File created: 20-Jan-93 13:46:52 **{DSK}<python>lde>lispcore>library>CHAT.;3**

changes to: (RECORDS EMACSCOMMANDS)

previous date: 21-Dec-92 10:50:12 {DSK}<python>lde>lispcore>library>CHAT.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ CHATCOMS
       [(COMS                                                    ; CHAT typein
              (FNS CHAT CHAT.STARTUP CHAT.PROMPT.FOR.INPUT CHAT.CHOOSE.EMULATOR CHAT.SET.EMULATOR CHAT.INIT
                   FIND.CHAT.PROTOCOL CHAT.TYPEIN CHAT.BIN CHAT.CLOSE CHAT.DEACTIVATE.WINDOW CHAT.CLOSEFN
                   CHAT.CLOSE.CONNECTION CHAT.LOGIN)
              (VARIABLES CHAT.TTY.PROCESS CHAT.HOST.TO.PROTOCOL CHAT.HOSTINFO CHAT.OSTYPES CHAT.PROTOCOL.ABBREVS
                     CHAT.ALLHOSTS CHAT.DISPLAYTYPES CHAT.FONT CHAT.IN.EMACS? CHAT.INTERRUPTS CHAT.KEYACTIONS
                     CHAT.PROTOCOLTYPES CHAT.WAIT.TIME CHAT.WINDOW.REGION CHAT.WINDOW.SIZE CHATWINDOW
                     CLOSECHATWINDOWFLG DEFAULTCHATHOST NETWORKLOGINFO)
              (PROP (VARTYPE)
                    CHAT.OSTYPES CHAT.HOSTINFO NETWORKLOGINFO CHAT.PROTOCOL.ABBREVS CHAT.PROTOCOLTYPES))
        (COMS                                                    ; CHAT streams
              (FNS ADD.CHAT.MESSAGE CHAT.LOGINFO CHAT.SENDSCREENPARAMS CHAT.SETDISPLAYTYPE CHAT.FLUSH&WAIT
                   CHAT.ENDOFSTREAMOP CHAT.OPTIONMENU))
        (COMS                                                    ; CHAT typeout
              (FNS CHAT.TYPEOUT CHAT.TYPEOUT.CLOSE CHAT.DID.RESHAPE CHAT.SCREENPARAMS))
        (COMS                                                    ; window stuff
              (FNS GETCHATWINDOW CHAT.BUTTONFN CHAT.HOLD CHAT.MENU CHAT.CLEAR.FROM.MENU CHAT.TAKE.INPUT
                   CHAT.TAKE.INPUT1 DO.CHAT.OPTION CHAT.RECONNECT CHAT.RECONNECT.OFF CHAT.RESHAPEWINDOW
                   CHAT.TTYENTRYFN CHAT.TTYEXITFN CHAT.TYPESCRIPT CHAT.TYPESCRIPT1))
        [COMS                                                    ; for dialouts
              (FNS CHAT.CHOOSE.PHONE.NUMBER)
              (INITVARS (CHAT.PHONE.NUMBER.MENU)
                     (CHAT.PHONE.NUMBERS '(Other]
        (COMS                                                    ; for EMACS
              (FNS CHAT.EMACS.MOVE CHAT.SWITCH.EMACS)
              (VARIABLES CHAT.EMACSCOMMANDS))
        (COMS (FNS CHAT.ICONFN)
              (BITMAPS TTYKBD TTYKBDMASK)
              (VARS TTYKBDICONSPECREGION)
              (INITVARS (TTYKBDICONSPEC)))
        (ADDVARS (CHATMENUITEMS))
        (INITVARS (CHATMENU)
                                                                 ; Cached menu variables
               (CHAT.REOPENMENU)
               (CHAT.HOSTMENU)
               (CHATWINDOWLST)
               (CHAT.DRIVERTYPES)
               (CHATDEBUGFLG))
        (DECLARE%: EVAL@COMPILE DONTCOPY (LOCALVARS . T)
               (FILES (SOURCE)
                      CHATDECLS)
               (RECORDS EMACSCOMMANDS)
               (GLOBALVARS CHATMENUITEMS))
        (INITVARS (INVERTWINDOWFN 'INVERTW))
        (COMS (FNS \SPAWN.CHAT)
              (DECLARE%: DONTEVAL@LOAD DOCOPY [ADDVARS (BackgroundMenuCommands
                                                        ("Chat" '(\SPAWN.CHAT)
                                                               "Runs a new CHAT process; prompts for host"
                                                               (SUBITEMS ("No Login" '(\SPAWN.CHAT 'NONE)
                                                                                "Runs CHAT without doing
                                                                                automatic login"]
                     (P (SETQ BackgroundMenu))
                     (FILES DMCHAT)
                                                                 ; need DMCHAT since it's the default emulator
                     (INITRECORDS CHAT.STATE])

;; CHAT typein

(DEFINEQ

(CHAT
  [LAMBDA (HOST LOGOPTION INITSTREAM WINDOW FROMMENU)            ; Edited 15-Feb-90 11:34 by bvm
    (LET (SUCCESS RESULT PROC)
         [if [AND [OR HOST (SETQ HOST (COND
                                        ([AND FROMMENU
                                              (OR CHAT.HOSTMENU
                                                  (LET ((HOSTS CHAT.ALLHOSTS))
                                                       [if [AND DEFAULTCHATHOST (NOT (CL:MEMBER DEFAULTCHATHOST
                                                                                                CHAT.ALLHOSTS :TEST
                                                                                                'STRING-EQUAL]
```

```
                                                    then (SETQ HOSTS (SORT (CONS DEFAULTCHATHOST HOSTS)
                                                                           (FUNCTION UALPHORDER]
                                                (if HOSTS
                                                    then (SETQ CHAT.HOSTMENU (create MENU
                                                                                     ITEMS _
                                                                                     (APPEND HOSTS
                                                                                             '(Other))
                                                                                     TITLE _ "Host"]
                                                                                     ; From background menu, and there are some hosts to choose
                                                                                     ; from, so offer menu
                                         (MENU CHAT.HOSTMENU))
                                        (DEFAULTCHATHOST)
                                        (T                                           ; No hosts, no default--fall thru and prompt user
                                           'Other]
                    (OR (NEQ HOST 'Other)
                        (SETQ HOST (if FROMMENU
                                       then                                          ; Doing a mouse interaction
                                            (CHAT.PROMPT.FOR.INPUT "Chat to host: " (AND WINDOW (GETPROMPTWINDOW
                                                                                                WINDOW)))
                                   else (PROMPTFORWORD "
                                                Host: " NIL "Enter name of host to chat to, or <cr> to abort" NIL
                                                NIL 'TTY (CHARCODE (CR]
                    then                                                             ; Have a host--get the process started.  Want to get this proc
                                                                                     ; going as soon as possible so we can give it the tty
                         (SETQ PROC (ADD.PROCESS `(CHAT.STARTUP ',HOST ',LOGOPTION ',INITSTREAM ',WINDOW
                                                                ',FROMMENU)
                                                 'RESTARTABLE
                                                 'NO))
                         (do                                                         ; Wait for it to open or fail
                             (if (NOT (PROCESSP PROC))
                                 then (RETURN (SETQ RESULT (PROCESS.RESULT PROC)))
                               elseif (PROCESSPROP PROC 'CHAT.STARTUP)
                                 then (RETURN (SETQ RESULT (SETQ SUCCESS HOST)))
                                 else (BLOCK 1000]
                (COND
                   ((AND (NOT SUCCESS)
                         WINDOW
                         (WINDOWPROP WINDOW 'CHATHOST))                          ; Window not useable, let it reconnect
                    (WINDOWPROP WINDOW 'BUTTONEVENTFN (FUNCTION CHAT.RECONNECT))
                    (REMOVEPROMPTWINDOW WINDOW)))
                RESULT])
```

(CHAT.STARTUP

