File created: 11-Jun-90 15:00:11 **{DSK}<usr>local>lde>lispcore>library>DES.;2**

changes to: (VARS DESCOMS)

previous date: 24-Jul-87 18:29:27 {DSK}<usr>local>lde>lispcore>library>DES.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ DESCOMS
       ((COMS                                                    ; Entry points
              (FNS DES.BREAKOUT.BLOCKS DES.MAKE.BLOCKS DES.ECB.ENCRYPT DES.ECB.DECRYPT DES.CBC.ENCRYPT
                   DES.CBC.DECRYPT DES.CBCC.ENCRYPT DES.CBCC.DECRYPT DES.PASSWORD.TO.KEY DES.MAKE.KEY))
        (COMS                                                    ; Implementation
              (FNS DES.CORRECT.KEY.PARITY DES.CRYPT.BLOCK DES.KEY.COPY DES.KEY.EQUAL DES.LOOPBODY
                   DES.MAKE.INTERNAL.KEYS DES.PERM.E DES.PERM.INITIAL DES.PERM.INV.INITIAL DES.PERM.P
                   DES.PERM.PC1 DES.PERM.PC2 DES.REC32.LS28 DES.SMAP REC32.XOR REC48.XOR REC64.XOR REC64.XOR.CHK
                   ))
        (VARS DES.PARITY.TABLE DES.SBOX.1 DES.SBOX.2 DES.SBOX.3 DES.SBOX.4 DES.SBOX.5 DES.SBOX.6 DES.SBOX.7
              DES.SBOX.8 DES.SHIFTS (DESKEYSLST))
        (DECLARE%: EVAL@COMPILE DONTCOPY
               (RECORDS DES.REC.E1 DES.REC.E2 DES.REC32.4 DES.REC32.LS28.IN DES.REC32.LS28.OUT DES.REC48.6
                      DESBLOCK DESKEY DESKEY.P REC32 REC32.W REC48 REC48.W REC64 REC64.W))))
```

;; Entry points

```
(DEFINEQ
```

(DES.BREAKOUT.BLOCKS

```
  [LAMBDA (L)                                                    ; Edited 22-May-87 15:30 by bvm:
    ;; Converts a list of DES 64-bit blocks into a "sequence unspecified" for courier.
    (for E in L join (LIST (fetch (DESBLOCK W1) of E)
                           (fetch (DESBLOCK W2) of E)
                           (fetch (DESBLOCK W3) of E)
                           (fetch (DESBLOCK W4) of E])
```

(DES.MAKE.BLOCKS

```
  [LAMBDA (L)                                                    ; Edited 24-Jul-87 18:28 by bvm:
    ;; Convert L, a courier "sequence unspecified" into a list of DES 64-bit blocks, padded on the right if needed by zeros.
    [if NIL
        then (for K on L by (CDDDDR K) collect (create DESBLOCK
                                                       W1 _ (CAR K)
                                                       W2 _ (COND
                                                              ((CADR K))
                                                              (0))
                                                       W3 _ (COND
                                                              ((CADDR K))
                                                              (0))
                                                       W4 _ (COND
                                                              ((CADDDR K))
                                                              (0]
    (while L collect (LET ((B (create DESBLOCK)))                ; Blocks are initialized by allocator to zero.
                       (\PUTBASE B 0 (CAR L))
                       [if (SETQ L (CDR L))
                           then (\PUTBASE B 1 (CAR L))
                                (if (SETQ L (CDR L))
                                    then (\PUTBASE B 2 (CAR L))
                                         (if (SETQ L (CDR L))
                                             then (\PUTBASE B 3 (CAR L))
                                                  (SETQ L (CDR L]
                       B])
```

(DES.ECB.ENCRYPT

```
  [LAMBDA (KEY DAT)                                              (* jwo%: "25-Jun-85 12:24")
    (LET ((IKLST (DES.MAKE.INTERNAL.KEYS KEY)))
         (DES.CRYPT.BLOCK IKLST DAT 'ENCRYPT])
```

(DES.ECB.DECRYPT

```
  [LAMBDA (KEY DAT)                                              (* jwo%: "25-Jun-85 12:24")
    (LET ((IKLST (DES.MAKE.INTERNAL.KEYS KEY)))
         (DES.CRYPT.BLOCK IKLST DAT 'DECRYPT])
```

(DES.CBC.ENCRYPT

```
  [LAMBDA (L)                                                    (* jwo%: " 3-Jul-85 17:11")
```

```
    (ERROR "Not Implemented!"])
```

(DES.CBC.DECRYPT

```
  [LAMBDA (KEY L N)                                          (* jwo%: " 8-Aug-85 23:36")
    (LET ((IKLST (DES.MAKE.INTERNAL.KEYS KEY)))
         (CONS (DES.CRYPT.BLOCK IKLST (CAR L)
                     'DECRYPT)
               (COND
                  [(IGREATERP N 1)
                   (DREVERSE (for I from N to 2 by -1 collect (REC64.XOR (CAR (NTH L (SUB1 I)))
                                                                         (DES.CRYPT.BLOCK IKLST
                                                                                (CAR (NTH L I))
                                                                                'DECRYPT]
                  (T NIL])
```

(DES.CBCC.ENCRYPT

```
  [LAMBDA (L)                                                (* jwo%: " 9-Aug-85 01:04")
```

;;; Note: I'm not bothering with this one right now because I don't think anybody needs it to do Strong Authentication.

