File created: 27-Feb-2024 22:46:53 **{DSK}<home>larry>il>medley>sources>LLSTK.;5**

edit by: lmm

changes to: (RECORDS FX)
(VARS LLSTKCOMS)

previous date: 27-Feb-2024 22:31:40 {DSK}<home>larry>il>medley>sources>LLSTK.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ LLSTKCOMS
       [(DECLARE%: DONTCOPY (EXPORT (RECORDS BF FX FSB STK)
                                    (CONSTANTS \#ALINK.OFFSET)
                                    (GLOBALVARS \PENDINGINTERRUPT \KBDSTACKBASE \MISCSTACKBASE \STACKOVERFLOW)
                                    (MACROS \MYALINK ADDSTACKBASE STACKADDBASE STACKGETBASE STACKGETBASEPTR
                                          STACKPUTBASE STACKPUTBASEPTR \MISCAPPLY*)
                                    (RECORDS STACKP)
                                    (CONSTANTS * STACKTYPES)
                                    (CONSTANTS \StackAreaSize (\InitStackSize (ITIMES \StackAreaSize 12)))
                                    (CONSTANTS \MAXSAFEUSECOUNT)
                                    (RECORDS NAMETABLESLOT FVARSLOT PVARSLOT STKTEMPSLOT BINDMARKSLOT)
                                    (CONSTANTS \NT.IVAR \NT.PVAR \NT.FVAR))
                (RECORDS STACKCELL))
        (COMS                                                   ; For LAMBDA* and Common Lisp functions.
              (FNS \MYARGCOUNT \ARG0 \SETARG0))
        (COMS                                                   ; basic spaghetti for allocating, moving and reclaiming stack
                                                                ; frames
              (FNS \HARDRETURN \DOHARDRETURN \DOGC1 \DOGC \DOHARDRETURN1 \DOSTACKOVERFLOW \MOVEFRAME
                   \INCUSECOUNT \DECUSECOUNT \MAKESTACKP \SMASHLINK \FREESTACKBLOCK \EXTENDSTACK))
        (COMS                                                   ; Some ugly stack-munging ufns
              (FNS \SLOWRETURN \COPY.N.UFN \POP.N.UFN \STORE.N.UFN \UNWIND.UFN))
        (COMS                                                   ; The unwinder
              (FNS SI::NON-LOCAL-GO SI::NON-LOCAL-RETURN SI::NON-LOCAL-RETURN-VALUES SI::INTERNAL-THROW
                   SI::INTERNAL-THROW-VALUES SI::UNWIND-TO-BLIP SI::UNWIND SI::VARIABLE-NAME-IN-FRAME
                   SI::PVAR-VALUE-IN-FRAME)
              (FNS \DISCARDFRAME \SMASHRETURN))
        (COMS                                                   ; parsing stack for gc
              (FNS \GCSCANSTACK))
        (COMS                                                   ; setting up stack from scratch
              (FNS CLEARSTK HARDRESET RELSTK RELSTKP)
              (FNS SETUPSTACK \SETUPSTACK1 \MAKEFRAME \RESETSTACK \RESETSTACK0 \SETUPUSERSTACK \SETUPGUARDBLOCK
                   \MAKEFREEBLOCK \REPEATEDLYEVALQT \DUMMYKEYHANDLER \DUMMYTELERAID \CAUSEINTERRUPT
                   \CONTEXTAPPLY \INTERRUPTFRAME \INTERRUPTED \CODEFORTFRAME \DOMISCAPPLY \DOMISCAPPLY1)
              (INITVARS \SAVED.USER.CONTEXT \NEED.HARDRESET.CLEANUP)
              (GLOBALVARS \SAVED.USER.CONTEXT \NEED.HARDRESET.CLEANUP))
        (COMS                                                   ; HARDRESET recovery code
              (FNS \GATHER-CLEANUP-FORMS \GATHER-CLEANUP-FORMS1 \GATHER-SPECIAL-BINDINGS \HARDRESET-CLEANUP
                   \HARDRESET-CLEANUP1 \HARDRESET-CLEANUP-RUN)
              (VARS *HARDRESET-IGNORE-VARS*)
              (GLOBALVARS *HARDRESET-IGNORE-VARS*))
        (COMS                                                   ; Ufns for RETCALL
              (FNS \DORETCALL \RETCALL))
        (INITVARS (STACKTESTING T))
        (COMS                                                   ; Stack overflow handler
              (FNS \DOSTACKFULLINTERRUPT STACK.FULL.WARNING \CLEANUP.STACKFULL)
              (INITVARS (\PENDINGINTERRUPT)
                     (\STACKOVERFLOW)
                     (AUTOHARDRESETFLG T))
              (ADDVARS (RESETFORMS (SETQ \STACKOVERFLOW)))
              (GLOBALVARS AUTOHARDRESETFLG))
        (DECLARE%: DONTCOPY
               (ADDVARS [INEWCOMS (FNS SETUPSTACK \SETUPSTACK1 \SETUPGUARDBLOCK \MAKEFREEBLOCK)
                                (ALLOCAL (ADDVARS (LOCKEDFNS \RESETSTACK0 \MAKEFRAME \SETUPSTACK1 \MAKEFREEBLOCK
                                                             \FAULTHANDLER \KEYHANDLER \DUMMYKEYHANDLER \DOTELERAID
                                                             \DUMMYTELERAID \DOHARDRETURN \DOGC \CAUSEINTERRUPT
                                                             \INTERRUPTFRAME \CODEFORTFRAME \DOSTACKOVERFLOW
                                                             \UNLOCKPAGES \DOMISCAPPLY)
                                                 (LOCKEDVARS \InterfacePage \DEFSPACE \STACKSPACE \KBDSTACKBASE
                                                        \MISCSTACKBASE \SAVED.USER.CONTEXT \RUNNING.PROCESS
                                                        \NEED.HARDRESET.CLEANUP]
                                        (EXPANDMACROFNS ADDSTACKBASE STACKADDBASE))
               EVAL@COMPILE
               (ADDVARS (DONTCOMPILEFNS SETUPSTACK)))
        (LOCALVARS . T)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                               (NLAML)
                                                                               (LAMA SI::INTERNAL-THROW-VALUES
                                                                                     SI::INTERNAL-THROW
                                                                                     SI::NON-LOCAL-RETURN-VALUES
                                                                                     SI::NON-LOCAL-RETURN])

(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

[ACCESSFNS BF ((BFBLOCK (ADDSTACKBASE DATUM)))                    ; basic frame pointer
       (BLOCKRECORD BFBLOCK ((FLAGS BITS 3)
                             (NIL BITS 3)
                             (RESIDUAL FLAG)                      ; true if this is not a full BF
                             (PADDING BITS 1)
                             (USECNT BITS 8)
                             (IVAR WORD)))
       (TYPE? (IEQ (fetch (BF FLAGS) of DATUM)
                   \STK.BF))
       (ACCESSFNS BF ((NARGS (IDIFFERENCE (FOLDLO (IDIFFERENCE DATUM (fetch (BF IVAR) of DATUM))
                                                  WORDSPERCELL)
                                          (fetch (BF PADDING) of DATUM)))
                      [SIZE (IPLUS 2 (IDIFFERENCE DATUM (fetch (BF IVAR) of DATUM]
                      (CHECKED (AND (type? BF DATUM)
                                    (for I from (fetch (BF IVAR) of DATUM) to (IDIFFERENCE DATUM 2) by 2
                                       always (IEQ \STK.NOTFLAG (fetch (BF FLAGS) of I]

[ACCESSFNS FX ((FXBLOCK (ADDSTACKBASE DATUM)))                    ; frame extension index
       (BLOCKRECORD FXBLOCK ((FLAGS BITS 3)                       ; = \STK.FX
                             (FAST FLAG)
                             (NIL FLAG)
                             (INCALL FLAG)                        ; set when fncall microcode has to punt
                             (VALIDNAMETABLE FLAG)                ; if on, NAMETABLE field is filled in.  If off, is same as
                                                                  ; FNHEADER
                             (NOPUSH FLAG)                        ; when returning to this frame, don't push a value.  Set by
                                                                  ; interrupt code
                             (USECNT BITS 8)
                             (%#ALINK WORD)                       ; low bit is SLOWP
                             (FNHEADER FULLXPOINTER)
                             (NEXTBLOCK WORD)
                             (PC WORD)
                             (NAMETABLE# FULLXPOINTER)
                             (%#BLINK WORD)
                             (%#CLINK WORD)))
       (BLOCKRECORD FXBLOCK ((FLAGBYTE BYTE)
                             (NIL BYTE)
                             (NIL BITS 15)                        ; most of the bits of #ALINK
                             (SLOWP FLAG)                         ; if on, then BLINK and CLINK fields are valid.  If off, they are
                                                                  ; implicit
                             (NIL FULLXPOINTER)                   ; FNHEADER
                             (NIL WORD)                           ; NEXSTBLOCK
                             (NIL WORD)                           ; PC
                             (NAMETABHI WORD)
                             (NAMETABLO WORD)))
       (TYPE? (IEQ (fetch (FX FLAGS) of DATUM)
                   \STK.FX))
       (ACCESSFNS FX ((NAMETABLE (COND
                                    ((fetch (FX VALIDNAMETABLE) of DATUM)
                                     (fetch (FX NAMETABLE#) of DATUM))
                                    (T (fetch (FX FNHEADER) of DATUM)))
                                 (PROGN (replace (FX FAST) of DATUM with NIL)
                                        (replace (FX NAMETABLE#) of DATUM with NEWVALUE)
                                        (replace (FX VALIDNAMETABLE) of DATUM with T)))
                      (FRAMENAME (fetch (FNHEADER FRAMENAME) of (fetch (FX NAMETABLE) of DATUM)))
                      (INVALIDP (EQ DATUM 0))                     ; true when A/CLink points at nobody, i.e. FX is bottom of stack
                      [FASTP (NOT (fetch (FX SLOWP) of DATUM))
                             (PROGN (CHECK (NULL NEWVALUE))
                                    (COND
                                       ((fetch (FX FASTP) of DATUM)
                                        (replace (FX %#BLINK) of DATUM with (fetch (FX DUMMYBF) of DATUM))
                                        (replace (FX %#CLINK) of DATUM with (fetch (FX %#ALINK) of DATUM))
                                        (replace (FX SLOWP) of DATUM with T]
                      [BLINK (COND
                                ((fetch (FX FASTP) of DATUM)
                                 (fetch (FX DUMMYBF) of DATUM))
                                (T (fetch (FX %#BLINK) of DATUM)))
                             (PROGN (replace (FX %#BLINK) of DATUM with NEWVALUE)
                                    (COND
                                       ((fetch (FX FASTP) of DATUM)
                                        (replace (FX %#CLINK) of DATUM with (fetch (FX %#ALINK) of DATUM))
                                        (replace (FX SLOWP) of DATUM with T]
                      [CLINK (IDIFFERENCE (COND
                                             ((fetch (FX FASTP) of DATUM)
                                              (fetch (FX %#ALINK) of DATUM))
                                             (T (fetch (FX %#CLINK) of DATUM)))
                                          \#ALINK.OFFSET)
                             (PROGN (replace (FX %#CLINK) of DATUM with (IPLUS NEWVALUE \#ALINK.OFFSET))
                                    (COND
                                       ((fetch (FX FASTP) of DATUM)
                                        (replace (FX %#BLINK) of DATUM with (fetch (FX DUMMYBF) of DATUM))
                                        (replace (FX SLOWP) of DATUM with T]
                      [ALINK (IDIFFERENCE (FLOOR (fetch (FX %#ALINK) of DATUM)
                                                 WORDSPERCELL)
```

```
                                \#ALINK.OFFSET)
                          (PROGN [COND
                                 ((fetch (FX FASTP) of DATUM)
                                  (replace (FX %#BLINK) of DATUM with (fetch (FX DUMMYBF) of DATUM))
                                  (replace (FX %#CLINK) of DATUM with (fetch (FX %#ALINK) of DATUM]
                                 (replace (FX %#ALINK) of DATUM with (IPLUS NEWVALUE \#ALINK.OFFSET
                                                                            (SUB1 WORDSPERCELL]
                    [ACLINK (SHOULDNT)
                          (PROGN [COND
                                 ((fetch (FX FASTP) of DATUM)
                                  (replace (FX %#BLINK) of DATUM with (fetch (FX DUMMYBF) of DATUM]
                                 (replace (FX %#CLINK) of DATUM with (IPLUS NEWVALUE \#ALINK.OFFSET))
                                 (replace (FX %#ALINK) of DATUM with (IPLUS NEWVALUE \#ALINK.OFFSET
                                                                            (SUB1 WORDSPERCELL]
                                                                    ; replaces A & C Links at once more efficiently than separately
                    (DUMMYBF (IDIFFERENCE DATUM WORDSPERCELL))

                    ;; This is either an actual BF or "residual" BF that provides enough BF to find its IVAR slot.  This means that when a FX is
                    ;; copied, the cell preceding the FX is copied too.

