                   (IL:EDITP)
                                                                   -))
                 (IL:**USEREXEC** IL:ERRORSET IL:LISPX IL:ERRORSET IL:ERRORSET IL:USEREXEC))
           (PROGN IL:**BREAK** IL:EVAL ((IL:ERRORSET IL:BREAK1A IL:ERRORSET IL:BREAK1)
                                        (IL:BREAK1)))
           (IL:BLKAPPLY IL:**BREAK** PROGN IL:EVAL IL:ERRORSET IL:BREAK1A IL:ERRORSET IL:BREAK1)
           (IL:*PROG*LAM (NIL IL:EVALA IL:*ENV*)
                 (NIL IL:CLISPBREAK1))))

(IL:DECLARE\: IL:DOCOPY IL:DONTEVAL@LOAD

(IL:RPAQ? IL:AUTOBACKTRACEFLG NIL)

(IL:RPAQ? IL:BACKTRACEFONT )
)

(IL:DECLARE\: IL:DOEVAL@COMPILE IL:DONTCOPY

(IL:DECLARE\: IL:EVAL@COMPILE

(IL:RECORD IL:BKMENUITEM (IL:LABEL IL:BKMENUINFO))
)
)


(DEFVAR IL:*SHORT-BACKTRACE-FILTER* 'XCL::INTERESTING-FRAME-P "Used to determine what BT sees, applied to
                                        each stack frame")


(DEFPARAMETER IL:|MaxBkMenuWidth| 125)

(IL:DEFINEQ

(IL:BAKTRACE
  (IL:LAMBDA (IL:IPOS IL:EPOS IL:SKIPFNS IL:FLAGS IL:FILE)        ; Edited 2-Jun-87 18:26 by amd

    ;; FLAGS is a bit mask telling BACKTRACE what is to be printed.  1 is variables, 2 is eval blips, 4 is everything, 8 suppresses function name and
    ;; 'UNTRACE:' , and 16 uses access links.

    (PROG ((*PRINT-LEVEL* 2)
           (*PRINT-LENGTH* 10)
           (IL:POS (IL:STKNTH 0 IL:IPOS))
           (IL:N 0)
           IL:FN IL:X IL:Y IL:Z (IL:PLVLFILEFLG T))
          (OR IL:FILE (IL:SETQ IL:FILE T))
          (AND (IL:NEQ IL:CLEARSTKLST T)
```

```
            (IL:SETQ IL:CLEARSTKLST (CONS IL:POS IL:CLEARSTKLST)))
    ;; POS is used as a scratch-position.  N is an offset from FROM.  whenever baktrace needs to look at a stkname or stack position, it (re) uses
    ;; POS and computes (STKNTH N IPOS POS).
    IL:LP
        (IL:SETQ IL:FN (IL:STKNAME IL:POS))
    IL:LP1
        (COND
           ((AND IL:SKIPFNS (SOME #'(LAMBDA (IL:SKIPFN)
                                            (FUNCALL IL:SKIPFN IL:FN))
                                  IL:SKIPFNS)))
           (T (COND
                 ((IL:NEQ IL:FLAGS 0)
                  (IL:BACKTRACE (IL:SETQ IL:POS (IL:STKNTH IL:N IL:IPOS IL:POS))
                         IL:POS
                         (IL:LOGOR IL:FLAGS 8)
                         IL:FILE
                         'IL:SHOWPRINT)                          ; Tells BACKTRACE not to print 'UNTRACE:' or the function
                                                                 ; name.
                                                                 ; The SETQ would be unnecessary in spaghetti

                  ))
              (IL:PRIN2 (IL:|if| (EQ IL:FN 'EVAL)
                                 IL:|then| (IL:STKARG 1 IL:POS)
                                 IL:|else| IL:FN)
                        IL:FILE T)                               ; Prints function name.
              (IL:PRIN1 IL:BREAKDELIMITER IL:FILE)))
        (COND
           ((AND (IL:SETQ IL:POS (IL:STKNTH (IL:SETQ IL:N (IL:SUB1 IL:N))
                                            IL:IPOS IL:POS))
                 (NOT (IL:EQP IL:POS IL:EPOS)))
            (GO IL:LP)))
        (IL:RELSTK IL:POS)
        (IL:TERPRI IL:FILE)
        (RETURN))))
```

(IL:BAKTRACE1

```
  (IL:LAMBDA (IL:LST IL:N IL:IPOS IL:POS)                        ; Edited 2-Jun-87 18:28 by amd
    ;; 'MATCHES' LST against stack starting at POS.  Returns NIL or offset corresponding to last functionthat matches
    (PROG (IL:TEM)
      IL:LP
        (COND
           ((NULL IL:LST)
            (RETURN IL:N))
           ((NULL (IL:SETQ IL:POS (IL:STKNTH (IL:SUB1 IL:N)
                                             IL:IPOS IL:POS)))
            (GO IL:OUT))
           ((EQ (IL:SETQ IL:TEM (CAR IL:LST))
                (IL:STKNAME IL:POS))
            ;; make this check first if user WANTS to put the name of a dummy frame in baktracelst, he can.  e.g.  this is necessary in order to
            ;; have the sequence *PROG*LAM EVALA *ENV* disappear
            (IL:SETQ IL:N (IL:SUB1 IL:N)))
           ((IL:DUMMYFRAMEP IL:POS)
            (IL:SETQ IL:N (IL:SUB1 IL:N))
            (GO IL:LP))
           ((EQ IL:TEM 'IL:&)
            (IL:SETQ IL:N (IL:SUB1 IL:N)))
           ((IL:NLISTP IL:TEM)
            (GO IL:OUT))
           ((NULL (SOME #'(LAMBDA (IL:X)
                                  (COND
                                     ((EQ IL:X '-)               ; Optional match
                                      T)
                                     ((IL:SETQ IL:X (IL:BAKTRACE1 IL:X IL:N IL:IPOS IL:POS))
                                      (IL:SETQ IL:N IL:X))))
                        IL:TEM))
            (GO IL:OUT)))
        (IL:SETQ IL:LST (CDR IL:LST))
        (GO IL:LP)
      IL:OUT
        (RETURN NIL))))