```
  [LAMBDA (HOST LOGOPTION INITSTREAM WINDOW FROMMENU)                            ; Edited  9-Dec-92 12:30 by jds
    (PROG (STREAMS RESULT OPENFN DISPLAYTYPE SLASH PROTOCOL)
          [COND
             ((OR FROMMENU CHAT.TTY.PROCESS)                                     ; Grab tty right away, not when we finally get connected.
              (TTY.PROCESS (THIS.PROCESS]
          [COND
             [[AND (SETQ SLASH (STRPOS "/" HOST))
                   (SETQ PROTOCOL (CDR (CL:ASSOC (SUBSTRING HOST (ADD1 SLASH))
                                                 CHAT.PROTOCOL.ABBREVS :TEST 'STRING-EQUAL]
                                                                                 ; Caller explicitly specified protocol to use
              (COND
                 ([NOT (SETQ OPENFN (CDR (ASSOC PROTOCOL CHAT.PROTOCOLTYPES]
                  (SETQ RESULT (CONCAT "The " PROTOCOL " Chat protocol is not loaded."))
                  (GO FAIL))
                 (T (SETQ HOST (SUBSTRING HOST 1 (SUB1 SLASH)))
                    (SETQ OPENFN (CL:FUNCALL OPENFN HOST PROTOCOL))
                    (COND
                       ((NOT OPENFN)
                        (SETQ RESULT (CONCAT HOST " is not a recognized " PROTOCOL " host."))
                        (GO FAIL]
             ((AND [SETQ PROTOCOL (CDR (CL:ASSOC HOST CHAT.HOST.TO.PROTOCOL :TEST 'STRING-EQUAL]
                   (SETQ OPENFN (CDR (ASSOC PROTOCOL CHAT.PROTOCOLTYPES)))
                   (SETQ OPENFN (CL:FUNCALL OPENFN HOST PROTOCOL)))
                                                                                 ; use protocol that worked the last time.  Clear PROTOCOL to
                                                                                 ; skip the test below for remembering it
              (SETQ PROTOCOL NIL))
             (T                                                                  ; Try all protocols
                (for PAIR in CHAT.PROTOCOLTYPES when (SETQ OPENFN (CL:FUNCALL (CDR PAIR)
                                                                             HOST))
                   do                                                            ; Value returned is (CanonicalHostName OpenFn)
                      (SETQ PROTOCOL (CAR PAIR))
                      (RETURN OPENFN]
          (COND
             ((NOT OPENFN)                                                       ; Don't know how to talk to this host
              (SETQ RESULT (CONCAT "Unknown Chat host: " HOST))
              (GO FAIL)))
          (SETQ HOST (CAR OPENFN))
          (COND
             ((NOT (CL:MEMBER HOST CHAT.ALLHOSTS :TEST 'STRING-EQUAL))
              (SETQ CHAT.ALLHOSTS (SORT (CONS HOST CHAT.ALLHOSTS)
                                        (FUNCTION UALPHORDER)))
              (SETQ CHAT.HOSTMENU)))
          (SETQ DISPLAYTYPE (CHAT.CHOOSE.EMULATOR HOST))                         ; Do this first so openfn can see the terminal type
```

```
          (COND
             ((NOT (SETQ STREAMS (CL:FUNCALL (CADR OPENFN)
                                         HOST
                                         (fetch (CHATDISPLAYTYPE DPYNAME) of DISPLAYTYPE)
                                         LOGOPTION)))
              (RETURN "Failed")))
          [COND
             (PROTOCOL                                          ; Remember protcol that worked for next time
                    (LET [(TEM (CL:ASSOC HOST CHAT.HOST.TO.PROTOCOL :TEST 'STRING-EQUAL]
                         (COND
                            (TEM (RPLACD TEM PROTOCOL))
                            (T (push CHAT.HOST.TO.PROTOCOL (CONS HOST PROTOCOL]
          (SETQ WINDOW (GETCHATWINDOW HOST WINDOW (fetch (CHATDISPLAYTYPE DPYNAME) of DISPLAYTYPE)))
          [COND
             ((LISTP (CDR STREAMS))                             ; (instream outstream . props)
              [for TAIL on (CDDR STREAMS) by (CDDR TAIL) do     ; Handle props.  Only 1 interesting one right now.
                                                          (CASE (CAR TAIL)
                                                              (LOGOPTION (SETQ LOGOPTION (CADR TAIL)))))]
              (SETQ STREAMS (CONS (CAR STREAMS)
                                  (CADR STREAMS]
          (CHAT.INIT STREAMS WINDOW HOST DISPLAYTYPE)
          (LET [(PROC (THIS.PROCESS))
                (STATE (WINDOWPROP WINDOW 'CHATSTATE]
               (PROCESSPROP PROC 'KEYACTION (SETQ \CURRENTKEYACTION
                                           (LET [(KEY (KEYACTIONTABLE \CURRENTKEYACTION))
                                                 (OSINFO (CDR (ASSOC (GETHOSTINFO HOST 'OSTYPE)
                                                                     CHAT.OSTYPES)))
                                                 (HOSTINFO (CDR (CL:ASSOC HOST CHAT.HOSTINFO :TEST
                                                                          'STRING-EQUAL]
                                                (for PAIR in (CURRENTINTERRUPTS KEY)
                                                   when (<= (CAR PAIR)
                                                            255)
                                                   do (INTCHAR (CAR PAIR)
                                                               NIL NIL KEY))
                                                                ; turn off all interrupts in charset 0
                                                (for INTERRUPTS in (LIST CHAT.INTERRUPTS (LISTGET OSINFO
                                                                                                  :INTERRUPTS)
                                                                         (LISTGET HOSTINFO :INTERRUPTS))
                                                   do (for PAIR in INTERRUPTS
                                                         do (INTCHAR (CAR PAIR)
                                                                     NIL NIL KEY)))
                                                (for KEYACTIONS in (LIST CHAT.KEYACTIONS (LISTGET OSINFO
                                                                                                  :KEYACTIONS)
                                                                         (LISTGET HOSTINFO :KEYACTIONS))
                                                   do (for PAIR in KEYACTIONS do (KEYACTION (CAR PAIR)
                                                                                           (CDR PAIR)
                                                                                           KEY)))
                                               KEY)))
               (PROCESSPROP PROC 'NAME (CONCAT "CHAT#" HOST))
               (PROCESSPROP PROC 'CHAT.STARTUP T)               ; Signal success to CHAT, who's waiting for us to get going.
               (PROCESSPROP PROC 'TTYENTRYFN (FUNCTION CHAT.TTYENTRYFN))
               (PROCESSPROP PROC 'WINDOW WINDOW)
               (CLEARW WINDOW)
               (WINDOWPROP WINDOW 'PROCESS PROC)
               (WINDOWPROP WINDOW 'CHATHOST (CONS HOST LOGOPTION))
               [PROCESSPROP PROC 'RESTARTFORM `(CHAT.TYPEIN ',HOST ',WINDOW 'HARDRESET]
                                                                ; In case of hard reset
               (PROCESSPROP PROC 'RESTARTABLE 'HARDRESET)       ; We are now restartable

               )
          (RETURN (CHAT.TYPEIN HOST WINDOW LOGOPTION INITSTREAM))
       FAIL

      ;; Come here with RESULT set to description of failure
          (COND
             (FROMMENU (printout (COND
                                    (WINDOW (GETPROMPTWINDOW WINDOW))
                                    (T PROMPTWINDOW))
                                 T RESULT)))
          (RETURN RESULT])
```

(CHAT.PROMPT.FOR.INPUT

```
  [LAMBDA (PROMPT WINDOW MINLENGTH)                             ; Edited 15-Feb-90 14:51 by bvm
    ;; Prompt in WINDOW for a string.  If WINDOW is NIL, pop up a one-line window, allowing at least space for MINLENGTH characters of input.
    (RESETLST
        (RESETSAVE (TTY.PROCESS (THIS.PROCESS)))
        (RESETSAVE NIL
               (LIST (if WINDOW
                         then                                   ; Just clear it now and when we're done
                              (CLEARW WINDOW)
                              (FUNCTION CLEARW)
                       else                                     ; Make our own and close it on way out
                            [SETQ WINDOW (LET [(FONT (DEFAULTFONT 'DISPLAY]
                                              (CREATEW (MAKEWITHINREGION
```

```
                                                    (create REGION
                                                           LEFT _ LASTMOUSEX
                                                           BOTTOM _ LASTMOUSEY
                                                           WIDTH _ [WIDTHIFWINDOW
                                                                    (+ (STRINGWIDTH PROMPT FONT)
                                                                       (TIMES (OR MINLENGTH 40)
                                                                              (CHARWIDTH (CHARCODE A)
                                                                                         FONT]
                                                           HEIGHT _ (HEIGHTIFWINDOW (FONTPROP FONT
                                                                                              'HEIGHT]
                                (WINDOWPROP WINDOW 'PAGEFULLFN (FUNCTION NILL))
                                (FUNCTION CLOSEW))
                          WINDOW))
      (TTYINPROMPTFORWORD PROMPT NIL NIL WINDOW NIL NIL (CHARCODE (CR))))])
```

(CHAT.CHOOSE.EMULATOR

```
[LAMBDA (HOST)                                          ; Edited 25-Oct-89 17:42 by bvm
```

;; Returns a record of type CHATDISPLAYTYPE to be used for this session

```
(COND
   [(FIXP CHAT.DISPLAYTYPES)
    (COND
       (CHAT.EMULATORTYPE (create CHATDISPLAYTYPE
                                  HOST _ NIL
                                  DPYNAME _ CHAT.EMULATORTYPE
                                  DPYCODE _ CHAT.DISPLAYTYPES]
   ((LISTP CHAT.DISPLAYTYPES)
    (OR (CL:ASSOC HOST CHAT.DISPLAYTYPES :TEST 'STRING-EQUAL)
        (FASSOC NIL CHAT.DISPLAYTYPES)))
   (T (ERROR "Please set CHAT.DISPLAYTYPES to be a list of (HOST TTY-TYPE-# EMULATORTYPE)")
      NIL])
```

(CHAT.SET.EMULATOR