```
    (ERROR "Not Implemented!"])
```

(DES.CBCC.DECRYPT

```
  [LAMBDA (KEY L N)                                          (* jwo%: " 8-Aug-85 22:49")
    [COND
       ((NULL N)
        (SETQ N (LENGTH L]
    (LET [(PL (DES.CBC.DECRYPT KEY L (SUB1 N]
         (NCONC1 PL (REC64.XOR (CAR (NTH L (SUB1 N)))
                               (REC64.XOR (REC64.XOR.CHK PL (SUB1 N))
                                          (DES.CRYPT.BLOCK (DES.MAKE.INTERNAL.KEYS KEY)
                                                 (CAR (NTH L N))
                                                 'DECRYPT])
```

(DES.PASSWORD.TO.KEY

```
  [LAMBDA (PASSWORD)                                         (* jwo%: " 8-Aug-85 23:54")
    ;; Algorithm documented on page 27 of XSIS Authentication Protocol specification.
    (bind (NEWKEY _ (DES.MAKE.KEY))
          (STR _ (CONCAT PASSWORD))
          (BLOCK _ (create DESBLOCK)) until (STREQUAL STR "")
       do (PROGN [replace (DESBLOCK W1) of BLOCK with (LET ((CHAR (GNC STR)))
                                                           (COND
                                                              [CHAR (L-CASECODE (\DECRYPT.PWD.CHAR (CHCON1 CHAR]
                                                              (T 0]
                 [replace (DESBLOCK W2) of BLOCK with (LET ((CHAR (GNC STR)))
                                                           (COND
                                                              [CHAR (L-CASECODE (\DECRYPT.PWD.CHAR (CHCON1 CHAR]
                                                              (T 0]
                 [replace (DESBLOCK W3) of BLOCK with (LET ((CHAR (GNC STR)))
                                                           (COND
                                                              [CHAR (L-CASECODE (\DECRYPT.PWD.CHAR (CHCON1 CHAR]
                                                              (T 0]
                 [replace (DESBLOCK W4) of BLOCK with (LET ((CHAR (GNC STR)))
                                                           (COND
                                                              [CHAR (L-CASECODE (\DECRYPT.PWD.CHAR (CHCON1 CHAR]
                                                              (T 0]
                 (SETQ NEWKEY (DES.ECB.ENCRYPT NEWKEY BLOCK)))
          finally (RETURN (DES.CORRECT.KEY.PARITY NEWKEY])
```

(DES.MAKE.KEY

```
  [LAMBDA (L)                                                (* jwo%: " 8-Aug-85 22:18")
    (create DESBLOCK
            W1 _ (COND
                    ((CAR L))
                    (0))
            W2 _ (COND
                    ((CADR L))
                    (0))
            W3 _ (COND
                    ((CADDR L))
                    (0))
            W4 _ (COND
                    ((CADDDR L))
                    (0])

)

;; Implementation

(DEFINEQ
```

(DES.CORRECT.KEY.PARITY

```
[LAMBDA (KEY)                                                    (* jwo%: " 3-Jul-85 16:52")
  (replace (DESKEY.P P1) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B1) of KEY)))
  (replace (DESKEY.P P2) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B2) of KEY)))
  (replace (DESKEY.P P3) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B3) of KEY)))
  (replace (DESKEY.P P4) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B4) of KEY)))
  (replace (DESKEY.P P5) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B5) of KEY)))
  (replace (DESKEY.P P6) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B6) of KEY)))
  (replace (DESKEY.P P7) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B7) of KEY)))
  (replace (DESKEY.P P8) of KEY with (ELT DES.PARITY.TABLE (fetch (DESKEY.P B8) of KEY)))
  KEY])
```

(DES.CRYPT.BLOCK

```
[LAMBDA (KLST DAT DIRECTION)                                     (* jwo%: "29-Jun-85 21:28")
  (LET ((LR (DES.PERM.INITIAL DAT)))
       [if (EQ DIRECTION 'ENCRYPT)
           then [for I from 1 to 16 do (SETQ LR (DES.LOOPBODY (CAR LR)
                                                              (CDR LR)
                                                              (CAR (NTH KLST I]
         else (for I from 16 to 1 do (SETQ LR (DES.LOOPBODY (CAR LR)
                                                            (CDR LR)
                                                            (CAR (NTH KLST I]
       (DES.PERM.INV.INITIAL (CDR LR)
              (CAR LR])
```

(DES.KEY.COPY

```
[LAMBDA (K)                                                      (* jwo%: " 6-Jul-85 15:26")
  (create DESKEY
          W1 _ (fetch (DESKEY W1) of K)
          W2 _ (fetch (DESKEY W2) of K)
          W3 _ (fetch (DESKEY W3) of K)
          W4 _ (fetch (DESKEY W4) of K])
```

(DES.KEY.EQUAL

```
[LAMBDA (X Y)                                                    (* jwo%: " 6-Jul-85 15:16")
  (AND (EQ (fetch (DESKEY W1) of X)
           (fetch (DESKEY W1) of Y))
       (EQ (fetch (DESKEY W2) of X)
           (fetch (DESKEY W2) of Y))
       (EQ (fetch (DESKEY W3) of X)
           (fetch (DESKEY W3) of Y))
       (EQ (fetch (DESKEY W4) of X)
           (fetch (DESKEY W4) of Y])
```

(DES.LOOPBODY

```
[LAMBDA (L R K)                                                  (* jwo%: "22-Jun-85 21:34")
  (CONS R (REC32.XOR L (DES.PERM.P (DES.SMAP (REC48.XOR (DES.PERM.E R)
                                                        K])
```

(DES.MAKE.INTERNAL.KEYS

```
[LAMBDA (K)                                                      ; Edited 22-May-87 15:49 by bvm:

  ;; Returns the "key schedule" for key K, a list of 16 48-bit numbers.

  ;; The last FOR loop is the actual internal key construction algorithm.  The goop wrapped around it variously checks if this key is already first on the
  ;; DESKEYSLST cache, is later on DESKEYSLST (in which case it does move-to-front), or attaches the newly constructed keys to the front of
  ;; DESKEYSLST.  We should restrict the length of DESKEYSLST so that the cache doesn't grow indefinitely...

  [if (NOT (AND DESKEYSLST (DES.KEY.EQUAL (CAAR DESKEYSLST)
                                          K)))
      then (for TL on DESKEYSLST while (CDR TL) when (DES.KEY.EQUAL (CAADR TL)
                                                                    K)
              do (RETURN (LET ((TEMP (CDR TL)))                  ; Promote this entry to the front
                              (RPLACD TL (CDR TEMP))
                              (RPLACD TEMP DESKEYSLST)
                              (SETQ DESKEYSLST TEMP)))
              finally                                            ; Compute afresh
                     (push DESKEYSLST (CONS (DES.KEY.COPY K)
                                            (LET* ((CD (DES.PERM.PC1 K))
                                                   (C (CAR CD))
                                                   (D (CDR CD)))

                                              ;; C & D are 28-bit quantities, a permutation of 56 of the bits of K.  Cycle the pieces.  Total
                                              ;; shift over this whole loop is 28 bits.