)


(XCL:DEFCOMMAND ("BT" :DEBUGGER) NIL "Print backtrace of external frames"
   (PRINT-BACKTRACE :FROM IL:LASTPOS :TEST 'XCL::INTERESTING-FRAME-P)
   (VALUES))


(XCL:DEFCOMMAND ("BT!" :DEBUGGER) NIL "Print backtrace of all frames"
   (PRINT-BACKTRACE :FROM IL:LASTPOS :TEST NIL)
   (VALUES))
```

```
(XCL:DEFCOMMAND ("BTV" :DEBUGGER) NIL "Print backtrace of frames and special bindings"
   (PRINT-BACKTRACE :FROM IL:LASTPOS :PRINT-VARIABLES T)
   (VALUES))


(XCL:DEFCOMMAND ("BTV!" :DEBUGGER) NIL "Print backtrace of all frame information"
   (PRINT-BACKTRACE :FROM IL:LASTPOS :PRINT-VARIABLES T :PRINT-JUNK T)
   (VALUES))


(XCL:DEFCOMMAND ("DBT" :DEBUGGER) NIL (ATTACH-BACKTRACE-MENU NIL T)
   (VALUES))


(XCL:DEFCOMMAND ("DBT!" :DEBUGGER) NIL (ATTACH-BACKTRACE-MENU)
   (VALUES))


(DEFUN ATTACH-BACKTRACE-MENU (&OPTIONAL IL:TTYWINDOW IL:SKIP)
   (DECLARE (SPECIAL IL:\\TERM.OFD IL:BACKTRACEFONT))
   (OR IL:TTYWINDOW (IL:SETQ IL:TTYWINDOW (IL:WFROMDS (IL:TTYDISPLAYSTREAM))))
   (PROG ((IL:POS (IL:STKNTH 0 (IL:GETWINDOWPROP IL:TTYWINDOW 'STACK-POSITION)))
          IL:BTW IL:BKMENU (*PRINT-LEVEL* 2)                    ; for the FORMAT below
          (*PRINT-LENGTH* 3)
          (*PRINT-ESCAPE* T)
          (*PRINT-GENSYM* T)
          (*PRINT-PRETTY* NIL)
          (*PRINT-CIRCLE* NIL)
          (*PRINT-RADIX* 10)
          (*PRINT-ARRAY* NIL)
          (IL:*PRINT-STRUCTURE* NIL)
          (IL:TTYREGION (IL:WINDOWPROP IL:TTYWINDOW 'IL:REGION)))
         (IL:SETQ IL:BKMENU (IL:|create| IL:MENU
                                 IL:ITEMS IL:_ (IL:|for| IL:N IL:|from| 0 IL:|bind| IL:NAME
                                                  IL:|repeatwhile| (IL:SETQ IL:POS (IL:STKNTH -1 IL:POS IL:POS))
                                                  IL:|eachtime| (IL:SETQ IL:NAME (IL:STKNAME IL:POS))
                                                  IL:|when| (OR (NULL IL:SKIP)
                                                                (FUNCALL (COND
                                                                            ((EQ IL:SKIP T)
                                                                             IL:*SHORT-BACKTRACE-FILTER*)
                                                                            (T IL:SKIP))
                                                                         IL:POS))
                                                  IL:|collect| (IL:|create| IL:BKMENUITEM
                                                                       IL:LABEL IL:_
                                                                       (PRIN1-TO-STRING
                                                                        (IL:|if| (EQ IL:NAME 'EVAL)
                                                                            IL:|then| (IL:STKARG 1 IL:POS IL:NAME)
                                                                           IL:|else| IL:NAME))
                                                                       IL:BKMENUINFO IL:_ IL:N))
                                 IL:WHENSELECTEDFN IL:_ 'BACKTRACE-ITEM-SELECTED
                                 IL:MENUOUTLINESIZE IL:_ 0
                                 IL:MENUFONT IL:_ IL:BACKTRACEFONT
                                 IL:MENUCOLUMNS IL:_ 1))
         (COND
            ((IL:SETQ IL:BTW (IL:|for| IL:ATW IL:|in| (IL:ATTACHEDWINDOWS IL:TTYWINDOW)
                              IL:|when| (AND (IL:SETQ IL:BTW (IL:WINDOWPROP IL:ATW 'IL:MENU))
                                             (EQ (IL:|fetch| (IL:MENU IL:WHENSELECTEDFN) IL:|of| (CAR IL:BTW))
                                                 'BACKTRACE-ITEM-SELECTED))
                              IL:|do|                           ; test for an attached window that has a backtrace menu in it.
                                    (RETURN IL:ATW)))           ; if there is already a backtrace window, delete the old menu
                                                                ; from it.
             (IL:DELETEMENU (CAR (IL:WINDOWPROP IL:BTW 'IL:MENU))
                    NIL IL:BTW)
             (IL:WINDOWPROP IL:BTW 'IL:EXTENT NIL)
             (IL:CLEARW IL:BTW))
            ((IL:SETQ IL:BTW (IL:CREATEW (REGION-NEXT-TO (IL:WINDOWPROP IL:TTYWINDOW 'IL:REGION)
                                                         (IL:WIDTHIFWINDOW (IL:IMIN (IL:|fetch| (IL:MENU IL:IMAGEWIDTH)
                                                                                           IL:|of| IL:BKMENU)
                                                                                    IL:|MaxBkMenuWidth|))
                                                         (IL:|fetch| (IL:REGION IL:HEIGHT) IL:|of| IL:TTYREGION)
                                                         :LEFT)))   ; put bt window at left of TTY window unless ttywindow is near
                                                                    ; left edge.
             (IL:ATTACHWINDOW IL:BTW IL:TTYWINDOW (IF (IL:IGREATERP (IL:|fetch| (IL:REGION IL:LEFT)
                                                                                IL:|of| (IL:WINDOWPROP IL:BTW 'IL:REGION))
                                                                    (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| IL:TTYREGION))
                                                      'IL:RIGHT
                                                      'IL:LEFT)
                    NIL
                    'IL:LOCALCLOSE)
             (IL:WINDOWPROP IL:BTW 'IL:PROCESS (IL:WINDOWPROP IL:TTYWINDOW 'IL:PROCESS))
                                                                    ; so that button clicks will switch TTY