```
[LAMBDA (CHAT.STATE WINDOW NEWEMULATOR)                 ; Edited 20-Mar-90 15:49 by bvm
  (LET ((TYPEOUTPROC (fetch (CHAT.STATE TYPEOUTPROC) of CHAT.STATE))
        INSTREAM)
       (COND
          (NEWEMULATOR (DEL.PROCESS TYPEOUTPROC)
                 (CLEARW WINDOW)
                 (replace (CHAT.STATE TERM.STATE) of CHAT.STATE with NIL)
                 [LET [(OLDTITLE (WINDOWPROP WINDOW 'TITLE]
                      (WINDOWPROP WINDOW 'TITLE (CONCAT NEWEMULATOR (SUBSTRING OLDTITLE (STRPOS " " OLDTITLE]
                 [replace (CHAT.STATE TYPEOUTPROC) of CHAT.STATE with (ADD.PROCESS
                                                                       `(CHAT.TYPEOUT ,WINDOW
                                                                                      ',NEWEMULATOR
                                                                                      ',CHAT.STATE]
                 (if (SETQ INSTREAM (fetch (CHAT.STATE INSTREAM) of CHAT.STATE))
                     then                               ; Inform the host.
                          (CHAT.SETDISPLAYTYPE INSTREAM (for TRIPLE in CHAT.DISPLAYTYPES
                                                           when (EQ (CADDR TRIPLE)
                                                                    NEWEMULATOR)
                                                           do ; This is kind of crufty, since we don't know in general what the
                                                              ; right code is, so we just guess by taking a sample entry from
                                                              ; here
                                                              (RETURN (CADR TRIPLE)))
                                  NEWEMULATOR])
```

(CHAT.INIT

```
[LAMBDA (STREAMS WINDOW HOST DISPLAYTYPE)               ; Edited 11-Jun-90 14:37 by mitani
  (LET* [(INSTREAM (CAR STREAMS))
         (OUTSTREAM (CDR STREAMS))
         (DPYNAME (fetch (CHATDISPLAYTYPE DPYNAME) of DISPLAYTYPE))
         (STATE (create CHAT.STATE
                        RUNNING? _ T
                        CHATINEMACS _ CHAT.IN.EMACS?
                        INSTREAM _ INSTREAM
                        OUTSTREAM _ OUTSTREAM
                        WINDOW _ WINDOW
                        DSP _ (WINDOWPROP WINDOW 'DSP]
        (WINDOWPROP WINDOW 'CHATSTATE STATE)
        (COND
           [(EQ DPYNAME 'TEDIT)
            (replace (CHAT.STATE TEXTSTREAM) of STATE with (TEDITSTREAM.INIT WINDOW (FUNCTION TEDITCHAT.MENUFN]
           (T (WINDOWPROP WINDOW 'CURSORMOVEDFN NIL)
              (WINDOWPROP WINDOW 'RESHAPEFN (FUNCTION CHAT.RESHAPEWINDOW))
              (WINDOWPROP WINDOW 'BUTTONEVENTFN (FUNCTION CHAT.BUTTONFN))
              (WINDOWPROP WINDOW 'REPAINTFN NIL)
              (WINDOWPROP WINDOW 'NEWREGIONFN NIL)
              (WINDOWPROP WINDOW 'WINDOWENTRYFN 'GIVE.TTY.PROCESS)
              (WINDOWPROP WINDOW 'RIGHTBUTTONFN NIL)
              (WINDOWPROP WINDOW 'CURSOROUTFN NIL)
              (WINDOWPROP WINDOW 'SCROLLFN NIL)))
        (WINDOWADDPROP WINDOW 'CLOSEFN (FUNCTION CHAT.CLOSEFN))
        (WINDOWPROP WINDOW 'ICONWINDOW NIL)
```

```
          (WINDOWPROP WINDOW 'ICONFN (FUNCTION CHAT.ICONFN))
          (STREAMPROP INSTREAM 'OLDEOSOP (fetch (STREAM ENDOFSTREAMOP) of INSTREAM))
          (STREAMPROP INSTREAM 'DISPLAYTYPE DISPLAYTYPE)
          (replace (STREAM ENDOFSTREAMOP) of INSTREAM with (FUNCTION CHAT.ENDOFSTREAMOP])
```

(FIND.CHAT.PROTOCOL

```
  [LAMBDA (NAME)                                          ; Edited 15-Feb-90 15:00 by bvm
```

;;; Find a protocol for use by CHAT by calling the filter fns on CHAT.PROTOCOLS. The fns should return a CHAT.PROTOCOL that can be used to ;;; contact NAME or NIL.

```
    (for PAIR in CHAT.PROTOCOLTYPES bind RESULT when (SETQ RESULT (CL:FUNCALL (CDR PAIR)
                                                                      NAME))
       do (RETURN RESULT])
```

(CHAT.TYPEIN

```
  [LAMBDA (HOST WINDOW LOGOPTION INITSTREAM)                  ; Edited 15-Feb-90 12:18 by bvm
    (DECLARE (SPECVARS STREAM))                               ; so that menu can change it
    (PROG* ((THISPROC (THIS.PROCESS))
            (DEFAULTSTREAM T)
            (STATE (WINDOWPROP WINDOW 'CHATSTATE))
            (CHATSTREAM (fetch (CHAT.STATE OUTSTREAM) of STATE))
            (INSTREAM (fetch (CHAT.STATE INSTREAM) of STATE))
           STREAM CH CHATPROMPTWINDOW LOCALECHOSTREAM)
          (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (WINDOW)
                                         (AND RESETSTATE (NEQ RESETSTATE 'HARDRESET)
                                              (CHAT.CLOSE WINDOW T]
                               WINDOW))                       ; If an error occurs, or process is killed, this will flush the
                                                              ; connection etc
          [IF (NEQ LOGOPTION 'HARDRESET)
              THEN                                            ; Only do this the first time
                   (LET ((DISPLAYTYPE (STREAMPROP INSTREAM 'DISPLAYTYPE))
                         DISPLAYNAME)
                        [COND
                           (DISPLAYTYPE (CHAT.SETDISPLAYTYPE INSTREAM (fetch (CHATDISPLAYTYPE DPYCODE)
                                                                             of DISPLAYTYPE)
                                        (SETQ DISPLAYNAME (fetch (CHATDISPLAYTYPE DPYNAME) of DISPLAYTYPE
                                                                                          ]
                        (CHAT.SCREENPARAMS STATE INSTREAM WINDOW)
                        (replace (CHAT.STATE TYPEOUTPROC) of STATE
                           with (ADD.PROCESS `(CHAT.TYPEOUT ,WINDOW ',DISPLAYNAME ',STATE)
                                      'NAME
                                      'CHAT.TYPEOUT
                                      'RESTARTABLE
                                      'HARDRESET))
                        (AND (NEQ LOGOPTION 'NONE)
                             (CHAT.LOGIN HOST LOGOPTION WINDOW STATE))
                        (COND
                           (INITSTREAM (NLSETQ (SETQ STREAM (COND
                                                              ((STRINGP INITSTREAM)
                                                               (OPENSTRINGSTREAM INITSTREAM))
                                                              (T (OPENSTREAM INITSTREAM 'INPUT]
          (TTYDISPLAYSTREAM WINDOW)                           ; So that \TTYBACKGROUND flashes the caret where we expect
          (bind OUTPUTSTREAM while (EQ (fetch (CHAT.STATE RUNNING?) of STATE)
                                       T)
             do (COND
                   ((NULL STREAM)
                    (SETQ STREAM DEFAULTSTREAM)))
                (SETQ OUTPUTSTREAM (if (fetch (CHAT.STATE LOCALECHO) of STATE)
                                       then [OR LOCALECHOSTREAM (SETQ LOCALECHOSTREAM
                                                                  (CL:MAKE-BROADCAST-STREAM CHATSTREAM
                                                                         (GETSTREAM WINDOW 'OUTPUT]
                                   else CHATSTREAM))
                [COND
                   [(EQ STREAM T)
                    ;; Handle terminal specially
                    (OR (TTY.PROCESSP)
                        (\WAIT.FOR.TTY))
                    (COND
                       ((\SYSBUFP)
                        (do (SETQ CH (\GETKEY))
                            (COND
                               ((<= CH \MAXTHINCHAR)
                                (BOUT OUTPUTSTREAM CH))
                               [(EQ (LRSH CH 8)
                                    1)                        ; META char set => ascii meta
                                (BOUT OUTPUTSTREAM (LOGOR 128 (LOGAND CH 127]
                               (T                             ; Not in charset zero, not a meta. Most hosts don't
                                                              ; understand.(PRINTCCODE CH CHATSTREAM) (CHARSET
                                                              ; CHATSTREAM 0)
                                  (FLASHWINDOW WINDOW)))
                          repeatwhile (\SYSBUFP))
                        (FORCEOUTPUT CHATSTREAM]
                   (T (until (EOFP STREAM) do (BOUT OUTPUTSTREAM (\BIN STREAM)))
```

```
                    (FORCEOUTPUT CHATSTREAM)
                    (CLOSEF STREAM)
                    (SETQ STREAM)
                    (COND
                       ((SETQ CHATPROMPTWINDOW (GETPROMPTWINDOW WINDOW NIL NIL T))
                                                                  ; Indicate completion of Input if came from menu command
                        (CLEARW CHATPROMPTWINDOW]
                 (\TTYBACKGROUND))
```

;; Get here if we close connection.