                                                   (for I from 1 to 16
                                                      collect (PROGN (for J from 1 to (ELT DES.SHIFTS I)
                                                                        do (SETQ C (DES.REC32.LS28 C))
                                                                           (SETQ D (DES.REC32.LS28 D)))
                                                                     (DES.PERM.PC2 C D]
  (CDAR DESKEYSLST])
```

(DES.PERM.E

```
[LAMBDA (X)                                                      (* jwo%: "21-Jun-85 17:40")
```

```
    (create DES.REC48.6
           S1 _ (IPLUS (ITIMES 32 (fetch (REC32 BIT32) of X))
                       (fetch (DES.REC.E1 B1.5) of X))
           S2 _ (fetch (DES.REC.E2 B4.9) of X)
           S3.1 _ (fetch (DES.REC.E1 B8.11) of X)
           S3.2 _ (fetch (DES.REC.E1 B12.13) of X)
           S4 _ (IPLUS (ITIMES 2 (fetch (DES.REC.E2 B12.16) of X))
                       (fetch (REC32 BIT17) of X))
           S5 _ (IPLUS (ITIMES 32 (fetch (REC32 BIT16) of X))
                       (fetch (DES.REC.E2 B17.21) of X))
           S6.1 _ (fetch (DES.REC.E1 B20.21) of X)
           S6.2 _ (fetch (DES.REC.E1 B22.25) of X)
           S7 _ (fetch (DES.REC.E2 B24.29) of X)
           S8 _ (IPLUS (ITIMES 2 (fetch (DES.REC.E1 B28.32) of X))
                       (fetch (REC32 BIT1) of X])
```

(DES.PERM.INITIAL

```
  [LAMBDA (X)                                                    (* jwo%: "21-Jun-85 16:40")
    (CONS (create REC32
                  BIT1 _ (fetch (REC64 BIT58) of X)
                  BIT2 _ (fetch (REC64 BIT50) of X)
                  BIT3 _ (fetch (REC64 BIT42) of X)
                  BIT4 _ (fetch (REC64 BIT34) of X)
                  BIT5 _ (fetch (REC64 BIT26) of X)
                  BIT6 _ (fetch (REC64 BIT18) of X)
                  BIT7 _ (fetch (REC64 BIT10) of X)
                  BIT8 _ (fetch (REC64 BIT2) of X)
                  BIT9 _ (fetch (REC64 BIT60) of X)
                  BIT10 _ (fetch (REC64 BIT52) of X)
                  BIT11 _ (fetch (REC64 BIT44) of X)
                  BIT12 _ (fetch (REC64 BIT36) of X)
                  BIT13 _ (fetch (REC64 BIT28) of X)
                  BIT14 _ (fetch (REC64 BIT20) of X)
                  BIT15 _ (fetch (REC64 BIT12) of X)
                  BIT16 _ (fetch (REC64 BIT4) of X)
                  BIT17 _ (fetch (REC64 BIT62) of X)
                  BIT18 _ (fetch (REC64 BIT54) of X)
                  BIT19 _ (fetch (REC64 BIT46) of X)
                  BIT20 _ (fetch (REC64 BIT38) of X)
                  BIT21 _ (fetch (REC64 BIT30) of X)
                  BIT22 _ (fetch (REC64 BIT22) of X)
                  BIT23 _ (fetch (REC64 BIT14) of X)
                  BIT24 _ (fetch (REC64 BIT6) of X)
                  BIT25 _ (fetch (REC64 BIT64) of X)
                  BIT26 _ (fetch (REC64 BIT56) of X)
                  BIT27 _ (fetch (REC64 BIT48) of X)
                  BIT28 _ (fetch (REC64 BIT40) of X)
                  BIT29 _ (fetch (REC64 BIT32) of X)
                  BIT30 _ (fetch (REC64 BIT24) of X)
                  BIT31 _ (fetch (REC64 BIT16) of X)
                  BIT32 _ (fetch (REC64 BIT8) of X))
          (create REC32
                  BIT1 _ (fetch (REC64 BIT57) of X)
                  BIT2 _ (fetch (REC64 BIT49) of X)
                  BIT3 _ (fetch (REC64 BIT41) of X)
                  BIT4 _ (fetch (REC64 BIT33) of X)
                  BIT5 _ (fetch (REC64 BIT25) of X)
                  BIT6 _ (fetch (REC64 BIT17) of X)
                  BIT7 _ (fetch (REC64 BIT9) of X)
                  BIT8 _ (fetch (REC64 BIT1) of X)
                  BIT9 _ (fetch (REC64 BIT59) of X)
                  BIT10 _ (fetch (REC64 BIT51) of X)
                  BIT11 _ (fetch (REC64 BIT43) of X)
                  BIT12 _ (fetch (REC64 BIT35) of X)
                  BIT13 _ (fetch (REC64 BIT27) of X)
                  BIT14 _ (fetch (REC64 BIT19) of X)
                  BIT15 _ (fetch (REC64 BIT11) of X)
                  BIT16 _ (fetch (REC64 BIT3) of X)
                  BIT17 _ (fetch (REC64 BIT61) of X)
                  BIT18 _ (fetch (REC64 BIT53) of X)
                  BIT19 _ (fetch (REC64 BIT45) of X)
                  BIT20 _ (fetch (REC64 BIT37) of X)
                  BIT21 _ (fetch (REC64 BIT29) of X)
                  BIT22 _ (fetch (REC64 BIT21) of X)
                  BIT23 _ (fetch (REC64 BIT13) of X)
                  BIT24 _ (fetch (REC64 BIT5) of X)
                  BIT25 _ (fetch (REC64 BIT63) of X)
                  BIT26 _ (fetch (REC64 BIT55) of X)
                  BIT27 _ (fetch (REC64 BIT47) of X)
                  BIT28 _ (fetch (REC64 BIT39) of X)
                  BIT29 _ (fetch (REC64 BIT31) of X)
                  BIT30 _ (fetch (REC64 BIT23) of X)
                  BIT31 _ (fetch (REC64 BIT15) of X)
                  BIT32 _ (fetch (REC64 BIT7) of X])
```

(DES.PERM.INV.INITIAL