            ))
         (IL:ADDMENU IL:BKMENU IL:BTW (IL:|create| IL:POSITION
                                           IL:XCOORD IL:_ 0
                                           IL:YCOORD IL:_ (IL:IDIFFERENCE (IL:WINDOWPROP IL:BTW 'IL:HEIGHT)
```

```
                                                    (IL:|fetch| (IL:MENU IL:IMAGEHEIGHT)
                                                       IL:|of| IL:BKMENU))))
```

;; IL:ADDMENU sets up buttoneventfn for window that we don't want. We want to catch middle button events before the menu handler, so that we ;; can pop up edit/inspect menu for the frame currently selected. So replace the buttoneventfn, and can nuke the cursorin and cursormoved guys, ;; cause don't need them.

```
      (IL:WINDOWPROP IL:BTW 'IL:BUTTONEVENTFN 'BACKTRACE-MENU-BUTTONEVENTFN)
      (IL:WINDOWPROP IL:BTW 'IL:CURSORINFN NIL)
      (IL:WINDOWPROP IL:BTW 'IL:CURSORMOVEDFN NIL)))
```

(DEFUN **REGION-NEXT-TO** (IL:REGION &OPTIONAL IL:WIDTH IL:HEIGHT IL:WHERE IL:TRIED-ONCE?)

;; returns the region that is next to REGION and has a width of WIDTH and a height of HEIGHT. WHERE can be :TOP :BOTTOM :LEFT or :RIGHT. ;; If the region would not fit on the screen it is put on the opposite of WHERE.

```
(PROG ((IL:RLEFT (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| IL:REGION))
       (IL:RBOTTOM (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| IL:REGION))
       (IL:RWIDTH (IL:|fetch| (IL:REGION IL:WIDTH) IL:|of| IL:REGION))
       (IL:RHEIGHT (IL:|fetch| (IL:REGION IL:HEIGHT) IL:|of| IL:REGION))
       IL:NLFT IL:NBTM)
      (OR IL:WIDTH (SETF IL:WIDTH IL:RWIDTH))
      (OR IL:HEIGHT (SETF IL:HEIGHT IL:RHEIGHT))
      (ECASE IL:WHERE
          (:TOP
             (IF (> (+ (SETF IL:NBTM (IL:|fetch| (IL:REGION IL:TOP) IL:|of| IL:REGION))
                       IL:HEIGHT)
                    IL:\\CURSORDESTHEIGHT)
                 (IF IL:TRIED-ONCE?
                     ;; top was tried since bottom wouldn't fit
                     (IL:SETQ IL:NBTM 0)
                     ;; try :BOTTOM
                     (RETURN (REGION-NEXT-TO IL:REGION IL:WIDTH IL:HEIGHT :BOTTOM T)))
                 (INCF IL:NBTM))
             (SETF IL:NLFT IL:RLEFT))
          (:BOTTOM
             (IF (< (SETF IL:NBTM (- IL:RBOTTOM IL:HEIGHT))
                    0)
                 (IF IL:TRIED-ONCE?
                     ;; doesn't fit either place, put it down from top.
                     (SETF IL:NBTM (- IL:\\CURSORDESTHEIGHT IL:HEIGHT))
                     ;; try :TOP
                     (RETURN (REGION-NEXT-TO IL:REGION IL:WIDTH IL:HEIGHT :TOP T))))
             (SETF IL:NLFT IL:RLEFT))
          (:LEFT
             (IF (< (SETF IL:NLFT (- IL:RLEFT IL:WIDTH))
                    0)
                 (IF IL:TRIED-ONCE?
                     ;; doesn't fit either place put at right of screen
                     (IL:SETQ IL:NLFT (- IL:\\CURSORDESTWIDTH IL:WIDTH))
                     ;; try :RIGHT
                     (RETURN (REGION-NEXT-TO IL:REGION IL:WIDTH IL:HEIGHT :RIGHT T))))
             (SETF IL:NBTM (IL:IMAX (+ IL:RBOTTOM (- IL:RHEIGHT IL:HEIGHT))
                                    0)))
          (:RIGHT
             (IF (> (+ (SETF IL:NLFT (+ IL:RLEFT IL:RWIDTH))
                       (IL:SUB1 IL:WIDTH))
                    IL:\\CURSORDESTWIDTH)
                 (IF IL:TRIED-ONCE?
                     ;; doesn't fit either place put at left of screen
                     (SETF IL:NLFT 0)
                     ;; try :LEFT
                     (RETURN (REGION-NEXT-TO IL:REGION IL:WIDTH IL:HEIGHT :LEFT T))))
             (SETF IL:NBTM (IL:IMAX (+ IL:RBOTTOM (- IL:RHEIGHT IL:HEIGHT))
                                    0))))
      (RETURN (IL:CREATEREGION IL:NLFT IL:NBTM IL:WIDTH IL:HEIGHT))))
```

(DEFUN **BACKTRACE-MENU-BUTTONEVENTFN** (WINDOW &AUX (MENU (CAR (IL:LISTP (IL:WINDOWPROP WINDOW 'IL:MENU)))))