```
       [SELECTQ (fetch (CHAT.STATE RUNNING?) of STATE)
            (CLOSE (CHAT.CLOSE WINDOW))
            (ABORT (CHAT.CLOSE WINDOW T))
            (NIL                                                  ; Already dead.
                 )
            (SHOULDNT (CONCAT "Unknown state in CHAT: " (fetch (CHAT.STATE RUNNING?) of STATE]
       (BLOCK])
```

(CHAT.BIN

```
  [LAMBDA (OUTSTREAM STATE)                                       (* rda%: "20-Aug-84 23:09")
    (until (\SYSBUFP) bind (FIRSTTIME _ T) do (COND
                                         (FIRSTTIME (FORCEOUTPUT OUTSTREAM)
                                                (SETQ FIRSTTIME NIL)))
                                       (\TTYBACKGROUND))
    (\GETKEY])
```

(CHAT.CLOSE

```
  [LAMBDA (WINDOW ABORTED CLOSING)                                (* lmm "24-Oct-86 16:39")
```

;; Close chat connection that is using WINDOW. Also serves as the CLOSEFN of this window, when CLOSING is NIL

```
    (DECLARE (GLOBALVARS HIGHLIGHTSHADE))
    (PROG ((CHATSTATE (WINDOWPROP WINDOW 'CHATSTATE))
           (ACTIVE? (OPENWP WINDOW))
           ICON PROC FILE KEEP)
          (DETACHALLWINDOWS WINDOW)                               ; Restore REPLACE mode for BITBLT
          (DSPOPERATION 'REPLACE WINDOW)                          ; Turn scrolling back on
          (DSPSCROLL 'ON WINDOW)
          (COND
             [CHATSTATE (DEL.PROCESS (fetch (CHAT.STATE TYPEOUTPROC) of CHATSTATE))
                    [COND
                       ((SETQ FILE (fetch (CHAT.STATE TYPESCRIPTSTREAM) of CHATSTATE))
                        (COND
                           (ACTIVE? (TERPRI WINDOW)
                                  (PRIN1 "Closing " WINDOW)
                                  (PRINT (CLOSEF FILE)
                                         WINDOW))
                           (T (CLOSEF FILE]
                    (AND ACTIVE? (\CHECKCARET WINDOW))
                    (replace (CHAT.STATE RUNNING?) of (WINDOWPROP WINDOW 'CHATSTATE NIL) with NIL)
                    (OR ABORTED (PROGN (ALLOW.BUTTON.EVENTS)
                                       (CHAT.CLOSE.CONNECTION (fetch (CHAT.STATE INSTREAM) of CHATSTATE)
                                              (fetch (CHAT.STATE OUTSTREAM) of CHATSTATE]
             (T (RETURN)))
          (SETQ CHATWINDOWLST (DREMOVE WINDOW CHATWINDOWLST))
          (SETQ PROC (WINDOWPROP WINDOW 'PROCESS NIL))
```

;; Save the process running, if any; don't do anything with it until after we close the window, if we're going to, so that windows don't flip around
;; excessively

```
          (WINDOWPROP WINDOW 'CLOSEFN NIL)                        ; Clear all CLOSE functions so that next time this chatwindow is
                                                                  ; reused it will be clean
          (COND
             [ACTIVE?                                             ; Change title to indicate closure
                    (CHAT.DEACTIVATE.WINDOW WINDOW)
                    (COND
                       ((AND (NOT (SETQ KEEP (WINDOWPROP WINDOW 'KEEPCHAT NIL)))
                             (NOT CLOSING)
                             (OR CLOSECHATWINDOWFLG (NEQ WINDOW CHATWINDOW)))
                        (CLOSEW WINDOW)))
                    [COND
                       ((EQ KEEP 'NEW)                            ; Invoked via the New command -- start up a new connection in
                                                                  ; this window
                        (ADD.PROCESS (LIST (FUNCTION CHAT)
                                           NIL NIL NIL WINDOW T]
                    (COND
                       (PROC                                      ; Do this last, because if we are PROC, DEL.PROCESS won't
                                                                  ; return
                             (DEL.PROCESS PROC]
             ((AND (SETQ ICON (WINDOWPROP WINDOW 'ICONWINDOW))
                   (OPENWP ICON))                                 ; Shade the icon if the chat window is currently closed
              (ICONW.SHADE ICON HIGHLIGHTSHADE)                   ; And arrange for middle-button to offer Reconnect option
              (WINDOWPROP ICON 'OLDBUTTONEVENTFN (WINDOWPROP ICON 'BUTTONEVENTFN (FUNCTION CHAT.RECONNECT])
```

(CHAT.DEACTIVATE.WINDOW

```
  [LAMBDA (WINDOW)                                                (* bvm%: " 4-Sep-85 19:41")
```

```
    (LET [(TITLE (WINDOWPROP WINDOW 'TITLE]
         (WINDOWPROP WINDOW 'TITLE (CONCAT (SUBSTRING TITLE 1 (IPLUS (OR (STRPOS ", height" TITLE)
                                                                         0)
                                                                     -1))
                                           ", closed"))
         (WINDOWPROP WINDOW 'BUTTONEVENTFN (FUNCTION CHAT.RECONNECT))
         (WINDOWPROP WINDOW 'EXPANDFN NIL])
```

(CHAT.CLOSEFN

```
  [LAMBDA (WINDOW)                                                    (* lmm "24-Oct-86 16:39")
```

;; Close this chat connection making sure that the window gets closed. Used as CLOSEFN of the chat window.

```
    (CHAT.CLOSE WINDOW NIL T])
```

(CHAT.CLOSE.CONNECTION

```
  [LAMBDA (INSTREAM OUTSTREAM)                                        (* rda%: "23-Aug-84 15:25")
```

;;; Close the streams for a connection if they are open.

```
    (COND
       ((OPENP INSTREAM)
        (CLOSEF INSTREAM)))
    (COND
       ((OPENP OUTSTREAM)
        (CLOSEF OUTSTREAM]))
```

(CHAT.LOGIN

```
  [LAMBDA (HOST OPTION WINDOW CHATSTATE)                              ; Edited  9-Nov-89 14:02 by bvm
```

;; Login to HOST. If a job already exists on HOST, Attach to it unless OPTION overrides.

```
    (PROG ((OSTYPE (GETOSTYPE HOST))
           (LOGINFO (GETHOSTINFO HOST 'LOGINFO))
           (STATE (WINDOWPROP WINDOW 'CHATSTATE))
           NAME/PASS COM INSTREAM OUTSTREAM)
          (OR LOGINFO (RETURN))
          (SETQ INSTREAM (fetch (CHAT.STATE INSTREAM) of STATE))
          [SETQ COM (COND
                       (OPTION)
                       ((ASSOC 'ATTACH LOGINFO)
                        (OR [CHAT.LOGINFO INSTREAM HOST (CAR (SETQ NAME/PASS
                                                               (\INTERNAL/GETPASSWORD HOST NIL NIL NIL NIL
                                                                      OSTYPE]
                            'LOGIN))
                       (T                                             ; Don't know how to do anything but login, so silly to try anything
                                                                      ; else
                          'LOGIN]
          (COND
             ((NULL (SETQ LOGINFO (ASSOC COM LOGINFO)))
              (printout PROMPTWINDOW T "Login option " COM " not implemented for this type of host"))
             (T (SETQ OUTSTREAM (fetch (CHAT.STATE OUTSTREAM) of STATE))
                (if (AND (SETQ LOGINFO (CDR LOGINFO))
                         (NULL NAME/PASS)
                         (OR (FMEMB 'USERNAME LOGINFO)
                             (FMEMB 'PASSWORD LOGINFO)))
                    then                                              ; Don't ask for password until we know we need it
                         (SETQ NAME/PASS (\INTERNAL/GETPASSWORD HOST NIL NIL NIL NIL OSTYPE)))
                (for X in LOGINFO do (SELECTQ X
                                             (CR (BOUT OUTSTREAM (CHARCODE CR))
                                                 (FORCEOUTPUT OUTSTREAM))
                                             (LF (BOUT OUTSTREAM (CHARCODE LF))
                                                 (FORCEOUTPUT OUTSTREAM))
                                             (USERNAME (PRIN3 (CAR NAME/PASS)
                                                              OUTSTREAM))
                                             (PASSWORD (PRIN3 (\DECRYPT.PWD (CDR NAME/PASS))
                                                              OUTSTREAM))
                                             (WAIT                    ; Some systems do not permit typeahead
                                                   (COND
                                                      ((NOT (CHAT.FLUSH&WAIT INSTREAM))
                                                                       ; Couldn't sync, so wait longer.
                                                       (DISMISS CHAT.WAIT.TIME)))
                                                   (DISMISS CHAT.WAIT.TIME))
                                             (PRIN3 X OUTSTREAM)))
                (FORCEOUTPUT OUTSTREAM])

)

(DEFGLOBALVAR CHAT.TTY.PROCESS T
   "If true, Chat always grabs the tty when it starts; if NIL, Chat only grabs tty when invoked by mouse
   command.")

(DEFGLOBALVAR CHAT.HOST.TO.PROTOCOL NIL
   "A-list of (host . protocol), giving preferred transport protocol (key in CHAT.PROTOCOLTYPES)")
```

```
(DEFGLOBALVAR CHAT.HOSTINFO NIL
   "A-list of (host . proplist) for Chat.  Only recognized prop for now is :KEYACTIONS.")