```
[LAMBDA (L R)                                          (* jwo%: "24-Jun-85 23:27")
  (create REC64
          BIT58 _ (fetch (REC32 BIT1) of L)
          BIT50 _ (fetch (REC32 BIT2) of L)
          BIT42 _ (fetch (REC32 BIT3) of L)
          BIT34 _ (fetch (REC32 BIT4) of L)
          BIT26 _ (fetch (REC32 BIT5) of L)
          BIT18 _ (fetch (REC32 BIT6) of L)
          BIT10 _ (fetch (REC32 BIT7) of L)
          BIT2 _ (fetch (REC32 BIT8) of L)
          BIT60 _ (fetch (REC32 BIT9) of L)
          BIT52 _ (fetch (REC32 BIT10) of L)
          BIT44 _ (fetch (REC32 BIT11) of L)
          BIT36 _ (fetch (REC32 BIT12) of L)
          BIT28 _ (fetch (REC32 BIT13) of L)
          BIT20 _ (fetch (REC32 BIT14) of L)
          BIT12 _ (fetch (REC32 BIT15) of L)
          BIT4 _ (fetch (REC32 BIT16) of L)
          BIT62 _ (fetch (REC32 BIT17) of L)
          BIT54 _ (fetch (REC32 BIT18) of L)
          BIT46 _ (fetch (REC32 BIT19) of L)
          BIT38 _ (fetch (REC32 BIT20) of L)
          BIT30 _ (fetch (REC32 BIT21) of L)
          BIT22 _ (fetch (REC32 BIT22) of L)
          BIT14 _ (fetch (REC32 BIT23) of L)
          BIT6 _ (fetch (REC32 BIT24) of L)
          BIT64 _ (fetch (REC32 BIT25) of L)
          BIT56 _ (fetch (REC32 BIT26) of L)
          BIT48 _ (fetch (REC32 BIT27) of L)
          BIT40 _ (fetch (REC32 BIT28) of L)
          BIT32 _ (fetch (REC32 BIT29) of L)
          BIT24 _ (fetch (REC32 BIT30) of L)
          BIT16 _ (fetch (REC32 BIT31) of L)
          BIT8 _ (fetch (REC32 BIT32) of L)
          BIT57 _ (fetch (REC32 BIT1) of R)
          BIT49 _ (fetch (REC32 BIT2) of R)
          BIT41 _ (fetch (REC32 BIT3) of R)
          BIT33 _ (fetch (REC32 BIT4) of R)
          BIT25 _ (fetch (REC32 BIT5) of R)
          BIT17 _ (fetch (REC32 BIT6) of R)
          BIT9 _ (fetch (REC32 BIT7) of R)
          BIT1 _ (fetch (REC32 BIT8) of R)
          BIT59 _ (fetch (REC32 BIT9) of R)
          BIT51 _ (fetch (REC32 BIT10) of R)
          BIT43 _ (fetch (REC32 BIT11) of R)
          BIT35 _ (fetch (REC32 BIT12) of R)
          BIT27 _ (fetch (REC32 BIT13) of R)
          BIT19 _ (fetch (REC32 BIT14) of R)
          BIT11 _ (fetch (REC32 BIT15) of R)
          BIT3 _ (fetch (REC32 BIT16) of R)
          BIT61 _ (fetch (REC32 BIT17) of R)
          BIT53 _ (fetch (REC32 BIT18) of R)
          BIT45 _ (fetch (REC32 BIT19) of R)
          BIT37 _ (fetch (REC32 BIT20) of R)
          BIT29 _ (fetch (REC32 BIT21) of R)
          BIT21 _ (fetch (REC32 BIT22) of R)
          BIT13 _ (fetch (REC32 BIT23) of R)
          BIT5 _ (fetch (REC32 BIT24) of R)
          BIT63 _ (fetch (REC32 BIT25) of R)
          BIT55 _ (fetch (REC32 BIT26) of R)
          BIT47 _ (fetch (REC32 BIT27) of R)
          BIT39 _ (fetch (REC32 BIT28) of R)
          BIT31 _ (fetch (REC32 BIT29) of R)
          BIT23 _ (fetch (REC32 BIT30) of R)
          BIT15 _ (fetch (REC32 BIT31) of R)
          BIT7 _ (fetch (REC32 BIT32) of R])
```

(DES.PERM.P

```
[LAMBDA (X)                                            (* jwo%: "21-Jun-85 22:45")
  (create REC32
          BIT1 _ (fetch (REC32 BIT16) of X)
          BIT2 _ (fetch (REC32 BIT7) of X)
          BIT3 _ (fetch (REC32 BIT20) of X)
          BIT4 _ (fetch (REC32 BIT21) of X)
          BIT5 _ (fetch (REC32 BIT29) of X)
          BIT6 _ (fetch (REC32 BIT12) of X)
          BIT7 _ (fetch (REC32 BIT28) of X)
          BIT8 _ (fetch (REC32 BIT17) of X)
          BIT9 _ (fetch (REC32 BIT1) of X)
          BIT10 _ (fetch (REC32 BIT15) of X)
          BIT11 _ (fetch (REC32 BIT23) of X)
          BIT12 _ (fetch (REC32 BIT26) of X)
          BIT13 _ (fetch (REC32 BIT5) of X)
          BIT14 _ (fetch (REC32 BIT18) of X)
          BIT15 _ (fetch (REC32 BIT31) of X)
```

```
          BIT16 _ (fetch (REC32 BIT10) of X)
          BIT17 _ (fetch (REC32 BIT2) of X)
          BIT18 _ (fetch (REC32 BIT8) of X)
          BIT19 _ (fetch (REC32 BIT24) of X)
          BIT20 _ (fetch (REC32 BIT14) of X)
          BIT21 _ (fetch (REC32 BIT32) of X)
          BIT22 _ (fetch (REC32 BIT27) of X)
          BIT23 _ (fetch (REC32 BIT3) of X)
          BIT24 _ (fetch (REC32 BIT9) of X)
          BIT25 _ (fetch (REC32 BIT19) of X)
          BIT26 _ (fetch (REC32 BIT13) of X)
          BIT27 _ (fetch (REC32 BIT30) of X)
          BIT28 _ (fetch (REC32 BIT6) of X)
          BIT29 _ (fetch (REC32 BIT22) of X)
          BIT30 _ (fetch (REC32 BIT11) of X)
          BIT31 _ (fetch (REC32 BIT4) of X)
          BIT32 _ (fetch (REC32 BIT25) of X])
```

(DES.PERM.PC1