```
  (UNLESS (OR (IL:LASTMOUSESTATE IL:UP)
              (NULL MENU))
      (IL:TOTOPW WINDOW)
      (COND
         ((IL:LASTMOUSESTATE IL:MIDDLE)
          ;; look for a selected frame in this menu, and then pop up the editor invoke menu for that frame. don't change the selection, just present
          ;; the edit menu.
          (LET* ((TTYWINDOW (IL:WINDOWPROP WINDOW 'IL:MAINWINDOW))
                 (POS (IL:WINDOWPROP TTYWINDOW 'LASTPOS)))
```

```
                ;; don't have to worry about releasing POS because we only look at it here (nobody here hangs on to it) and we will be around for
                ;; less time than LASTPOS.  The debugger is responsible for releasing LASTPOS.
                (IL:INSPECT/AS/FUNCTION (IF (AND (SYMBOLP (IL:STKNAME POS))
                                                 (IL:GETD (IL:STKNAME POS)))
                                            (IL:STKNAME POS)
                                            'IL:NILL)
                                        POS TTYWINDOW)))
        (T (LET ((SELECTION (IL:MENU.HANDLER MENU (IL:WINDOWPROP WINDOW 'IL:DSP))))
                (WHEN SELECTION
                    (IL:DOSELECTEDITEM MENU (CAR SELECTION)
                           (CDR SELECTION))))))))
```

```
(DEFUN BACKTRACE-ITEM-SELECTED (ITEM MENU BUTTON)
   ;; When a frame name is selected in the backtrace menu, this is the function that gets called.
   (DECLARE (SPECIAL IL:BRKENV))
   (LET* ((FRAMESPECFN (IL:|fetch| (IL:BKMENUITEM IL:BKMENUINFO) IL:|of| ITEM))
                                                                  ; number offset from the break position of the frame
          (TTYWINDOW (IL:WINDOWPROP (IL:WFROMMENU MENU)
                                    'IL:MAINWINDOW))
          (BKPOS (IL:WINDOWPROP TTYWINDOW 'STACK-POSITION))
          (POS (IL:STKNTH (- FRAMESPECFN)
                          BKPOS)))
         (LET ((LP (IL:WINDOWPROP TTYWINDOW 'LASTPOS)))
              (AND LP (IL:STKNTH 0 POS LP)))
         (LET ((OLDITEM (IL:|fetch| (IL:MENU IL:MENUUSERDATA) IL:|of| MENU)))
              ;; change the item selected from OLDITEM to ITEM.  Only do this on left buttons now.  Middle just pops up the edit menu, doesn't
              ;; select. -woz
              (WHEN OLDITEM (IL:MENUDESELECT OLDITEM MENU))
              (IL:MENUSELECT ITEM MENU))
         ;; Change the lexical environment so it is the one in effect as of this frame.
         (IL:PROCESS.EVAL (IL:WINDOWPROP TTYWINDOW 'PROCESS)
                `(SETQ IL:BRKENV ',(IL:FIND-LEXICAL-ENVIRONMENT POS))
                T)
         (LET ((FRAMEWINDOW (XCL:WITH-PROFILE (IL:PROCESS.EVAL (IL:WINDOWPROP TTYWINDOW 'IL:PROCESS)
                                                               `(LET ((PROFILE (XCL:COPY-PROFILE (XCL:FIND-PROFILE
                                                                                                   "READ-PRINT"))))
                                                                  (SETF (XCL::PROFILE-ENTRY-VALUE 'XCL:*EVAL-FUNCTION*
                                                                                                  PROFILE)
                                                                        XCL:*EVAL-FUNCTION*)
                                                                  (XCL:SAVE-PROFILE PROFILE))
                                                               T)
                                             (IL:INSPECTW.CREATE POS #'(LAMBDA (POS)
                                                                         (STACK-FRAME-PROPERTIES POS T))
                                                    'STACK-FRAME-FETCHFN
                                                    'STACK-FRAME-STOREFN NIL 'STACK-FRAME-VALUE-COMMAND NIL
                                                    (FORMAT NIL "~S  Frame" (IL:STKNAME POS))
                                                    NIL
                                                    (MAKE-FRAME-INSPECT-WINDOW TTYWINDOW)
                                                    'STACK-FRAME-PROPERTY))))
              (WHEN (NOT (IL:WINDOWPROP FRAMEWINDOW 'IL:MAINWINDOW))
                  (IL:ATTACHWINDOW FRAMEWINDOW TTYWINDOW (IF (IL:IGREATERP (IL:|fetch| (IL:REGION IL:BOTTOM)
                                                                                 IL:|of| (IL:WINDOWPROP FRAMEWINDOW
                                                                                                        'IL:REGION))
                                                                           (IL:|fetch| (IL:REGION IL:BOTTOM)
                                                                              IL:|of| (IL:WINDOWPROP TTYWINDOW
                                                                                                     'IL:REGION)))
                                                             'IL:TOP
                                                             'IL:BOTTOM)
                                   NIL
                                   'IL:LOCALCLOSE)
                  (IL:WINDOWADDPROP FRAMEWINDOW 'IL:CLOSEFN 'IL:DETACHWINDOW)))))
```