                    (IVAR (fetch (BF IVAR) of (fetch (FX DUMMYBF) of DATUM)))
                    [CHECKED (AND (type? FX DATUM)
                                  (OR (IEQ (fetch (FX DUMMYBF) of DATUM)
                                           (fetch (FX BLINK) of DATUM))
                                      (AND (fetch (BF RESIDUAL) of (fetch (FX DUMMYBF) of DATUM))
                                           (IEQ (fetch (BF IVAR) of (fetch (FX DUMMYBF) of DATUM))
                                                (fetch (BF IVAR) of (fetch (FX BLINK) of DATUM]
                    (FIRSTPVAR (IPLUS DATUM (fetch (FX FXSIZE) of T)))
                                                                    ; stack offset of PVAR0
                    (FXSIZE (PROGN 10))                             ; fixed overhead from flags thru clink
                    (PADDING (PROGN 4))                             ; doublecell of garbage for microcode use
                    (FIRSTTEMP (IPLUS (fetch (FX FIRSTPVAR) of DATUM)
                                      (fetch (FX NPVARWORDS) of DATUM)
                                      (fetch (FX PADDING) of DATUM)))
                                                                    ; note that NPVARWORDS is obtained from the FNHEADER
                    (SIZE (IDIFFERENCE (fetch (FX NEXTBLOCK) of DATUM)
                                       DATUM]

(ACCESSFNS FSB
       ;; FREE STACK BLOCK --
       ;;  A piece of stack space that's free.
       ;; The first word contains 120000Q
       ;; The 2nd word is the size of the block, in words.
       ((FSBBLOCK (ADDSTACKBASE DATUM))
        (CHECKED (IEQ (fetch (FSB FLAGWORD) of DATUM)
                      \STK.FSB.WORD)))
       (BLOCKRECORD FSBBLOCK ((FLAGS BITS 3)
                              (DUMMY BITS 13)
                              (SIZE WORD)))
       (BLOCKRECORD FSBBLOCK ((FLAGWORD WORD)
                              (SIZE WORD)))                         ; free stack block
       (TYPE? (IEQ (fetch (FSB FLAGS) of DATUM)
                   \STK.FSB)))

[ACCESSFNS STK ((STKBLOCK (ADDSTACKBASE DATUM)))                    ; unspecified stack block
       (BLOCKRECORD STKBLOCK ((FLAGS BITS 3)))
       (BLOCKRECORD STKBLOCK ((FLAGWORD WORD]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \#ALINK.OFFSET 10)

(CONSTANTS \#ALINK.OFFSET)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \PENDINGINTERRUPT \KBDSTACKBASE \MISCSTACKBASE \STACKOVERFLOW)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \MYALINK DMACRO (NIL ((OPCODES MYALINK))))

(PUTPROPS ADDSTACKBASE DMACRO (= . STACKADDBASE))

(PUTPROPS STACKADDBASE DMACRO ((N)
                               (VAG2 \STACKHI N)))

(PUTPROPS STACKGETBASE DMACRO ((N)
                               (\GETBASE (STACKADDBASE N)
                                      0)))

(PUTPROPS STACKGETBASEPTR DMACRO ((N)
                                  (\GETBASEPTR (STACKADDBASE N)
```

```
                                        0)))

(PUTPROPS STACKPUTBASE DMACRO ((N V)
                                (\PUTBASE (STACKADDBASE N)
                                       0 V)))

(PUTPROPS STACKPUTBASEPTR DMACRO ((N V)
                                  (\PUTBASEPTR (STACKADDBASE N)
                                         0 V)))

(PUTPROPS \MISCAPPLY* MACRO ((FN ARG1 ARG2)
                             (UNINTERRUPTABLY
                                 (replace (IFPAGE MISCSTACKFN) of \InterfacePage with FN)
                                 (replace (IFPAGE MISCSTACKARG1) of \InterfacePage with ARG1)
                                 (replace (IFPAGE MISCSTACKARG2) of \InterfacePage with ARG2)
                                 (\CONTEXTSWITCH \MiscFXP)
                                 (fetch (IFPAGE MISCSTACKRESULT) of \InterfacePage))))
)

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD STACKP ((STACKP0 WORD)
                     (EDFXP WORD))
       (BLOCKRECORD STACKP ((STACKPOINTER FULLXPOINTER)))
       (TYPE? (STACKP DATUM)))
)

(RPAQQ STACKTYPES (\STK.GUARD \STK.FX \STK.BF \STK.NOTFLAG \STK.FSB \STK.FLAGS.SHIFT (\STK.FSB.WORD
                                                                                      (LLSH \STK.FSB
                                                                                            \STK.FLAGS.SHIFT))
                          (\STK.GUARD.WORD (LLSH \STK.GUARD \STK.FLAGS.SHIFT))
                          (\STK.BF.WORD (LLSH \STK.BF \STK.FLAGS.SHIFT))))

(DECLARE%: EVAL@COMPILE

(RPAQQ \STK.GUARD 7)

(RPAQQ \STK.FX 6)

(RPAQQ \STK.BF 4)

(RPAQQ \STK.NOTFLAG 0)

(RPAQQ \STK.FSB 5)

(RPAQQ \STK.FLAGS.SHIFT 13)

(RPAQ \STK.FSB.WORD (LLSH \STK.FSB \STK.FLAGS.SHIFT))

(RPAQ \STK.GUARD.WORD (LLSH \STK.GUARD \STK.FLAGS.SHIFT))

(RPAQ \STK.BF.WORD (LLSH \STK.BF \STK.FLAGS.SHIFT))

(CONSTANTS \STK.GUARD \STK.FX \STK.BF \STK.NOTFLAG \STK.FSB \STK.FLAGS.SHIFT (\STK.FSB.WORD (LLSH \STK.FSB
                                                                                              \STK.FLAGS.SHIFT
                                                                                               ))
       (\STK.GUARD.WORD (LLSH \STK.GUARD \STK.FLAGS.SHIFT))
       (\STK.BF.WORD (LLSH \STK.BF \STK.FLAGS.SHIFT)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \StackAreaSize 768)

(RPAQ \InitStackSize (ITIMES \StackAreaSize 12))

(CONSTANTS \StackAreaSize (\InitStackSize (ITIMES \StackAreaSize 12)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \MAXSAFEUSECOUNT 200)

(CONSTANTS \MAXSAFEUSECOUNT)
)

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD NAMETABLESLOT ((VARTYPE BYTE)
                            (VAROFFSET BYTE)))

[BLOCKRECORD FVARSLOT ((BINDLO WORD)
                       (BINDHI WORD))
       (ACCESSFNS FVARSLOT ((LOOKEDUP (EVENP (fetch BINDLO of DATUM)))
                            (BINDINGPTR (\VAG2 (fetch BINDHI of DATUM)
                                               (fetch BINDLO of DATUM))
                                   (PROGN (replace BINDLO of DATUM with (\LOLOC NEWVALUE))
```

```
                                        (replace BINDHI of DATUM with (\HILOC NEWVALUE]

[BLOCKRECORD PVARSLOT ((PVHI BITS 4)
                       (PVVALUE XPOINTER))
       (ACCESSFNS PVARSLOT ((BOUND (EQ (fetch (PVARSLOT PVHI) of DATUM)
                                       0)
                                   (if (NULL NEWVALUE)
                                       then (replace (PVARSLOT PVHI) of DATUM with 255)
                                     else (ERROR "Illegal replace" NEWVALUE]

[BLOCKRECORD STKTEMPSLOT ((STKTMPHI BITS 4)
                          (VALUE XPOINTER))
       (ACCESSFNS STKTEMPSLOT ((BINDINGPTRP (NEQ (fetch STKTMPHI of DATUM)
                                                 0]

[BLOCKRECORD BINDMARKSLOT ((BINDMARKP FLAG)
                           (NIL BITS 15))
       (BLOCKRECORD BINDMARKSLOT ((BINDNEGVALUES WORD)
                                  (BINDLASTPVAR WORD)))
       (ACCESSFNS BINDMARKSLOT ((BINDNVALUES (PROGN                ; Value stored in high half is one's complement of number of
                                                                   ; values bound
                                                   (LOGXOR (fetch BINDNEGVALUES of DATUM)
                                                           65535]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \NT.IVAR 0)

(RPAQQ \NT.PVAR 128)

(RPAQQ \NT.FVAR 192)

(CONSTANTS \NT.IVAR \NT.PVAR \NT.FVAR)
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE

[BLOCKRECORD STACKCELL ((STACKNONPOINTERBITS BITS 8)
                        (STACKHIBITS BITS 8)
                        (STACKLOBITS WORD))
       (ACCESSFNS STACKCELL ((VALIDPOINTERP (EQ 0 (fetch (STACKCELL STACKNONPOINTERBITS) of DATUM)))
                             (VALIDPOINTER (\GETBASEPTR DATUM 0]
)
)

;; For LAMBDA* and Common Lisp functions.

(DEFINEQ

(\MYARGCOUNT
  [LAMBDA NIL                                                       (* lmm " 6-OCT-81 23:15")
    ;; Opcode put out by the compiler in lambda* functions.  Returns number of arguments of the caller, to be bound to the lambda* variable.
    ;; Microcoded on some machines.
    (fetch (BF NARGS) of (fetch (FX BLINK) of (\MYALINK])

(\ARG0
  [LAMBDA (N)                                                       (* lmm " 6-OCT-81 23:15")
    ;; call to this function put out by compiler when compiling ARG for local argument.  Returns Nth argument of parent's frame
    (PROG [(BFLINK (fetch (FX BLINK) of (\MYALINK]                  ; BFLINK is the basic frame we are looking at
          (CHECK (type? BF BFLINK))
          (RETURN (COND
                     [[AND (IGREATERP N 0)
                           (NOT (IGREATERP N (fetch (BF NARGS) of BFLINK]
                                                                    ; N must be between 1 and the number of arguments
                      (GETBASEPTR \STACKSPACE (IPLUS (fetch (BF IVAR) of BFLINK)
                                                     (LLSH (SUB1 N)
                                                           1]
                     (T (LISPERROR "ILLEGAL ARG" N])

(\SETARG0
  [LAMBDA (N VAL)                                                   (* bvm%: " 5-Feb-85 16:10")
    ;; call to this function put out by compiler when compiling SETARG for local argument.  Sets Nth argument of parent's frame
    (PROG [(BFLINK (fetch (FX BLINK) of (\MYALINK]                  ; BFLINK is the basic frame we are looking at
          (CHECK (type? BF BFLINK))
          (RETURN (COND
                     ([AND (IGREATERP N 0)
                           (NOT (IGREATERP N (fetch (BF NARGS) of BFLINK]
                                                                    ; N must be between 1 and the number of arguments
```

```
                    (\PUTBASEPTR (ADDSTACKBASE (IPLUS (fetch (BF IVAR) of BFLINK)
                                                      (UNFOLD (SUB1 N)
                                                              WORDSPERCELL)))
                                 0 VAL))
                          (T (LISPERROR "ILLEGAL ARG" N])

)

;; basic spaghetti for allocating, moving and reclaiming stack frames

(DEFINEQ

(\HARDRETURN
  [LAMBDA (VAL)                                                    (* lmm "20-JUL-81 13:45")
    ;; Called by the microcode instead of returning to a frame whose use count is greater than one or alink neq clink.
    (\CONTEXTSWITCH \HardReturnFXP)
    VAL])

(\DOHARDRETURN
  [LAMBDA NIL                                                      (* lmm "27-JUL-81 09:07")
    (PROG NIL
      LP  (\DOHARDRETURN1 (fetch (IFPAGE HardReturnFXP) of \InterfacePage))
          (\CONTEXTSWITCH \HardReturnFXP)
          (GO LP])

(\DOGC1
  [LAMBDA NIL                                                      (* lmm " 1-SEP-81 00:53")
    ;; Body of the GC hard context.
    (\GCSCANSTACK)
    (\GCMAPSCAN)                                                   ; map thru, releasing entries
    (\GCMAPUNSCAN)                                                 ; map thru, unmarking stack entries
    NIL])

(\DOGC
  [LAMBDA NIL                                                      (* lmm " 1-SEP-81 00:52")
    (PROG NIL
      LP  (\DOGC1)
          (\CONTEXTSWITCH \GCFXP)
          (GO LP])

(\DOHARDRETURN1
  [LAMBDA (HRFRAME)                                                ; Edited 16-Apr-87 15:00 by bvm:
    ;; executed in the hard return context.  HRFRAME is the context in which \HARDRETURN was invoked.  We want to fix \HARDRETURN's caller to
    ;; do a return to its caller
    [COND
       ((EQ (fetch (FX FNHEADER FRAMENAME) of HRFRAME)
            '\CONTEXTSWITCH)
        ;; We really want to mung \HARDRETURN frame, not \CONTEXTSWITCH.  Test is needed in case \CONTEXTSWITCH is microcode
        (SETQ HRFRAME (fetch (FX CLINK) of HRFRAME]
    (PROG ((RETURNER (fetch (FX CLINK) of HRFRAME))
           RETURNEE AX NEW SIZE RETBF NAMETABLE-ON-STACK)
          (CHECK (EQ (fetch (FX FNHEADER FRAMENAME) of HRFRAME)
                     '\HARDRETURN))
          (SETQ RETURNEE (fetch (FX CLINK) of RETURNER))
          [CHECK (fetch (FX CHECKED) of RETURNEE)
                 (fetch (FX CHECKED) of RETURNER)
                 (NOT (fetch (FX FASTP) of RETURNER))
                 (OR (NEQ RETURNEE (fetch (FX ALINK) of RETURNER))
                     (COND
                        ((NEQ (fetch (FX USECNT) of RETURNEE)
                              0)                                   ; use count of RETURNEE gt 1, must copy RETURNEE
                         T)
                        ((type? FSB (SETQ AX (fetch (FX NEXTBLOCK) of RETURNEE)))
                                                                   ; returnee followed by a free block, but that free block is too small
                         (ILEQ (fetch (FSB SIZE) of AX)
                               \MinExtraStackWords))
                        (T                                         ; must copy in all other cases
                                                                   (* was ((EQ AX (fetch (BF IVAR) of
                                                                   (SETQ AX (fetch (FX BLINK) of RETURNER))))
                                                                   (* ; "returnee followed by RETURNER's BF but it doesn't have a
                                                                   non-zero usecount") (NEQ (fetch
                                                                   (BF USECNT) of AX) 0)))
                           T]
          (COND
             ((NEQ RETURNEE (SETQ AX (fetch (FX ALINK) of RETURNER)))
                                                                   ; ALINK and CLINK of returner not the same.  Fix.
                                                                   ; Set new ALINK before decrementing count on old value
              (replace (FX ALINK) of RETURNER with RETURNEE)
              (\DECUSECOUNT AX)))
```

```
          (COND
             ((COND
                 ((NEQ (fetch (FX USECNT) of RETURNEE)
                       0)                                    ; use count of RETURNEE gt 1, must copy RETURNEE
                  T)
                 ((type? FSB (SETQ AX (fetch (FX NEXTBLOCK) of RETURNEE)))
                                                            ; returnee followed by a free block, but that free block is too small
                                                            ; Should really require microcode merge the free blocks
                  (while [type? FSB (SETQ NEW (IPLUS AX (fetch (FSB SIZE) of AX]
                     do (add (fetch (FSB SIZE) of AX)
                             (fetch (FSB SIZE) of NEW)))
                  (ILEQ (fetch (FSB SIZE) of AX)
                        \MinExtraStackWords))
                 ([EQ AX (fetch (BF IVAR) of (SETQ AX (fetch (FX BLINK) of RETURNER]
                                                            ; returnee followed by RETURNER's BF but it doesn't have a
                                                            ; non-zero usecount
                  (NEQ (fetch (BF USECNT) of AX)
                       0))
                 (T                                         ; must copy in all other cases
                    T))                                     ; Must copy returnee to a new block because there isn't enough
                                                            ; room to return a value to it
              (FLIPCURSORBAR 5)
              (SETQ SIZE (IPLUS (fetch (FX SIZE) of RETURNEE)
                                WORDSPERCELL))
              [SETQ NEW (\FREESTACKBLOCK SIZE RETURNER (COND
                                                          ((AND (fetch (FX VALIDNAMETABLE) of RETURNEE)
                                                                (EQ (fetch (FX NAMETABHI) of RETURNEE)
                                                                    \STACKHI))
                                                            ; frame contains a name table, so we care that the alignment of
                                                            ; the new block be same as old
                                                           [CHECK (LET ((N (fetch (FX NAMETABLO) of RETURNEE)))
                                                                       (AND (> N RETURNEE)
                                                                            (< N (fetch (FX NEXTBLOCK)
                                                                                    of RETURNEE]
                                                           (SETQ NAMETABLE-ON-STACK T)
                                                           (IMOD (- RETURNEE WORDSPERCELL)
                                                                 WORDSPERQUAD]
                                                            ; Find a free stack block
              (\BLT (ADDSTACKBASE NEW)
                    (ADDSTACKBASE (IDIFFERENCE RETURNEE WORDSPERCELL))
                    SIZE)                                   ; copy frame and dummy bf pointer too
              (replace (BF RESIDUAL) of NEW with T)
              (add NEW WORDSPERCELL)                        ; now NEW points to the FX
              (replace (FX NEXTBLOCK) of NEW with (IDIFFERENCE (IPLUS NEW SIZE)
                                                               WORDSPERCELL))
              (replace (FX BLINK) of NEW with (SETQ RETBF (fetch (FX BLINK) of RETURNEE)))
                                                            ; Point to the real BF, not the residual
              (replace (FX USECNT) of NEW with 0)
              (CHECK (fetch (BF CHECKED) of RETBF))
              [COND
                 (NAMETABLE-ON-STACK                        ; Frame's nametable is on the stack, so it moved at the same
                                                            ; time the frame did
                        (add (fetch (FX NAMETABLO) of NEW)
                             (IDIFFERENCE NEW RETURNEE]
              (add (fetch (BF USECNT) of RETBF)
                   1)                                       ; increment use count of basic frame of returnee because we
                                                            ; made another FX which points to it
              (replace (FX FASTP) of RETURNEE with NIL)
              (\INCUSECOUNT (SETQ AX (fetch (FX CLINK) of RETURNEE)))
                                                            ; increment use count of CLINK of returnee because we made a
                                                            ; copy of returnee
              (COND
                 ((NEQ AX (SETQ AX (fetch (FX ALINK) of RETURNEE)))
                  (\INCUSECOUNT AX)))
              (\DECUSECOUNT RETURNEE)
              (replace (FX ACLINK) of RETURNER with NEW)
              (CHECK (fetch (FX CHECKED) of NEW)
                     (fetch (FX CHECKED) of RETURNER))
              (SETQ RETURNEE NEW)
              (FLIPCURSORBAR 5)))
          (\SMASHLINK HRFRAME RETURNEE RETURNEE])
```

(\DOSTACKOVERFLOW