```
  [LAMBDA (X)                                          (* jwo%: "22-Jun-85 20:51")
    (CONS (create REC32
                  BIT1 _ (fetch (REC64 BIT57) of X)
                  BIT2 _ (fetch (REC64 BIT49) of X)
                  BIT3 _ (fetch (REC64 BIT41) of X)
                  BIT4 _ (fetch (REC64 BIT33) of X)
                  BIT5 _ (fetch (REC64 BIT25) of X)
                  BIT6 _ (fetch (REC64 BIT17) of X)
                  BIT7 _ (fetch (REC64 BIT9) of X)
                  BIT8 _ (fetch (REC64 BIT1) of X)
                  BIT9 _ (fetch (REC64 BIT58) of X)
                  BIT10 _ (fetch (REC64 BIT50) of X)
                  BIT11 _ (fetch (REC64 BIT42) of X)
                  BIT12 _ (fetch (REC64 BIT34) of X)
                  BIT13 _ (fetch (REC64 BIT26) of X)
                  BIT14 _ (fetch (REC64 BIT18) of X)
                  BIT15 _ (fetch (REC64 BIT10) of X)
                  BIT16 _ (fetch (REC64 BIT2) of X)
                  BIT17 _ (fetch (REC64 BIT59) of X)
                  BIT18 _ (fetch (REC64 BIT51) of X)
                  BIT19 _ (fetch (REC64 BIT43) of X)
                  BIT20 _ (fetch (REC64 BIT35) of X)
                  BIT21 _ (fetch (REC64 BIT27) of X)
                  BIT22 _ (fetch (REC64 BIT19) of X)
                  BIT23 _ (fetch (REC64 BIT11) of X)
                  BIT24 _ (fetch (REC64 BIT3) of X)
                  BIT25 _ (fetch (REC64 BIT60) of X)
                  BIT26 _ (fetch (REC64 BIT52) of X)
                  BIT27 _ (fetch (REC64 BIT44) of X)
                  BIT28 _ (fetch (REC64 BIT36) of X))
          (create REC32
                  BIT1 _ (fetch (REC64 BIT63) of X)
                  BIT2 _ (fetch (REC64 BIT55) of X)
                  BIT3 _ (fetch (REC64 BIT47) of X)
                  BIT4 _ (fetch (REC64 BIT39) of X)
                  BIT5 _ (fetch (REC64 BIT31) of X)
                  BIT6 _ (fetch (REC64 BIT23) of X)
                  BIT7 _ (fetch (REC64 BIT15) of X)
                  BIT8 _ (fetch (REC64 BIT7) of X)
                  BIT9 _ (fetch (REC64 BIT62) of X)
                  BIT10 _ (fetch (REC64 BIT54) of X)
                  BIT11 _ (fetch (REC64 BIT46) of X)
                  BIT12 _ (fetch (REC64 BIT38) of X)
                  BIT13 _ (fetch (REC64 BIT30) of X)
                  BIT14 _ (fetch (REC64 BIT22) of X)
                  BIT15 _ (fetch (REC64 BIT14) of X)
                  BIT16 _ (fetch (REC64 BIT6) of X)
                  BIT17 _ (fetch (REC64 BIT61) of X)
                  BIT18 _ (fetch (REC64 BIT53) of X)
                  BIT19 _ (fetch (REC64 BIT45) of X)
                  BIT20 _ (fetch (REC64 BIT37) of X)
                  BIT21 _ (fetch (REC64 BIT29) of X)
                  BIT22 _ (fetch (REC64 BIT21) of X)
                  BIT23 _ (fetch (REC64 BIT13) of X)
                  BIT24 _ (fetch (REC64 BIT5) of X)
                  BIT25 _ (fetch (REC64 BIT28) of X)
                  BIT26 _ (fetch (REC64 BIT20) of X)
                  BIT27 _ (fetch (REC64 BIT12) of X)
                  BIT28 _ (fetch (REC64 BIT4) of X])
```

(DES.PERM.PC2

```
  [LAMBDA (C D)                                        (* jwo%: "25-Jun-85 14:34")
    (create REC48
          BIT1 _ (fetch (REC32 BIT14) of C)
          BIT2 _ (fetch (REC32 BIT17) of C)
          BIT3 _ (fetch (REC32 BIT11) of C)
```

```
          BIT4 _ (fetch (REC32 BIT24) of C)
          BIT5 _ (fetch (REC32 BIT1) of C)
          BIT6 _ (fetch (REC32 BIT5) of C)
          BIT7 _ (fetch (REC32 BIT3) of C)
          BIT8 _ (fetch (REC32 BIT28) of C)
          BIT9 _ (fetch (REC32 BIT15) of C)
          BIT10 _ (fetch (REC32 BIT6) of C)
          BIT11 _ (fetch (REC32 BIT21) of C)
          BIT12 _ (fetch (REC32 BIT10) of C)
          BIT13 _ (fetch (REC32 BIT23) of C)
          BIT14 _ (fetch (REC32 BIT19) of C)
          BIT15 _ (fetch (REC32 BIT12) of C)
          BIT16 _ (fetch (REC32 BIT4) of C)
          BIT17 _ (fetch (REC32 BIT26) of C)
          BIT18 _ (fetch (REC32 BIT8) of C)
          BIT19 _ (fetch (REC32 BIT16) of C)
          BIT20 _ (fetch (REC32 BIT7) of C)
          BIT21 _ (fetch (REC32 BIT27) of C)
          BIT22 _ (fetch (REC32 BIT20) of C)
          BIT23 _ (fetch (REC32 BIT13) of C)
          BIT24 _ (fetch (REC32 BIT2) of C)
          BIT25 _ (fetch (REC32 BIT13) of D)
          BIT26 _ (fetch (REC32 BIT24) of D)
          BIT27 _ (fetch (REC32 BIT3) of D)
          BIT28 _ (fetch (REC32 BIT9) of D)
          BIT29 _ (fetch (REC32 BIT19) of D)
          BIT30 _ (fetch (REC32 BIT27) of D)
          BIT31 _ (fetch (REC32 BIT2) of D)
          BIT32 _ (fetch (REC32 BIT12) of D)
          BIT33 _ (fetch (REC32 BIT23) of D)
          BIT34 _ (fetch (REC32 BIT17) of D)
          BIT35 _ (fetch (REC32 BIT5) of D)
          BIT36 _ (fetch (REC32 BIT20) of D)
          BIT37 _ (fetch (REC32 BIT16) of D)
          BIT38 _ (fetch (REC32 BIT21) of D)
          BIT39 _ (fetch (REC32 BIT11) of D)
          BIT40 _ (fetch (REC32 BIT28) of D)
          BIT41 _ (fetch (REC32 BIT6) of D)
          BIT42 _ (fetch (REC32 BIT25) of D)
          BIT43 _ (fetch (REC32 BIT18) of D)
          BIT44 _ (fetch (REC32 BIT14) of D)
          BIT45 _ (fetch (REC32 BIT22) of D)
          BIT46 _ (fetch (REC32 BIT8) of D)
          BIT47 _ (fetch (REC32 BIT1) of D)
          BIT48 _ (fetch (REC32 BIT4) of D])
```

(DES.REC32.LS28