```
(DEFUN STACK-FRAME-PROPERTIES (POS &OPTIONAL LOTS?)
   (LET*
    ((TOTAL-SLOTS (IL:STKNARGS POS T)                             ; STKNARGS takes an extra arg which means to include
                                                                  ; internally bound names as well as those in the basic frame

          )
     (NUM-ARGS (IL:STKNARGS POS))                                 ; number of argument variables
     (FNNAME (IL:STKNAME POS))                                    ; (novalue "no such value")
     (ARGLIST (AND (SYMBOLP FNNAME)
                   (IL:GETD FNNAME)
                   (IL:LISTP (IL:SMARTARGLIST FNNAME T)))))
    `((,FNNAME)
      ,@(IF (EQ FNNAME 'EVAL)                                     ; then open the lexical environment
            (LIST* '("EXPRESSION" 1)
                   (LET ((ENVIRONMENT (IL:STKARG 2 POS)))
                        (WHEN (IL:ENVIRONMENT-P ENVIRONMENT)
                            (MAPCAN #'(LAMBDA (SUB-ENV-NAME SUB-ENV-GET &OPTIONAL (SUB-ENV (FUNCALL SUB-ENV-GET
                                                                                                    ENVIRONMENT)))
                                      (WHEN SUB-ENV
```

```
                                                    (LIST* `(,(STRING-DOWNCASE (SYMBOL-NAME SUB-ENV-NAME)))
                                                           (DO ((PLIST SUB-ENV (CDDR PLIST))
                                                                (PROP-SPECS NIL))
                                                               ((NULL PLIST)
                                                                PROP-SPECS)
                                                             (PUSH `(,(FIRST PLIST)
                                                                     ,SUB-ENV-NAME)
                                                                   PROP-SPECS)))))
                                          '(VARS FUNCTIONS BLOCKS TAGBODIES)
                                          '(IL:ENVIRONMENT-VARS IL:ENVIRONMENT-FUNCTIONS IL:ENVIRONMENT-BLOCKS
                                                  IL:ENVIRONMENT-TAGBODIES)))))
          (IL:BIND MODE ARGNAME IL:|for| I IL:|from| 1 IL:|to| NUM-ARGS
             IL:COLLECT (PROGN (IL:|while| (IL:FMEMB (SETF ARGNAME (POP ARGLIST))
                                                     LAMBDA-LIST-KEYWORDS)
                                IL:|do| (SETF MODE ARGNAME))
                               ;; STKARGNAME returns symbol if bound special
                               (LIST (OR (IL:STKARGNAME I POS)
                                         (COND
                                            ((CHARACTERP ARGNAME)
                                             ;; for special forms might start with #\( or #\{
                                             (SETQ ARGLIST NIL)
                                             (FORMAT NIL "arg ~D" (- I 1)))
                                            ((CASE MODE
                                                 ((NIL &OPTIONAL) ARGNAME)
                                                 (T NIL))
                                             (STRING ARGNAME))
                                            (T (FORMAT NIL "arg ~D" (- I 1)))))
                                     I))))
    ,@(LET ((SLOTS (IL:BIND ARGNAME (NOVALUE IL:_ "no such value") IL:FOR PVAR IL:FROM 0 IL:AS I
                            IL:|from| (1+ NUM-ARGS) IL:|to| TOTAL-SLOTS
                            IL:|when| (AND (IL:NEQ NOVALUE (IL:STKARG I POS NOVALUE))
                                           (OR (SETF ARGNAME (IL:STKARGNAME I POS T))
                                               (AND LOTS? (SETQ ARGNAME (FORMAT NIL "local ~D" PVAR)))))
                            IL:|collect| (LIST ARGNAME I))))
             (AND SLOTS (CONS '("locals")
                              SLOTS))))))
```

```
(DEFUN STACK-FRAME-FETCHFN (FRAMESPEC WHICHSPEC)
   (LET (FN)
        (COND
           ((NULL (CDR WHICHSPEC))
            ;; this is a dummy with no value
            (FIRST WHICHSPEC))
           ((SETQ FN (CDR (ASSOC (CADR WHICHSPEC)
                                 '((VARS . IL:ENVIRONMENT-VARS)
                                   (FUNCTIONS . IL:ENVIRONMENT-FUNCTIONS)
                                   (BLOCKS . IL:ENVIRONMENT-BLOCKS)
                                   (TAGBODIES . IL:ENVIRONMENT-TAGBODIES))
                                 :TEST
                                 'EQ)))                          ; eval frame with lexical environment
            (GETF (FUNCALL FN (IL:STKARG 2 FRAMESPEC))
                  (CAR WHICHSPEC)))
           (T ;; CAR is name, CADR is offset
              (IL:STKARG (SECOND WHICHSPEC)
                     FRAMESPEC)))))