```
  [LAMBDA NIL                                               ; Edited 9-Dec-86 13:59 by bvm:
    (PROG NIL
          (if \NEED.HARDRESET.CLEANUP
              then ;; Bootstrapping after hardreset mess.  Done here so that if death occurs, ^D from Raid will get us back.
                   (replace (IFPAGE SubovFXP) of \InterfacePage with (fetch (IFPAGE ResetFXP) of \InterfacePage))
                                                            ; So that if someone really tries to use this context, a reset will
                                                            ; occur--should never happen
                   (SETQ \RECLAIM.COUNTDOWN NIL)            ; inhibit gc
                   (\GATHER-CLEANUP-FORMS)
                   (replace (IFPAGE SubovFXP) of \InterfacePage with (\SETUPUSERSTACK \NEED.HARDRESET.CLEANUP))
                                                            ; Create the initial user stack, then switch back to it.
                                                            ; \SETUPUSERSTACK also reenables gc and clears
                                                            ; \NEED.HARDRESET.CLEANUP
```

```
                (\CONTEXTSWITCH \SubovFXP))
      LP  (replace (IFPAGE SubovFXP) of \InterfacePage with (\MOVEFRAME (fetch (IFPAGE SubovFXP) of \InterfacePage
                                                                          )))
          (\CONTEXTSWITCH \SubovFXP)
          (GO LP])
```

(\MOVEFRAME

```
  [LAMBDA (OLDFRAME)                                          ; Edited 11-Nov-87 13:00 by bvm
    (FLIPCURSORBAR 10)

;;; Called from \DOSTACKOVERFLOW when there isn't enough space to run in OLDFRAME --- Either we're at the end of stack space, in which case we
;;; can just extend the stack a bit, or we need to move OLDFRAME to somewhere else that has more free space after it.

    (PROG ((NXT (fetch (FX NEXTBLOCK) of OLDFRAME))
           OLDSIZE AX NEW NAMETABLE-ON-STACK AT-EOS FREESIZE)
      TRYFSB
          [COND
             ((type? FSB NXT)                                 ; Frame is followed by a free stack block, so maybe it's just not
                                                              ; big enough
              (if [type? FSB (SETQ NEW (+ NXT (SETQ FREESIZE (fetch (FSB SIZE) of NXT]
                  then                                        ; Oh, we just haven't merged our free blocks.  Merge and try
                                                              ; again.  Probably the microcode should be doing this.
                       [do (add FREESIZE (fetch (FSB SIZE) of NEW)) repeatwhile (type? FSB (SETQ NEW
                                                                                           (+ NXT FREESIZE]
                       (replace (FSB SIZE) of NXT with FREESIZE)
                       (SETQ NEW OLDFRAME)
                       (GO OUT)
                 elseif (EQ NEW (fetch (IFPAGE EndOfStack) of \InterfacePage))
                  then ;; Frame is at end of stack.  We have a problem here:  We'd like to avoid eating up stack when there might be oodles of
                       ;; space earlier in the stack.  However, in the case where we really do need more stack, it's painful to search the entire
                       ;; stack fruitlessly for a free block every time the current computation goes a little deeper.
                       (if (AND (> NEW \GuardStackAddr)
                                (NOT \STACKOVERFLOW))
                           then                               ; Compromise: do the search anyway if extending the stack
                                                              ; would trigger a stack overflow interrupt.
                                (SETQ AT-EOS T)
                         elseif (\EXTENDSTACK)
                           then                               ; Extend succeeded
                                (SETQ NEW OLDFRAME)
                                (GO OUT]
          (CHECK (fetch (FX CHECKED) of OLDFRAME)
                 (EQ (fetch (FX USECNT) of OLDFRAME)
                     0)
                 (NOT \INTERRUPTABLE))

      ;; Must copy OLDFRAME to a new block because there isn't enough room to run in it.  Get a free block big enough to hold the frame.
          [SETQ NEW (\FREESTACKBLOCK (SETQ OLDSIZE (+ (fetch (FX SIZE) of OLDFRAME)
                                                      WORDSPERCELL))
                                     OLDFRAME
                                     (COND
                                        ((AND (fetch (FX VALIDNAMETABLE) of OLDFRAME)
                                              (EQ (fetch (FX NAMETABHI) of OLDFRAME)
                                                  \STACKHI))     ; frame contains a name table, so we care that the alignment of
                                                                 ; the new block be same as old
                                         [CHECK (LET ((N (fetch (FX NAMETABLO) of OLDFRAME)))
                                                     (AND (> N OLDFRAME)
                                                          (< N (fetch (FX NEXTBLOCK) of OLDFRAME]
                                         (SETQ NAMETABLE-ON-STACK T)
                                         (IMOD (- OLDFRAME WORDSPERCELL)
                                               WORDSPERQUAD]
          (if (AND AT-EOS (> NEW OLDFRAME))
              then                                            ; Sigh, we had to extend the stack after all.  Just do it the easy
                                                              ; way.  FREESTACKBLOCK returned a guard block--just turn it
                                                              ; back into a free block and do the simple extend case.
                   (replace (FSB FLAGWORD) of NEW with \STK.FSB.WORD)
                   (GO TRYFSB))
          (\BLT (ADDSTACKBASE NEW)
                (ADDSTACKBASE (- OLDFRAME WORDSPERCELL))
                OLDSIZE)                                      ; copy frame and dummy bf pointer too
          (replace (BF RESIDUAL) of NEW with T)
          (add NEW WORDSPERCELL)                              ; now NEW points to the FX
          (replace (FX NEXTBLOCK) of NEW with (- (+ NEW OLDSIZE)
                                                 WORDSPERCELL))
          (CHECK (fetch (BF CHECKED) of (fetch (FX BLINK) of OLDFRAME)))
          (replace (FX BLINK) of NEW with (fetch (FX BLINK) of OLDFRAME))
                                                              ; Point at true BF, not residual
          [COND
             (NAMETABLE-ON-STACK                              ; Frame's nametable is on the stack, so it moved at the same
                                                              ; time the frame did
                     (CHECK (EVENP (- NEW OLDFRAME)
                                   WORDSPERQUAD))
                     (add (fetch (FX NAMETABLO) of NEW)
                          (- NEW OLDFRAME]
          [COND
             ((fetch (BF RESIDUAL) of (fetch (FX DUMMYBF) of OLDFRAME))
```

```
            (\MAKEFREEBLOCK (- OLDFRAME WORDSPERCELL)
                   OLDSIZE))
           (T (\MAKEFREEBLOCK OLDFRAME (- OLDSIZE WORDSPERCELL]
     OUT (FLIPCURSORBAR 10)                                    ; Restore cursor
         (RETURN NEW])
```

(\INCUSECOUNT

```
[LAMBDA (FRAME)                                                (* bvm%: "23-Mar-84 18:01")
  [COND
     ((NOT (fetch (FX INVALIDP) of FRAME))
      (CHECK (NOT \INTERRUPTABLE)
             (fetch (FX CHECKED) of FRAME))
      (COND
         ((IGREATERP (add (fetch (FX USECNT) of FRAME)
                          1)
                     \MAXSAFEUSECOUNT)
          (\MP.ERROR \MP.USECOUNTOVERFLOW "Stack frame use count maximum exceeded" FRAME)))
      (PROG ((SCANPTR (fetch (FX NEXTBLOCK) of FRAME)))         ; scan for BF ptr
            (SELECTC (fetch (STK FLAGS) of SCANPTR)
                 (\STK.NOTFLAG (until (type? BF (add SCANPTR WORDSPERCELL))))
                 (\STK.BF)
                 (RETURN))
            [CHECK (OR (fetch (BF RESIDUAL) of SCANPTR)
                       (EQ (fetch (BF IVAR) of SCANPTR)
                           (fetch (FX NEXTBLOCK) of FRAME]
            (COND
               ((type? FX (add SCANPTR WORDSPERCELL))
                (CHECK (fetch (FX CHECKED) of SCANPTR))
                (replace (FX FASTP) of SCANPTR with NIL]
  FRAME])
```

(\DECUSECOUNT

```
[LAMBDA (FRAME)                                                (* lmm " 4-SEP-81 09:29")
  (PROG (TEMP ALINK BLINK SIZE CLINK)
        (CHECK (NOT \INTERRUPTABLE))
    TOP (COND
           ((fetch (FX INVALIDP) of FRAME)                     ; reached top of stack
            (RETURN)))
        (CHECK (fetch (FX CHECKED) of FRAME))
        (COND
           ((NEQ (fetch (FX USECNT) of FRAME)
                 0)                                            ; USECNT (= use count + 1) greater than 1, merely decrement it
            (add (fetch (FX USECNT) of FRAME)
                 -1)
            (RETURN FRAME)))                                   ; ok, now free it
        (SETQ ALINK (fetch (FX ALINK) of FRAME))
        (SETQ BLINK (fetch (FX BLINK) of FRAME))
        (SETQ CLINK (fetch (FX CLINK) of FRAME))
        (SETQ SIZE (fetch (FX SIZE) of FRAME))
        (COND
           ((fetch (BF RESIDUAL) of (fetch (FX DUMMYBF) of FRAME))
            (\MAKEFREEBLOCK (IDIFFERENCE FRAME WORDSPERCELL)
                   (IPLUS SIZE WORDSPERCELL)))
           (T (\MAKEFREEBLOCK FRAME SIZE)))
        (CHECK (fetch (BF CHECKED) of BLINK))
        (COND
           ((EQ (fetch (BF USECNT) of BLINK)
                0)                                             ; frame extension count+1=0 --- release basic frame
            (\MAKEFREEBLOCK (fetch (BF IVAR) of BLINK)
                   (fetch (BF SIZE) of BLINK)))
           (T                                                  ; merely decrement extension count
              (add (fetch (BF USECNT) of BLINK)
                   -1)))
        (COND
           ((NEQ ALINK CLINK)
            (\DECUSECOUNT ALINK)))
        (SETQ FRAME CLINK)
        (GO TOP])
```

(\MAKESTACKP

```
[LAMBDA (ED FX)                                                (* bvm%: " 5-Jun-85 17:21")
```

;; Create a STACKP cell, possibly reusing ED, and pointing to FX

```
  (UNINTERRUPTABLY
      (COND
         ((NEQ FX 0)
          (\INCUSECOUNT FX)))
      (COND
         [(OR (STACKP ED)
              (TYPENAMEP ED 'PROCESS))
          (LET ((OLDFX (fetch (STACKP EDFXP) of ED)))
               (COND
                  ((NEQ OLDFX 0)
                   (\DECUSECOUNT OLDFX]
```

```
          (T (SETQ ED (CREATECELL \STACKP))
             (replace (STACKP STACKP0) of ED with \STACKHI)))
       (replace (STACKP EDFXP) of ED with FX))
    ED])
```

(\SMASHLINK

```
[LAMBDA (CALLER ALINK CLINK)                                    (* bvm%: " 5-Feb-85 16:19")
                                                                ; Smashes caller's ALINK and/or CLINK with ALINK and CLINK
   (OR CALLER (SETQ CALLER (\MYALINK)))
   (UNINTERRUPTABLY
       (PROG ((OLDALINK (fetch (FX ALINK) of CALLER))
              (OLDCLINK (fetch (FX CLINK) of CALLER))
              BLINK)
             (COND
                (ALINK (COND
                           ((NEQ ALINK (OR CLINK OLDCLINK))     ; Don't increment twice if ALINK comes out same as CLINK
                            (\INCUSECOUNT ALINK)))
                       (replace (FX ALINK) of CALLER with ALINK)))
             (COND
                (CLINK (COND
                           ((OR ALINK (NEQ CLINK OLDALINK))     ; If we're only setting the CLINK, and we're setting it to be the
                                                                ; same as the ALINK, don't bump count
                            (\INCUSECOUNT CLINK)))
                       (replace (FX CLINK) of CALLER with CLINK)
                       (\DECUSECOUNT OLDCLINK)))                ; must be careful to increment any use counts before
                                                                ; decrementing any
             (COND
                ((AND (NEQ OLDALINK OLDCLINK)
                      ALINK)
                 (\DECUSECOUNT OLDALINK)))
             (COND
                ((AND (EQ (OR ALINK (SETQ ALINK OLDALINK))
                          (OR CLINK (SETQ CLINK OLDCLINK)))
                      (EQ (fetch (FX USECNT) of CLINK)
                          0)
                      (EQ (SETQ BLINK (fetch (FX BLINK) of CALLER))
                          (fetch (FX DUMMYBF) of CALLER))
                      (EQ (fetch (BF IVAR) of BLINK)
                          (fetch (FX NEXTBLOCK) of CLINK))
                      (EQ (fetch (BF USECNT) of BLINK)
                          0)
                      (NOT (fetch (FX NOPUSH) of CLINK))
                      (NOT (fetch (FX INCALL) of CLINK)))
                 ;; We have made CALLER fast again: its alink and clink are same, usecnt of blink and caller are normal, bf is contiguous with
                 ;; CALLER and CALLER's caller
                 (replace (FX SLOWP) of CALLER with NIL)))
             (RETURN CALLER)))])
```

(\FREESTACKBLOCK