```
  [LAMBDA (X)                                                    ; Edited 20-May-87 17:20 by bvm:
    ;; X points at a two-word block.  Rotate bits 0-27 left one bit (low 4 bits unchanged).
    (create DES.REC32.LS28.OUT
          B1.15 _ (fetch (DES.REC32.LS28.IN B2.16) of X)
          B16 _ (fetch (DES.REC32.LS28.IN B17) of X)
          B17.27 _ (fetch (DES.REC32.LS28.IN B18.28) of X)
          B28 _ (fetch (DES.REC32.LS28.IN B1) of X])
```

(DES.SMAP

```
  [LAMBDA (X)                                                    (* jwo%: "21-Jun-85 23:06")
    (create DES.REC32.4
          NIB1 _ (ELT DES.SBOX.1 (fetch (DES.REC48.6 S1) of X))
          NIB2 _ (ELT DES.SBOX.2 (fetch (DES.REC48.6 S2) of X))
          NIB3 _ (ELT DES.SBOX.3 (IPLUS (ITIMES 4 (fetch (DES.REC48.6 S3.1) of X))
                                        (fetch (DES.REC48.6 S3.2) of X)))
          NIB4 _ (ELT DES.SBOX.4 (fetch (DES.REC48.6 S4) of X))
          NIB5 _ (ELT DES.SBOX.5 (fetch (DES.REC48.6 S5) of X))
          NIB6 _ (ELT DES.SBOX.6 (IPLUS (ITIMES 16 (fetch (DES.REC48.6 S6.1) of X))
                                        (fetch (DES.REC48.6 S6.2) of X)))
          NIB7 _ (ELT DES.SBOX.7 (fetch (DES.REC48.6 S7) of X))
          NIB8 _ (ELT DES.SBOX.8 (fetch (DES.REC48.6 S8) of X])
```

(REC32.XOR

```
  [LAMBDA (X Y)                                                  (* jwo%: "24-Jun-85 17:42")
    (create REC32.W
          WORD1 _ (LOGXOR (fetch (REC32.W WORD1) of X)
                          (fetch (REC32.W WORD1) of Y))
          WORD2 _ (LOGXOR (fetch (REC32.W WORD2) of X)
                          (fetch (REC32.W WORD2) of Y])
```

(REC48.XOR

```
  [LAMBDA (X Y)                                                  (* jwo%: "24-Jun-85 17:50")
    (create REC48.W
          WORD1 _ (LOGXOR (fetch (REC48.W WORD1) of X)
                          (fetch (REC48.W WORD1) of Y))
```

```
          WORD2 _ (LOGXOR (fetch (REC48.W WORD2) of X)
                          (fetch (REC48.W WORD2) of Y))
          WORD3 _ (LOGXOR (fetch (REC48.W WORD3) of X)
                          (fetch (REC48.W WORD3) of Y])
```

(REC64.XOR

```
  [LAMBDA (X Y)                                                    (* jwo%: " 3-Jul-85 17:27")
    (create REC64.W
          WORD1 _ (LOGXOR (fetch (REC64.W WORD1) of X)
                          (fetch (REC64.W WORD1) of Y))
          WORD2 _ (LOGXOR (fetch (REC64.W WORD2) of X)
                          (fetch (REC64.W WORD2) of Y))
          WORD3 _ (LOGXOR (fetch (REC64.W WORD3) of X)
                          (fetch (REC64.W WORD3) of Y))
          WORD4 _ (LOGXOR (fetch (REC64.W WORD4) of X)
                          (fetch (REC64.W WORD4) of Y])
```

(REC64.XOR.CHK

```
  [LAMBDA (L N)                                                    (* jwo%: " 9-Aug-85 00:58")
    (LET ((BLK (create REC64.W
                       WORD1 _ 0
                       WORD2 _ 0
                       WORD3 _ 0
                       WORD4 _ 0)))
         [for I from 1 to N do (PROGN [replace (REC64.W WORD1) of BLK with (LOGXOR (fetch (REC64.W WORD1)
                                                                                     of BLK)
                                                                                  (fetch (REC64.W WORD1)
                                                                                     of (CAR (NTH L I]
                                      [replace (REC64.W WORD2) of BLK with (LOGXOR (fetch (REC64.W WORD2)
                                                                                     of BLK)
                                                                                  (fetch (REC64.W WORD2)
                                                                                     of (CAR (NTH L I]
                                      [replace (REC64.W WORD3) of BLK with (LOGXOR (fetch (REC64.W WORD3)
                                                                                     of BLK)
                                                                                  (fetch (REC64.W WORD3)
                                                                                     of (CAR (NTH L I]
                                      (replace (REC64.W WORD4) of BLK with (LOGXOR (fetch (REC64.W WORD4)
                                                                                     of BLK)
                                                                                  (fetch (REC64.W WORD4)
                                                                                     of (CAR (NTH L I]
         BLK])