(DEFGLOBALVAR CHAT.OSTYPES '([UNIX :KEYACTIONS ((BS (127 127]
                                                                ; make the BS key send DEL when talking to UNIX hosts

                              )
                            "A-list of (host . proplist).  Only recognized prop is :KEYACTIONS.")

(DEFGLOBALVAR CHAT.PROTOCOL.ABBREVS NIL
   "A-list of (abbrev . protocol) for use in the host/x syntax.")

(DEFGLOBALVAR CHAT.ALLHOSTS NIL
   "List of hosts to Chat to (clear CHAT.HOSTMENU if you change this).")

(CL:DEFVAR CHAT.DISPLAYTYPES '((NIL 10 DM2500))
                             "List of triples (host code driver) telling the preferred driver (a symbol)
                             for host.  Code is numeric value for use with PupChat.  Host = NIL gives
                             default preference.")

(DEFGLOBALVAR CHAT.FONT NIL
   "Font to use in a Chat window (fixed-width is required for accurate terminal emulation)")

(CL:DEFVAR CHAT.IN.EMACS? NIL
   "Initial state of Emacs feature on opening a connection.")

(DEFGLOBALVAR CHAT.INTERRUPTS NIL
   "List of (charcode interrupt)'s of Lisp interrupts to leave enabled during Chat.")

(DEFGLOBALVAR CHAT.KEYACTIONS NIL
   "List of (keyname . actions) to set during a chat connection (see also :KEYACTION property in CHAT.OSTYPES,
   CHAT.HOSTINFO)")

(DEFGLOBALVAR CHAT.PROTOCOLTYPES NIL
   "List of (protocol . filterfn) describing possible Chat transport protocols.")

(DEFGLOBALVAR CHAT.WAIT.TIME 2000
   "Msecs to wait during the 'WAIT' part of a login")

(DEFGLOBALVAR CHAT.WINDOW.REGION NIL
   "A Lisp region in which to create the first Chat window.")

(DEFGLOBALVAR CHAT.WINDOW.SIZE NIL
   "A size (width . height) in pixels to use in prompting for Chat windows.")

(DEFGLOBALVAR CHATWINDOW NIL
   "The default window to use for Chat connection")

(DEFGLOBALVAR CLOSECHATWINDOWFLG NIL
   "If true, ALL chat windows, including the initial one, are closed on exit.")

(DEFGLOBALVAR DEFAULTCHATHOST NIL
   "Where (CHAT) with no arguments chats to.")

(DEFGLOBALVAR NETWORKLOGINFO
   [LIST '(TENEX (LOGIN "LOGIN " USERNAME " " PASSWORD "
                        ")
                 (ATTACH "ATTACH " USERNAME " " PASSWORD "
                         ")
                 (WHERE "WHERE " USERNAME CR "ATTACH " USERNAME " " PASSWORD CR))
         (LIST 'TOPS20 '(LOGIN "LOGIN " USERNAME CR PASSWORD CR)
               (LIST 'ATTACH "ATTACH " 'USERNAME (MKSTRING #\Escape)
                     'CR
                     'PASSWORD
                     'CR)
               '(WHERE "LOGIN " USERNAME CR PASSWORD CR))
         ;; use LF when logging in to unix, as SUN OS 3.X telnet servers will only accept this (4.0 accepts either).
         '(UNIX (LOGIN WAIT LF WAIT USERNAME LF WAIT PASSWORD LF))
```

```
        '(IFS (LOGIN "Login " USERNAME " " PASSWORD CR)
              (ATTACH))
        '(NS (LOGIN "Logon" CR USERNAME CR PASSWORD CR))
        '(VMS (LOGIN USERNAME CR PASSWORD CR]
   "A-list of (ostype . loginfo), where loginfo is a plist specifying what to send for different logging
   commands: LOGIN, ATTACH, or WHERE.  Each property value is a list of strings mixed with the symbols USERNAME,
   PASSWORD, WAIT, CR, LF.")

(PUTPROPS CHAT.OSTYPES VARTYPE ALIST)

(PUTPROPS CHAT.HOSTINFO VARTYPE ALIST)

(PUTPROPS NETWORKLOGINFO VARTYPE ALIST)

(PUTPROPS CHAT.PROTOCOL.ABBREVS VARTYPE ALIST)

(PUTPROPS CHAT.PROTOCOLTYPES VARTYPE ALIST)
```

;; CHAT streams

```
(DEFINEQ
```

(ADD.CHAT.MESSAGE

```
  [LAMBDA (STREAM MSG)                                          (* rda%: "22-Aug-84 18:07")
    (STREAMPROP STREAM 'MESSAGE (CONCAT (OR (STREAMPROP STREAM 'MESSAGE)
                                            "")
                                        MSG])
```

(CHAT.LOGINFO

```
  [LAMBDA (INSTREAM HOST NAME)                                  ; Edited 15-Feb-90 15:00 by bvm
```

;;; Invoke the LOGINFO method for INSTREAM, if any.

```
    (LET [(FN (STREAMPROP INSTREAM 'LOGINFO]
         (COND
            (FN (CL:FUNCALL FN HOST NAME])
```

(CHAT.SENDSCREENPARAMS

```
  [LAMBDA (INSTREAM HEIGHT WIDTH)                               ; Edited 15-Feb-90 15:01 by bvm
```

;;; Invoke the SENDSCREENPARAMS method for INSTREAM, if any.

```
    (LET [(FN (STREAMPROP INSTREAM 'SENDSCREENPARAMS]
         (AND FN (CL:FUNCALL FN INSTREAM HEIGHT WIDTH])
```

(CHAT.SETDISPLAYTYPE

```
  [LAMBDA (INSTREAM CODE NAME)                                  ; Edited 14-Feb-90 14:49 by bvm
```

;;; Invoke the SETDISPLAYTYPE method for INSTREAM. CODE is a numeric code from CHAT.DISPLAYTYPES, NAME is the driver name

```
    (LET [(FN (STREAMPROP INSTREAM 'SETDISPLAYTYPE]
         (AND FN (CL:FUNCALL FN INSTREAM CODE NAME])
```

(CHAT.FLUSH&WAIT

```
  [LAMBDA (INSTREAM)                                            ; Edited 15-Feb-90 15:02 by bvm
```

;;; Invoke the FLUSH&WAIT method for INSTREAM

```
    (LET [(FN (STREAMPROP INSTREAM 'FLUSH&WAIT]
         (AND FN (CL:FUNCALL FN INSTREAM])
```

(CHAT.ENDOFSTREAMOP

```
  [LAMBDA (STREAM)                                              ; Edited 11-Jun-90 14:37 by mitani
```

;;; Return -1 to indicate EOS to CHAT, and restore the streams EOS op incase it's needed for other things.

```
    (replace (STREAM ENDOFSTREAMOP) of STREAM with (OR (STREAMPROP STREAM 'EOSOP)
                                                       (FUNCTION \EOSERROR)))
    -1])
```

(CHAT.OPTIONMENU

```
  [LAMBDA (INSTREAM)                                            (* ejs%: "23-Jun-85 17:04")
```

;;; Apply the menu-building method for INSTREAM, if any.

```
    (LET* [(FN (STREAMPROP INSTREAM 'OPTIONMENU))
           (MENU (COND
                    ((FNTYP FN)
                     (APPLY* FN INSTREAM))
                    ((type? MENU FN)
                     FN]
```

```
          (AND MENU (fetch (MENU ITEMS)
                           MENU])

)

;; CHAT typeout

(DEFINEQ
```

(CHAT.TYPEOUT

```
  [LAMBDA (WINDOW DPYNAME CHAT.STATE)                              ; Edited 12-Aug-88 10:35 by drc:
    (bind (CNT _ 1)
          (HANDLECHARFN _ (CADR (FASSOC DPYNAME CHAT.DRIVERTYPES)))
          (INSTREAM _ (fetch (CHAT.STATE INSTREAM) of CHAT.STATE))
          (TERM.STATE _ (FETCH (CHAT.STATE TERM.STATE) of CHAT.STATE))
          (TYPEIN.PROCESS _ (WINDOWPROP WINDOW 'PROCESS))
          [OUTSTREAM _ (COND
                         ((EQ DPYNAME 'TEDIT)
                          (WINDOWPROP WINDOW 'TEXTSTREAM))
                         (T (WINDOWPROP WINDOW 'DSP]
       TYPESCRIPTSTREAM CRPENDING MSG CH first [IF (NOT TERM.STATE)
                                                   THEN          ; First time, ask terminal to get itself set up
                                                        (replace (CHAT.STATE TERM.STATE) of CHAT.STATE
                                                           with (SETQ TERM.STATE (CL:FUNCALL
                                                                                   (CADDR (FASSOC DPYNAME
                                                                                                  CHAT.DRIVERTYPES
                                                                                                  ))
                                                                                   CHAT.STATE]
                                                                                                  ; TERM.HOME CHAT.STATE