```
[LAMBDA (N START ALIGN)                                         ; Edited 16-Apr-87 14:53 by bvm:
   ;; Scan stack space searching for a free block of size at least n, starting scan at start (or beginning of stackspace if START=NIL).  The block
   ;; returned has the quadword alignment requested by ALIGN (0 or 2) if ALIGN is non-NIL.
   (PROG ((WANTEDSIZE (IPLUS N \StackAreaSize \MinExtraStackWords))
          FREEPTR FREESIZE (EASP (fetch EndOfStack of \InterfacePage))
          SCANPTR)
         [CHECK (OR (NULL START)
                    (IGEQ START (fetch StackBase of \InterfacePage]
     STARTOVER
         (SETQ SCANPTR (OR START (fetch StackBase of \InterfacePage)))
     SCAN
         (SELECTC (fetch (STK FLAGS) of SCANPTR)
             (\STK.FSB (GO FREESCAN))
             (\STK.GUARD (COND
                            ((ILESSP SCANPTR EASP)              ; Guard block not at end of stack, treat as a free block
                             (GO FREESCAN)))                    ; reached end
                         (COND
                            (START                              ; had a starting place, just wrap around
                                   (SETQ SCANPTR (fetch StackBase of \InterfacePage))
                                   (GO SCAN))
                            (T                                  ; Scanned the entire stack --- add a new page
                               (GO NEWPAGE))))
             (\STK.FX                                           ; frame extension
                      (CHECK (fetch (FX CHECKED) of SCANPTR))
                      (SETQ SCANPTR (fetch (FX NEXTBLOCK) of SCANPTR)))
             (PROG ((ORIG SCANPTR))                             ; must be a basic frame
                   (until (type? BF SCANPTR) do (CHECK (EQ (fetch (STK FLAGS) of SCANPTR)
                                                           \STK.NOTFLAG))
                                                (add SCANPTR WORDSPERCELL))
                   [CHECK (COND
                             ((fetch (BF RESIDUAL) of SCANPTR)
                              (EQ SCANPTR ORIG))
                             (T (AND (fetch (BF CHECKED) of SCANPTR)
                                     (EQ ORIG (fetch (BF IVAR) of SCANPTR]
```

```
                    (add SCANPTR WORDSPERCELL)))
      NEXT
           (COND
              ((NEQ SCANPTR START)
               (CHECK (ILEQ SCANPTR EASP))
               (GO SCAN)))
      NEWPAGE
           [COND
              ((SETQ EASP (\EXTENDSTACK))
               (GO STARTOVER))
              (T (while T do (\MP.ERROR \MP.STACKFULL "Stack Full -- Type LU to see stack; ^D to flush to top"]
      FREESCAN
           (SETQ FREEPTR SCANPTR)
           (SETQ FREESIZE (fetch (FSB SIZE) of SCANPTR))
      FREE                                                          ; Merge free blocks
           (SETQ SCANPTR (IPLUS FREEPTR FREESIZE))
           (COND
              ((SELECTC (fetch (STK FLAGS) of SCANPTR)
                    (\STK.FSB T)
                    (\STK.GUARD (ILESSP SCANPTR EASP))
                    NIL)
               (add FREESIZE (fetch (FSB SIZE) of SCANPTR))
               (GO FREE)))
           (COND
              ((IGEQ FREESIZE WANTEDSIZE)                           ; Found a large enough block -- Split the block
               [SETQ WANTEDSIZE (COND
                                   ((OR (NULL ALIGN)
                                        (EQ ALIGN (IMOD FREEPTR WORDSPERQUAD)))
                                                                    ; alignment ok.  Assumes that \MinExtraStackWords is multiple
                                                                    ; of 4
                                    \MinExtraStackWords)
                                   (T                               ; no, adjust alignment
                                      (IPLUS WORDSPERCELL \MinExtraStackWords]
               (SETQ SCANPTR (\SETUPGUARDBLOCK (IPLUS FREEPTR WANTEDSIZE)
                                N))                                 ; Block to return: desired size (n), properly aligned
               (\MAKEFREEBLOCK FREEPTR WANTEDSIZE)                  ; Leave a little free block before it
               (\MAKEFREEBLOCK (IPLUS SCANPTR N)
                      (IDIFFERENCE (IDIFFERENCE FREESIZE WANTEDSIZE)
                             N))                                    ; and a big free block after
               (RETURN SCANPTR))
              (T (\MAKEFREEBLOCK FREEPTR FREESIZE)))
           (GO NEXT])
```

(\EXTENDSTACK

```
  [LAMBDA NIL                                                       (* bvm%: "18-JAN-83 12:12")
    (PROG ((EASP (fetch (IFPAGE EndOfStack) of \InterfacePage))
           SCANPTR)
          (RETURN (COND
                     ((ILESSP EASP \LastStackAddr)
                      (if (AND (IGREATERP EASP \GuardStackAddr)
                               (NOT \STACKOVERFLOW))
                          then (replace STACKOVERFLOW of \INTERRUPTSTATE with T)
                               (SETQ \STACKOVERFLOW (SETQ \PENDINGINTERRUPT T)))
                      (OR (\DONEWPAGE (ADDSTACKBASE (SETQ SCANPTR (IPLUS EASP 2)))
                                      T)
                          (\DOLOCKPAGES (ADDSTACKBASE SCANPTR)
                                        1))
                      ;; Create, if necessary, new page and lock it.  Second clause happens when page already existed.  \DONEWPAGE
                      ;; instead of \NEWPAGE etc. because we are in a safe context (and might even be in the misc context)
                      (\MAKEFREEBLOCK SCANPTR (IDIFFERENCE WORDSPERPAGE 2))
                      (\SETUPGUARDBLOCK (SETQ SCANPTR (IPLUS EASP WORDSPERPAGE))
                                        2)
                      (replace EndOfStack of \InterfacePage with SCANPTR)
                      (\MAKEFREEBLOCK EASP 2)
                      SCANPTR])
```

)

;; Some ugly stack-munging ufns

(DEFINEQ

(\SLOWRETURN

```
  [LAMBDA NIL                                                       (* lmm "30-Dec-84 03:31")
    ;; force caller to slow return
    (replace (FX FASTP) of (\MYALINK) with NIL])
```

(\COPY.N.UFN

```
  [LAMBDA (ALPHA)                                                   (* lmm " 2-Jan-85 01:29")
    (\.GETBASE32 \STACKSPACE (IDIFFERENCE (fetch (FX NEXTBLOCK) of (\MYALINK))
                                          (IPLUS ALPHA WORDSPERCELL])
```

(\POP.N.UFN

```
  [LAMBDA (N)                                                    ; Edited  5-Jul-88 18:08 by amd
    (\SLOWRETURN)
    (LET ((AL (\MYALINK))
          NEXT VAL LEN)
         (SETQ NEXT (fetch (FX NEXTBLOCK) of AL))
         [SETQ VAL (\GETBASEPTR \STACKSPACE (SETQ NEXT (IDIFFERENCE NEXT (SETQ LEN (UNFOLD (ADD1 N)
                                                                                           WORDSPERCELL]
         (\MAKEFREEBLOCK NEXT LEN)
         (replace (FX NEXTBLOCK) of AL with NEXT)
         (replace (FX NOPUSH) of AL with T])
```

(\STORE.N.UFN

```
  [LAMBDA (VAL ALPHA)                                            (* lmm " 2-Jan-85 01:30")
    (\.PUTBASE32 \STACKSPACE (IDIFFERENCE (fetch (FX NEXTBLOCK) of (\MYALINK))
                                          (IPLUS ALPHA WORDSPERCELL))
          VAL])
```

(\UNWIND.UFN

```
  [LAMBDA (N.KEEP)                                               ; Edited 27-Sep-88 11:48 by jds
```

;;; UFN for UNWIND opcode. The two bytes are the desired stack depth to unwind to and a flag indicating whether to push TOS when done.

```
    (LET* ((CALLER (\MYALINK))
           (NEXT (fetch (FX NEXTBLOCK) of CALLER))
           (SP NEXT)
           (DESIREDSP (IPLUS (IDIFFERENCE (fetch (FX FIRSTPVAR) of CALLER)
                                          WORDSPERCELL)
                             (UNFOLD (LRSH N.KEEP 8)
                                     WORDSPERCELL)))
           (PUSHP (NEQ (LOGAND N.KEEP 255)
                       0))
           OLDTOS)
          [COND
             (PUSHP                                              ; Save old top of stack
                    (SETQ OLDTOS (\GETBASEPTR (STACKADDBASE (IDIFFERENCE SP WORDSPERCELL))
                                              0]
          (UNINTERRUPTABLY
              [while (GREATERP (add SP (IMINUS WORDSPERCELL))
                               DESIREDSP)
                 bind (PVAR0BASE _ (STACKADDBASE (fetch (FX FIRSTPVAR) of CALLER)))
                 when (fetch BINDMARKP of (STACKADDBASE SP))
                 do                                              ; Unbind stuff.  Bind mark says how many pvars were bound,
                                                                 ; and gives the offset of the last of them
                    (LET [(LASTPVAR (fetch BINDLASTPVAR of (STACKADDBASE SP]
                         (to (fetch BINDNVALUES of (STACKADDBASE SP)) do (\PUTBASE PVAR0BASE LASTPVAR 65535)
                                                                         (SETQ LASTPVAR (IDIFFERENCE LASTPVAR
                                                                                                     WORDSPERCELL]
              (replace (FX NEXTBLOCK) of CALLER with (add DESIREDSP WORDSPERCELL))
              (\MAKEFREEBLOCK DESIREDSP (IDIFFERENCE NEXT DESIREDSP))
              (COND
                 ((NOT PUSHP)                                    ; Keep return value from being pushed
                  (replace (FX NOPUSH) of CALLER with T)))
              ;; Now explicitly slow return to caller, since we have violated the fast return assumptions by blowing away stack between here and
              ;; there
              (\SLOWRETURN)
              OLDTOS)])

)

;; The unwinder

(DEFINEQ
```

(SI::NON-LOCAL-GO

```
  [LAMBDA (BLIP PC)                                              (* bvm%: " 4-Nov-86 16:30")
```

;;; Performs a non-local GO. BLIP is the control blip of the target frame; PC is the place to resume execution. We unwind the stack and return to the
;;; target frame at the specified PC.

```
    (LET ((TARGET (SI::UNWIND-TO-BLIP BLIP NIL)))
         (if TARGET
             then                                                ; Unwound ok.  Stack now has me pointing at the BLIP frame.
                                                                 ; Adjust the pc and return to it.
                  (replace (FX PC) of TARGET with PC)
                 T
           else (CL:ERROR 'ILLEGAL-GO])
```

(SI::NON-LOCAL-RETURN

```
  [LAMBDA BLIP&VALUES                                            (* bvm%: " 4-Nov-86 22:13")
```

;;; Effective arg list is (BLIP &REST VALUES), done this way to avoid consing. Returns the multiple values VALUES to/from the frame that binds the

;;; control blip BLIP. Information in the frame says whether return goes to the frame (at a specified pc) or from it.

```
(if (SI::UNWIND-TO-BLIP (ARG BLIP&VALUES 1)
          T)
    then                                                ; Unwound ok.  Stack now has me pointing at the BLIP frame.
                                                        ; Return multiple values to it.
         [if (EQ BLIP&VALUES 2)
             then                                       ; given exactly one value, so we can take normal return
                  (ARG BLIP&VALUES 2)
           else (CL:VALUES-LIST (for I from 2 to BLIP&VALUES collect (ARG BLIP&VALUES I]
  else (CL:ERROR 'ILLEGAL-RETURN])
```

(SI::NON-LOCAL-RETURN-VALUES

```
[CL:LAMBDA (BLIP VALUES)                                (* bvm%: " 4-Nov-86 22:14")
```

;;; Returns the multiple values VALUES to/from the frame that binds the control blip BLIP. Information in the frame says whether return goes to the
;;; frame (at a specified pc) or from it.

```
(if (SI::UNWIND-TO-BLIP BLIP T)
    then                                                ; Unwound ok.  Stack now has me pointing at the BLIP frame.
                                                        ; Return multiple values to it.
         (if (AND VALUES (NULL (CDR VALUES)))
             then                                       ; fast return of one value
                  (CAR VALUES)
           else (CL:VALUES-LIST VALUES))
  else (CL:ERROR 'ILLEGAL-RETURN])
```

(SI::INTERNAL-THROW

```
[LAMBDA TAG&VALUES                                      (* bvm%: " 4-Nov-86 22:39")
```

;;; Effective arg list is (TAG &REST VALUES), done this way to avoid consing. THROW's the multiple values VALUES to TAG. TAG is bound as the
;;; control blip of the catch frame.

```
(if (SI::UNWIND-TO-BLIP (ARG TAG&VALUES 1)
          'CL:THROW)
    then                                                ; Unwound ok.  Stack now has me pointing at the BLIP frame.
                                                        ; Return multiple values to it.
         [if (EQ TAG&VALUES 2)
             then                                       ; given exactly one value, so we can take normal return
                  (ARG TAG&VALUES 2)
           else (CL:VALUES-LIST (for I from 2 to TAG&VALUES collect (ARG TAG&VALUES I]
  else (CL:ERROR 'ILLEGAL-THROW :TAG (ARG TAG&VALUES 1])
```

(SI::INTERNAL-THROW-VALUES

```
(CL:LAMBDA (TAG VALUES)                                 (* bvm%: " 4-Nov-86 22:14")
```

;;; THROW's the multiple values VALUES to TAG. TAG is bound as the control blip of the catch frame.

```
(if (SI::UNWIND-TO-BLIP TAG 'CL:THROW)
    then                                                ; Unwound ok.  Stack now has me pointing at the BLIP frame.
                                                        ; Return multiple values to it.
         (if (AND VALUES (NULL (CDR VALUES)))
             then                                       ; fast return of one value
                  (CAR VALUES)
           else (CL:VALUES-LIST VALUES))
  else (CL:ERROR 'ILLEGAL-THROW :TAG TAG))))
```

(SI::UNWIND-TO-BLIP

```
[LAMBDA (BLIP THROWP UNWINDER)                          ; Edited 18-Feb-91 16:12 by jds
```

;;; Searches stack from caller of UNWINDER backwards for a frame that binds BLIP as its control blip. Returns that frame or that frame's caller,
;;; depending on how we are supposed to return (if THROWP is NIL, always return TO the frame; else frame says). Returns NIL on failure to find BLIP.
;;; UNWINDER defaults to the caller.

;;;

;;; For this implementation, control blips and catch tags are stored in pvar1. The var's name is SI::*CATCH-RETURN-FROM* if control exits the frame,
;;; or SI::*CATCH-RETURN-TO* if control is to return to the frame, in which case the frame's special var SI::*CATCH-RETURN-PC* has the pc value.