)

(RPAQ DES.PARITY.TABLE
      (READARRAY-FROM-LIST 128 'BIT 0
            '(1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1
                0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1
                0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 NIL)))

(RPAQ DES.SBOX.1
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(14 0 4 15 13 7 1 4 2 14 15 2 11 13 8 1 3 10 10 6 6 12 12 11 5 9 9 5 0 3 7 8 4 15 1 12 14 8 8 2 13
                 4 6 9 2 1 11 7 15 5 12 11 9 3 7 14 3 10 10 0 5 6 0 13 NIL)))

(RPAQ DES.SBOX.2
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(15 3 1 13 8 4 14 7 6 15 11 2 3 8 4 14 9 12 7 0 2 1 13 10 12 6 0 9 5 11 10 5 0 13 14 8 7 10 11 1
                 10 3 4 15 13 4 1 2 5 11 8 6 12 7 6 12 9 0 3 5 2 14 15 9 NIL)))

(RPAQ DES.SBOX.3
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(10 13 0 7 9 0 14 9 6 3 3 4 15 6 5 10 1 2 13 8 12 5 7 14 11 12 4 11 2 15 8 1 13 1 6 10 4 13 9 0 8
                 6 15 9 3 8 0 7 11 4 1 15 2 14 12 3 5 11 10 5 14 2 7 12 NIL)))

(RPAQ DES.SBOX.4
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(7 13 13 8 14 11 3 5 0 6 6 15 9 0 10 3 1 4 2 7 8 2 5 12 11 1 12 10 4 14 15 9 10 3 6 15 9 0 0 6 12
                10 11 1 7 13 13 8 15 9 1 4 3 5 14 11 5 12 2 7 8 2 4 14 NIL)))

(RPAQ DES.SBOX.5
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(2 14 12 11 4 2 1 12 7 4 10 7 11 13 6 1 8 5 5 0 3 15 15 10 13 3 0 9 14 8 9 6 4 11 2 8 1 12 11 7 10
                1 13 14 7 2 8 13 15 6 9 15 12 0 5 9 6 10 3 4 0 5 14 3 NIL)))

(RPAQ DES.SBOX.6
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(12 10 1 15 10 4 15 2 9 7 2 12 6 9 8 5 0 6 13 1 3 13 4 14 14 0 7 11 5 3 11 8 9 4 14 3 15 2 5 12 2
                 9 8 5 12 15 3 10 7 11 0 14 4 1 10 7 1 6 13 0 11 8 6 13 NIL)))

(RPAQ DES.SBOX.7
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(4 13 11 0 2 11 14 7 15 4 0 9 8 1 13 10 3 14 12 3 9 5 7 12 5 2 10 15 6 8 1 6 1 6 4 11 11 13 13 8
                12 1 3 4 7 10 14 7 10 9 15 5 6 0 8 15 0 14 5 2 9 3 2 12 NIL)))
```

```
(RPAQ DES.SBOX.8
      (READARRAY-FROM-LIST 64 'BYTE 0
            '(13 1 2 15 8 13 4 8 6 10 15 3 11 7 1 4 10 12 9 5 3 6 14 11 5 0 0 14 12 9 7 2 7 2 11 1 4 14 1 7 9 4
                 12 10 14 8 2 13 0 15 6 12 10 9 13 0 15 3 3 5 5 6 8 11 NIL)))

(RPAQ DES.SHIFTS (READARRAY-FROM-LIST 16 'BYTE 1
                       '(1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1 NIL)))

(RPAQQ DESKEYSLST NIL)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD DES.REC.E1 ((B1.5 BITS 5)
                         (G6.7 BITS 2)
                         (B8.11 BITS 4)
                         (B12.13 BITS 2)
                         (G14.16 BITS 3)
                         (G17.19 BITS 3)
                         (B20.21 BITS 2)
                         (B22.25 BITS 4)
                         (G26.27 BITS 2)
                         (B28.32 BITS 5))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD DES.REC.E2 ((G1.3 BITS 3)
                         (B4.9 BITS 6)
                         (G10.11 BITS 2)
                         (B12.16 BITS 5)
                         (B17.21 BITS 5)
                         (G22.23 BITS 2)
                         (B24.29 BITS 6)
                         (G30.32 BITS 3))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD DES.REC32.4 ((NIB1 BITS 4)
                          (NIB2 BITS 4)
                          (NIB3 BITS 4)
                          (NIB4 BITS 4)
                          (NIB5 BITS 4)
                          (NIB6 BITS 4)
                          (NIB7 BITS 4)
                          (NIB8 BITS 4))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD DES.REC32.LS28.IN ((B1 BITS 1)
                                (B2.16 BITS 15)
                                (B17 BITS 1)
                                (B18.28 BITS 11))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD DES.REC32.LS28.OUT ((B1.15 BITS 15)
                                 (B16 BITS 1)
                                 (B17.27 BITS 11)
                                 (B28 BITS 1))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD DES.REC48.6 ((S1 BITS 6)
                          (S2 BITS 6)
                          (S3.1 BITS 4)
                          (S3.2 BITS 2)
                          (S4 BITS 6)
                          (S5 BITS 6)
                          (S6.1 BITS 2)
                          (S6.2 BITS 4)
                          (S7 BITS 6)
                          (S8 BITS 6))
       (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD DESBLOCK ((W1 BITS 16)
                       (W2 BITS 16)
                       (W3 BITS 16)
                       (W4 BITS 16))
       (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD DESKEY ((W1 BITS 16)
                     (W2 BITS 16)
                     (W3 BITS 16)
                     (W4 BITS 16))
       (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD DESKEY.P ((B1 BITS 7)
                       (P1 BITS 1)
                       (B2 BITS 7)
                       (P2 BITS 1)
                       (B3 BITS 7)
```