(DEFUN STACK-FRAME-STOREFN (FRAMESPEC WHICHSPEC NEWVALUE)
   (LET (FN)
        (COND
           ((NULL (CDR WHICHSPEC))                               ; no value, can't replace
            NIL)
           ((SETQ FN (CDR (ASSOC (CADR WHICHSPEC)
                                 '((VARS . IL:ENVIRONMENT-VARS)
                                   (FUNCTIONS . IL:ENVIRONMENT-FUNCTIONS)
                                   (BLOCKS . IL:ENVIRONMENT-BLOCKS)
                                   (TAGBODIES . IL:ENVIRONMENT-TAGBODIES))
                                 :TEST
                                 'EQ)))                          ; eval frame with lexical environment
            (LET ((PLIST (FUNCALL FN (IL:STKARG 2 FRAMESPEC))))  ; don't want to depend on setf knowing how to do this; we can
                                                                 ; side effect since fields are always present.
                 (SETF (GETF PLIST (CAR WHICHSPEC))
                       NEWVALUE)))
           (T (IL:SETSTKARG (SECOND WHICHSPEC)
                     FRAMESPEC NEWVALUE)))))

(DEFUN STACK-FRAME-VALUE-COMMAND (VALUE PROP DATUM WINDOW)
   ;; property command function for inspect windows onto stack frames.  Recognizes certain PROP as function names.
   (IF (AND (LISTP PROP)
            (NULL (CDR PROP)))
```

```
       (COND
          ((SYMBOLP VALUE)
           (IL:INSPECT/AS/FUNCTION VALUE DATUM WINDOW))
          ((AND (CONSP VALUE)
                (SYMBOLP (SECOND VALUE)))
           (IL:INSPECT/AS/FUNCTION (SECOND VALUE)
                  DATUM WINDOW)))
       (IL:DEFAULT.INSPECTW.VALUECOMMANDFN VALUE PROP DATUM WINDOW)))
```

```
(DEFUN STACK-FRAME-PROPERTY (PROP DATUM)
   ;; returns the thing to be printed as the value
   (COND
      ((AND (CONSP PROP)
            (NULL (CDR PROP)))                                  ; frame function name
       NIL)
      ((CONSP DATUM)                                            ; multiple frame window
       (SECOND PROP))
      (T (FIRST PROP))))
```

```
(DEFUN MAKE-FRAME-INSPECT-WINDOW (TTYWINDOW)
   (LET (TTYREGION BTWINDOW)
        (COND
           ((SETF BTWINDOW (IL:|for| ATW IL:|in| (IL:ATTACHEDWINDOWS TTYWINDOW) IL:|when| (IL:WINDOWPROP ATW
                                                                                              'FRAME-INSPECT)
                           IL:|do|                              ; test for an attached window that is the frame window.
                                 (%RELEASE-STACK-DATUM ATW)
                                 (RETURN ATW))))
            (T (SETF TTYREGION (IL:WINDOWREGION TTYWINDOW))     ; create frame window and set its fixed properties.
               (SETF BTWINDOW (IL:CREATEW (REGION-NEXT-TO TTYREGION NIL 150 :TOP)
                                 "Back Trace Frame Window"))
               ;; keep size of frame window fixed so that tty portion can grow.  No very elegant way to do this but ...
               (IL:WINDOWPROP BTWINDOW 'FRAME-INSPECT T)
               (IL:WINDOWPROP BTWINDOW 'IL:MAXSIZE '(300 . 150))  ; save backtrace window with window.
               (IL:WINDOWPROP BTWINDOW 'IL:PROCESS (IL:WINDOWPROP TTYWINDOW 'IL:PROCESS))
               (IL:WINDOWADDPROP BTWINDOW 'IL:CLOSEFN #'(LAMBDA (W)
                                                          (%RELEASE-STACK-DATUM W)
                                                          ;; clear storage -- if/why this is necessary is now unclear
                                                          (IL:WINDOWPROP W 'IL:SELECTABLEITEMS NIL))
                                 T)))
        BTWINDOW))
```

```
(DEFUN %RELEASE-STACK-DATUM (W)
   (LET ((ST (IL:WINDOWPROP W 'DATUM)))
        (IF (IL:STACKP ST)
            (IL:RELSTK ST)
            (IF (LISTP ST)
                (MAPC 'IL:RELSTK ST)))))
```

```
(DEFUN PRINT-BACKTRACE (&KEY (FROM 'XCL:PRINT-BACKTRACE)
                             TO TEST PRINT-VARIABLES PRINT-JUNK OUTPUT (LINK :ALINK)
                             &AUX
                             (*PRINT-LEVEL* 2)
                             (*PRINT-LENGTH* 10))
   (IL:BAKTRACE FROM TO (IF TEST
                            (LIST #'(LAMBDA (X)
                                      (NOT (FUNCALL TEST X)))))
         (+ (IF PRINT-VARIABLES
                1
                0)
            (IF PRINT-JUNK
                32
                0)
            8
            (CASE LINK
                (:ALINK 16)
                (T 0)))
         OUTPUT))
```

```
(XCL:DEFCOMMAND ("STOP" :DEBUGGER :QUIET) NIL "Exit this debugger level"
   (IL:SETQ IL:BRKVALUES '(IL:ERROR!))
   (THROW 'DEBUGGER-EXIT NIL))
```