```
(bind [TARGET _ (OR UNWINDER (SETQ UNWINDER (\MYALINK]
      PC until (fetch (FX INVALIDP) of (SETQ TARGET (fetch (FX CLINK) of TARGET)))
   when (AND (EQ BLIP (\GETBASEPTR (ADDSTACKBASE (+ (fetch (FX FIRSTPVAR) of TARGET)
                                                     WORDSPERCELL))
                                   0))
             (SELECTQ (SI::VARIABLE-NAME-IN-FRAME TARGET (NEW-SYMBOL-CODE (IPLUS (LLSH PVARCODE 16)
                                                                                 1)
                                                                          (SI::IPLUS PVARCODE 1)))
                 (SI::*CATCH-RETURN-TO*
                      [COND
                         (THROWP                        ; we're doing a RETURN/THROW where we accomplish the task
                                                        ; like GO
                                 (OR [SMALLP (SETQ PC (SI::PVAR-VALUE-IN-FRAME TARGET
                                                      (NEW-SYMBOL-CODE 'SI::*CATCH-RETURN-PC*
```

```
                                                                    (\ATOMVALINDEX 'SI::*CATCH-RETURN-PC*]
                                        (ERROR "Catch return-to frame lacks PC" TARGET]
                        T)
                    (SI::*CATCH-RETURN-FROM*
                        [COND
                           (THROWP                                  ; if THROW then this is the RETURN-FROM flavor
                                                                    ; Go one frame further back
                                   (SETQ TARGET (fetch (FX CLINK) of TARGET]
                        T)
                    (PROGN                                          ; blip matches contents of pvar1 but the name is wrong.  This is
                                                                    ; important for THROW, less so for GO and RETURN-FROM.
                           NIL)))
    do (SI::UNWIND TARGET 'ERROR UNWINDER)
    (COND
       (PC                                                          ; a THROW TO needs a pc adjustment
            (replace (FX PC) of TARGET with PC)))
    (RETURN TARGET])
```

(SI::UNWIND

```
  [LAMBDA (TARGET RESETSTATE UNWINDER)                              (* bvm%: " 4-Nov-86 22:24")
    (DECLARE (CL:SPECIAL RESETSTATE))
```

;;; Unwinds the stack between UNWINDER and TARGET. UNWINDER defaults to the caller. Returns to caller with it positioned to return to TARGET.
;;; RESETSTATE is the value to be seen by RESETSAVEs along the way.

;;; A TARGET of -1 means unwind the stack until you get to a frame with non-null use count. This is for returning to different stack groups.

;;; We are assuming that nobody from UNWINDER to here binds specvars (except RESETSTATE).

```
    (if (NEQ TARGET UNWINDER)
        then                                                        ; TARGET and UNWINDER could be indentical in the case
                                                                    ; where a frame THROWs from itself.
          (LET ((USECNTARGET (MINUSP TARGET))
                CLEANUPFN)
               (OR RESETSTATE (SETQ RESETSTATE 'ERROR))
               (OR UNWINDER (SETQ UNWINDER (\MYALINK)))
               (for (FRAME _ (fetch (FX CLINK) of UNWINDER))
                  until (if USECNTARGET
                            then (OR (fetch (FX INVALIDP) of FRAME)
                                     (NEQ (fetch (FX USECNT) of FRAME)
                                          0))
                          else (EQ FRAME TARGET))
                  do (SETQ CLEANUPFN (AND (EQ (fetch (FX FNHEADER FRAMENAME) of FRAME)
                                              'SI::*UNWIND-PROTECT*)
                                          (\GETBASEPTR (ADDSTACKBASE (fetch (FX IVAR) of FRAME))
                                                 0)))
                     ;; cleanup forms are stored in first ivar.  Go straight to the FNHEADER of the frame, since there is never an interpreted
                     ;; *UNWIND-PROTECT* frame
                     (SETQ FRAME (\DISCARDFRAME UNWINDER))          ; Discard frame, set FRAME to next ancestor
                     (if CLEANUPFN
                         then                                       ; only run the cleanup form after we have blown away the frame,
                                                                    ; so the dynamic bindings are right
                              (CL:FUNCALL CLEANUPFN])
```

(SI::VARIABLE-NAME-IN-FRAME

```
  [LAMBDA (FRAME CODE)                                              ; Edited 18-Feb-91 16:09 by jds
```

;;; Returns name of the var whose name table encoding is CODE (i.e., xVARCODE+n for xVARn).

```
    (LET ((NT (fetch (FX NAMETABLE) of FRAME))
          NAME)
         (for (NTBASE _ (\ADDBASE NT (fetch (FNHEADER OVERHEADWORDS) of T))) by (\ADDBASE NTBASE (CONSTANT (
                                                                                                  WORDSPERNAMEENTRY
                                                                                                        )))
           as [NT2BASE _ (\ADDBASE NT (+ (fetch (FNHEADER OVERHEADWORDS) of T)
                                         (fetch (FNHEADER NTSIZE) of NT]
           by (\ADDBASE NT2BASE (CONSTANT (WORDSPERNTOFFSETENTRY))) until (NULL-NTENTRY (SETQ NAME
                                                                                              (GETSTKNAMEENTRY NTBASE
                                                                                                     0)))
           when (EQP (GETSTKNTOFFSETENTRY NT2BASE 0)
                     CODE)
           do (RETURN (\INDEXATOMVAL NAME])
```

(SI::PVAR-VALUE-IN-FRAME

```
  [LAMBDA (FRAME ATOM#)                                             ; Edited 18-Feb-91 14:56 by jds
```

;;; Returns value of pvar binding of atom number ATOM# in FRAME. FRAME is guaranteed not to be an interpreter frame

```
    (for OFFSET from (fetch (FNHEADER OVERHEADWORDS) of T) by (WORDSPERNTOFFSETENTRY)
       bind (NT _ (fetch (FX FNHEADER) of FRAME))
            TMP NAME
       until (NULL-NTENTRY (SETQ NAME (GETSTKNAMEENTRY NT OFFSET)))
       do (COND
```

```
            ([AND (EQP NAME ATOM#)
                  (EQP [NTSLOT-VARTYPE (SETQ TMP (GETSTKNTOFFSETENTRY NT (+ OFFSET (fetch (FNHEADER NTSIZE)
                                                                                      of NT]
                       PVARCODE)
                  (fetch (PVARSLOT BOUND) of (SETQ TMP (ADDSTACKBASE (+ (fetch (FX FIRSTPVAR) of FRAME)
                                                                        (UNFOLD (NTSLOT-OFFSET TMP)
                                                                                WORDSPERCELL]
                                                                    ; Found ATOM# in as a bound pvar
             (RETURN (fetch (PVARSLOT PVVALUE) of TMP])

)

(DEFINEQ

(\DISCARDFRAME
  [LAMBDA (CHILD)                                                   (* bvm "22-Nov-86 15:15")
    ;; Splice out CHILD's parent.  Return its new parent.
    (UNINTERRUPTABLY
        (PROG ((OLDALINK (fetch (FX ALINK) of CHILD))
               (OLDCLINK (fetch (FX CLINK) of CHILD))
               NEWCLINK BLINK)
              (if (NEQ OLDALINK OLDCLINK)
                  then (\DECUSECOUNT OLDALINK))
              (SETQ NEWCLINK (fetch (FX CLINK) of OLDCLINK))
              (replace (FX ACLINK) of CHILD with NEWCLINK)          ; Set new A&C links to new parent.  This also makes CHILD
                                                                    ; slow.  Now we're ready to wipe out OLDCLINK
              (LET ((BLINK (fetch (FX BLINK) of OLDCLINK))
                    (SIZE (fetch (FX SIZE) of OLDCLINK))
                    (ALINK (fetch (FX ALINK) of OLDCLINK))
                    OLDUSECOUNT)
                   (if (NEQ ALINK NEWCLINK)
                       then                                         ; dec usecnt of ALINK of frame we're discarding.  Normally
                                                                    ; ALINK = CLINK = NEWCLINK, so we don't have to touch it
                            (\DECUSECOUNT ALINK))
                   (if (EQ (SETQ OLDUSECOUNT (fetch (FX USECNT) of OLDCLINK))
                           0)
                       then ;; normal case, this frame really will be discarded.  This following code is an optimization of the \INCUSECOUNT +
                            ;; \DECUSECOUNT pair you would get by just doing the straightforward \SMASHLINK.
                            (COND
                               ((fetch (BF RESIDUAL) of (fetch (FX DUMMYBF) of OLDCLINK))
                                                                    ; free the dummy bf as well
                                (\MAKEFREEBLOCK (IDIFFERENCE OLDCLINK WORDSPERCELL)
                                       (IPLUS SIZE WORDSPERCELL)))
                               (T (\MAKEFREEBLOCK OLDCLINK SIZE)))
                            (CHECK (fetch (BF CHECKED) of BLINK))
                            (COND
                               ((EQ (fetch (BF USECNT) of BLINK)
                                    0)                              ; frame extension count+1=0 so release basic frame
                                (\MAKEFREEBLOCK (fetch (BF IVAR) of BLINK)
                                       (fetch (BF SIZE) of BLINK)))
                               (T                                   ; merely decrement extension count
                                  (add (fetch (BF USECNT) of BLINK)
                                       -1)))
                     else ;; Can't discard frame because someone's pointing at it.  However, we can chop off its parents, leaving any holder with
                          ;; a stack it can't return to.  Also have to decrement use count to account for dropping the pointer to it from CHILD.
                          (replace (FX USECNT) of OLDCLINK with (SUB1 OLDUSECOUNT))
                          (replace (FX ACLINK) of OLDCLINK with 0)))
             (RETURN NEWCLINK))])

(\SMASHRETURN
  [LAMBDA (CALLER FRAME STKP)                                       (* bvm "22-Nov-86 15:34")
    ;; Modify CALLER's a & c links to make it return to FRAME.  If FRAME is an ancestor, we can unwind as we go.  If FRAME is not an ancestor,
    ;; things get fuzzy.  If STKP is supplied, it is a stack pointer that should be released afterwards, though we will release it early if possible.
    (LET ((MYCALLER (\MYALINK))
          DESTPROC)
         (OR CALLER (SETQ CALLER MYCALLER))
         (if [for (FX _ MYCALLER) when (EQ FX FRAME) do (RETURN T) repeatuntil (fetch (FX INVALIDP)
                                                                                     of (SETQ FX (fetch (FX CLINK)
                                                                                                    of FX]
             then                                                   ; direct ancestor -- blow away everything in between
                  (if STKP
                      then ;; Since FRAME is a direct ancestor, it is safe to release stack pointer before unwinding, because its release cannot
                           ;; end up releasing FRAME--we have an implicit pointer to it via the control stack.  By releasing STKP now, UNWIND
                           ;; may be able to dispose of more.
                           (RELSTK STKP))
                  (SI::UNWIND FRAME NIL CALLER)
           else                                                     ; returning to different stack group.  Better be in same process.
                (\SMASHLINK NIL FRAME)
                (SETQ DESTPROC *CURRENT-PROCESS*)
                (\SMASHLINK NIL MYCALLER)
                (if (EQ DESTPROC (THIS.PROCESS))
```

```
                    then                                        ; unwind as long as nobody else is pointing at stack
                         (SI::UNWIND -1 NIL CALLER)             ; then return to requested place
                         (\SMASHLINK CALLER FRAME FRAME)
                         (if STKP
                             then ;; have to release stack pointer AFTER unwinding, because it could be the only pointer to FRAME
                                  (RELSTK STKP))
                    else                                        ; Trying to return to a different process.  This could cause utter
                                                                ; confusion.
                         (CL:ERROR "Attempt to return to a different process: ~A at frame ~S" (PROCESSPROP
                                                                                                DESTPROC
                                                                                                'NAME)
                                   (fetch (FX FNHEADER) of FRAME])

)

;; parsing stack for gc

(DEFINEQ

(\GCSCANSTACK
  [LAMBDA NIL                                                   ; Edited 23-Jun-88 16:06 by rtk
                                                                ; scan stack space calling SCANREF on all pointers
    (PROG ((SCANPTR (fetch StackBase of \InterfacePage))
           (EASP (fetch EndOfStack of \InterfacePage))
           SCANBASE Q)
      LP   (SELECTC (fetch (STK FLAGS) of SCANPTR)
                    (\STK.FX                                    ; frame extension
                             (PROG (NTEND NEXT)
                                   (CHECK (fetch (FX CHECKED) of SCANPTR))
                                   (\STKREF (fetch (FX FNHEADER) of SCANPTR))
                                   [SETQ SCANBASE (ADDSTACKBASE (SETQ Q (fetch (FX FIRSTPVAR) of SCANPTR]
                                   (FRPTQ (fetch (FX FNHEADER NLOCALS) of SCANPTR)
                                          [COND
                                             ((fetch (PVARSLOT BOUND) of SCANBASE)
                                              (\STKREF (fetch (STACKCELL VALIDPOINTER) of SCANBASE]
                                          (SETQ SCANBASE (\ADDBASE SCANBASE WORDSPERCELL)))
                                   (SETQ NEXT (SETQ Q (fetch (FX NEXTBLOCK) of SCANPTR)))
                                   (CHECK (IGREATERP NEXT SCANPTR))
                                   [COND
                                      ((fetch (FX VALIDNAMETABLE) of SCANPTR)
                                                                ; Frame has separate nametable.
                                       (COND
                                          ((EQ (fetch (FX NAMETABHI) of SCANPTR)
                                               \STACKHI)
                                           ;; Nametable is on stack.  Need to refcnt its framename and skip that section of the stack, since it
                                           ;; does not contain pointers
                                           (CHECK (ILEQ (fetch (FX NAMETABLO) of SCANPTR)
                                                        Q))
                                           (\STKREF (fetch (FNHEADER FRAMENAME) of (fetch (FX NAMETABLE#) of SCANPTR)))
                                           (SETQ Q (fetch (FX NAMETABLO) of SCANPTR))
                                           (SETQ NTEND (IPLUS Q (fetch (FNHEADER OVERHEADWORDS) of T)
                                                              (UNFOLD (fetch (FNHEADER NTSIZE) of (ADDSTACKBASE Q))
                                                                      2)))
                                                                ; Need to skip the region from Q to NTEND

                                           )
                                          (T                    ; Nametable elsewhere, just reference it
                                             (\STKREF (fetch (FX NAMETABLE) of SCANPTR]
                                   (SETQ SCANPTR (fetch (FX FIRSTTEMP) of SCANPTR))
                                 SCANTEMPS
                                   (SETQ SCANBASE (ADDSTACKBASE SCANPTR))
                                   (while (ILESSP SCANPTR Q) do [COND
                                                                   ((fetch (STACKCELL VALIDPOINTERP) of SCANBASE)
                                                                    (\STKREF (fetch (STACKCELL VALIDPOINTER)
                                                                                of SCANBASE]
                                                                (add SCANPTR WORDSPERCELL)
                                                                (SETQ SCANBASE (\ADDBASE SCANBASE WORDSPERCELL)))
                                   (COND
                                      (NTEND                    ; Skip over NT, scan after it
                                             (SETQ SCANPTR NTEND)
                                             (SETQ Q NEXT)
                                             (SETQ NTEND)
                                             (GO SCANTEMPS)))
                                   ;; in case this frame was truncated, don't trust the FIRSTTEMP field.
                                   (SETQ SCANPTR NEXT)))
                    (\STK.GUARD (AND (EQ SCANPTR EASP)
                                     (RETURN))
                                (add SCANPTR (fetch (FSB SIZE) of SCANPTR)))
                    (\STK.FSB (if NIL
                                  then                          ; to merge free blocks
                                       (SETQ Q (IPLUS SCANPTR (fetch (FSB SIZE) of SCANPTR)))
                                       (CHECK (NEQ SCANPTR Q))
                                       (while (type? FSB Q) do (add (fetch (FSB SIZE) of SCANPTR)
                                                                    (SETQ NV (fetch (FSB SIZE) of Q)))
```