```
                        (P3 BITS 1)
                        (B4 BITS 7)
                        (P4 BITS 1)
                        (B5 BITS 7)
                        (P5 BITS 1)
                        (B6 BITS 7)
                        (P6 BITS 1)
                        (B7 BITS 7)
                        (P7 BITS 1)
                        (B8 BITS 7)
                        (P8 BITS 1))
       (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD REC32 ((BIT1 BITS 1)
                    (BIT2 BITS 1)
                    (BIT3 BITS 1)
                    (BIT4 BITS 1)
                    (BIT5 BITS 1)
                    (BIT6 BITS 1)
                    (BIT7 BITS 1)
                    (BIT8 BITS 1)
                    (BIT9 BITS 1)
                    (BIT10 BITS 1)
                    (BIT11 BITS 1)
                    (BIT12 BITS 1)
                    (BIT13 BITS 1)
                    (BIT14 BITS 1)
                    (BIT15 BITS 1)
                    (BIT16 BITS 1)
                    (BIT17 BITS 1)
                    (BIT18 BITS 1)
                    (BIT19 BITS 1)
                    (BIT20 BITS 1)
                    (BIT21 BITS 1)
                    (BIT22 BITS 1)
                    (BIT23 BITS 1)
                    (BIT24 BITS 1)
                    (BIT25 BITS 1)
                    (BIT26 BITS 1)
                    (BIT27 BITS 1)
                    (BIT28 BITS 1)
                    (BIT29 BITS 1)
                    (BIT30 BITS 1)
                    (BIT31 BITS 1)
                    (BIT32 BITS 1))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD REC32.W ((WORD1 BITS 16)
                      (WORD2 BITS 16))
       (CREATE (\ALLOCBLOCK 1)))

(BLOCKRECORD REC48 ((BIT1 BITS 1)
                    (BIT2 BITS 1)
                    (BIT3 BITS 1)
                    (BIT4 BITS 1)
                    (BIT5 BITS 1)
                    (BIT6 BITS 1)
                    (BIT7 BITS 1)
                    (BIT8 BITS 1)
                    (BIT9 BITS 1)
                    (BIT10 BITS 1)
                    (BIT11 BITS 1)
                    (BIT12 BITS 1)
                    (BIT13 BITS 1)
                    (BIT14 BITS 1)
                    (BIT15 BITS 1)
                    (BIT16 BITS 1)
                    (BIT17 BITS 1)
                    (BIT18 BITS 1)
                    (BIT19 BITS 1)
                    (BIT20 BITS 1)
                    (BIT21 BITS 1)
                    (BIT22 BITS 1)
                    (BIT23 BITS 1)
                    (BIT24 BITS 1)
                    (BIT25 BITS 1)
                    (BIT26 BITS 1)
                    (BIT27 BITS 1)
                    (BIT28 BITS 1)
                    (BIT29 BITS 1)
                    (BIT30 BITS 1)
                    (BIT31 BITS 1)
                    (BIT32 BITS 1)
                    (BIT33 BITS 1)
                    (BIT34 BITS 1)
                    (BIT35 BITS 1)
                    (BIT36 BITS 1)
```

```
                    (BIT37 BITS 1)
                    (BIT38 BITS 1)
                    (BIT39 BITS 1)
                    (BIT40 BITS 1)
                    (BIT41 BITS 1)
                    (BIT42 BITS 1)
                    (BIT43 BITS 1)
                    (BIT44 BITS 1)
                    (BIT45 BITS 1)
                    (BIT46 BITS 1)
                    (BIT47 BITS 1)
                    (BIT48 BITS 1))
      (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD REC48.W ((WORD1 BITS 16)
                      (WORD2 BITS 16)
                      (WORD3 BITS 16))
      (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD REC64 ((BIT1 BITS 1)
                    (BIT2 BITS 1)
                    (BIT3 BITS 1)
                    (BIT4 BITS 1)
                    (BIT5 BITS 1)
                    (BIT6 BITS 1)
                    (BIT7 BITS 1)
                    (BIT8 BITS 1)
                    (BIT9 BITS 1)
                    (BIT10 BITS 1)
                    (BIT11 BITS 1)
                    (BIT12 BITS 1)
                    (BIT13 BITS 1)
                    (BIT14 BITS 1)
                    (BIT15 BITS 1)
                    (BIT16 BITS 1)
                    (BIT17 BITS 1)
                    (BIT18 BITS 1)
                    (BIT19 BITS 1)
                    (BIT20 BITS 1)
                    (BIT21 BITS 1)
                    (BIT22 BITS 1)
                    (BIT23 BITS 1)
                    (BIT24 BITS 1)
                    (BIT25 BITS 1)
                    (BIT26 BITS 1)
                    (BIT27 BITS 1)
                    (BIT28 BITS 1)
                    (BIT29 BITS 1)
                    (BIT30 BITS 1)
                    (BIT31 BITS 1)
                    (BIT32 BITS 1)
                    (BIT33 BITS 1)
                    (BIT34 BITS 1)
                    (BIT35 BITS 1)
                    (BIT36 BITS 1)
                    (BIT37 BITS 1)
                    (BIT38 BITS 1)
                    (BIT39 BITS 1)
                    (BIT40 BITS 1)
                    (BIT41 BITS 1)
                    (BIT42 BITS 1)
                    (BIT43 BITS 1)
                    (BIT44 BITS 1)
                    (BIT45 BITS 1)
                    (BIT46 BITS 1)
                    (BIT47 BITS 1)
                    (BIT48 BITS 1)
                    (BIT49 BITS 1)
                    (BIT50 BITS 1)
                    (BIT51 BITS 1)
                    (BIT52 BITS 1)
                    (BIT53 BITS 1)
                    (BIT54 BITS 1)
                    (BIT55 BITS 1)
                    (BIT56 BITS 1)
                    (BIT57 BITS 1)
                    (BIT58 BITS 1)
                    (BIT59 BITS 1)
                    (BIT60 BITS 1)
                    (BIT61 BITS 1)
                    (BIT62 BITS 1)
                    (BIT63 BITS 1)
                    (BIT64 BITS 1))
      (CREATE (\ALLOCBLOCK 2)))

(BLOCKRECORD REC64.W ((WORD1 BITS 16)
                      (WORD2 BITS 16)
```

```
                        (WORD3 BITS 16)
                        (WORD4 BITS 16))
       (CREATE (\ALLOCBLOCK 2)))
)
)

(PUTPROPS DES COPYRIGHT ("Venue & Xerox Corporation" 1985 1987 1990))
```

FUNCTION INDEX

RECORD INDEX

VARIABLE INDEX