      while (IGEQ (SETQ CH (BIN INSTREAM))
                  0)
      do (while (fetch (CHAT.STATE HELD) of CHAT.STATE) do (BLOCK))
         (\CHECKCARET OUTSTREAM)
         (COND
            ((SETQ MSG (GETSTREAMPROP INSTREAM 'MESSAGE))
             (PRIN1 MSG OUTSTREAM)
             (PUTSTREAMPROP INSTREAM 'MESSAGE NIL)))             ; Print any protocol related msgs that might have come along
                                                                 ; while we where asleep
         (SPREADAPPLY* HANDLECHARFN (SETQ CH (LOGAND CH (MASK.1'S 0 7)))
               CHAT.STATE TERM.STATE)
         [COND
            ((SETQ TYPESCRIPTSTREAM (fetch (CHAT.STATE TYPESCRIPTSTREAM) of CHAT.STATE))
             (COND
                ((SELCHARQ CH
                       (CR (PROG1 CRPENDING (SETQ CRPENDING T)))
                       (LF (COND
                              (CRPENDING (\OUTCHAR TYPESCRIPTSTREAM (CHARCODE EOL))
                                                                 ; Have the typescript turn crlf into whatever it likes for eol
                                     (SETQ CRPENDING NIL))
                              (T T)))
                       (PROGN (COND
                                 (CRPENDING (\BOUT TYPESCRIPTSTREAM (CHARCODE CR))
                                        (SETQ CRPENDING NIL)))
                              T))
                 (\BOUT TYPESCRIPTSTREAM CH]
         [COND
            (CHATDEBUGFLG (COND
                             ((OR (EQ CHATDEBUGFLG T)
                                  (IGREATERP (add CNT 1)
                                         CHATDEBUGFLG))
                              (BLOCK)
                              (SETQ CNT 1]
         (COND
            ([AND (TTY.PROCESSP TYPEIN.PROCESS)
                  (OR \LONGSYSBUF (NEQ 0 (fetch (RING READ) of \SYSBUFFER]
             ;; block if there's any type ahead to make sure we see keyboard input in case the output stream never blocks.
             (BLOCK)))
      finally (SELECTQ CH
                  (-1 (CHAT.TYPEOUT.CLOSE WINDOW OUTSTREAM CHAT.STATE 'CLOSE "closed"))
                  (-2 (CHAT.TYPEOUT.CLOSE WINDOW OUTSTREAM CHAT.STATE 'ABORT "aborted"))
                  (CHAT.TYPEOUT.CLOSE WINDOW OUTSTREAM CHAT.STATE 'CLOSE "closed somehow"))
              (COND
                 ((NOT (OPENWP WINDOW))
                  (DEL.PROCESS (WINDOWPROP WINDOW 'PROCESS])
```

(CHAT.TYPEOUT.CLOSE

```
  [LAMBDA (WINDOW OUTSTREAM CHAT.STATE NEWSTATE MSG)               ; Edited  9-Nov-89 14:55 by bvm
    (COND
       ((OPENWP WINDOW)
        (printout OUTSTREAM T "[Connection " MSG " by remote host]" T)))
    (replace (CHAT.STATE RUNNING?) of CHAT.STATE with NEWSTATE)
    (LET [(CHATPROC (WINDOWPROP WINDOW 'PROCESS]
         (if (AND CHATPROC (NOT (TTY.PROCESSP CHATPROC)))
             then
```

```
                ;; Ordinarily, typein process notices that we've closed and will gracefully clean up, but currently it's hung waiting for tty.  I could
                ;; give it the tty explicitly, but that might disrupt the user's typing to some other process right now, especially if (tty.process t)
                ;; chooses not to give it back to the same place.  So we'll just explicitly kill it (it does have a cleanup form to handle closing the
                ;; window, etc).
                (DEL.PROCESS CHATPROC])
```

(CHAT.DID.RESHAPE

```
  [LAMBDA (CHAT.STATE)
    (DECLARE (USEDFREE INSTREAM DSP))                                    (* ejs%: "12-May-85 15:23")
    ;; Invoked in the type-out process when window is reshaped
    (with CHAT.STATE CHAT.STATE (CHAT.SCREENPARAMS CHAT.STATE INSTREAM DSP)
          (TERM.RESET.DISPLAY.PARMS CHAT.STATE])
```

(CHAT.SCREENPARAMS

```
  [LAMBDA (CHAT.STATE INSTREAM WINDOW)                                   (* ejs%: "12-May-85 15:51")
```

;;; Sends screen width, height to partner and updates title. If INSTREAM is NIL then only update title.

```
    (PROG ((HEIGHT (IMIN [IQUOTIENT (WINDOWPROP WINDOW 'HEIGHT)
                                    (IABS (DSPLINEFEED NIL (WINDOWPROP WINDOW 'DSP]
                         127))
           (WIDTH (IMIN (LINELENGTH NIL WINDOW)
                        127))
           (TITLE (WINDOWPROP WINDOW 'TITLE))
           EMACSMODE TITLEMIDDLE)
          (COND
             (INSTREAM (CHAT.SENDSCREENPARAMS INSTREAM HEIGHT WIDTH)))
          (WINDOWPROP WINDOW 'TITLE (CONCAT (SUBSTRING TITLE 1 (SUB1 (OR (SETQ TITLEMIDDLE (STRPOS ", height"
                                                                                                    TITLE))
                                                                          0)))
                                            ", height = " HEIGHT ", width = " WIDTH
                                            (COND
                                               [[OR (SETQ EMACSMODE (fetch (CHAT.STATE CHATINEMACS) of CHAT.STATE)
                                                          )
                                                    (AND TITLEMIDDLE (NOT (FIXP (NTHCHAR TITLE -1]
                                                (CONCAT ", Emacs " (COND
                                                                      (EMACSMODE "ON")
                                                                      (T "OFF"]
                                               (T ""])

)

;; window stuff

(DEFINEQ

(GETCHATWINDOW
  [LAMBDA (HOST WINDOW DPYTYPE)                                          (* bvm%: " 5-Sep-85 12:04")
    ;; Return a window, possibly new, to run a chat connection to HOST.  Uses WINDOW if possible
    (PROG ((TITLE (CONCAT (L-CASE DPYTYPE T)
                          " Chat connection to " HOST))
           DSP STATE)
          [COND
             [[AND (OR (WINDOWP WINDOW)
                       (WINDOWP (SETQ WINDOW CHATWINDOW)))
                   (OR [NOT (SETQ STATE (WINDOWPROP WINDOW 'CHATSTATE]
                       (COND
                          ((NOT (fetch (CHAT.STATE RUNNING?) of STATE))
                                                                         ; Connection in CHATWINDOW is dead
                           (CHAT.CLOSE WINDOW NIL T)
                           T]                                            ; Old window not in use.  This shouldn't happen, but...
              (WINDOWPROP WINDOW 'TITLE TITLE)
              (SETQ DSP (WINDOWPROP WINDOW 'DSP]
             (T (SETQ DSP (WINDOWPROP (SETQ WINDOW (LET ((SIZE (LISTP CHAT.WINDOW.SIZE)))
                                                     (DECODE.WINDOW.ARG (AND (NULL CHATWINDOWLST)
                                                                             CHAT.WINDOW.REGION)
                                                            (CAR SIZE)
                                                            (CDR SIZE)
                                                            TITLE)))
                                      'DSP))
                (DSPSCROLL T DSP)
                (OR CHATWINDOW (SETQ CHATWINDOW WINDOW]
          (push CHATWINDOWLST WINDOW)
          (RETURN WINDOW])
```

(CHAT.BUTTONFN

```
  [LAMBDA (WINDOW)                                                       (* ejs%: "12-May-85 17:59")
    (COND
       [(LASTMOUSESTATE LEFT)
        (PROG (CHAT.STATE CHAT.PROC)
              (COND
```

```
                ((AND (SETQ CHAT.STATE (WINDOWPROP WINDOW 'CHATSTATE))
                       (fetch (CHAT.STATE CHATINEMACS) of CHAT.STATE)
                       (SETQ CHAT.PROC (fetch (CHAT.STATE TYPEOUTPROC) of CHAT.STATE)))
                 (PROCESS.APPLY CHAT.PROC (FUNCTION CHAT.EMACS.MOVE)
                        (LIST CHAT.STATE)))
                (T (CHAT.HOLD WINDOW]
       ((LASTMOUSESTATE MIDDLE)
        (CHAT.MENU WINDOW])
```

(CHAT.HOLD

```
  [LAMBDA (WINDOW)                                          (* ejs%: "12-May-85 16:33")
```

;;; Toggle HOLD while button is down

```
    (PROG [(STATE (WINDOWPROP WINDOW 'CHATSTATE]
          (TOTOPW WINDOW)
          (OR STATE (RETURN))
          [COND
             ((NOT (fetch (CHAT.STATE HELD) of STATE))
              (replace (CHAT.STATE HELD) of STATE with T)
              (UNINTERRUPTABLY
                (UNTILMOUSESTATE UP))]
          (replace (CHAT.STATE HELD) of STATE with NIL])
```

(CHAT.MENU