```
                                                    (add Q NV))
                                (SETQ SCANPTR Q))
                           (add SCANPTR (fetch (FSB SIZE) of SCANPTR)))
                (LET ((ORIG SCANPTR))                                   ; must be a basic frame
                     (SETQ SCANBASE (ADDSTACKBASE SCANPTR))
                     (until (type? BF SCANPTR) do (CHECK (EQ (fetch (STK FLAGS) of SCANPTR)
                                                             \STK.NOTFLAG))
                                                  (\STKREF (fetch (STACKCELL VALIDPOINTER) of SCANBASE))
                                                  (add SCANPTR WORDSPERCELL)
                                                  (SETQ SCANBASE (\ADDBASE SCANBASE WORDSPERCELL)))
                     [CHECK (COND
                                ((fetch (BF RESIDUAL) of SCANPTR)
                                 (EQ SCANPTR ORIG))
                                (T (AND (fetch (BF CHECKED) of SCANPTR)
                                        (EQ ORIG (fetch (BF IVAR) of SCANPTR]
                     (add SCANPTR WORDSPERCELL)))
          (GO LP])

)

;; setting up stack from scratch

(DEFINEQ

(CLEARSTK
  [LAMBDA (FLG)                                                         (* bvm%: " 5-Feb-85 16:29")
    (PROG (LST)
          [\MAPMDS \STACKP (FUNCTION (LAMBDA (PAGE)
                                     (PROG ((I 0)
                                            (PTR (create POINTER
                                                         PAGE# _ PAGE))
                                            FX)
                                       LPE [COND
                                              ((AND (EQ (fetch (STACKP STACKP0) of PTR)
                                                        \STACKHI)
                                                    (NEQ (SETQ FX (fetch (STACKP EDFXP) of PTR))
                                                         0))
                                               (SELECTQ FLG
                                                    (NIL [COND
                                                             (NIL
                                                              ; Disallow this, we can't have this global smashing in the process
                                                              ; world
                                                                (UNINTERRUPTABLY
                                                                     (PROGN (replace (STACKP EDFXP) of PTR
                                                                               with 0)
                                                                            (\DECUSECOUNT FX)))])
                                                    (**CLEAR**
                                                              ; Called by HARDRESET
                                                              (replace (STACKP EDFXP) of PTR with 0))
                                                    (push LST PTR]
                                            (COND
                                              ((NEQ (SETQ I (IPLUS I WORDSPERCELL))
                                                    \MDSIncrement)
                                               (SETQ PTR (\ADDBASE PTR WORDSPERCELL))
                                               (GO LPE]
          (RETURN LST])

(HARDRESET
  [LAMBDA NIL                                                           (* bvm%: "12-JAN-82 12:06")
    ;; this is what Raid's ^D does
    (\CONTEXTSWITCH \ResetFXP])

(RELSTK
  [LAMBDA (POS)                                                         (* lmm "27-JUL-81 09:42")
    [AND (STACKP POS)
         (PROG ((FX (fetch EDFXP of POS)))
               (COND
                  ((NEQ FX 0)
                   (UNINTERRUPTABLY
                       (\DECUSECOUNT FX)
                       (replace EDFXP of POS with 0))]
    POS])

(RELSTKP
  [LAMBDA (X)                                                           ; Edited 10-Nov-87 17:39 by bvm
    (AND (STACKP X)
         (LET ((FRAME (fetch EDFXP of X)))
              ;; Test for stack pointer released explicitly, or if somebody has already returned to/around the frame in question (in which case my
              ;; clink is zero, but that's ok for T).
              (OR (EQ FRAME 0)
                  (AND (fetch (FX INVALIDP) of (fetch (FX CLINK) of FRAME))
                       (NEQ (fetch (FX FRAMENAME) of FRAME)
```

```
                          T])

)

(DEFINEQ
```

(SETUPSTACK

```
  [LAMBDA (INITFLG)                                                    (* lmm "22-JUN-83 15:08")
    ;; INITFLG is on if coming from MAKEINIT.  Kludge because fn definitions are not available during MAKEINIT
    (CREATEPAGES \STACKSPACE (IQUOTIENT \InitStackSize WordsPerPage)
          NIL T)                                                       ; create initial stack pages
    (\SETUPGUARDBLOCK 0 WORDSPERCELL)                                  ; start stack with mini-guard block
    (replace (IFPAGE CurrentFXP) of \InterfacePage with (\SETUPSTACK1 WORDSPERCELL 0 0 (IDIFFERENCE \StackAreaSize
                                                                                              2)
                                                          0 RESETPC RESETPTR NIL INITFLG))
    (replace (IFPAGE ResetFXP) of \InterfacePage with 0)
    (replace (IFPAGE FAULTFXP) of \InterfacePage with 0)
    (replace (IFPAGE SubovFXP) of \InterfacePage with 0)
    (replace (IFPAGE KbdFXP) of \InterfacePage with 0)
    (\SETUPGUARDBLOCK (IDIFFERENCE \StackAreaSize 2)
          2)
    (replace (IFPAGE StackBase) of \InterfacePage with (\SETUPGUARDBLOCK \StackAreaSize (IDIFFERENCE
                                                                                          (IDIFFERENCE
                                                                                                \InitStackSize
                                                                                                \StackAreaSize)
                                                                                          2)))
    (replace (IFPAGE EndOfStack) of \InterfacePage with (\SETUPGUARDBLOCK (IDIFFERENCE \InitStackSize 2)
                                                          2])
```

(\SETUPSTACK1

```
  [LAMBDA (STKP ALINK CLINK STKEND NARGS PC DEFPTR ARGS INITFLG ARGSLENGTH)
                                                                       ; Edited  6-Apr-88 18:34 by rtk
    (COND
       ([OR INITFLG (IGREATERP (IDIFFERENCE STKEND STKP)
                               (IPLUS (PROG1 (fetch (FNHEADER STKMIN) of DEFPTR)
                                                                       ; Space needed to call this fn

                                             )
                                      (PROG1 WORDSPERQUAD              ; Extra slop
                                             ]                         ; Don't build a frame if there isn't space!
        (PROG ((SP STKP))
              (if ARGSLENGTH
                  then (SETQ ARGSLENGTH (MIN ARGSLENGTH NARGS))
                       (\BLT (ADDSTACKBASE SP)
                             ARGS
                             (UNFOLD ARGSLENGTH WORDSPERCELL))
                       (add SP (TIMES ARGSLENGTH WORDSPERCELL))
                       (SETQ ARGS))
              (FRPTQ NARGS (PUTBASEPTR \STACKSPACE SP (AND ARGS (pop ARGS)))
                                                                       ; store args
                     (add SP WORDSPERCELL))
              (AND (PROG1 (COND
                            ((ODDP SP WORDSPERQUAD)
                             (PUTBASEPTR \STACKSPACE SP NIL)           ; Clear out the padding word
                             (add SP WORDSPERCELL)
                             T))
                          (replace (STK FLAGWORD) of SP with \STK.BF.WORD))
                   (replace (BF PADDING) of SP with 1))
              (replace (BF IVAR) of SP with STKP)
              (SETQ STKP (IPLUS SP WORDSPERCELL))
              (replace (FX FLAGBYTE) of STKP with (CONSTANT (CL:READ-FROM-STRING "#B11000001")))
        ;; flag byte has 110 = fx, fast=nil, native=nil, incall=nil, validnametable=nil, nopush=t
              (replace (FX USECNT) of STKP with 0)
              (replace (FX %#BLINK) of STKP with SP)
              (replace (FX %#ALINK) of STKP with (IPLUS ALINK \#ALINK.OFFSET 1))
              (replace (FX %#CLINK) of STKP with (IPLUS CLINK \#ALINK.OFFSET))
              (replace (FX FNHEADER) of STKP with DEFPTR)
              (replace (FX PC) of STKP with PC)
              (SETQ SP (fetch (FX FIRSTPVAR) of STKP))
              [COND
                 ((NOT INITFLG)                                        ; function definitions not available during MAKEINIT
                  (RPTQ (UNFOLD (ADD1 (fetch (FNHEADER PV) of DEFPTR))
                                CELLSPERQUAD)
                        (PROGN                                         ; Fill in Pvar region with 'unbound'
                               (\PUTBASE \STACKSPACE SP 65535)
                               (add SP 2]
              (replace (FX NEXTBLOCK) of STKP with (add SP (fetch (FX PADDING) of STKP)))
                                                                       ; Need extra junk quad after the (null) pvar region
              (\MAKEFREEBLOCK SP (IDIFFERENCE STKEND SP))
              (RETURN STKP])
```

(\MAKEFRAME

```
  [LAMBDA (FN ST END ALINK CLINK ARGS ARGLOCN)                         (* lmm " 5-Feb-86 14:44")
```

```
    (CHECK (fetch (LITATOM CCODEP) of FN))
    (PROG ((DEF (fetch (LITATOM DEFPOINTER) of FN)))
          (RETURN (\SETUPSTACK1 ST ALINK CLINK END (COND
                                                     ((fetch (FNHEADER LSTARP) of DEF)
                                                      0)
                                                     (T (fetch (FNHEADER NA) of DEF)))
                                (fetch (FNHEADER STARTPC) of DEF)
                                DEF ARGS NIL ARGLOCN])
```

(\RESETSTACK

```
[LAMBDA NIL                                                    ; Edited 25-Jan-90 15:18 by jds
```

;; Do Hard reset. We get here only by a (\CONTEXTSWITCH \ResetFXP).

```
(PROG NIL
  LP  (\RESETSTACK0)
      (\CONTEXTSWITCH \ResetFXP)
      (GO LP])
```

(\RESETSTACK0

```
[LAMBDA NIL                                                    ; Edited  9-Dec-86 14:05 by bvm
  (PROG [(BASE \StackAreaSize)
         (RPROC (AND \RUNNING.PROCESS (NULL \NEED.HARDRESET.CLEANUP]
        [if RPROC
            then
```

;; Save frame of current process at time of hard reset. All other process frames are stored ok in process handles.
;; \RUNNING.PROCESS is NIL at the beginning of the world or if process world not turned on. Want a frame in user space, not
;; system context space, since only the user stack has interesting bindings to gather.

```
                 (SETQ \SAVED.USER.CONTEXT (for I from \ResetFXP to \FAULTFXP
                                              bind TMP (OLDBASE _ (fetch (IFPAGE StackBase) of \InterfacePage))
                                              when (< OLDBASE (SETQ TMP (\GETBASE \InterfacePage I)))
                                              do (RETURN TMP)
                                              finally                ; The newer contexts aren't as nice about living in consecutive
                                                                     ; locations--have to expand out
                                                     (RETURN (if (< OLDBASE (SETQ TMP (fetch (IFPAGE TELERAIDFXP)
                                                                                         of \InterfacePage)))
                                                                 then TMP
                                                                elseif (< OLDBASE (SETQ TMP (fetch (IFPAGE MiscFXP)
                                                                                               of \InterfacePage)))
                                                                 then TMP]
        (replace (IFPAGE FAULTFXP) of \InterfacePage with (\MAKEFRAME (FUNCTION \FAULTHANDLER)
                                                                      BASE
                                                                      (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                      0 0))
        (replace (IFPAGE HardReturnFXP) of \InterfacePage with (\MAKEFRAME (FUNCTION \DOHARDRETURN)
                                                                           BASE
                                                                           (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                           0 0))
        (replace (IFPAGE TELERAIDFXP) of \InterfacePage with (\MAKEFRAME (COND
                                                                            ((fetch (LITATOM CCODEP)
                                                                                of (FUNCTION \DOTELERAID))
                                                                             (FUNCTION \DOTELERAID))
                                                                            (T (FUNCTION \DUMMYTELERAID)))
                                                                          BASE
                                                                          (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                          0 0))
```

;; NOTE: Anything below the key handler is considered super uninterruptable

```
        (replace (IFPAGE KbdFXP) of \InterfacePage with (\MAKEFRAME (COND
                                                                      ((fetch (LITATOM CCODEP)
                                                                          of '\KEYHANDLER)
                                                                       (FUNCTION \KEYHANDLER))
                                                                      (T '\DUMMYKEYHANDLER))
                                                                    (SETQ \KBDSTACKBASE BASE)
                                                                    (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                    0 0))
        (replace (IFPAGE GCFXP) of \InterfacePage with (\MAKEFRAME (FUNCTION \DOGC)
                                                                   BASE
                                                                   (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                   0 0))
        (replace (IFPAGE SubovFXP) of \InterfacePage with (\MAKEFRAME (FUNCTION \DOSTACKOVERFLOW)
                                                                      BASE
                                                                      (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                      0 0))
        (replace (IFPAGE MiscFXP) of \InterfacePage with (\MAKEFRAME (FUNCTION \DOMISCAPPLY)
                                                                     (SETQ \MISCSTACKBASE BASE)
                                                                     (SETQ BASE (IPLUS BASE \StackAreaSize))
                                                                     0 0))
        (replace (IFPAGE StackBase) of \InterfacePage with BASE) ; StackBase distinguishes system contexts from user stack
        (replace (IFPAGE ResetFXP) of \InterfacePage with (if RPROC
                                                              then
```

;; We now have enough stack set up that we can fault. Do stack scan for hardreset cleanup in the
;; stackoverflow context (hopefully not used during same!). \NEED.HARDRESET.CLEANUP is flag to
;; stackoverflow context to do something special. It also saves the address at which to start building
;; user stack when finished. We leave old EndOfStack alone until \SETUPUSERSTACK runs and
;; possibly adjusts it.

```
                                                    (SETQ \NEED.HARDRESET.CLEANUP BASE)
                                                    (if (type? FSB BASE)
                                                        then
                                                ; have to make sure that last context has valid end, one that
                                                ; microcode won't try to merge
                                                            (\SETUPGUARDBLOCK BASE 2))
                                                    (fetch (IFPAGE SubovFXP) of \InterfacePage)
                                                else (\SETUPUSERSTACK BASE])
```

(\SETUPUSERSTACK

```
[LAMBDA (BASE)                                          ; Edited  5-Apr-90 19:22 by jds
```

;; Create initial base of user stack starting in stack space at location BASE. Return the resulting FX.