```
(XCL:DEFCOMMAND ("^" :DEBUGGER :QUIET) NIL "Abort out of debugger"
   (IL:SETQ IL:BRKVALUES '(IL:ERROR!))
   (THROW 'DEBUGGER-EXIT NIL))
```

```
(XCL:DEFCOMMAND ("RETURN" :DEBUGGER) (&OPTIONAL (IL:EXPRESSION NIL)
```

```
                                          &ENVIRONMENT IL:ENV) "Return value from debugger"
   (XCL:CONDITION-CASE (PROGN (IL:SETQ IL:BRKVALUES (LIST 'RETURN (MULTIPLE-VALUE-LIST (DEBUGGER-EVAL
                                                                                        IL:EXPRESSION
                                                                                        IL:ENV))
                                                                 (IL:STKNTH 0 IL:BRKPOS)))
                              (THROW 'DEBUGGER-EXIT NIL))
          (SI::DEBUGGER-EVAL-ABORTED (IL:C)
                 (VALUES :ABORTED (SI::DEBUGGER-EVAL-ABORTED-CONDITION IL:C)))))


(XCL:DEFCOMMAND ("PR" :DEBUGGER) (&OPTIONAL IL:NAME-OR-NUMBER) "Select and invoke a proceed case."
   (ESCAPE-FROM-DEBUGGER T IL:NAME-OR-NUMBER)
   (VALUES))


(XCL:DEFCOMMAND ("PR!" :DEBUGGER) (&OPTIONAL IL:NAME-OR-NUMBER) "Select and invoke a proceed case."
   (ESCAPE-FROM-DEBUGGER NIL IL:NAME-OR-NUMBER)
   (VALUES))


(XCL:DEFCOMMAND ("PROCEED" :DEBUGGER) (&OPTIONAL IL:NAME-OR-NUMBER) "Select and invoke a proceed case."
   (ESCAPE-FROM-DEBUGGER T IL:NAME-OR-NUMBER)
   (VALUES))


(XCL:DEFCOMMAND ("OK" :DEBUGGER :QUIET) NIL (DECLARE (SPECIAL IL:BRKENV))
   "Exit/proceed from debugger"
   (XCL:CONDITION-CASE (PROGN (WHEN (TYPEP IL:BRKCOND 'SI::BREAKPOINT)
                                    ;; if at a breakpoint, OK means to eval the expression if necessary  and return
                                    (UNLESS IL:BRKVALUES          ; EQ only if already evaluated
                                        (IL:SETQ IL:BRKVALUES (CONS T (MULTIPLE-VALUE-LIST (DEBUGGER-EVAL
                                                                                             IL:BRKEXP
                                                                                             IL:BRKENV)))))
                                    (THROW 'DEBUGGER-EXIT NIL))
                              (CONDITIONS:CONTINUE)            ; will escape if a proceed case named PROCEED is enabled
                              (ESCAPE-FROM-DEBUGGER)           ; If all else fails, ask the user what to do...

                             )
          (SI::DEBUGGER-EVAL-ABORTED (IL:C)
                 (VALUES :ABORTED (SI::DEBUGGER-EVAL-ABORTED-CONDITION IL:C)))))


(DEFUN EXIT-DEBUGGER ()
   (SETF IL:BRKVALUES '(IL:ERROR!))
   (THROW 'DEBUGGER-EXIT NIL))


(DEFUN INVOKE-ESCAPE-FROM-MENU ()
   (LET ((MENU (MENU-FROM-ESCAPE-LIST (COLLECT-ACTIVE-ESCAPES IL:BRKCOND))))
        (IF MENU
            (LET ((CASE (IL:MENU MENU)))
                 (WHEN CASE (CONDITIONS:INVOKE-RESTART-INTERACTIVELY CASE)))
            (FORMAT *DEBUG-IO* "~&No restarts enabled.~%"))))


(DEFUN ESCAPE-FROM-DEBUGGER (SHADOW? &OPTIONAL NAME-OR-NUMBER)
   (LET* ((ESCAPES (COLLECT-ACTIVE-ESCAPES IL:BRKCOND (NOT SHADOW?)))
          (KEYS (KEYLIST-FROM-ESCAPE-LIST ESCAPES)))
         (IF ESCAPES
             (ETYPECASE NAME-OR-NUMBER
                 (NULL (LET ((ESCAPE (PROGN (IL:ASKUSEREXPLAIN KEYS NIL NIL "
                                                           ")
                                            (IL:ASKUSER NIL NIL "Proceed how? " KEYS T))))
                            (WHEN ESCAPE (CONDITIONS:INVOKE-RESTART-INTERACTIVELY ESCAPE))))
                 ((INTEGER (0)) (LET ((ESCAPE (NTH (1- (THE INTEGER NAME-OR-NUMBER))
                                                   ESCAPES)))
                                     (IF ESCAPE
                                         (CONDITIONS:INVOKE-RESTART-INTERACTIVELY ESCAPE)
                                         (FORMAT *DEBUG-IO* "~&No such restart number: ~D~%" NAME-OR-NUMBER))))
                 (SYMBOL (LET ((ESCAPE (FIND (THE SYMBOL NAME-OR-NUMBER)
                                             ESCAPES :KEY 'CONDITIONS:RESTART-NAME :TEST 'EQ)))
                              (IF ESCAPE
                                  (CONDITIONS:INVOKE-RESTART-INTERACTIVELY ESCAPE)
                                  (FORMAT *DEBUG-IO* "~&No restart named ~S~%" NAME-OR-NUMBER)))))
             (FORMAT *DEBUG-IO* "~&No restarts enabled.~%"))))