```
  [LAMBDA (WINDOW)                                          (* lmm "20-Oct-86 18:03")
    (DECLARE (GLOBALVARS CHATMENU CHAT.REOPENMENU)
             (SPECVARS WINDOW STATE))                       ; Called by MIDDLE
    (PROG ((STATE (WINDOWPROP WINDOW 'CHATSTATE))
           COMMAND)
          [COND
             ((NOT STATE)                                   ; No Connection here;  try to reestablish
              (RETURN (COND
                         ((LASTMOUSESTATE MIDDLE)
                          (CHAT.RECONNECT WINDOW))
                         (T (TOTOPW WINDOW]
          (replace (CHAT.STATE HELD) of STATE with T)
          (\CHECKCARET WINDOW)
          (SELECTQ [SETQ COMMAND
                    (MENU (create MENU
                                  ITEMS _
                                  (APPEND CHATMENUITEMS
                                          [AND (CDR CHAT.DRIVERTYPES)
                                               (for X in CHAT.DRIVERTYPES
                                                  collect `(,(CONCAT (CAR X)
                                                                     " Mode")
                                                            '(LAMBDA (STATE WINDOW)
                                                               (CHAT.SET.EMULATOR STATE WINDOW
                                                                      ',(CAR X]
                                          (STREAMPROP (fetch (CHAT.STATE INSTREAM) of STATE)
                                                 'OPTIONS)
                                          [if (fetch (CHAT.STATE LOCALECHO) of STATE)
                                              then '(("Local Echo OFF" 'ECHO "Turn off local echoing"))
                                            else '(("Local Echo ON" 'ECHO "Turn on local echoing"]
                                          '((Close 'Close "Closes the connection and returns")
                                            (Suspend 'Suspend "Closes the connection but leaves window up")
                                            (New 'New "Closes this connection and prompts for a new host")
                                            (Freeze 'Freeze "Holds typeout in this window until you bug it again")
                                            (Clear (FUNCTION CHAT.CLEAR.FROM.MENU)
                                                   "Clears window, sets roll mode")
                                            ("Dribble" (FUNCTION CHAT.TYPESCRIPT)
                                                   "Starts a typescript of window typeout")
                                            ("Input" (FUNCTION CHAT.TAKE.INPUT)
                                                   "Allows input from a file")
                                            ("Emacs" (FUNCTION CHAT.SWITCH.EMACS)
                                                   "Toggle EMACS positioning"]
              (ECHO (replace (CHAT.STATE LOCALECHO) of STATE with (NOT (fetch (CHAT.STATE LOCALECHO) of STATE))))
              (Close (replace (CHAT.STATE RUNNING?) of STATE with 'CLOSE)
                                                            ; Ask CHAT.TYPEIN to shut things down.

                     )
              (New (replace (CHAT.STATE RUNNING?) of STATE with 'CLOSE)
                   (WINDOWPROP WINDOW 'KEEPCHAT 'NEW))
              (Suspend (replace (CHAT.STATE RUNNING?) of STATE with 'CLOSE)
                       (WINDOWPROP WINDOW 'KEEPCHAT T))
              (Freeze                                       ; Leave in HELD state
                      (RETURN))
              (NIL)
              (APPLY* COMMAND STATE WINDOW))
          (replace (CHAT.STATE HELD) of STATE with NIL])
```

(CHAT.CLEAR.FROM.MENU

```
  [LAMBDA (STATE WINDOW)                                    ; Edited 15-Feb-90 18:43 by bvm
    (CLEARW WINDOW)
```

```
    (TERM.RESET.DISPLAY.PARMS STATE)
    (TERM.HOME STATE])
```

(CHAT.TAKE.INPUT

```
  [LAMBDA (STATE WINDOW)                                    (* bvm%: " 1-Jun-84 17:43")
    (PROCESS.APPLY (WINDOWPROP WINDOW 'PROCESS)
           (FUNCTION CHAT.TAKE.INPUT1)
           (LIST WINDOW])
```

(CHAT.TAKE.INPUT1

```
  [LAMBDA (WINDOW)                                          ; Edited  4-Dec-86 22:53 by lmm
    (DECLARE (USEDFREE STREAM))                             ; In CHAT.TYPEIN
    (PROG ((PWINDOW (GETPROMPTWINDOW WINDOW))
           FILE)
          (CLEARW PWINDOW)
          (COND
             ((AND STREAM (NEQ STREAM T))
              (printout PWINDOW "Can't, still reading " (FULLNAME STREAM)))
             (T (SETQ FILE (PROMPTFORWORD "Take input from file (cr to return): " NIL NIL PWINDOW))
                (LET ((*LAST-CONDITION* NIL))
                     (COND
                        ((NULL FILE)
                         (CLEARW))
                        [[NLSETQ (SETQ FILE (OPENSTREAM FILE 'INPUT]
                         (CLEARW PWINDOW)
                         (printout PWINDOW "Reading " (FULLNAME (SETQ STREAM FILE]
                        (T (CLEARW PWINDOW)
                           (PRIN1 *LAST-CONDITION* PWINDOW])
```

(DO.CHAT.OPTION

```
  [LAMBDA (CHAT.STATE WINDOW)                               (* ejs%: "12-May-85 15:52")
```

;;; Pop up a menu of protocol specific options.

```
    (PROG [(MENU (CHAT.OPTIONMENU (fetch (CHAT.STATE INSTREAM) of CHAT.STATE]
          (COND
             (MENU (MENU MENU))
             (T (printout PROMPTWINDOW "This protocol has no options."])
```

(CHAT.RECONNECT

```
  [LAMBDA (WINDOW)                                          (* bvm%: " 4-Sep-85 19:52")
    (LET* ((MAINW (OR (WINDOWPROP WINDOW 'ICONFOR)
                      WINDOW))
           (STATE (WINDOWPROP MAINW 'CHATHOST))
           FN)
          (COND
             ((NULL STATE)
              (APPLY* (CHAT.RECONNECT.OFF WINDOW)
                      WINDOW))
             ((NOT (LASTMOUSESTATE MIDDLE))
              (APPLY* (OR (WINDOWPROP WINDOW 'OLDBUTTONEVENTFN)
                          (FUNCTION TOTOPW))
                      WINDOW))
             ([MENU (OR CHAT.REOPENMENU (SETQ CHAT.REOPENMENU (create MENU
                                                                      ITEMS _ '((ReConnect T "Will reestablish
                                                                                 this Chat connection"]
              (CHAT.RECONNECT.OFF WINDOW)                   ; Don't let this command get issued twice
              (TTY.PROCESS (ADD.PROCESS (LIST 'CHAT (KWOTE (CAR STATE))
                                              (KWOTE (CDR STATE))
                                              NIL MAINW T])
```

(CHAT.RECONNECT.OFF

```
  [LAMBDA (WINDOW)                                          (* bvm%: " 4-Sep-85 19:51")
```

;;; Removes CHAT.RECONNECT as the buttonfn for WINDOW and returns new buttonfn

```
    (LET [(FN (OR (WINDOWPROP WINDOW 'OLDBUTTONEVENTFN NIL)
                  (FUNCTION TOTOPW]
         (WINDOWPROP WINDOW 'BUTTONEVENTFN FN)
         FN])
```

(CHAT.RESHAPEWINDOW

```
  [LAMBDA (WINDOW OLDIMAGE IMAGEREGION OLDSCREENREGION)     (* ejs%: "14-Jun-85 15:08")
```

;; RESHAPEFN for the chat window

```
    (RESHAPEBYREPAINTFN WINDOW OLDIMAGE IMAGEREGION)
```

;; Note: Don't pass OLDSCREENREGION to RESHAPEBYREPAINTFN or it may try to leave the image fixed and move the coordinate system.
;; Our code assumes that the bottom of the window is zero. If someone gets ambitious, can figure out how to change the rest of Chat code so it
;; does not make that assumption

```
    (LET* [(CHAT.STATE (WINDOWPROP WINDOW 'CHATSTATE))
```

```
           (CHAT.PROC (AND CHAT.STATE (fetch (CHAT.STATE TYPEOUTPROC) of CHAT.STATE]
          (COND
             ((AND (PROCESSP CHAT.PROC)
                    (NOT (RELPROCESSP CHAT.PROC)))
              (PROCESS.APPLY CHAT.PROC (FUNCTION CHAT.DID.RESHAPE)
                     (LIST CHAT.STATE])
```

(CHAT.TTYENTRYFN

```
  [LAMBDA (PROCESS)                                          (* lmm "14-Oct-86 11:28")
    (PROG ((WINDOW (PROCESSPROP PROCESS 'WINDOW))
           STATE)
          (COND
             ([AND WINDOW (SETQ STATE (WINDOWPROP WINDOW 'CHATSTATE]
              (replace (CHAT.STATE HELD) of STATE with NIL])
```

(CHAT.TTYEXITFN

```
  [LAMBDA (PROCESS NEWPROCESS)                               (* lmm "14-Oct-86 11:26")
    NIL])
```

(CHAT.TYPESCRIPT

```
  [LAMBDA (STATE)                                            (* ejs%: "12-May-85 16:08")
    (PROG ((PROC (fetch (CHAT.STATE TYPEOUTPROC) of STATE)))
          (COND
             (PROC (PROCESS.APPLY PROC (FUNCTION CHAT.TYPESCRIPT1)
                          (LIST STATE])
```

(CHAT.TYPESCRIPT1