```
(PROG1 (\MAKEFRAME (FUNCTION \CODEFORTFRAME)
                   BASE
                   (PROGN [COND
                              ((IGREATERP (SETQ BASE (fetch (IFPAGE EndOfStack) of \InterfacePage))
                                          \InitStackSize)
                               ;; Trim stack back, unlocking pages.  This way you don't permanently lock entire stack segment if you get a stack
                               ;; overflow
                               [\UNLOCKPAGES (ADDSTACKBASE \InitStackSize)
                                             (ADD1 (IDIFFERENCE (FOLDLO BASE WORDSPERPAGE)
                                                                (FOLDLO \InitStackSize WORDSPERPAGE]
                               (replace (IFPAGE EndOfStack) of \InterfacePage with (SETQ BASE (IDIFFERENCE
                                                                                                \InitStackSize
                                                                                                2]
                          BASE)
                   0 0)
       (\SETUPGUARDBLOCK BASE 2)
       (SETQ \NEED.HARDRESET.CLEANUP NIL)
```

;; If we're coming up in the INIT, maybe need to do MAIKO MOVDs NOW:

```
       (AND \DOFAULTINIT (EQ (fetch MachineType of \InterfacePage)
                             \MAIKO)
            (\CONTEXTSWITCH \FAULTFXP))
       (SETQ \RECLAIM.COUNTDOWN \RECLAIMMIN)                ; reenable gc

       )])
```

(\SETUPGUARDBLOCK

```
[LAMBDA (STKP LEN)                                      (* lmm "27-JUL-81 09:34")
  (replace (FSB FLAGWORD) of STKP with \STK.GUARD.WORD)
  (replace (FSB SIZE) of STKP with LEN)
  STKP])
```

(\MAKEFREEBLOCK

```
[LAMBDA (STK SIZE)                                      (* lmm "27-JUL-81 09:33")
  (PROGN                                                ; must be careful here, because stack is inconsistent in this
                                                        ; region
         (replace (FSB SIZE) of STK with SIZE)
         (replace (FSB FLAGWORD) of STK with \STK.FSB.WORD])
```

(\REPEATEDLYEVALQT

```
[LAMBDA NIL                                             (* lmm "10-JUN-81 16:41")
  (PROG ((\INTERRUPTABLE T))
    LP  (\RESETSYSTEMSTATE)
        (EVALQT)
        (GO LP])
```

(\DUMMYKEYHANDLER

```
[LAMBDA NIL                                             (* lmm " 4-APR-82 21:47")
```

;; installed instead of KEYHANDLER by RESETSTACK when KEYHANDLER is not CCODEP, e.g. inside MICROTEST where LLKEY is not loaded

```
  (PROG NIL
    LP  (\CONTEXTAPPLY \KbdFXP (FUNCTION \CAUSEINTERRUPT)
                       \KbdFXP)
        (\CONTEXTSWITCH \KbdFXP)
        (GO LP])
```

(\DUMMYTELERAID

```
[LAMBDA NIL                                             (* bvm%: "14-MAR-83 22:09")
  (PROG NIL
    LP  (\CONTEXTSWITCH \TeleRaidFXP)
        (GO LP])
```

(\CAUSEINTERRUPT

```
[LAMBDA (CNTXT FN)                                      (* bvm%: " 6-APR-83 15:40")
```

;; Builds a frame for FN (default is \INTERRUPTFRAME) on top of the fx in the CNTXT slot of interface page, returning T on success

```
    (PROG ((FRAME (\GETBASE \InterfacePage CNTXT))
           NXT)
          (COND
             ((ILESSP FRAME (fetch (IFPAGE StackBase) of \InterfacePage))
                 ;; I can't actually test \INTERRUPTABLE, because that might fault!  I assume that any system context that lives is uninterruptable.
                 ;; This is mainly so I don't build an \INTERRUPTED frame on top of the fault handler.  [Used to be test for context lower than the
                 ;; keyboard handler, but this is much safer.]
                 ;; You might want to allow a RAID interrupt here, but that could be VERY dangerous if a fault is in progress, so best wait.
              (RETURN)))
          (SETQ NXT (fetch (FX NEXTBLOCK) of FRAME))
          (CHECK (fetch (FX CHECKED) of FRAME)
                 (type? FSB NXT))
          (RETURN (COND
                     ((SETQ FRAME (\MAKEFRAME (OR FN (FUNCTION \INTERRUPTFRAME))
                                              NXT
                                              (IPLUS NXT (fetch (FSB SIZE) of NXT))
                                              FRAME FRAME))
                      (\PUTBASE \InterfacePage CNTXT FRAME)
                      T])
```

(\CONTEXTAPPLY

```
  [LAMBDA (CNTXT FN ARG)                                    (* lmm "13-OCT-81 10:01")
    (PROG ((MYALINK (\MYALINK)))
          (\SMASHLINK NIL (GETBASE \InterfacePage CNTXT))
          (RETURN (PROG1 (SPREADAPPLY* FN ARG)
                         (\SMASHLINK NIL MYALINK])
```

(\INTERRUPTFRAME

```
  [LAMBDA NIL                                               ; Edited 30-Jan-91 00:23 by jds
    (COND
       (WINDFLG (\INTERRUPTED))
       (T (INTERRUPTED])
```

(\INTERRUPTED

```
  [LAMBDA NIL                                               (* lmm " 5-DEC-82 20:53")
    (COND
       (\INTERRUPTABLE (INTERRUPTED))
       (T                                                   ; Wrong, we weren't interruptable after all.  Tell keyboard to try
                                                            ; again later
          (SETQ \PENDINGINTERRUPT T])
```

(\CODEFORTFRAME

```
  [LAMBDA NIL                                               ; Edited 11-Jan-91 14:32 by jds
    (\CALLME 'T)
    (CLEARSTK '**CLEAR**)
    (INITIALEVALQT)
    (PROG NIL
      LP  (\REPEATEDLYEVALQT)
          (GO LP])
```

(\DOMISCAPPLY

```
  [LAMBDA NIL                                               (* bvm%: "30-NOV-82 12:28")
    (\DOMISCAPPLY1])
```

(\DOMISCAPPLY1

```
  [LAMBDA NIL                                               (* bvm%: "30-NOV-82 12:29")
```

;;; Utility context to perform selected operations in a 'safe' area of the stack. Use \MISCAPPLY* macro to 'call'.

```
    ;; The compiler emits a BIND for the SPREADAPPLY* below, hence we cannot do this at the root of the stack.  Sigh [ought to be able to now, yes?
    ;; --bvm]
    (PROG NIL
      LP  (replace (IFPAGE MISCSTACKRESULT) of \InterfacePage with (SPREADAPPLY* (fetch (IFPAGE MISCSTACKFN)
                                                                                    of \InterfacePage)
                                                                                (fetch (IFPAGE MISCSTACKARG1)
                                                                                   of \InterfacePage)
                                                                                (fetch (IFPAGE MISCSTACKARG2)
                                                                                   of \InterfacePage)))
          (\CONTEXTSWITCH \MiscFXP)
          (GO LP])

)

(RPAQ? \SAVED.USER.CONTEXT NIL)

(RPAQ? \NEED.HARDRESET.CLEANUP NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY
```

```
(GLOBALVARS \SAVED.USER.CONTEXT \NEED.HARDRESET.CLEANUP)
)
```

;; HARDRESET recovery code

```
(DEFINEQ
```

(\GATHER-CLEANUP-FORMS

```
  [LAMBDA NIL                                                    (* bvm%: "27-Sep-86 19:07")
    (for PROC in \PROCESSES bind FRAME do (SETQ FRAME (fetch PROCFX of PROC))
                                          (replace PROCHARDRESETINFO of PROC
                                             with (if (OR (NEQ FRAME 0)
                                                          (AND (EQ PROC \RUNNING.PROCESS)
                                                               (SETQ FRAME \SAVED.USER.CONTEXT)))
                                                    then (\GATHER-CLEANUP-FORMS1 FRAME)
                                                  else                ; no stack but not the running process?
                                                       'ERROR])
```

(\GATHER-CLEANUP-FORMS1

```
  [LAMBDA (FRAME)                                                ; Edited 18-Feb-91 16:12 by jds
    (bind (EOS _ (fetch (IFPAGE EndOfStack) of \InterfacePage))
          (BOS _ (fetch (IFPAGE StackBase) of \InterfacePage))
          BINDINGS IVAR FOUND-UNWIND B R NT until (fetch (FX INVALIDP) of FRAME)
       do [COND
             ([AND (< FRAME EOS)
                   (> FRAME BOS)
                   (type? FX FRAME)
                   (\VALIDADDRESSP (SETQ NT (fetch (FX NAMETABLE) of FRAME)))
                   [COND
                      ((fetch (FX FASTP) of FRAME)
                       (SETQ IVAR (fetch (FX DUMMYBF) of FRAME))   ; BF contiguous with FX, assume ok address
                       T)
                      (T                                           ; expanded out so that we only check validity of BF in the case
                                                                   ; where there's really a separate one
                         (AND (< (SETQ IVAR (fetch (FX %#BLINK) of FRAME))
                                 EOS)
                              (>= IVAR BOS)
                              (type? BF IVAR]
                   (\VALIDADDRESSP (SETQ IVAR (ADDSTACKBASE (fetch (BF IVAR) of IVAR]
                                                                   ; be careful, since stack could be screwed up
              (COND
                 ((AND FOUND-UNWIND (SETQ B (\GATHER-SPECIAL-BINDINGS FRAME NT IVAR)))
                                                                   ; Gather up bindings
                  (push BINDINGS B)))
              (SELECTQ (fetch (FNHEADER FRAMENAME) of NT)
                   (SI::*UNWIND-PROTECT*                           ; cleanup forms are stored in first ivar
                        (push BINDINGS (LIST NIL (\GETBASEPTR (ADDSTACKBASE (fetch (FX IVAR) of FRAME))
                                                              0)))
                        (SETQ FOUND-UNWIND T))
                   (\MAKE.PROCESS0                                 ; Top of process.  If has *RESETFORMS* is implicit RESETLST
                                                                   ; to take care of
                        [COND
                           ([AND (NULL FOUND-UNWIND)
                                 (SETQ R (SI::PVAR-VALUE-IN-FRAME FRAME (NEW-SYMBOL-CODE 'SI::*RESETFORMS*
                                                                                         (\ATOMPNAMEINDEX 'SI::*RESETFORMS*]
                                                                   ; act as though we saw unwind-protect, then the binding of
                                                                   ; *RESETFORMS*, and that's all
                            (RETURN `(((SI::*RESETFORMS* ,R))
                                      (NIL SI::RESETUNWIND])
                   NIL)
              (SETQ FRAME (fetch (FX CLINK) of FRAME)))
             (T                                                    ; stack screwed up, so fail
                (RETURN 'ERROR]
       finally (RETURN BINDINGS])
```

(\GATHER-SPECIAL-BINDINGS

```
  [LAMBDA (FRAME NT IVAR)                                        ; Edited 18-Feb-91 14:59 by jds
```

;;; Gather up all specials bound in FRAME. NT is frame's name table, IVAR is the start of its BF. In case of duplicate names, only top value need be ;;; gathered.

```
    (AND (NEQ (fetch (FNHEADER NTSIZE) of NT)
              0)
         (for OFFSET from (fetch (FNHEADER OVERHEADWORDS) of T) by (CONSTANT (WORDSPERNAMEENTRY))
            bind TMP NAME BINDINGS CODE until (OR (NULL-NTENTRY (SETQ NAME (GETSTKNAMEENTRY NT OFFSET)))
                                                  (EQP [SETQ CODE (NTSLOT-VARTYPE
                                                                   (SETQ TMP (GETSTKNTOFFSETENTRY
                                                                              NT
                                                                              (IPLUS OFFSET (fetch (FNHEADER NTSIZE
                                                                                                   )
                                                                                             of NT]
                                                       FVARCODE))
            unless (OR (FMEMB (SETQ NAME (\INDEXATOMVAL NAME))
                              *HARDRESET-IGNORE-VARS*)
                       (ASSOC NAME BINDINGS))
```

```
            do (SELECTC CODE
                    (IVARCODE [push BINDINGS (LIST NAME (\GETBASEPTR IVAR (UNFOLD (NTSLOT-OFFSET TMP)
                                                                                  WORDSPERCELL])
                    (PVARCODE [COND
                                ([fetch (PVARSLOT BOUND) of (SETQ TMP (ADDSTACKBASE (IPLUS (fetch (FX FIRSTPVAR)
                                                                                             of FRAME)
                                                                                           (UNFOLD (NTSLOT-OFFSET
                                                                                                    TMP)
                                                                                                   WORDSPERCELL]
                                 (push BINDINGS (LIST NAME (fetch (PVARSLOT PVVALUE) of TMP])
                    (PROGN                                          ; trashed name table, bail out
                           (RETURN T)))
            finally (RETURN BINDINGS])
```

(\HARDRESET-CLEANUP

```
  [LAMBDA (PROCESS)                                                 ; Edited 21-Jan-91 14:10 by jds
```

;;; BINDINGS is a list containing all the interesting dynamic bindings of a process, intermixed with cleanup forms from UNWIND-PROTECT frames.
;;; Each element is either a list of bindings (pairs) for a single frame, or the pair (NIL cleanupfn) for an UNWIND.PROTECT. The list is in reverse order;
;;; i.e., first element corresponds to bottom of stack, and the last element is the first cleanup to run. Our task is to bind all these variables, in the
;;; appropriate order, and run the cleanup forms. Cleanup forms cannot do THROWs, because the stack is not around, just the variables.

```
    (PROG ((BINDINGS (fetch PROCHARDRESETINFO of PROCESS))
           (NVARS 0)
           (VARIABLES (LIST NIL))
           VARTAIL TABLE NNILS NTSIZE LINEARBINDINGS MASTERLIST VAR VALUE INDEX OLDVAL)
          (if (NLISTP BINDINGS)
              then                                                  ; couldn't get cleanups.  Might want to signal some error or post
                                                                    ; a warning here.
                  (RETURN))
          (replace PROCHARDRESETINFO of PROCESS with NIL)
          [SETQ TABLE (HASHARRAY (TIMES 2 (LENGTH BINDINGS]
          (SETQ VARTAIL VARIABLES)
          [for X in BINDINGS
             do (if (NULL (CAR X))
                    then                                            ; a cleanup form.  Push the set of variables bound recently, then
                                                                    ; the cleanup form
                         (if LINEARBINDINGS
                             then (push MASTERLIST X LINEARBINDINGS))
                         (SETQ LINEARBINDINGS NIL)
                  else                                              ; a list of binding pairs
                       (for PAIR in X do (SETQ VALUE (CADR PAIR))
                                         (SETQ OLDVAL (GETHASH (SETQ VAR (CAR PAIR))
                                                               TABLE))
                                                                    ; hash entries are of the form (index . values)
                                         (if (MEMB VAR LINEARBINDINGS)
                                             then                   ; a newer binding for same var with no intervening unwind
                                                                    ; overrides old binding
                                                  (RPLACA (CDR OLDVAL)
                                                          VALUE)
                                           else (push LINEARBINDINGS VAR)
                                                (if OLDVAL
                                                    then (RPLACD OLDVAL (CONS VALUE (CDR OLDVAL)))
                                                  else [SETQ VARTAIL (CDR (RPLACD VARTAIL (CONS VAR NIL]
                                                       (PUTHASH VAR (LIST (SETQ INDEX (add NVARS 1))
                                                                          VALUE)
                                                                TABLE]
          (RETURN (.CALLAFTERPUSHINGNILS. (SETQ NNILS (+ NVARS (SETQ NTSIZE (CEIL [ADD1 (UNFOLD NVARS
                                                                                                (CONSTANT (
                                                                                                WORDSPERNAMEENTRY
                                                                                                          ]
                                                                                       WORDSPERQUAD))
                                                                              (FOLDHI (fetch (FNHEADER OVERHEADWORDS) of T)
                                                                                      WORDSPERCELL)
                                                                              (SUB1 CELLSPERQUAD)))
                                          (\HARDRESET-CLEANUP1 NNILS NVARS NTSIZE MASTERLIST (CDR VARIABLES)
                                                 TABLE])
```

(\HARDRESET-CLEANUP1

```
  [LAMBDA (NNILS NVARS NTSIZE MASTERLIST VARIABLES TABLE)           ; Edited 30-Jan-91 19:05 by jds
```

;;; Construct a name table in caller consisting of the bindings specified by args. NNILS is the number of NILs pushed onto the end of frame, to be used
;;; for the bindings themselves and for a name table. NVARS is the number of vars to bind. NTSIZE is the size of the name table in cells. Thus NNILS
;;; = NVARS+NTSIZE+name table overhead.

;;; The variables and bindings themselves are given by the remaining args. VARIABLES is a list of length NVARS containing the variable names.
;;; TABLE is a hash table mapping each variable name to a list (index . bindings), where index is the position of the var in VARIABLES (first = 1) and
;;; bindings is a list of values for the binding, most recent one first. MASTERLIST is a list whose elements alternate between a cleanup specification in
;;; the form (NIL cleanupFn) and a list of variables that were bound at the time.

;;; Procedure is to bind all the variables to their most recent values. Then walk down MASTERLIST, calling the cleanup fns and "popping" the bindings
;;; of the indicated variables along the way.