(DEFUN MENU-FROM-ESCAPE-LIST (ESCAPES)
   (WHEN ESCAPES
       (IL:|create| IL:MENU
              IL:TITLE IL:_ "Ways to proceed..."
              IL:ITEMS IL:_ (MAPCAR #'(LAMBDA (ESCAPE)
                                              (LIST (PRINC-TO-STRING ESCAPE)
                                                    ESCAPE))
                                    ESCAPES))))
```

```
(DEFUN KEYLIST-FROM-ESCAPE-LIST (ESCAPES)
   (WHEN ESCAPES
       (LET ((KEYLIST (IL:|for| ESC IL:|in| ESCAPES IL:|as| I IL:|from| 1 IL:|bind| MESSAGE IL:|eachtime| (SETF MESSAGE
                                                                                                              (
                                                                                                               PRINC-TO-STRING
                                                                                                               ESC))
                              IL:|collect| `(,I ,MESSAGE IL:NOECHOFLG T IL:EXPLAINSTRING ,(FORMAT NIL "~D~:[~; (~:*~S)~]
                                                                                                    - ~A" I (
                                                                                                     CONDITIONS:RESTART-NAME
                                                                                                           ESC)
                                                                                                    MESSAGE)
                                            IL:CONFIRMFLG T RETURN (PROGN (IL:TERPRI T)
                                                                          ',ESC)))))
             (SETF (CDR (LAST KEYLIST))
                   '(("N" "No - don't proceed " IL:NOECHOFLG T IL:CONFIRMFLG T IL:AUTOCONFIRMFLG T RETURN
                          (IL:TERPRI T))))
             KEYLIST)))

(DEFUN COLLECT-ACTIVE-ESCAPES (CONDITION &OPTIONAL ALL)
   (LET ((ESCAPES (IL:ENVAPPLY XCL:*EVAL-FUNCTION* `((LET ((IL:BRKCOND ',CONDITION))
                                                          (CONDITIONS:COMPUTE-RESTARTS)))
                          (IL:STKNTH -1 'XCL:DEBUGGER)
                          NIL T)))
        (IF (NOT ALL)
            (DELETE-DUPLICATES ESCAPES :FROM-END T :TEST #'(LAMBDA (ESCAPE-1 ESCAPE-2)
                                                                   (AND (CONDITIONS:RESTART-NAME ESCAPE-1)
                                                                        (EQ (CONDITIONS:RESTART-NAME ESCAPE-1)
                                                                            (CONDITIONS:RESTART-NAME ESCAPE-2))))
                    )
            ESCAPES)))

(DEFUN IL:FIND-LEXICAL-ENVIRONMENT (&OPTIONAL (IL:STACKPOS IL:LASTPOS))
    ;; used by DEBUGGER to find a lexical environment to use when evaluating commands
    (DECLARE (SPECIAL IL:LASTPOS))
    (LET ((IL:POS (IL:STKPOS 'EVAL NIL IL:STACKPOS)))
         (AND IL:POS (PROG1 (IL:STKARG 2 IL:POS)
                            (IL:RELSTK IL:POS)))))

(IL:DEFINEQ

(IL:FIND-STACK-FRAME
  (IL:LAMBDA (IL:FRAME-SPEC)                                    (IL:* IL:|lmm| " 7-Nov-86 03:39")
    ;; handle debugger commands like @ which take a frame description. Smash LASTPOS to point at new position.
    (LET ((IL:POS (IL:FIND-DEBUGGER-ENTRY-FRAME T))
          IL:TOKEN)
         (IL:WHILE IL:FRAME-SPEC IL:DO (IL:SETQ IL:POS (IL:CASE-EQUALP (IL:SETQ IL:TOKEN (IL:POP IL:FRAMESPEC))
                                                                       ("@"
                                                                           ; leave LASTPOS alone
                                                                           (IL:STKNTH 0 IL:LASTPOS IL:POS))
                                                                       ("="
                                                                           ; eval
                                                                           (IL:STKNTH 0 (EVAL (IL:POP IL:FRAMESPEC))))
                                                                       (T (COND
                                                                             ((IL:NUMBERP IL:TOKEN)
                                                                              (IL:STKNTH IL:TOKEN IL:POS IL:POS))
                                                                             (T (OR (IL:STKPOS IL:TOKEN NIL
                                                                                               (IL:STKNTH -1 IL:POS IL:POS)
                                                                                               IL:POS)
                                                                                    (IL:ERROR IL:TOKEN '"not found" T)))))
                                                                       )))
         (PROG1 (IL:SETQ IL:LASTPOS (IL:STKNTH 0 IL:POS IL:LASTPOS))
                (IL:RELSTK IL:POS)))))

)

(IL:PUTPROPS XCL:DEBUGGER IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE (XCL:DEFPACKAGE "DEBUGGER"
                                                                                             (:PREFIX-NAME "DBG")
                                                                                             (:NICKNAMES "DBG"))))

(IL:PUTPROPS XCL:DEBUGGER IL:FILETYPE :COMPILE-FILE)

(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA IL:WBREAK)
)

(IL:PUTPROPS XCL:DEBUGGER IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990 1991 2021))
```

FUNCTION INDEX

VARIABLE INDEX

COMMAND INDEX

PROPERTY INDEX

RECORD INDEX