```
  [LAMBDA (CHAT.STATE)                                       ; Edited 15-Feb-90 14:51 by bvm
    ;; Called in context of type-out proc to change the dribble file
    (with CHAT.STATE CHAT.STATE (LET ((PWINDOW (GETPROMPTWINDOW WINDOW))
                                      FILE OLDFILE)
                                     (if (NOT (STRING-EQUAL (SETQ FILE (CHAT.PROMPT.FOR.INPUT "Typescript to file
                                                                              (cr to close): " PWINDOW))
                                                            T))
                                         then (COND
                                                 [[OR (NULL FILE)
                                                      (NLSETQ (SETQ FILE (OPENSTREAM FILE 'OUTPUT 'NEW]
                                                  (COND
                                                     (TYPESCRIPTSTREAM (printout PWINDOW (CLOSEF TYPESCRIPTSTREAM
                                                                                                 )
                                                                                       " closed.  ")))
                                                  (replace TYPESCRIPTSTREAM of CHAT.STATE with (SETQ
                                                                                                 TYPESCRIPTSTREAM
                                                                                                 FILE))
                                                  (AND FILE (printout PWINDOW "Opened " (FULLNAME FILE]
                                                 (T (printout PWINDOW "Could not open " FILE])

)

;; for dialouts

(DEFINEQ

(CHAT.CHOOSE.PHONE.NUMBER
  [LAMBDA NIL                                                ; Edited 15-Feb-90 14:58 by bvm
    ;; Prompt user for phone number
    (DECLARE (GLOBALVARS CHAT.PHONE.NUMBER.MENU CHAT.PHONE.NUMBERS))
    (LET ([NUMBER (COND
                     [(CDR CHAT.PHONE.NUMBERS)
                      (MENU (OR CHAT.PHONE.NUMBER.MENU (SETQ CHAT.PHONE.NUMBER.MENU
                                                             (create MENU
                                                                     ITEMS _ CHAT.PHONE.NUMBERS
                                                                     TITLE _ "Phone Number  "]
                     (T 'Other]
          NEWNUMBER)
          (COND
             ((NEQ NUMBER 'Other)
              NUMBER)
             ((SETQ NUMBER (CHAT.PROMPT.FOR.INPUT "Please enter a phone number in the form (800)555-1212: " 16))
              [push CHAT.PHONE.NUMBERS (LIST NUMBER (SETQ NEWNUMBER
                                                          (CONCATCODES (for CHAR in (CHCON NUMBER) collect CHAR
                                                                          when (AND (IGEQ CHAR (CHARCODE 0))
                                                                                    (ILEQ CHAR (CHARCODE 9]
              (SETQ CHAT.PHONE.NUMBER.MENU NIL)
              NEWNUMBER])

)

(RPAQ? CHAT.PHONE.NUMBER.MENU )

(RPAQ? CHAT.PHONE.NUMBERS '(Other))
```

;; for EMACS

```
(DEFINEQ

(CHAT.EMACS.MOVE
  [LAMBDA (CHAT.STATE)                                    (* ejs%: "12-May-85 15:44")

;;; This function is invoked in the context of the typeout process, so that we can easily see where we are on the display, and so that we don't hang up the
;;; mouse if connection gets in trouble

    (with CHAT.STATE CHAT.STATE (PROG ((CLOC (CURSORPOSITION NIL WINDOW))
                                       DROW CCOLUMN)
                                ;; The characters are FONTHEIGHT high by FONTWIDTH wide
                                      [COND
                                         ((IGEQ XPOS FONTWIDTH)   ; Go back to column 0
                                          (BOUT OUTSTREAM (fetch EMCOL0 of CHAT.EMACSCOMMANDS]
                                      (SETQ DROW (IDIFFERENCE (IQUOTIENT YPOS FONTHEIGHT)
                                                         (IQUOTIENT (fetch YCOORD of CLOC)
                                                                FONTHEIGHT)))
                                ;; Positive DROW means go DOWN
                                      [COND
                                         ((ILESSP DROW 0)         ; Go up DROW rows
                                          (COND
                                             ((NEQ DROW -1)
                                              (BOUT OUTSTREAM (fetch EMARG of CHAT.EMACSCOMMANDS))
                                              (PRIN3 (MKSTRING (IMINUS DROW))
                                                     OUTSTREAM)))
                                          (BOUT OUTSTREAM (fetch EMUP of CHAT.EMACSCOMMANDS)))
                                         ((IGREATERP DROW 0)      ; Go down DROW rows
                                          (COND
                                             ((NEQ DROW 1)
                                              (BOUT OUTSTREAM (fetch EMARG of CHAT.EMACSCOMMANDS))
                                              (PRIN3 (MKSTRING DROW)
                                                     OUTSTREAM)))
                                          (BOUT OUTSTREAM (fetch EMDOWN of CHAT.EMACSCOMMANDS]
                                      (SETQ CCOLUMN (IQUOTIENT (fetch XCOORD of CLOC)
                                                               FONTWIDTH))
                                      [COND
                                         ((IGREATERP CCOLUMN 0)   ; Now go to the correct column
                                          (COND
                                             ((NEQ CCOLUMN 1)
                                              (BOUT OUTSTREAM (fetch EMARG of CHAT.EMACSCOMMANDS))
                                              (PRIN3 (MKSTRING CCOLUMN)
                                                     OUTSTREAM)))
                                          (BOUT OUTSTREAM (fetch EMFORWARD of CHAT.EMACSCOMMANDS]
                                      (FORCEOUTPUT OUTSTREAM])

(CHAT.SWITCH.EMACS
  [LAMBDA (CHATSTATE WINDOW)                              (* ejs%: "12-May-85 17:05")

;;; Toggles the value of CHAT.IN.EMACS?

    (replace (CHAT.STATE CHATINEMACS) of CHATSTATE with (NOT (fetch (CHAT.STATE CHATINEMACS) of CHATSTATE)))
                                                          ; Now update title to show Emacs state
    (CHAT.SCREENPARAMS CHATSTATE NIL WINDOW])

)


(DEFGLOBALVAR CHAT.EMACSCOMMANDS '(21 16 14 6 1)
                                 "List of 5 character codes that perform Emacs functions Arg, Up 1 Line,
                                 Down 1 Line, Forward Character, Beginning of Line")

(DEFINEQ

(CHAT.ICONFN
  [LAMBDA (WINDOW OLDICON)                                (* bvm%: " 4-Sep-85 19:23")
    (DECLARE (GLOBALVARS TTYKBDICONSPEC TTYKBD TTYKBDMASK TTYKBDICONSPECREGION))
    (COND
       ((TTY.PROCESSP (WINDOWPROP WINDOW 'PROCESS))
        (TTY.PROCESS T)))
    (COND
       ((FNTYP 'TITLEDICONW)
        (OR OLDICON (TITLEDICONW (OR TTYKBDICONSPEC (SETQ TTYKBDICONSPEC
                                                          (create TITLEDICON
                                                                  ICON _ TTYKBD
                                                                  MASK _ TTYKBDMASK
                                                                  TITLEREG _ TTYKBDICONSPECREGION)))
                                 (CAR (WINDOWPROP WINDOW 'CHATHOST))
                                 (FONTCREATE 'HELVETICA 8])

)
```

(RPAQQ **TTYKBD**

)

(RPAQQ **TTYKBDMASK**

)

(RPAQQ **TTYKBDICONSPECREGION** (4 3 56 14))

(RPAQ? **TTYKBDICONSPEC**)

(ADDTOVAR **CHATMENUITEMS**)

(RPAQ? **CHATMENU**)

(RPAQ? **CHAT.REOPENMENU**)

(RPAQ? **CHAT.HOSTMENU**)

(RPAQ? **CHATWINDOWLST**)

(RPAQ? **CHAT.DRIVERTYPES**)

(RPAQ? **CHATDEBUGFLG**)

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(FILESLOAD (SOURCE)
       CHATDECLS)

(DECLARE%: EVAL@COMPILE

(RECORD EMACSCOMMANDS (EMARG EMUP EMDOWN EMFORWARD EMCOL0))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS CHATMENUITEMS)
)
)

(RPAQ? INVERTWINDOWFN 'INVERTW)

(DEFINEQ
```

(**SPAWN.CHAT**

```
  [LAMBDA (LOGOPTION)                                   ; Edited 25-May-88 16:20 by bvm
    ;; From the Background Menu, runs CHAT as a process
    (ADD.PROCESS `(CHAT NIL ',LOGOPTION NIL NIL T])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(ADDTOVAR BackgroundMenuCommands ("Chat" '(\SPAWN.CHAT)
                                         "Runs a new CHAT process; prompts for host"
                                         (SUBITEMS ("No Login" '(\SPAWN.CHAT 'NONE)
                                                          "Runs CHAT without doing automatic login"))))

(SETQ BackgroundMenu)

(FILESLOAD DMCHAT)

(/DECLAREDATATYPE 'CHAT.STATE
       '(FLAG FLAG FLAG FLAG FLAG FLAG (BITS 1)
              POINTER POINTER POINTER POINTER POINTER POINTER WORD WORD WORD WORD WORD WORD WORD WORD WORD
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '46)
```

)

(PUTPROPS **CHAT COPYRIGHT** ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1989 1990 1992 1993))

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX

RECORD INDEX