```
    (LET ((CALLER (\MYALINK))
          NILSTART NT HEADER VAR0CODE PVARBASE)
```

```
        ;; Create a nametable inside CALLER where HARDRESET-CLEANUP1 pushed all those nils
        (SETQ HEADER (fetch (FX FNHEADER) of CALLER))                ; The function header of code for HARDRESET-CLEANUP
        (SETQ NT (ADDSTACKBASE (CEIL (IPLUS (SETQ NILSTART (IDIFFERENCE (fetch (FX NEXTBLOCK) of CALLER)
                                                                        (UNFOLD NNILS WORDSPERCELL)))
                                            (UNFOLD NVARS WORDSPERCELL))
                                    WORDSPERQUAD)))
        ;; NILSTART is the start of the block of NILs pushed by caller.  The first NVARS cells of it will be used for bindings.  Following that (rounded
        ;; up to quadword) comes NT, the address of our synthesized nametable.  To our caller, the whole block of NILs looks like dynamic stack,
        ;; but we will create a name table out of it that pretends the first chunk of it is PVARs.  To everyone else the distinction is immaterial.
        (SETQ VAR0CODE (SUB1 (FOLDLO (IDIFFERENCE NILSTART (fetch (FX FIRSTPVAR) of CALLER))
                                     WORDSPERCELL)))
        ;; VAR0CODE is the name table code for our "zero'th" var.  i.e., the nth var we will bind has code VAR0CODE+n, meaning it appears to be
        ;; that pvar in the frame.
        (SETQ PVARBASE (ADDSTACKBASE (IDIFFERENCE NILSTART WORDSPERCELL)))
        ;; PVARBASE is the address (PVARSLOT) in which our "zero'th" var would be stored.  i.e., the nth var we will bind is located at (\ADDBASE
        ;; PVARBASE (UNFOLD n WORDSPERCELL)).
        (UNINTERRUPTABLY
            ;; Create name table with initial contents
            (for VAR in VARIABLES as VAR# from 1 as NT1 from (fetch (FNHEADER OVERHEADWORDS) of T)
               by (CONSTANT (WORDSPERNAMEENTRY)) as NT2 from (IPLUS (fetch (FNHEADER OVERHEADWORDS) of T)
                                                                    NTSIZE)
               by (CONSTANT (WORDSPERNTOFFSETENTRY)) do (\PUTBASEPTR PVARBASE (UNFOLD VAR# WORDSPERCELL)
                                                                     (CADR (GETHASH VAR TABLE)))
                                                        (SETSTKNAME-RAW NT NT1 (\ATOMVALINDEX VAR))
                                                        (SETSTKNTOFFSET-RAW NT NT2 PVARCODE (+ VAR0CODE VAR#)))
            ;; now fix up header of NT
            (replace (FNHEADER %#FRAMENAME) of NT with '\HARDRESET-CLEANUP)
            (replace (FNHEADER NTSIZE) of NT with NTSIZE)
            (replace (FX NAMETABLE) of CALLER with NT))
        (for OP in MASTERLIST bind INFO SLOT NEXT ERRORS-SEEN
           do [COND
                 [(NULL (CAR OP))                                    ; a cleanup form
                  (COND
                     ((\HARDRESET-CLEANUP-RUN (CADR OP))
                      (SETQ ERRORS-SEEN T]
                 (T                                                  ; pop bindings
                    (for VAR in OP do (SETQ INFO (GETHASH VAR TABLE))
                                                                     ; INFO = (var# . activebindings)
                                      (COND
                                         ((NULL INFO)
                                          (HELP "HARDRESET miscalculation -- Trying to unbind var that is not
                                                bound" VAR))
                                         (T (SETQ SLOT (\ADDBASE PVARBASE (UNFOLD (CAR INFO)
                                                                                  WORDSPERCELL)))
                                            (COND
                                               ((SETQ NEXT (CDDR INFO))
                                                                     ; there is another value
                                                (RPLACD INFO NEXT)
                                                (replace (PVARSLOT PVVALUE) of SLOT with (CAR NEXT)))
                                               (T                    ; no more values, so unbind it
                                                  (REMHASH VAR TABLE)
                                                  (replace (PVARSLOT BOUND) of SLOT with NIL]
           finally (RETURN (COND
                              (ERRORS-SEEN 'ERROR)
                              (T T])
```

(**HARDRESET-CLEANUP-RUN**

```
  [LAMBDA (CLEANUPFN)                                                ; Edited  1-Jun-88 17:03 by bvm
    ;; Actually call a cleanup function.  Return T if it caused an error, NIL otherwise.  This is a separate fn so that the vars it binds and refers to are not
    ;; cached inside the caller, who wants to be able to bind and unbind at will.
    (HANDLER-BIND [(CL:ERROR (FUNCTION (LAMBDA (C)
                                       (RETFROM '\HARDRESET-CLEANUP-RUN T]
           (CL:FUNCALL CLEANUPFN)
           NIL])

)

(RPAQQ *HARDRESET-IGNORE-VARS* (SI::*CLEANUP-FORMS* SI::*DUMMY-FOR-CATCH* SI::*CATCH-RETURN-FROM*
                                SI::*CATCH-RETURN-TO* *FORM* *ARGVAL* *FN* *TAIL* *FIRSTTAIL* \INTERNAL
                                \INTERRUPTABLE SI::*NLSETQFLAG* *PROCEED-CASES*))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *HARDRESET-IGNORE-VARS*)
)

;; Ufns for RETCALL

(DEFINEQ
```

(\DORETCALL

```
  [LAMBDA (NARGS RETURNER)                                          (* lmm " 5-Feb-86 15:58")
    (LET* [(RCFRAME (fetch (IFPAGE MiscFXP) of \InterfacePage))
           (RETURNER (fetch (FX CLINK)
                            RCFRAME))
           [FN (\VAG2 0 (LET ((PC (fetch (FX PC)
                                         RETURNER))
                              (FNHEADER (fetch (FX FNHEADER)
                                               RETURNER)))
                             (LOGOR (LSH (\GETBASEBYTE FNHEADER PC)
                                         8)
                                    (\GETBASEBYTE FNHEADER (ADD1 PC]
           (RETURNEE (fetch (FX CLINK)
                            RETURNER))
           (ARGLOC (DIFFERENCE (fetch (FX NEXTBLOCK)
                                      RETURNER)
                               (UNFOLD NARGS WORDSPERCELL]
          (CHECK (EQ (fetch (FX FNHEADER FRAMENAME) of RCFRAME)
                     '\RETCALL)
                 (AND (LITATOM FN)
                      (CCODEP FN))
                 (fetch (FX CHECKED)
                        RCFRAME)
                 (fetch (FX CHECKED)
                        RETURNER)
                 (fetch (FX CHECKED)
                        RETURNEE))
          (\INCUSECOUNT RETURNEE)
          (\DECUSECOUNT RCFRAME)
          (replace (IFPAGE MiscFXP) of \InterfacePage with (LET ((START (\FREESTACKBLOCK 1024 RETURNEE)))
                                                             (OR (\MAKEFRAME FN START
                                                                     (PLUS START (fetch (FSB SIZE)
                                                                                        START))
                                                                     RETURNEE RETURNEE (ADDSTACKBASE ARGLOC)
                                                                     NARGS)
                                                                 (RAID "couldn't make a frame"])
```

(\RETCALL

```
  [LAMBDA (NARGS)                                                   (* lmm " 5-Feb-86 15:05")
    (\MISCAPPLY* '\DORETCALL NARGS])

)

(RPAQ? STACKTESTING T)
```

;; Stack overflow handler

```
(DEFINEQ
```

(\DOSTACKFULLINTERRUPT

```
  [LAMBDA NIL                                                       (* bvm%: " 4-Nov-85 17:34")
    (replace STACKOVERFLOW of \INTERRUPTSTATE with NIL)
    (RESETLST
        (RESETSAVE NIL (LIST (FUNCTION \CLEANUP.STACKFULL)))
        (STACK.FULL.WARNING T))])
```

(STACK.FULL.WARNING

```
  [LAMBDA (FLG)                                                     (* bvm%: " 4-Nov-85 18:11")
    (DECLARE (SPECVARS FLG))                                        ; Otherwise compiler optimizes this away
    (COND
       (FLG ;; True on call from \DOSTACKFULLINTERRUPT and NIL after we get into break.  This way user can say OK to resume computation
            (SETQ FLG NIL)
            (PROG ((HELPFLAG 'BREAK!))
                  (LISPERROR "STACK OVERFLOW" NIL T])
```

(\CLEANUP.STACKFULL

```
  [LAMBDA NIL                                                       (* bvm%: " 5-Nov-85 11:22")
```

;;; On a RESETSAVE around the stack full break, so that ^ or ^D from the break will do a HARDRESET

```
    (COND
       ((SELECTQ AUTOHARDRESETFLG
             (NIL NIL)
             ((ERROR RESET)
                  (EQ RESETSTATE AUTOHARDRESETFLG))
             (SELECTQ RESETSTATE
                  ((ERROR RESET)
                       T)
                  NIL))
        (SETQ \STACKOVERFLOW)
        (HARDRESET])
```

```
)

(RPAQ? \PENDINGINTERRUPT )

(RPAQ? \STACKOVERFLOW )

(RPAQ? AUTOHARDRESETFLG T)

(ADDTOVAR RESETFORMS (SETQ \STACKOVERFLOW))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS AUTOHARDRESETFLG)
)

(DECLARE%: DONTCOPY

(ADDTOVAR INEWCOMS
          (FNS SETUPSTACK \SETUPSTACK1 \SETUPGUARDBLOCK \MAKEFREEBLOCK)
          (ALLOCAL (ADDVARS (LOCKEDFNS \RESETSTACK0 \MAKEFRAME \SETUPSTACK1 \MAKEFREEBLOCK \FAULTHANDLER
                                       \KEYHANDLER \DUMMYKEYHANDLER \DOTELERAID \DUMMYTELERAID \DOHARDRETURN \DOGC
                                       \CAUSEINTERRUPT \INTERRUPTFRAME \CODEFORTFRAME \DOSTACKOVERFLOW \UNLOCKPAGES
                                       \DOMISCAPPLY)
                            (LOCKEDVARS \InterfacePage \DEFSPACE \STACKSPACE \KBDSTACKBASE \MISCSTACKBASE
                                   \SAVED.USER.CONTEXT \RUNNING.PROCESS \NEED.HARDRESET.CLEANUP))))

(ADDTOVAR EXPANDMACROFNS ADDSTACKBASE STACKADDBASE)


(ADDTOVAR DONTCOMPILEFNS SETUPSTACK)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA SI::INTERNAL-THROW-VALUES SI::INTERNAL-THROW SI::NON-LOCAL-RETURN-VALUES SI::NON-LOCAL-RETURN)
)
```

FUNCTION INDEX

CLEARSTK 17
HARDRESET 17
SI::INTERNAL-THROW 13
SI::INTERNAL-THROW-VALUES 13
SI::NON-LOCAL-GO 12
SI::NON-LOCAL-RETURN 12
SI::NON-LOCAL-RETURN-VALUES 13
SI::PVAR-VALUE-IN-FRAME 14
RELSTK 17
RELSTKP 17
SETUPSTACK 18
STACK.FULL.WARNING 25
SI::UNWIND 14
SI::UNWIND-TO-BLIP 13
SI::VARIABLE-NAME-IN-FRAME 14
\ARG0 5
\CAUSEINTERRUPT 20
\CLEANUP.STACKFULL 25
\CODEFORTFRAME 21
\CONTEXTAPPLY 21
\COPY.N.UFN 11
\DECUSECOUNT 9
\DISCARDFRAME 15
\DOGC 6
\DOGC1 6
\DOHARDRETURN 6
\DOHARDRETURN1 6
\DOMISCAPPLY 21
\DOMISCAPPLY1 21
\DORETCALL 25
\DOSTACKFULLINTERRUPT 25
\DOSTACKOVERFLOW 7
\DUMMYKEYHANDLER 20
\DUMMYTELERAID 20
\EXTENDSTACK 11
\FREESTACKBLOCK 10
\GATHER-CLEANUP-FORMS 22
\GATHER-CLEANUP-FORMS1 22
\GATHER-SPECIAL-BINDINGS 22
\GCSCANSTACK 16
\HARDRESET-CLEANUP 23
\HARDRESET-CLEANUP-RUN 24
\HARDRESET-CLEANUP1 23
\HARDRETURN 6
\INCUSECOUNT 9
\INTERRUPTED 21
\INTERRUPTFRAME 21
\MAKEFRAME 18
\MAKEFREEBLOCK 20
\MAKESTACKP 9
\MOVEFRAME 8
\MYARGCOUNT 5
\POP.N.UFN 12
\REPEATEDLYEVALQT 20
\RESETSTACK 19
\RESETSTACK0 19
\RETCALL 25
\SETARG0 5
\SETUPGUARDBLOCK 20
\SETUPSTACK1 18
\SETUPUSERSTACK 20
\SLOWRETURN 11
\SMASHLINK 10
\SMASHRETURN 15
\STORE.N.UFN 12
\UNWIND.UFN 12

CONSTANT INDEX

\#ALINK.OFFSET 3
\InitStackSize 4
\MAXSAFEUSECOUNT 4
\NT.FVAR 5
\NT.IVAR 5
\NT.PVAR 5
\STK.BF 4
\StackAreaSize 4
\STK.BF.WORD 4
\STK.FLAGS.SHIFT 4
\STK.FSB 4
\STK.FSB.WORD 4
\STK.FX 4
\STK.GUARD 4
\STK.GUARD.WORD 4
\STK.NOTFLAG 4

VARIABLE INDEX

HARDRESET-IGNORE-VARS 24
AUTOHARDRESETFLG 26
DONTCOMPILEFNS 26
EXPANDMACROFNS 26
INEWCOMS 26
RESETFORMS 26
STACKTESTING 25
STACKTYPES 4
\NEED.HARDRESET.CLEANUP 21
\PENDINGINTERRUPT 26
\SAVED.USER.CONTEXT 21
\STACKOVERFLOW 26

RECORD INDEX

BF 2
BINDMARKSLOT ..5
FSB 3
FVARSLOT 4
FX 2
NAMETABLESLOT .4
PVARSLOT 5
STACKCELL 5
STACKP 4
STK 3
STKTEMPSLOT ...5

MACRO INDEX

ADDSTACKBASE 3
STACKADDBASE 3
STACKGETBASE 3
STACKGETBASEPTR 3
STACKPUTBASE 4
STACKPUTBASEPTR 4
\MISCAPPLY* 4
\MYALINK 3